Introduction

La méthode de Monte-Carlo consiste à tirer des échantillons selon une loi uniforme (individus, réalisation d'un variable aléatoire) dans un domaine et à regarder ceux qui sont présent ou ceux qui ne le sont pas. On peut dériver ce principe à des applications plus larges :

- **Estimation numérique :** calcul d'intégrales, estimation de surface et de volumes, évaluation statistique, algorithme des champignons.
- **Simulation :** modèles probabilistes (chaînes de Markov, diffusion thermique), modélisation de système complexes (simulation physique, finance, biologie).
- **Optimisation :** recuit simulé, recherche stochastique (exploration d'espaces de solutions via des échantillons aléatoires), algorithmes génétiques.
- **Apprentissage :** apprentissage par renforcement, réseaux de neurones

Dans ce TP, on se concentre sur les méthodes d'estimation numérique, application directe de Monte-Carlo.

PARTIE 1 : Calculs de surfaces et de volumes

On peut déduire à partir de théorème centrale limite un intervalle de confiance à 95% pour la moyenne d'une variable aléatoire X :

$$[-\frac{as}{\sqrt{n}} + \bar{X}, \frac{as}{\sqrt{n}} + \bar{X}]$$

Cet intervalle de confiance donne une estimation fiable de la moyenne E(X) à partir de n > 50. *s est l'écart type empirique de X, n le nombre d'individus de X considérés lors du calcul de X_barre et de la moyenne empirique, a = 1.96 pour un intervalle de confiance à 95%.*

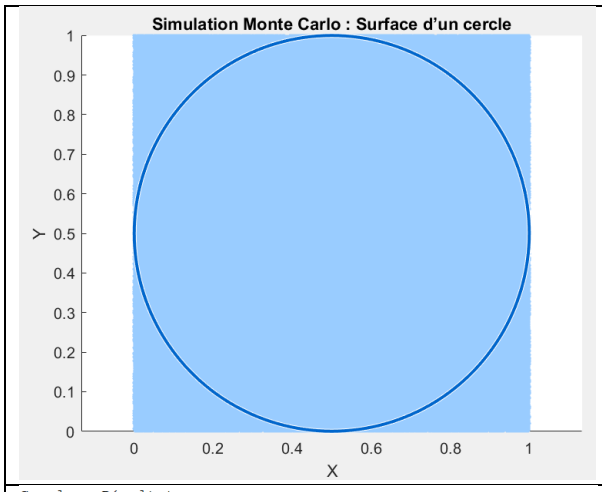
Cette méthode d'estimation de la moyenne d'une variable aléatoire est la méthode de Monte-Carlo.

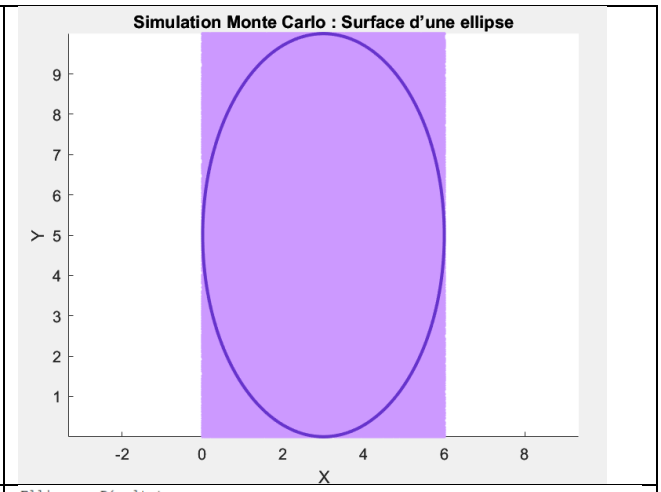
Pour estimer des surfaces à l'aide de Monte-Carlo, on tire des points (individus, échantillons de la variable aléatoire X) dans un domaine D donné sur lequel se trouve notre fonction f (tel que Df inclus dans D).

On peut définir une variable aléatoire Z qui décrit le fait d'être à l'intérieur ou non d'un domaine délimité par la fonction f. Cette variable aléatoire est donc une loi de Bernoulli qui est donc tel que E(Z) = p (probabilité d'être à l'intérieur ou l'extérieur de la zone). De fait, on peut estimer grâce à l'intervalle de confiance de la moyenne de Monte Carlo la valeur de p. Puis en multipliant p par AB, l'intervalle de confiance de l'aire du domaine de f :

$$[-\frac{ABas}{\sqrt{n}} + \frac{ABn_1}{n}, \frac{ABas}{\sqrt{n}} + \frac{ABn_1}{n}]$$

On cherche donc à appliquer cette estimation à un calcul d'aire de cercle et d'ellipse. En tirant des points dans une zone donnée et en calculant ceux qui tombent ou non dans la zone, on évalue :





Cercle : Résultats

N	Air_Estime	Air_Exacte	IC_Lower	IC_Upper
1000	0.805	0.7854	0.78044	0.82956
10000	0.7815	0.7854	0.7734	0.7896
1e+05	0.78676	0.7854	0.78422	0.7893
1e+06	0.78515	0.7854	0.78435	0.78596

