Computational Finance and its implementation in Python with applications to option pricing

Andrea Mazzon

Ludwig Maximilians Universität München

Main contents

1. Monte-Carlo method for option pricing and variance reduction techniques
   - The Monte-Carlo method: motivation and a brief overview
   - Variance reduction techniques
     - Introduction
     - Antithetic variables
     - Control variates
2. Option pricing under the Binomial model
   - Motivation and setting
   - Simulation of the Binomial model
   - Calibration of the Binomial model
   - American options valuation
3. Option valuation via solution of SDEs and PDEs
   - The Feynman-Kac formula
   - Monte-Carlo simulation of continuous time stochastic processes
   - Finite differences methods for the numerical solution of PDEs
     - Introduction
     - Convergence of finite difference methods
     - Fiinite difference methods to price options on local volatility models

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

1. Monte-Carlo method for option pricing and variance reduction techniques
   - The Monte-Carlo method: motivation and a brief overview
   - Variance reduction techniques
     - Introduction
     - Antithetic variables
     - Control variates

2. Option pricing under the Binomial model
   - Motivation and setting
   - Simulation of the Binomial model
   - Calibration of the Binomial model
   - American options valuation

3. Option valuation via solution of SDEs and PDEs
   - The Feynman-Kac formula
   - Monte-Carlo simulation of continuous time stochastic processes
   - Finite differences methods for the numerical solution of PDEs
     - Introduction
     - Convergence of finite difference methods
     - Fiinite difference methods to price options on local volatility models

Motivation

- A common problem we face in mathematical finance is the risk neutral valuation of a derivative.
- As you know, the price of a derivative is expressed by the (possibly discounted) expectation of its payoff at maturity, under a pricing measure (also called risk neutral, or martingale measure).
- That is, we have to compute the expectation of a random variable.
- Problem: most often, there is no way to get an analytic formula for the expectation of complex derivates, or even simpler derivatives written on an underlying with non trivial dynamics.
- Broad idea: we can approximate the price by averaging some possible, simulated realizations of the payoff.
- The strong law of large numbers and some other convergence results may help us.

A bit more precisely..

- Consider a random variable $X : \Omega \to \mathbb{R}^N$ defined on a probability space $(\Omega, \mathcal{F}, P)$. The probability measure P may be viewed as a risk neutral measure.
- Also consider a (payoff) function $f : \mathbb{R}^N \to \mathbb{R}$ such that Var$[f(X)] < \infty$.
- The aim is to compute the expectation

$$\mu := \mathbb{E}^P[f(X)] = \int_\Omega f(X)dP.$$

- Suppose there is no analytic formula to derive μ above. We have to find an approximation $\hat{\mu}$.

We can define independent drawings of X

- Given $X:\Omega\to\mathbb{R}$ and $(\Omega,\mathcal{F},P)$ as above, introduce:

$$\begin{aligned}\tilde{\Omega} &:= \Omega\times\Omega\times\cdot\cdot\times\Omega = \{\tilde{\omega}=(\omega_1,\dots,\omega_n),\quad \omega_i\in\Omega\},\\ \tilde{\mathcal{F}} &:= \sigma(\mathcal{F}\times\mathcal{F}\times\cdot\cdot\times\mathcal{F}),\\ \tilde{P}\left(\prod_{i=1}^n A_i\right) &:= \prod_{i=1}^n P(A_i),\quad A_i\in\mathcal{F}.\end{aligned}$$

- Also define the random variable $\tilde{X}=(\tilde{X}_1,\dots,\tilde{X}_n)$ by $\tilde{X}_i(\tilde{\omega}):=X(\omega_i)$.
- This is a way to see $\tilde{X}(\tilde{\omega})$ as n different realizations $X(\omega_i)$, $i=1,\dots,n$ of one random variable X, or as one realization of n i.i.d. random variables $\tilde{X}_i(\tilde{\omega})$, $i=1,\dots,n$.
- This interpretation is at the base of the Monte-Carlo method, as it permits to exploit the Strong Law of Large Numbers.
- A similar construction and interpretation can be given for a N-dimensional random variable X.

Convergence results for sequences of i.i.d. random variables

Theorem: Strong Law of Large Numbers

Let $(X_i)_{i\in\mathbb{N}}$ be i.i.d. integrable real valued random variables on $(\Omega, \mathcal{F}, P)$, and set

$$\mu := \mathbb{E}^P[X_i], \quad i \in \mathbb{N}.$$

Then

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i = \mu \quad P - a.s.$$

Theorem: Tschebyscheff Inequality

Let $(X_i)_{i\in\mathbb{N}}$ be i.i.d. square integrable real valued random variables on $(\Omega, \mathcal{F}, P)$, and set

$$\mu := \mathbb{E}^P[X_i], \quad \sigma^2 := \mathbb{E}^P\left[(X_i-\mu)^2\right], \qquad i \in \mathbb{N}.$$

Then for any $\epsilon, \delta > 0$ and any $n \in \mathbb{N}$ we have

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \epsilon\right) \leq \frac{\sigma^2}{\epsilon^2 n}$$

and

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \frac{\sigma}{\delta^{1/2} n^{1/2}}\right) \leq \delta.$$

Application to Monte-Carlo

Lemma

Let $(X_i)_{i\in\mathbb{N}}$ be a collection of i.i.d. integrable random variables on $(\Omega, \mathcal{F}, P)$ with values in $\mathbb{R}^N$, and let $f : \mathbb{R}^N \to \mathbb{R}$. Then the random variables $(f(X_i))_{i\in\mathbb{N}}$ are also i.i.d.

- The lemma above, together with the convergence results of the previous slide, allows us to approximate

$$\mu := \mathbb{E}^P[f(X)] = \int_\Omega f(X)dP$$

  by

$$\hat{\mu} := \frac{1}{n}\sum_{i=1}^{n} f(X_i),$$

  where $(X_i)_{i=1,\dots,n}$ are independent realizations of X.
- We can generate numerically n realizations of a random variable X with a given distribution P^X, starting from a sequence of (pseudo!) random numbers.
- One must give a *seed*, i.e., a starting point for the pseudo-random numbers sequence.
- The realizations will not be purely random, and not purely independent.

Pro and cons of Monte-Carlo

- Pro:
  - It is very simple to understand and easy to implement.
  - The accuracy does not depend on the domain dimension (i.e., if we simulate N-dimensional random variables the accuracy is the same).
  - The accuracy can be increased by just adding more valuations without loosing the previous estimates.
  - The function f does not need to be continuous, but only square integrable.
- Cons:
  - Look at the Tschebyscheff Inequality: we only have a probabilistic bound. The worst case error is ∞.
  - The estimates depend on the generated random sequence. The sequence is not purely random. First, one has to find a good random number generator.
- There are techniques that can be used to increase the accuracy. In the next slides we will see few of them.

Low-discrepancy sequences

Remark

If X has uniform distribution or has a cumulative distribution function F which is easy to invert (in that case a realization x_i can be generated as $x_i = F^{-1}(u_i)$, with u_i realization of $U \sim U((0,1))$) then approximating $\mathbb{E}[f(X)]$ reduces to approximate

$$\int_0^1 G(x)dx, \tag{1}$$

for $G = f \circ F^{-1}$.

Theorem: Koksma-Hlawka inequality

If G has bounded total variation on $(0,1)$, then for any points $x_1, \dots, x_n \in (0,1)$ it holds

$$\left| \frac{1}{n} \sum_{i=1}^{n} G(x_i) - \int_0^1 G(x)dx \right| \le V(G) D^*(x_1, \dots, x_n),$$

where

$$V(G) = \sup_S \sum_i |G(y_{i+1}) - G(y_i)|$$

over all partitions $S := \{0 = y_1 < y_2 < \dots < y_n = 1\}$ and $D^*(x_1, \dots, x_n)$ is the star discrepancy

$$D^*(x_1, \dots, x_n) = \sup_{b \in (0,1)} \left| \frac{|\#\{x_i : 0 \le x_i \le b\}|}{n} - b \right|.$$

Low-discrepancy sequences

- The result in the previous slide also holds for higher dimensions (here we just wanted to simplify the notation).
- It gives the motivation to look for low discrepancy sequences.
- Most well known low discrepancy sequences: Van der Corput, Halton, Sobol, Hammersley, Sobol, Niederreiter.
- Here we don't focus on Low discrepancy sequences. A bit of references if you want to go deeper on this:
  - J. Dick and F. Pillichshammer, *Digital Nets and Sequences. Discrepancy Theory and Quasi-Monte Carlo Integration*, Cambridge University Press, Cambridge, 2010
  - M. Drmota and R. F. Tichy, *Sequences, discrepancies and applications*, Lecture Notes in Math., 1651, Springer, Berlin, 1997.
  - L. Kuipers, H. Niederreiter, *Uniform distribution of sequences*, Dover Publications, 2005.
  - . . . the course *Numerical Methods for Financial Mathematics* at our master!
- We focus instead on variance reduction techniques.

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

1. Monte-Carlo method for option pricing and variance reduction techniques
   - The Monte-Carlo method: motivation and a brief overview
   - Variance reduction techniques
     - Introduction
     - Antithetic variables
     - Control variates

2. Option pricing under the Binomial model
   - Motivation and setting
   - Simulation of the Binomial model
   - Calibration of the Binomial model
   - American options valuation

3. Option valuation via solution of SDEs and PDEs
   - The Feynman-Kac formula
   - Monte-Carlo simulation of continuous time stochastic processes
   - Finite differences methods for the numerical solution of PDEs
     - Introduction
     - Convergence of finite difference methods
     - Fiinite difference methods to price options on local volatility models

Motivation - 1

- Consider a random variable $X : \Omega \to \mathbb{R}^N$ defined on a probability space $(\Omega, \mathcal{F}, P)$ and a (payoff) function $f : \mathbb{R}^N \to \mathbb{R}$ such that Var$[f(X)] < \infty$.
- Monte-Carlo method: choosing $n \in \mathbb{N}$ large enough, we approximate

$$\hat{\mu} := \frac{1}{n}\sum_{k=1}^{n} f(X_i) \approx \mu := \mathbb{E}^P[f(X)],$$

where $(X_i)_{i=1,\dots,n}$ are realizations of X, i.e., have same distribution as X.

- The estimator is of course *unbiased*, i.e.,

$$\mathbb{E}^P\left[\hat{\mu}\right] = \mathbb{E}^P\left[\frac{1}{n}\sum_{k=1}^{n} f(X_i)\right] = \mathbb{E}^P\left[f(X)\right] =: \mu$$

- We are interested in the variance of our estimator, i.e., in the quantity

$$\text{Var}\left(\hat{\mu}\right) = \mathbb{E}^P\left[\left(\frac{1}{n}\sum_{k=1}^{n} f(X_i) - \mu\right)^2\right].$$

Motivation - 2

- We have seen that if $(X_i)_{i=1,\dots,n}$ are independent, we have convergence results for our estimator. Moreover,

$$\text{Var}\left(\hat{\mu}\right) = \mathbb{E}^P\left[\left(\frac{1}{n}\sum_{k=1}^{n} f(X_i) - \mu\right)^2\right] = \frac{1}{n}\text{Var}[f(X)].$$

- It makes sense: the larger the number n of simulated realizations of X, the smaller the variance of our estimator.
- In particular, we have to increase the number of simulations by a factor of C to reduce the standard deviation by a factor of $\sqrt{C}$.
- The question now is: can we do it better?
- Variance reduction techniques aim to reduce the variance of our estimator, without increasing the number of simulations.

Some variance reduction techniques

Three well known variance reduction techniques are:

- Antithetic variables
- Control variates
- Importance sampling

We will focus mostly on the first two techniques, together with applied examples. Here some references if you want to deepen Importance sampling:

- A, Bouhari. *Adaptative Monte Carlo Method, A Variance Reduction Technique*. Monte Carlo Methods and Their Applications. 10 (1): 1-24, 2004.
- P. J. Smith, M. Shafi, H. Gao. *Quick simulation: A review of importance sampling techniques in communication systems*. IEEE Journal on Selected Areas in Communications. 15 (4): 597-613, 1997.
- Again, the course *Numerical Methods for Financial Mathematics* at our master!

