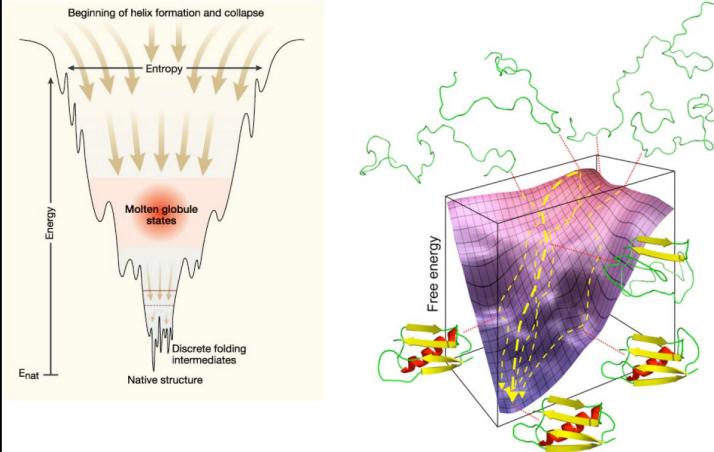


Introduction à la Modélisation Moléculaire

(techniques d'optimisation)

Georges Czaplicki, UPS / IPBS-CNRS
cgeorge@ipbs.fr, tél. : 05 61 17 54 04

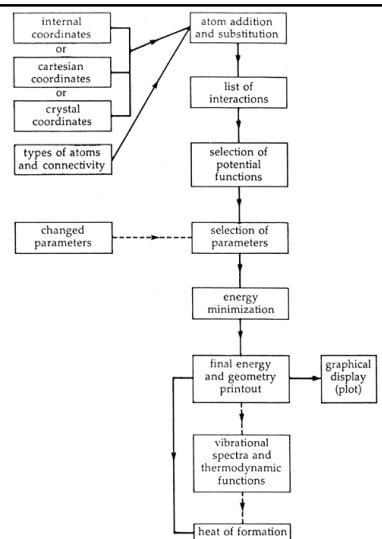
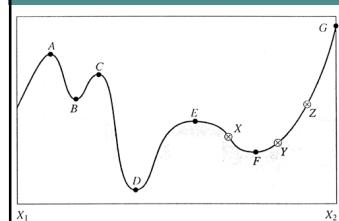
Paysage énergétique de protéines



