Arithmétique des corps finis dans la bibliothèque LinBox

Pascal Giorgi & Gilles Villard

Équipe *Arénaire*

Laboratoire de l'informa tique du
parallélisme
ENS lyon - CNRS - INRIA

Introduction

- Il existe déjà plusieurs bibliothéques d'algèbre linéaire (ALP, Givaro, ...).
- LinBox est une bibliothèque d'algèbre linéaire utilisant ces bibliothèques.
- Utilisation générique d'objets exogènes (Corps Finis, Grandes Matrices ...).
- LinBox fournit une interfaces uniques pour chaque types d'objets.
- Facilité de comparaison des bibliothèques utilisées.

CORPS FINIS.

Plan

I) Généricité des corps finis dans LinBox.

II) Différentes arithmétiques de corps finis.

III) Efficacité de ces arithmétiques et surcoût de LinBox.

LinBox et les corps finis

- Deux Types de corps finis : $\mathbb{Z}/\mathbf{p}\mathbb{Z}$ et $\mathbf{GF}(\mathbf{p}^{\mathbf{k}})$.
- Définition de Domaine de corps finis pour chaque bibliothèques.

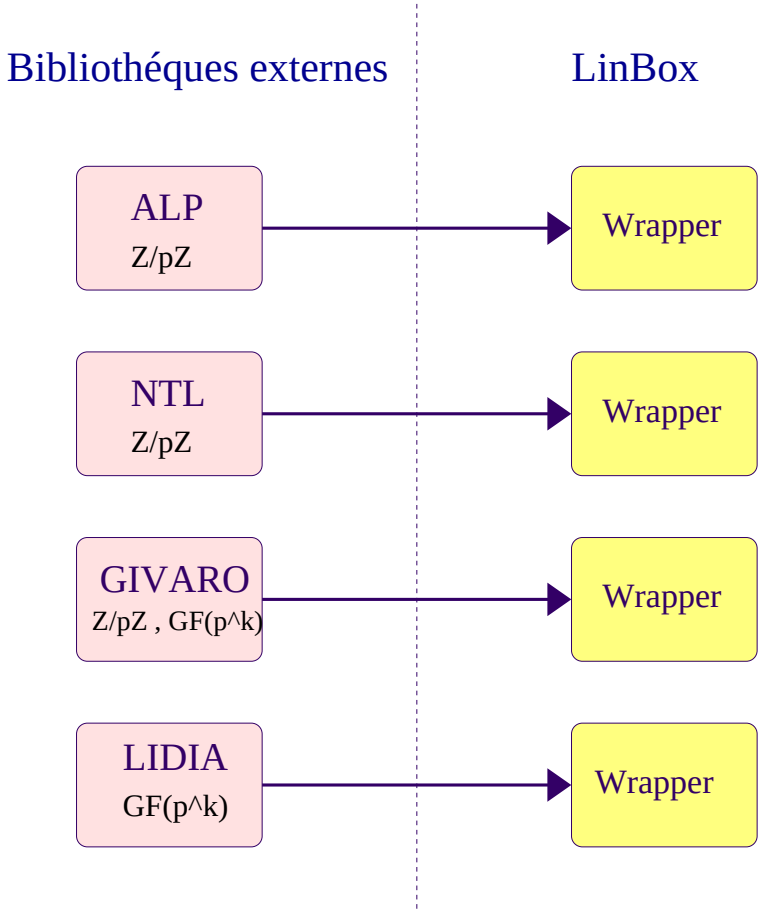
  Exemples :

$$\mathbf{Bib\_zpz} = \{\mathbb{Z}/\mathbf{p}\mathbb{Z} \ / \ \mathbf{p} \in \mathbb{N} \text{ et } \mathbf{p} \text{ premier}\}$$
$$\mathbf{Bib\_gfq} = \{\mathbf{GF}(\mathbf{p}^{\mathbf{k}}) \ / \ \mathbf{p}, \mathbf{k} \in \mathbb{N} \text{ et } \mathbf{p} \text{ premier}\}$$

  Bib représente le nom de la bibliothèque.

