

Algorithmes et complexité

illustration en arithmétique des ordinateurs

Nathalie Revol INRIA, projet Arénaire, LIP, ENS Lyon
Nathalie.Revol@ens-lyon.fr

APMEP, Lyon, 16 avril 2003

Algorithme. . .

suite d'instructions à effectuer pour accomplir une tâche

Syntaxe **rigide** quand on s'adresse à un ordinateur !

Exemple : conduite de réunion

Démarrage

- rappeler l'ordre du jour
- définir les règles du jeu

Premier thème

- rappeler les objectifs
- lancer la discussion, l'orateur. . .
- animer : technique des questions
- déléguer la gestion du temps, le compte rendu
- conclure : rappeler les objectifs et les résultats obtenus

Idem pour les thèmes suivants

Conclusion

- conclusion générale pour tous les thèmes
- remercier les participants

Algorithme : conduite de réunion

Initialisations

- créer la liste des sujets de l'ordre du jour . . .

Faire en parallèle

- **pour chaque thème dans la liste faire**
 - rappeler les objectifs
 - lancer la discussion, l'orateur. . .
 - animer :

{	si groupe apathique alors
	poser des questions
	sinon si groupe emballé alors

 - rappeler les règles du jeu
 - conclure sur ce thème, le supprimer de la liste
- gestion du temps
- prise de notes pour le compte rendu
- gestion des moyens techniques

Conclusion. . .

Assembleur : gérer le temps

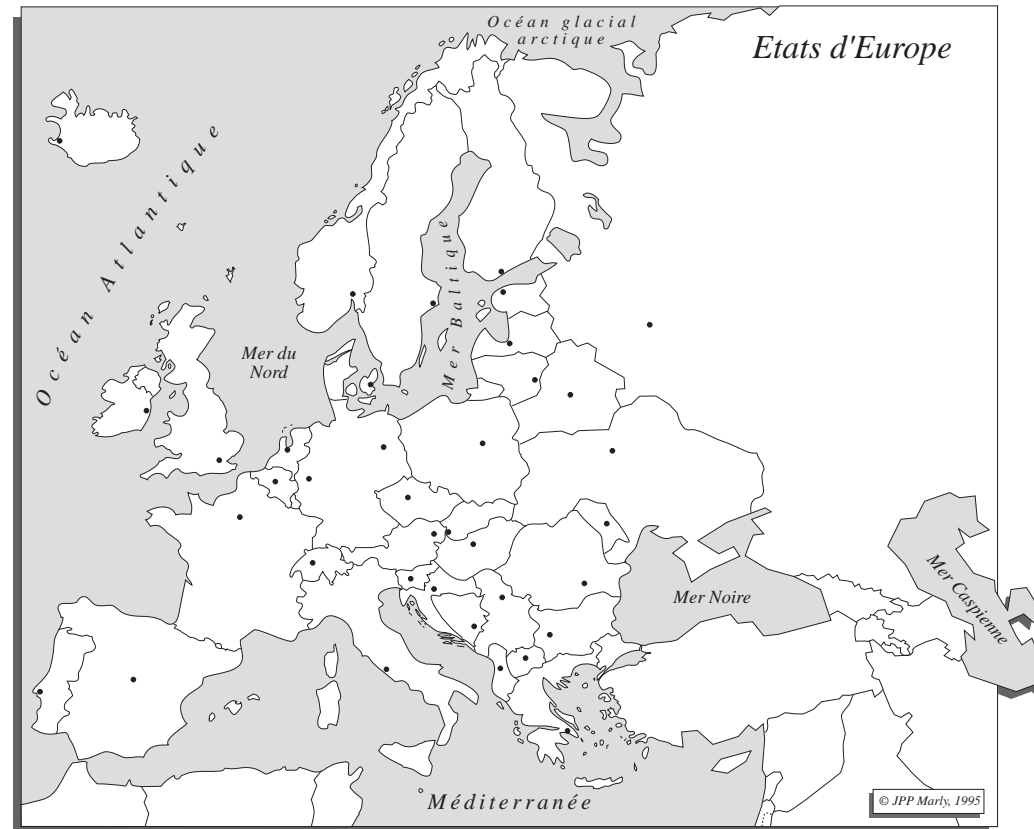
Mode d'emploi du chronomètre

- presser sur le bouton A pour sélectionner le mode chronomètre
- presser sur le bouton C pour démarrer / arrêter le chronomètre
- presser sur C pendant 3 secondes pour remettre le chrono à zéro

Assembleur :

langage de très bas niveau, compréhensible par le chrono / l'ordinateur.

Complexité



carte : <http://www.cfdp.ch/geofri/>

Complexité

Reachability

Sur une carte routière avec n villes, peut-on aller de Paris à Copenhague ?
de Berlin à Dublin ?

Algorithme pour trouver la réponse en $\mathcal{O}(n^2)$.

Problème du voyageur de commerce

Sur une carte de France métropolitaine avec n villes, comment établir une tournée la plus courte possible qui passe une fois et une seule par chacune des n villes ?

Aucun algorithme pour trouver la réponse en $\mathcal{O}(n^\alpha)$ (?).

Plan de l'exposé

- Algorithmes pour l'arithmétique exacte
(~ école primaire)
- Algorithmes pour l'arithmétique flottante
(~ notation scientifique)
- Arithmétique par intervalles
- Complexité

Systemes redondants de numération

Avizienis, 1961

En base β , chiffres $\in \{-a, \dots, +a\}$
avec a entier et $2a \geq \beta + 1$ et $a \leq \beta - 1$,

Le nombre x est représenté par $x_{n-1}x_{n-2} \cdots x_2x_1x_0$ si $x = \sum_{i=0}^{n-1} x_i \beta^i$.

Si $2a + 1 \geq \beta$ alors on peut représenter tous les entiers.

Si $2a + 1 > \beta$ alors un nombre peut avoir **plusieurs** représentations : le système est dit **redondant**.

Exemple : en base $\beta = 10$ avec $a = 6$, 1546 s'écrit 1546, $2(-5)46$ noté $2\bar{5}46$, $2(-4)(-6)6$ noté $2\bar{4}\bar{6}6$, $2\bar{4}\bar{5}\bar{4}$, $16\bar{6}6$, $16\bar{5}\bar{4}$. . .

