```
gail.InitializeWorkspaceDisplay %initialize the workspace and the display
 parameters
inp.timeDim.timeVector = 1/52:1/52:6/52; %weekly monitoring for three months
inp.assetParam.initPrice = 30; %initial stock price
inp.assetParam.interest = 0.01; %risk-free interest rate
inp.assetParam.volatility = 0.4; %volatility
inp.payoffParam.strike = 30; %strike price
inp.payoffParam.putCallType = {'call'}; %looking at a put option
inp.priceParam.absTol = 0.1; %absolute tolerance of a dime
inp.priceParam.relTol = 0; %zero relative tolerance
%inp.priceParam.cubMethod = 'Sobol'; %Sobol sampling
%inp.payoffParam.optType = {'amean'}; %looking at an arithmetic mean option
EuroCall = optPrice(inp); %construct an optPrice object


%default for European option
disp(['Baseline:'])
disp(['The price of this European call option is $'
 num2str(EuroCall.exactPrice)])
disp('1.')
disp('a)')
ArithMeanCall = optPrice(EuroCall); %make a copy
ArithMeanCall.payoffParam.optType = {'amean'}; %change from European to Asian
 arithemetic mean
```

```
Baseline:
The price of this European call option is $1.6413
1.
a)
```

Next we generate the price using the `genOptPrice` method of the `optPrice` object.

```
[ArithMeanCallPrice,out] = genOptPrice(ArithMeanCall); %uses meanMC_g to
 compute the price
disp(['The price of this Asian arithemetic mean call option is $'
 num2str(ArithMeanCallPrice) ...
   ' +/- $' num2str(max(ArithMeanCall.priceParam.absTol, ...
   ArithMeanCall.priceParam.relTol*ArithMeanCallPrice)) ])
disp(['   and it took ' num2str(out.nPaths) ' paths and ' ...
   num2str(out.time) ' seconds'])

disp('b)')

AsiaEuro = optPayoff(ArithMeanCall);
AsiaEuro.payoffParam = ...
   struct('optType',{{'amean','euro'}}, ... %note two kinds of option payoffs
   'putCallType', {{'call','call'}}); %this needs to have the same dimension
```

```
The price of this Asian arithemetic mean call option is $1.0691 +/- $0.1
   and it took 24112 paths and 0.0065343 seconds
b)
```

The price of the Asian arithmetic mean call option is smaller than the price of the European call option.

```
[AsiaEuroPrice, AEout] = meanMC_g(@(n) YoptPrice_CV(AsiaEuro,n), ...
   inp.priceParam.absTol, inp.priceParam.relTol);
disp(['The price of the Asian call option with European call as control
 variate is $' ...
   num2str(AsiaEuroPrice,'%5.2f')])
disp(['   and this took ' num2str(AEout.ntot) ' paths and ' ...
   num2str(AEout.time) ' seconds'])
disp(['   which is ' num2str(AEout.ntot/out.nPaths) ...
   ' of the paths and ' num2str(AEout.time/out.time) ' of the time'])
disp('of the time without control variates')

disp('c)')
gail.InitializeWorkspaceDisplay
inp.timeDim.timeVector = 1/52:1/52:6/52; %weekly monitoring for three months
inp.assetParam.initPrice = 30; %initial stock price
inp.assetParam.interest = 0.01; %risk-free interest rate
inp.assetParam.volatility = 0.4; %volatility
inp.payoffParam.strike = 30; %strike price
inp.priceParam.absTol = 0.1; %absolute tolerance of a dime
inp.priceParam.relTol = 0; %zero relative tolerance
inp.payoffParam.optType = {'amean'}; %looking at an arithmetic mean option
inp.payoffParam.putCallType = {'call'}; %looking at a call option
```

```
The price of the Asian call option with European call as control variate is
 $1.06
   and this took 14803 paths and 0.0077818 seconds
   which is 0.61393 of the paths and 1.1909 of the time
of the time without control variates
c)
```

The Asian arithmetic mean put without antithetic variates

Next we create an Asian arithmetic mean put `optPrice` object and use Monte Carlo to compute the price.

```
AMeanCall = optPrice(inp); %construct an optPrice object
[AMeanCallPrice,Aout] = genOptPrice(AMeanCall);
disp(['The price of the Asian arithmetic mean call option is $' ...
   num2str(AMeanCallPrice,'%5.2f')])
disp(['   and this took ' num2str(Aout.nPaths) ' paths and ' ...
   num2str(Aout.time) ' seconds'])
```

```
The price of the Asian arithmetic mean call option is $1.06
   and this took 24240 paths and 0.0064709 seconds
```

The Asian arithmetic mean put with antithetic variates

Since this functionality is not available in GAIL yet, we need to create our own function that generates the two sets of payoffs from the Brownian motion and its additive inverse, and then takes the average. We have written such a function:

```
function YAnti=YoptPrice_Anti(optPayoffObj,n)
% YOPTPRICE_Anti creates payoffs from antithetic Brownian motion sampling
% for an Asian arithmetic mean put.

bmObj = brownianMotion(optPayoffObj); %make a Brownian motion object
BMPaths =  genPaths(bmObj,n); %ordinary Brownian motion paths
temp1 = (optPayoffObj.assetParam.interest -
 (optPayoffObj.assetParam.volatility.^2)/2) ...
   .* optPayoffObj.timeDim.timeVector; %(r-sigma^2/2)*t
stockPrice1 = optPayoffObj.assetParam.initPrice*exp(bsxfun(@plus, temp1, ...
   optPayoffObj.assetParam.volatility.*BMPaths)); %with original Brownian
 paths
stockPrice2 = optPayoffObj.assetParam.initPrice*exp(bsxfun(@minus, temp1, ...
   optPayoffObj.assetParam.volatility.*BMPaths)); %with minus Brownian paths

YAnti = (max(optPayoffObj.payoffParam.strike - mean(stockPrice1,2),0) ...
   + max(optPayoffObj.payoffParam.strike - mean(stockPrice2,2),0)) ...
   .* (0.5*exp(-optPayoffObj.assetParam.interest *
 optPayoffObj.timeDim.endTime));
   % the averge of the Asian arithmetic mean put payoffs using the two
   % stock price paths
```

In the future, this function should not be needed because GAIL will contain this functionality.

Now we call `meanMC_g`:

```
[AMeanPriceAnti, AAntiout] = meanMC_g(@(n) YoptPrice_Anti(AMeanCall,n), ...
   inp.priceParam.absTol, inp.priceParam.relTol);
disp(['The price of the Asian arithmetic mean call option is $' ...
   num2str(AMeanPriceAnti,'%5.2f')])
disp(['   and this took ' num2str(AAntiout.ntot) ' paths and ' ...
   num2str(AAntiout.time) ' seconds'])
disp(['   which is ' num2str(AAntiout.ntot/Aout.nPaths) ...
   ' of the paths and ' num2str(AAntiout.time/Aout.time) ' of the time'])
disp('      without antithetic variates')
```

```
The price of the Asian arithmetic mean call option is $1.05
   and this took 14094 paths and 0.017042 seconds
   which is 0.58144 of the paths and 2.6337 of the time
      without antithetic variates
```

Published with MATLAB® R2022a