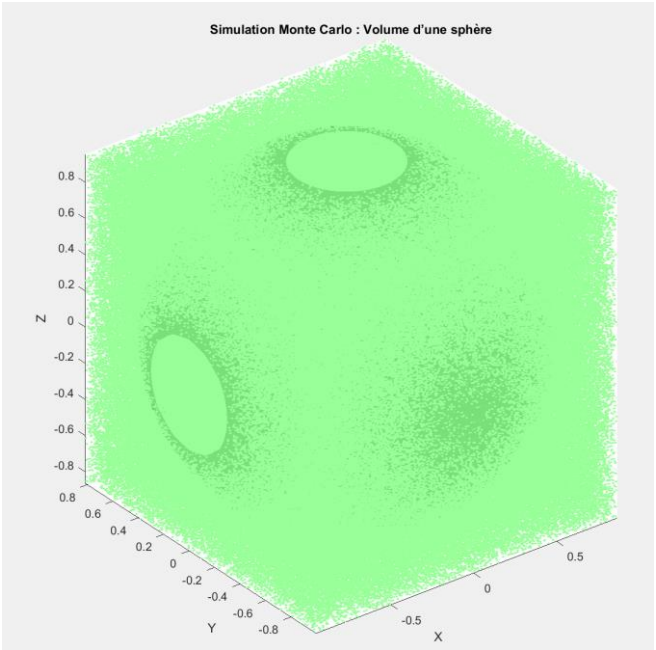
Ellipse : Résultats

N	Air_Estime	Air_Exacte	IC_Lower	IC_Upper
1000	46.56	47.124	46.56-2.8547i	46.56+2.8547i
10000	47.424	47.124	47.424-0.91966i	47.424+0.91966i
1e+05	47.245	47.124	47.245-0.28971i	47.245+0.28971i
1e+06	47.129	47.124	47.129-0.091387i	47.129+0.091387i

IC_Lower & IC_Upper sont les bornes de l'intervalle de confiance. N le nombre de point tirés.

On constate comme prévu que l'aire estimée est d'autant plus précise que l'on tire d'échantillons. L'air estimé est toujours comprise dans l'intervalle de confiance (à une partie imaginaire près pour l'ellipse).

Nous avons donc calculé correctement les airs pour des surfaces. Qu'en est il des volumes comme une sphère dans un cube ?



L'unique différence réside dans l'ajout d'un axe, qui se traduit par l'ajout d'un C multiplicatif (axe z) dans la formule de l'intervalle de confiance et dans la formule empirique de l'aire (du volume maintenant). La volume exacte vaut 4/3 * pi * R^3.

Sphère : Résultats

N	Volume_Estime	Volume_Exacte	IC_Lower	IC_Upper
1000	4.32	4.1888	4.32-0.23473i	4.32+0.23473i
10000	4.152	4.1888	4.152-0.070905i	4.152+0.070905i
1e+05	4.193	4.1888	4.193-0.022679i	4.193+0.022679i
1e+06	4.1894	4.1888	4.1894-0.0071645i	4.1894+0.0071645i

L'estimation est à peu près exacte même pour seulement 1000 tirages.

Les résultats obtenus sur les surfaces et le volume vont dans le sens de ce qui avait déjà été démontré mathématiquement, il s'agit donc d'une méthode d'évaluation fiable.

PARTIE 2 : Calculs d'intégrales

Pour le calcul d'intégrales, on se ramène à un problème de Monte-Carlo via la définition de l'espérance d'une variable aléatoire continue pour une loi uniforme dans [0, 1], de fait on aura alors, par application du théorème des moments :

$$I = \int_0^1 f(x)dx = E(f(X))$$

Il ne reste alors plus qu'à appliquer Monte-Carlo pour estimer l'espérance et donc l'intégrale.

Dans le cas de dimensions supérieures, cela revient juste à changer l'expression de l'espérance (ajout de plusieurs variables dans f) pour s'adapter à la dimension, il faudra alors tester n^4 points pour un problème de dimension 4 :

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 f(x,y,z,t)dxdxdzdt \simeq \frac{1}{n}\sum_{i=1}^{n} f(X_i, Y_i, Z_i, T_i)$$

$$X \sim U[0,1], Y \sim U[0,1], Z \sim U[0,1], T \sim U[0,1]$$

Ça tombe bien, on tout d'abord estimer I et J :

$$I = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \exp(x+y+z+t)dxdydzdt \quad J = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \exp(xyzt)dxdydzdt$$

L'intégrale J a une valeur exacte non triviale, pour l'intégrale I, on a :

$$I_{\text{exact}} = \left(\int_0^1 e^x dx\right)^4 = \left(e^1 - 1\right)^4 = (e-1)^4$$

Resultats pour I :

N	I_Estime	I_Exact	Variance	IC_Lower	IC_Upper
1000	8.6101	8.7172	5.0704	8.2958	8.9244
10000	8.7459	8.7172	5.3132	8.6417	8.85
1e+05	8.6976	8.7172	5.2963	8.6648	8.7305
1e+06	8.7167	8.7172	5.2988	8.7063	8.7271

La valeur estimée de I se rapproche de sa valeur exacte à mesure que N augmente. L'application de Monte-Carlo aux intégrales fonctionne. Mais dans le cas de J, cette méthode a ses limites. La convergence lente (lié au théorème centrale limite) est amplifiée par les forte fluctuations de exp(xyzt), surtout pour des petits nombres de tirages. Par ailleurs, ces fluctuations augmentent la variance de l'estimation de Monte Carlo rendant donc moins précise l'estimation de J (élargissement de l'intervalle de confiance).

C'est pourquoi on utilise la méthode des variables de contrôles. Il s'agit d'introduire une fonction auxiliaire corrélée à celle intégrée (ici exp(xyzt)), mais dont l'intégrale exacte est connue. Cela permet alors d'ajuster l'estimation de J, en utilisant g pour compenser les fluctuations inutiles.