1 Monte-Carlo method for option pricing and variance reduction techniques

- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
    - Introduction
    - Antithetic variables
    - Control variates

2 Option pricing under the Binomial model

- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs

- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
    - Introduction
    - Convergence of finite difference methods
    - Fiinite difference methods to price options on local volatility models

Let's start from a simple result..

Lemma

Let $f, h : \mathbb{R} \to \mathbb{R}$ be two monotone functions, both increasing or both decreasing, and let $X : \Omega \to \mathbb{R}$ be a random variable defined on a probability space $(\Omega, \mathcal{F}, P)$. Then

$$\mathbb{E}^P[f(X)h(X)] \geq \mathbb{E}^P[f(X)]\mathbb{E}^P[h(X)].$$

Proof

The monotonicity assumption on f and h implies that for any $x, y \in \mathbb{R}$ we have

$$(f(x) - f(y))\,(h(x) - h(y)) \geq 0.$$

Therefore, for any i.i.d. real valued random variables X and Y on $(\Omega, \mathcal{F}, P)$ it holds

$$(f(X) - f(Y))\,(h(X) - h(Y)) \geq 0$$

and then

$$\mathbb{E}^P\left[(f(X) - f(Y))\,(h(X) - h(Y))\right] \geq 0,$$

so that

$$\mathbb{E}^P\left[f(X)h(X)\right] + \mathbb{E}^P\left[f(Y)h(Y)\right] \geq \mathbb{E}^P\left[f(Y)h(X)\right] + \mathbb{E}^P\left[f(X)h(Y)\right].$$

Since X and Y are identically distributed, it follows that

$$2\mathbb{E}^P\left[f(X)h(X)\right] \geq 2\mathbb{E}^P\left[f(Y)h(X)\right],$$

and since they are also independent, this implies that

$$\mathbb{E}^P[f(X)h(X)] \geq \mathbb{E}^P[f(X)]\mathbb{E}^P[h(X)].$$

An interesting consequence

Proposition

Let $f : \mathbb{R} \to \mathbb{R}$ be a monotone function, and $X : \Omega \to \mathbb{R}$ a random variable defined on a probability space $(\Omega, \mathcal{F}, P)$. Then

$$\text{Cov}[f(X), f(-X)] \leq 0.$$

Proof

We have that

$$\text{Cov}[f(X), f(-X)] = \mathbb{E}^P[f(X)f(-X)] - \mathbb{E}^P[f(X)]\mathbb{E}^P[f(-X)].$$

The result then follows since a direct application of the Lemma of the previous slide with $h(x) := -f(-x)$ implies that

$$\mathbb{E}^P[f(X)]\mathbb{E}^P[f(-X)] \geq \mathbb{E}^P[f(X)f(-X)].$$

Application to Monte-Carlo

- Let $f : \mathbb{R} \to \mathbb{R}$ be a monotone function, and let $X : \Omega \to \mathbb{R}$ be a symmetric random variable defined on a probability space $(\Omega, \mathcal{F}, P)$.
- From the last proposition we know that
$$\text{Cov}[f(X), f(-X)] \leq 0.$$
- Idea: choose n even and generate $n/2$ realizations of X, call them $(X_i)_{i=1,\dots,n/2}$. Then define $X_{n/2+i} := -X_i$, $i = 1, \dots, n/2$.
- Since X is symmetric, the estimator is unbiased:
$$\mathbb{E}^P[\hat{\mu}] = \frac{1}{n}\mathbb{E}\left[\sum_{k=1}^{n/2} f(X_i) + \sum_{k=1}^{n/2} f(-X_i)\right] = \frac{1}{n}\left(\sum_{k=1}^{n/2} \mathbb{E}^P[f(X_i)] + \sum_{k=1}^{n/2} \mathbb{E}^P[f(-X_i)]\right) = \mu.$$
- What about the variance?
$$\begin{aligned}
\text{Var}[\hat{\mu}] &= \frac{1}{n^2}\text{Var}\left[\sum_{k=1}^{n/2} f(X_i) + \sum_{k=1}^{n/2} f(-X_i)\right] \\
&= \frac{1}{n^2}\left(n\text{Var}[f(X)] + \text{Cov}\left(\sum_{k=1}^{n/2} f(X_i), \sum_{k=1}^{n/2} f(-X_i)\right)\right) \\
&= \frac{1}{n}\text{Var}[f(X)] + \frac{1}{n}\text{Cov}[f(X), f(-X)] \leq \frac{1}{n}\text{Var}[f(X)].
\end{aligned}$$

Application to Monte-Carlo

- To recap: if X is symmetric, then setting $X_{n/2+i} := -X_i$ for $i = 1, \dots, n/2$ gives us an unbiased estimator $\hat{\mu}$ such that

$$\text{Var}[\hat{\mu}] \leq \frac{1}{n}\text{Var}[f(X)].$$

- But $\frac{1}{n}\text{Var}[f(X)]$ is the variance of the classical estimator, when we generate n i.i.d. realizations of X!
- In this way, we reduce the variance of the estimator.
- This approach is known as Antithetic variables.

Antithetic variables for non symmetric X

- Let $f : \mathbb{R} \to \mathbb{R}$ be a monotone function, and let $X : \Omega \to \mathbb{R}$ be a random variable defined on a probability space $(\Omega, \mathcal{F}, P)$.
- Suppose X to be not symmetric. How can we apply Antithetic variables to reduce the variance of our estimator?
- Call F the cumulative distribution function of X. Suppose that we know (at least a good approximation of) F^{-1}.
- Well known result: let $U \sim$ Unif$(0,1)$ and define $Y := F^{-1}(U)$. Then X and Y have same distribution.
- Let $U \sim$ Unif$(0,1)$. Because of the result above, we have

$$\mathbb{E}^P[f(X)] = \mathbb{E}^P[h(U)]$$

  with $h(x) = f \circ F^{-1}$.
- Simulate independent realizations $(U_i)_{i=1,\dots,n/2}$ and define $U_{n/2+i} := 1 - U_i$, $i = 1, \dots, n/2$.
- The associated estimator is unbiased since

$$\mathbb{E}^P[h(U)] = \mathbb{E}^P[h(1-U)]$$

- Similarly to before, it can also be seen that since f and F is monotone,

$$Cov[h(U), h(1-U)] \leq 0.$$

- So this is also an Antithetic variables approach.

Example: valuation of a call option under Black-Scholes

- We want to test the benefits of using Antithetic variables in the valuation of a call option under the Black-Scholes model.
- This is indeed a case when we have of course the benchmark of the analytic formula for a call option.
- In particular, we want to approximate the expectation $\mathbb{E}^P[g(X_T)]$ for $T > 0$, in the case when

  $$g(x) = (x - K)^+$$

  with $K > 0$ and $X = (X_t)_{0 \leq t \leq T}$ is a stochastic process with initial value $X_0 = x_0$ and dynamics

  $$dX_t = rX_t dt + \sigma X_t dW_t, \quad 0 \leq t \leq T,$$

  where $W = (W_t)_{0 \leq t \leq T}$ is P-Brownian motion.
- Interpretation: r is the risk free rate and P is the martingale measure, i.e., the probability measure under which the discounted process $(e^{-rt}X_t)_{0 \leq t \leq T}$ is a martingale.

Valuation with Antithetic variables

- The problem reduces to the valuation of the expectation

$$\mathbb{E}^P[(X-K)^+]$$

  where X is the random variable

$$X = x_0 e^{(r-\sigma^2/2)T+\sigma\sqrt{T}Z},$$

  with $Z \sim \mathcal{N}(0,1)$.

- That is, we have to valuate

$$\mathbb{E}^P\left[f(Z)\right]$$

  where

$$f(z) = \left(x_0 e^{(r-\sigma^2/2)T+\sigma\sqrt{T}z} - K\right)^+.$$

- So, we have a function of a symmetric random variable! We can directly use Antithetic variables.
- We simulate $n/2$ realizations $(z_i)_{i=1,\dots,n/2}$ of a standard normal random variable and then define $z_{i+n/2} = -z_i$, $i = 1, \dots, n/2$.

Implementation with Python

- In the Python package

  `montecarlovariancereduction.antitheticvariables`

  you can find the code relative to the comparison of Antithetic variables against the standard Monte-Carlo method.
- In particular, in the class `GenerateBlackScholes` we generate the values of

  $$X = x_0 e^{(r-\sigma^2)T+\sigma\sqrt{T}Z},$$

  starting from the ones of Z. We do this using both the standard Monte-Carlo approach and the Antithetic variables approach illustrated in the previous slide.
- Note that the method

  `numpy.random.standard_normal(n)`

  generates n returns of a standard normal random variable. In this case, we give no seed: it will be different every time this method is called.

Experiment and results

In

`antitheticVariablesTest`

and

`compareStandardMCWithAV`

we do the following experiment:

- We fix the parameters $x_0 = K = 100$, $T = 3$, $r = 0.05$, $\sigma = 0.5$.
- For any number of simulations $n = 10^3$, 10^4, 10^5 and 10^6, we perform 100 different valuations of the price of the call option, both with the standard and the Antithetic variables Monte-Carlo method.
- We then compute the average percentage error for both the methods.

The following table illustrates the results:

	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$
av. $\%$ error standard MC	6.25	2.07	0.59	0.20
av. $\%$ error AV	5.51	1.77	0.53	0.17

1. Monte-Carlo method for option pricing and variance reduction techniques
   - The Monte-Carlo method: motivation and a brief overview
   - Variance reduction techniques
     - Introduction
     - Antithetic variables
     - Control variates

2. Option pricing under the Binomial model
   - Motivation and setting
   - Simulation of the Binomial model
   - Calibration of the Binomial model
   - American options valuation

3. Option valuation via solution of SDEs and PDEs
   - The Feynman-Kac formula
   - Monte-Carlo simulation of continuous time stochastic processes
   - Finite differences methods for the numerical solution of PDEs
     - Introduction
     - Convergence of finite difference methods
     - Fiinite difference methods to price options on local volatility models

Setting and motivation

- Let $X, Y : \Omega \to \mathbb{R}$ be two random variables defined on a probability space $(\Omega, \mathcal{F}, P)$.
- Suppose you know the analytic value of

$$\mu_X := \mathbb{E}^P[X], \qquad \sigma_X^2 := \text{Var}[X], \qquad \sigma_{XY} := \text{Cov}[X, Y],$$

  and also suppose $\sigma_{XY} > 0$.
- Assume you want to approximate

$$\mu_Y := \mathbb{E}^P[Y].$$

- The goal is to find an unbiased estimator of μ_Y which has low variance.

