American Options & Least-Squares Approach

D. Marazzina

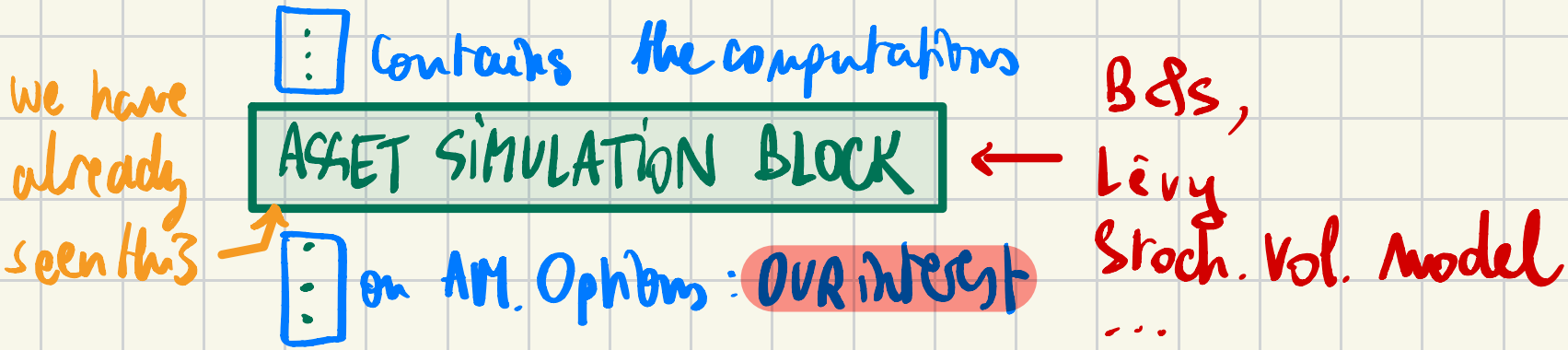
Finanza Computazionale

28/10/2024

American Option & MC

Goal: to study how we can deal with American Options in a Monte Carlo framework.

In the code, we will have:



1) Not really ~~American~~, but Bermudan

option: "American", but in discrete

monitoring.

Bermudan Option

PUT

$$\text{price} = \mathbb{E}^{\mathbb{Q}}\left[\max_{\tau \in \mathcal{T}} e^{-r(\tau - 0)} (K - S_\tau)^+ \,\middle|\, S_0 \right]$$

where $\mathcal{T} = \{0, \Delta t, 2\Delta t, \ldots, T\}$, $\Delta t = \frac{T}{M}$.

"Stopping time problem" : we have to look @ each time whether we have to early-exercise our option.

So the question is: how to couple the MC tools with the early exercise ?

IDEA - I build $[S_{ij}]$ with $i = 1, \dots, N_{sim}$

and $j = 0, \dots, M$ s.t $S_{ij} = S_i(j\Delta t)$.

INCORRECT: → gives us an UPPER BOUND.

- I compute :

$$price^* = \frac{1}{N_{sim}} \sum_{i=1}^{N_{sim}} \max_{j \in \{0,\dots,M\}} e^{-r j \Delta t} (K - S_{ij})^+$$

Why is it INCORRECT? Because we choose the max once we know everything (i.e from 0 to T). We are exploiting knowledge of the future.

↳ Knowing the future is for sure the best strategy. That's why it's an upper bound : price Bermudan ≤ price*.

- Another strategy could be:

(*)
$$\text{price}^1 = \frac{1}{N_{SIM}} \sum_{i=1}^{N_{SIM}} e^{-r\tau_i} \left(K - S_i(\tau_i)\right)^+$$

$$\tau_i = \min\left\{ T, \inf\{ t>0 : S_i(t) \le \beta \} \right\}$$

for a given β.

STOPPING TIME.

