Algorithmes et Aléatoire

Jean-Marc.Vincent@univ-grenoble-alpes.fr[1]

[1] Laboratoire LIG
Équipe-Projet INRIA POLARIS
Université Grenoble-Alpes

L3-INFO : Algorithmique et Modélisation
Grenoble 2024

I. LE CALCUL PROBABILISTE : calculer avec de l'aléatoire
II. GÉNÉRATION produire du hasard
III. MODÈLE : calculer avec de l'aléatoire
IV. LOIS UNIFORMES : transformation de l'aléatoire
V. TYPES D'ALGORITHMES : usage de la randomisation
VI. ARITHMÉTIQUE : nombres premiers

UGA Université Grenoble Alpes

QUI MET LA TABLE ?

Décision équitable

QUI MET LA TABLE ?

Décision équitable

QUI MET LA TABLE ?

Décision équitable ?

QUI MET LA TABLE ?

Décision équitable ?

Simuler une *pièce équitable* avec une pièce biaisée

QUI MET LA TABLE ?

Décision équitable ?

Simuler une *pièce équitable* avec une pièce biaisée

Pile

Face

QUI MET LA TABLE ?

Décision équitable ?

Simuler une *pièce équitable* avec une pièce biaisée

Pile

Face

Indication : lancer plusieurs fois la pièce

PIÈCE BIAISÉE

Lancer 2 fois la pièce








PIÈCE BIAISÉE

Lancer 2 fois la pièce


$\Rightarrow$ **Relancer**






PIÈCE BIAISÉE

Lancer 2 fois la pièce


$\Rightarrow$ **Relancer**


$\Rightarrow$ **Relancer**




PIÈCE BIAISÉE

Lancer 2 fois la pièce


$\Rightarrow$ **Relancer**


$\Rightarrow$ **Relancer**


$\Rightarrow$ **Renvoyer pile**


UGA Université Grenoble Alpes

PIÈCE BIAISÉE

Lancer 2 fois la pièce


$\Rightarrow$ **Relancer**


$\Rightarrow$ **Relancer**


$\Rightarrow$ **Renvoyer pile**


$\Rightarrow$ **Renvoyer Face**

RÉFÉRENCES

Quel est le scientifique ?

in testing for either even or odd. To cite a human example, for simplicity, in tossing a coin it is probably easier to make two consecutive tosses independent than to toss heads with probability exactly one-half. If independence of successive tosses is assumed, we can reconstruct a 50–50 chance out of even a badly biased coin by tossing twice. If we get heads-heads or tails-tails, we reject the tosses and try again. If we get heads-tails (or tails-heads), we accept the result as heads (or tails). The resulting process is rigorously unbiased, although the amended process is at most 25 percent as efficient as ordinary coin-tossing.

RÉFÉRENCES

Quel est le scientifique ?

in testing for either even or odd. To cite a human example, for simplicity, in tossing a coin it is probably easier to make two consecutive tosses independent than to toss heads with probability exactly one-half. If independence of successive tosses is assumed, we can reconstruct a 50–50 chance out of even a badly biased coin by tossing twice. If we get heads-heads or tails-tails, we reject the tosses and try again. If we get heads-tails (or tails-heads), we accept the result as heads (or tails). The resulting process is rigorously unbiased, although the amended process is at most 25 percent as efficient as ordinary coin-tossing.

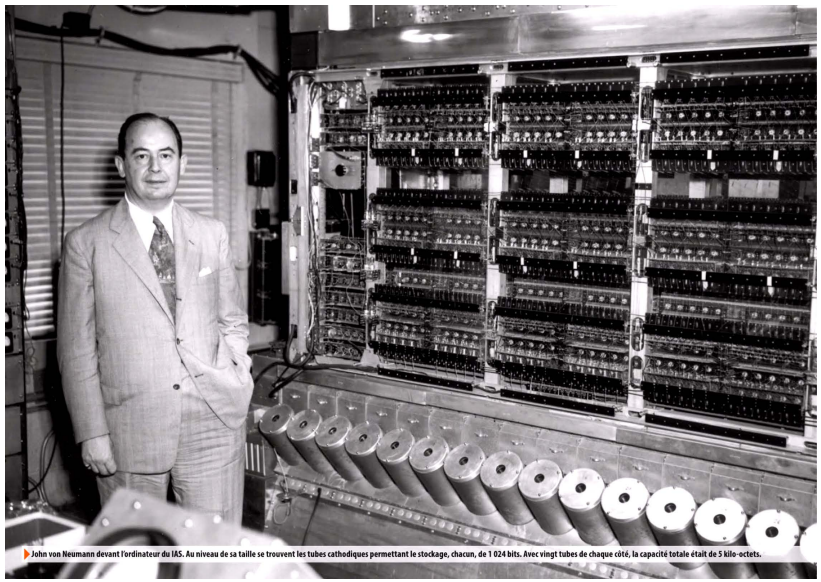
Various Techniques Used in Connection With Random Digits

By John von Neumann

Summary written by George E. Forsythe

J. Res. Nat. Bur. Stand. Appl. Math. Series 3, 36-38 (1951)

Biographie



John von Neumann devant l'ordinateur du IAS. Au niveau de sa taille se trouvent les tubes cathodiques permettant le stockage, chacun, de 1 024 bits. Avec vingt tubes de chaque côté, la capacité totale était de 5 kilo-octets.

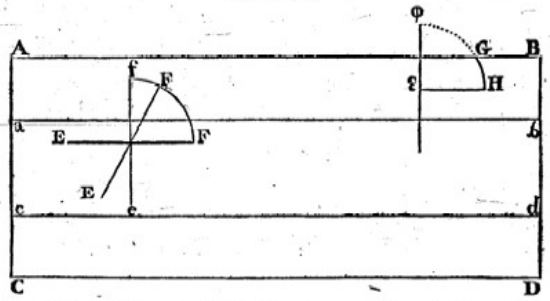
- Générateur du milieu du carré
- Méthode de rejet (1947)
-

ALGORITHME RANDOMISÉ

Une aiguille sur le parquet

est simplement divisé par des joints parallèles, on jette en l'air une baguette, & que l'un des joueurs parie que la baguette ne croisera aucune des parallèles du parquet, & que l'autre au contraire parie que la baguette croisera quelques-unes de ces parallèles; on demande le sort de ces deux joueurs. *On peut jouer ce jeu sur un damier avec une aiguille à coudre ou une épingle sans tête.*

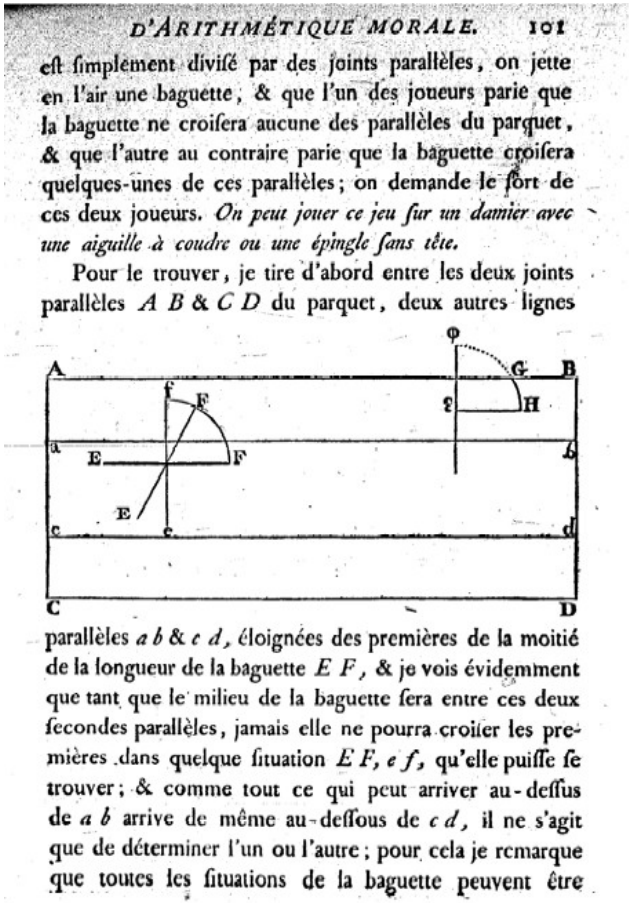
Pour le trouver, je tire d'abord entre les deux joints parallèles *A B* & *C D* du parquet, deux autres lignes



parallèles *a b* & *c d*, éloignées des premières de la moitié de la longueur de la baguette *E F*, & je vois évidemment que tant que le milieu de la baguette sera entre ces deux secondes parallèles, jamais elle ne pourra croiser les premières dans quelque situation *E F*, *e f*, qu'elle puisse se trouver; & comme tout ce qui peut arriver au-dessus de *a b* arrive de même au-dessous de *c d*, il ne s'agit que de déterminer l'un ou l'autre; pour cela je remarque que toutes les situations de la baguette peuvent être

ALGORITHME RANDOMISÉ

Une aiguille sur le parquet



D'ARITHMÉTIQUE MORALE. 101

eſt ſimplement diviſé par des joints parallèles, on jette en l'air une baguette, & que l'un des joueurs parie que la baguette ne croiſera aucune des parallèles du parquet, & que l'autre au contraire parie que la baguette croiſera quelques-unes de ces parallèles; on demande le ſort de ces deux joueurs. *On peut jouer ce jeu ſur un damier avec une aiguille à coudre ou une épingle ſans tête.*

Pour le trouver, je tire d'abord entre les deux joints parallèles *A B* & *C D* du parquet, deux autres lignes

parallèles *a b* & *c d*, éloignées des premières de la moitié de la longueur de la baguette *E F*, & je vois évidemment que tant que le milieu de la baguette ſera entre ces deux ſecondes parallèles, jamais elle ne pourra croiſer les premières dans quelque ſituation *E F*, *e f*, qu'elle puiſſe ſe trouver; & comme tout ce qui peut arriver au-deſſus de *a b* arrive de même au-deſſous de *c d*, il ne s'agit que de déterminer l'un ou l'autre; pour cela je remarque que toutes les ſituations de la baguette peuvent être

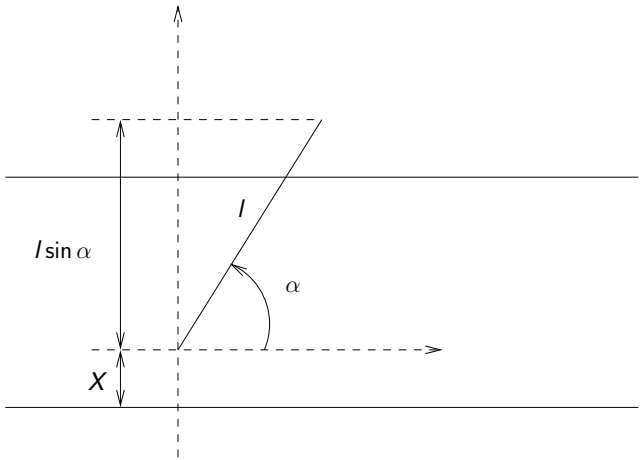
Buffon (1707-1788) - biographie

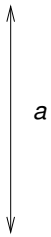
Georges-Louis Leclerc, comte de Buffon (7 septembre 1707 à Montbard - 16 avril 1788 à Paris), est un naturaliste, mathématicien, biologiste, cosmologiste et écrivain français. Ses théories ont influencé deux générations de naturalistes, parmi lesquels notamment Jean-Baptiste de Lamarck et Charles Darwin. La localité éponyme Buffon, dans la Côte-d'Or, fut la seigneurie de la famille Leclerc.
Les premiers travaux de Buffon ont été consacrés aux mathématiques. Il faut surtout signaler le Mémoire sur le jeu de franc carreau, qui présente l'originalité de faire intervenir le calcul infinitésimal dans le calcul des probabilités. Par la suite, Buffon utilisera les mathématiques dans ses recherches sur la résistance du bois et sur le refroidissement des planètes, ainsi que dans son Essai d'arithmétique morale (Supplément, t. IV, 1777), mais ces travaux montrent que, pour lui, les mathématiques ne sont qu'un moyen de préciser l'idée qu'il peut avoir des choses, et non une discipline autonome. Il est ingénieur plus que mathématicien. Par contre, il est philosophe de tempérament. Le tome I de l'Histoire naturelle (1749) s'ouvre par un discours De la manière d'étudier et de traiter l'histoire naturelle, qui est une réflexion sur la valeur de la connaissance humaine. Rompant à la fois avec l'idéalisme rationaliste et l'empirisme sceptique, Buffon affirme la validité d'une science fondée sur les faits, mais sachant en dégager les lois, débarassée de toute téléologie, d'une science qui sans doute ne vaut que pour l'homme, mais qui est la seule que l'homme puisse atteindre. Par la suite, Buffon admettra que l'homme peut découvrir les vraies lois de la nature (De la nature, 1re et 2e vues, Histoire naturelle, t. XII et XIII, 1764-1765). Son tempérament rationaliste l'emporte alors sur sa formation philosophique, d'inspiration sceptique.

