Projet VBA-Python : Pricing d'options avec arbre trinomial

Ce projet a été réalisé dans le cadre du cours « Programmation comparée VBA-Python » du master 272 Ingénierie Economique et Financière de l'université Paris-Dauphine par :

- Thibault Charbonnier
- Baptiste Dufour

Il a consisté en la création d'un Pricer d'option via un arbre trinomial comprenant les capacités suivantes :

- Pricer des options Européennes, Américaines, Bermudiennes et Digitales
- Prendre un compte un versement de dividende du sous-jacent
- Fournir une interface graphique pour chaque projet (Python & VBA)
- Lier Python et Excel via Xlwings

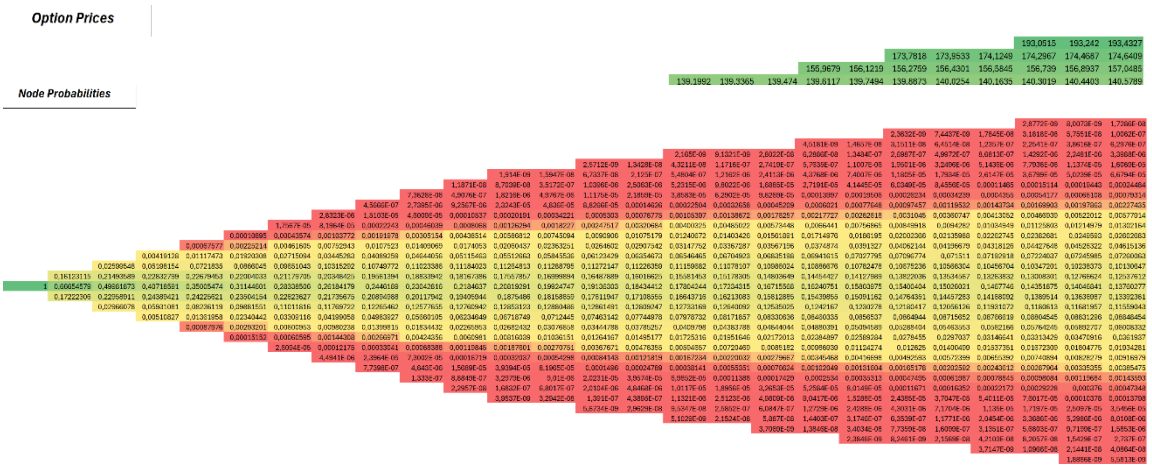
Dans ce document nous aborderons quelques points que nous considérons intéressants, principalement les graphiques qui permettent d'apprécier nos résultats visuellement et les axes d'amélioration supplémentaires que nous avons apportés.

Le projet VBA

Notre projet VBA se décompose en 7 feuilles :

- L'interface de pricing avec les paramètres et les résultats.
- 3 feuilles de plot : prix du sous-jacent, prix de l'option, probabilités des nœuds.
- 1 feuille contenant la distribution des Grecques selon le spot.
- 2 feuilles pour étudier deux phénomènes : la convergence et le gap.

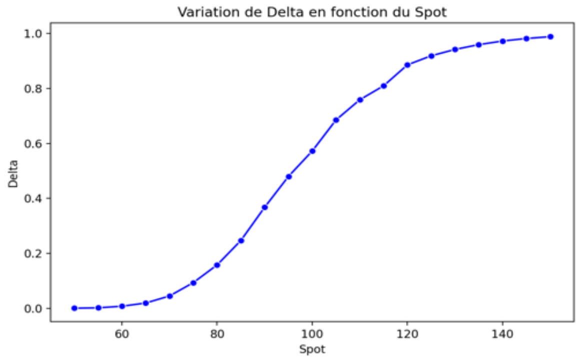
Plots intéressants :

