

Another counter-example to antithetic sampling

March 13, 2002

The antithetic sampling method (section 4.4.1, page 220) is arguably the easiest approach to variance reduction. However, it is not guaranteed to work, unless a certain monotonicity condition is met. Example 4.6 on page 223 shows that when the function we are integrating is non-monotonic, an increase in variance may result. Section 7.2.2 aims at providing a more practical example of how antithetic sampling may not work properly when the monotonicity condition is not met.

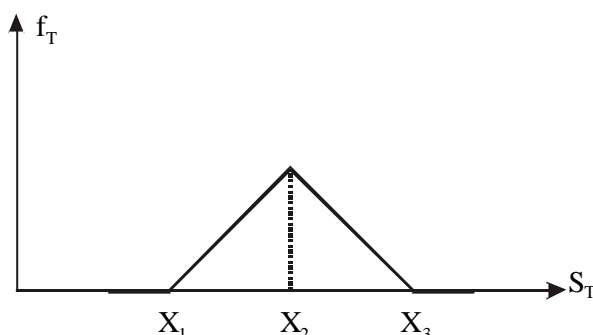


Figure 1: Payoff from a butterfly spread.

This supplement gives a more convincing counter-example based on a popular option trading strategy, the butterfly spread (see, e.g., [1, chapter 8]). The butterfly spread is a trading strategy involving options on the same underlying asset, with the same maturity, but with different strike prices. The payoff from this combination is illustrated in figure 1. It can be obtained by buying one call option with strike price X_1 , one call option with strike price X_3 ($X_1 < X_3$), and by selling two call options with a strike X_2 halfway between the other two. Since the butterfly spread is simply a combination of European calls, an option with that payoff may be directly priced by using Black-Scholes formula.

Since the payoff is clearly non-monotonic, and we know the “correct” price, it is interesting check whether antithetic sampling works in this case.¹ A crude Monte Carlo approach leads to the code in figure 2. Note the use of vectors `In1` and `In2` to collect the indexes corresponding to replications in which the terminal asset price falls in the increasing region of the payoff ($X_1 < S_T < X_2$) or in the decreasing region ($X_2 < S_T < X_3$); outside those regions the payoff is zero. The two vectors are used to avoid for loops. The function `MCButterfly` receives the usual input arguments, plus the three strikes.

The function `MCAVButterfly` of figure 3 is a modification based on antithetic sampling. The vector `Veps` contains the samples from the standard normal distribution, which are changed

¹This supplement should be used in conjunction with the book: P. Brandimarte, *Numerical Methods in Finance: a MATLAB-Based Introduction*, Wiley, 2001. Please refer to the web page (www.polito.it/~brandimarte) for further updates and supplements. Any comment is welcome. My e-mail address is: brandimarte@polito.it.

¹As stressed in the book, we use such examples for purely didactic reasons; clearly, there is little use for Monte Carlo simulation when the correct result may be obtained by an analytical formula.

```

function [P, CI] = MCButterfly(S0,r,T,sigma,NRepl,X1,X2,X3)
nuT = (r-0.5*sigma^2)*T;
siT = sigma*sqrt(T);
Veps = randn(NRepl,1);
Stocks = S0*exp(nuT + siT*Veps);
In1 = find((Stocks > X1) & (Stocks < X2));
In2 = find((Stocks >= X2) & (Stocks < X3));
Payoff = exp(-r*T)*[(Stocks(In1)-X1); (X3-Stocks(In2)); ...
    zeros(NRepl - length(In1) - length(In2),1)];
[P, V, CI] = normfit(Payoff);

```

Figure 2: Crude Monte Carlo code to price a butterfly spread combination.

in sign to obtain the antithetic stock price samples `Stocks2`. Note that in this case we must preserve the order of the samples in order to pair the corresponding payoffs properly.

It is common to choose X_2 close to the current stock price S_0 , as this strategy is based on the bet that the stock price will not move too much. Let us check the results in such a case. Using `blsprice` we may get the theoretical result.

```

>> S0 = 60;
>> X1 = 55;
>> X2 = 60;
>> X3 = 65;
>> T = 5/12;
>> r = 0.1;
>> sigma = 0.4;
>> calls = blsprice(S0, [X1, X2, X3], r, T, sigma);
>> Pth = calls(1) - 2*calls(2) + calls(3)
Pth =
    0.6124

```

Next, we may compare the two Monte Carlo methods (as usual, we use half the replications with antithetic sampling to get a fair comparison, since in this case the parameter `NRepl` refers to the number of antithetic *pairs*):

```

>> randn('seed',0)
>> [P, CI] = MCButterfly(S0,r,T,sigma,10000,X1,X2,X3);
>> P
P =
    0.6145
>> CI(2) - CI(1)
ans =
    0.0156
>> [P, CI] = MCAVButterfly(S0,r,T,sigma,50000,X1,X2,X3);
>> P
P =
    0.6121
>> CI(2) - CI(1)

```

```

function [P, CI] = MCAVButterfly(S0,r,T,sigma,NRepl,X1,X2,X3)
nuT = (r-0.5*sigma^2)*T;
siT = sigma*sqrt(T);
Veps = randn(NRepl,1);
Stocks1 = S0*exp(nuT + siT*Veps);
Stocks2 = S0*exp(nuT - siT*Veps);
Payoff1 = zeros(NRepl,1);
Payoff2 = zeros(NRepl,1);
In = find((Stocks1 > X1) & (Stocks1 < X2));
Payoff1(In) = (Stocks1(In) - X1);
In = find((Stocks1 >= X2) & (Stocks1 < X3));
Payoff1(In) = (X3 - Stocks1(In));
In = find((Stocks2 > X1) & (Stocks2 < X2));
Payoff2(In) = (Stocks2(In) - X1);
In = find((Stocks2 >= X2) & (Stocks2 < X3));
Payoff2(In) = (X3 - Stocks2(In));
Payoff = 0.5 * exp(-r*T) * (Payoff1 + Payoff2);
[P, V, CI] = normfit(Payoff);

```

Figure 3: Using antithetic sampling to price a butterfly spread combination.

```

ans =
    0.0216

```

Apparently, the result obtained with antithetic sampling is closer to the correct result, but in practice you would consider the confidence interval, which is larger with antithetic sampling. This does not mean that you will always have an increase in variance, as this depends on the input data (try changing the strikes to see this). Anyway, since one run does not tell us much, a better comparison may be carried out by checking the mean square error with respect to the exact result:

```

>> randn('seed',0)
>> for i=1:100, V1(i) = MCButterfly(S0,r,T,sigma,100000,X1,X2,X3);, end
>> for i=1:100, V2(i) = MCAVButterfly(S0,r,T,sigma,50000,X1,X2,X3);, end
>> mean((V1 - Pth).^2)
ans =
    1.5550e-005
>> mean((V2 - Pth).^2)
ans =
    3.8167e-005

```

Indeed, we see the mean square error is rather small, yet it is doubled by antithetic sampling. The reader is encouraged to try a control variate approach to this problem.

References

- [1] J.C. Hull. *Options, Futures, and Other Derivatives (4th ed.)*. Prentice Hall, Upper Saddle River, NJ, 2000.