Control variates - 1

- Consider n independent realizations (X_i, Y_i) of (X, Y), $i = 1, \dots, n$, and define

$$\hat{\mu}_X := \frac{1}{n}\sum_{i=1}^{n} X_i, \qquad \hat{\mu}_Y := \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

- Note that

$$\text{Cov}[\hat{\mu}_X, \hat{\mu}_Y] = \frac{1}{n}\sigma_{XY}.$$

- What about an estimator

$$\hat{\mu}_Y^{CV} := \hat{\mu}_Y - \beta(\hat{\mu}_X - \mu_X)$$

  for a given $\beta > 0$?
- It is unbiased:

$$\mathbb{E}^P[\hat{\mu}_Y^{CV}] = \mathbb{E}^P[\hat{\mu}_Y] - \beta\mathbb{E}^P[\hat{\mu}_X - \mu_X] = \mu_Y.$$

- What about the variance?

$$\text{Var}[\hat{\mu}_Y^{CV}] = \frac{1}{n}\sigma_Y^2 + \beta^2\frac{1}{n}\sigma_X^2 - 2\beta\frac{1}{n}\sigma_{XY}.$$

- It is minimized by $\beta = \frac{\sigma_{XY}}{\sigma_X^2}$. For such a value of β, we find

$$\text{Var}[\hat{\mu}_Y^{CV}] = \text{Var}[\hat{\mu}_Y] - \frac{1}{n}\frac{\sigma_{XY}}{\sigma_X^2}.$$

Control variates - 2

- We have seen that taking

$$\hat{\mu}_Y^{CV} := \hat{\mu}_Y - \beta(\hat{\mu}_X - \mu_X), \qquad \beta = \frac{\sigma_{XY}}{\sigma_X^2}$$

  gives an optimal variance

$$\text{Var}[\hat{\mu}_Y^{CV}] = \text{Var}[\hat{\mu}_Y] - \frac{1}{n}\frac{\sigma_{XY}}{\sigma_X^2}.$$

- Note that the gain of the new estimator with respect to the old one only depends on the correlation of X and Y:

$$\frac{\text{Var}[\hat{\mu}_Y^{CV}]}{\text{Var}[\hat{\mu}_Y]} = 1 - \frac{\sigma_{XY}}{n\sigma_X^2\text{Var}[\hat{\mu}_Y]} = 1 - \frac{\sigma_{XY}}{\sigma_X^2\sigma_Y^2} = 1 - \rho_{XY}^2.$$

- Problem: we have to compute $\beta = \frac{\sigma_{XY}}{\sigma_X^2}$, but often we don't know σ_X^2 and σ_{XY}.
- Solution: estimate σ_X^2 and σ_{XY} from the generated sample, i.e., set

$$\hat{\sigma}_X^2 = \frac{1}{n-1}\sum_{i=1}^n (X_i - \hat{\mu}_X)^2, \qquad \hat{\sigma}_{XY} = \frac{1}{n-1}\sum_{i=1}^n (X_i - \hat{\mu}_X)(Y_i - \hat{\mu}_Y)$$

  and choose

$$\beta = \frac{\hat{\sigma}_{XY}}{\hat{\sigma}_X^2}.$$

- Note that this last choice of β actually depends on the generated sample.
- The associated estimator $\hat{\mu}_Y^{CV} := \hat{\mu}_Y - \beta(\hat{\mu}_X - \mu_X)$ is thus unbiased only asymptotically.

Multi-dimensional control variates

Exercise

Consider now the case when X has values in $\mathbb{R}^N$, $N \geq 1$.
Assume you know the $N \times N$ matrix Cov$(X) =: \Sigma_X$ and the N-dimensional vector Cov$(X, Y) = \sigma_{X,Y}$. Also assume that Σ_X is positive definite.
Consider the estimator

$$\hat{\mu}_Y^{CV} = \hat{\mu}_Y - (\hat{\mu}_X - \mu_X)^T \beta,$$

where β is a N-dimensional vector.
Find the optimal β that minimizes the variance of the estimator above and compute the variance for the optimal β you found.

Solution to the exercise

We have that

$$\text{Var}(\hat{\mu}_Y^{CV}) = \text{Var}(\hat{\mu}_Y) + \beta^T \text{Cov}(\hat{\mu}_X)\beta - 2\beta^T \text{Cov}(\hat{\mu}_X, \hat{\mu}_Y),$$

where

$$\text{Cov}(\hat{\mu}_X) = \text{Cov}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n}\Sigma_X,$$

since the realizations of X are i.i.d., and similarly

$$\text{Cov}(\hat{\mu}_X, \hat{\mu}_Y) = \frac{1}{n}\sigma_{X,Y}.$$

Then we want to find the value of β that minimizes

$$\phi(\beta) := \beta^T \Sigma_X \beta - 2\beta^T \Sigma_{X,Y}.$$

Since Σ_X is positive definite, the function ϕ is convex, and it is minimized by the vector β such that

$$\Sigma_X \beta - \Sigma_{X,Y} = 0,$$

i.e.,

$$\beta = \Sigma_X^{-1}\Sigma_{X,Y}.$$

With such a choice of β, we get

$$\text{Var}(\hat{\mu}_Y^{CV}) = \text{Var}(\hat{\mu}_Y) - \frac{1}{n}\Sigma_{X,Y}^T \Sigma_X^{-1} \Sigma_{X,Y}.$$

Application: Cliquet options

- Cliquet options are an example of exotic, path dependent options. In particular, their payoff depends on the returns of the underlying.
- Let $X = (X_t)_{t \in [0,T]}$ be a stochastic process on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, P)$.
- Fix a partition

  $$0 = t_0 < t_1 < \cdots < t_N := T$$

  of the interval $[0, T]$.
- For any $n = 1, \dots, N$ define $R_n^* := (R_n)_{[F_\ell, C_\ell]}$ for $F_\ell < C_\ell$, where

  $$R_n := \frac{X_{t_n}}{X_{t_{n-1}}} - 1$$

  is the n-th return and $(x)_{[a,b]} := \min\left(\max(x, a), b\right)$, $a < b$, is the truncation of x.
- The payoff of the Cliquet option with local floor and cap F_ℓ, C_ℓ, global floor and cap $F_g < C_g$ and monitoring dates $0 < t_1 < \cdots < t_N := T$ is then

  $$R_g^* := (R_g)_{[F_g, C_g]}$$

  where

  $$R_g = R_1^* + R_2^* + \cdots + R_N^*.$$

Control variates for Cliquet options: motivation

- There is no analytic formula for the expectation of the payoff of a Cliquet option, not even under the Black-Scholes model.
- Observation: there is of course a positive correlation between $R_g^* := (R_g)_{[F_g, C_g]}$ and R_g, and also between R_g^* and R_k^*, $k = 1, \dots, N$, since

$$R_g = R_1^* + R_2^* + \cdots + R_N^*.$$

- Can we find an analytic formula for the expectation of R_g and R_n^*, at least under a suitable model as Black-Scholes?

Control variates for Cliquet options - 1

Lemma

Let $b > a$. The truncating function $(x)_{[a,b]} := \min\left(\max(x,a), b\right)$ can be rewritten as

$$(x)_{[a,b]} = a + (x-a)^+ - (x-b)^+.$$

Proof

We have that

$$\begin{aligned} a + (x-a)^+ - (x-b)^+ &= a + \max(x-a, 0) + \min(b-x, 0) \\ &= \max(x, a) + \min(b-x, 0). \end{aligned}$$

We then easily see that both $\min\left(\max(x,a), b\right)$ and the function above are equal to a when $x < a$, x if $a \le x \le b$ and b if $x > b$.

Control variates for Cliquet options - 2

- The lemma in the previous slide tells us that, defining $Y_n := R_n + 1$, the quantity R_n^* can be seen as the difference between two payoffs of call options, plus a constant:

$$R_n^* = F_\ell + (Y_n - (F_\ell + 1))^+ - (Y_n - (C_\ell + 1))^+.$$

- That is, we have an analytic formula for the expectation of R_n^*, at least if Y_n is log-normal or normal.
- It is it reasonable to expect that R_g^* and R_g are more correlated than R_g^* and R_n^*.
- So, what about an analytic formula for the expectation of

$$R_g = R_1^* + R_2^* + \cdots + R_N^*?$$

This comes directly from the one for R_n^*.

Control variates for Cliquet options under Black-Scholes

- We assume that our underlying X follows dynamics

  $$dX_t = rX_t dt + \sigma X_t dW_t, \quad 0 \le t \le T$$

  under the martingale measure P.
- Then the returns are given by

  $$R_n := \frac{X_{t_n}}{X_{t_{n-1}}} - 1 = \exp\left\{ \left(r - \frac{1}{2}\sigma^2 \right)(t_n - t_{n-1}) + \sigma(W_{t_n} - W_{t_{n-1}}) \right\} - 1,$$

  for any $n = 1, \dots, N$.
- The random variables $Y_n := R_n + 1$, $n = 1, \dots, N$, are independent and log-normally distributed.
- Since

  $$R_n^* = F_\ell + (Y_n - (F_\ell + 1))^+ - (Y_n - (C_\ell + 1))^+,$$

  we can get $\mathbb{E}^P[R_n^*]$ via Black-Scholes formula, for any $n = 1, \dots, N$.
- Moreover, we get

  $$\mathbb{E}^P[R_g] = \mathbb{E}^P[R_1^*] + \dots + \mathbb{E}^P[R_N^*].$$

Application in Python

- In

  `montecarlovariancereduction.controlvariates`

  you can find the code for the application of Control variates in the case of Cliquet option under the Black-Scholes model. We assume $T_k - T_{k-1}$ constant.
- In `cliquetOptionTest` we compare the classical Monte-Carlo approach, Monte-Carlo with Antithetic variables and Monte-Carlo with control variates on two aspects, for 30 tests with 10^4 simulations:
  - variance of the estimates
  - time (in seconds) needed for a single estimate.
- The results are shown in the following table.

	classical MC	MC with AV	MC with CV
variance	$3.94 \cdot 10^{-6}$	$1.32 \cdot 10^{-6}$	$4.79 \cdot 10^{-7}$
time	0.21	0.23	0.48

- You can see that Control variates effectively reduce the variance. However, as it is now, it is slower. Exercise: change the implementation (also of the class `CliquetOption` if needed) in order to make the Control variates application faster without loosing accuracy.

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

Motivation

The multi-period Binomial model for option pricing is widely used by practitioners in financial applications mainly because:

- It is very easy to understand and simulate.
- It is particularly convenient to price options involving a choice of the holder, like American and Bermudan options.
- It approximates the Black-Scholes model when the length of the periods tends to zero.
- Option pricing is not based on pure Monte-Carlo techniques but relies on weighting the payoff relative to any scenario by the (analytic!) probability of the scenario.

The setting

- Consider a multi-period model with times $t = 0, 1, \dots, T$, and consider a probability space $(\Omega, \mathcal{F}, \mathbb{F}, P)$, where $\mathbb{F} = (\mathcal{F}_t)_{t=0,\dots,T}$ is a filtration representing information.
- Suppose there exist:
  - A risk free asset defined by $S_t^0 = (1+\rho)^t$, $t = 0, \dots, T$, with a deterministic interest rate $\rho > 0$.
  - A risky asset adapted to $\mathbb{F}$ defined by

    $$S_t = S_0 \cdot Y_1 \cdot \dots \cdot Y_t, \quad t = 1, \dots, T,$$

    where Y_t can take the two values d, u with $0 < d < 1+\rho < u$, for any $t = 1, \dots, T$, and $(Y_t)_{t=1,\dots,T}$ are i.i.d. and such that Y_{t+1} is independent of $\mathcal{F}_t$.
- Then it holds

  $$S_t^0 = S_{t-1}^0(1+\rho), \quad t = 1, \dots, T$$

  and

  $$S_t = S_{t-1}Y_t, \quad t = 1, \dots, T.$$

Admissible strategies

At every time $t = 0, \dots, T-1$, an investor can construct a portfolio of value V_t, trading on the risk-free asset S^0 and on the risky asset S.

- The value of the portfolio is given by

$$V_t = \alpha_t S_t + \beta_t S_t^0, \quad t = 1, \dots, T,$$

  where $(\alpha_t)_{t=1,\dots,T}$ and $(\beta_t)_{t=1,\dots,T}$ are $\mathbb{F}$-predictable, discrete processes.
- The strategy (α, β) must be self-financing: it must hold

$$V_t = \alpha_t S_t + \beta_t S_t^0 = \alpha_{t+1} S_t + \beta_{t+1} S_t^0, \quad t = 1, \dots, T.$$

Arbitrage theory

Definition

A portfolio V is an arbitrage if:

- V is obtained by a self-financing strategy;
- $P(V_0 = 0) = 1$;
- $P(V_t \geq 0) = 1$ and $P(V_t > 0) > 0$ for some t.

Proposition

The market is arbitrage free only if $d < 1 + \rho < u$.

- Suppose $1 + \rho \leq d < u$, and consider the self-financing portfolio defined by

  $$V_t = S_t - \frac{S_0}{S_0^0} S_t^0, \quad t = 0, 1, \ldots, T.$$

  Then we have $V_0 = 0$ and

  $$V_1 = S_1 - \frac{S_0}{S_0^0} S_1^0 \geq S_0 d - S_0(1 + \rho) > 0.$$

- If $d < u \leq 1 + \rho$, changing the signs to the strategy above leads to an arbitrage.