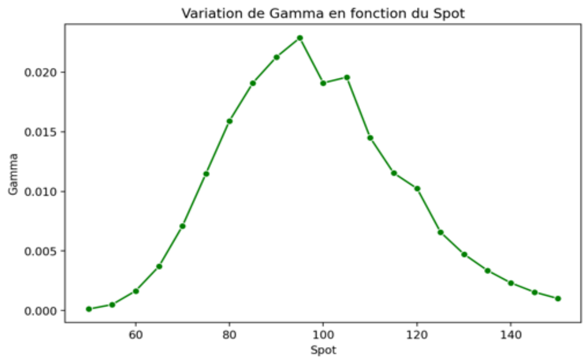


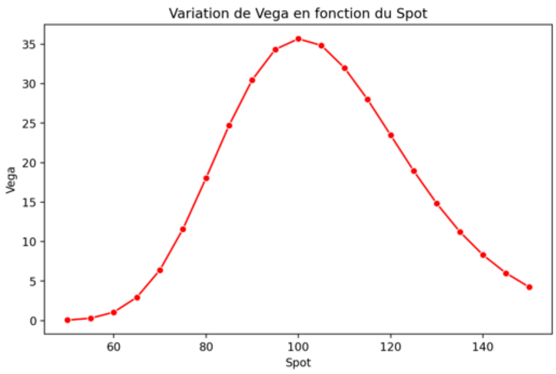
Underlying Prices

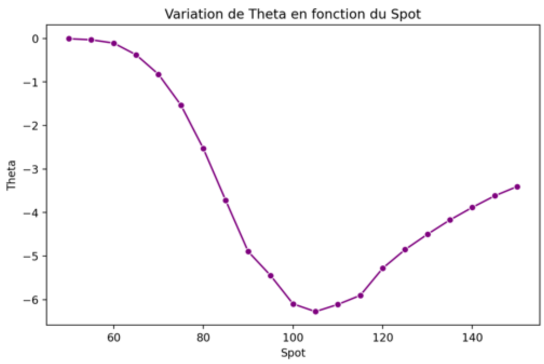
																							293,8524	294,1424	294,4327
																				274,2849	274,5556	274,8265	275,0977	275,3692	275,6409
																		256,273	256,5259	256,7791	257,0324	257,2861	257,54	257,7941	258,0485
															239,2079	239,444	239,6802	239,9168	240,1535	240,3905	240,6277	240,8651	241,1028	241,3407	241,5789
													223,4995	223,72	223,9408	224,1618	224,383	224,6044	224,826	225,0479	225,27	225,4922	225,7148	225,9375	226,1604
											208,8226	209,0287	209,2349	209,4414	209,6481	209,8549	210,062	210,2693	210,4768	210,6845	210,8924	211,1005	211,3088	211,5173	211,7261
										195,3021	195,4948	195,6877	195,8808	196,0741	196,2676	196,4612	196,6551	196,8492	197,0434	197,2378	197,4325	197,6273	197,8223	198,0175	198,2129
									182,6569	182,8372	183,0176	183,1982	183,379	183,5599	183,7411	183,9224	184,1039	184,2855	184,4674	184,6494	184,8316	185,014	185,1966	185,3793	185,5623
								170,8305	170,9991	171,1678	171,3367	171,5058	171,675	171,8445	172,014	172,1838	172,3537	172,5238	172,694	172,8644	173,035	173,2057	173,3767	173,5477	173,719
							159,7698	159,9275	160,0853	160,2433	160,4014	160,5597	160,7181	160,8767	161,0355	161,1944	161,3534	161,5127	161,672	161,8316	161,9913	162,1511	162,3111	162,4713	162,6316
						149,4253	149,5727	149,7203	149,8681	150,016	150,164	150,3122	150,4605	150,609	150,7576	150,9064	151,0553	151,2043	151,3535	151,5029	151,6524	151,802	151,9518	152,1018	152,2519
					139,7505	139,8884	140,0265	140,1646	140,3029	140,4414	140,58	140,7187	140,8576	140,9966	141,1357	141,275	141,4144	141,5539	141,6936	141,8334	141,9734	142,1135	142,2537	142,3941	142,5346
				130,7021	130,8311	130,9602	131,0895	131,2188	131,3483	131,4779	131,6077	131,7375	131,8675	131,9976	132,1279	132,2583	132,3888	132,5194	132,6502	132,7811	132,9121	133,0433	133,1746	133,306	133,4375
			122,2396	122,3603	122,481	122,6019	122,7228	122,8439	122,9652	123,0865	123,208	123,3296	123,4513	123,5731	123,695	123,8171	123,9393	124,0616	124,184	124,3065	124,4292	124,552	124,6749	124,7979	124,9211
		114,325	114,4379	114,5508	114,6638	114,777	114,8902	115,0036	115,1171	115,2307	115,3444	115,4582	115,5721	115,6862	115,8003	115,9146	116,029	116,1435	116,2581	116,3728	116,4877	116,6026	116,7177	116,8328	116,9481
	106,9229	107,0284	107,134	107,2397	107,3456	107,4515	107,5575	107,6636	107,7699	107,8762	107,9827	108,0892	108,1959	108,3027	108,4095	108,5165	108,6236	108,7308	108,8381	108,9455	109,053	109,1606	109,2683	109,3761	109,4841