Techniques de minimisation

- basées sur :
- valeurs de la fonction
 - valeurs de la dérivée

Exemple de profile d'une fonction en 1D

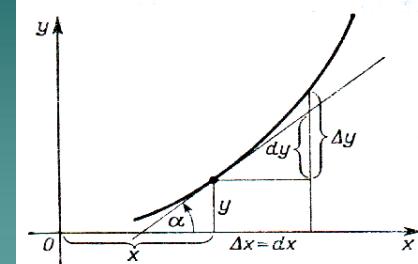


Dérivées

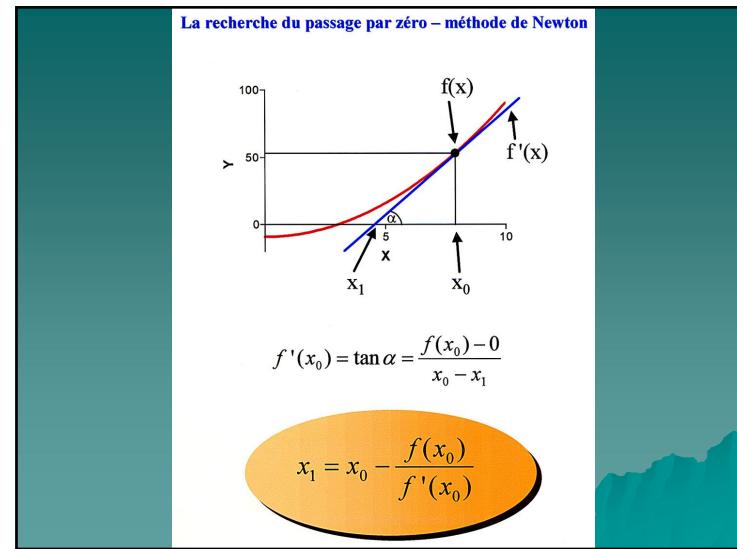
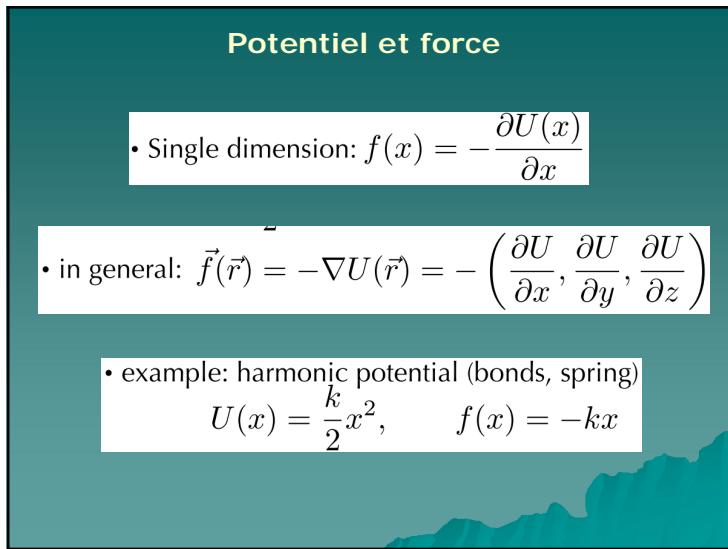
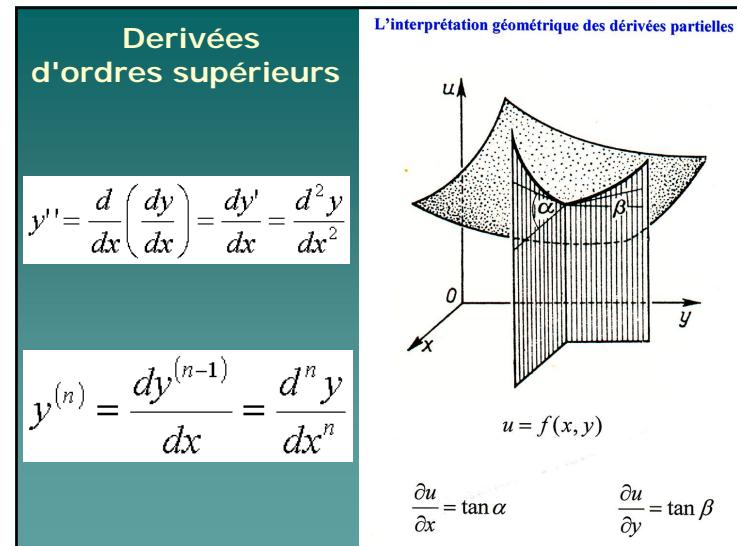
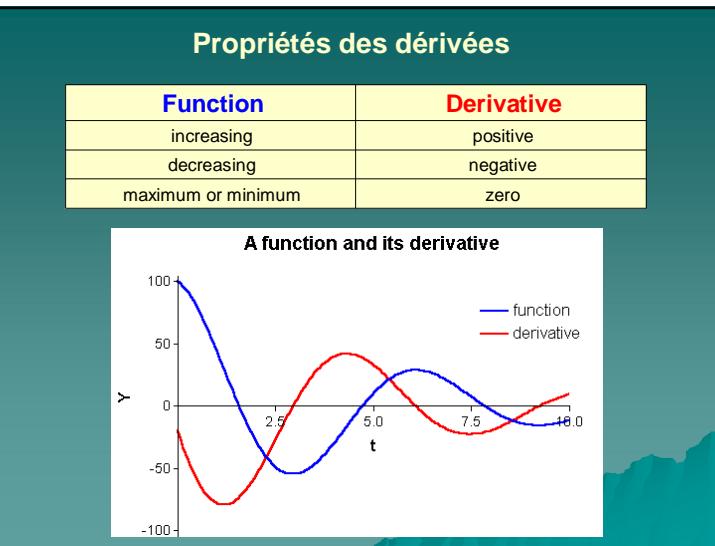
Définition:

$$f'(x) = \frac{dy}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Interprétation géométrique :



$$\tan \alpha = \frac{\Delta y}{\Delta x} \xrightarrow{\Delta x \rightarrow 0} \frac{dy}{dx} = f'(x)$$



Application de la méthode de Newton

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

La méthode de Newton peut être utilisée pour trouver le minimum d'une fonction :

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

Application de la méthode de Newton aux algorithmes numériques

Exemple : comment calculer la racine carrée de A ?
Solution : Créer une fonction dont le passage par zéro donne la solution du problème...

$$x = \sqrt{A}$$

$$x - \sqrt{A} = 0$$

...sous la condition que la fonction soit correctement créée !

$$x = \sqrt{A}$$

$$x^2 = A$$

$$x^2 - A = 0$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - A}{2x_i} = \frac{x_i}{2} + \frac{A}{2x_i}$$

Résultat du calcul pour A=9	et A=16 :
4.500000000000000	8.000000000000000
3.250000000000000	5.000000000000000
3.009615384615385	4.100000000000000
3.00015360039322	4.001219512195122
3.00000000039321	4.000000185844589
3.000000000000000	4.000000000000004
3.000000000000000	4.000000000000000

Minimization techniques: line search

Figure 4-1. Energy Contour Surface of a Simple Function
An energy contour surface for the function $x^2 + 5y^2$. Each contour represents an increase of two arbitrary energy units.