- Intégration des domaines par des *Wrappers*

- *Les différents wrappers possèdent tous les mêmes méthodes (noms, paramètres et actions).*



Exemple de code LinBox

```
 template<class Domain >
void Fct(Domain& field)
{   typedef typename Domain::element elt;
  elt a,b,r;                  // Déclaration de a,b,r comme éléments de "Domaine"

  field.init(a);
  field.init(b);              // Initialisation de a,b,r sur le corps fini "field"
  field.init(r);

  field.read(cin,a);
  field.read(cin,b);

  field.mul(r,a,b);           // r ← a * b dans le corps fini "field"

   field.write(cout,r);
}
```

```
void main()
{
  // Declaration of a Z/pZ field over Givaro library with p=53.
  givaro_zpz<Std16> K(53);

  // Declaration of a GF(p^k) field over Lidia library with p=53 and k= 5.
  lidia_gfq Q(53,5);

  cout<<" Z/pZ of Library Givaro \n";
  Fct(K);

  cout<<" GF(p^k) of Library Lidia \n";
  Fct(Q);
}
```

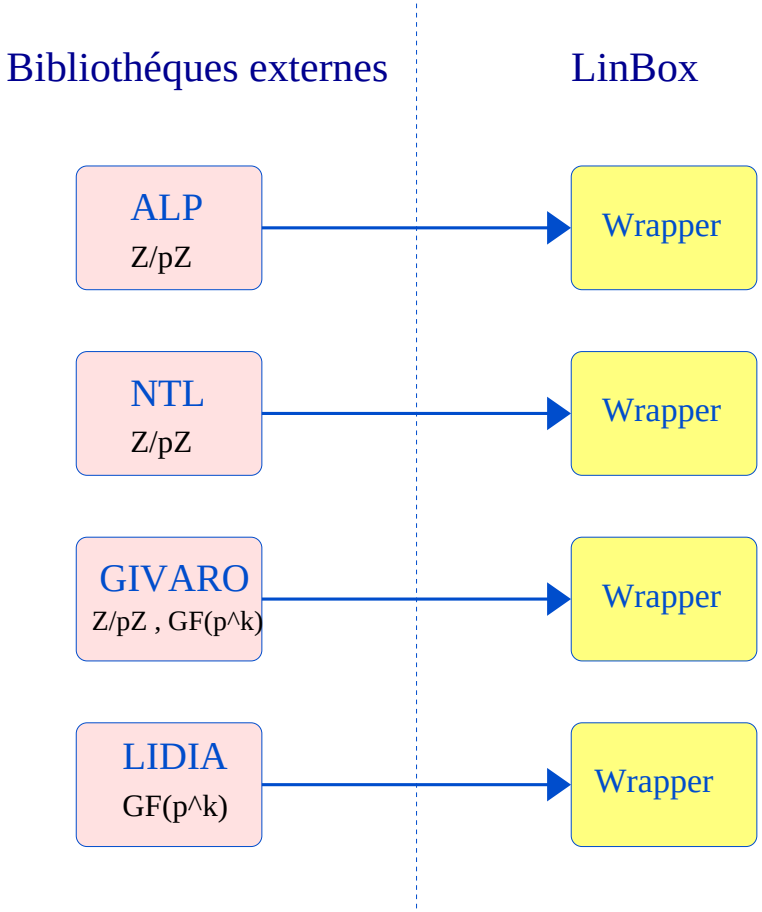
Wrappers LinBox

- Avantages :
  - Généricité des différents corps finis de LinBox.
  - Facilité d'utilisation grâce à des fonctions templates.
- Inconvénients :
  - Utilisation de plusieurs objets.
  - Taille du code compilé important.