Resultats pour J (sans controle) :

N	J_Estime	Variance	IC_Lower	IC_Upper
1000	1.0641	0.10186	1.0578	1.0704
10000	1.067	0.10686	1.0649	1.0691
1e+05	1.0694	0.11278	1.0687	1.0701
1e+06	1.0692	0.11202	1.0689	1.0694

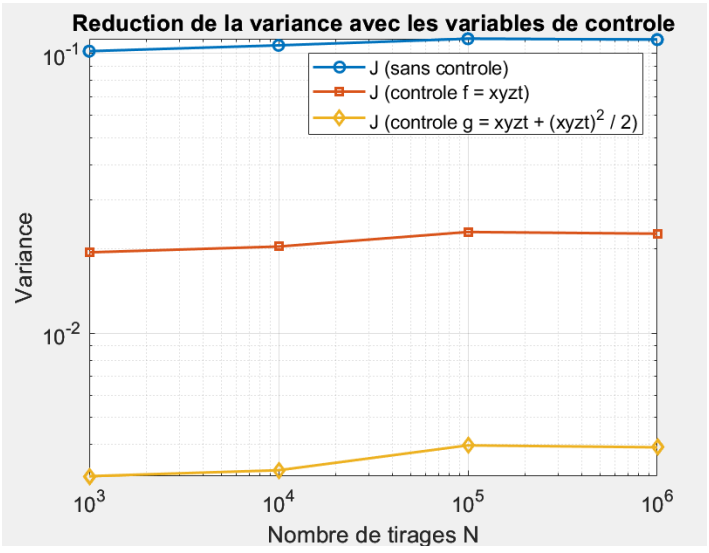
Resultats pour J avec controle f = xyzt :

N	Jf_Estime	Variance	IC_Lower	IC_Upper
1000	1.0683	0.019443	1.0671	1.0695
10000	1.0689	0.020396	1.0685	1.0693
1e+05	1.0694	0.022988	1.0693	1.0696
1e+06	1.0694	0.022672	1.0693	1.0694

Resultats pour J avec controle g = xyzt + (xyzt)^2 / 2 :

N	Jg_Estime	Variance	IC_Lower	IC_Upper
1000	1.0692	0.0030805	1.0691	1.0694
10000	1.0693	0.0032341	1.0692	1.0694
1e+05	1.0694	0.0039695	1.0694	1.0694
1e+06	1.0694	0.0039062	1.0694	1.0694

On montre donc numériquement que la méthode des variables de contrôles permet donc bien d'ajuster la précision du résultat estimé et de réduire la variance (donc resserrement de l'intervalle de confiance). Pour illustrer la réduction de la variance :



PARTIE 3 : Méthodes de quasi-Monte Carlo et quantification

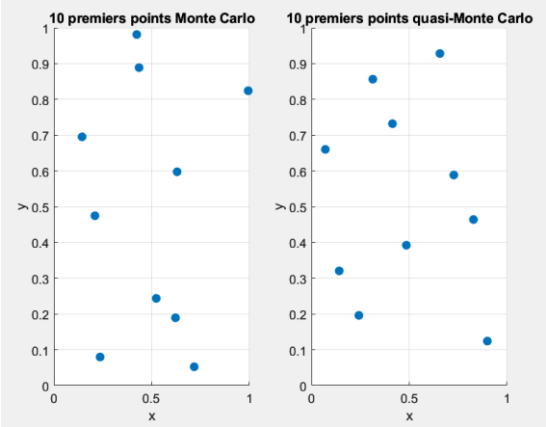
La méthode de quasi-monte Carlo permet d'améliorer la convergence en tirant de manière plus « intelligente » les points, ce qui permet d'éviter les trous ou les amas à certains endroits. Les points sont alors tirés de manière quasi-aléatoire en suivant des suites comme celles de Faure, Halton… au lieu d'une loi uniforme. Cela permet de passer d'une complexité O(n^d) à O(log(n)^d/n).

Cas uniforme

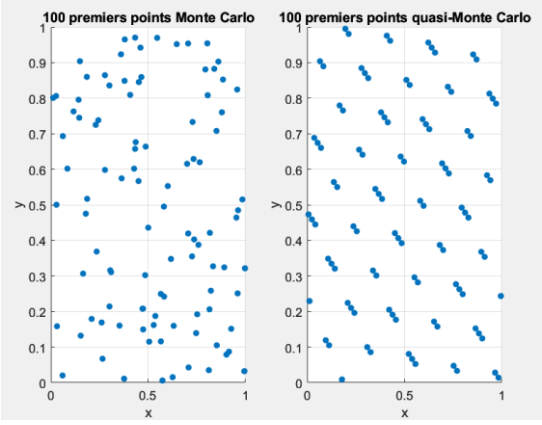
On commence à comparer Monte-Carlo et Quasi-Monte-Carlo dans un cas uniforme pour calculer I :

$$I = \int_0^1 \int_0^1 \exp(x+y)dxdy$$

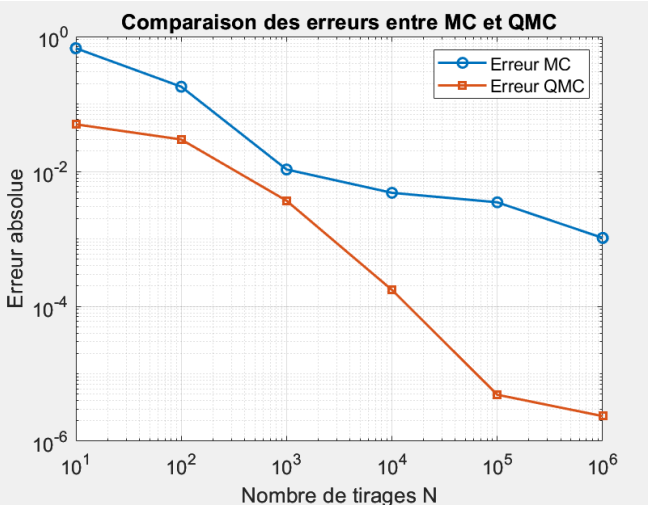
Le tirage des points est tel que :



Dans le cas de 10 points, on a l'impression qu'il s'agit d'un même tirage aléatoire, alors que dans le cas de 100 :



On voit ici que le cas du tirage uniforme conduit à des zones vides et plus disparates au début, ce qui peut limiter la précision initiale. Cependant, à mesure que le nombre de points augmente, Monte Carlo classique peut mieux recouvrir l'ensemble du domaine, tandis que quasi-Monte Carlo peut souffrir d'une saturation locale dans certaines zones. Ces différences pourraient avoir des implications sur la précision finale selon le contexte.



