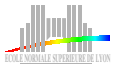
LinBox : Algèbre linéaire exacte sur les corps finis et applications

Pascal Giorgi

Laboratoire LIP - École Normale Supérieure de Lyon

LIP  - ENS Lyon  - CNRS  - INRIA  - UCBL 

Projet LinBox

- Projet international USA-Canada-France, fincancement NSF/CNRS
  31 chercheurs
  $\Longrightarrow$ algébre linéaire exacte
- Site web: *www.linalg.org*
- Bibliothèque générique C++, licence GPL, (80000 lignes de code)

Principaux développements:

- algorithmes (rang, systémes linéaire,...)
- matrices (boîtes noires, conteneurs)
- domaines de calcul (corps finis, entiers, rationnels)
- généricité (plug&play)

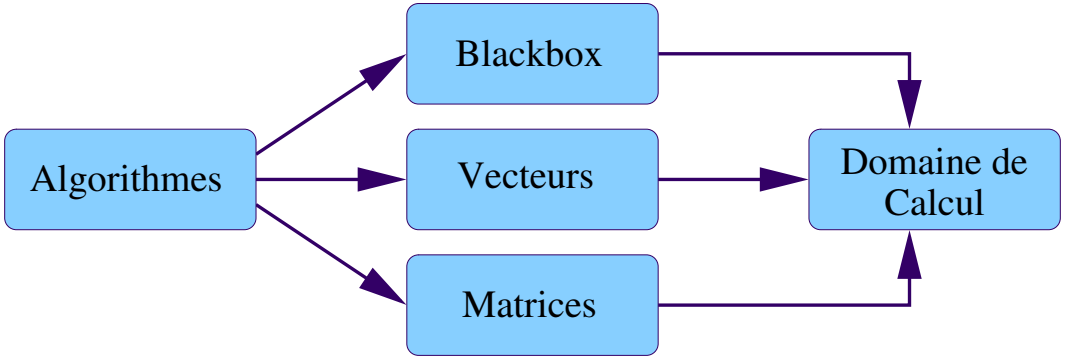
Plan de l'exposé

- Strucure et généricité de LinBox corps finis, Boîte noire, Matrice
- Algorithmes sur un corps fini méthode d'élimination, méthode itérative (Krylov/Lanczoz)
- Extension des problèmes sur $\mathbb{Z}$ Méthodes d'approximation et interpolation Systèmes linéaires diophantiens

Structure et généricité de LinBox

Structure de LinBox

- Trois niveaux d'implantation (permettant la réutilisation et la reconfiguration)



- Matrices, Blackbox, Domaines de calcul respectent des interfaces.
- Interface= classe C++ virtuelle, template *archetype*:

  - définit l'interface commune aux objets.
  - fournit une instance de code compilé.
  - contrôle l'explosion de code.
  - utilisation optionnelle.

Structure des corps finis (domaine des coefficients)

- Encapsulation des types : eléments, générateur aléatoire d'éléments.
- Eléments: aucune information sur le corps d'appartenance.
- Corps: méthodes affectation, égalité, arithmétique, IO pour les éléments:

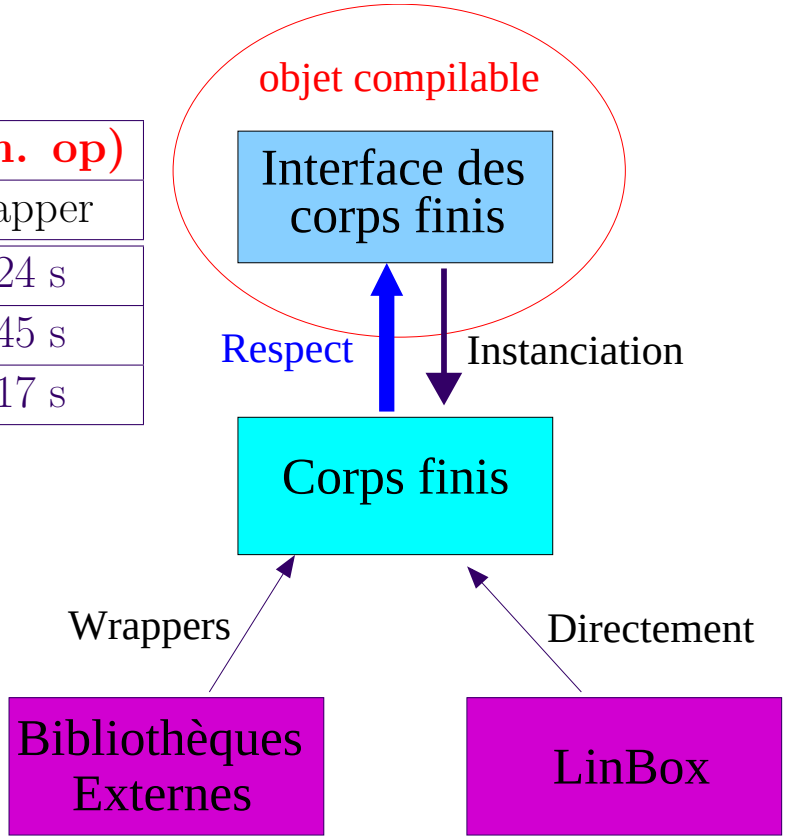
x = y	:	F.assign(x,y)
x == y	:	F.areEqual(x,y)
x = y + z	:	F.add(x,y,z)
cout<< x	:	F.write(cout,x)

- Implantation : directe, plug-ins (au travers de wrappers).

Interfaces des Corps Finis

Timings for different LinBox field levels (arith. op)				
Library	Z/pZ	loop	Directly	Wrapper
NTL::ZZ_p	1978146594666	100.000	0.24 s	0.24 s
//	//	1.000.000	2.45 s	2.45 s
NTL::zz_p	553	300.000	0.16 s	0.17 s

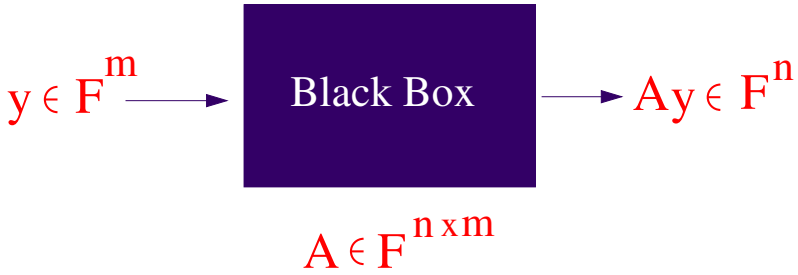
- Interface:
  - objet à part entière
  - validations des corps finis
  - évite la réécriture (template)



Wrappers et arithmétiques

- NTL (*www.shoup.net/ntl*)
  - entiers simple-multi précision, polynômes & arithmétique modulaire
  - multiplication flottante ($a * b \bmod p = a * b - \lfloor \frac{a*b}{p} \rfloor * p$)
- Givaro (*www-apache.imag.fr/software/givaro/*)
  - entiers simple precision (16,32,64 bits)
  - arithmétique modulaire, tabulée, générateur, Montgomery
- LiDIA (*www.informatik.tu-darmstadt.de/TI/LiDIA/*)
  - polynômes, entiers simple-multi précision
  - arithmétique modulaire, générateur
- LinBox (*www.linalg.org*)
  - classe générique (+,-,*,/) & arithmétique modulaire
  - spécialisation (entiers 16,32,64 bits, flottants double précision)

Structure des matrices boîtes noires (Blackbox)



- Modèle des Blackbox paramétré par une classe de vecteurs (domaine d'application).
- Domaine de calcul passé comme paramètre ou specifié comme attribut.
- Seule l'application à un vecteur est autorisée :

