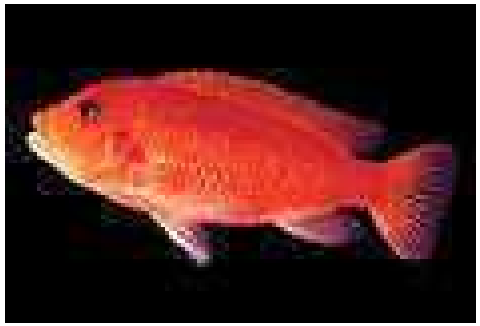
Calcul du polynôme caractéristique sur des corps finis

Clément Pernet

1er Avril 2004



Ecole Jeunes Chercheurs en Algorithmique et Calcul Formel

Cadre:

- Algèbre linéaire dense
- Calcul exact: corps finis

Démarche:

- Tirer parti des efforts pour le calcul numérique (architectures, BLAS)
- Utilisation d'algorithmes en blocs pour deux avantages:
  - Théorique: complexités en n^ω
  - Pratique: respectent la localité des données en mémoire

$\Rightarrow$ Mise en oeuvre pour le calcul du polynôme caractéristique

Partie 1

Routines efficaces pour l'algèbre linéaire exacte et dense

Principe

Les BLAS: Basic Linear Algebra Subroutines

- Pour le calcul numérique
- Conçus pour une architecture donnée. $\Rightarrow$ Haute efficacité pratique.
  ATLAS: BLAS générique

L'approche FFLAS: Finite Field Linear Algebra Subroutines

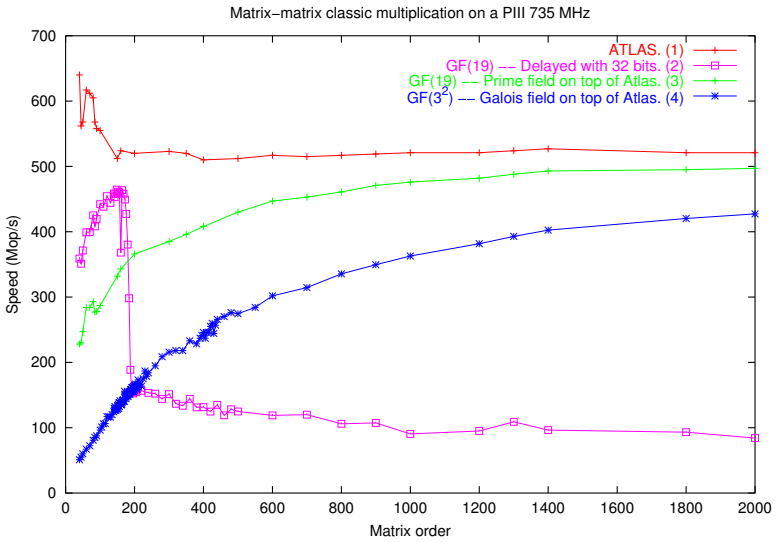
- Conversions: Corps fini (`int, long`) $\rightarrow$ réels (`double`)
- Calcul avec les BLAS
- Conversions du résultat: réels $\rightarrow$ Corps fini.

Contraintes: de validité et d'efficacité

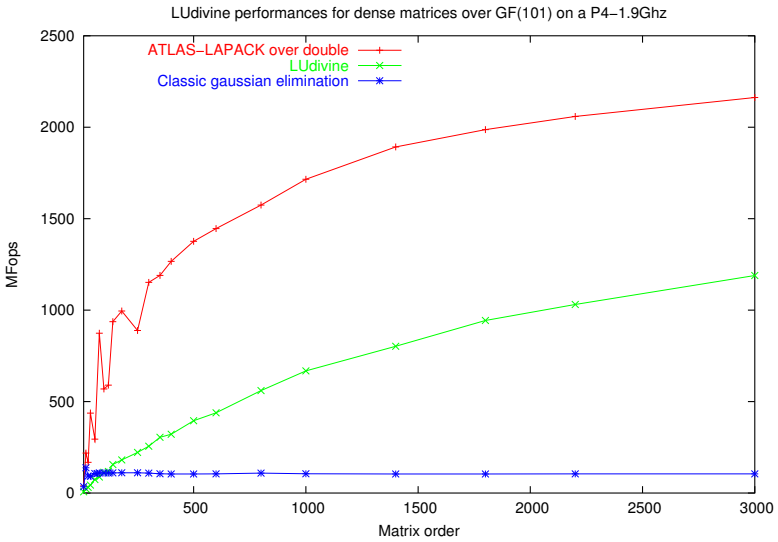
- Limitée aux opérations $+, -, \times$
- Limitée aux 53 bits de la mantisse des `double`
- Efficace si $\#conversions << \#operations$