UGA Université Grenoble Alpes

MODELISATION

Notations





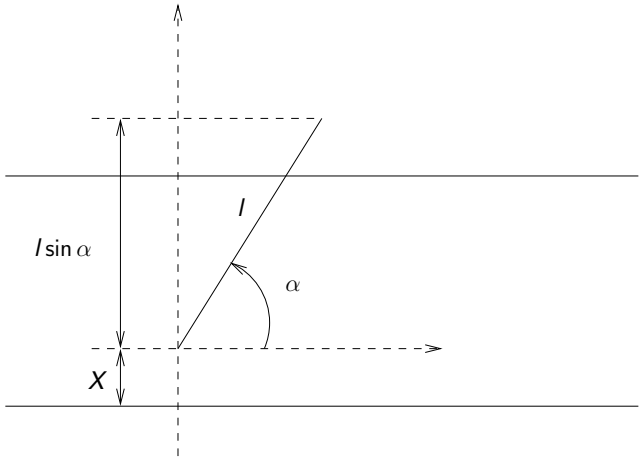
X de loi uniforme sur $[0 : l]$

α de loi uniforme sur $[0, \pi]$

$\mathbb{P}(X + l \sin \alpha \geqslant a)$

MODELISATION

Notations



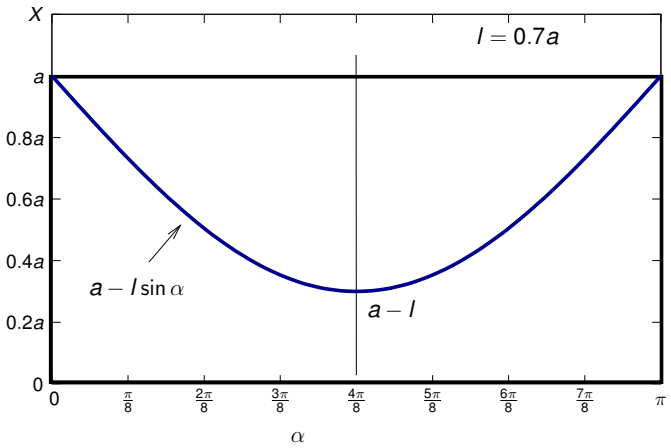
X de lc

α de lo

a

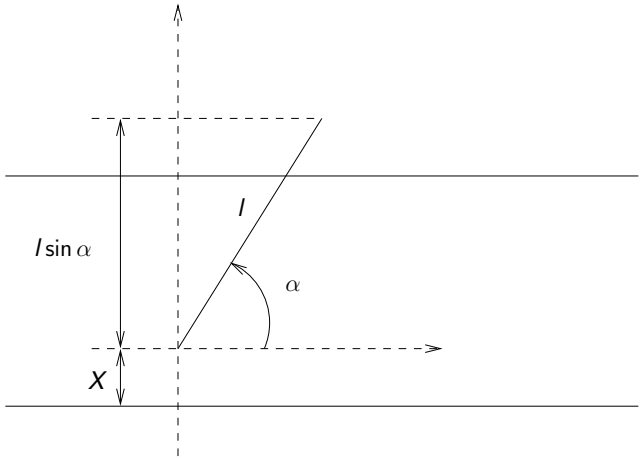
$\mathbb{P}(X +$

Représentation par une aire



MODELISATION

Notations



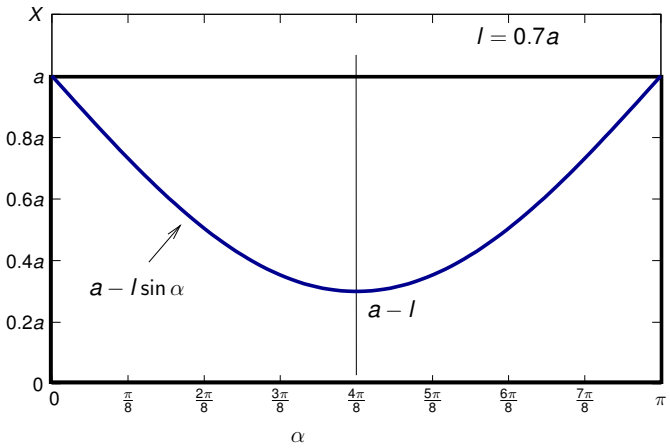
X de lc

α de lo

a

$\mathbb{P}(X +$

Représentation par une aire

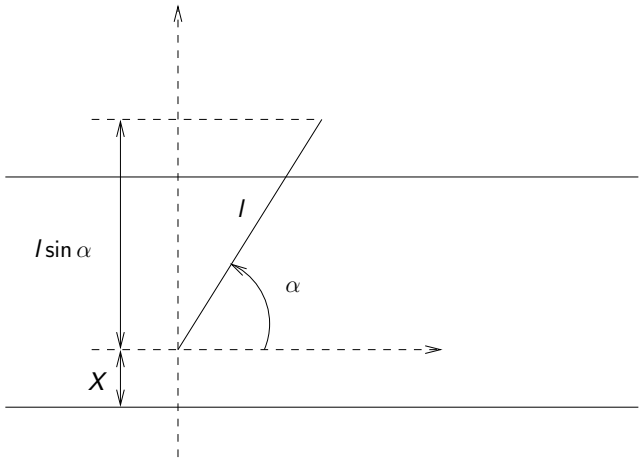


Calcul

$$
\begin{aligned}
\mathbb{P}(X + l \sin\alpha \geqslant a) &= 1 - \mathbb{P}(X + l \sin\alpha \leqslant a); \\
&= 1 - \mathbb{P}(X \leqslant a - l \sin\alpha); \\
&= 1 - \frac{\text{Surface sous la courbe bleue}}{\text{Surface du rectangle}}; \\
&= 1 - \frac{1}{\pi a}\int_0^{\pi} (a - l \sin\alpha) d\alpha; \\
&= 1 - \frac{1}{\pi a}[a.\alpha + l \cos\alpha]_0^{\pi} = \frac{2l}{\pi a}.
\end{aligned}
$$

MODELISATION

Notations



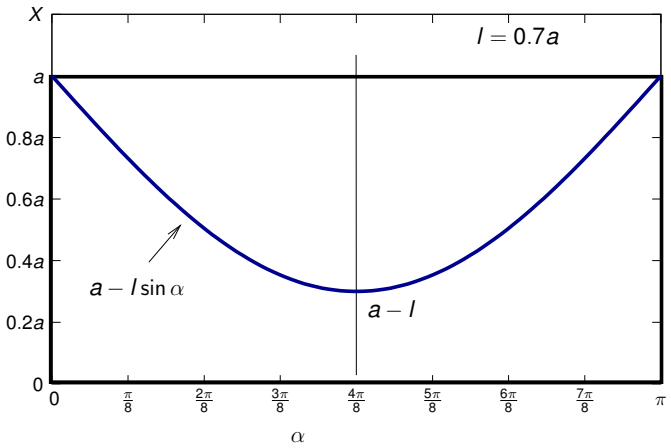
X de lc

α de lo

a

$\mathbb{P}(X +$

Représentation par une aire



Probabilité de toucher :

$$\mathbb{P}(X + l \sin \alpha \geqslant a) = \frac{2l}{\pi a}.$$

Calcul en répétant l'expérience un *grand* nombre de fois
(Loi des grands nombres, théorème de la limite centrale)
Méthodes de Monte-Carlo

Calcul

$$\begin{aligned}
\mathbb{P}(X + l \sin \alpha \geqslant a) &= 1 - \mathbb{P}(X + l \sin \alpha \leqslant a); \\
&= 1 - \mathbb{P}(X \leqslant a - l \sin \alpha); \\
&= 1 - \frac{\text{Surface sous la courbe bleue}}{\text{Surface du rectangle}}; \\
&= 1 - \frac{1}{\pi a} \int_0^{\pi} (a - l \sin \alpha) d\alpha; \\
&= 1 - \frac{1}{\pi a} [a.\alpha + l \cos \alpha]_0^{\pi} = \frac{2l}{\pi a}.
\end{aligned}$$

MÉTHODE DE MONTE-CARLO

Schéma numérique d'intégration

Manhattan project Simulation de réactions nucléaires $\Rightarrow$ équations aux dérivées partielles

$$I = \int_a^b f(x)dx \simeq \frac{1}{N}\sum_{i=1} f(U_i).$$

Stanislaw Marcin Ulam (math)
Enrico Fermi (physique)
John von Neumann (math app)
Nicholas Metropolis (physique)

MÉTHODE DE MONTE-CARLO

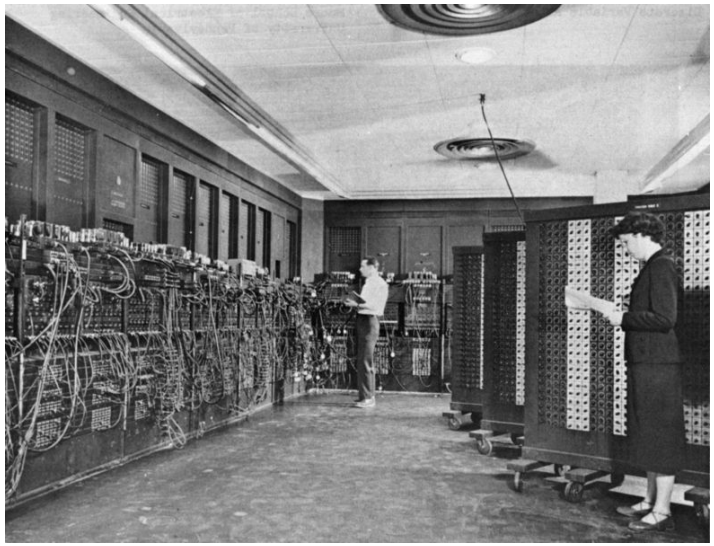
Schéma numérique d'intégration

Manhattan project Simulation de réactions nucléaires $\Rightarrow$ équations aux dérivées partielles

$$I = \int_a^b f(x)dx \simeq \frac{1}{N}\sum_{i=1} f(U_i).$$

Stanislaw Marcin Ulam (math)
Enrico Fermi (physique)
John von Neumann (math app)
Nicholas Metropolis (physique)

Ordinateur : Eniac 1943



Electronic Numerical Integrator And Computer
John Mauchly et J. Presper Eckert
University of Pennsylvania.