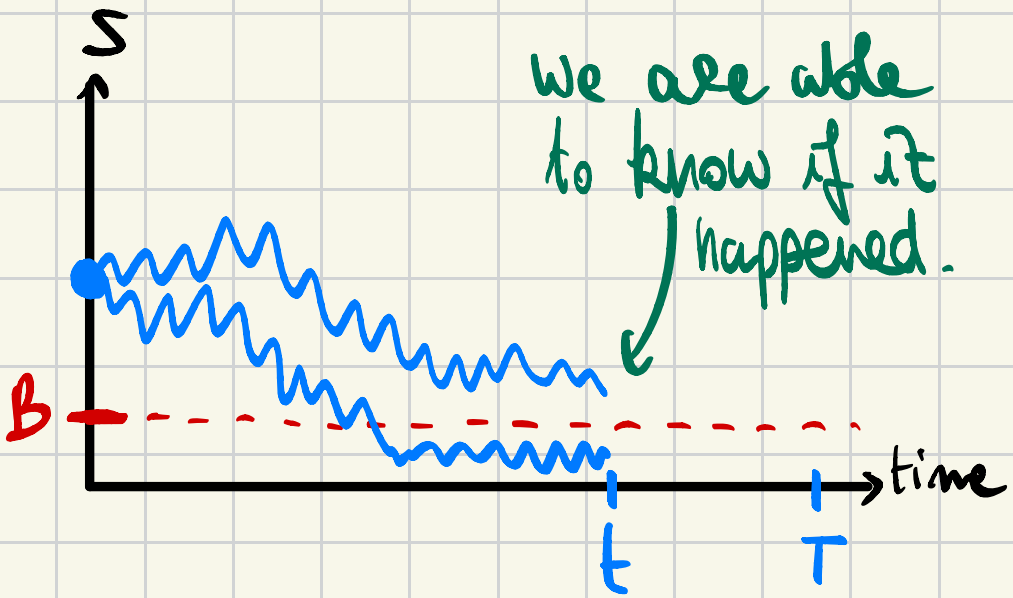
$S_i(t) \le \beta$: stopping time vs $S_i(t) = \max_{\tau \in [0,T]} S_i(\tau)$: not a stopping time.

A

$X_t = \mathbb{1}\{S_i(t) \le \beta\}$

B

A



X_t is $\mathcal{F}_t$ - measurable

B

$$X_t = \mathbb{1}_{\{S_i(t) = \max_{\tau \in [0,T]} S_i(\tau)\}}$$

For sure, it's $\mathcal{F}_T$ - measurable. But it's not $\mathcal{F}_t$ - measurable for $t < T$.

So the idea in (*) is to replace the $\max_{\tau \in [0,T]}$ by a Stopping time.

↳ It will give us a LOWER BOUND.

So we obtain:

$$\max_{\beta} \text{price}_1(\beta) \leq \text{price BM} \leq \text{price}^*$$

But we don't know if this interval is small.

So we need to solve this problem.

? We want to use MC ⊕ we want to avoid future knowledge

↓

FIRST IDEA : "Nested Monte Carlo".

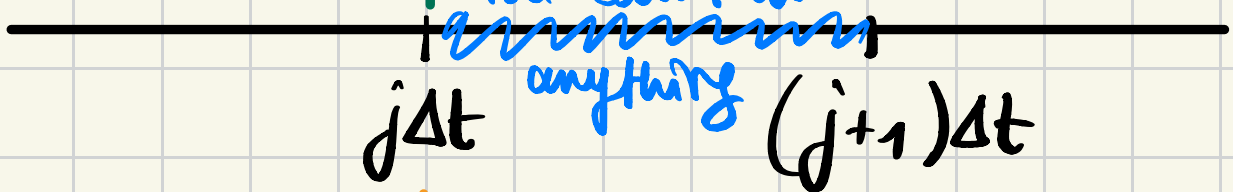
↓

Highly Computationally Intensive

In Bermudan, @ time $t = j\Delta t$, you have two possibilities:

- Not exercise;
- exercise.