Figure 4-2. Energy Surface for Eq. 4-1
The derivative vector from the initial point $a(x_0, y_0)$ defines the line search direction. Note that the derivative vector does not point directly toward the minimum. Compare this representation with that in Figure 4-3, where the derivative $-d$ points directly toward the minimum. Note that the minimum (point e) occurs precisely at the point where the derivative vector is tangent to the energy contours, which implies that the subsequent derivative vectors are orthogonal to the previous derivatives.

Figure 4-3. Cross Section of the Energy Surface as Defined by the Intersection of the Line Search Path in Figure 4-2 with the Energy Surface
The independent variable α is a one-dimensional parameter that is adjusted so as to minimize the value of the function $E(x, y)$, where x and y are parameterized in terms of α in Eq. 4-3. Point a corresponds to the initial point (when α is 0), and point c is the local one-dimensional minimum. Points b and d , along with a , bound the minimum and form the basis for an iterative search for the minimum.

Steepest descent

Good point: stable and initially very fast
Bad points: very slow at the final stage, slow convergence for some types of functions

$$x_{i+1} = x_i - \varepsilon_i \cdot \nabla V_i$$

Figure 4-4. Minimization Path following a Steepest-Descent Path
When complete line searches starting from point a are used, the minimum is reached after 12 iterations. This is the slowest possible rate of convergence. On average, only 3 function evaluations are needed for each line search using a quadratic interpolation procedure. If the line search is incomplete, it may overshoot the local minimum, resulting in an inefficient, oscillating trajectory.

Figure 4-5. Minimization Path following a Steepest-Descent Path without Line Searches
The searching starts from point p_1 and converges on the minimum in about 12 iterations. Although this is faster than the complete line search case in Figure 4-4, the steepest descent is five times faster since, on average, each iteration uses only 1.3 function evaluations. The reason for this is that the incomplete line search approximates the function gradient, which makes the function evaluation the most time-consuming portion of the calculation.

Conjugate gradients

$$v_k = -g_k + \gamma_k v_{k-1}$$

$$\gamma_k = \frac{g_k \cdot g_k}{g_{k-1} \cdot g_{k-1}}$$

Good points:

- fixes the slow convergence problem

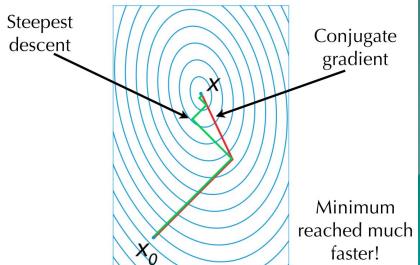
- uses variable search directions instead of gradients only

Bad points:

- works on quadratic functions only, needs periodic restarts for other functions



Memory from previous step



Methods based on second derivative

Newton-Raphson algorithm:

$$\tan(\phi) = f'(x_0) = \frac{f'(x_0)}{x_0 - x_1} \quad x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{F}^{-1}(\mathbf{x}_i) \cdot \nabla V_i$$

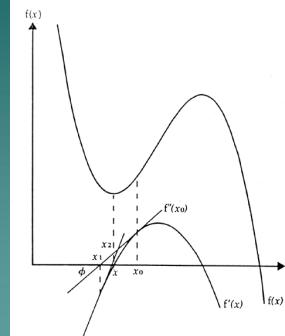
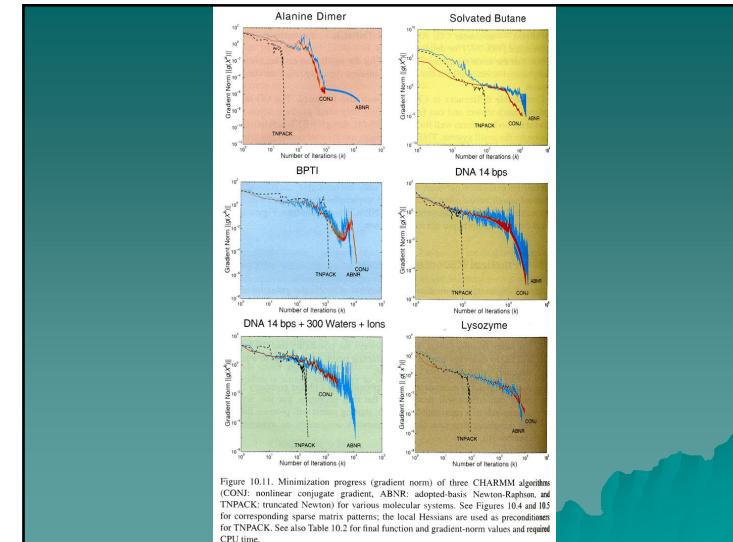
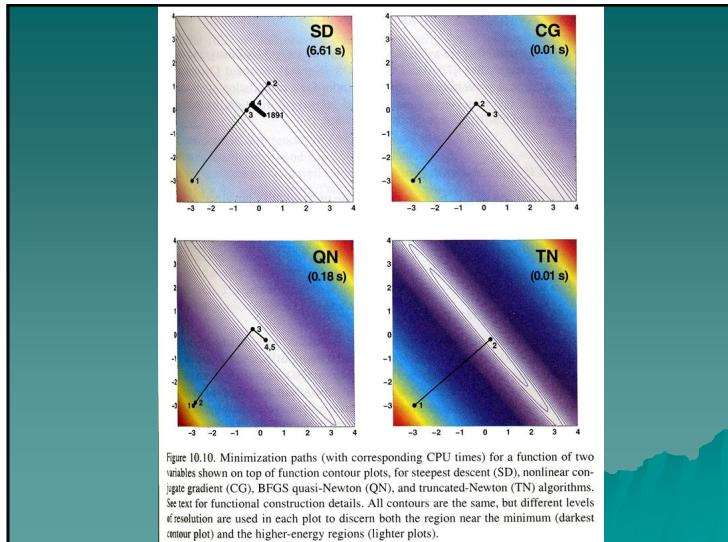


