MCCI

SOLVE THE IMPOSSIBLE

Monte Carlo Method and its applications

Xiaoshu Wang

What is Monte Carlo Method?

- Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle. (Wikipedia)
- Or more frankly speaking, Monte Carlo method can simulate the stochastic properties of a system by constructing a probabilistic model similar to the performance of the system and conducting lots and lots of randomized trials. It is a simulation process.

History

- Before the Monte Carlo method was developed, simulations tested a previously understood deterministic problem, and statistical sampling was used to estimate uncertainties in the simulations. Monte Carlo simulations invert this approach, solving deterministic problems using probabilistic metaheuristics.
- An early variant of the Monte Carlo method was devised to solve the Buffon's needle problem, in which π can be estimated by dropping needles on a floor made of parallel equidistant strips.
- In the late 1940s, Stanislaw Ulam invented the modern version of the Markov Chain Monte Carlo method while he was working on nuclear weapons projects.
- The theory of more sophisticated mean-field type particle Monte Carlo methods had certainly started by the mid-1960s, Quantum Monte Carlo, and more specifically diffusion Monte Carlo methods can also be interpreted during that period.

MCCI
SOLVE THE IMPOSSIBLE

History

- The use of Sequential Monte Carlo in advanced signal processing and Bayesian inference is more recent.
- The mathematical foundations and the first rigorous analysis of these particle algorithms were written by Pierre Del Moral in 1996.
- Now it is very widely used in physics, engineering, biology, applied statistics, artificial intelligence, design, finance, and even in climate change, rescue, and law.

Why is it so powerful?

- The traditional empirical methods can not approximate the real process, it is difficult to get satisfactory results, while the Monte Carlo method can realistically simulate the actual process, so the solution of the problem is very consistent with the actual and can get very satisfactory results. It is a computational method based on probability and statistical theory and is a method that uses random numbers (or more commonly pseudo-random numbers) to solve many computational problems. The power of computers eliminates the need for complicated mathematical interpretations and calculations, making them understandable and approachable to most people. Also, it can save a lot of time, improve the efficiency.

MCCI
SOLVE THE IMPOSSIBLE

A classic example

- First, consider a quadrant inscribed in a unit square (with each length of 1).
- Then uniformly scatter a given number of points over the square
- Count the number of points which has a distance from the origin of less than 1.
- The ratio of number of points in the quadrant and total number of points is the estimation of $\pi/4$.
- As we can see from the GIF, when n gets bigger, the estimation is closer to π. (Wikipedia)

Common approaches

- 1. Define the domains of inputs
- 2. Construct or describe a probabilistic process for nonstochastic or stochastic process
- 3. Randomized sample from the know probabilistic process
- 4. Perform computation on the inputs to get outputs
- 5. Aggregate the outputs and make inference

MCCI
SOLVE THE IMPOSSIBLE

Monte Carlo Simulation

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1					t	τ	z	S(t)	d_1	d_2	C_{BS}	Δ(t)	Cash Flow	B(t)	C_{rep}
2		S(0)	$ 100.00		0.000	1.000		$100.00	0.300	0.100	$9.93	0.6179 -		-$51.87	$9.93
3		K	$ 100.00		0.053	0.947	-0.0126	$99.82	0.283	0.088	$9.50	0.6113	$0.66	-$51.31	$9.70
4		r	4.00%		0.105	0.895	0.1769	$104.08	0.495	0.306	$11.95	0.6897	-$8.16	-$59.58	$12.20
5		σ	20.00%		0.158	0.842	-0.1488	$100.68	0.312	0.129	$9.38	0.6225	$6.76	-$52.95	$9.73
6		T	1		0.211	0.789	-0.1677	$96.97	0.093	-0.084	$6.90	0.5371	$8.28	-$44.78	$7.30
7		Δt	0.053		0.263	0.737	0.1199	$99.79	0.245	0.073	$8.17	0.5968	-$5.96	-$50.83	$8.72
8		small_t	0.00001		0.316	0.684	0.2472	$105.74	0.586	0.420	$11.74	0.7210	-$13.13	-$64.07	$12.17
9					0.368	0.632	0.3542	$114.85	1.109	0.950	$18.66	0.8664	-$16.70	-$80.90	$18.60
10		Error	$0.63		0.421	0.579	-0.2006	$109.78	0.841	0.689	$14.08	0.8000	$7.29	-$73.78	$14.04
11					0.474	0.526	0.0778	$111.88	0.992	0.846	$15.44	0.8393	-$4.40	-$78.34	$15.56
12					0.526	0.474	0.1067	$114.79	1.208	1.071	$17.59	0.8866	-$5.43	-$83.93	$17.83
13					0.579	0.421	-0.3235	$106.66	0.692	0.562	$10.48	0.7554	$13.99	-$70.13	$10.45
14					0.632	0.368	0.1126	$109.58	0.936	0.814	$12.37	0.8253	-$7.65	-$77.93	$12.51
15					0.684	0.316	0.1730	$114.15	1.346	1.234	$15.96	0.9109	-$9.77	-$87.87	$16.12
16					0.737	0.263	-0.1074	$111.48	1.213	1.111	$13.18	0.8875	$2.61	-$85.45	$13.50
17					0.789	0.211	0.0117	$111.90	1.363	1.271	$13.17	0.9136	-$2.92	-$88.54	$13.69
18					0.842	0.158	-0.0307	$111.23	1.459	1.379	$12.16	0.9277	-$1.57	-$90.30	$12.89
19					0.895	0.105	0.2157	$117.02	2.520	2.455	$17.45	0.9941	-$7.77	-$98.27	$18.07
20					0.947	0.053	-0.1480	$113.22	2.775	2.729	$13.43	0.9972	-$0.35	-$98.83	$14.08
21				T----->	1.000	0.000	0.2232	$119.32	279.247	279.247	$19.32	1.0000	-$0.33	-$99.37	$19.95
22															
23															