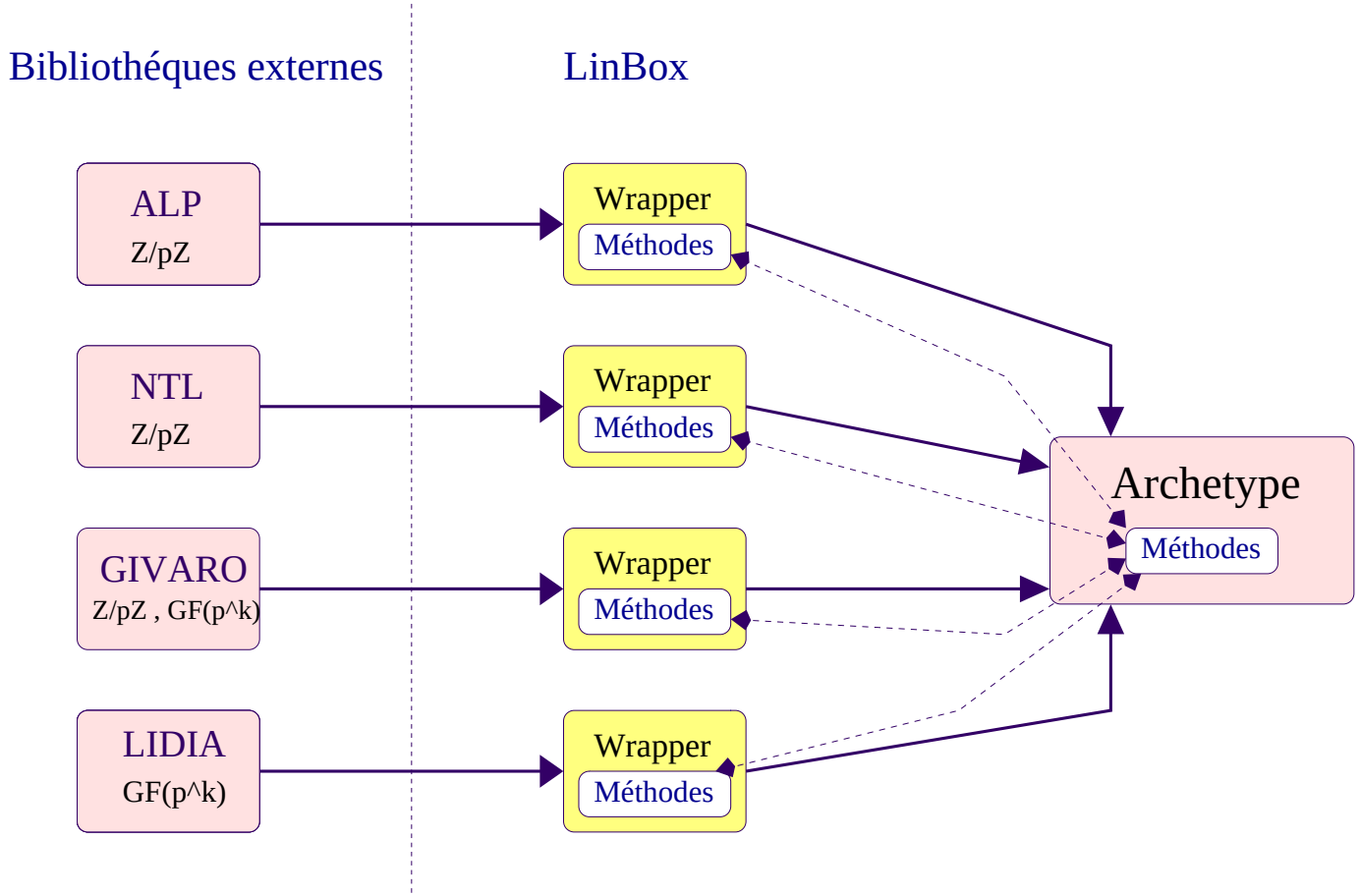
Interface standard

- Définition d'un corps finis virtuel : **L'Archetype**.
- L'Archetype définit l'interface.
- L'Archetype est un objet à part entière.
- Il définit l'ensemble de tous les domaines de corps finis possibles.

- *Les différents wrappers possèdent tous les mêmes méthodes (noms, paramètres et actions).*



- *L'archetype impose les méthodes des wrappers (noms, paramètres et actions).*



Exemple de code LinBox avec l'Archetype

```
 template<class Domain >
void Fct(Domain& field)
{   typedef typename Domain::element elt;
  elt a,b,r;                  // Déclaration de a,b,r comme éléments de "Domaine"

  field.init(a);
  field.init(b);                  // Initialisation de a,b,r sur le corps fini "field"
  field.init(r);

  field.read(cin,a);
  field.read(cin,b);

  field.mul(r,a,b);                  // r ← a * b dans le corps fini "field"

   field.write(cout,r);
}
```

```
void main()
{
  // Declaration of a Z/pZ field over Givaro library with p=53.
  givaro_zpz<Std16> K(53);

  // Declaration of a GF(p^k) field over Lidia library with p=53 and k= 5.
  lidia_gfq Q(53,5);

  Field_archetype Q_arch(& Q);          // Construction des archetypes des corps finis.
  Field_archetype K_arch(& K);

  Fct(Q_arch);
  Fct(K_arch);
}
```

Archetype LinBox

- Avantages :
  - Manipulation d'un seul type d'objet grâce à l'interface.
  - Validation de code sans connaître les wrappers.
- Inconvénients :
  - Temps d'exécution plus important.