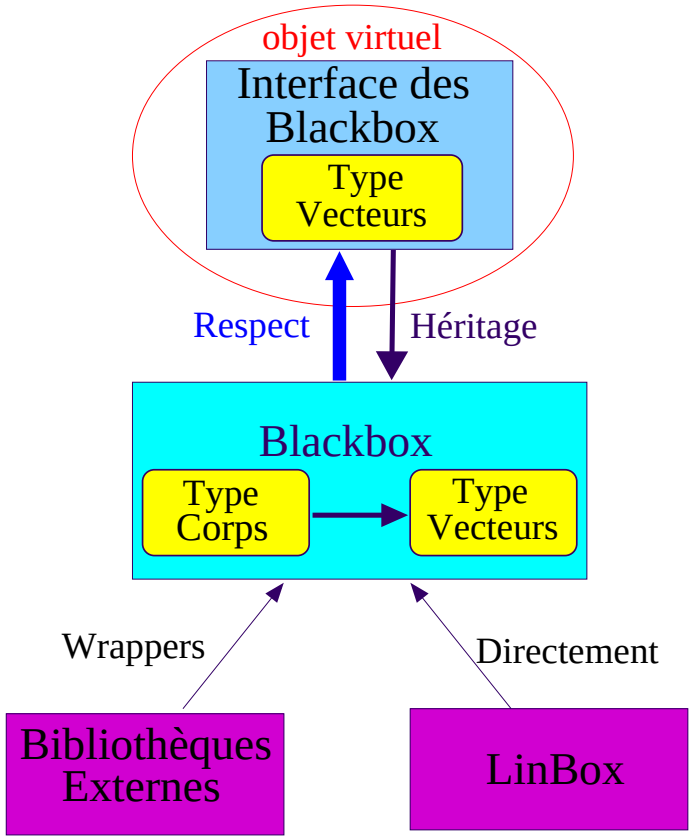
$$x = Ay \quad : \quad \text{A.apply(x,y)}$$
$$x = A^T y \quad : \quad \text{A.applyTranspose(x,y)}$$

- Récupération des dimensions de la matrice:

A.rowdim()
A.coldim()

Interfaces des BlackBox

- Interface: purement virtuelle
- Paramétrée par un type de vecteur (compatible STL)
- Vecteurs: définis sur un type d'élément
- LinBox fournit 3 types de vecteurs :
  - Dense vector
  - Sparse sequence vector
  - Sparse map vector



Opérations arithmétiques sur les matrices

- Opération hybride (exacte-numérique):
  Approche **FFLAS** package [Dumas-Gautier-Pernet 2002]
  - conversions éléments $\Longleftrightarrow$ flottants
  - opération numérique (BLAS)
  - conversions flottants $\Longleftrightarrow$ éléments

  Interêt:
  - Minimise le nombre de réductions modulaires
  - Avantage des routines numériques BLAS (bloc, optimisation de cache)

  $\Rightarrow$ 67.58s pour une multiplication d'ordre 5000 sur GF(101)

- LinBox: domaine générique au travers d'une interface matrice BLAS

Algorithmes sur les corps finis

Algèbre linéaire sur un corps

- Depuis 1969, multiplication matrices d'ordre n en moins que $O(n^3)$
  [Strassen 1969]: $O(n^{2.81})$
  ...

- Meilleurs algorithmes $\Longrightarrow$ multiplication matrices (complexité $O(n^\omega)$)
  [Strassen 1969]: inversion, systèmes linéaires, determinant
  [Bunch-Hopcroft 1974]: généralisation matrices non génériques
  [Ibarra-Moran-Hui 1982]: cas singulier: rang, noyaux

- Cas creux
  [Wiedemann 1986], systéme linéaire en $O(n^2)$, probabiliste

Méthode à base d'élimination

Elimination de Gauss $\Rightarrow$ simplification de la matrice

- Gauss $\Leftrightarrow$ Décomposition LUP
- Algorithme de décomposition LSP [Ibarra-Moran-Hui 1982]
- Algorithme par bloc en $O(n^{\omega})$ $\Longrightarrow$ { multiplication matrices ; résolution systèmes triangulaires }
  - pivot = matrice triangulaire
  - elimination = multiplication et addition matrices

Implantation via les bibliothèques numériques BLAS

- Multiplication matrice:
  → FFLAS package

- Systèmes linéaires triangulaires: algorithme bloc récursif

  Résolution hybride (exacte-numérique):
  recursif → solution représentable exactement sur les flottants
  résolution numérique via BLAS et conversion

FFPACK package [Dumas-Giorgi-Pernet 2004]
→ implantation efficace (en-place) LSP/LQUP
→ Intégration à LinBox (interface avec MAPLE)

Performance FFPACK package

- Décomposition LQUP sur GF(101)

n	400	700	1000	2000	3000	5000
LQUP	0.05s	0.18s	0.46s	2.80s	7.79s	32.9s
Fgemm	0.04s	0.23s	0.62s	4.28s	14.72s	67.58s
Ratio	1.25	0.78	0.74	0.65	0.53	0.48