Addition sans propagation de retenue

En base β avec des chiffres compris entre $-a$ et a , calcul de

$$s_n \cdots s_0 = x_{n-1} \cdots x_0 + y_{n-1} \cdots y_0$$

1. Calculer pour $i = 0 \cdots n - 1$

$$t_{i+1} = \begin{cases} -1 & \text{si } x_i + y_i \leq -a \\ 0 & \text{si } -a + 1 \leq x_i + y_i \leq a - 1 \\ 1 & \text{si } x_i + y_i \geq a \end{cases}$$
$$w_i = x_i + y_i - \beta t_{i+1}$$

2. Calculer pour $i = 0 \cdots n$:

$$s_i = w_i + t_i \text{ avec } w_n = t_0 = 0.$$

Exemple : $\beta = 10, a = 6$

i		4	3	2	1	0
x_i		1	$\bar{2}$	5	3	$\bar{5}$
y_i		3	5	1	$\bar{5}$	$\bar{6}$
$x_i + y_i$		4	3	6	$\bar{2}$	$\bar{11}$
t_{i+1}	0	0	1	0	$\bar{1}$	
w_i		4	3	$\bar{4}$	$\bar{2}$	$\bar{1}$
s_i		4	4	$\bar{4}$	$\bar{3}$	$\bar{1}$

Écriture usuelle : $x = 1\bar{2}53\bar{5} = 8525, y = 351\bar{5}\bar{6} = 35044.$

Somme = 43569.

Addition dans un système redondant de numération

Les $t_i \simeq$ retenues de l'addition usuelle

mais pas de dépendance entre t_i et t_{i+1}

⇒ le calcul de chaque chiffre peut se faire en parallèle.

Complexité : $\mathcal{O}(1)$ si on a assez de ressources de calcul.

Remarque : si $\beta = 2$, les conditions $2a \geq b + 1$ et $a \leq b - 1$ ne peuvent pas être satisfaites simultanément.

Autres algorithmes, avec les chiffres $\bar{1}$, 0 et 1 pour la base 2.

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

57	86
114	43
228	21
456	10
912	5
1824	2
3648	1

Multiplication des paysans russes ou comment éviter d'apprendre ses tables

$$57 * 86 = 4902$$

57	86
114	43
228	21
456	10
912	5
1824	2
3648	1
<hr/>	
4902	

Multiplication : on peut aller plus vite. . .

Si A et B s'écrivent chacun avec $2n$ chiffres en base β ,

$$A = a_H\beta^n + a_L, \quad B = b_H\beta^n + b_L$$

$$A \times B = a_Hb_H\beta^{2n} + (a_Hb_L + a_Lb_H)\beta^n + a_Lb_L$$

soit 4 multiplications de nombres 2 fois plus court. . .

Multiplication de Karatsuba

On peut aussi écrire le produit comme

$$A \times B = a_Hb_H\beta^{2n} + [(a_H + a_L)(b_H + b_L) - a_Hb_H - a_Lb_L]\beta^n + a_Lb_L$$

avec 3 multiplications de nombres 2 fois plus court. . .

on multiplie 2 nombres de n chiffres en $\mathcal{O}(n^{\log_2 3})$.

Multiplication : on peut aller encore plus vite. . .

Multiplication rapide

Multiplication basée sur la transformée de Fourier rapide (FFT) (ou plutôt discrète : DFT) : la complexité asymptotique de la multiplication est

$$\mathcal{O}(n \log n \log \log n)$$

mais elle est difficile à implanter et ne bat d'autres méthodes plus simples qu'à partir d'environ 57 000 chiffres décimaux.

Exponentiation

$$3^{13} = 1\ 594\ 323$$

$$\begin{array}{r} 3 \\ 9 \\ 81 \\ 6561 \end{array} \quad \begin{array}{r} 13 \\ 6 \\ 3 \\ 1 \end{array}$$

puis multiplication $3 * 81 * 6561 = 1\ 594\ 323$.

Autre façon de l'écrire : $13 = 8 + 4 + 1 = 1101_2$
 $3^{13} = 3^8 \times 3^4 \times 3^1$ avec $3^4 = (3^2)^2$ et $3^8 = (3^4)^2$.

On procède par élévations au carré successives et multiplications finales :
il y en a $\mathcal{O}(\log_2 n)$ si n est l'exposant.

Calculer à l'école primaire : les entiers

division

$$\begin{array}{r|l} 990 & 252 \\ -756 & \\ \hline 234 & 3 \end{array}$$

$$\begin{array}{r|l} 252 & 234 \\ -234 & \\ \hline 18 & 1 \end{array}$$

$$\begin{array}{r|l} \widehat{234} & 18 \\ -18 & \\ \hline 54 & 13 \\ -54 & \\ \hline 0 & \end{array}$$

Complexité de la division de 2 nombres à n chiffres :

$$\mathcal{O}(M(n))$$

où $M(n)$ est la complexité de la multiplication.

Calculer à l'école primaire : les entiers

pgcd

Rappel : $\text{pgcd}(a, b) = \text{pgcd}(b, r)$ avec $a = bq + r$ si $r \neq 0$.

Algorithme d'Euclide :

$$\text{pgcd}(990, 252) = \text{pgcd}(252, 234) = \text{pgcd}(234, 18) = 18.$$

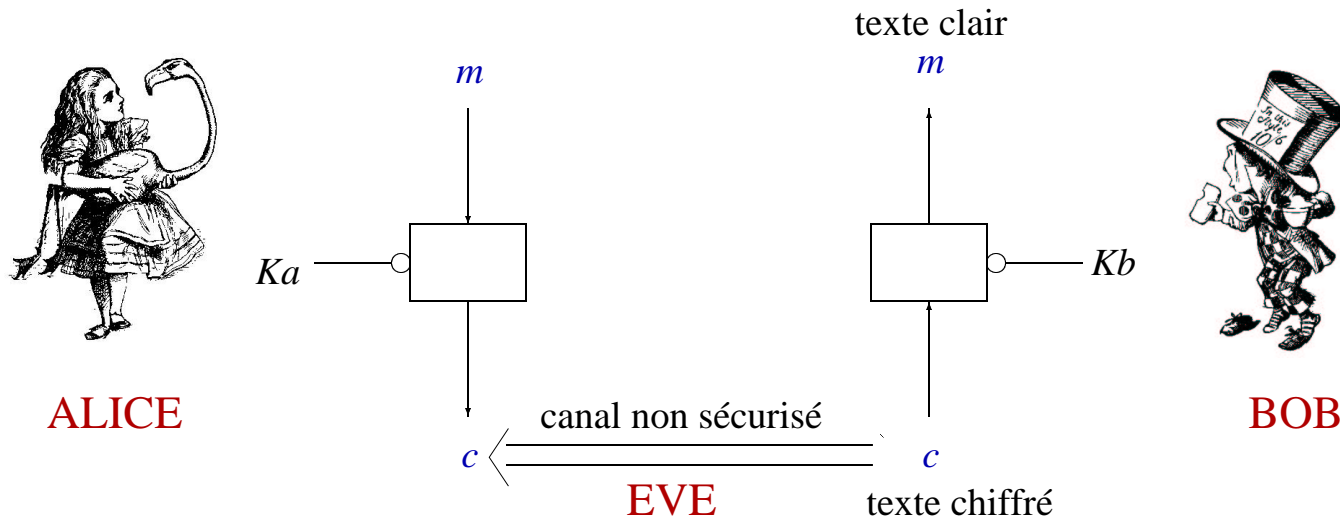
Complexité du pgcd de 2 nombres à n chiffres : $\mathcal{O}(M(n) \log n)$.