Equivalent martingale measure

In order for the market to be arbitrage-free and complete, there must exist a unique measure $Q \sim P$ such that $\frac{S}{S^0}$ is a martingale, i.e., such that

$$\mathbb{E}^Q \left[\frac{S_{t+1}}{S^0_{t+1}} \middle| \mathcal{F}_t \right] = \frac{S_t}{S^0_t}, \quad t = 0, \dots, T-1. \tag{2}$$

Note that the measure Q is identified by the probability $q := Q(Y_t = u)$.
Since

$$\mathbb{E}^Q \left[\frac{S_{t+1}}{S^0_{t+1}} \middle| \mathcal{F}_t \right] = \frac{(qu + (1-q)d)S_t}{S^0_t(1+\rho)}, \quad t = 0, \dots, T-1,$$

equation (2) holds if and only if $qu + (1-q)d = 1 + \rho$, that is,

$$q = \frac{1 + \rho - d}{u - d}.$$

Such Q exists and is unique as we have supposed $0 < d < 1 + \rho < u$, and

$$\frac{dQ}{dP}(\omega) = \left(\frac{q}{p} \right)^{n(\omega)} \left(\frac{1-q}{1-p} \right)^{T-n(\omega)},$$

where $p := P(Y_t = u)$ and $n(\omega)$ is the number of times $t = 1, \dots, T$ when $Y_t(\omega) = u$.

Replicating strategy

- Assume we want to find an admissible strategy (α_t, β_t), $t = 1, \dots, T$, such that the value of the portfolio
$$\alpha_t S_t + \beta_t (1+\rho)^t$$
equals the value V_t of an option at every time $t = 1, \dots, T$.
- From now on, fix $t = 1, \dots, T$, and suppose we know S_{t-1}.
- Call V_t^u the value of the option at time t when $Y_t = u$ and V_t^d the value of the option at time t when $Y_t = d$.
- It must hold
$$\begin{cases} \alpha_t u S_{t-1} + \beta_t (1+\rho)^t = V_t^u, \\ \alpha_t d S_{t-1} + \beta_t (1+\rho)^t = V_t^d. \end{cases}$$
- The solution to the system above is
$$\alpha_t = \frac{V_t^u - V_t^d}{S_{t-1}(u-d)},$$
$$\beta_t = \frac{u V_t^d - d V_t^u}{(1+\rho)^t (u-d)}.$$

and gives the right replicating strategy.

Option pricing from admissibility

- Remember that our strategy (α_t, β_t), $t = 1, \dots, T$, has to be admissible!
- This means that we must have that

$$\begin{aligned}
V_{t-1} &= \alpha_{t-1} S_{t-1} + \beta_{t-1}(1+\rho)^{t-1} \\
&= \alpha_t S_{t-1} + \beta_t (1+\rho)^{t-1} \\
&= \frac{V_t^u - V_t^d}{u-d} + \frac{uV_t^d - dV_t^u}{(1+\rho)(u-d)} \\
&= \frac{(1+\rho)(V_t^u - V_t^d) + uV_t^d - dV_t^u}{(1+\rho)(u-d)} \\
&= \frac{(1+\rho-d)V_t^u + (u-1-\rho)V_t^d}{(1+\rho)(u-d)} \\
&= \frac{qV_t^u + (1-q)V_t^d}{1+\rho} \\
&= \frac{1}{1+\rho}\mathbb{E}^Q[V_t | \mathcal{F}_{t-1}].
\end{aligned}$$

- Then we have that the value $(V_t)_{t=0,\dots,T}$ of the option is a martingale under Q.
- This gives us a pricing theorem.

Option pricing

Theorem

The value V_0 of a contingent claim with maturity T and payoff V_T depending on the realizations of S until time T, is given by

$$V_0 = \frac{1}{(1+\rho)^T}\mathbb{E}^Q[V_T].$$

Remark

Because of the theorem above, we always simulate our Binomial model under the risk neutral measure Q.

1. Monte-Carlo method for option pricing and variance reduction techniques
    - The Monte-Carlo method: motivation and a brief overview
    - Variance reduction techniques
        - Introduction
        - Antithetic variables
        - Control variates

2. Option pricing under the Binomial model
    - Motivation and setting
    - Simulation of the Binomial model
    - Calibration of the Binomial model
    - American options valuation

3. Option valuation via solution of SDEs and PDEs
    - The Feynman-Kac formula
    - Monte-Carlo simulation of continuous time stochastic processes
    - Finite differences methods for the numerical solution of PDEs
        - Introduction
        - Convergence of finite difference methods
        - Fiinite difference methods to price options on local volatility models

Motivation

- Our main goal here is to get the price of European and (most importantly) American options written on an underlying Binomial model.
- This valuation will approximate the price of the options written on an underlying log-normal model.
- We then simulate the realizations of the underlying model in Python, and get the payoff on the realizations, along with its expectation.
- Remember we have to price under the risk neutral measure Q: then we simulate the realizations of the process under Q.
- The most naive way we can imagine to do this is a brute force Monte-Carlo approximation..

Monte-Carlo method

- Imagine we want to valuate the discounted price of an European option with a given payoff function $f : \mathbb{R} \to \mathbb{R}$, written on the process S, with maturity T.
- Suppose we don't know any analytic formula in order to derive the price as

$$V_0 = \frac{1}{(1+\rho)^T}\mathbb{E}^Q[f(S_T)].$$

- We consider N *states of the world* $\omega_1, \omega_2, \dots, \omega_N \in \Omega$.
- To any $\omega_1, \omega_2, \dots, \omega_N$, we associate a given trajectory of the process $(S_t)_{t=0,\dots,T}$, with dynamics given under the measure Q.
- In particular, we suppose that the trajectories $(S_t(\omega_k))_{t=0,\dots,T}$, $k = 1, 2, \dots, N$ are independent of each other.
- Strong law of large numbers:

$$\frac{1}{n}\sum_{k=1}^{n} f(S_T(\omega_k)) \to \mathbb{E}^Q[f(S_T)] \quad \text{a.s., when } n \to \infty.$$

- The idea is to simulate such trajectories and approximate

$$\mathbb{E}^Q[f(S_T)] \approx \frac{1}{N}\sum_{k=1}^{N} f(S_T(\omega_k)).$$

Monte-Carlo method for the Binomial model

- Our first goal is then to generate a sequence of random numbers in order to simulate N independent trajectories $(S_t(\omega_k))_{t=0,\dots,T}$, $k = 1, 2, \dots, N$ of S under the risk neutral measure Q, and store them in a $(T+1) \times N$ matrix (this can be useful for path dependent options).
- First issue: it is not possible to generate a sequence of perfectly random numbers, the best we can get is a sequence of *pseudo*-random numbers.
- Idea: generate (with the help of Python in our case) a sequence of $T \cdot N$ uniformly distributed, pseudo-random numbers $0 < x_{i,j} < 1$, $i = 1, \dots, T$, $j = 1, \dots, N$.
- Fix $\rho > 0$, $u > 1 + \rho$, $d < 1$, $q = \frac{1+\rho-d}{u-d}$.
- For every $i = 1, \dots, T$, $j = 1, \dots, N$, define

$$Y_i(\omega_j) = \begin{cases} u & \text{if } x_{i,j} < q \\ d & \text{if } x_{i,j} \geq q \end{cases}$$

and

$$S_{i+1}(\omega_j) = Y_i(\omega_j) S_i(\omega_j).$$

Implementation in Python

- You can find the code relative to the simulation of the Binomial model with the pure Monte-Carlo approach described above in

  `binomialmodel.creationandcalibration.binomialModelMonteCarlo`

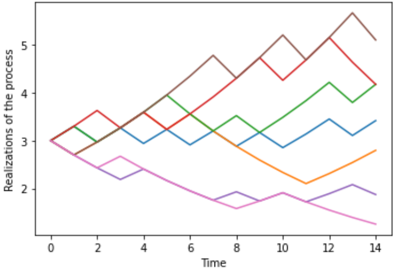
- Note that the class you find there extends the one in

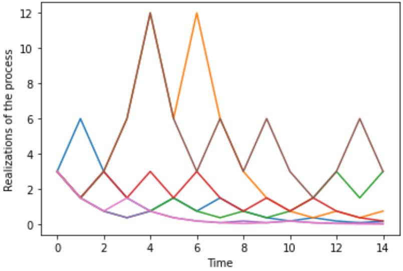
  `binomialmodel.creationandcalibration.binomialModel`.

- This is done in order to implement in the parent class some methods that do not strictly depend on the way in which we simulate the process.
- In this way, we don't have to copy and paste these methods in every class where we simulate the model in some way: object oriented programming feature.

Some paths

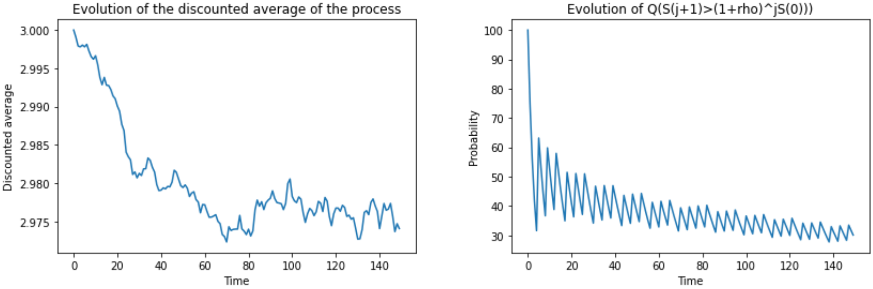
- We plot below some paths of the Binomial model.
- In the figure at the left we take $S_0 = 3$, $u = 1.1$, $d = 0.9$, $r = 0.05$, $T = 150$, having then $q = \frac{1+\rho-d}{u-d} = 0.75$.
- On the right, $S_0 = 3$, $u = 2$, $d = 0.5$, $r = 0.1$, $T = 150$, $q = \frac{1+\rho-d}{u-d} = 0.4$.





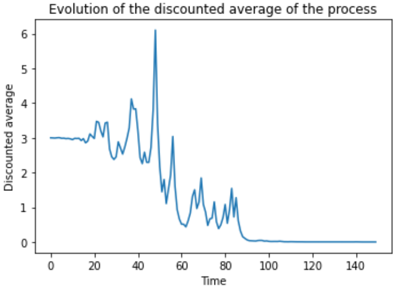
A first test

We show here the evolution of the discounted average of the process and of the probability $Q(S_{t_j} > (1+\rho)^{t_j} S_0)$, computed by using the Monte-Carlo method with 10^5 simulations, for $S_0 = 3$, $u = 1.1$, $d = 0.9$, $r = 0.05$, $T = 150$. In this case, we have $q = \frac{1+\rho-d}{u-d} = 0.75$.

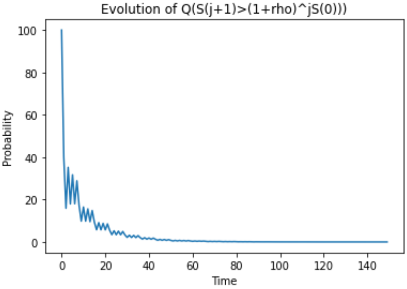


But something can go wrong..

Look at the evolution of the same quantities, again computed by using the Monte-Carlo method, choosing now $S_0 = 3$, $u = 2$, $d = 0.5$, $r = 0.1$, $T = 150$, $q = \frac{1+\rho-d}{u-d} = 0.4$.



Evolution of the discounted average of the process

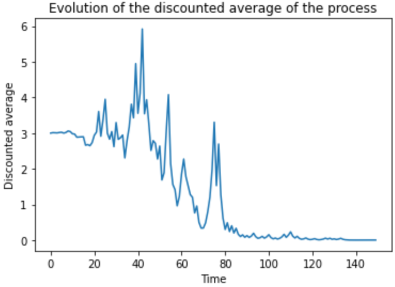


Evolution of Q(S(j+1)>(1+rho)^jS(0)))

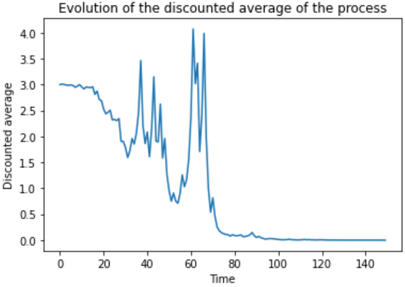
Why is the estimate of the average that inaccurate?