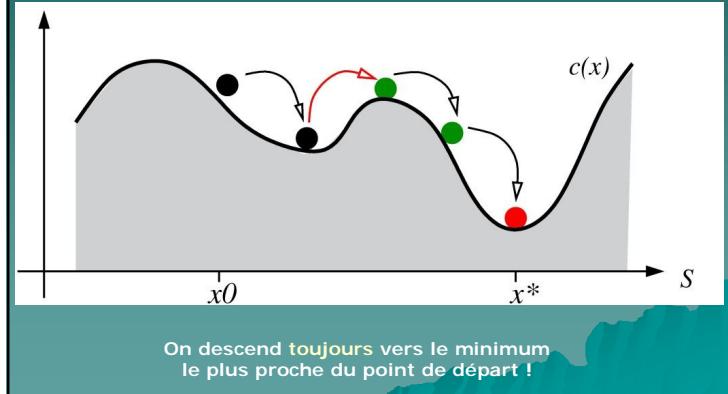
Figure 3.2. Minimization of a function $f(x)$ by the Newton-Raphson procedure. Iteration starting at x_0 maximizes the solution of $f'(x) = 0$ successively to x_1 and to x_2 , approaching the true solution x .

$\frac{\delta^2 V}{\delta x_1^2}$	$\frac{\delta^2 V}{\delta y_1 \delta x_2}$	$\frac{\delta^2 V}{\delta z_1 \delta x_2}$	—	—	—	—
$\frac{\delta^2 V}{\delta x_1 \delta y_2}$	$\frac{\delta^2 V}{\delta y_1 \delta y_2}$	$\frac{\delta^2 V}{\delta z_1 \delta y_2}$	—	—	—	—
$\frac{\delta^2 V}{\delta x_1 \delta z_2}$	$\frac{\delta^2 V}{\delta y_1 \delta z_2}$	$\frac{\delta^2 V}{\delta z_1 \delta z_2}$	—	—	—	—
$\frac{\delta^2 V}{\delta x_2^2}$	$\frac{\delta^2 V}{\delta y_2 \delta x_2}$	$\frac{\delta^2 V}{\delta z_2 \delta x_2}$	—	—	—	—
$\frac{\delta^2 V}{\delta x_2 \delta y_2}$	$\frac{\delta^2 V}{\delta y_2 \delta y_2}$	$\frac{\delta^2 V}{\delta z_2 \delta y_2}$	$\frac{\delta^2 V}{\delta x_2^2}$	$\frac{\delta^2 V}{\delta y_2 \delta z_2}$	$\frac{\delta^2 V}{\delta z_2 \delta z_2}$	—
$\frac{\delta^2 V}{\delta x_2 \delta z_2}$	$\frac{\delta^2 V}{\delta y_2 \delta z_2}$	$\frac{\delta^2 V}{\delta z_2 \delta z_2}$	$\frac{\delta^2 V}{\delta x_2 \delta z_2}$	$\frac{\delta^2 V}{\delta y_2 \delta z_2}$	$\frac{\delta^2 V}{\delta z_2 \delta z_2}$	—
—	—	—	—	—	—	etc.

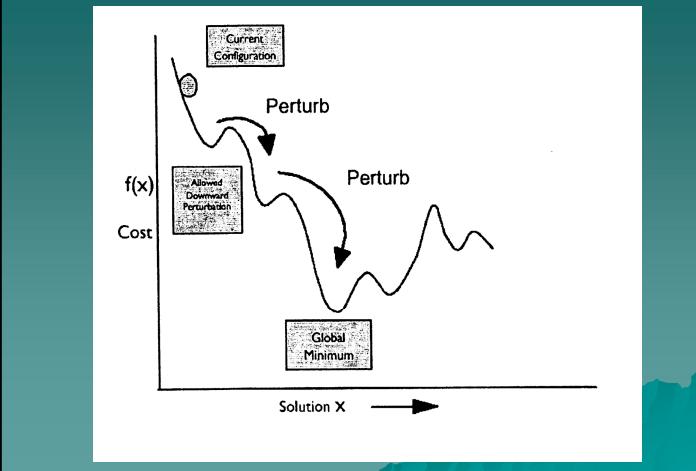
Figure 3.3. Elements of the block-diagonal and full \mathbf{F} matrix. The block-diagonal contains only 3×3 blocks where x , y and z belong to the same atom (top left and lower right). The full matrix also contains all other blocks.