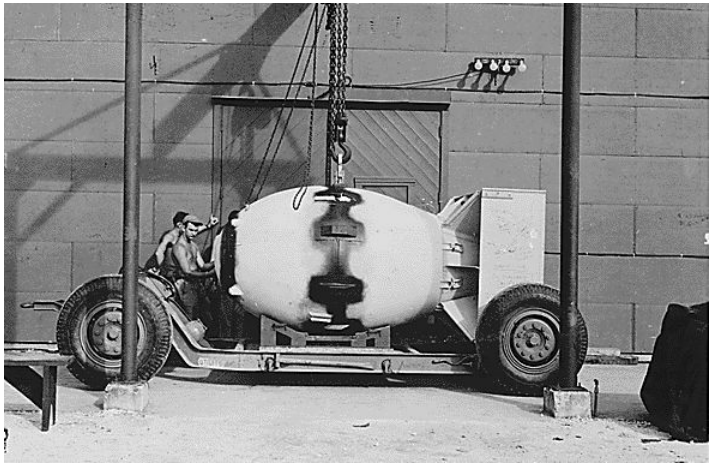
JOHN VON NEUMAN (1903-1957)

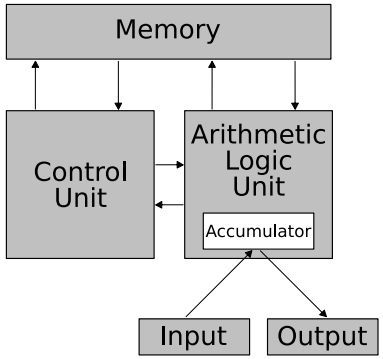
Mathématicien américain d'origine hongroise. Il a apporté d'importantes contributions tant en mécanique quantique, qu'en analyse fonctionnelle, en théorie des ensembles, en informatique, en sciences économiques ainsi que dans beaucoup d'autres domaines des mathématiques et de la physique. Il a de plus participé aux programmes militaires américains.

JOHN VON NEUMAN (1903-1957)

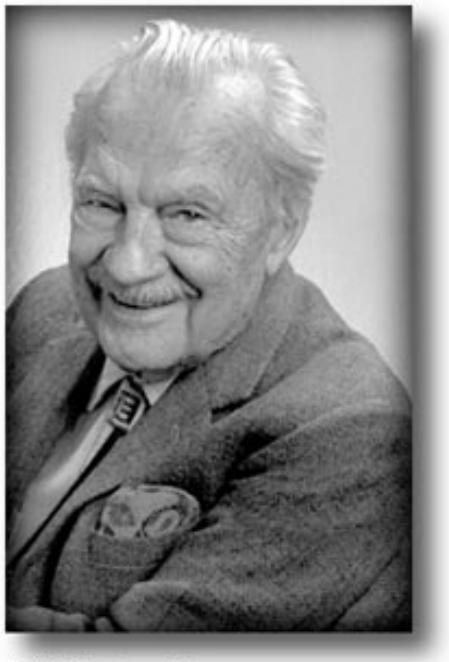
Mathématicien américain d'origine hongroise. Il a apporté d'importantes contributions tant en mécanique quantique, qu'en analyse fonctionnelle, en théorie des ensembles, en informatique, en sciences économiques ainsi que dans beaucoup d'autres domaines des mathématiques et de la physique. Il a de plus participé aux programmes militaires américains.

Architecture du processeur



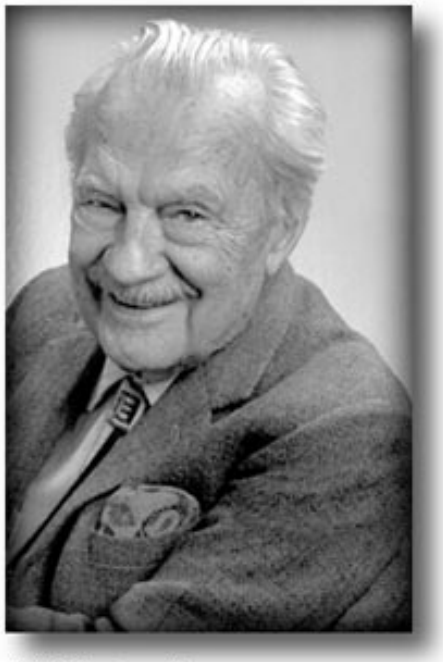


NICHOLAS METROPOLIS (1915-1999)


Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

NICHOLAS METROPOLIS (1915-1999)

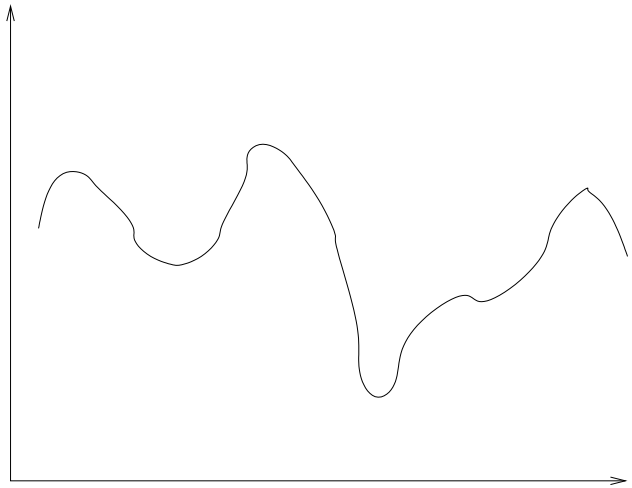

Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

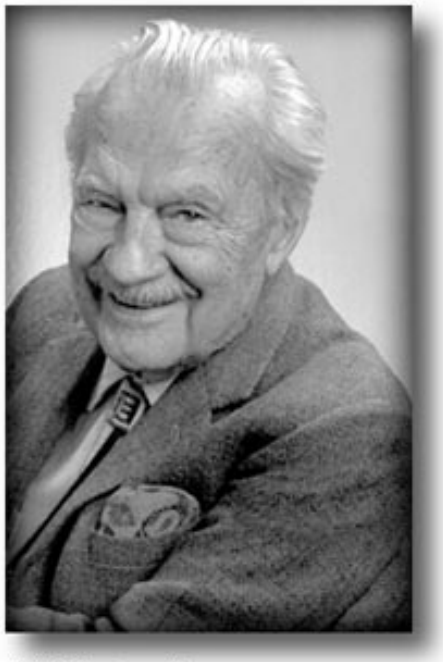
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



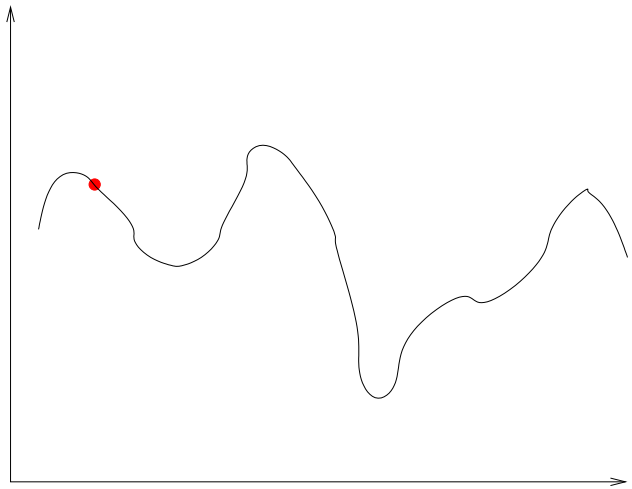
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

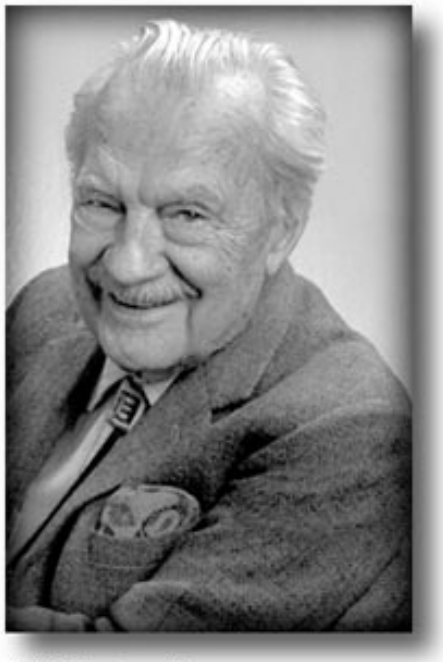
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



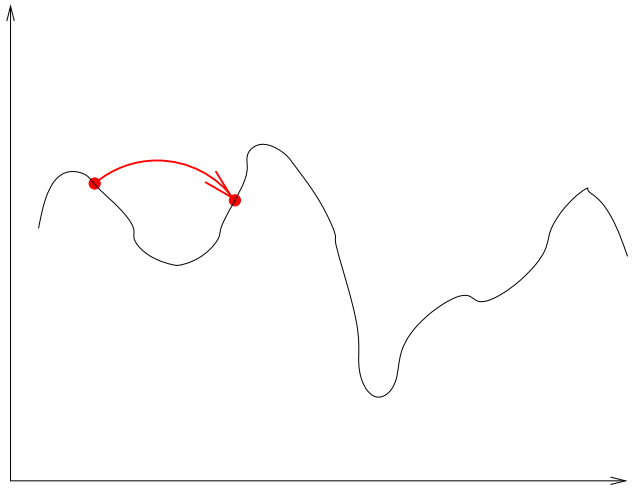
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

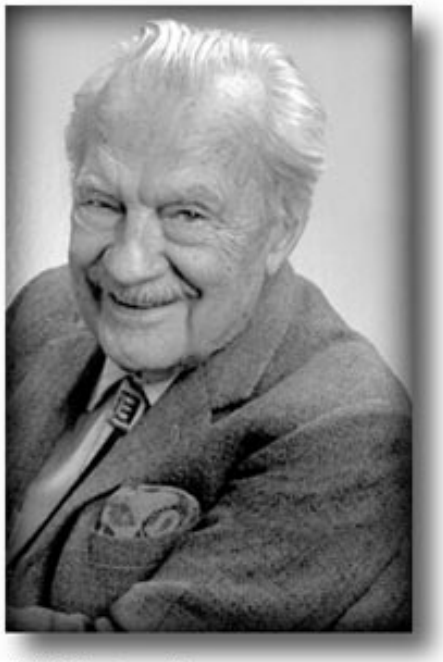
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)