100	100,0987	100,1975	100,2963	100,3953	100,4944	100,5935	100,6928	100,7922	100,8916	100,9912	101,0908	101,1906	101,2904	101,3904	101,4905	101,5906	101,6908	101,7912	101,8916	101,9922	102,0928	102,1936	102,2944	102,3954	102,4964
	93,71002	93,80249	93,89505	93,98771	94,08045	94,17329	94,26622	94,35924	94,45235	94,54556	94,63885	94,73224	94,82572	94,91929	95,01296	95,10672	95,20057	95,29451	95,38854	95,48267	95,57689	95,67121	95,76562	95,86012	95,95471
		87,81567	87,90233	87,98907	88,07589	88,16281	88,2498	88,33689	88,42406	88,51131	88,59865	88,68608	88,7736	88,8612	88,94888	89,03666	89,12452	89,21247	89,3005	89,38862	89,47683	89,56512	89,6535	89,74197	89,83053
			82,29208	82,37328	82,45457	82,53593	82,61738	82,6989	82,78051	82,8622	82,94396	83,02581	83,10774	83,18975	83,27184	83,35401	83,43627	83,5186	83,60102	83,68351	83,76609	83,84875	83,93149	84,01431	84,09722
				77,11592	77,19202	77,26819	77,34444	77,42076	77,49716	77,57363	77,65018	77,7268	77,8035	77,88028	77,95713	78,03406	78,11106	78,18814	78,26529	78,34252	78,41983	78,49722	78,57468	78,65221	78,72983
					72,26534	72,33665	72,40803	72,47948	72,551	72,6226	72,69426	72,76599	72,8378	72,90967	72,98162	73,05364	73,12573	73,19789	73,27012	73,34242	73,41479	73,48724	73,55975	73,63234	73,705
						67,71986	67,78669	67,85358	67,92053	67,98756	68,05465	68,1218	68,18902	68,25631	68,32367	68,39109	68,45858	68,52613	68,59375	68,66144	68,72919	68,79701	68,8649	68,93286	69,00088
							63,46029	63,52291	63,5856	63,64834	63,71115	63,77402	63,83695	63,89995	63,963	64,02612	64,0893	64,15254	64,21585	64,27921	64,34264	64,40614	64,46969	64,53331	64,59699
								59,46865	59,52733	59,58607	59,64487	59,70373	59,76264	59,82162	59,88065	59,93974	59,99889	60,05809	60,11736	60,17668	60,23606	60,2955	60,355	60,41456	60,47417
									55,72808	55,78307	55,83812	55,89322	55,94837	56,00358	56,05885	56,11417	56,16954	56,22497	56,28045	56,33598	56,39158	56,44722	56,50292	56,55868	56,61449
										52,22279	52,27433	52,32591	52,37754	52,42923	52,48097	52,53275	52,58459	52,63648	52,68842	52,74042	52,79246	52,84455	52,8967	52,9489	53,00115
											48,93799	48,98628	49,03462	49,08301	49,13144	49,17992	49,22845	49,27703	49,32566	49,37433	49,42305	49,47182	49,52064	49,56951	49,61842
													45,90505	45,95035	45,99569	46,04108	46,08651	46,13199	46,17751	46,22308	46,26869	46,31435	46,36005	46,4058	46,45159
															43,06008	43,10257	43,1451	43,18768	43,2303	43,27295	43,31566	43,3584	43,40118	43,44401	43,48688
																	40,39142	40,43128	40,47118	40,51112	40,55109	40,59111	40,63116	40,67126	40,71139
																			37,88816	37,92555	37,96297	38,00043	38,03793	38,07547	38,11304
																						35,57511	35,61021	35,64535	35,68053
																								33,37034	33,40327