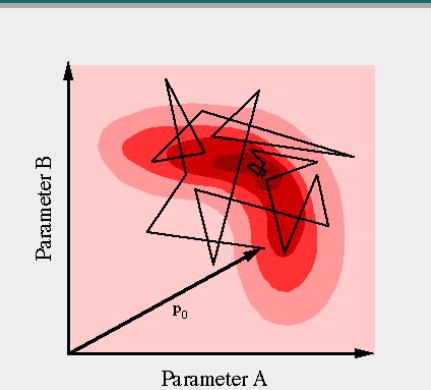
Le problème majeur lié avec toutes les techniques de minimisation basées sur le calcul de gradients



Algorithmes stochastiques



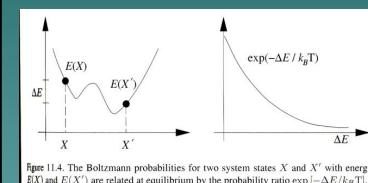
Exploration d'espace conformationnel de façon aléatoire



Stochastic algorithms: simulated annealing

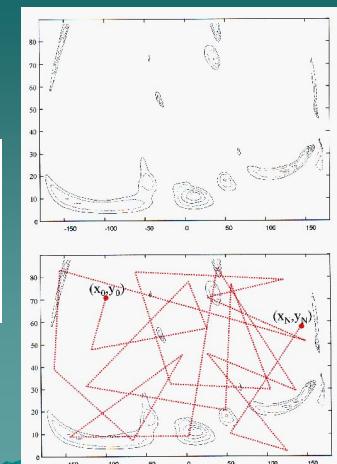
$$x'_i = x_i + r \cdot v_i$$

$$p = e^{-\frac{f'-f}{T}}$$

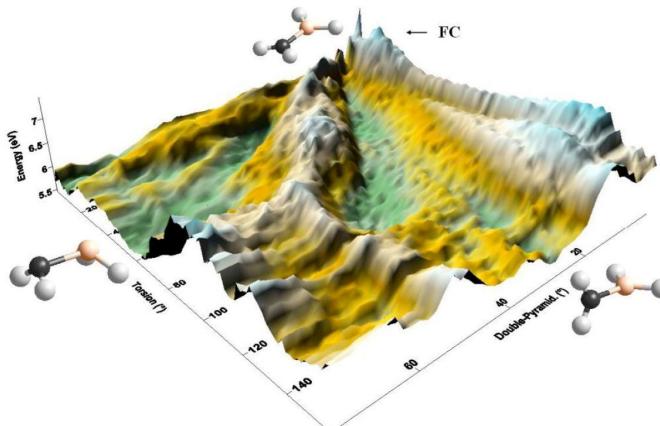


If $p > p'$, accept the new point
(p' generated randomly)

$$T' = r_T \cdot T$$



A 'real' potential energy surface



L'échantillonage

En statistique prélever un échantillon consiste à extraire un ou plusieurs individus d'une population. Les renseignements obtenus sur un échantillon permet de mieux connaître la population. Le recours à un échantillon répond en général à la nécessité pratique (manque de temps, de place, évaluation destructive d'une production...) ou économique (coût trop élevé) de s'abstraire de l'étude exhaustive de la population.

L'acte de sélection s'appelle **l'échantillonage**.

Les méthodes **MCMC** (pour **Markov Chain Monte Carlo**) sont une classe de méthodes d'échantillonage à partir de **distributions de probabilité**, qui se basent sur le parcours de chaînes de Markov.

Certaines méthodes utilisent des marches aléatoires sur les chaînes de Markov (Algorithme de Metropolis-Hastings, Échantillonnage de Gibbs), alors que d'autres algorithmes, plus complexes, introduisent des contraintes sur les parcours.

Distributions de probabilité

Une loi de probabilité décrit le comportement aléatoire d'un phénomène dépendant du hasard.

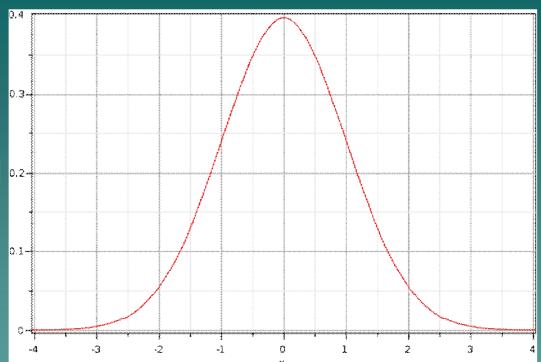
Des fluctuations (la variabilité) sont présentes dans presque toute valeur qui peut être mesurée lors de l'observation d'un phénomène, quelle que soit sa nature ; de plus presque toutes les mesures ont une part d'erreur intrinsèque. Les lois de probabilités permettent de modéliser ces incertitudes et de décrire des phénomènes physiques, biologiques, économiques, etc. Le domaine de la statistique permet de trouver des lois de probabilités adaptées aux phénomènes aléatoires.