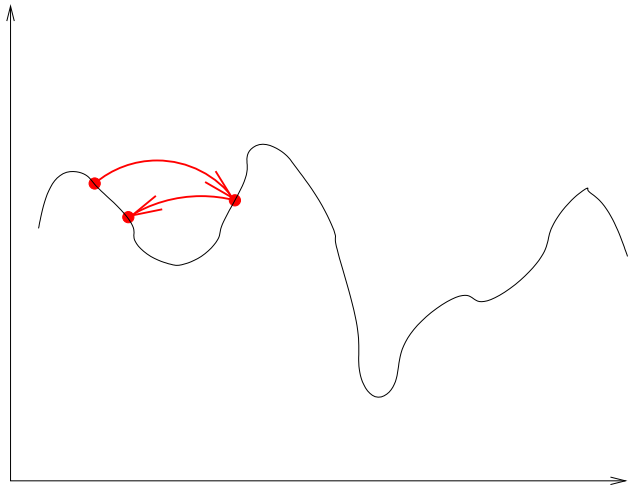

Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

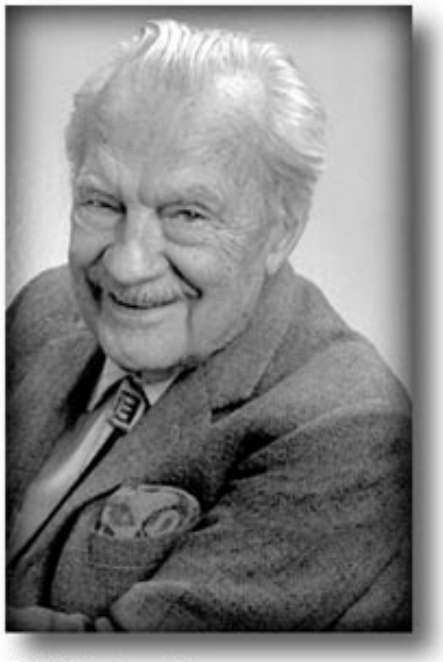
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



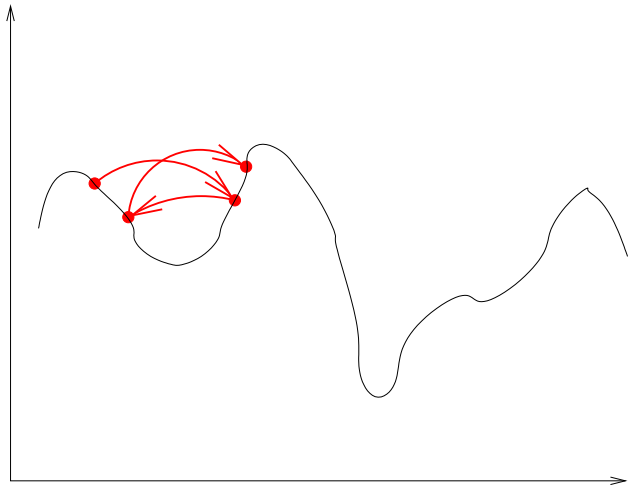
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

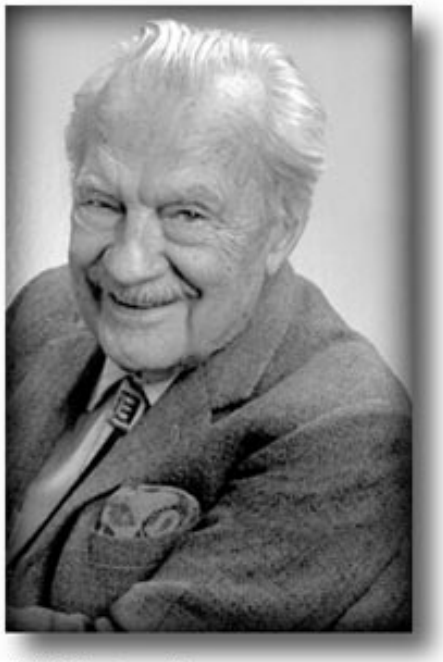
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



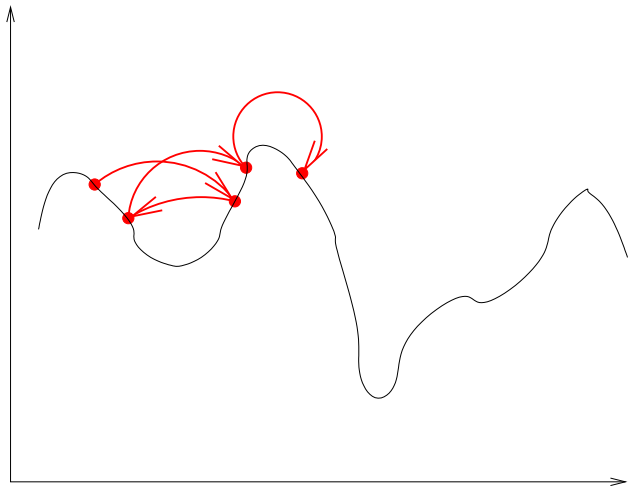
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

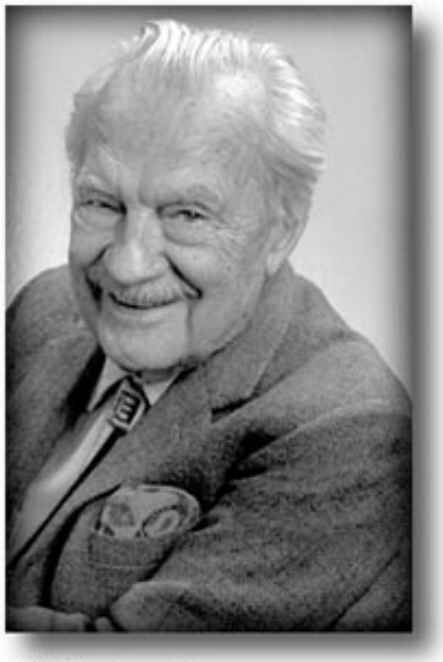
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



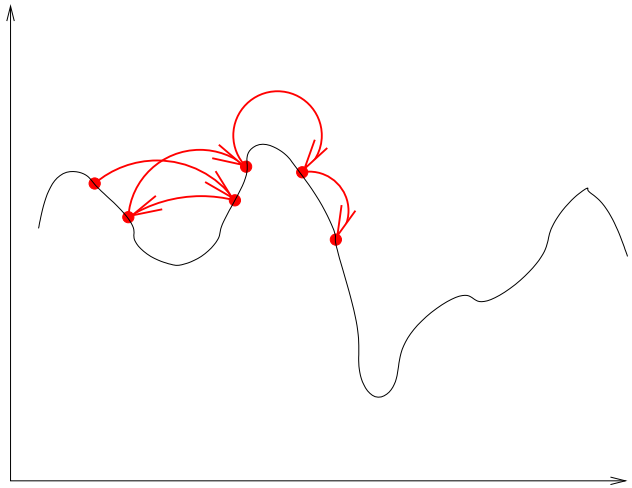
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

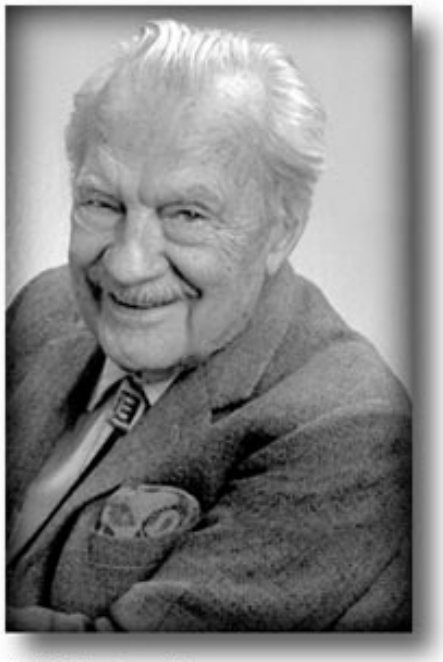
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



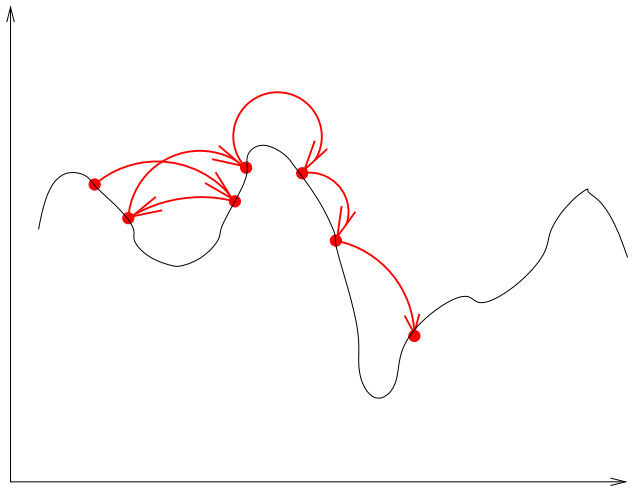
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

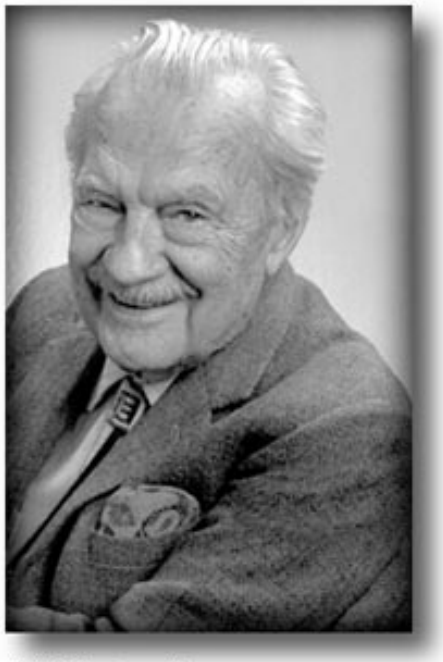
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



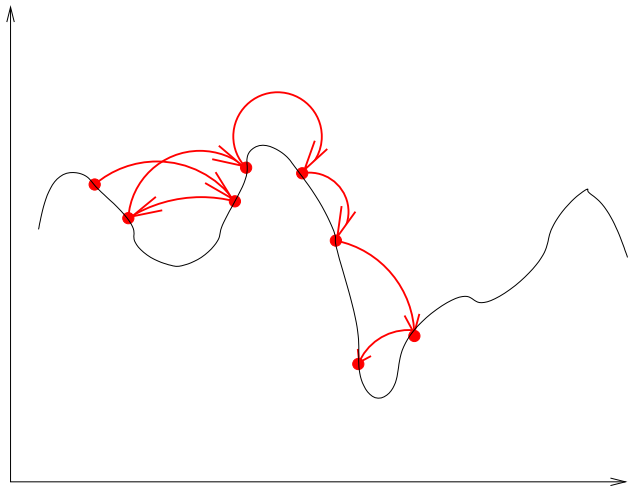
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

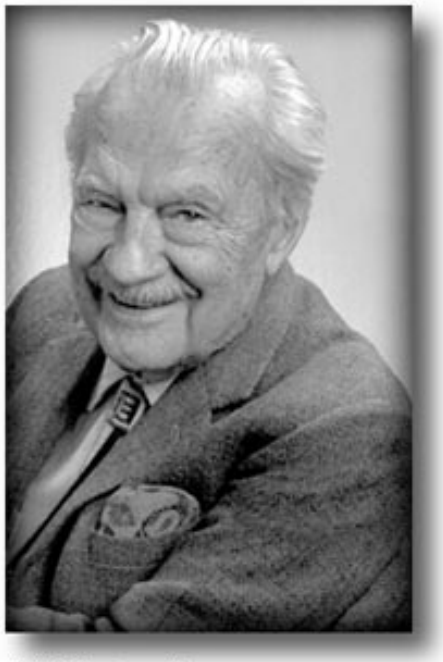
Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



NICHOLAS METROPOLIS (1915-1999)



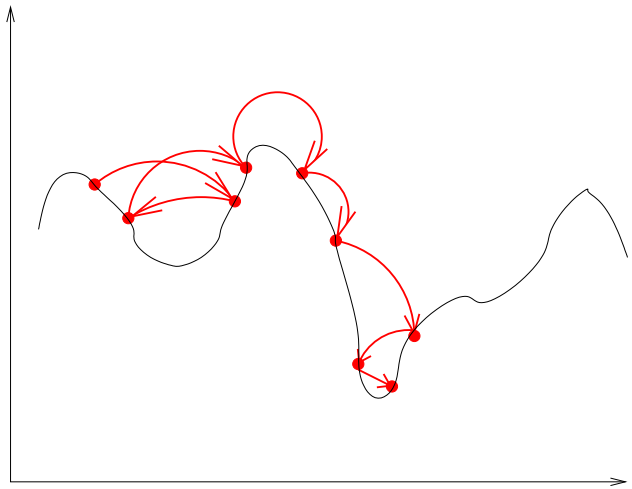
Nick Metropolis

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis-Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.