Toujours est il que l'écart entre la valeur calculée et la valeur exacte décroit beaucoup plus vite dans le cadre de quasi-monte Carlo. A moins donc de modéliser un problème beaucoup plus complexe, la saturation locale n'a ici pas d'impact. Voici les résultats numériques :

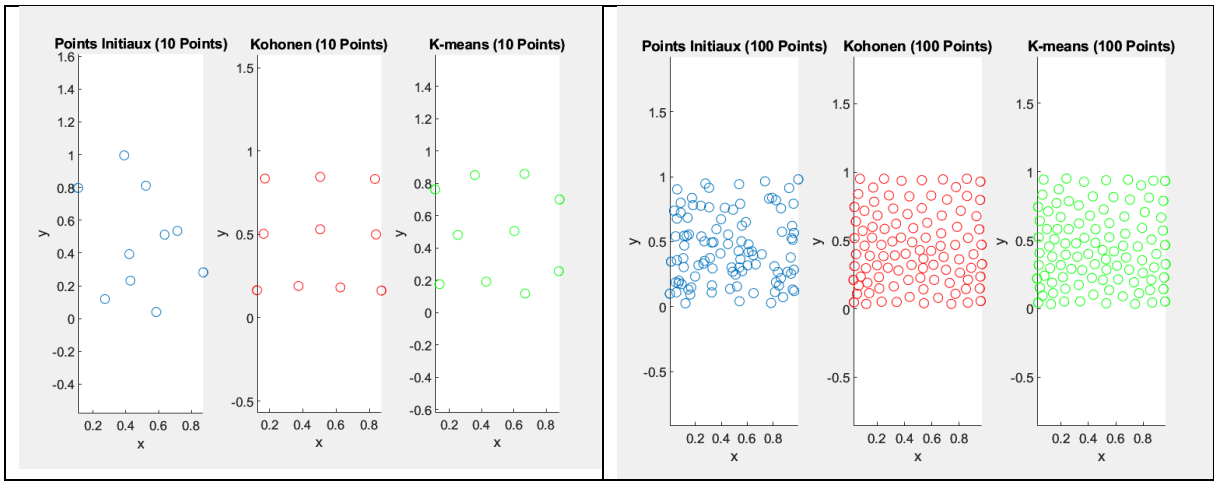
Résultats pour la méthode Monte Carlo (MC) :

N	I_MC	Variance_MC	IC_Lower_MC	IC_Upper_MC
10	3.6224	1.8841	2.4547	4.7902
100	2.7721	1.0655	2.5633	2.981
1000	2.9418	1.2287	2.8657	3.018
10000	2.9573	1.2272	2.9333	2.9814
1e+05	2.956	1.2227	2.9484	2.9636
1e+06	2.9515	1.2186	2.9491	2.9538

Résultats pour la méthode quasi-Monte Carlo (QMC) :

N	I_QMC	Variance_QMC	IC_Lower_QMC	IC_Upper_QMC
10	2.9026	1.0434	2.2559	3.5493
100	2.9227	1.1616	2.6951	3.1504
1000	2.9562	1.2284	2.88	3.0323
10000	2.9527	1.2194	2.9288	2.9766
1e+05	2.9525	1.2197	2.9449	2.96
1e+06	2.9525	1.2197	2.9501	2.9549

Bien sûr, il est possible de transformer de données purement aléatoire (loi uniforme) en tirages quasi aléatoire en utilisant des algorithmes de clustering comme Kohonen et K-means. Voici les résultats obtenus :



Résultats Kohonen :

Nk	I_Kohonen	Erreur_Absolue
10	2.8627	0.089743
100	2.732	0.22054

Résultats K-means :

Nk	I_kmeans	Erreur_Absolue
10	2.9209	0.031599
100	2.9504	0.002131

Les points initiaux sont dispersés de manière inhomogène. Après application de Kohonen et K-means, on voit que les points sont alors répartis de manière structurés dans le domaine d'intégration. Comme dit précédemment, l'efficacité de telles algorithmes est visible dans les cas où le nombre d'itérations est faible. Donc on voit clairement dans le cas de N=10, que les algorithmes de clusturing sont plus efficaces que Monte-Carlo. On voit que Kohonen est légèrement moins performant que Quasi-Monte Carlo lui-même moins performant que K-means au vu des résultats de l'intégrale (l'approximation de I est calculé comme une formule de quadrature aux centres des celluls de Voronoi et comme poids la taille des cellules).