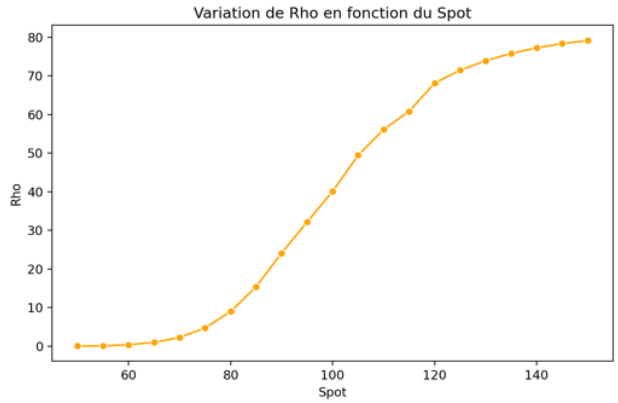
Grecques :





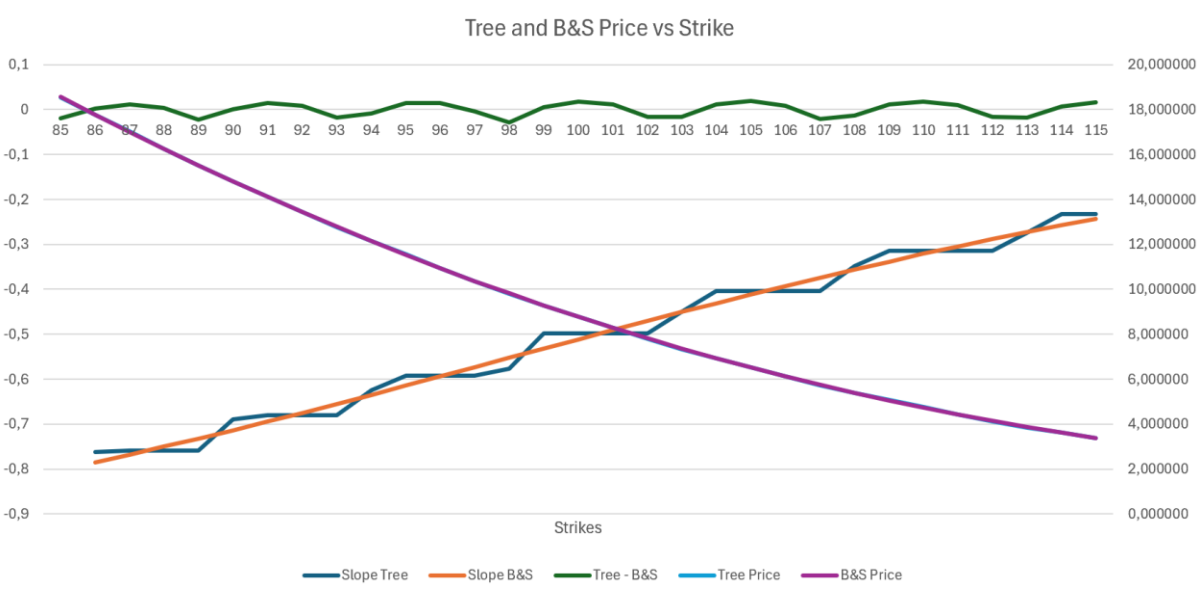






Convergence :