factorisation

$$990 = 2 \times 3 \times 3 \times 5 \times 11 \text{ et } 252 = 2 \times 2 \times 3 \times 3 \times 7.$$

Complexité : difficile! \Rightarrow certains algorithmes de cryptographie à clé publique, dont RSA.

Cryptographie à clé publique

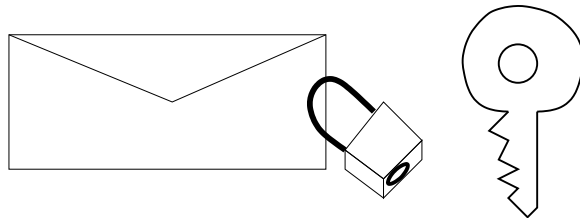


Algorithme RSA

Alice, Bob et la méchante Eve. . .

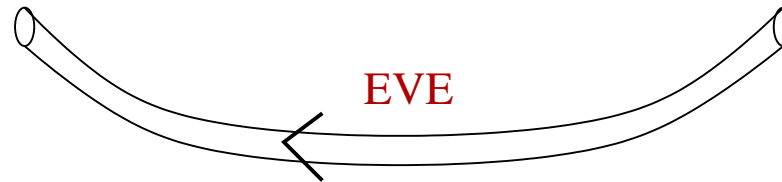
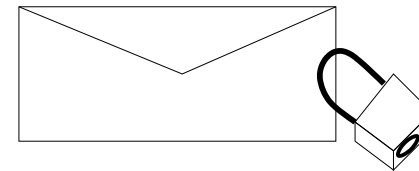
ALICE

décrypte le message
avec sa clé privée



BOB

encrypte le message
avec la clé publique d'Alice



communication non sécurisée

Algorithme RSA

Alice, Bob et la méchante Eve. . .

Alice

1. choisit 2 grands nombres premiers p et q : facile
2. calcule $N = p \times q$ (N doit avoir au moins 150 chiffres)
3. choisit $e \in \{2, \dots, \varphi(N) - 2\}$ premier avec $\varphi(N)$
où φ est la fonction d'Euler : $\varphi(N) = (p - 1) \times (q - 1)$
4. calcule d tel que $ed \equiv 1 \pmod{\varphi(N)}$
Bézout : $\exists(d, v) / ed + v.\varphi(N) = 1.$
5. publie (N, e) sa clé publique et cache (N, d) sa clé secrète.

Euler

$\forall x \in \{0 \dots N - 1\}$, si $x \wedge N = 1$ alors $x^{\varphi(N)} \equiv 1 \pmod{N}$,
càd $x^{\varphi(N)+1} \equiv x \pmod{N}$.

Alice, Bob et la méchante Eve. . .

Alice

publie (N, e) sa clé publique et cache (N, d) sa clé secrète.
(Bézout : $\exists(d, v)/ed + v.\varphi(N) = 1.$)

Euler

$\forall x \in \{0 \cdots N - 1\}$, si $x \wedge N = 1$ alors $x^{\varphi(N)+1} \equiv x \pmod N$.

Bob

1. écrit son message sous la forme d'un entier $x \in \{0 \cdots N - 1\}$
2. calcule $y = x^e \pmod N$: facile
3. envoie y

Alice calcule y^d :

$$x \equiv x^{\varphi(N)+1} \equiv x^{-v.\varphi(N)+1} \equiv x^{ed} \equiv y^d \pmod N$$

Les rationnels

$+$, $-$, \times , $/$: on sait faire
(beaucoup de calculs de pgcd).

Exemples de logiciels : Maple, Mathematica, certaines calculatrices (TI).

Avantage : calcul exact.

Inconvénients :

Nombres grossissent.

Complexité : dépend de la taille des nombres.

$\sqrt{\quad}$, \exp , \sin : on ne sait pas faire. . .

Plan de l'exposé

- Algorithmes pour l'arithmétique exacte
(~ école primaire)
- **Algorithmes pour l'arithmétique flottante**
(~ notation scientifique)
- Arithmétique par intervalles
- Complexité

Les nombres en notation scientifique : les flottants

Représentation

$$\pi \simeq 3.14 \times 10^0 = 0.314 \times 10^1 = 0.031 \times 10^2 = 314 \times 10^{-2} \dots$$

Nombre de chiffres utilisés dans la représentation : constant.

Calculs

$+$, $-$, \times , $/$: en temps constant

$\sqrt{\quad}$, \exp , \sin : en temps constant.

Arrondis

$$3.14 \times 10^0 + 5.36 \times 10^{-1} = 3.676 \times 10^0$$

sa représentation ne doit prendre que 3 chiffres

$$\Rightarrow 3.14 \times 10^0 + 5.36 \times 10^{-1} \simeq 3.68 \times 10^0.$$

Les nombres en notation scientifique : la norme IEEE 754 pour l'arithmétique flottante

Formats de représentation des nombres.

Ex. : nombre 32 bits :

<i>signe</i>	<i>exp.</i>	<i>mantisse</i>
1	8	24

Ex. : nombre 64 bits :

<i>signe</i>	<i>exp.</i>	<i>mantisse</i>
1	11	53

Résultat d'une opération = **arrondi du résultat exact** (si opération +, -, ×, / ou $\sqrt{\quad}$).