II) Arithmétique des différents corps finis utilisés

Arithmétiques des corps finis

$\mathbb{Z}/\mathbf{p}\mathbb{Z}$ **vs** $\mathbf{GF(p^k)}$: Difficultés d'implémentation différentes.

Représentation des éléments

- Entiers machines ou grandes précisions, Flottants.
- Générateurs, Polynômes.

Opérateurs de bases

- Niveau éléments :
  - Modulo sur des entiers machines, modulo paresseux.
  - Utilisation des flottants, conversion avec entiers.
  - Lecture table (représentation logarithmique).
- Niveau BLAS 1 :
  Conversion (q_adic), Test d'overflow.
- Niveau $\mathbb{Z}/2\mathbb{Z}$:
  Mots machine, opérateurs logiques.

Stratégies des bibliothèques utilisées

- **ALP :**
  - $\mathbb{Z}/\mathbf{p}\mathbb{Z}$, Entiers machine (32 bits), modulo machines.

- **NTL :**
  - $\mathbb{Z}/\mathbf{p}\mathbb{Z}$, Entiers longs, utilisation flottant, modulo paresseux.

- **Givaro :**
  - $\mathbb{Z}/\mathbf{p}\mathbb{Z}$, Entiers machine (16,32 bits), modulo machines + modulo paresseux.

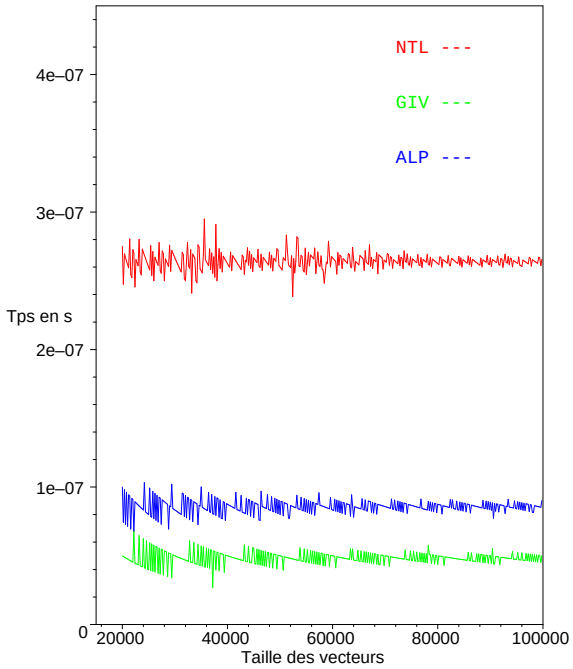
  - $\mathbf{GF}(\mathbf{p}^{\mathbf{k}})$, Générateurs, Entiers (machine,longs) ou flottants,
    Modulo paresseux + Lectures tables.

- **Lidia :**
  - $\mathbf{GF}(\mathbf{p}^{\mathbf{k}})$, Polynômes , Opérations sur polynômes.

III) Efficacité de ces arithmétiques et surcoût de LinBox.

Coût moyen d'une opération dans $\mathbb{Z}/\mathbf{p}\mathbb{Z}$

Test sur un produit scalaire de vecteurs (avec un p de 32 bits):