Analyse des gap et pentes :



Calcul des grecques :

On calcule les grecques par différences finies et on compare avec les grecques de Black & Scholes.

	Greeks Tree	Greeks B&S
Delta Δ	0,570	0,500
Gamma Γ	0,020	0,022
Vega v	35,666	36,168
Theta Θ	-5,813	-5,879
Rho P	40,365	40,496

Nombre de step en fonction de la précision :

On a également ajouté la possibilité de déterminer le nombre de step de l'abre en fonction d'une précision voulue.

On définit la précision comme :

$$\frac{\text{Gap}}{S_0} \approx \frac{1}{N} \frac{3}{8\sqrt{2\pi}} \frac{\sigma^2}{\sqrt{e^{\sigma^2} - 1}}$$

On a alors la relation suivante qui nous permet d'avoir le nombre de step :

$$NbSteps = \frac{T}{\Delta t} \approx \frac{3}{8\sqrt{2\pi}} \frac{S_0}{Gap} \frac{\sigma^2 T}{\sqrt{e^{\sigma^2 T} - 1}}$$

Sur l'interface on laisse l'utilisateur choisir :

Mode (*)	Step
NbSteps (**)	1000
Precision (***)	0,0033%

Mode (*)	Precision
NbSteps (**)	90
Precision (***)	0,0300%

Le projet Python

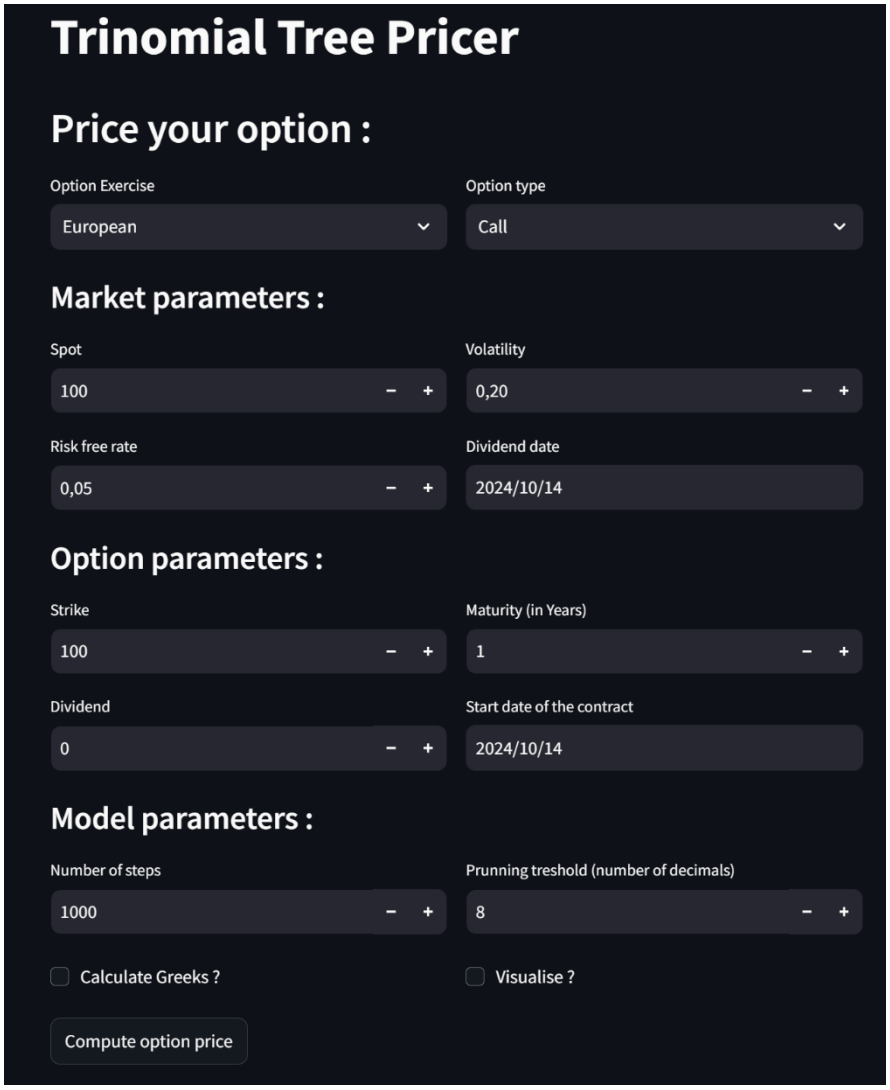
Pour réaliser ce projet en Python, voici les librairies utilisées :