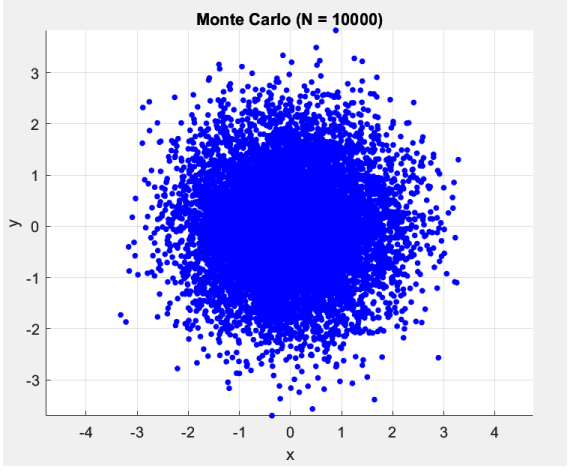
Cas Gaussien

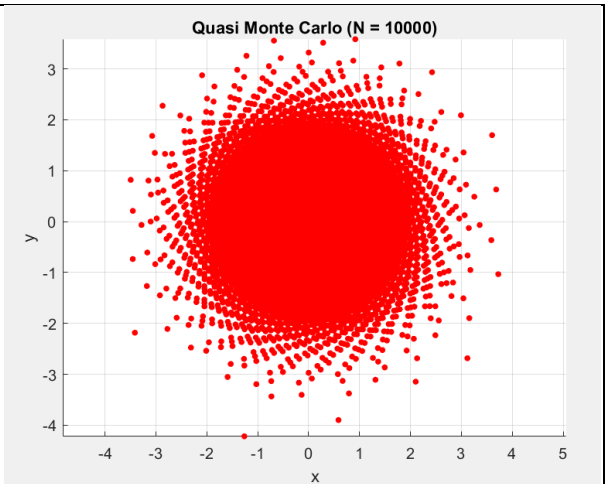
On change cette fois l'expression de l'intégrale tel que :

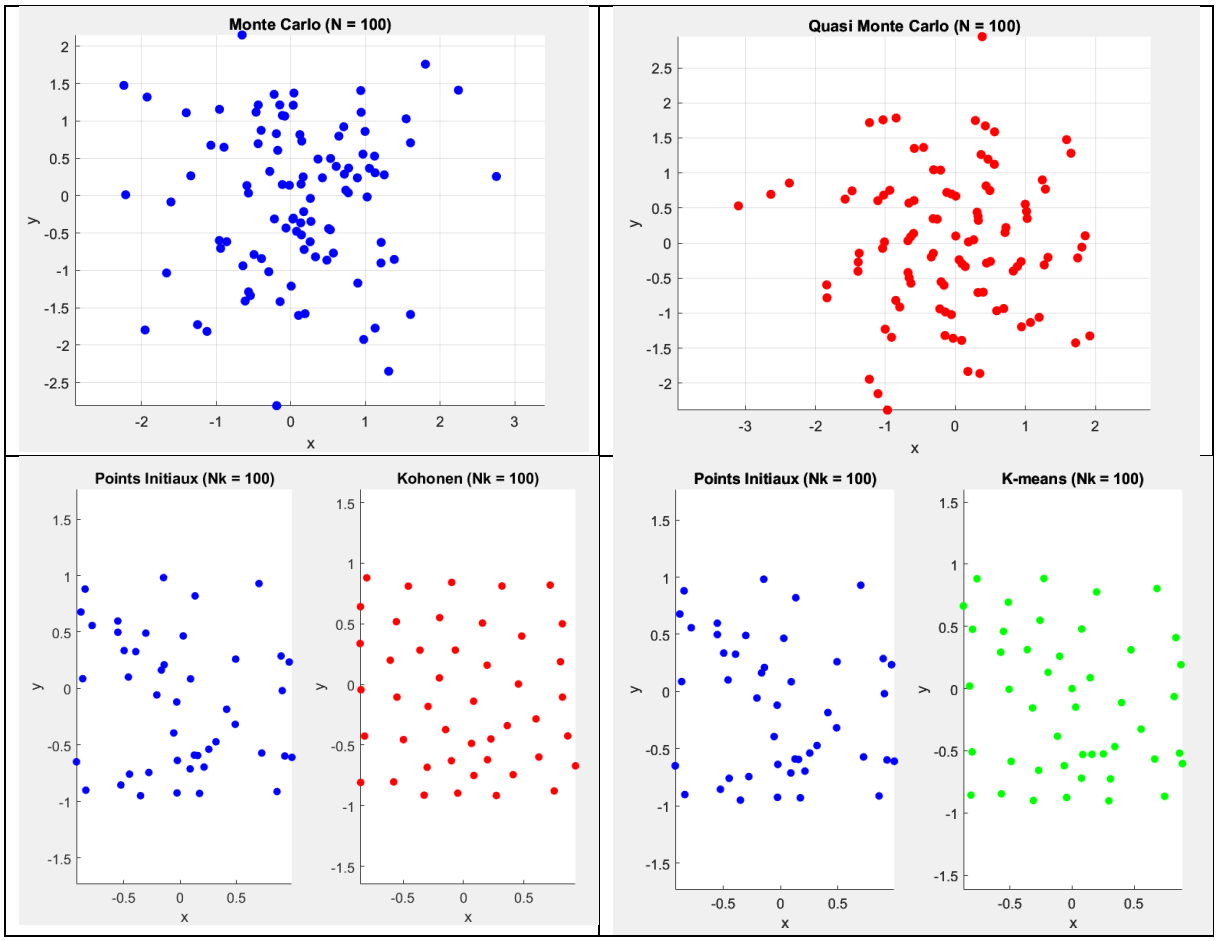
$$J = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \exp(-(\frac{x^2+y^2}{2})) \exp(x+y) dxdy.$$

Cela implique que les variables x et y ne peuvent être considérées comme totalement indépendantes dans le contexte de la distribution gaussienne bidimensionnelle. Un tirage de point uniforme va donc les concentrer près de l'origine à cause de l'exponentielle décroissante quadratique. On aura donc, avec un tirage uniforme, un amas de points près de l'origine, et plus de vide lorsque l'on s'en éloigne empêchant un calcul fiable de l'intégrale avec Monte Carlo.

Ce problème peut donc certainement être résolu via les algorithmes de clustering, et c'est l'objet de cette deuxième partie.