Avantages :

- reproductibilité des calculs d'une machine à l'autre
- algorithmes avec des erreurs optimales ou des majorations fines des erreurs
- preuves

Les flottants : problèmes

Propriétés algébriques perdues

les opérations ne sont plus associatives

\times n'est plus distributive par rapport à $+$

Exemple : programme de Gentleman

```
A := 1.0; B := 1.0;
while ( (A+1.0) - A) - 1.0 = 0.0 do
  A := 2.0*A;
while ( (A+B) - A) - B <> 0.0 do
  B := B+1.0;
result:=round(B);
```

Que calcule ce programme ?

Les flottants : encore de la recherche en cours

Algorithmes matériels de division avec une notation redondante :

on regarde seulement les 4 premiers bits du dividende courant.

Bug du Pentium :

$4195835.0 / 3145727.0 \Rightarrow 1.333739068$ au lieu de $1.33382044\dots$

Les flottants : encore de la recherche en cours

Algorithmes matériels de division

avec l'itération de Newton pour calculer $1/x$: la solution de l'équation $1/z - x = 0$ est $z = 1/x$.

Newton : $z_{k+1} = z_k - f(z_k)/f'(z_k)$.

Avec $f(z) = 1/z - x$, on obtient l'itération

$$z_{k+1} = z_k - \frac{1/z_k - x}{-1/z_k^2} = z_k - z_k(1 - z_k x)$$

Si on part d'une bonne approximation de $1/x$ (lue dans une table), on double le nombre de chiffres corrects à chaque étape \Rightarrow 3 ou 4 étapes suffisent.

Les flottants : encore de la recherche à faire !

Fonctions élémentaires

fonctions élémentaires : les calculer vite et bien

ystème	$\sin(10^{22})$ (Ng)
valeur exacte	-0.852200849767. . .
HP 48 GX	-0.852200849767
HP 700	0.0
IBM 3090/600S-VF AIX 370	0.0
Matlab 4.2c.1 Sparc	-0.8522
Matlab 4.2c.1 MacIntosh	0.8740
SG Indy	0.87402806
Sharp EL5806	-0.090748172
DEC Station 3100	NaN

Arithmétique flottante : attention, précision limitée !

Comptons jusqu'à 6 (Higham)

$$2 - 1$$

$$\left(\frac{1}{\cos(100\pi + \pi/4)} \right)^2$$

$$3.0 * (\tan(\operatorname{atan}(10000000.0)) / 10000000.0)$$

$$\left(\dots (\sqrt{\dots \sqrt{4}})^2 \dots \right)^2 \text{ (20 fois)}$$

$$5 \times \frac{(1 + e^{-100}) - 1}{(1 + e^{-100}) - 1}$$

$$\frac{\ln(e^{6000})}{1000}$$

Arithmétique flottante : attention, précision limitée !

Comptons jusqu'à 6 (Higham)

$$2 - 1$$

1.000000000000000000

$$\left(\frac{1}{\cos(100\pi + \pi/4)} \right)^2$$

2.00000000000001106

$$3.0 * (\tan(\operatorname{atan}(10000000.0)) / 10000000.0)$$

3.0000000030728171

$$\left(\dots (\sqrt{\dots \sqrt{4}})^2 \dots \right)^2 \text{ (20 fois)}$$

4.0000000006294343

$$5 \times \frac{(1 + e^{-100}) - 1}{(1 + e^{-100}) - 1}$$

NaN

$$\frac{\ln(e^{6000})}{1000}$$

$+\infty$

Arithmétique flottante : attention aux arrondis !

Les faits :

- 1982 : the Vancouver stock exchange introduced an index with nominal value 1000.000
- after each transaction, it was recomputed and truncated to the third place to the right of the decimal point
- after 22 months the index was 524.881
- the correct value was 1098.811

L'explication :

toutes les erreurs d'arrondis sont dans le même sens (" truncation ").

(d'après la page Web de Pete Stewart)

Flottants en précision multiple : plus de chiffres

Principe :

utiliser des représentations de longueur fixée
(\Rightarrow le produit de 2 nombres a la même longueur que les multiplicandes)
mais plus longues que celles des flottants machine.

Complexité :

\leq celle du calcul sur des entiers de même longueur.

Avantage :

plus de précision.

Inconvénient :

toujours aucune garantie sur les résultats.

Plan de l'exposé

- Algorithmes pour l'arithmétique exacte
(~ école primaire)
- Algorithmes pour l'arithmétique flottante
(~ notation scientifique)
- **Arithmétique par intervalles**
- Complexité

Calcul garanti avec les flottants : arithmétique par intervalles

(Moore 1966, Kulisch 1983, Neumaier 1990, Rump 1994, Alefeld and Mayer 2000. . .)

Principe

Nombres remplacés par des intervalles.

π remplacé par $[3.14159, 3.14160]$

Vecteurs remplacés par des vecteurs d'intervalles.

Matrices remplacées par des matrices d'intervalles.

Intérêt : incertitudes sur les données (de mesure. . .) prises en compte.

Calculs

Calcul garanti avec les flottants : arithmétique par intervalles

$$[-2, 3] + [5, 7] = [3, 10]$$

$$[-3, 2] * [-3, 2] = [-6, 9] \text{ est différent de } [-3, 2]^2 = [0, 9]$$

$$[-3, 2]/[0.5, 1] = [-6, 4]$$

$$X \diamond Y = \{x \diamond y / x \in X, y \in Y\}$$

$$\exp[-2, 3] = [\exp(-2), \exp(3)]$$

car exp est une fonction croissante.

$$\sin[\pi/3, \pi] = [0, 1]$$

attention, sin n'est pas monotone.

Sur ordinateur : utiliser les arrondis dirigés prévus par la norme IEEE-754.

Arithmétique par intervalles : avantages

Calcul garanti

le résultat cherché appartient à l'intervalle calculé.

Information globale

on sait encadrer l'image d'une fonction sur tout un intervalle.

Théorème de Brouwer-Schauder : si $f(I) \subset I$ alors f admet un point fixe dans I .