Il existe beaucoup de lois de probabilités différentes. Parmi toutes ces lois, la loi **normale** a une importance particulière puisque, d'après le théorème central limite, elle approche le comportement asymptotique de nombreuses lois de probabilités.

Planche de Galton

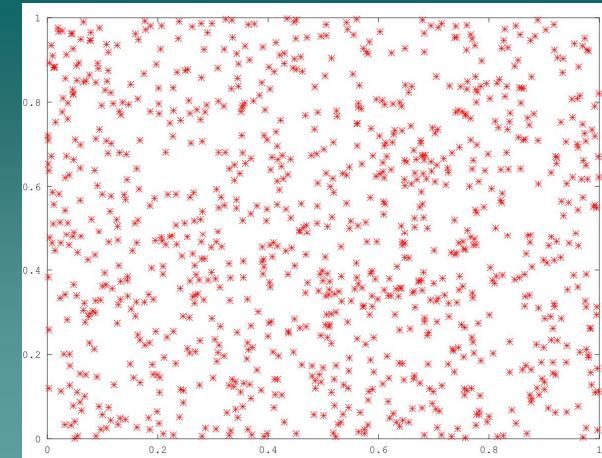
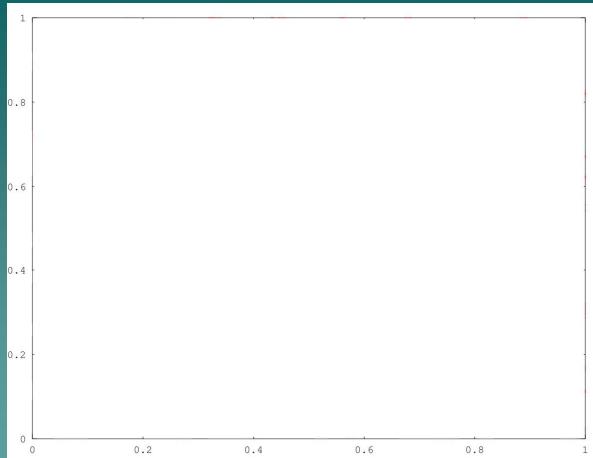
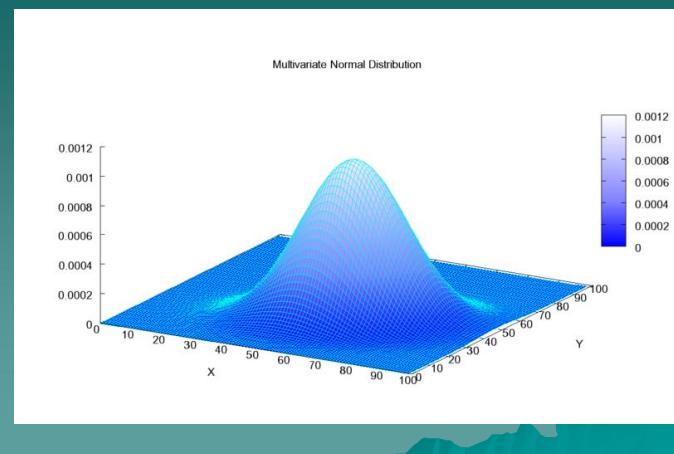


La loi normale (la distribution de Gauss)