On voit donc que les algorithmes de clustering permettent de tirer de manière homogène dans le domaine d'intégration contrairement à MC & QMC. Ce qui se répercute dans la précision finale du calcul de l'intégral J :

```
--- Résultats Monte Carlo ---
Nombre de points (N) : 100
Estimation J_MC      : 2.763303
Intervalle de Confiance [IC_MC] : [1.641092, 3.885514]

--- Résultats Quasi Monte Carlo ---
Nombre de points (N) : 100
Estimation J_QMC     : 2.386882

--- Résultats Kohonen ---
Nombre de clusters (Nk) : 100
Estimation J_Kohonen     : 2.902607
--- Résultats K-means ---
Nombre de clusters (Nk) : 100
Estimation J_Kmeans      : 2.489823
Valeur exacte J exact    : 2.718282
```

Monte Carlo donne une valeur certes proche de J_exact mais avec un intervalle de confiance très grand, donc le résultat est imprécis. K-means et Kohonen sont quant à eux plus proches de la valeur exacte comme attendu.

PARTIE 4 : Optimisation par un algorithme mimétique

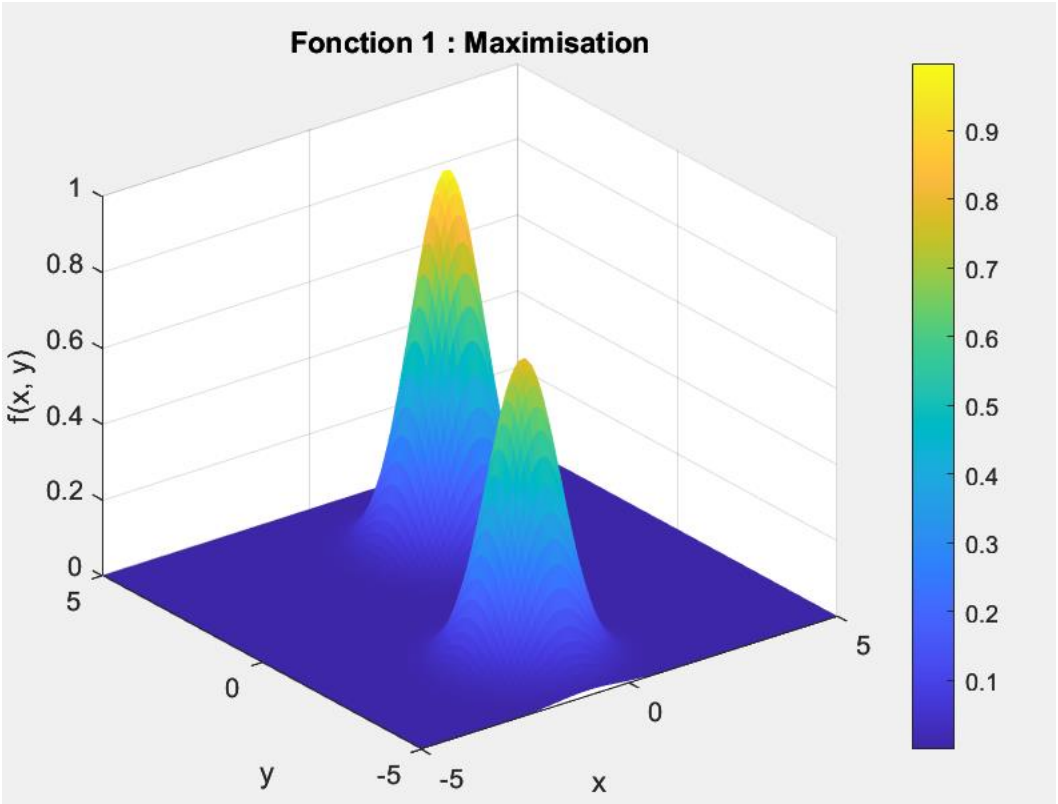
L'algorithme des champignons permet de trouver le maximum/minimum d'une fonction par utilisation des notions probabilistes. L'avantage de cet algorithme est, selon les paramètres pour la convergence, de pouvoir visualiser les différents minimum/maximum locaux. Ces extremums se présentent sous la forme de clusters (donc minimum locaux) si l'algorithme n'a pas encore convergé vers un seul point.

Cas du maximum

On choisit un domaine D = [-5, 5] ^{2} dans lequel on tire des points (individues, échantillons), puis on compare des couples d'invendus par f pour savoir lequel va prendre (approximativement, dans une zone donnée) la valeur de l'autre. L'individu cloné et éliminé (le plus petit dans le cas d'une recherche de maximum) réapparaît dans une zone aléatoire autour du premier (tel que ($x + \alpha_n X_n$, $y + \alpha_n Y_n$)).

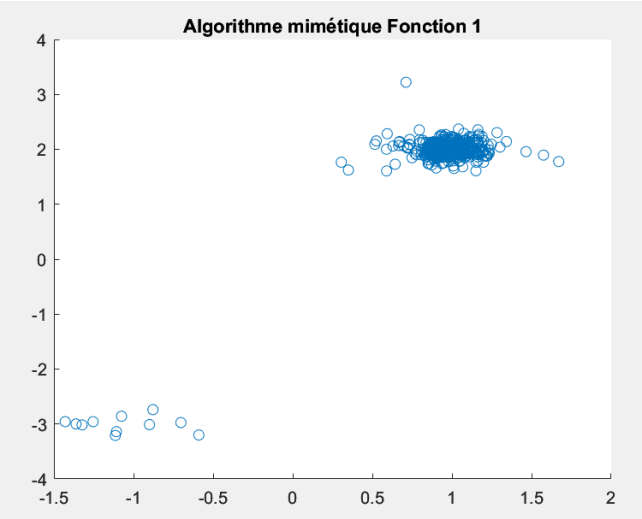
On choisit ici :