Mise en oeuvre

Le produit matriciel:



La factorisation LSP:



- Jusqu'à 95% des performances des BLAS numériques pour le produit matriciel.
- Algorithme LSP par blocs $\Rightarrow$ usage des BLAS, $O(n^\omega)$
- Facteur de gain de 10 par rapport à une implémentation directe de l'elimination de Gauss.

Partie 2

Calcul du polynôme caractéristique

Le polynôme caractéristique

Applications:

- Etude des valeurs propres d'une matrice, et de leur multiplicité
- La forme normale de Frobenius, permet de calculer l'équivalence de 2 matrices

Quelques rappels:

- $P^A_{car}(x) = det(A - xId)$
- $P^{A,v}_{min} \mid P^A_{min} \mid P^A_{car}$
- Bloc compagnon $C_P = \begin{bmatrix} 0 & & & -m_0 \\ 1 & 0 & & -m_1 \\ & \ddots & \ddots & \vdots \\ & & 1 & -m_{k-1} \end{bmatrix}$, $P(x) = X^k + \sum_{i=0}^{k-1} m_i x^i$
- Forme normale de Frobenius: Diagonale de C_{P_i}, avec $P_0 = P^A_{min}$ et $P_{i+1} | P_i$

  $\Rightarrow$ Lien avec le polynôme caractéristique: $\prod_i P_i = P^A_{car}$

Approche de Krylov

- Soit X, les k premières colonnes de la matrice de Krylov $K(A,v) = \left[\begin{array}{c|c|c|c} v & Av & \dots & A^{n-1}v \end{array} \right]$ et $P_{min}^{A,v}(x) = x^k + \sum_{i=0}^{k-1} m_i x^i$

- Propriété: $AX = XC_{P_{min}^{A,v}}$ avec $C_{P_{min}^{A,v}} = \begin{bmatrix} 0 & & & -m_0 \\ 1 & 0 & & -m_1 \\ & \ddots & \ddots & \vdots \\ & & 1 & -m_{k-1} \end{bmatrix}$

$\Rightarrow$ Cas 1: $k = n$. Alors X est inversible, $P_{car}^{A}(x) = P_{min}^{A,v}(x)$, calculé par $C_{P_{min}^{A,v}} = X^{-1}AX$. (c'est un simple changement de base)

$\Rightarrow$ Cas 2: $k < n$. On complète la base X en $\overline{X}$, pour obtenir $\overline{X}^{-1}A\overline{X} = \begin{bmatrix} C_{P_{min}^{A,v}} & B \\ 0 & A_2 \end{bmatrix}$. On a alors $P_{car}^{A}(x) = P_{min}^{A,v}(x)P_{car}^{A_2}(x)$.