Figure 1: simulation of one path of a call option and stock process

SOLVE THE IMPOSSIBLE

Monte Carlo Simulation

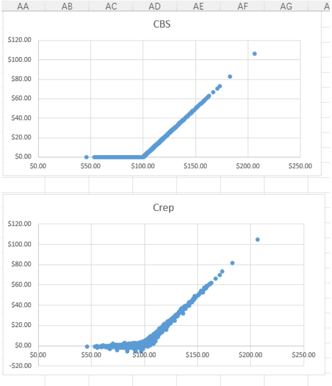

Figure 2: plots of numerical solution and simulation

- Here we have generated 1000 paths and plot the theoretical earnings and actual earnings. We can see that the earnings of our replication portfolio highly coincides with the result from theoretical model.

SOLVE THE IMPOSSIBLE

Monte Carlo Simulation

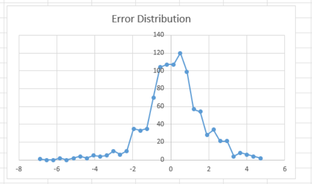

Figure 3: plot of error distribution

- Here is a plot of the error distribution, vertical axis denotes the frequency and horizontal axis denotes the error between theoretical model and simulation. As we can see that, the error distribution approximately follows a normal distribution with mean 0, so we can trade more often to reduce the variance of error distribution.

Monte Carlo Simulation

```
```{r model3}
nYears <- 10
nScenarios <- 10000
nPeriods <- 12
initialBalance <- 100000
capRate <- 1
floorRate <- -1

# Set up arrays to hold results
scenarioBalances <- matrix(NA, nScenarios, nYears * nPeriods)
avgScenarioBalances <- rep(NA, nYears * nPeriods)

# Loop over scenarios and periods
for (i in 1:nScenarios) {
  balance <- initialBalance
  for (j in 1:(nYears * nPeriods)) {
    # Calculate the monthly rate of return
    r <- rnorm(1, mean = 0.0329, sd = 0.1010)

    # Calculate the cap and floor rates for the period
    cap <- ifelse(r > capRate, capRate, r)
    floor <- ifelse(r < floorRate, floorRate, r)

    # Calculate the monthly return
    monthlyReturn <- (1 + cap)^(1/12) - 1
    monthlyFloor <- (1 + floor)^(1/12) - 1

    # Calculate the new balance for the period
    balance <- balance * (1 + monthlyReturn)

    # Apply the floor rate if necessary
    if (monthlyReturn < monthlyFloor) {
      balance <- balance * (1 + monthlyFloor)
    }

    # Store the balance for the period
    scenarioBalances[i, j] <- balance
  }
}
```

Figure 4: another example in R of modeling indexed annuity

SOLVE THE IMPOSSIBLE
```

Monte Carlo Simulation

````
# Calculate the average scenario balance for each period
for (i in 1:(nYears * nPeriods)) {
  avgScenarioBalances[i] <- mean(scenarioBalances[, i])
}
avgrate<-(avgScenarioBalances[i]/initialBalance)^(1/nYears)-1
avgrate
avgScenarioBalances[i]
# Plot the results
plot(1:(nYears * nPeriods), avgScenarioBalances, type = "l", xlab = "Time (months)", ylab = "Balance", main = "Fixed Indexed Annuity with Cap and Floor Rates")
```
````

R Console

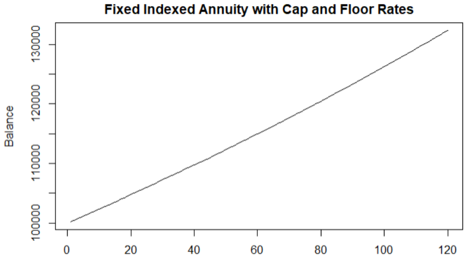

Figure 5: another example in R of modeling indexed annuity

MCCI
SOLVE THE IMPOSSIBLE
````

Advantage

- The error has nothing to do with dimensions
- Can easily handle continuous problem
- Can simulate complicated situations that cannot be modeled using other methods

MCCI
SOLVE THE IMPOSSIBLE

Disadvantage

- Monte Carlo simulation may require lots of iterations and the algorithm could be computationally intensive and time-consuming
- Highly relied on the goodness of assumptions of inputs, the model could be wrong at the beginning
- May not reach convergence

MCCI
SOLVE THE IMPOSSIBLE

References

- https://en.wikipedia.org/wiki/Monte_Carlo_method
- https://en.wikipedia.org/wiki/Black–Scholes_model

Q&A

SOLVE THE IMPOSSIBLE