Recuit simulé

Convergence vers un minimum global par une descente de gradient **stochastique**

$$X_{n+1} = X_n - \vec{grad}\Phi(X_n)\Delta(Random).$$



MACHINES ALÉATOIRES

Loterie

- Roue à secteurs
  Système chaotique amorti
- Pièces et dés
  Système chaotique avec sngularités
- Roulette, loto,...
- Mélange de cartes

graine : action d'une personne

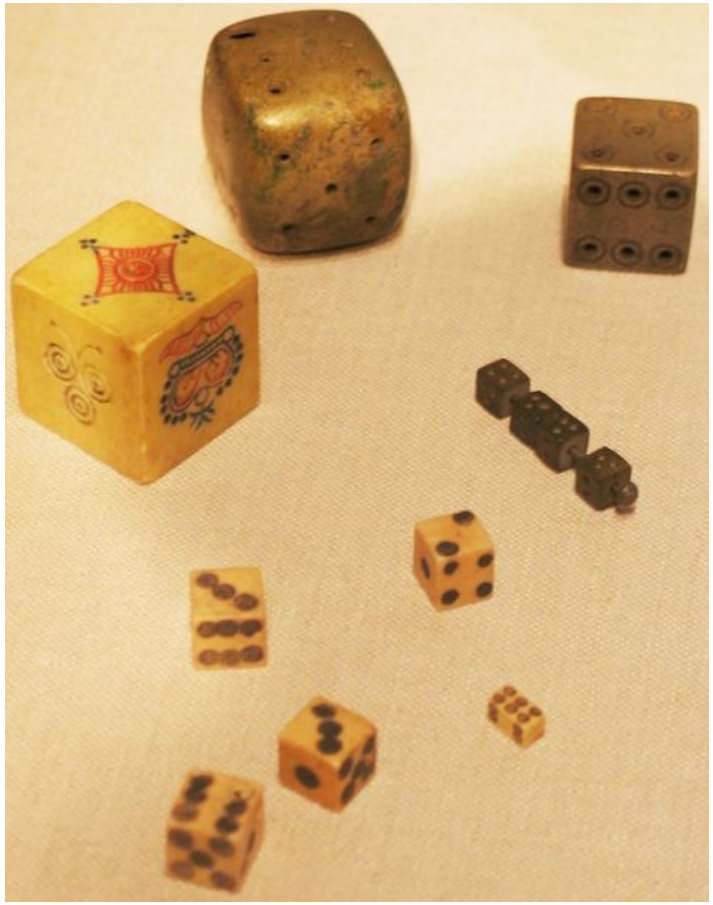
MACHINES ALÉATOIRES

Loterie

- Roue à secteurs
  Système chaotique amorti
- Pièces et dés
  Système chaotique avec sngularités
- Roulette, loto,...
- Mélange de cartes

graine : action d'une personne

Machines



ALÉATOIRE PRÉCALCULÉ

Rand corporation

Tables de nombres aléatoires

TABLE OF RANDOM DIGITS 261

13000	31726	01273	30818	23348	66905	55238	72887	45155	43316	11849
13001	59971	74215	36373	38132	71727	31941	39503	70144	96580	68006
13002	82251	26199	04946	24067	75828	72756	41805	62250	57512	99123
13003	77246	73455	17182	45771	89843	91330	77118	22870	36156	75891
13004	78873	64900	83492	45705	34186	83778	12290	17114	73589	37874
13005	54333	60475	16486	71100	54383	61838	39952	10178	34692	78547
13006	87210	45684	55500	32678	80868	03192	90866	75951	84050	88751
13007	35977	62057	35434	83696	52874	37732	31086	28169	79356	06036
13008	01798	05037	16853	85295	32464	15527	34792	45374	96984	41393
13009	28667	52168	96342	03730	28884	65518	91683	16757	73430	00848
13010	94504	03297	22697	87667	66015	56657	51227	35046	95545	71107
13011	89247	33522	44274	70790	92271	14420	19370	58305	07202	58986
13012	44432	85735	54509	66326	66220	04781	35731	74739	73897	93590
13013	53126	06561	71384	46293	97522	59633	27125	38801	33653	69624
13014	30621	85366	20868	18539	11538	08753	50963	38275	89449	66655
13015	80369	17972	93322	43948	65247	44771	04393	05460	27082	65286
13016	17497	46460	10430	98540	23064	52408	85005	71738	10212	05245
13017	80133	11188	56128	77467	30073	45780	20957	98680	47497	28685
13018	73527	74518	22980	30180	93817	00211	46282	28130	74189	89850
13019	08833	55503	24680	77715	71666	11331	18399	43678	85088	70590
13020	10858	14667	48980	86271	66747	79393	00990	11562	30219	06821
13021	61508	44310	30763	03362	20728	84849	85428	39827	71865	57280
13022	08293	26000	01555	88968	77354	67517	52733	85737	54643	29087
13023	30335	35992	46117	77484	38287	61494	78036	29525	95306	13961
13024	49751	04834	50324	90597	66374	18474	75790	13241	67204	83896
13025	14221	07945	62399	55232	25943	49355	45032	07555	36514	67805
13026	83101	00048	19592	05703	10632	44462	45491	77891	48988	28486
13027	55326	73308	17424	31926	51652	51438	12768	82177	78183	13485
13028	52820	34330	44858	49399	51734	61714	41469	64339	11999	21959
13029	23058	87173	31622	34329	85828	36494	15604	68594	66564	89627
13030	81463	87234	33416	45804	39256	21555	30259	79995	81974	33160
13031	07587	88348	87702	79839	09002	34181	80314	80144	72716	56288
13032	56833	83496	37037	88661	14114	25973	38708	42082	82602	17870
13033	45309	32040	99980	41798	92299	49815	02879	79355	91723	18657
13034	93724	77977	35611	83966	79874	91933	81586	08218	37238	97202
13035	21079	36714	11486	83300	88004	39467	95906	02664	01269	54713
13036	21352	95212	66334	24724	00572	87243	88020	99676	97899	86938
13037	84697	10199	08939	98579	33302	77186	07999	65278	82898	92532
13038	68041	35472	05070	23534	93191	30182	71955	87993	06673	76836
13039	72555	15689	72605	50364	43733	62670	29268	84828	27420	51240
13040	76663	72429	39756	24251	28109	63102	17593	49260	65206	37665
13041	71084	87584	70605	90069	43316	06041	25815	72542	41805	24667
13042	94987	48500	04702	39088	70985	66515	15760	76309	68260	09027
13043	13359	86002	64876	68263	20117	80277	47612	87109	07403	13052
13044	25740	20603	17977	02314	19017	61518	52720	00720	71495	71240
13045	04347	07656	07387	27503	60180	05298	01114	35889	39619	97793
13046	85829	57570	21017	61880	55920	27275	29915	88156	28033	51902
13047	16412	94371	31427	28452	50556	24001	98895	09549	16003	13469
13048	12406	75609	21909	82654	27759	90136	51207	16894	20893	06560
13049	26123	81313	92450	84101	00991	20881	23383	56838	64492	26914

UGA Université Grenoble Alpes

ALÉATOIRE PRÉCALCULÉ

Rand corporation

Tables de nombres aléatoires

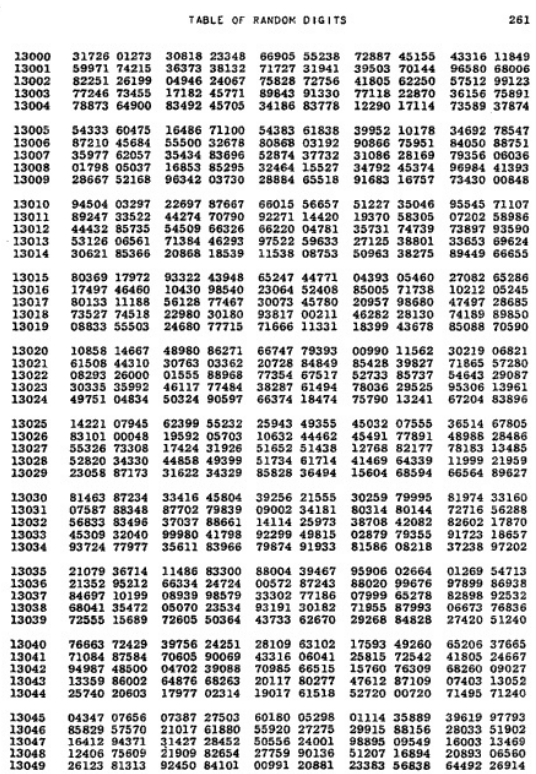


TABLE OF RANDOM DIGITS 261

13000	31726	01273	30818	23348	66905	55238	72887	45155	43316	11849
13001	59971	74215	36373	38132	71727	31941	39503	70144	96580	68006
13002	82251	26199	04946	24067	75828	72756	41805	62250	57512	99123
13003	77246	73455	17182	45771	89843	91330	77118	22870	36156	75891
13004	78873	64900	83492	45705	34186	83778	12290	17114	73589	37874
13005	54333	60475	16486	71100	54383	61838	39952	10178	34692	78547
13006	87210	45684	55500	32678	80868	03192	90866	75951	84050	88751
13007	35977	62057	35434	83696	52874	37732	31086	28169	79356	06036
13008	01798	05037	16853	85295	32464	15527	34792	45374	96984	41393
13009	28667	52168	96342	03730	28884	65518	91683	16757	73430	00848
13010	94504	03297	22697	87667	66015	56657	51227	35046	95545	71107
13011	89247	33522	44274	70790	92271	14420	19370	58305	07202	58986
13012	44432	85735	54509	66326	66220	04781	35731	74739	73897	93590
13013	53126	06561	71384	46293	97522	59633	27125	38801	33653	69624
13014	30621	85366	20868	18539	11538	08753	50963	38275	89449	66655
13015	80369	17972	93322	43948	65247	44771	04393	05460	27082	65286
13016	17497	46460	10430	98540	23064	52408	85005	71738	10212	05245
13017	80133	11188	56128	77467	30073	45780	20957	98680	47497	28685
13018	73527	74518	22980	30180	93817	00211	46282	28130	74189	89850
13019	08833	55503	24680	77715	71666	11331	18399	43678	85088	70590
13020	10858	14667	48980	86271	66747	79393	00990	11562	30219	06821
13021	61508	44310	30763	03362	20728	84849	85428	39827	71865	57280
13022	08293	26000	01555	88968	77354	67517	52733	85737	54643	29087
13023	30335	35992	46117	77484	38287	61494	78036	29525	95306	13961
13024	49751	04834	50324	90597	66374	18474	75790	13241	67204	83896
13025	14221	07945	62399	55232	25943	49355	45032	07555	36514	67805
13026	83101	00048	19592	05703	10632	44462	45491	77891	48988	28486
13027	55326	73308	17424	31926	51652	51438	12768	82177	78183	13485
13028	52820	34330	44858	49399	51734	61714	41469	64339	11999	21959
13029	23058	87173	31622	34329	85828	36494	15604	68594	66564	89627
13030	81463	87234	33416	45804	39256	21555	30259	79995	81974	33160
13031	07587	88348	87702	79839	09002	34181	80314	80144	72716	56288
13032	56833	83496	37037	88661	14114	25973	38708	42082	82602	17870
13033	45309	32040	99980	41798	92299	49815	02879	79355	91723	18657
13034	93724	77977	35611	83966	79874	91933	81586	08218	37238	97202
13035	21079	36714	11486	83300	88004	39467	95906	02664	01269	54713
13036	21352	95212	66334	24724	00572	87243	88020	99676	97899	86938
13037	84697	10199	08939	98579	33302	77186	07999	65278	82898	92532
13038	68041	35472	05070	23534	93191	30182	71955	87993	06673	76836
13039	72555	15689	72605	50364	43733	62670	29268	84828	27420	51240
13040	76663	72429	39756	24251	28109	63102	17593	49260	65206	37665
13041	71084	87584	70605	90069	43316	06041	25815	72542	41805	24667
13042	94987	48500	04702	39088	70985	66515	15760	76309	68260	09027
13043	13359	86002	64876	68263	20117	80277	47612	87109	07403	13052
13044	25740	20603	17977	02314	19017	61518	52720	00720	71495	71240
13045	04347	07656	07387	27503	60180	05298	01114	35889	39619	97793
13046	85829	57570	21017	61880	55920	27275	29915	88156	28033	51902
13047	16412	94371	31427	28452	50556	24001	98895	09549	16003	13469
13048	12406	75609	21909	82654	27759	90136	51207	16894	20893	06560
13049	26123	81313	92450	84101	00991	20881	23383	56838	64492	26914