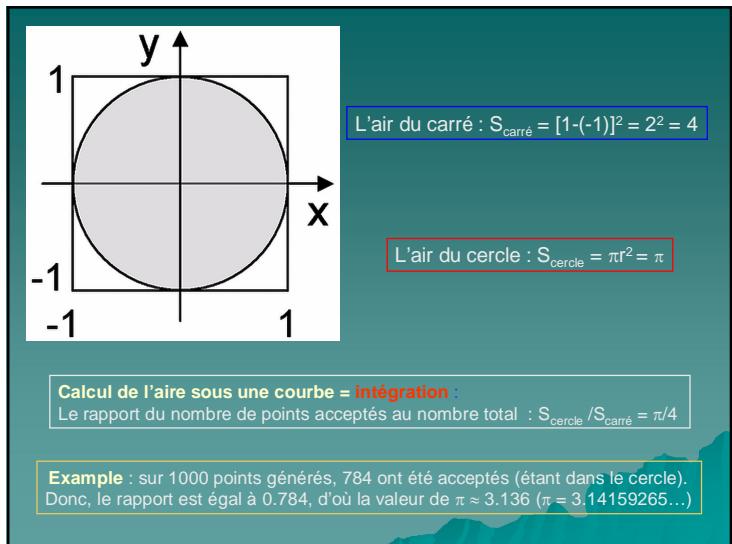
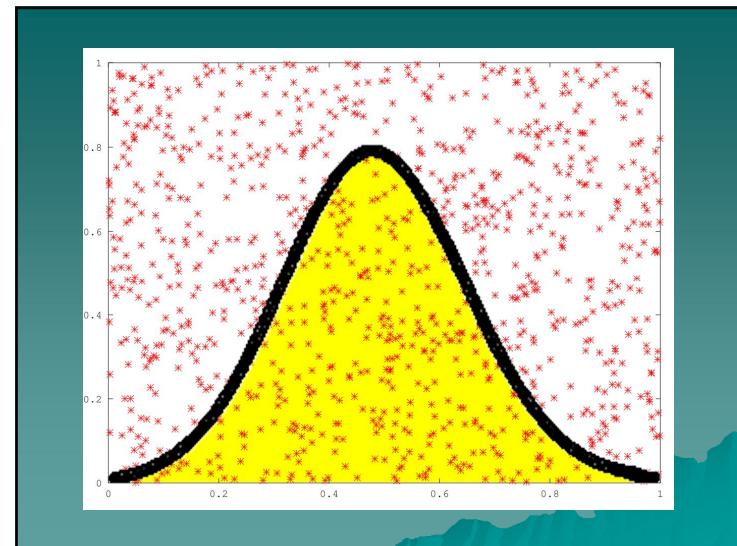
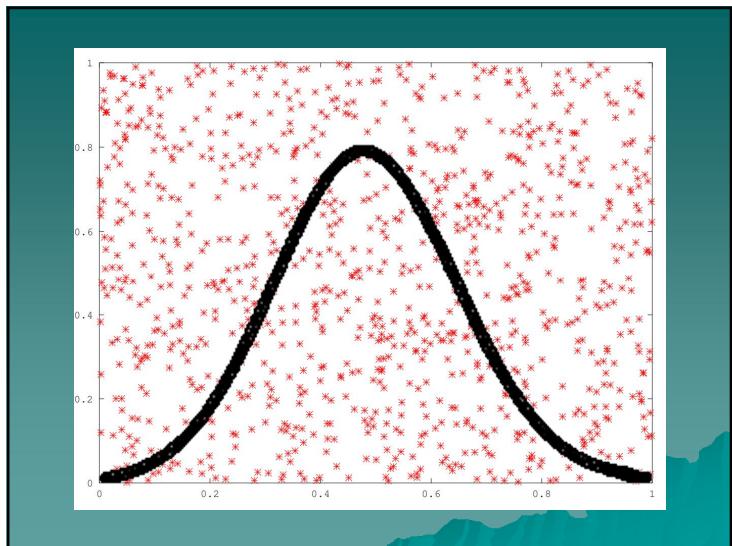


$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

La loi normale en deux dimensions



1000 points aléatoires générés sur l'intervalle (0,1) dans deux dimensions



Algorithme de Metropolis-Hastings

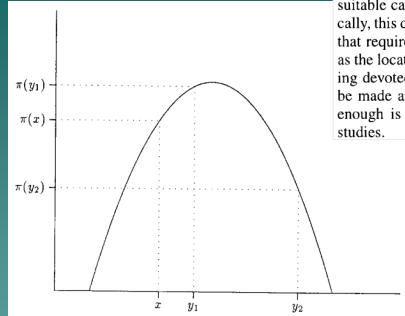
Etant donné une distribution de probabilité \mathbf{P} , cet algorithme définit une chaîne de Markov, dont la distribution de points représente les caractéristiques de \mathbf{P} .

La première version de l'algorithme, décrite en 1953 par Metropolis *et al.*, considérait le cas particulier de la distribution de Boltzmann. En 1970 , W. K. Hastings a étendu l'algorithme au cas de n'importe quelle distribution. Voici ses principes :

- nous effectuons une marche aléatoire dans l'espace de paramètres (à chaque moment t , quand la position est décrite par la coordonnée x_t , le choix du point consécutif x_{t+1} ne dépend que du point actuel x_t , ce qui représente la définition de la chaîne de Markov)
- pour sortir du point x_t , une **proposition** x_{t+1} est générée d'après une distribution de probabilité \mathbf{Q} . Le plus souvent il s'agit de la loi normale, décrivant les fluctuations, où on passe de x_t à x_{t+1} avec la probabilité $\mathbf{Q}(x_t, x_{t+1})$. Si \mathbf{Q} est symétrique, $\mathbf{Q}(x_t, x_{t+1}) = \mathbf{Q}(x_{t+1}, x_t)$.
- la probabilité p que la proposition sera acceptée est donnée par l'expression :

$$p = \frac{\mathbf{P}(x_{t+1}) \cdot \mathbf{Q}(x_t, x_{t+1})}{\mathbf{P}(x_t) \cdot \mathbf{Q}(x_{t+1}, x_t)} \xrightarrow{\text{si } \mathbf{Q} \text{ est symétrique}} p = \frac{\mathbf{P}(x_{t+1})}{\mathbf{P}(x_t)}$$

Un exemple de la distribution de probabilité Q



To implement the M-H algorithm, it is necessary that a suitable candidate-generating density be specified. Typically, this density is selected from a family of distributions that requires the specification of such tuning parameters as the location and scale. Considerable recent work is being devoted to the question of how these choices should be made and, although the theory is far from complete, enough is known to conduct most practical simulation studies.