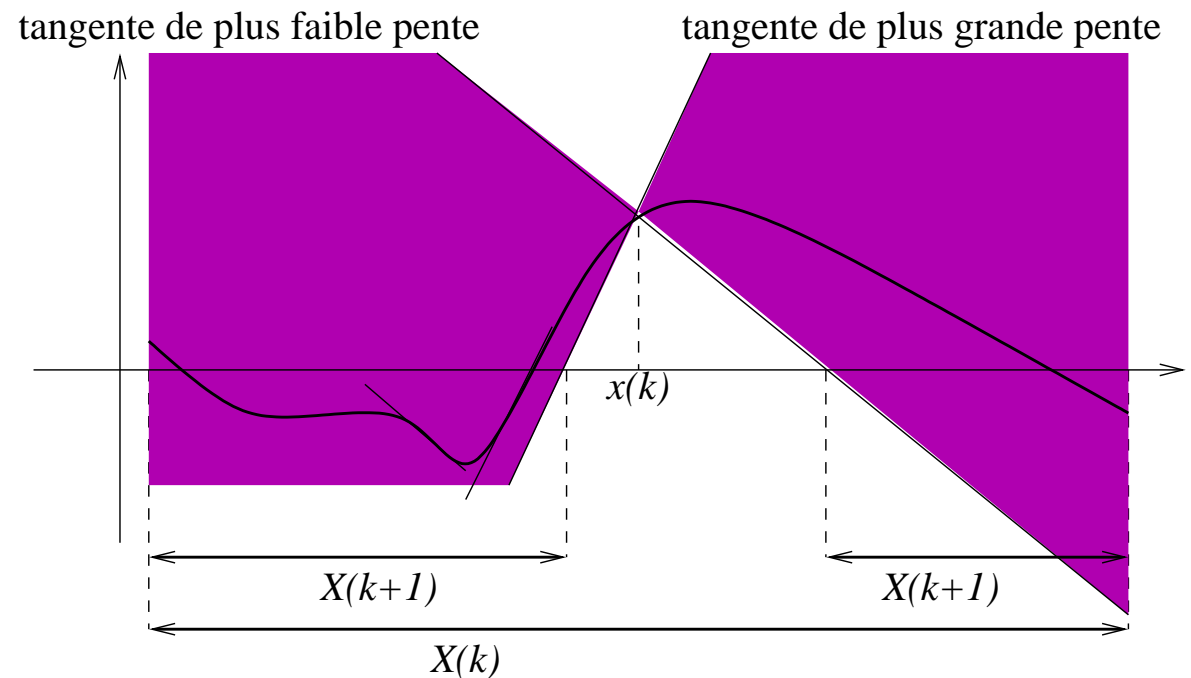
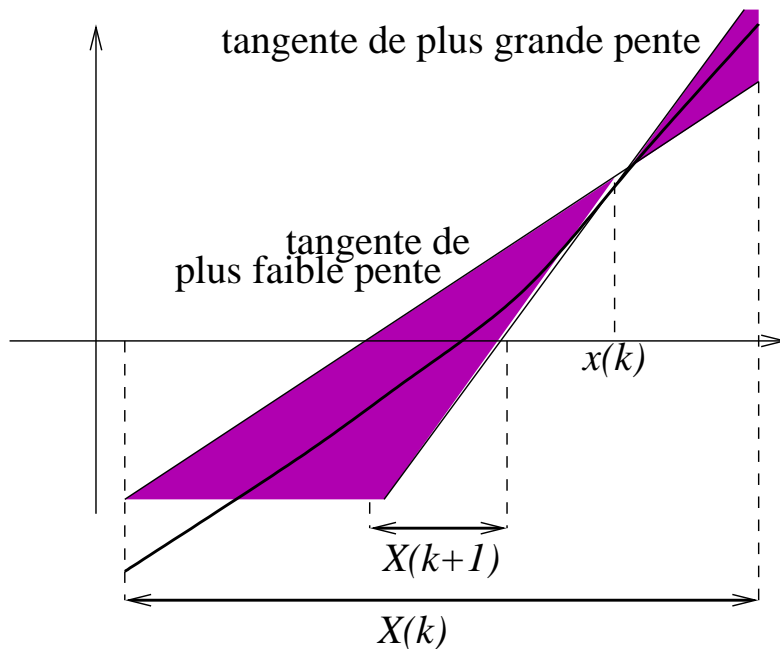
Application à Newton.

Algorithme de Hansen pour l'optimisation globale.

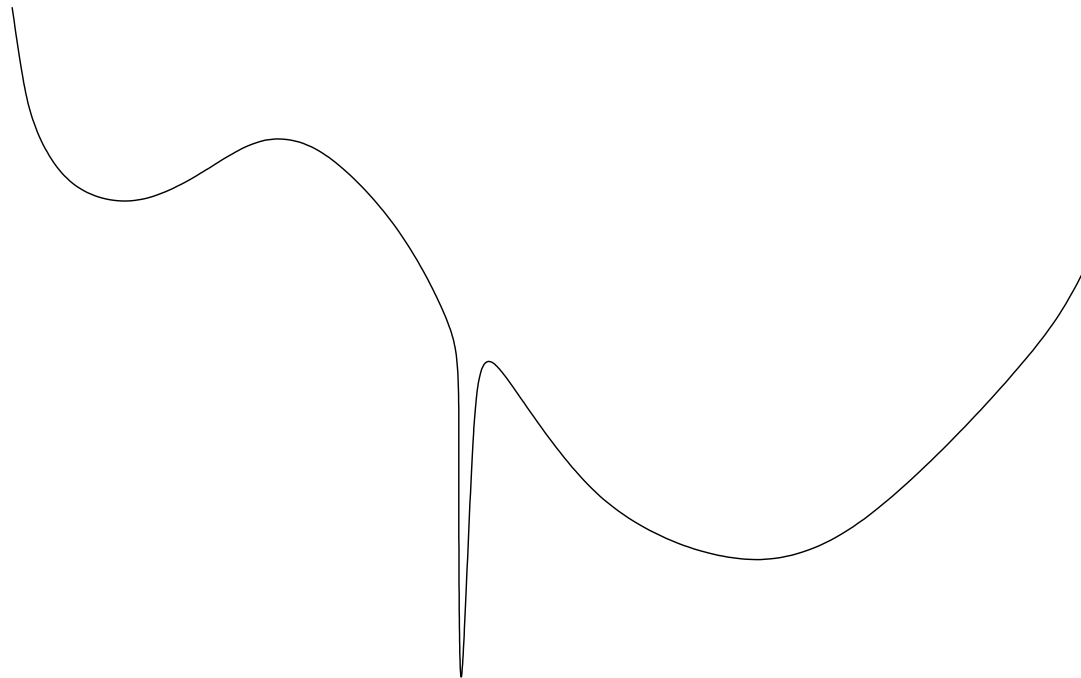
Arithmétique par intervalles : exemple d'algorithme

Algorithme de Newton par intervalles

(Hansen & Greenberg 1983, Mayer 1995, van Hentenryck et al. 1997. . .)