Random bits

- RandomNumber.org
- Hotbits
  http://www.fourmilab.ch/hotbits/
- Marsaglia's generator
  http://www.stat.fsu.edu/pub/diehard/
- New dice roller : http://gamesbyemail.com/News/DiceOMatic
- etc.

UGA Université Grenoble Alpes

PSEUDO-ALÉATOIRE EN PYTHON

Python

Librairie random
Générateur : Mersenne Twister (1998)

$$x_n = x_{n-(N-M)} \oplus (x^U_{n-N} | x^L_{n-N+1})A$$

avec le jeu de paramètres adéquat :
période = $2^{19937} - 1$

- random() génère un nombre pseudo-aléatoire (réel) dans $[0, 1[$
- seed() initialise le générateur (reproductibilité) (temps système par défaut)
- getstate() état interne du générateur
- setstate() réinitialise le générateur à partir d'un état interne

PSEUDO-ALÉATOIRE EN PYTHON

Python

Librairie random
Générateur : Mersenne Twister (1998)

$$x_n = x_{n-(N-M)} \oplus (x^U_{n-N} | x^L_{n-N+1})A$$

avec le jeu de paramètres adéquat :
période = $2^{19937} - 1$

- random() génère un nombre pseudo-aléatoire (réel) dans $[0, 1[$
- seed() initialise le générateur (reproductibilité) (temps système par défaut)
- getstate() état interne du générateur
- setstate() réinitialise le générateur à partir d'un état interne

Utilisation

- randrange(a,b)
  nombre entier $a \leqslant k < b$
- randint(a,b)
  nombre entier $a \leqslant k \leqslant b$
- choice(liste)
- shuffle (liste)
  mélange en place
- sample(population,k)
  selection sans remplacement
- lois classiques de probabilité
  normale, exponentielle, triangulaire,...

UGA Université Grenoble Alpes

MODÈLE DE MACHINE AVEC DE L'ALÉATOIRE

Modèle de machine :

- machine de Turing (ou équivalente)
- source externe de bits aléatoires

Formalisation :

La séquence des appels à la fonction getrandbits() est modélisée par une suite de variables aléatoires réelles $\{X_k\}_{n\mathbb{N}}$ **indépendantes** et de même loi **uniforme** sur $E = \{0, 1\}$.

Pour tout n et toute séquence $a_1, \cdots, a_n$ de bits

$$\mathbb{P}(X_1 = a_1, \cdots, X_n = a_n) = \frac{1}{2^n}$$

On peut généraliser en convenant d'une représentation des réels sur k bits (random)

HISTOIRES DE DÉS (SUITE)

Pièce $\mapsto$ Dé-8

À partir d'une pièce de monnaie écrire un générateur aléatoire d'un dé à 8 faces :

HISTOIRES DE DÉS (SUITE)

Pièce $\mapsto$ Dé-8

À partir d'une pièce de monnaie écrire un générateur aléatoire d'un dé à 8 faces :

Dé-8()

Données: Une fonction "Pièce()" générateur aléatoire de $\{0, 1\}$
Résultat: Une séquence i.i.d. de loi uniforme sur $\{1, \cdots, 8\}$

$A_0 =$Pièce()
$A_1 =$Pièce()
$A_2 =$Pièce()
$S = A_0 + 2 * A_1 + 4 * A_2 + 1$
return S

HISTOIRES DE DÉS : PREUVE DE L'ALGORITHME

Spécification :
une séquence d'appels à la fonction **Dé-8()** est modélisée par une séquence de variables aléatoires i.i.d. de loi uniforme sur $\{1, \cdots, 8\}$.

HISTOIRES DE DÉS : PREUVE DE L'ALGORITHME

Spécification :
une séquence d'appels à la fonction **Dé-8()** est modélisée par une séquence de variables aléatoires i.i.d. de loi uniforme sur $\{1, \cdots, 8\}$.

Hypothèse :
$P_0, P_1, \cdots, P_n, \cdots$ séquence des appels à **Pièce()** iid de loi uniforme sur $\{0, 1\}$

HISTOIRES DE DÉS : PREUVE DE L'ALGORITHME

Spécification :
une séquence d'appels à la fonction **Dé-8()** est modélisée par une séquence de variables aléatoires i.i.d. de loi uniforme sur $\{1, \cdots, 8\}$.

Hypothèse :
$P_0, P_1, \cdots, P_n, \cdots$ séquence des appels à **Pièce()** iid de loi uniforme sur $\{0, 1\}$

Preuve :
On note $S_0, S_1, \cdots, S_n, \cdots$ la séquence de variables aléatoires modélisant les résultats obtenus par appels successifs de **Dé-8()**.
Soit $n \in \mathbb{N}$ et $(x_0, x_1, \cdots, x_n) \in \{1, \cdots, 8\}^n$. Il faut montrer que

$$\mathbb{P}(S_0 = x_0, \cdots, S_n = x_n) = \frac{1}{8^{n+1}}$$

HISTOIRES DE DÉS :PREUVE DE L'ALGORITHME (SUITE)

On a

$$\mathbb{P}(S_0 = x_0, \cdots, S_n = x_n)$$

$$= \mathbb{P}(S_0 = x_0) \cdots \mathbb{P}(S_n = x_n)$$

car S_k ne dépend que de $P_{3k}, P_{3k+1}, P_{3k+2}$ et que les P_i sont indépendants ;
les $S_0, \cdots, S_n, \cdots$ sont donc indépendants ;

$$= \mathbb{P}(S_0 = x_0) \cdots \mathbb{P}(S_0 = x_n) \text{ car } (P_{3k}, P_{3k+1}, P_{3k+2}) \text{ ont même loi.}$$

HISTOIRES DE DÉS :PREUVE DE L'ALGORITHME (SUITE)

On a

$$\mathbb{P}(S_0 = x_0, \cdots, S_n = x_n)$$

$$\begin{aligned} &= \mathbb{P}(S_0 = x_0) \cdots \mathbb{P}(S_n = x_n) \\ &\quad \text{car } S_k \text{ ne dépend que de } P_{3k}, P_{3k+1}, P_{3k+2} \text{ et que les } P_i \text{ sont indépendants ;} \\ &\quad \text{les } S_0, \cdots, S_n, \cdots \text{ sont donc indépendants ;} \\ &= \mathbb{P}(S_0 = x_0) \cdots \mathbb{P}(S_0 = x_n) \text{ car } (P_{3k}, P_{3k+1}, P_{3k+2}) \text{ ont même loi.} \end{aligned}$$

Or pour i dans $\{1, \cdots, 8\}$, $i-1$ s'écrit de manière unique en binaire $i - 1 =_2 a_2 a_1 a_0$.

$$\begin{aligned} \mathbb{P}(S_0 = i) &= \mathbb{P}(P_0 = a_0, P_1 = a_1, P_2 = a_2) \\ &= \mathbb{P}(P_0 = a_0)\mathbb{P}(P_1 = a_1)\mathbb{P}(P_2 = a_2) \text{ les appels à Piece() sont indépendants} \\ &= \frac{1}{2}\frac{1}{2}\frac{1}{2} = \frac{1}{8} \text{ car même loi uniforme sur } \{0, 1\}. \end{aligned}$$

d'où

$$\mathbb{P}(S_0 = x_0, \cdots, S_n = x_n) = \frac{1}{8^{n+1}} \qquad \text{cqfd.}$$

UGA Université Grenoble Alpes

HISTOIRES DE DÉS (SUITE)

Pièce $\mapsto$ Dé-2^k

À partir d'une pièce de monnaie écrire un générateur aléatoire d'un dé à 2^k faces :

HISTOIRES DE DÉS (SUITE)

Pièce $\mapsto$ Dé-2^k

À partir d'une pièce de monnaie écrire un générateur aléatoire d'un dé à 2^k faces :

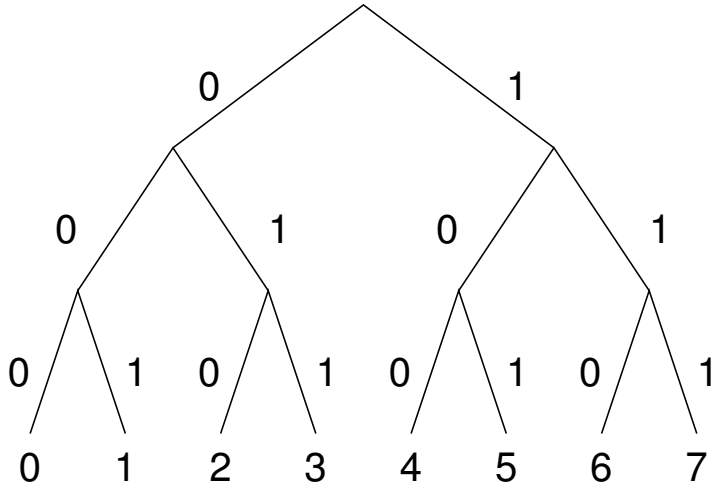
Dé(k)

Données: Une fonction "Pièce()" générateur aléatoire de $\{0,1\}$
Résultat: Une séquence i.i.d. de loi uniforme sur $\{1,\cdots,2^k\}$

S=0
for $i=1$ **to** k
  S=Pièce() +2.S `// cf Schéma de Hörner`
$S=S+1$
return S

Preuve : Identique au **Dé-8**, unicité de la décomposition binaire d'un entier de $\{0,\cdots,2^k-1\}$ par un vecteur de k bits.