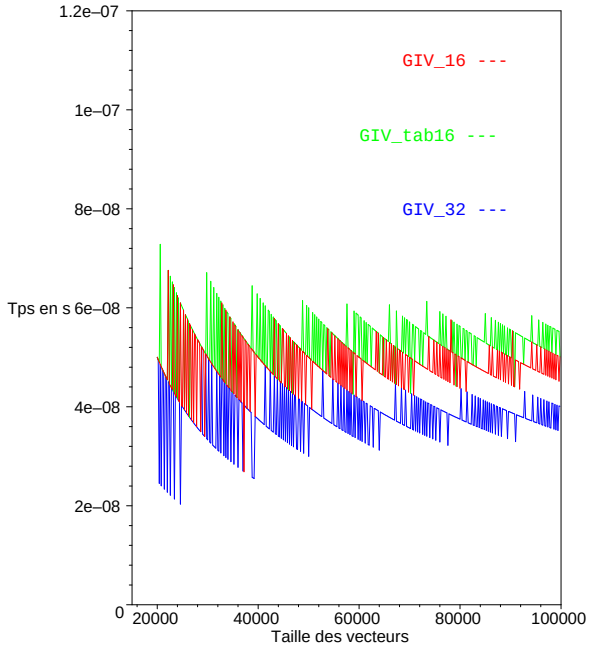


NTL

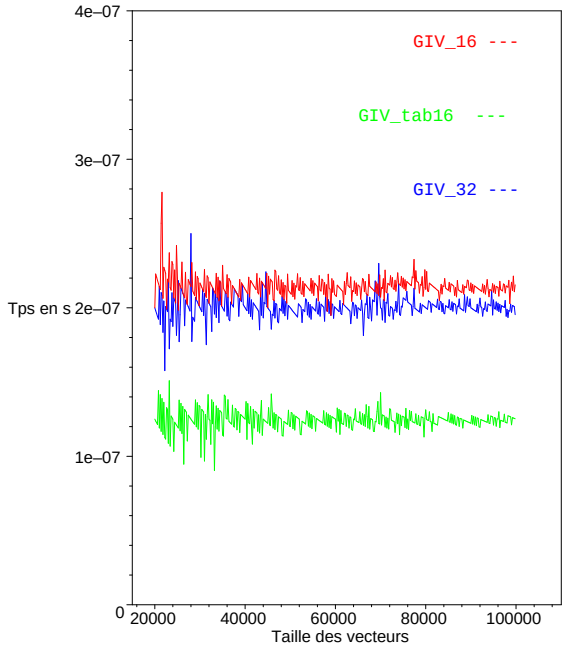
Givaro

ALP

L'efficacité d'une bibliothèque dépend de l'architecture processeur:



Intel



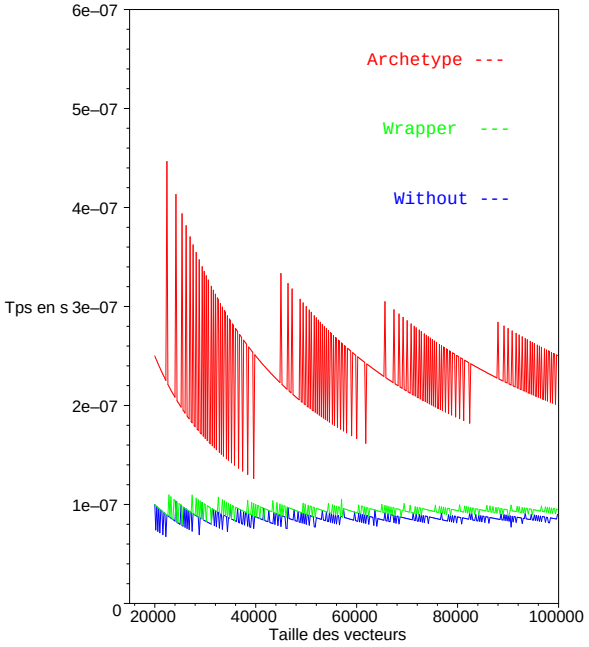
Sparc

Givaro 16 bits

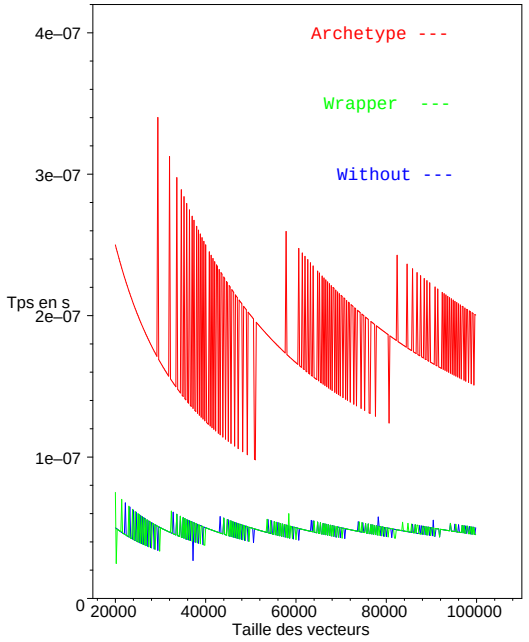
Givaro Log

Givaro 32 bits

Surcoût de LinBox



ALP



Givaro

Archetype

Wrapper

Directement

Conclusion

- Généricité des objets LinBox: Taille du code vs Efficacité.
- Arithmétiques des corps finis non optimisées BLAS 1 (en général).

- Vision de l'arithmétique des corps finis pour BLAS 1,2,3.
- Étude de nouvelle représentation (courbe elliptique).