- With the parameters above, the anaytic average of the discounted process is equal to S_0, due to many realizations such that $S_{t_j} < (1+\rho)^{t_j} S_0$ and few, extremely high realizations.
- If you buy S at time $t = 0$, and you hold it for 150 time steps, you make a positive gain with a very low probability, but the gain can be extremely high.
- Problem: The approximated average is strongly impacted by whether or not those paths leading to high gains are simulated or not.

Let's choose two different seeds, for the same parameters



Evolution of the discounted average of the process



Evolution of the discounted average of the process

Maybe a pure Monte-Carlo approach is not the best solution..

- We have seen that, if the volatility is high, the Monte-Carlo approach can be very inaccurate for many time steps.
- Moreover, it is time consuming (this is a problem common to all brute-force Monte-Carlo approaches)
- Idea: let us exploit some analytic properties of the Binomial model..

Some simple observations

- At the n-th time step, $n+1$ realizations of the process are possible: $S_0u^n, S_0u^{n-1}d, \dots, S_0ud^{n-1}, S_0d^n$.
- The number of ups and downs is given by a Bernoulli distribution:

$$P(S_n = S_0u^kd^{n-k}) = \binom{n}{k} q^k(1-q)^{n-k}.$$

- Using the expression above, we can compute

$$\begin{aligned}\mathbb{E}^Q[f(S_n)] &= \sum_{k=0}^{n} Q(S_n = S_0u^kd^{n-k})f(S_0u^kd^{n-k}) \\ &= \sum_{k=0}^{n} \binom{n}{k} q^k(1-q)^{n-k} f(S_0u^kd^{n-k}).\end{aligned}$$

Implementation in Python

- The idea is then to generate all the possible realizations of the process up to a given time, and to weight them by their probability.
- You can find the code relative to this approach in

  `binomialmodel.creationandcalibration.binomialModelSmart`,

  whose class also extends the one in

  `binomialmodel.creationandcalibration.binomialModel`.
- Doing some tests in

  `binomialmodel.creationandcalibration.binomialModelSmartTest`.

  you can observe that, in this way, the average of the discounted process is stable.
- Moreover, this approach is of course much faster.

Computation of $Q(S_n > S_0(1+\rho)^n)$, $n = 1, \dots, T$

- Note that for any $k = 0, \dots, n$ it holds

$$\begin{aligned} S_n = S_0 u^k d^{n-k} > S_0(1+\rho)^n &\iff u^k d^{n-k} > (1+\rho)^n \\ &\iff \left(\frac{u}{d}\right)^k > \left(\frac{1+\rho}{d}\right)^n \\ &\iff k > n \log_{\frac{u}{d}} \left(\frac{1+\rho}{d}\right). \end{aligned}$$

- Then we have

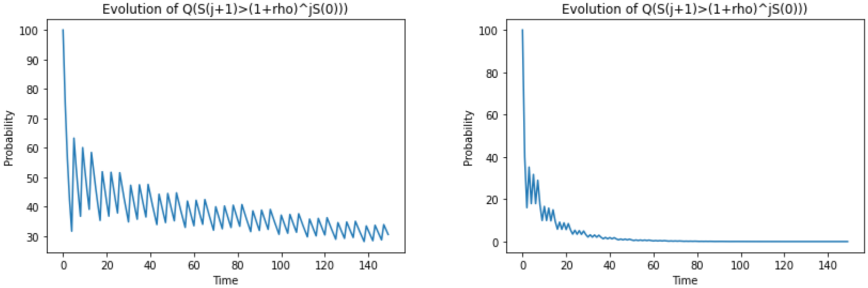
$$\begin{aligned} Q\left(S_n > S_0(1+\rho)^n\right) &= \sum_{k=\bar{k}}^{n} Q(S_n = S_0 u^k d^{n-k}) \\ &= \sum_{k=\bar{k}}^{n} \binom{n}{k} q^k (1-q)^{n-k}, \end{aligned}$$

where

$$\bar{k} = \min\left\{k \in \mathbb{N} : \; k > n \log_{\frac{u}{d}} \left(\frac{1+\rho}{d}\right)\right\} \leq n.$$

Evolution of the probability plotted with Python

We show here the evolution of the probability computed above, over 150 time steps. On the left, we have parameters $S_0 = 3$, $u = 1.1$, $d = 0.9$, $\rho = 0.1$, $q = \frac{1+\rho-d}{u-d} = 0.75$. On the right, $S_0 = 3$, $u = 2$, $d = 0.5$, $\rho = 0.05$, $q = \frac{1+\rho-d}{u-d} = 0.4$.



European option valuation in Python

- As seen before, an application of the simulation of the Binomial model in this way is the valuation of European options, under the pricing measure Q.
- In

  `binomialmodel.optionValuation.europeanOption,`

  you can see some methods relative to this.
- In particular, we compute the expectation of the payoff of European options as

$$\mathbb{E}^Q[f(S_n)] = \sum_{k=0}^{n} Q(S_n = S_0 u^k d^{n-k}) f(S_0 u^k d^{n-k})$$
$$= \sum_{k=0}^{n} \binom{n}{k} q^k (1-q)^{n-k} f(S_0 u^k d^{n-k}).$$

- We also compute the value of a general option for every time $t = 0, \dots, T-1$, and the corresponding self-financing, replicating strategy (α_t, β_t), $t = 0, \dots, T-1$, described before.
- As an exercise, you can check if the final value of the portfolio given by that strategy equals the payoff, for an option of your choice.

1. Monte-Carlo method for option pricing and variance reduction techniques
   - The Monte-Carlo method: motivation and a brief overview
   - Variance reduction techniques
     - Introduction
     - Antithetic variables
     - Control variates

2. Option pricing under the Binomial model
   - Motivation and setting
   - Simulation of the Binomial model
   - Calibration of the Binomial model
   - American options valuation

3. Option valuation via solution of SDEs and PDEs
   - The Feynman-Kac formula
   - Monte-Carlo simulation of continuous time stochastic processes
   - Finite differences methods for the numerical solution of PDEs
     - Introduction
     - Convergence of finite difference methods
     - Fiinite difference methods to price options on local volatility models

Calibration of the parameters u and d

- Recall that we have

$$S_t = S_0 \cdot Y_1 \cdot \dots \cdot Y_t, \quad t = 1, \dots, T,$$

where

$$Y_t = \begin{cases} u > 1+\rho \text{ with (risk-neutral) probability } q = \frac{1+\rho-d}{u-d} \\ d < 1 \text{ with probability } 1-q \end{cases}, \quad t = 1, \dots, T.$$

- Our goal is to calibrate the up and downs parameters u and d, supposing we know the risk neutral probability $q = \frac{1+\rho-d}{u-d}$ and the interest rate $\rho > 0$, and that we can observe

$$\text{Var}[\log(S_T/S_0)] := \mathbb{E}^Q[\log(S_T/S_0)^2] - \mathbb{E}^Q[\log(S_T/S_0)]^2$$

for a given maturity T.

- Observe first that since

$$\log(S_T/S_0) = \sum_{t=1}^{T} \log(Y_t),$$

and since $(Y_t)_{t=1,\dots,T}$ are equi-distributed, we get

$$\text{Var}[\log(S_T/S_0)] = T\text{Var}[\log(Y_T)].$$

One result about variance

Proposition

Let

$$Y_t = \begin{cases} u > 1+\rho \text{ with (risk-neutral) probability } q = \frac{1+\rho-d}{u-d} \\ d < 1 \text{ with probability } 1-q, \end{cases} \quad , \quad t = 1, \dots, T.$$

Then for any $t = 1, \dots, T$ we have

$$\text{Var}[\log Y_t] = q(1-q)\log(u/d)^2.$$

Proof

$$\begin{aligned}
\text{Var}[\log Y_t] &= \mathbb{E}^Q[\log(Y_t)^2] - \mathbb{E}^Q[\log(Y_t)]^2 \\
&= \mathbb{E}^Q[\log(Y_t)^2] - (q\log(u) + (1-q)\log(d))^2 \\
&= q\log(u)^2 + (1-q)\log(d)^2 \\
&\quad - q^2\log(u)^2 - (1-q)^2\log(d)^2 - 2q(1-q)\log(u)\log(d) \\
&= q(1-q)\log(u)^2 + q(1-q)\log(d)^2 - 2q(1-q)\log(u)\log(d) \\
&= q(1-q)\left(\log(u) - \log(d)\right)^2 \\
&= q(1-q)\log(u/d)^2.
\end{aligned}$$

Calibration with Python

- Thanks to

  $$q = \frac{1+\rho-d}{u-d}$$

  and to

  $$\text{Var}[\log Y_t] = q(1-q)\log(u/d)^2,$$

  along with

  $$\sigma_{obs}^2 := \text{Var}[\log(S_T/S_0)] = T\text{Var}[\log(Y_T)],$$

  we can get u and d from q, ρ and σ_{obs}^2 by solving the nonlinear system

  $$\begin{cases} \frac{1+\rho-d}{u-d} = q \\ \log(u/d)^2 = \frac{\sigma_{obs}^2}{Tq(1-q)} \end{cases} \tag{3}$$

- We can find an approximated solution of (3) by the `fsolve` function of Python.
- Look at

  `binomialmodel.creationandcalibration.binomialModelCalibration`

  to see an implementation of the calibration of u and d as showed above.

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

American options

- The holder of an American option with payoff f and maturity T on an underlying X has the right, at any time $t \in [0, T]$, to hold the contract or to exercise the payoff $f(X_T)$.
- The valuation of American options is more complicated than the one of European options, since it involves an optimal exercise problem.
- In order to valuate such an option at time t, indeed, the conditional expectation at time t of the future value of the option has to be computed, and then compared against the present value of the payoff.
- However, the Monte-Carlo computation of a conditional expectation is very time consuming.
- One of the strengths of the Binomial model with respect to other settings is that it permits a favourable pricing of American options.
- Also when dealing with continuous time processes, with suitable dynamics, one may approximate them with a Binomial model in order to get the price.

American options valuation under the Binomial model

- At any time $t = 1, \dots, T$, call $S_t(k)$ and $V_t(k)$ the value of the underlying and of the option, respectively, in the scenario with k ups and $t-k$ downs up to time t.
- Idea: proceed backward.
- First we compute the payoff $f(S_T(k)) = f(S_0 u^k d^{T-k})$, for any $k = 0, \dots, T$.
- We have of course $V_T(k) = f(S_T(k))$, for any $k = 0, \dots, T$.
- At time $T-1$, for any $k = 0, \dots, T-1$ we compute

$$\begin{aligned} V_{T-1}(k) &= \max\left(f(S_{T-1}(k)), \frac{1}{1+\rho}\left(qV_T(k+1) + (1-q)V_T(k)\right)\right) \\ &= \max\left(f(S_0 u^k d^{T-1-k}), \frac{1}{1+\rho}\left(qV_T(k+1) + (1-q)V_T(k)\right)\right). \end{aligned}$$

- For any $t = 1, \dots, T-2$ we compute with the same argument

$$V_t(k) = \max\left(f(S_0 u^k d^{t-k}), \frac{1}{1+\rho}\left(qV_t(k+1) + (1-q)V_t(k)\right)\right).$$

- We finally get the value of the option at initial time as

$$V_0 = \max\left(f(S_0), \frac{1}{1+\rho}\left(qV_1(1) + (1-q)V_1(0)\right)\right).$$

Implementation in Python

You can find the code relative to the the valuation of American options in

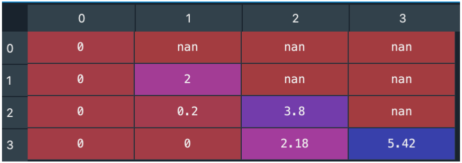
`binomialmodel.optionValuation.AmericanOption`,

with some tests in

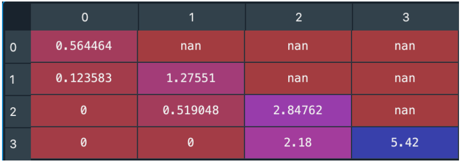
`binomialmodel.optionValuation.AmericanOptionTest`.