Seaborn, networkx et matplotlib (Visualisation), Scipy (Loi normale), Pandas (Dataframes), Streamlit (Interface Graphique).

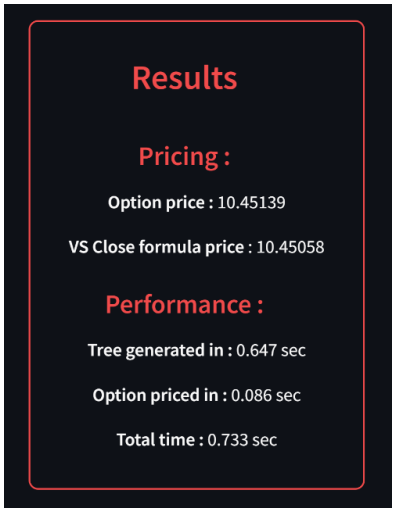
Interface

Nous avons créé une interface graphique avec le package Streamlit en 100% Python afin que l'utilisateur puisse interagir avec le Pricer. **Pour lancer l'interface, entrez la commande « streamlit run app.py » dans le cmd à la racine du projet.**

- **<u>Paramétrisation sur l'interface :</u>**



- **Résultats sur l'interface :**



Option Greeks	Value
Delta	0.6462
Gamma	0.0186
Vega	37.5589
Theta	6.4001
Rho	53.0875

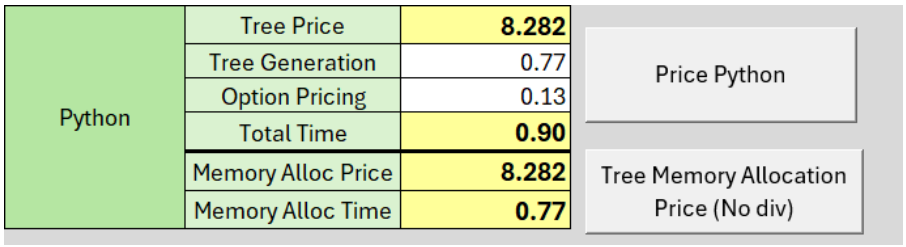
Xlwings

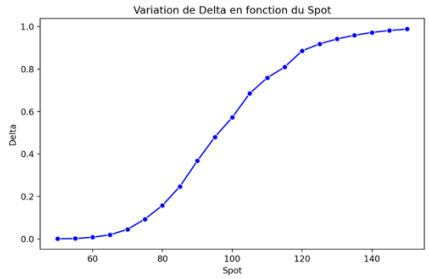
Nous avons également lié notre projet python au projet Excel via Xlwings.