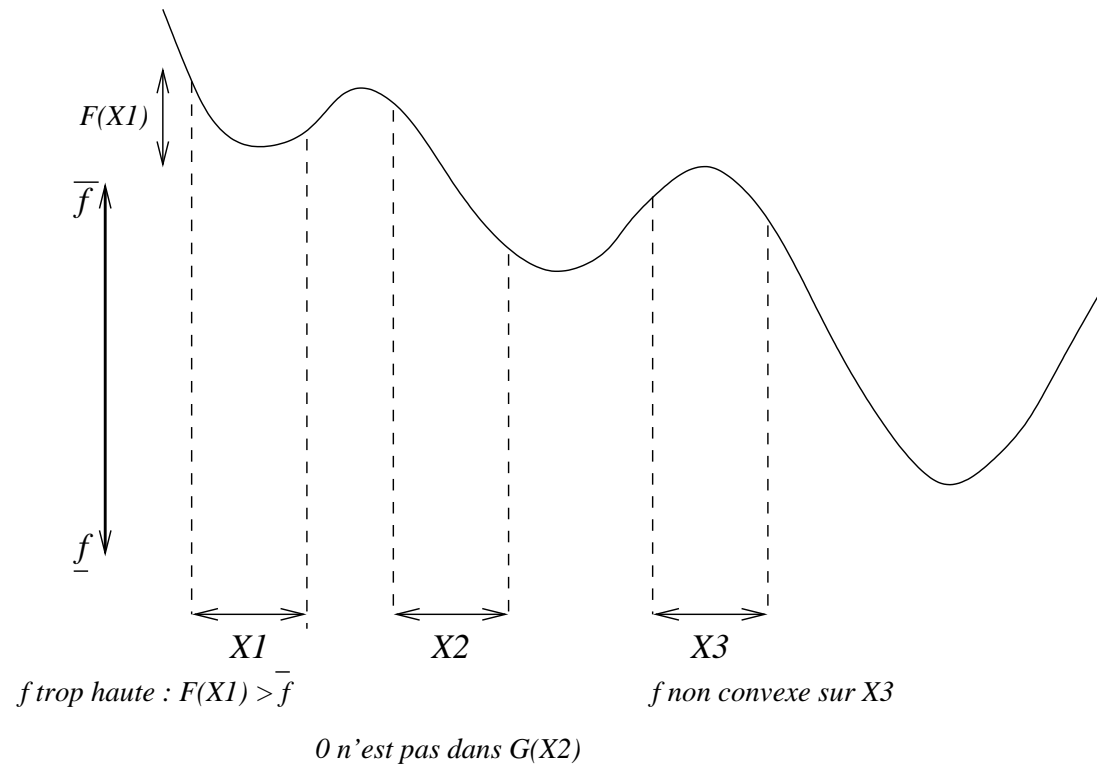


Arithmétique par intervalles : optimisation globale d'une fonction continue



Arithmétique par intervalles : algorithme de Hansen

(Hansen 1992, Ratschek and Rokne 1988, Baker Kearfott 1996)



Arithmétique par intervalles : algorithme de Hansen

\mathcal{L} = liste des pavés en attente $:= \{X_0\}$

tant que $\mathcal{L} \neq \emptyset$ faire

sortir X de \mathcal{L}

rejeter X ?

oui si $F(X) > \bar{f}$

oui si $\text{Grad}F(X) \not\cong 0$

oui si $HF(X)$ à diagonale non > 0

réduire X

Newton sur le gradient

résoudre $Y \subset X$ tel que $F(Y) \leq \bar{f}$

couper Y en **2** : Y_1 et Y_2

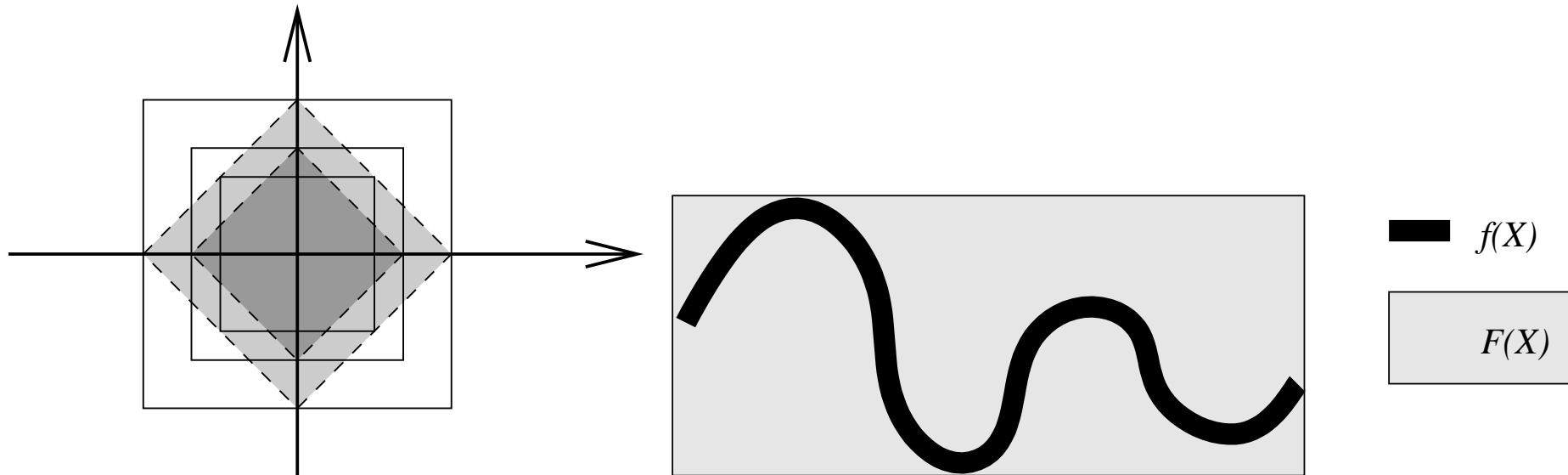
ranger Y_1 et Y_2 dans \mathcal{L}

Arithmétique par intervalles : problèmes

Décorrrelation des variables ou *variable dependency*

$$I * I = \{x * y ; x \in I, y \in I\} \neq I^2 = \{x^2 ; x \in I\}$$

Effet enveloppant ou *wrapping effect*



Arithmétique par intervalles plus de précision

Aspirateur autonome

- par intervalles : ne casse rien mais passe loin du vase en porcelaine de Chine
- en précision arbitraire : passe près du vase en porcelaine de Chine et le casse peut-être
- par intervalles en précision arbitraire : passe près du vase en porcelaine de Chine et ne le casse pas !