Complexité arithmétique pour $\omega = 3$: LQUP = 2/3 * Multiplication matrices

- Inversion sur GF(101)

n	400	700	1000	2000	3000	5000
Inv	0.18s	0.70s	1.79s	10.84s	32.33s	139.5s
Fgemm	0.04s	0.23s	0.62s	4.28s	14.72s	67.58s
Ratio	4.50	3.04	2.89	2.53	2.20	2.07

Complexité arithmétique pour $\omega = 3$: Inverse = 4/3 * Multiplication matrices

Méthode itérative (Krylov)

- Conservation de la structure des matrice (ex: creuse)
- Algorithmes basés sur le produit matrice-vecteur

  ex: [Wiedemann 1986]

$$A^{-1}b = \sum_{i=0}^{\dots} c_i.A^i b \, , \; c_i \in \mathsf{K}$$

Systèmes linéaires creux

- Algorithme proposé par [Wiedeman 1986]
  Soient $A \in \mathsf{K}^{n\times n}$, $b \in \mathsf{K}^n$ et $\Pi^{A,b}(\lambda)$ polynome minimal $\{A^i b\}_{i=0}^{\infty}$.
  $\Pi^{A,b}(A)b = c_0.b + c_1.Ab + +.. + c_d.A^d b = 0$, $c_0 \neq 0 \in \mathsf{K}$

  $\Rightarrow b = A.\frac{1}{c_0}(c_1.b + c_2.Ab + ... + c_d.A^{d-1}b) = A.y$

  Lemme: [Wiedemann 1986]
  Soit $u \in \mathsf{K}^n$ *aléatoire:*

  (forte probabilité) Polynôme minimal $\{A^i b\}_{i=0}^{\infty}$ = Polynôme minimal $\{uA^i b\}_{i=0}^{\infty}$

- Calcul générateur: algorithme théorie des codes [Berlekamp-Massey 1969]
- Algorithme par blocs [Coppersmith 1994,Giorgi-Jeannerod-Villard 2003]
  parallélisme, petit corps

Matrices à coefficients sur $\mathbb{Z}$

Problématique différente:
taille des données $\Rightarrow$ la complexité

- Transposition des méthodes (corps finis $\rightarrow$ entiers): pas satisfaisante
- Approche classique :
  théorème des restes chinois ($O(n^{1+\epsilon})\times$ coût algébrique)
- Cas des systémes linéaires:
  restes chinois : $O(n^{\omega+1+\epsilon})$
  approximation p-adique : $O(n^{3+\epsilon})$
  high order lifting [Storjohan 2002]:$O(n^{\omega+\epsilon})$

Systèmes linéaires sur $\mathbb{Z}$

- Algorithme p-adique[Moenck et Carter 1979, Dixon 1982]:

- soient $A \in \mathbb{Z}^{n\times n}$, $b \in \mathbb{Z}^n$ trouver $x \in \mathbb{Q}^n / Ax = b$
  idée: calculer $Ay \equiv b \bmod p^{k+1}$, p premier

$$y = x_{[0]} + x_{[1]}p + x_{[2]}p^2 + ... + x_{[k]}p^k$$

- k choisi assez grand (Hadamard, Cramers: $k \approx n \log n$)
  $\Rightarrow$ reconstruction de x (fractions continus y/p^{k+1})

- calcul des chiffres p-adiques : systèmes linéaire sur $\mathbb{Z}_p$

j=0:

$$A.x_{[0]} \equiv b \bmod p$$

j=1:

$$A.(x_{[0]} + px_{[1]}) \equiv b \bmod p^2 \, , b_{[0]} = \frac{b - A.x_{[0]}}{p}$$

$$A.x_{[1]} \equiv b_{[0]} \bmod p$$

j=2:

$$A.(x_{[0]} + px_{[1]} + p^2x_{[2]}) \equiv b \bmod p^3 \, , b_{[1]} = \frac{b_{[0]} - A.x_{[1]}}{p}$$

$$A.x_{[2]} \equiv b_{[1]} \bmod p$$

j=i+1:

$$A.(x_{[0]} + x_{[1]}p + \ldots + x_{[i+1]}p^{i+1}) \equiv b \bmod p^{i+2} \, , b_{[i]} = \frac{b[i-1] - A.x_{[i]}}{p}$$

$$A.x_{[i+1]} \equiv b_{[i]} \bmod p$$

implantation dans LinBox

- définition d'interface pour l'approximation (conteneur/itérateur)
- Résolution des systèmes sur $\mathbb{Z}_p$ via :
  - Wiedemann
  - Inversion + produit matrice-vecteur
- optimisations :
  - utilisation maximale: FFLAS et FFPACK
  - construction approximation: pas de bébé / pas de géant
  - reconstruction rationnelle: seulement les facteurs inconnus
- Comparaison avec la bibliothèque NTL

Performances: méthode d'elimination

- Systèmes avec coefficients 32bits

n	50	150	350	500	650	1000
LinBox	0.08s	0.96s	7.47s	18.57s	36.65s	115.43s
NTL	0.06s	1.92s	24.07s	72.49s	159.88s	688.57s

- Systèmes avec coefficients 128bits

n	50	150	350	500	650	1000
LinBox	1.67s	18.18s	126.78s	311.56s	602.5s	1870.85s
NTL	0.39s	11.69s	141.86s	448.3s	922.59s	4124.55s

Systèmes linéaires diophantiens

idée : [Giesbrecht 1997] En combinant 2 solutions rationnelles y_1, y_2 de dénominateur d_1, d_2 on obtient une nouvelle solution rationnelle y_3 de dénominateur $d_3 = \gcd(d_1, d_2)$

exemple:

$$A.y_1 = A.(\frac{1}{2}.x_1) = b\,, \quad x_1 \in \mathbb{Z}^n$$

$$A.y_2 = A.(\frac{1}{3}.x_2) = b\,, \quad x_2 \in \mathbb{Z}^n$$

$$\gcd(2,3) = 1 = 2 * 2 - 1 * 3$$

$$A.(2 * x_1 - x_2) = 4b - 3b = b$$

Lemme: *soient* $y_1, y_2 \in \mathbb{Q}^n$ *2 solutions de* $Ax = b$
soient d, s_1, s_2 *tels que:* $d = \gcd(d(y_1), d(y_2)) = s_1 d(y_1) + s_2 d(y_2)$ *alors:*

$$y_3 = \frac{s_1 d(y_1) y_1 + s_2 d(y_2) y_2}{d} \text{ est une solution de } Ax = b$$

Algorithme (solutions diophantiennes):
combiner plusieurs solutions rationnelles $\Rightarrow$ dénominateur=1

Problème: pas de solutions diophantiennes
$\rightsquigarrow$ certificat d'inconsistence sur $\mathbb{Z}$ [Giesbrecht-Lobo-Saunders 1998]

$$u \in \mathbb{Z}^n,\ uA \equiv 0 \bmod d\,,\ ub \not\equiv 0 \bmod d$$

Algorithme: (mise en place)

- Solutions rationnelles aléatoires ([Kaltofen-Saunders 1991])
  perturbation du système
- Certifier l'inconsistence [Giesbrecht-Lobo-Saunders 1998]
  vecteur aléatoire du noyau
- Certifier la minimalité du dénominateur [Mulder-Storjohann 2004]
  $z \in \mathbb{Q}^{1\times n}/zA \in \mathbb{Z}$, $d(zb)$ est un facteur du dénominateur
- Convergence (extension anneau, préconditionnement)
  O(1) solutions rationnelles

Complexité: $O\ (n^{\omega})$

conclusion

LinBox:

- Boîtes à outil: Corps finis, Blackbox, Matrices, Vecteurs
- Plug-ins (BLAS, Maple, NTL)
- Méthodes itératives: efficace pour les matrices creuses
- Implantations efficaces (corps finis, entiers)
- Combinaison solutions rationnelles $\rightarrow$ solution diophantienne

Perspectives

- Développer: outils pour les entiers (interface d'anneaux, CRT)
- Implanter: systèmes linéaires diophantiens
- Généraliser: interaction entre logiciels de calcul formel (ROXANE,Maple)

Questions:

⋆ en pratique: approche classique (p-adic, CRT) sc vs high order lifting ?
⋆ Peut-on résoudre un système linéaire singulier directement ?