Dans le variant de Metropolis, la fonction Q est symétrique, mais dans l'approche de Metropolis-Hastings la symétrie n'est pas nécessaire.

Acceptance Rates

It is important to monitor the *acceptance rate* (the fraction of candidate draws that are accepted) of your Metropolis-Hastings algorithm.

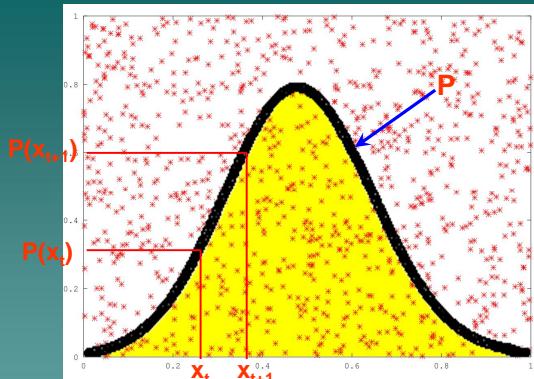
If your acceptance rate is too high, the chain is probably not mixing well (not moving around the parameter space quickly enough).

If your acceptance rate is too low, your algorithm is too inefficient (rejecting too many candidate draws).

What is too high and too low depends on your specific algorithm, but generally

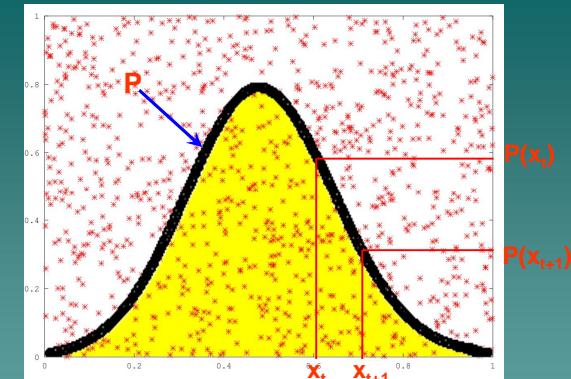
- ▶ random walk: somewhere between 0.25 and 0.50 is recommended
- ▶ independent: something close to 1 is preferred

Exemple admettons \mathbf{P} = loi normale, \mathbf{Q} = distribution uniforme (symétrique)



$$p = \frac{\mathbf{P}(x_{t+1})}{\mathbf{P}(x_t)} = \frac{0.6}{0.3} = 2 \quad (\text{si } p \geq 1, \text{ la proposition est acceptée})$$

Exemple admettons \mathbf{P} = loi normale, \mathbf{Q} = distribution uniforme (symétrique)



$$p = \frac{\mathbf{P}(x_{t+1})}{\mathbf{P}(x_t)} = \frac{0.3}{0.6} = 0.5 \quad (\text{la proposition n'est pas rejetée d'emblée, elle est accepté avec la probabilité 50\%})$$

Question : Comment faire pour accepter un point avec la probabilité p ???

Réponse : Super simple !!! La valeur de p varie entre 0 et 1. Donc, il suffit de générer un chiffre aléatoire q avec la distribution uniforme de probabilité sur l'intervalle (0,1) et de le comparer avec la valeur p :

- si $p \geq q$, le nouveau point est accepté
- si $p < q$, le nouveau point est rejeté

Dans notre cas, si $p = 0.5$, on a 50% de chance que la valeur q sera moins grande que p , et 50% de chance qu'elle sera moins grande. Donc, on acceptera la proposition une fois sur deux (ce qui correspond à 50% de probabilité d'acceptation).

Stochastic MC algorithms: simulated annealing

$$P_i \propto e^{-\left(\frac{E_i}{kT}\right)}$$

$$p = \frac{P_j}{P_i} = \frac{e^{-\left(\frac{E_j}{kT}\right)}}{e^{-\left(\frac{E_i}{kT}\right)}} = e^{-\left(\frac{E_j - E_i}{kT}\right)}$$

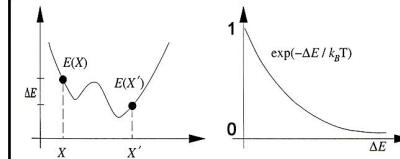


Figure 11.4. The Boltzmann probabilities for two system states X and X' with energies $E(X)$ and $E(X')$ are related at equilibrium by the probability ratio $\exp[-\Delta E/k_B T]$.

If $p > q$, accept the new point
(q generated randomly)