Choix d'arithmétique

Arithmétiques

- exacte : entière et rationnelle
- flottante en précision fixe
- flottante en précision arbitraire
- intervalles en précision fixe
- intervalles en précision arbitraire
- mais aussi : en ligne, stochastique, paresseuse. . .

Comment choisir ? selon ses besoins

- rapidité
- précision
- garantie
- . . .

Plan de l'exposé

- Algorithmes pour l'arithmétique exacte
(~ école primaire)
- Algorithmes pour l'arithmétique flottante
(~ notation scientifique)
- Arithmétique par intervalles
- **Complexité**

La classe P

Définition

$P = \{ \text{problèmes pour lesquels il existe un algorithme de résolution en temps polynomial en la longueur des entrées} \}$.

Exemples

addition : $\mathcal{O}(n)$ pour 2 entiers à n chiffres

multiplication : $\mathcal{O}(n \log n \log \log n)$ pour 2 entiers à n chiffres

division : $\mathcal{O}(M(n))$ pour 2 entiers à n chiffres

pgcd : $\mathcal{O}(M(n) \log n)$ pour 2 entiers à n chiffres

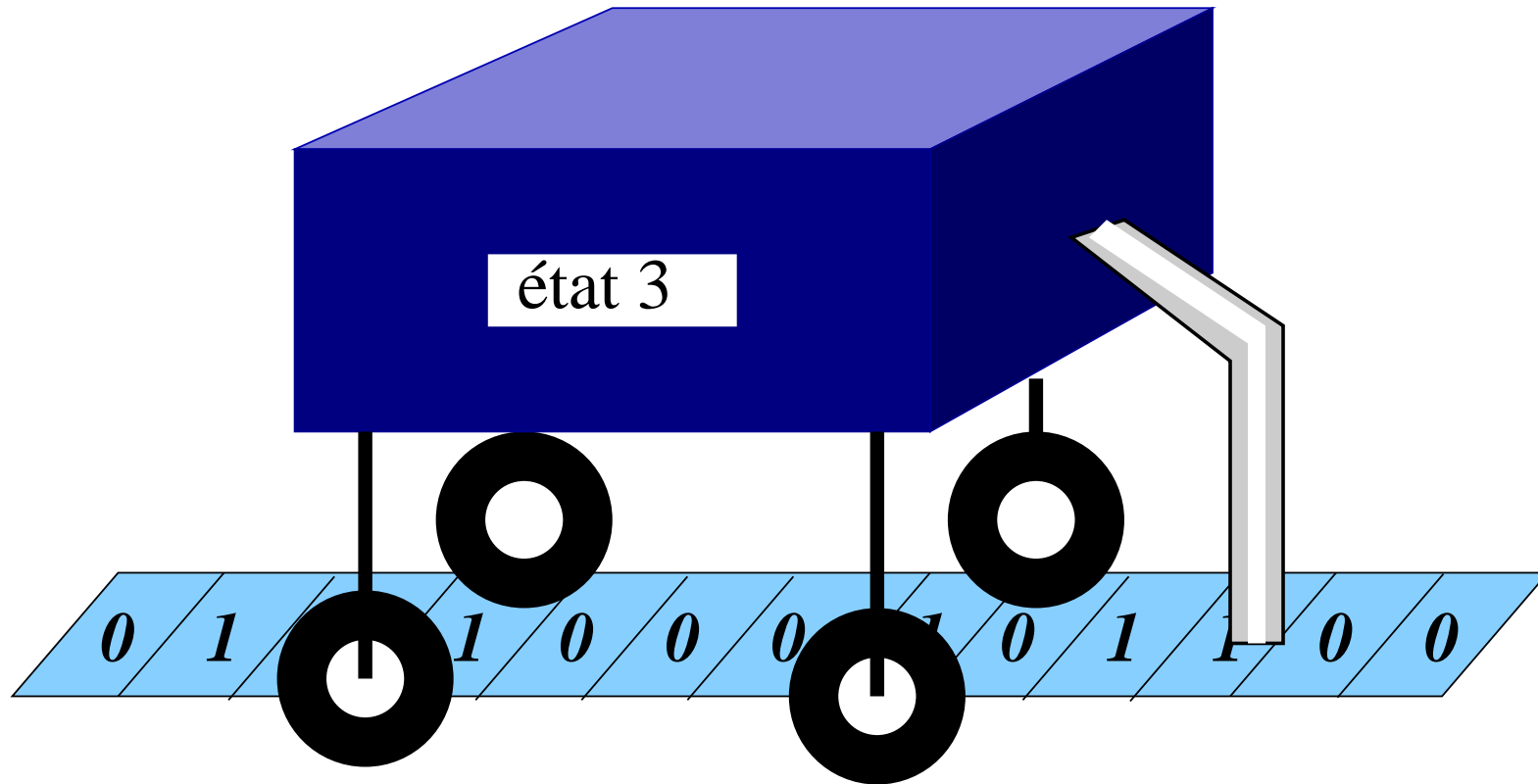
tester la primalité d'un entier à n chiffres (août 2002)

Sur quel ordinateur ?

sur un modèle théorique de machine appelé machine de Turing. . .

c'est la même sur tout ordinateur réel et avec tout langage de programmation !

Machine de Turing



Réduction d'un problème à un autre

Ex : la division se ramène à la multiplication.

Pour calculer A/B avec A et B deux entiers à n chiffres : on calcule $X = \beta^n/B$ un inverse approché de B (multiplié par β^n) puis $Q = (A \times X)/\beta^n$.

Si on veut le reste : $R = A - B \times Q$.

Newton modifié pour calculer X :

$$X_{2k} = X_k \beta^k + X_k \left(\beta^{2k} - (B_{2k} \times X_k) / \beta^k \right) / \beta^k$$

avec X_k entier à k chiffres, B_k l'entier constitué à partir des k chiffres "de gauche" de B .