$$f(x,y) = a\exp(-b((x-1)^2+(y-2)^2)) + c\exp(-d((x+1)^2+(y+3)^2))$$

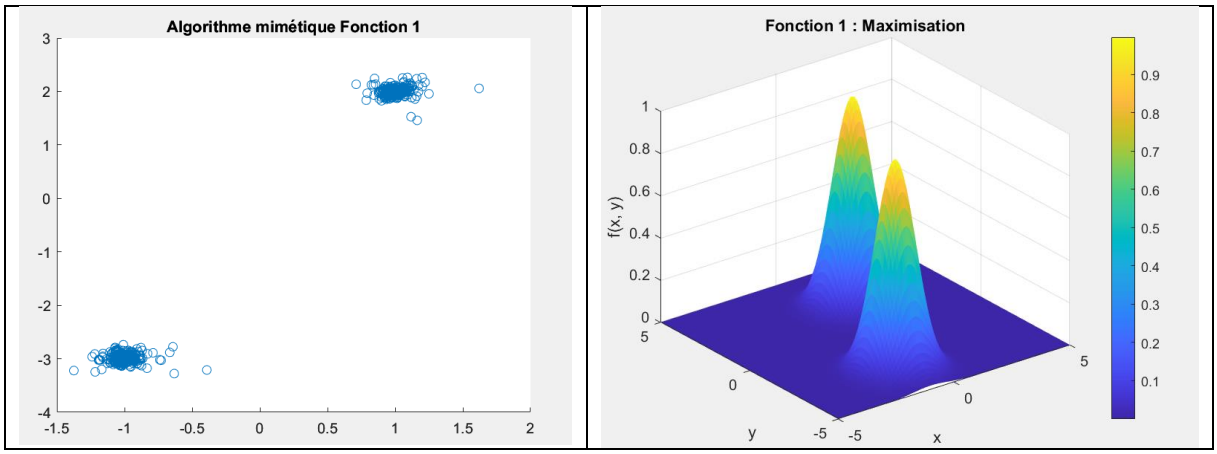


On l'appelle algorithmes des champignons, car disons que les individus qui se trompent de chemin (vers l'extremum global, un maximum de champignons) sont progressivement éliminés (reconduits…) pour être remis dans le chemin de l'invendu proche de la recherche (le plus de champignons, l'extremum global) …

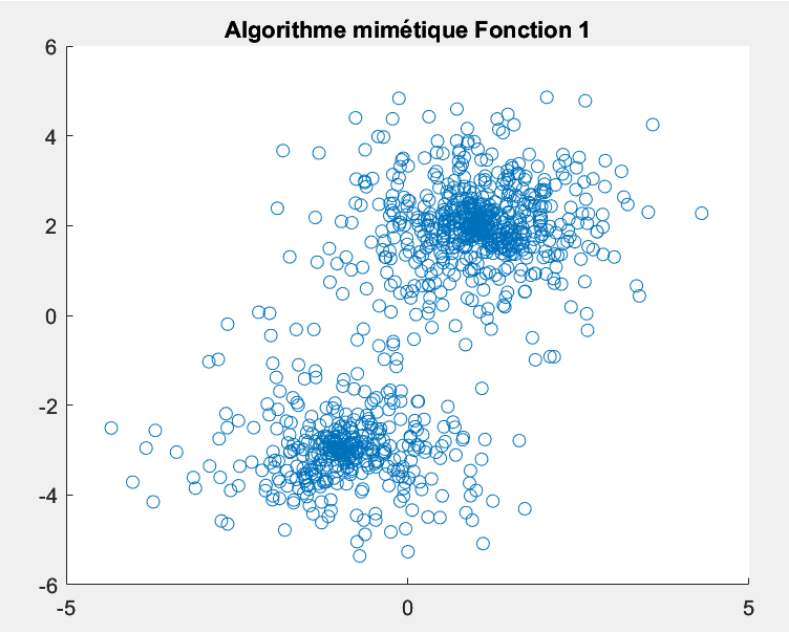
On obtient un maximum global en (1,2) pour f :



Pour rendre le cas plus complexe, on peut modifier la valeur de c (coefficient de f), on obtient alors deux amas de points :



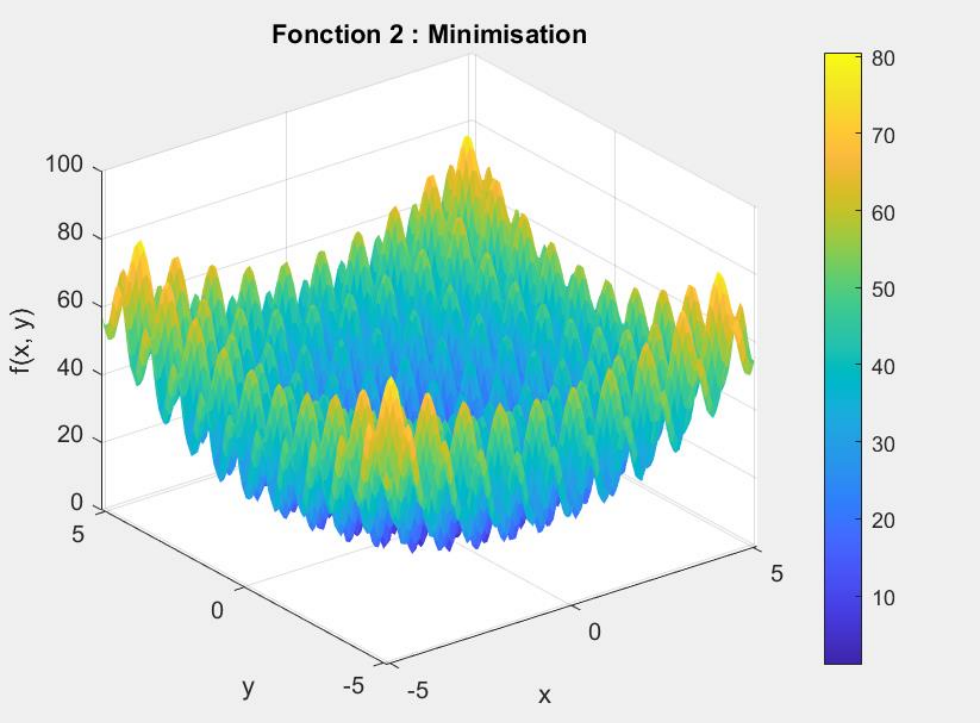
Poser c=1 au lieu de 0.8 permet de créer des maximas locaux compétitifs (au niveau des deux exponentielles). Dans cette situation comme les deux maximas sont très proches, il est difficile de déterminer le maximum global, mais on peut tout de même jouer sur epsilon et le nombre d'itération et on voit alors que plus de points s'approchent de (1,2) :



