```
% DOPutMCAV.m %use control variates to price barrier
options
function [P,CI,NCrossed] =
DOPutMCAV(S0,X,r,T,sigma,Sb,NSteps,NRepl)

% Generate asset paths
Payoff = zeros(NRepl,1);
Payoff1 = zeros(NRepl,1);
Payoff2 = zeros(NRepl,1);
NCrossed = 0;

for i=1:NRepl
   Path =
(AssetPaths1(S0,r,sigma,T,NSteps,1)+AssetPaths1(S0,r,-
sigma,T,NSteps,1))/2;
   crossed = any(Path <= Sb);
   if crossed == 0
      Payoff(i) = max(0, X - Path(NSteps+1));
   else
      Payoff(i) = 0;
      NCrossed = NCrossed + 1;
   end
end
[P,aux,CI] = normfit( exp(-r*T) * Payoff);
```

```
>> CompDOPutMCAV

DOPutMCAV =

    1.9195


CI =

    1.8942
    1.9448


NCrossed =

     1

>>
```

```
%DoPutHalton.m
%pricing down and out Put using HaltonPaths
function Price =
DoPutHalton(S0,X,r,T,sigma,NPoints,Base1,Base2,Sb)
nuT = (r - 0.5*sigma^2)*T;
siT = sigma * sqrt(T);

% Use Box Muller to generate standard normals
H1 = GetHalton(ceil(NPoints/2),Base1);
H2 = GetHalton(ceil(NPoints/2),Base2);

VLog = sqrt(-2*log(H1));
Norm1 = VLog .* cos(2*pi*H2);
Norm2 = VLog .* sin(2*pi*H2);
Norm = [Norm1 ; Norm2];
%generate asset paths
Payoff = zeros(NPoints,1);
NCrossed = 0;
Path = S0*exp(nuT+siT*Norm);

for i=1:NPoints

   crossed = any(Path <= Sb);
   if crossed == 0
      Payoff(i) = max(0, X - Path(NPoints));
   else
      Payoff(i) = 0;
      NCrossed = NCrossed + 1;
   end
end
[Price,aux,CI] = normfit( exp(-r*T) * Payoff);
Price
aux
CI
```

```
>> ComputeDoPutHalton

Price =

    6.9588


aux =
```

```
    9.3622e-16


CI =

    6.9588
    6.9588

>>
```