Example

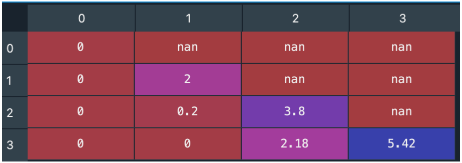
We consider a put option with payoff $f(x) = (20 - x)^+$, and choose parameters $T = 3$, $S_0 = 20$, $u = 1.1$, $d = 0.9$, $\rho = 0.05$.
The triangular matrices below show us an analysis of the American put option for such parameters (row 3 shows the values for $t = 3$ and so on).
The upper left and upper right matrices show the amount one would get if exercising the option or holding the contract, respectively; the lower left one the values of the option; the lower right one has 1 in the exercise region and 0 in the hold region



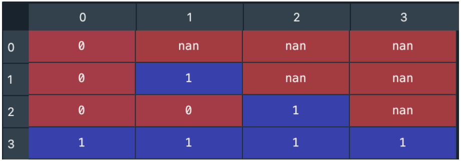
	0	1	2	3
0	0	nan	nan	nan
1	0	2	nan	nan
2	0	0.2	3.8	nan
3	0	0	2.18	5.42



	0	1	2	3
0	0.564464	nan	nan	nan
1	0.123583	1.27551	nan	nan
2	0	0.519048	2.84762	nan
3	0	0	2.18	5.42



	0	1	2	3
0	0	nan	nan	nan
1	0	2	nan	nan
2	0	0.2	3.8	nan
3	0	0	2.18	5.42



	0	1	2	3
0	0	nan	nan	nan
1	0	1	nan	nan
2	0	0	1	nan
3	1	1	1	1

Approximating a Black-Scholes model with a Binomial model

- Consider a continuous, adapted stochastic process $X=(X_t)_{t\geq 0}$ with dynamics

$$dX_t = rX_t dt + \sigma X_t dW_t, \quad t \geq 0,$$

where $r,\sigma > 0$ and $W=(W_t)_{t\geq 0}$ is a Brownian motion.

- Suppose you want to price an American option with underlying X and maturity $T>0$.
- It can be seen that the dynamics of $X=(X_t)_{0\leq t\leq T}$ can be approximated by N time steps of a Binomial model with parameters

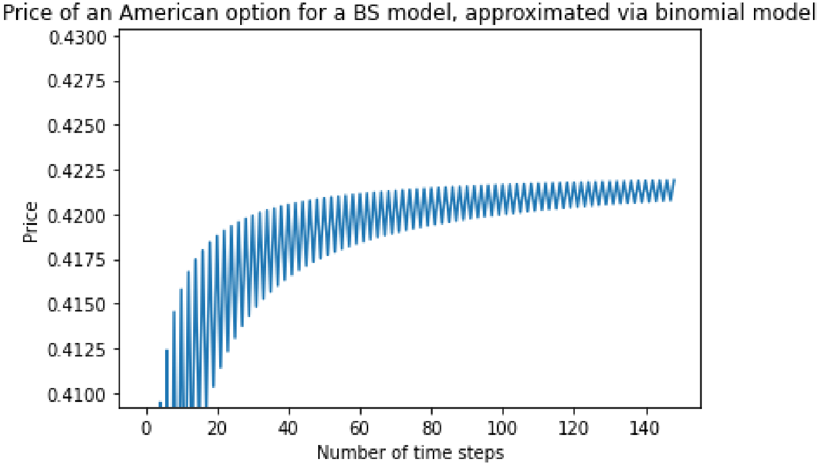
$$u = e^{\sigma\sqrt{T/N}}, \qquad d = 1/u, \qquad \rho = e^{r\sqrt{T/N}}, \tag{4}$$

for N large enough, see for example A. A. Dar, and N. Anuradha, *Comparison: binomial model and Black Scholes model.* Quantitative finance and Economics 2.1 (2018): 230-245.

- The idea is to approximate the price of the American option of maturity T with the price of an American option with maturity N written a Binomial model with parameters as in (4), for N large enough.
- Indeed, the price of the American option written on the Binomial model can be found as illustrated before.

Example: not such a nice behaviour

We consider an American put option with payoff $f(x) = (1 - x)^+$ and maturity $T = 3$, written on a Black-Scholes model with parameters $r = 0.02$, $\sigma = 0.7$.
The plot below shows the approximated price via the derivation under the Binomial model, for an increasing number of times steps up to $N = 150$.



Control variates for American call and put options

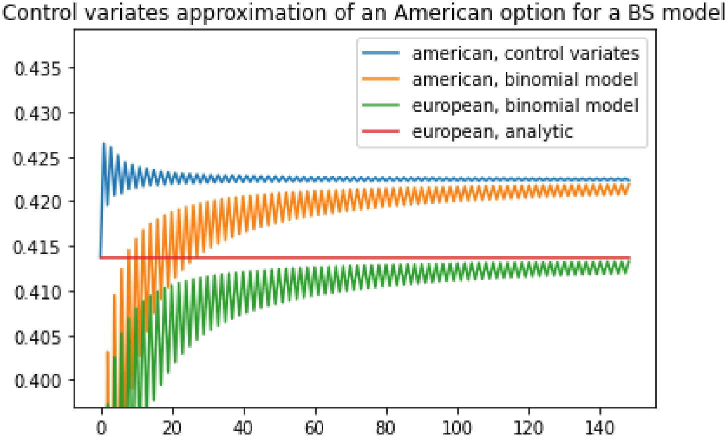
- First idea: we know the analytic price of an European put (or call) option under the Black-Scholes model. For example, call P^E the Black-Scholes formula price of an European put option.
- Also call:
  - P_N^E the price of an European put approximated by the Binomial model with N time steps;
  - P^A the analytic price of an American put;
  - P_N^A the price of an American put approximated by the Binomial model with N time steps.
- Second idea: we know the euristics $P^A - P_N^A \approx P^E - P_N^E$.
- We then approximate

$$P^A \approx P_N^A + (P^E - P_N^E)$$

- This approximates the price of an American put option via control variates.
- Same thing for a call option.

A nicer behaviour with control variates

We consider again an American put option with payoff $f(x) = (1-x)^+$ and maturity $T = 3$, written on a Black-Scholes model with parameters $r = 0.02$, $\sigma = 0.7$: same situation as before.
The plot below compares the prices introduced in the previous slide, for an increasing number of times steps up to $N = 150$.



Implementation in Python

- You can find some experiments relative to the stability of approximations of prices of American options with the Binomial model in

  `binomialmodel.optionValuation.AmericanOptionPriceConvergence`,
- The code performing the control variates approach can be found in

  `binomialmodel.optionValuation.controlVariates`,

  with some tests in

  `binomialmodel.optionValuation.controlVariatesTest`.

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

Motivation

- In this section we examine the deep connection between Stochastic Differential Equations (SDEs) and Partial Differential Equations (PDEs) provided by the Feynman-Kac formula.
- In particular, such a result gives a representation of the solution u of a PDE with a final time condition in terms of expectation of the final time condition applied to the solution of an associated SDE.
- This has of course applications in option pricing: on one hand, one can price an option by (numerically) solving a PDE. On the other hand, one can solve a PDE by Monte-Carlo methods.

The setting

- Fix a filtered probability space $(\Omega, \mathcal{F}, P, \mathbb{F})$.
- Consider the SDE in $\mathbb{R}^n$ given by

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t, \quad t \geq 0, \tag{5}$$

  denote by D a bounded domain in $\mathbb{R}^n$ and assume that:
    - the coefficients of the SDE are locally bounded;
    - for every $t \geq 0$ and $x \in D$ there exists a solution $X^{t,x}$ of (5) such that $X_t^{t,x} = x$, relative to a Brownian motion W on the space $(\Omega, \mathcal{F}, P, \mathbb{F})$.
- Define the operator $\mathcal{A}$ associated to (5) by

$$\mathcal{A}_t u(t,x) := \frac{1}{2} \sum_{i,j=1}^{N} c_{ij}(t,x) \partial_{x_i x_j} u(t,x) + \sum_{j=1}^{N} b_j(t,x) \partial_{x_j} u(t,x), \tag{6}$$

  where c is the $N \times N$ matrix $c := \sigma\sigma^T$.
- Fix $T > 0$ and consider the classical Cauchy-Dirichlet problem

$$\begin{cases} \mathcal{A}u - au + \partial_t u = 0 & \text{in } Q \\ u = \varphi & \text{in } \partial_p Q, \end{cases} \tag{7}$$

  where $a, \varphi : \mathbb{R}^N \to \mathbb{R}$ are given functions,

$$Q := (0, T) \times D$$

  and

$$\partial_p Q = \partial Q \setminus (\{0 \times D\}).$$

Feynman-Kac formula

Theorem: Feynman-Kac formula

Let $\varphi \in C(\partial_p Q)$ and $a \in C(Q)$, such that $a_0 := \inf a$ is finite. If $u \in C^2(Q) \cap C(\bar{Q})$ is a solution of problem (7) then, for every $(t, x) \in Q$, we have

$$u(t,x) = \mathbb{E}^P \left[e^{-\int_t^{\tau \wedge T} a(s,X_s)ds} \varphi(\tau \wedge T, X_{\tau \wedge T}) \right],$$

where $X^{t,x}$ is a solution of (5) such that $X_t^{t,x} = x$ and τ is the exit time of such a process from the domain D.

Applications

- The Feynman-Kac formula allows us to valuate options with payoff φ on an underlying driven by (5) by numerically solving (7).
- In the next slides, we will then consider the valuation of such payoffs both with Monte-Carlo methods to approximate a solution to (5) and with finite difference methods to get an approximate solution to (7).
- We compare the effectiveness of these approaches both in terms of time and accuracy coding in Python.
- In particular, we are interested in local volatility models, i.e., processes with dynamics

  $$dX_t = rX_t dt + X_t \sigma(t, X_t) dW_t, \quad t \geq 0,$$

  where $r > 0$ is the risk-free rate.

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

Motivation

- We first want to consider option pricing approximation via the Monte-Carlo approximation and discretization of continuous time stochastic processes. We will then move to PDE methods.
- We have already seen some applications when we dealt with continuous time stochastic processes, see for example the valuation of Cliquet option under the Black-Scholes model.
- For the Black-Scholes model things were relatively easy because we had an expression for the value itself of the process.
- However, under more general and complicated models, we don’t have the value of the process but we only know the SDE that the process solves.
- In this cases, we have to approximate the evolution of the process by numerically solving the SDE it satisfies.
- In order to do that, we first have to discretize the continuous time interval on which the process is defined.

Time discretization and discretization error

- Consider a stochastic Itô process $X = (X_t)_{0 \le t \le T}$, having dynamics

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t, \quad 0 \le t \le T. \tag{8}$$

- Our goal is to discretize the time interval $[0, T]$ by a time discretization

$$0 = t_0 < t_1 < \cdots < t_N = T,$$

  in such a way that $\max_k |t_k - t_{k-1}| \to 0$ when $N \to \infty$, and approximate $X = (X_t)_{0 \le t \le T}$ in $[0, T]$ with a discretized version $\hat{X} = (\hat{X}_t)_{t \in \{0, t_1, \dots, t_N\}}$.
- That is, we discretize the SDE in (8).
- Discretization error in t_k: $X_{t_k} - \hat{X}_{t_k}$, $k = 1, \dots N$.
- Do not confuse the discretization error with the Monte-Carlo error: the discretization error is only due to the fact that we are discretizing a continuous time SDE: we are approximating the continuous time SDE with a discrete time SDE.
- If μ and σ in (8) depend on the space variable x, then the discretization error can be different from zero and propagate over time.
- If μ and σ in (8) do not depend on the space variable x, we take the exact solution of the SDE as a discretization scheme and we have zero discretization error.

Strong convergence

Definition

Call $\hat{X}^{(n)}$ the discretization of X via a time discretization with n times.