Early ex.: $(K - S_i(j\Delta t))^+$



No early ex.:

$$\mathbb{E}^{\mathbb{Q}}\left[e^{-r\Delta t} P_{BM}((j+1)\Delta t, S((j+1)\Delta t)) \mid \mathcal{F}_{j\Delta t} \right]$$

↖ We have to compute an expectation.

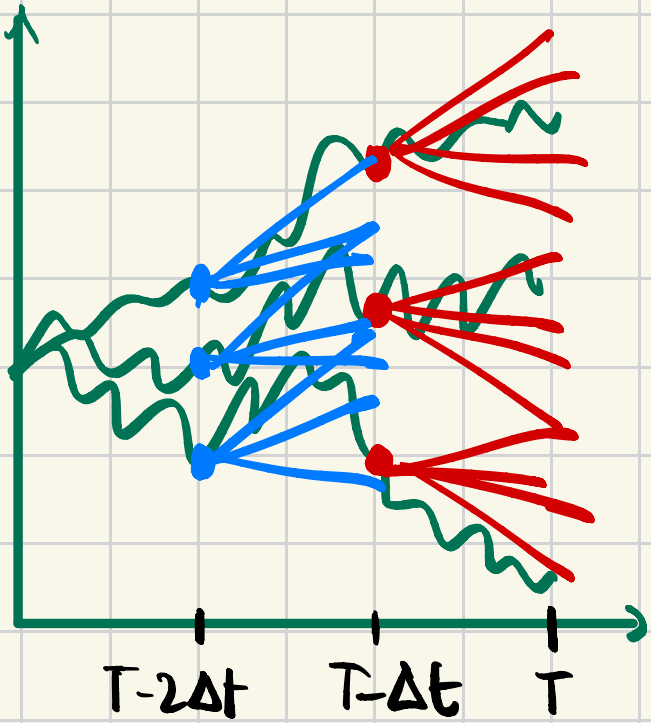
↳ How? By a MC! NESTED MC

S_{ij}

For each simulation :

(time before mat.) $(M-1)\Delta t$ ⟶ T (maturity)

$(K - S_i((M-1)\Delta t))^+$ Early exercise price.



$$\mathbb{E}^{\mathbb{Q}}\left[e^{-r\Delta t} P_{BM}(M\Delta t) \mid \mathcal{F}_{(M-1)\Delta t}\right]$$

$$\mathbb{E}^{\mathbb{Q}}_{(M-1)\Delta t}\left[e^{-r\Delta t}(K - S_i(M\Delta t))^+\right]$$

I perform simulation to compute this expected value.

VERY COMP. COMPLEX

If you have only 5 or 6 monitoring dates, NESTED MC can work. But if it's a daily monitoring Bermudan option, the complexity is too high.

The solution is to exploit :

Regression Method.

@ any time, $P_{BM} = \max(IV, CV)$

"intrinsic" "continuation"

IV ↑ exercise.

CV ↑ No exercise.

$$CV(t) = \mathbb{E}_t^{\mathbb{Q}}\left[e^{-r\Delta t} \, P_{BM}\left(S(t+\Delta t)\right)\right]$$

let us try to approximate this $\longrightarrow$

(since it's not possible to use MC).

$$\simeq \sum_{\ell=0}^{L} \alpha_\ell \left(S(t)\right)^\ell \qquad \text{"Regression"}$$

$$= \alpha_0 + \alpha_1 S(t) + \alpha_2 S(t)^2 + \dots$$

↳ we don't exploit the future: $S(t)$ is known @ time t.

IDEA "Longstaff & Schwartz Algorithm".] cf. article in .pdf.

↳ Algo. to estimate $\alpha_0, \alpha_1, \dots$: using $(S_{ij})_{ij}$ "again".

Early Exercise

May we use MC simulations for American Put Options?

Yes, under a SDO framework!!!