Arrêt au bout de $\lceil \log_2 n \rceil$ étapes.

Réduction d'un problème à un autre

Complexité :

- à l'étape k : deux multiplications de 2 nombres à k chiffres et deux $+/-$
complexité : $2M(k) + 2k$
- au total :

$$\sum_{i=1}^{\log n} 2M(2^i) \leq CM(n)$$

puisque $M(n) \leq D.n^2$, on a $\sum_{i=1}^{\log n} M(i) \leq 2M(2n)$.

La classe NP

Définition

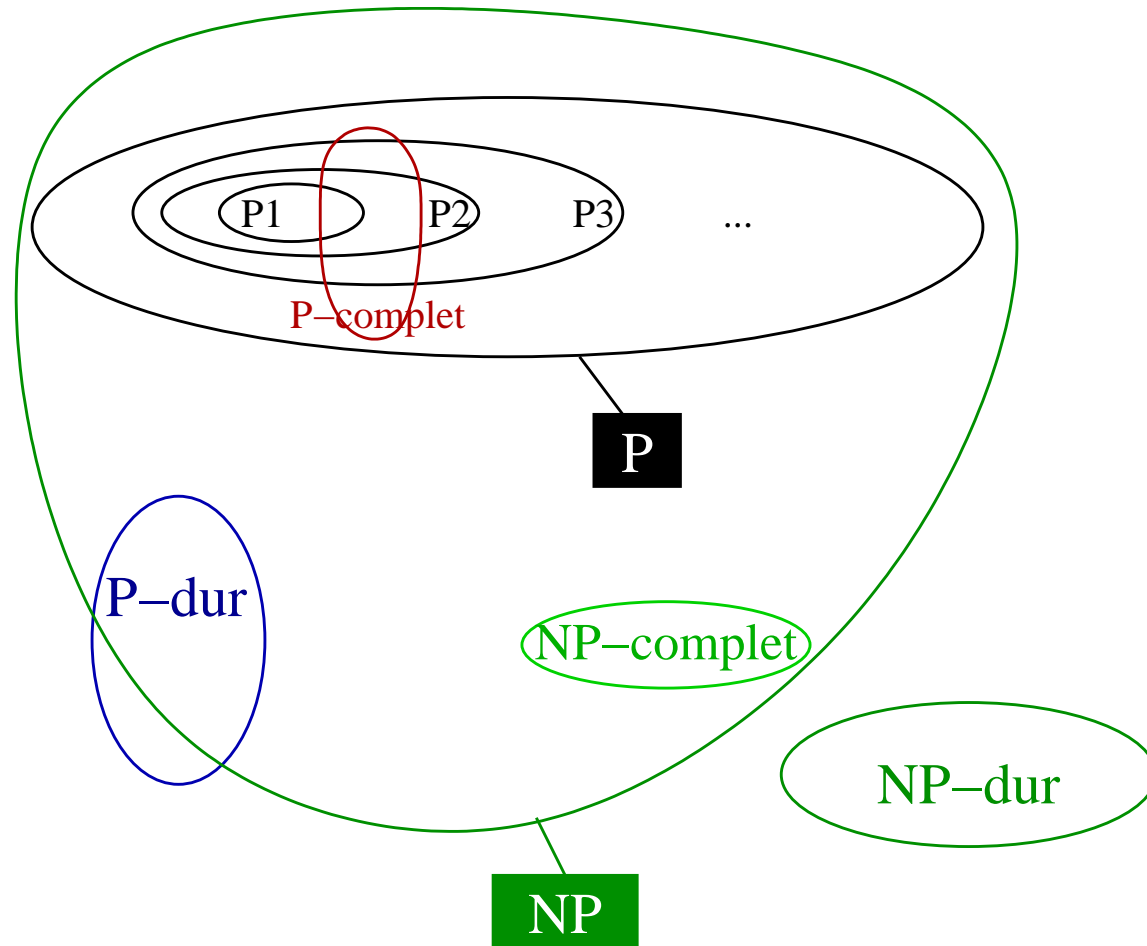
$NP = \{ \text{problèmes pour lesquels il existe un algorithme de vérification de la solution en temps polynomial en la longueur des entrées} \}$.

Exemples

factorisation d'entiers : il est facile de vérifier une décomposition

isomorphisme de graphes : si la bijection candidate est donnée, facile !

Hiérarchies



Conclusions

On sait faire

les opérations arithmétiques exactes et flottantes
l'algèbre linéaire

...

On ne sait pas faire

la factorisation d'entiers (?)
l'isomorphisme de graphes

...

On essaie de classer les problèmes

les uns par rapport aux autres : P versus NP

P, P-complet, P-dur

NP, NP-complet, NP-dur

pour justifier pourquoi on ne sait pas faire ! (sauf si $P = NP$)

Références

Vulgarisation

- *Histoire d'algorithmes - Du caillou à la puce*, J.-L. Chabert et al., Belin 1994
- *Logique, informatique et paradoxes*, J.-P. Delahaye, Belin 1995
- *Histoire des codes secrets*, S. Singh, Le livre de poche 2001

Modérément spécialisés

- *Qualité des calculs sur ordinateurs - Vers des arithmétiques plus fiables ?*, coordonné par M. Daumas et J.-M. Muller, Masson 1997
- *La cryptologie moderne*, A. Canteau et F. Lévy-dit-Véhel, Revue Armements, 2001, <http://www-rocq.inria.fr/codes/Anne.Canteau/crypto-moderne.pdf>
- *Arithmétique en précision arbitraire*, P. Zimmermann, Réseaux et systèmes répartis, vol. 13, no 4-5, 2001, <http://www.inria.fr/rrrt/rr-4272.html>
- *Arithmétique par intervalles*, N. Revol, Réseaux et systèmes répartis, vol. 13, no 4-5, 2001, <http://www.inria.fr/rrrt/rr-4297.html>

Spécialisés

- *Modern computer algebra*, J. von zur Gathen and J. Gerhard, CUP, 1999
- *Arithmétique des ordinateurs*, J.-M. Muller, Masson 1989
- *Elementary functions*, J.-M. Muller, Birkhauser, 1997.
- *Interval methods for systems of equations*, A. Neumaier, CUP, 1990
- *Computational complexity*, C. Papadimitriou, Addison Wesley, 1994