REPRÉSENTATION BINAIRE :



$5 =_2 101,\ 2 =_2 010,\ 42 =_2 101010\cdots$

HISTOIRES DE DÉS (SUITE)

Pièce $\mapsto$ Dé-6

À partir d'une pièce de monnaie écrire un générateur aléatoire d'un dé à 6 faces :

HISTOIRES DE DÉS (SUITE)

Pièce $\mapsto$ Dé-6

À partir d'une pièce de monnaie écrire un générateur aléatoire d'un dé à 6 faces :

Dé-6()

Données: Une fonction **Dé-8()** générateur aléatoire de $\{1, \cdots, 8\}$
Résultat: Une séquence i.i.d. de loi uniforme sur $\{1, \cdots, 6\}$

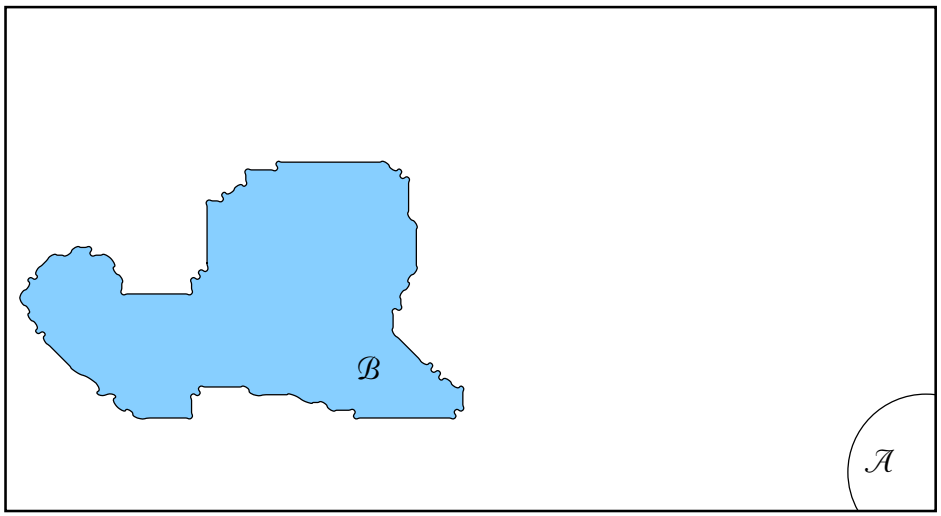
repeat
  $X =$**Dé-8()**
until $X \leqslant 6$
return X

Preuve : voir plus tard

MÉTHODE BASÉE SUR LE REJET

Principe

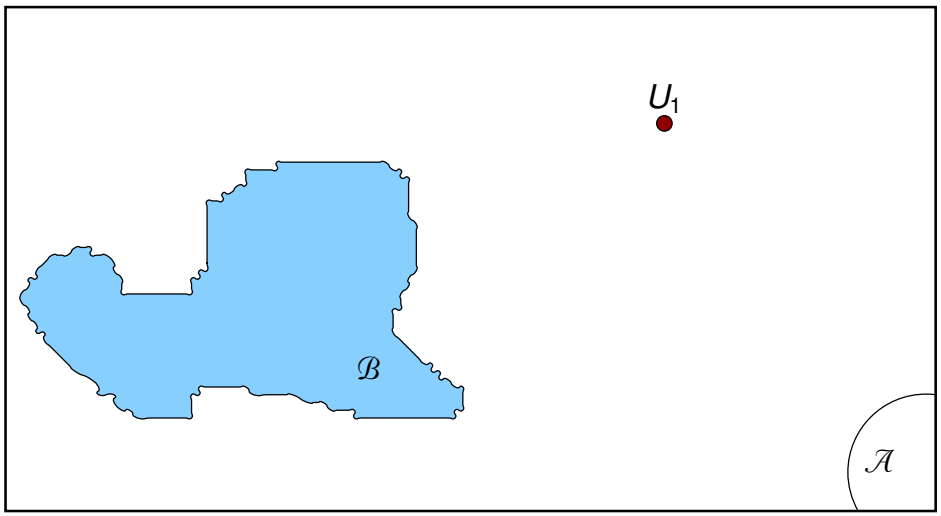
Générer uniformément sur $\mathcal{A}$ accepter si le point est dans $\mathcal{B}$.



MÉTHODE BASÉE SUR LE REJET

Principe

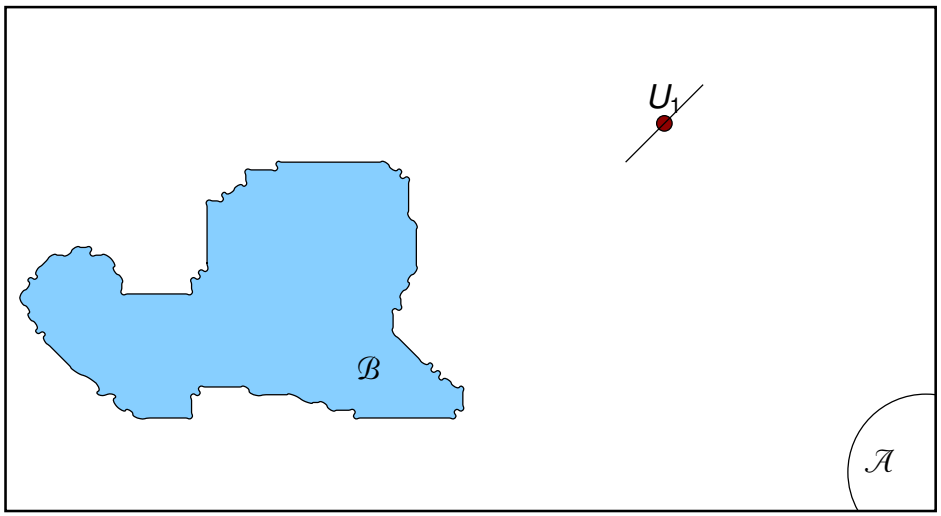
Générer uniformément sur $\mathcal{A}$ accepter si le point est dans $\mathcal{B}$.



MÉTHODE BASÉE SUR LE REJET

Principe

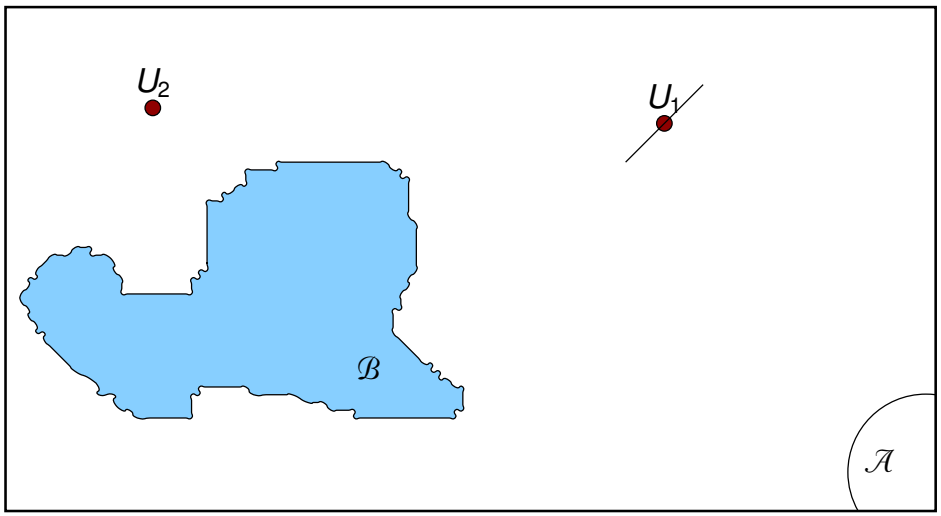
Générer uniformément sur $\mathcal{A}$ accepter si le point est dans $\mathcal{B}$.



MÉTHODE BASÉE SUR LE REJET

Principe

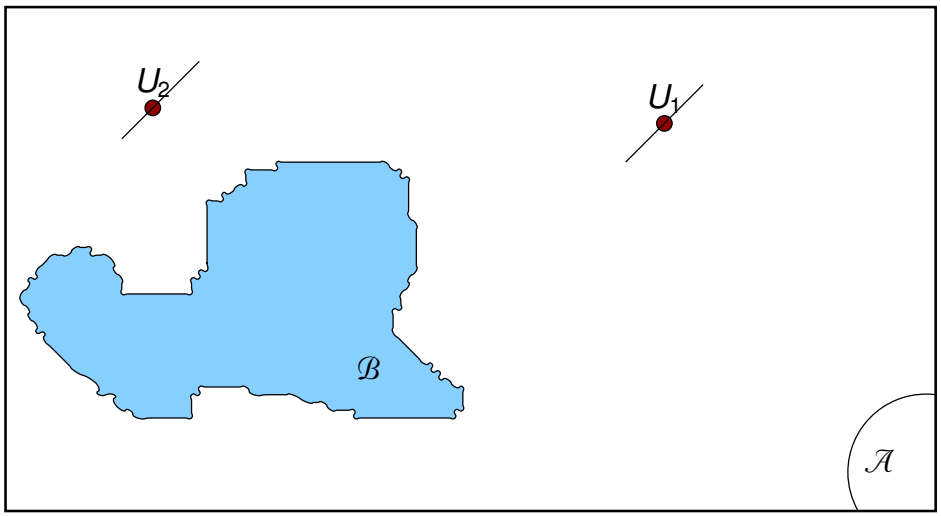
Générer uniformément sur $\mathcal{A}$ accepter si le point est dans $\mathcal{B}$.



MÉTHODE BASÉE SUR LE REJET

Principe

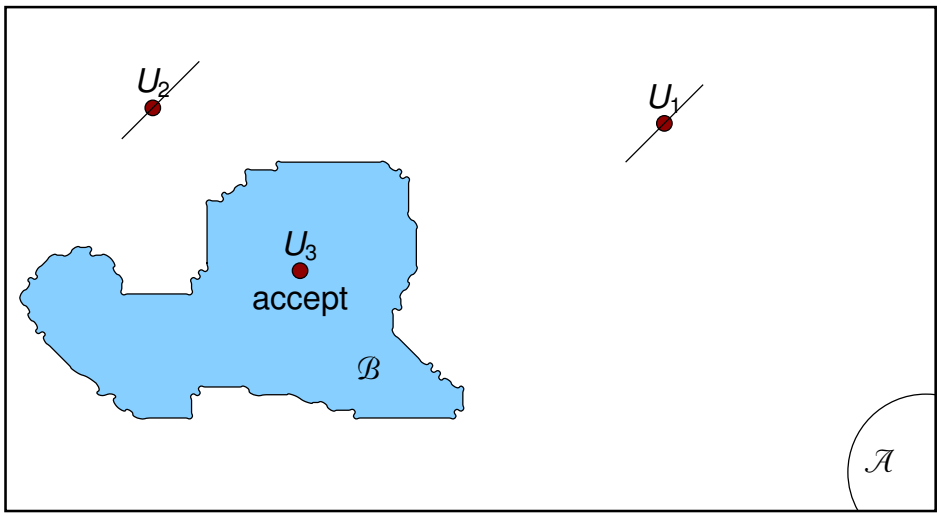
Générer uniformément sur $\mathcal{A}$ accepter si le point est dans $\mathcal{B}$.