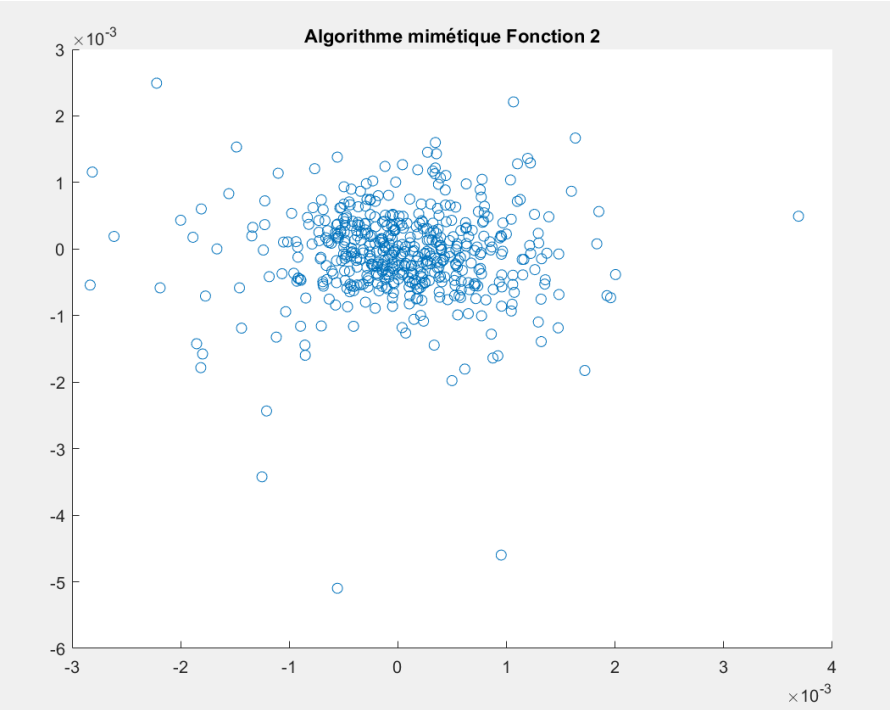
Cas du minimum

On change le domaine D tel que D = [5.12, 5.12] ^{2} et la fonction tel que :

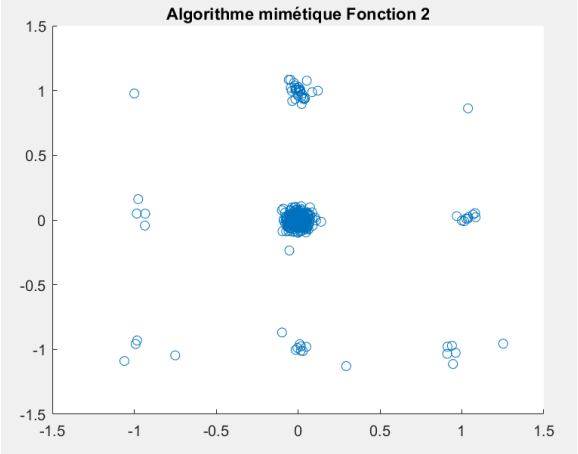
$$f(x,y) = 20 + x^2 + y^2 - 10(\cos(2\pi x) + \cos(2\pi y))$$



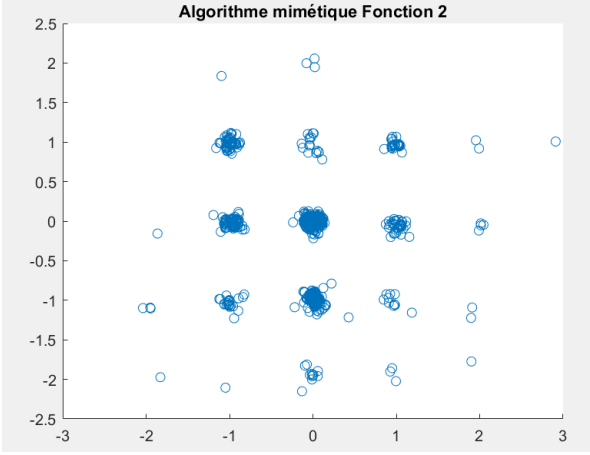
On visualise la population tel que :



On a donc une convergence vers le point (0,0) qui est le minimum global de notre fonction. Si on diminue sur le nombre d'itérations, on peut faire apparaître les minima locaux, c'est-à-dire les individus qui s'accumulent en clusters avant d'être comparés avec le minimum global réel. Voici :



Pour 500 individus (points) tirés, et 4000 itérations.



Pour 1000 individus, et 5000 itérations

Ici les clusters sont déséquilibrés donc on voit clairement que (0,0) sera sans doute le minimum global mais, en fonction du nombre d'individus qui s'agglutinent, on peut déduire le second minimum (celui au milieu en haut), puis le troisième… Plus on prend d'individus, plus on est à même d'affirmer avec certitude ces minimums.

Animations

On peut montrer le phénomène de convergence via une animation de l'évolution de la population. Les deux animations pour le minimum et le maximum sont joints (evolution_maximum.avi & evolution_minimum.avi).