1. We say that $\hat{X}^{(n)}$ converges strongly to X, if and only if

$$\lim_{n\to\infty}\left(\sup_{0\le t\le T}|\hat{X}_t^{(n)}-X_t|\right)=0.$$

2. We say that $\hat{X}^{(n)}$ converges with strong order $\gamma>0$ to X, if and only if there exists $C>0$ such that

$$\mathbb{E}\left(\sup_{0\le t\le T}|\hat{X}_t^{(n)}-X_t|\right)\le Ch_n^{\gamma}$$

where

$$h_n=\max\{t_k^{(n)}-t_{k-1}^{(n)}|k=1,\dots,n\}.$$

Weak convergence

Definition

Call $\hat{X}^{(n)}$ the discretization of X via a time discretization with n times.

1. We say that $\hat{X}^{(n)}$ converges weakly to X, if and only if for fixed $t \in [0, T]$ and any Lipschitz-continuous function f we have
$$\lim_{n \to \infty} \mathbb{E}[f(\hat{X}_t^{(n)})] - \mathbb{E}[f(X_t)] = 0.$$
2. We say that $\hat{X}^{(n)}$ converges with weak order $\gamma > 0$ to X, if and only if for fixed $t \in [0, T]$ and any Lipschitz-continuous function f there exists $C > 0$ such that
$$|\mathbb{E}[f(\hat{X}_t^{(n)})] - \mathbb{E}[f(X_t)]| \leq C h_n^{\gamma}$$
where
$$h_n = \max\{t_k^{(n)} - t_{k-1}^{(n)} | k = 1, \ldots, n\}.$$

Euler-Maruyama scheme

Definition

Consider a stochastic Itô process $X = (X_t)_{0 \leq t \leq T}$, having dynamics

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t, \quad 0 \leq t \leq T,$$

and a time discretization

$$0 = t_0 < t_1 < \cdots < t_N = T.$$

The time-discrete stochastic process $\hat{X}$ defined by

$$d\hat{X}_{t_{k+1}} = \mu(t_k, \hat{X}_{t_k})\Delta t_k + \sigma(t_k, \hat{X}_{t_k})\Delta W_{t_k}, \quad k = 0, \dots, N-1,$$

where $\Delta t_k := t_{k+1} - t_k$ and $\Delta W_{t_k} := W_{t_{k+1}} - W_{t_k}$, is called a Euler-Maruyama scheme of the process X.

Proposition

The Euler-Maruyama scheme converges to X with strong order $\frac{1}{2}$ and weak order 1.

Milstein scheme

Definition

Consider a stochastic Itô process $X = (X_t)_{0 \le t \le T}$, having dynamics

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t, \quad 0 \le t \le T,$$

and a time discretization

$$0 = t_0 < t_1 < \cdots < t_N = T.$$

The time-discrete stochastic process $\hat{X}$ defined by

$$d\hat{X}_{t_{k+1}} = \mu(t_k, \hat{X}_{t_k})\Delta t_k + \sigma(t_k, \hat{X}_{t_k})\Delta W_{t_k} + \frac{1}{2}\sigma(t_k, \hat{X}_{t_k})\partial_X \sigma(t_k, \hat{X}_{t_k})(\Delta W_{t_k} - \Delta t_k)^2,$$

$k = 0, \dots, N-1$, where $\Delta t_k := t_{k+1} - t_k$ and $\Delta W_{t_k} := W_{t_{k+1}} - W_{t_k}$, is called a Milstein scheme of the process X.

Remark

If the function σ depends on the spatial variable x, the Milstein scheme provides an improvement of the approximation of the stochastic integral $\int \sigma dW$.

Example: discretization of a (particular) log-normal process

Consider the process X following the dynamics

$$dX_t = \mu(t, X_t) X_t dt + \sigma(t) dW_t, \quad 0 \le t \le T.$$

- Directly discretizing X with an Euler-Maruyama or with a Milstein scheme, the discretization error might be relatively large: note for example that $\hat{X}_{t_1}$ may attain negative values and is normal distributed, whereas $X_{t_1} > 0$ and is log-normally distributed.
- By discretizing instead $\log(X)$, and then transforming it back via the exponential, one would obtain a log-normal random variable $\hat{X}_{t_1}$.
- When the function μ only depends on time, one can even simulate the exact solution of the SDE, and have zero discretization error.

Implementation in Python

- In the package

  `processSimulation`

  you can find some code for the discretization and simulation of continuous time Itô stochastic processes and some relative experiments.
- In particular, the class in

  `abstractProcessSimulation`

  is an abstract class that provides the implementation of a general scheme, depending on methods providing the drift and the (possibly corrected) diffusion, that are supposed to be implemented in the derived classes.
- The class has also two attributes `transform` and `inverseTransform` that account for the fact that one could simulate a convenient transformation f of the process (this is the case for example with log-normal processes) and return f^{-1}.
- We extend such a class by two derived classes, you can find in

  `eulerDiscretizationForBlackScholes`

  and

  `standardEulerDiscretization`,

  respectively. The first one simulates the Black-Scholes model by discretizing and simulating the logarithm, whereas the second one provides the Euler-Maruyama scheme for a general Itô process.

Tests in Python

- In `testBlackScholesSimulation` we test the two classes above in the valuation of an European call option.
- In `testKnockOutOption` we test the Monte-Carlo discretization and simulation of a Black-Scholes model looking at the valuation of a down-and-out call option. This is an option that gives the usual payoff of a call option with strike K but only conditional to the fact that the value of the underlying S remains within the interval (B_d, ∞) for all times until maturity [1].

  The analytic price of such an option under the Black-Scholes model is computed by the function

  `blackScholesDownAndOut`

  in

  `analyticFormulas.analyticFormulas`,

  In such a test, one can note that a fine time discretization is needed.

---

[1] Up-and-out options give instead the usual payoff conditional to the fact that the underlying S remains within the interval $(0, B_u)$, with $B_u > K$, for all times until maturity.

Additional references

- This has been a very short introduction about the discretization and the simulation of continuous time stochastic processes.
- If you want to go deeper, these are some references you can consult:
  - C. Fries. *Mathematical Finance: Theory, Modeling, Implementation*. John Wiley & Sons, 2007
  - P. Kloeden, E. Platen, *Numerical Solution of Stochastic Differential Equations*, Applications of Mathematics. Stochastic Modelling and Applied Probability, Vol. 23, Springer, Berlin, 1999.
  - T. Sauer, *Numerical solution of stochastic differential equations in finance.*, Handbook of computational finance. Springer, Berlin, Heidelberg, 2012. 529-550.

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
    - Introduction
    - Antithetic variables
    - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- **Finite differences methods for the numerical solution of PDEs**
    - Introduction
    - Convergence of finite difference methods
    - Fiinite difference methods to price options on local volatility models

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

Motivation

- Let $X = (X_t)_{0 \le t \le T}$ be a one-dimensional process, following the dynamics
$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t, \quad 0 \le t \le T. \tag{9}$$
- The Feynman-Kac formula we have seen before tells us that, under some regularity and integrability conditions on b and σ, that are also needed to ensure existence and uniqueness of a solution to (9), the discounted expectation
$$e^{-r(T-t)}\mathbb{E}^P\left[\varphi(\tau \wedge T, X^{t,x}_{\tau \wedge T})\right],$$
where $X^{t,x}$ is the solution to (9) such that $X_t = x$, is given by the solution on (t, x) of the classical Cauchy-Dirichlet problem
$$\begin{cases} \partial_t u(t,x) + \frac{1}{2}\sigma^2(t,x)\partial_{xx}u(t,x) + b(t,x)\partial_x u(t,x) - ru(t,x) = 0 \quad \text{in } Q \\ u = \varphi \quad \text{in } \partial_p Q, \end{cases}$$
where
$$Q := (0, T) \times D,$$
D is an interval of $\mathbb{R}$ and
$$\partial_p Q = \partial Q \setminus (\{0 \times D\}).$$
- This gives us a way to compute (or approximate) the price of an option with payoff φ.

Finite difference methods

- We use finite differences methods.
- The domain $Q = (0, T) \times D$ is substituted with a grid of points, discretizing both the time and the space interval.
- The derivatives are approximated with finite differences (exploiting Taylor series).
- A numerical solution consisting of a set of values that approximate the solution of the problem at the grid of points is computed.
- In the following, we give some fundamental, general properties of finite difference methods.

1 Monte-Carlo method for option pricing and variance reduction techniques
- The Monte-Carlo method: motivation and a brief overview
- Variance reduction techniques
  - Introduction
  - Antithetic variables
  - Control variates

2 Option pricing under the Binomial model
- Motivation and setting
- Simulation of the Binomial model
- Calibration of the Binomial model
- American options valuation

3 Option valuation via solution of SDEs and PDEs
- The Feynman-Kac formula
- Monte-Carlo simulation of continuous time stochastic processes
- Finite differences methods for the numerical solution of PDEs
  - Introduction
  - Convergence of finite difference methods
  - Fiinite difference methods to price options on local volatility models

Some notations

- From now on, denote by N_x the number of intervals in the space discretization and by N_t the number of intervals in the time discretization of a given finite difference scheme.
- also denote by $N = N_t \times N_x$ the dimension of a finite difference scheme.
- Let u_N be the approximated solution of the exact solution u by a finite difference scheme of dimension N. Let also φ_N be the discretization of the border conditions φ of the Cauchy-Dirichlet problem.
- We then have a finite difference scheme $\mathcal{P}_N(u_N, \varphi_N)$, which is an approximation to the PDE $\mathcal{P}(u, \varphi)$.

Important definition

Definition

A finite difference scheme is said to be:

- Convergent under a suitable norm $\|\cdot\|$ if
$$\lim_{N\to\infty} \|u_N - u\| = 0.$$
- Consistent if
$$\lim_{N\to\infty} \mathcal{P}_N(u,\varphi) = 0.$$
  and strongly consistent if
$$\mathcal{P}_N(u,\varphi) = 0 \text{ for every } N.$$
- Stable under a suitable norm $\|\cdot\|$ if there exist constants $K > 0, \beta > 0$ such that
$$\|u^n\| \leq K e^{\beta t} \|u^0\|,$$
  for every $n = 1, \ldots, N_t$, where u^j denotes the approximated solution at time step j and $t = n\Delta_t$.
  Intuitively: stability requires that errors do not accumulate as the computation proceeds from one times step to the next.

A fundamental result

Lax Equivalence Theorem

A consistent finite difference scheme is convergent if and only if it is stable.

1. Monte-Carlo method for option pricing and variance reduction techniques
   - The Monte-Carlo method: motivation and a brief overview
   - Variance reduction techniques
     - Introduction
     - Antithetic variables
     - Control variates

2. Option pricing under the Binomial model
   - Motivation and setting
   - Simulation of the Binomial model
   - Calibration of the Binomial model
   - American options valuation

3. Option valuation via solution of SDEs and PDEs
   - The Feynman-Kac formula
   - Monte-Carlo simulation of continuous time stochastic processes
   - Finite differences methods for the numerical solution of PDEs
     - Introduction
     - Convergence of finite difference methods
     - Fiinite difference methods to price options on local volatility models

Our case

- We consider the log-normal, local volatility model

$$dX_t = rX_t dt + \sigma(t, X_t)X_t dW_t, \quad 0 \le t \le T, \tag{10}$$

and thus the Cauchy-Dirichlet problem

$$\begin{cases} \partial_t u(t,x) + \frac{1}{2}x^2\sigma^2(t,x)\partial_{xx}u(t,x) + rx\partial_x u(t,x) - ru(t,x) = 0 \quad \text{in } Q := (0,T) \times (a,b) \\ u = \psi_1 \quad \text{in } (0,T) \times \{a\}, \\ u = \psi_2 \quad \text{in } (0,T) \times \{b\}, \\ u = \varphi \quad \text{in } T \times (a,b), \end{cases} \tag{11}$$

$0 \le a < b$.