MÉTHODE BASÉE SUR LE REJET

Principe

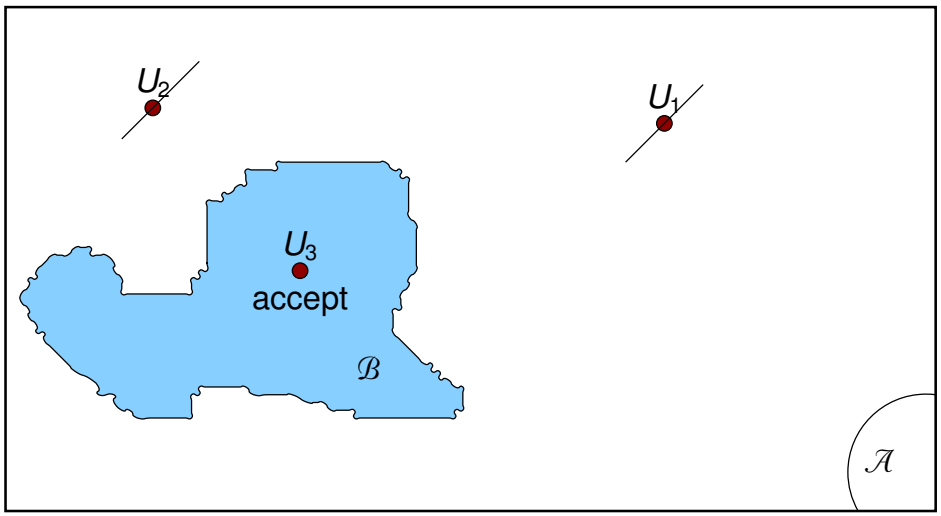
Générer uniformément sur $\mathcal{A}$ accepter si le point est dans $\mathcal{B}$.



MÉTHODE BASÉE SUR LE REJET

Principe

Générer uniformément sur $\mathcal{A}$ accepter si le point est dans $\mathcal{B}$.



Algorithme

Génère-unif($\mathcal{B}$)

Données:
Générateur uniforme sur $\mathcal{A}$
Résultat:
Générateur uniforme sur $\mathcal{B}$

repeat
| $X =$**Génère-unif($\mathcal{A}$)**
until $X \in \mathcal{B}$
return X

MÉTHODE BASÉE SUR LE REJET : PREUVE

Génère-unif($\mathcal{B}$)

Données:
Générateur uniforme sur $\mathcal{A}$
Résultat:
Générateur uniforme sur $\mathcal{B}$

$N = 0$
repeat
  $X =$**Génère-unif($\mathcal{A}$)**
  $N = N + 1$
until $X \in \mathcal{B}$
return X, N

Preuve

Tirages **Génère-unif($\mathcal{A}$)** : $X_1, X_2, \cdots, X_n, \cdots$

$$\begin{aligned} & \mathbb{P}(X \in \mathcal{C}, N = k) \\ = \; & \mathbb{P}(X_1 \notin \mathcal{B}, \cdots, X_{k-1} \notin \mathcal{B}, X_k \in \mathcal{C}) \\ = \; & \mathbb{P}(X_1 \notin \mathcal{B}) \cdots \mathbb{P}(X_{k-1} \notin \mathcal{B}) \mathbb{P}(X_k \in \mathcal{C}) \\ = \; & \left(1 - \frac{|\mathcal{B}|}{|\mathcal{A}|}\right)^{k-1} \frac{|\mathcal{C}|}{|\mathcal{A}|} \end{aligned}$$

$$\begin{aligned} \mathbb{P}(X \in \mathcal{C}) &= \sum_{k=1}^{+\infty} \mathbb{P}(X \in \mathcal{C}, N = k) \\ &= \sum_{k=1}^{+\infty} \left(1 - \frac{|\mathcal{B}|}{|\mathcal{A}|}\right)^{k-1} \frac{|\mathcal{C}|}{|\mathcal{A}|} = \frac{|\mathcal{C}|}{|\mathcal{B}|} \end{aligned}$$

Donc la loi est **uniforme** sur $\mathcal{B}$

UGA Université Grenoble Alpes

MÉTHODE BASÉE SUR LE REJET : COMPLEXITÉ

Génère-unif($\mathcal{B}$)

Données:
Générateur uniforme sur $\mathcal{A}$
Résultat:
Générateur uniforme sur $\mathcal{B}$

$N = 0$
repeat
  $X =$**Génère-unif($\mathcal{A}$)**
  $N = N + 1$
until $X \in \mathcal{B}$
return X, N

Complexité

N Nombre d'itérations

$$\begin{aligned}\mathbb{P}(N = k) &= \mathbb{P}(X \in \mathcal{B}, N = k) \\ &= \left(1 - \frac{|\mathcal{B}|}{|\mathcal{A}|}\right)^{k-1} \frac{|\mathcal{B}|}{|\mathcal{A}|}\end{aligned}$$

Loi géométrique de paramètre $p_a = \frac{|\mathcal{B}|}{|\mathcal{A}|}$.
Nombre moyen d'itérations

$$\begin{aligned}\mathbb{E}\, N &= \sum_{k=1}^{+\infty} k(1-p_a)^{k-1} p_a \\ &= \frac{1}{(1-(1-p_a))^2} p_a = \frac{1}{p_a}.\end{aligned}$$

$$Var\, N = \frac{1 - p_a}{p_a^2}$$

TYPES D'APPOCHES

Algorithme de Monte-Carlo

- utilise de l'aléatoire dans son exécution,
- a un temps d'exécution déterministe.

En général,

- le résultat rendu est potentiellement incorrect,
- le taux d'erreur est quantifiable précisément.

Algorithme de Las Vegas

- utilise de l'aléatoire dans son exécution,
- donne toujours un résultat correct,

En général

- le temps d'exécution qui dépend des tirages aléatoires, est aléatoire
- on peut étudier la distribution de probabilité.

UGA Université Grenoble Alpes

EXEMPLE : TRI RAPIDE

```
def tri_rapide(T):
    plus_grand = []
    liste_pivot = []
    plus_petit = []
    if len(T) <= 1:
        return T
    else:
        # segmentation
        pivot =T[0]
        for x in T:
            if x < pivot:
                plus_petit.append(x)
            elif x > pivot:
                plus_grand.append(x)
            else:
                liste_pivot.append(x)
        # fusion des résultats
        return tri_rapide(plus_petit) + liste_pivot + tri_rapide(plus_grand)
```

EXEMPLE : TRI RAPIDE

```
def tri_rapide(T):
    plus_grand = []
    liste_pivot = []
    plus_petit = []
    if len(T) <= 1:
        return T
    else:
        # segmentation
        pivot =T[0]
        for x in T:
            if x < pivot:
                plus_petit.append(x)
            elif x > pivot:
                plus_grand.append(x)
            else:
                liste_pivot.append(x)
        # fusion des résultats
        return tri_rapide(plus_petit) + liste_pivot + tri_rapide(plus_grand)
```

Coût de l'algorithme

- au mieux $\mathcal{O}(n \log n)$ (borne inférieure atteinte) arbre d'appels équilibré
- au pire $\mathcal{O}(n^2)$ arbre d'appels déséquilibré (fil)

UGA Université Grenoble Alpes

EXEMPLE : TRI RAPIDE (2)

```
def tri_rapide(T):
    plus_grand = []
    liste_pivot = []
    plus_petit = []
    if len(T) <= 1:
        return T
    else:
        # segmentation
        pivot =  random.choice(T)
        for x in T:
            if x < pivot:
                plus_petit.append(x)
            elif x > pivot:
                plus_grand.append(x)
            else:
                liste_pivot.append(x)
        # fusion des résultats
        return tri_rapide(plus_petit) + liste_pivot + tri_rapide(plus_grand)
```

EXEMPLE : TRI RAPIDE (2)

```
def tri_rapide(T):
    plus_grand = []
    liste_pivot = []
    plus_petit = []
    if len(T) <= 1:
        return T
    else:
        # segmentation
        pivot =  random.choice(T)
        for x in T:
            if x < pivot:
                plus_petit.append(x)
            elif x > pivot:
                plus_grand.append(x)
            else:
                liste_pivot.append(x)
        # fusion des résultats
        return tri_rapide(plus_petit) + liste_pivot + tri_rapide(plus_grand)
```

Coût de l'algorithme (algorithme de Las Vegas)

- au mieux $\mathcal{O}(n \log n)$ (borne inférieure atteinte) arbre d'appels équilibré
- au pire $\mathcal{O}(n^2)$ arbre d'appels déséquilibré (fil)
- en moyenne $\mathcal{O}(n \log n)$ variance $\sim n\sqrt{7 - 2\pi^2/3}$

UGA Université Grenoble Alpes

NOMBRES PREMIERS

Le problème de la factorisation d'entiers

Étant donné un entier n codé sur k bits, trouver un diviseur de n (ou trouver sa décomposition en facteurs premiers)

- problème reconnu difficile
- base d'algorithme de cryptographie (RSA) $N = pq$ ave p, q entiers premiers de grande taille

Génération aléatoire d'un grand entiers premier

On se donne une borne N_{max} et ou souhaite obtenir un nombre premier p, aléatoire, choisi uniformément parmi les nombres premiers $\leqslant N_{max}$.

```
def genere_premier(N_max)
        p = randrange(N_max)
        if test_primalite(p) :
                return p
        else :
                return genere_premier(N_max)
```

Coût moyen : $\pi(N) \sim \frac{N}{\log(N)}$ (théorème des nombres premiers)

Nombre moyen de tests $= \log(N_{max})$

de l'ordre de la taille de la représentation de N_{max}

TEST DE PRIMALITÉ : ALGORITHME DE MILLER-RABIN

Algorithm 38.1 (Miller's Primality Test) Given $n \geq 1$:

1. Let $n - 1 = 2^e m$, m odd.
2. Choose a random number $a \in \{1, 2, \ldots, n-1\}$.
3. Calculate $a^m \bmod n$. If $a^m \equiv 1(n)$, halt and output "prime".
4. Calculate $a^m, a^{2m}, a^{4m}, \ldots, a^{2^e m} = a^{n-1}$ modulo n by repeated squaring. If $a^{n-1} \not\equiv 1(n)$, halt and output "composite".
5. Find the largest k such that $a^{2^k m} \not\equiv 1(n)$. If $a^{2^k m} \equiv -1(n)$, output "prime", otherwise output "composite".

Extrait de : D. Kozen *The Design and Analysis of Algorithms* Springer (1991)

- si p est premier alors l'algorithme renvoie *premier*
- si p est composé alors l'algorithme renvoie *composé* avec une probabilité supérieure à $\frac{1}{2}$

UGA Université Grenoble Alpes

TAKE HOME MESSAGE

Usage de l'aléatoire

Outil algorithmique

- Approximation d'un résultat : algorithme de Monte-Carlo
  calcul de π, recuit simulé, test de primalité ...)
- Amélioration d'une méthode : algorithme de Las Vegas (résultat correct)
  quicksort, recherche randomisée, tables de hachage, ...

Propriété de l'aléatoire utilisée

- convergence des fréquences (balayer l'espace uniformément) : méthodes de Monte-Carlo
- imprédictibilité (pas d'algorithme facile pour prédire le suivant) : cryptographie

Production de l'aléatoire

Une machine produit du pseudo-aléatoire

- émulation du hasard (pseudo-générateur)
- source binaire aléatoire uniforme
- algorithme de transformation d'une source
- contrôle de l'aléatoire

Quelques références

Randomized Algorithms, *R. Motwani and P. Raghavan*, Cambridge University Press, 1995. Ouvrage historique de référence
Probability and Computing, *M. Mitzenmacher and E. Upfal*, Cambridge University Press, 2005. Ouvrage plus moderne, plus avancé.