Longstaff, Schwartz, *Valuing American Options by Simulation: a Simple Least-Squares Approach*
http://repositories.cdlib.org/anderson/fin/1-01

- As usual with Monte Carlo simulation, we generate sample paths $(S_0, S_1, \cdots, S_j, \cdots, S_M)$, where j is a discrete time index, $S_j = S(j\delta t)$, and $T = M\delta t$ is the expiration time of the option
- We denote by V_j the value of the option at time j
- We denote by IV_j the intrinsic value of the option at time j

$$IV_j = \max(K - S_j, 0)$$

- We denote by CV_j the continuation value of the option at time j

$$CV_j = E\left[e^{-r\delta t} V_{j+1} | S_j\right]$$

↑ This is the pb : how to estimate this ?

The dynamic programming recursion for the value function V is

$$V_M = \max\{K - S_M, 0\}$$
$$V_j = \max\{IV_j, CV_j\}, \; j = M-1, \cdots, 0$$

Assume that we simulate S till time j

- We can compute IV_j
- We cannot compute CV_j since V_{j+1} is unknown at time j

if we are at a given point of a sample path generated by Monte Carlo sampling, we cannot exploit knowledge of future prices along that path, as this would imply clairvoyance

MC Approach

1. We simulate a trajectory from time 0 to time T with M time steps
2. We compute the intrinsic value IV_j, $j = 0, \cdots, M$

Problem: In computing CV_j we cannot use the simulated values $S_{j+1}, \cdots, S_M$, since, at time j are "unknown"

Idea: Approximate the Continuation Value using only the value S_j, which is known!!!

Least Square Approach

Idea: Approximate the Continuation Value with polynomials

$$CV_j = E\left[e^{-r\delta t} V_{j+1} | S_j\right] \approx \sum_{k=1}^{L} \alpha_{k,j} \psi_k(S_j) = \sum_{k=1}^{L} \alpha_{k,j} S_j^{k-1}$$

- Consider a basis of monomials:
  $\psi_I(S) = 1$, $\psi_2(S) = S$, $\psi_3(S) = S^2, \cdots, \psi_L(S) = S^{L-1}$
- Orthogonal polynomials can also be used
- Note that we are using the same set of basis function for each time instant, but the weights in the linear combination will depend on time
- The weights $\alpha_{k,j}$ can be found by linear regression, going backward in time, considering all the simulated paths

Computing the weights

- In order to illustrate the method, we should start from the last time period
- Assume we have generated N sample paths, and let us denote by $S_{j,i}$ the price at time j on sample path i, $i = 1, \cdots, N$
- When $j = M$, the value function V_M is the payoff

$$e^{-r\delta t} \max(K - S_{M,i}, 0) = \sum_{k=1}^{L} \alpha_{k,M-1} S_{M-1,i}^{k-1} + e_i$$

  where e_i is the error due to our approximation
- The weights $\alpha_{k,j}$ do not depend on i

We have to solve the following least square problem

$$\min_{\alpha_{1,M-1},\cdots,\alpha_{K,M-1}} \sum_{i=1}^{N} e_i^2$$

- In the regression above, we have considered all of the generated sample paths
- It is better to consider only the subset of sample paths for which we have a decision to take at time $j = M - 1$
- This subset is simply the set of sample paths in which the option is in the money at time $j = M - 1$
- In fact if the option is not in the money, we have no reason to exercise; using only the sample paths for which the option is in the money is called the moneyness criterion and it improves the performance of the overall approach

The process is repeated going backward in time: if we are at time step j

1. For each path i such that

$$IV_{j,i} = \max(K - S_{j,i}, 0) > 0$$

there will be an exercise time $j_e > j$, which we set conventionally to M if the option will never be exercised in the future

2. Then the regression problem should be

$$\min_{\alpha_{1,j},\cdots,\alpha_{L,j}} \sum_{i=1}^{N} e_i^2$$

with

$$e_i = e^{-r(j_e-j)\delta t} \max(K - S_{j_e,i}, 0) - \sum_{k=1}^{L} \alpha_{k,j} S_{j,i}^{k-1}$$

3. approximate the Continuation Value

$$CV_{j,i} \approx \sum_{k=1}^{L} \alpha_{k,j} S_{j,i}^{k-1}$$

4. using the approximation of CV

$$V_{j,i} = \max(IV_{j,i}, CV_{j,i})$$

Thus

- At each step we have to solve a least square problem
- In this way we compute the weights α
- Thus we obtain an approximation of the Continuation Values