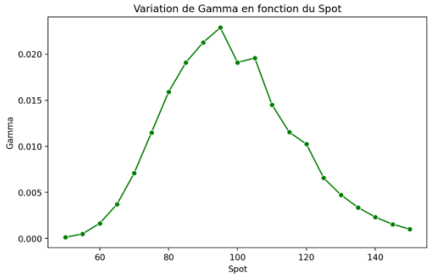
Nous avons donc créé 4 subroutines sous python et accessibles sous Excel permettant de :

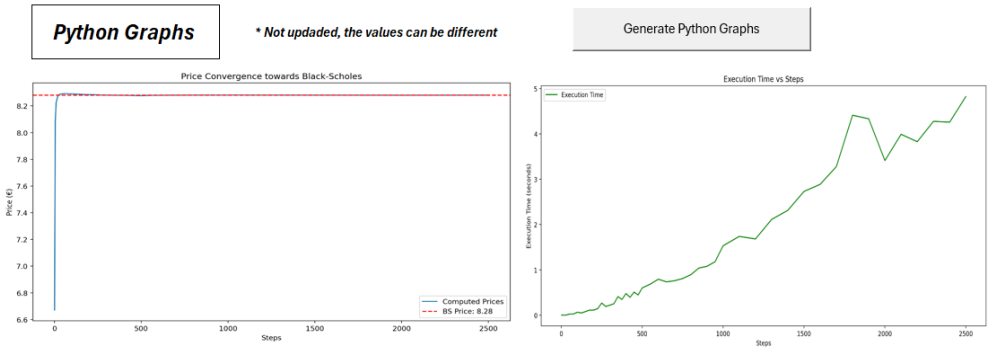
- Pricer une option européenne ou américaine avec dividende.
- Pricer une option avec la méthode de libération de l'allocation de mémoire.
- Générer la distribution des Grecs selon le spot.
- Générer les graphiques de convergence de la même manière que Excel mais via python.

Voici comment nous avons intégré sous Excel ses options.









Afin de récupérer les données utilisateurs, nous avons créé des fonctions spéciales récupérant les données de l'excel via python.

Dans ce projet nous avons intégré des fonctionnalités supplémentaires comme le calcul d'une option sans dividende avec la **méthode de la libération de la mémoire** et la création de graphiques permettant de visualiser la distribution des Grecs selon le spot.

Types d'option supplémentaires

Nous avons ajouté quelques fonctionnalités additionnelles à notre pricer.

Par exemple, nous avons ajouté deux types d'options supplémentaires :

- Options Bermudiennes :

De la même manière que les options Américaines, on doit pouvoir exécuter l'option avant la maturité.

Pour gérer la possibilité d'exécution, on définit une liste de time step où l'option peut être exécutée de la manière suivante :

```python
@cached_property
def exercise_steps(self) -> list[int]:
    '''
    Définit une liste qui contient tous les timeSteps où l'option peut être exécutée
    '''
    if isinstance(self.option, BermudeanCallOption) or isinstance(self.option, BermudeanPutOption):
        exercise_dates = self.option.exercise_dates
        time_delta_in_days = self.time_delta * 365
        return [ceil((ex_date - self.option.start_date).days/time_delta_in_days) for ex_date in exercise_dates]

    elif isinstance(self.option, AmericanCallOption) or isinstance(self.option, AmericanPutOption):
        return [i for i in range(self.nb_steps)]

    else:
        return []
```

Sur l'interface, on rajoute le choix des dates d'exécution :

- Options Digitales :

Pour l'option digitale, on change le payoff de l'option :

```python
@dataclass
class DigitalCallOption(Option):
    coupon : float = 1

    def payoff(self, price : float) -> float:
        if price > self.strike:
            return self.coupon
        else:
            return 0
```

```python
@dataclass
class DigitalPutOption(Option):
    coupon : float = 1

    def payoff(self, price : float) -> float:
        if price < self.strike:
            return self.coupon
        else:
            return 0
```

Sur l'interface, on ajoute le choix du « coupon » de l'option digitale :