- If the payoff is only given at terminal time (note that a Knock-out option is instead defined also by a zero boundary condition) then we have to define some additional conditions at a and b. These are called artificial boundary conditions.
- We want to find an approximate solution to (11), under final conditions given by the payoff function and border conditions which can be artificial or specified by the option, by using finite difference methods.
- In particular, we consider the following schemes:
  - Explicit Euler
  - Implicit Euler
  - Crank-Nicolson
  - Upwind of Explicit Euler

Boundary conditions and time inversion

- Considering the time to maturity $T-t$ as the time variable allows us to transform the final condition given by the payoff into an initial condition.
- That is, we do a change of variables $t \to T-t$ and transform (11) into

$$\begin{cases} \partial_t u(t,x) - \frac{1}{2}x^2\sigma^2(t,x)\partial_{xx}u(t,x) - rx\partial_x u(t,x) + ru(t,x) = 0 & \text{in } (0,T)\times(a,b) \\ u = \psi_1 \quad \text{in } (0,T)\times\{a\}, \\ u = \psi_2 \quad \text{in } (0,T)\times\{b\}, \\ u = \varphi \quad \text{in } \{0\}\times(a,b). \end{cases} \tag{12}$$

- The boundary conditions at a and b depend on the option we want to valuate. For example:
  - For a knock-out option with barriers L and U, one can conveniently set $a = L$, $b = U$, $\psi_1 \equiv \psi_2 \equiv 0$.
  - For an European call option, one may consider that, since the value $C(S)$ of the option is always smaller or equal than the associated value S of the underlying, one has $C = 0$ at zero.
    Moreover, when S is large, the put option has value close to zero, and by the put-call parity this implies that $C(S) \approx S - Ke^{-r(T-t)}$.
    By this arguments, it makes sense to consider $a = 0$, $b > 0$ big enough and set $\psi_1 \equiv 0$, $\psi_2(t,b) = b - Ke^{-r(T-t)}$.

Explicit Euler

- Using the Explicit Euler scheme, we provide the following discretization scheme for the PDE in (12):

$$u_j^{n+1} = u_j^n + \frac{1}{2}x_j^2(\sigma_j^n)^2\frac{\Delta_t}{(\Delta_x)^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n) + rx_j\frac{\Delta_t}{2\Delta_x}(u_{j+1}^n - u_{j-1}^n) - r\Delta_t u_j^n,$$

  $j = 1, \dots, N_x - 1$, with $x_j = \Delta_x \cdot j$ and $\sigma_j^n := \sigma(\Delta_t \cdot n, x_j)$.

- Note that u_0^{n+1} and $u_{N_x}^{n+1}$ are determined by the border conditions.
- It can be shown that Explicit Euler is consistent.
- For a PDE with variable coefficients, it is in general not easy to provide a stability analysis.
- However, theory and practice suggest that:
  - in the most common applications (i.e., when the interest rate r is not too large) the first derivative term $rx\partial_x u$ and the linear term ru can be ignored in the analysis.
  - the stability conditions known for the constant coefficients PDE hold locally for every x.
- With this -quite heuristic- approach in mind, and applying standard stability analysis for the Explicit Euler scheme with constant coefficients, we impose the stability condition

$$\Delta_t \leq \frac{(\Delta_x)^2}{\sigma(T,b)^2b^2},$$

  supposing σ to be non-decreasing both with respect to time and space, as it is the case for most applications.

Implicit Euler

- Problem: the constant of the stability condition depends on Δ_x.
- This imposes a very fine discretization in the time variable, since we have to choose a right boundary b big enough.
- This might considerably affect the performance of the method, both in terms of resources allocated and time.
- A classic solution to this issue is to consider instead an *implicit* Euler scheme, which provides the following discretization scheme for the PDE in (12):

$$\begin{aligned} u_j^{n+1} = u_j^n + \frac{1}{2} x_j^2 (\sigma_j^{n+1})^2 \frac{\Delta_t}{(\Delta_x)^2} (u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}) \\ + r x_j \frac{\Delta_t}{2\Delta_x} (u_{j+1}^{n+1} - u_{j-1}^{n+1}) - r\Delta_t t u_j^{n+1}. \end{aligned}$$

- In order to obtain the solution at time step $n+1$ from the one at n, one has to solve a no-trivial system of linear equations with $N_x - 1$ unknowns. This is why the scheme is called implicit.

Implicit Euler

The solution at time step $n+1$ is obtained from the solution at time step n by solving the system of linear equations

$$(I + \Delta_t A^{(n+1)})U^{n+1} = U^n + V^n,$$

where:

- $U^n = (u_1^n, u_2^n, \ldots, u_{N_x-1}^n)$ and $U^{n+1} = (u_1^{n+1}, u_2^{n+1}, \ldots, u_{N_x-1}^{n+1})$
- A^n is the $(N_x - 1) \times (N_x - 1)$ tridiagonal matrix defined by

$$\begin{aligned} A^n_{j,j} &= \frac{1}{(\Delta_x)^2}(\sigma_j^n)^2 + r, \quad j = 1, \ldots, N_x - 1 \\ A^n_{j,j+1} &= -\frac{1}{2(\Delta_x)^2}(\sigma_j^n)^2 - r\frac{1}{2\Delta_x}x_j, \quad j = 0, \ldots, N_x - 2 \\ A^n_{j,j+1} &= -\frac{1}{2(\Delta_x)^2}(\sigma_j^n)^2 + r\frac{1}{2\Delta_x}x_j, \quad j = 1, \ldots, N_x - 1 \end{aligned}$$

- $V^n = (v_a, 0, \ldots, 0, v_b)^T$ with

$$v_a = \psi_1(a)\left[\frac{1}{2(\Delta_x)^2}(\sigma_1^n)^2 + r\frac{1}{2\Delta_x}x_1\right]$$

  and

$$v_b = \psi_2(b)\left[\frac{1}{2(\Delta_x)^2}(\sigma_{N_x-1}^n)^2 + r\frac{1}{2\Delta_x}x_{N_x-1}\right].$$

Convergence analysis for Implicit Euler

Proposition

There exists a constant C_1, not depending on Δ_x, such that if $\Delta_t \leq C_1$ the matrix $(I + \Delta_t A^{(n+1)})$ is invertible for all $n = 0, \ldots, N_t$, so that it is possible to solve the system and use the Implicit Euler scheme.

Proposition

Under this condition, the Implicit Euler Scheme is consistent.

Proposition

Let $\|\cdot\|$ be the norm in $\mathbb{R}^{(N_x-1)\times N_t}$ defined by

$$\|Q\| = \max_{n=1,\ldots,N_t} \frac{1}{\sqrt{N_x - 1}} \|Q^n\|_2,$$

where $Q = (Q^1, \ldots, Q^{N_t})$, $Q^m \in \mathbb{R}^{N_x-1}$. Then there exists a constant C_2 independent on Δ_x such that the Implicit Euler scheme is stable if $\Delta_t \leq C_2$.

The last result is proved via the Energy Method. All the proofs can be found in Chapter 3 of Y. Achdou, and O. Pironneau, *Computational methods for option pricing*, Society for Industrial and Applied Mathematics, 2005.

Crank-Nicolson

- Idea: consider the solution of the PDE at points $(t_{n+1/2}, x_j)$, where
$$t_{n+1/2} = \frac{t_n + t_{n+1}}{2}.$$
- Such a scheme is usually more accurate than Euler's schemes and has the same kind of stability as Implicit Euler.
- Crank-Nicolson scheme provides the following discretization scheme for the PDE in (12):
$$\begin{aligned} u_j^{n+1} &= u_j^n + \frac{1}{4}x_j^2(\sigma_j^{n+1})^2\frac{\Delta_t}{(\Delta_x)^2}(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}) \\ &\quad + \frac{1}{4}x_j^2(\sigma_j^n)^2\frac{\Delta_t}{(\Delta_x)^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n) \\ &\quad + \frac{1}{2}rx_j\frac{\Delta_t}{2\Delta_x}(u_{j+1}^{n+1} - u_{j-1}^{n+1}) + \frac{1}{2}rx_j\frac{\Delta_t}{2\Delta_x}(u_{j+1}^n - u_{j-1}^n) \\ &\quad - \frac{1}{2}r\Delta_t u_j^{n+1} - \frac{1}{2}r\Delta_t u_j^n. \end{aligned}$$
- As it was the case for Implicit Euler, in order to obtain the solution at time step $n+1$ from the one at n, one has to solve a no-trivial system of linear equations with $N_x - 1$ unknowns.

Crank-Nicolson

The solution at time step $n+1$ is obtained from the solution at time step n by solving the system of linear equations

$$\left(I + \frac{1}{2}\Delta_t A^{(n+1)}\right) U^{n+1} = U^n + \frac{1}{2}V^n + \frac{1}{2}R^n,$$

where:

- $U^n = (u_1^n, u_2^n, \dots, u_{N_x-1}^n)$ and $U^{n+1} = (u_1^{n+1}, u_2^{n+1}, \dots, u_{N_x-1}^{n+1})$
- A^n is the $(N_x-1)\times(N_x-1)$ tridiagonal matrix already defined for the Implicit Euler scheme
- R^n is the vector coming from the values of the solution at the n-th step, i.e., the ones that are used in the Explicit Euler scheme.
- $V^n = (v_a, 0, \dots, 0, v_b)^T$ with

  $$v_a = \psi_1(a)\left[\frac{1}{2(\Delta_x)^2}(\sigma_1^n)^2 + r\frac{1}{2\Delta_x}x_1\right]$$

  and

  $$v_b = \psi_2(b)\left[\frac{1}{2(\Delta_x)^2}(\sigma_{N_x-1}^n)^2 + r\frac{1}{2\Delta_x}x_{N_x-1}\right].$$

Upwind of Explicit Euler scheme

- We also consider a slight modification to the Explicit Euler scheme introduced above, where we substitute the central differences approximation of the first space derivative with a forward difference approximation.
- This might increase stability since the term multiplying the first space derivative in the PDE we consider is positive.
- The scheme then reads

$$u_j^{n+1} = u_j^n + \frac{1}{2} x_j^2 (\sigma_j^n)^2 \frac{\Delta_t}{(\Delta_x)^2} (u_{j+1}^n - 2u_j^n + u_{j-1}^n) + r x_j \frac{\Delta_t}{\Delta_x} (u_{j+1}^n - u_j^n) - r \Delta_t u_j^n.$$

Implementation in Python

- In the package

  `finitedifferencemethods`

  you can find the implementation of the four finite difference methods described above with some tests in option pricing.
- In particular, the class in

  `pricingWithPDEs`

  provides an implementation of a general finite difference method for a general PDE with first time derivative and possibly higher order space derivates.
- In this class, the solution may be dynamically computed and plotted and also stored in a matrix.
- Such a class is extended in four classes, see `explicitEuler`, `implicitEuler`, `crankNicolson`, `upwind`, that solve the PDE associated to our local volatility model, yet for general initial (i.e., for zero maturity) and boundary conditions.
- Such conditions depend indeed on the option taken into consideration.

Tests in Python

- The module

  `testExplicitEulerAndUpwind`

  tests the stability of the Explicit Euler and Upwind method for the valuation of a call option.

- The module

  `testPricingMethods`

  compares the performances of the Explicit Euler, Implicit Euler and Crank-Nicolson methods in the valuation of a call option under the Black-Scholes module, so to have the benchmark of the analytic value. It does this both by printing the time needed to compute the solutions and the average error got for 30 tests, and plotting the approximated solutions given by the methods against the analytic one.

- In the module

  `testKnockOutOption`

  we compare the valuation of Knock-Out options by the Monte-Carlo approach for the simulation and discretization of a continuous time stochastic process solution to a given SDE and the Implicit Euler method to solve the associated PDE.

Some references for finite difference methods in the valuation of PDEs for option pricing

- Y. Achdou, and O. Pironneau, *Computational methods for option pricing*, Society for Industrial and Applied Mathematics, 2005
- J. W. Thomas *Numerical partial differential equations: finite difference methods*, Vol. 22. Springer Science & Business Media, 2013.
- Chapter 4 of the notes *Numerical Method in Finance*, by G. H. Meyer, Georgia Institute of Technology. Available at

  http://people.math.gatech.edu/ meyer/MA6635/chap4.pdf