Least Square in Matlab

Solve $Ax = b$, where A is a rectangular matrix, using

`x=A\b`

returns a least square solution

In our case at the time step j

$$A_{i,k} = S_{j,i}^{k-1} \quad x_k = \alpha_{k,j} \quad b_i = e^{-r(j_e - j)\delta t} \max(K - S_{j_e,i}, 0)$$

$i = 1, \cdots, N$, $k = 1, \cdots, L$

In the Money

Better to consider only the simulation i such that $IV_{j,i} > 0$

An Example

Assume that $M = 3$, $L = 3$ and that we simulate $N = 8$ paths as follows

$j = 0$	$j = 1$	$j = 2$	$j = 3$
1	1.09	1.08	1.34
1	1.16	1.26	1.54
1	1.22	1.07	1.03
1	0.93	0.97	0.92
1	1.11	1.56	1.52
1	0.76	0.77	0.90
1	0.92	0.84	1.01
1	0.88	1.22	1.34

```
S0=1; K=1.1; r=0.06; T=3;  % spot price = 1, T: maturity 3y
M=3; dt=T/M; % Yearly monitoring
N=8; %Number of simulations
%---- FROM Longstaff & Schwartz ---
SPaths=[ 1.09 1.08 1.34
         1.16 1.26 1.54
         1.22 1.07 1.03
         0.93 0.97 0.92
         1.11 1.56 1.52        % Example of values from
         0.76 0.77 0.90        % the article
         0.92 0.84 1.01
         0.88 1.22 1.34];
%---- OTHERWISE RANDOM NUMBERS ----
% sigma=0.08;   mu=r-sigma^2/2;
% SPaths=Asset(S0,mu,sigma,T,M,N);   % Your asset model:
% SPaths=SPaths[:,2:end]; %without S0 % it can be Hon, B&S...
```

```
% INITIALIZATION
ExerciseTime=M*ones(N,1); %initialize at M
% (we consider only the InMoney case,
%  thus option will be exercised)

% RECURSION
CashFlows=max(0,K-SPaths(:,M)); %payoff
for step=M-1:-1:1
    InMoney=find(SPaths(:,step)<K);
    S=SPaths(InMoney,step);
    %-- Regression -----------------
    % basis functions = [1,S,S^2]
    RegrMat=[ones(length(S),1), S, S.^2];
    YData=CashFlows(InMoney).*...
          exp(-r*dt*(ExerciseTime(InMoney)-step));
    alpha=RegrMat\YData;
```

% 1st step : for each option, exercise time = T.

initialization.

% BACKWARD IN TIME PROCEDURE.

% To choose the sims to consider for the choice to exercise or not.

% poly. of degree 2

% estimation of alpha : LS pb solving (rectangular matrix)

```
    %-- IV and CV ---------------------------------
    IV=K-S; %intrinsic value
    CV=RegrMat*alpha; %continuation value
    %-- Early Exercise ----------------------------
    % Paths with early exercise at time step
    Index=find(IV>CV);
    ExercisePaths=InMoney(Index);
    % Update Cashflows
    CashFlows(ExercisePaths)=IV(Index);
    % Update Exercise Time
    ExerciseTime(ExercisePaths)=step;
end
price=max(K-S0, ...
          mean(CashFlows.*exp(-r*dt*ExerciseTime)))
```

computation.

% index of sims where IV > CV

% keep the early ex. paths.

MC ⇝ European ✓ — Any
Path Dependent ✓ — Lévy
American (Bermuda) ✓

FFT ⇝ European (Carr-Madan) ✓ — Any
Barrier (CONV) ✓ — Lévy

SEE "PDE" Notebook:

NOW → NUMERICAL PDE: "Finite differente"

1st: 1) B&S

Then: 2) PIDE

(A) Theory on Lévy.

(B) NUMERICAL. Any Lévy