Calcul de $\overline{X}$

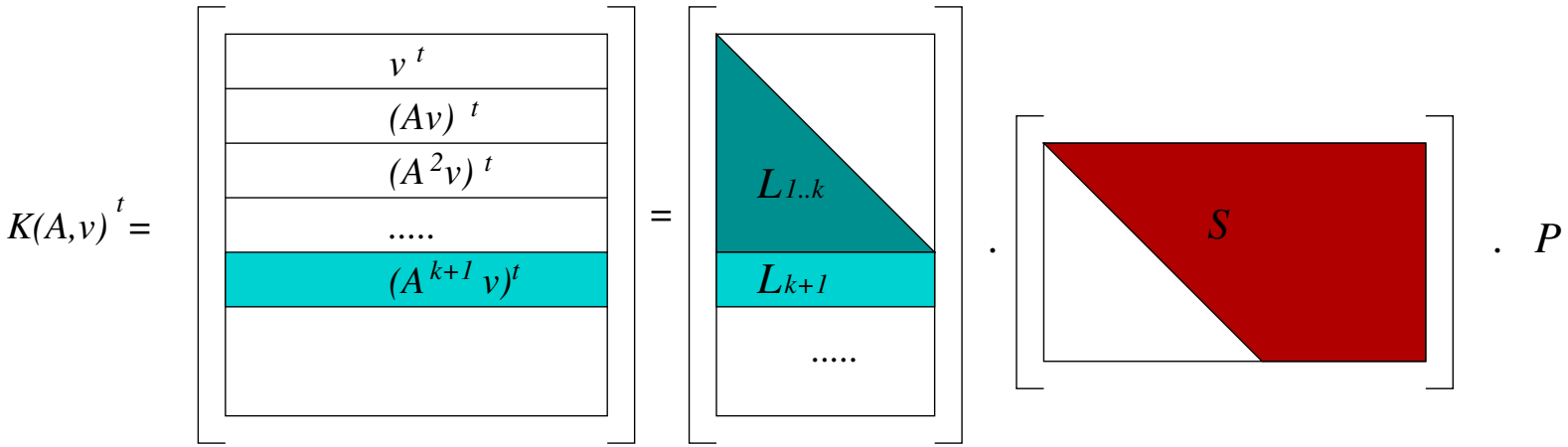
Principe: Pour compléter X en $\overline{X}$, ranger X en triangle (elimination), puis completer la diagonale par I_{n-k}

- On triangularise X^t en calculant $X^t = [L]\,[S_1|S_2]\,P$
- On complète la base: $\overline{L} = \begin{bmatrix} L & 0 \\ 0 & I_{n-k} \end{bmatrix}$. $\overline{S} = \begin{bmatrix} S_1 & S_2 \\ 0 & I_{n-k} \end{bmatrix}$,
- On prend alors $\overline{X} = (\overline{LS}P)^t$.
- Ainsi $\overline{X}^{-1} A \overline{X} = \begin{bmatrix} C_{P^{A,v}_{min}} & B \\ 0 & A_2 \end{bmatrix}$
- De plus $A_2 = A'_{22} - A'_{21} S_1^{-1} S_2$, avec $A' = P^t A P$

  $\Rightarrow$ 1 résolution de système triangulaire et 1 produit matriciel

Calcul de $Minpoly(A, v)$

- Au lieu de le calculer par $C_{P_{min}^{A,v}} = X^{-1}AX$, on peut utiliser la factorisation LSP de X^t



- $X_{k+1}^t = (A^k v)^t = \sum_{i=0}^{k-1} m_i (A^i v)^t = m \cdot X_{1\ldots k}$
- Ainsi L_{k+1} est combinaison linéaire des k premières lignes L_i
- Donc $m = L_{k+1}.L_{1\ldots k}^{-1}$

$\Rightarrow$ Complexité en $O(n^\omega log(n))$ en général, mais $O(n^2)$ dans ce cas, car X a déjà été factorisée.

L'algorithme `LU-Krylov` *

1. Tirer un vecteur aléatoire v
2. Calculer $X = K(A, v)^t$
3. Calculer $(L, S, P) = LSP(X)$
4. Calculer $m = L_{k+1}.L_{1\ldots k}^{-1}$, alors $M(x) = X^k - \sum_{i=0}^{k-1} m_i x^i$
5. Si ($deg(M) = n$) retourner $CharPoly(A) = M(x)$
6. Sinon
7. Calculer $A' = PA^tP^t$
8. retourner $Charpoly(A) = M(x)CharPoly(A'_{22} - A'_{21}S_1^{-1}S_2)$

* Collaboration avec Zhendong Wan (U. of Del.)

Caractéristiques de `LU-Krylov`

- Inspiré des idées de Krylov et Danilevski
- Etapes 2 et 3 de l'algorithme `LU-Krylov` peuvent se regrouper

  $\Rightarrow$ calcul des k premières lignes de $K(A, v)$ seulement
- Complexité $O(n^w \sum_{i=1}^{p} log(k_i) + n^2 \sum_{i=1}^{p} k_i^{\omega-2}) = O(n^3)$

  $\Rightarrow$ Pas d'amélioration théorique, mais introduction d'opérations par blocs
- Influence néfaste du nombre de facteurs invariants de la matrice (autant de factorisations LSP que de blocs)

L'idée de Keller-Gehrig

- But: Complexité en $n^\omega log(n)$

  $\Rightarrow$éviter le schéma itératif sur le nombre de blocs.

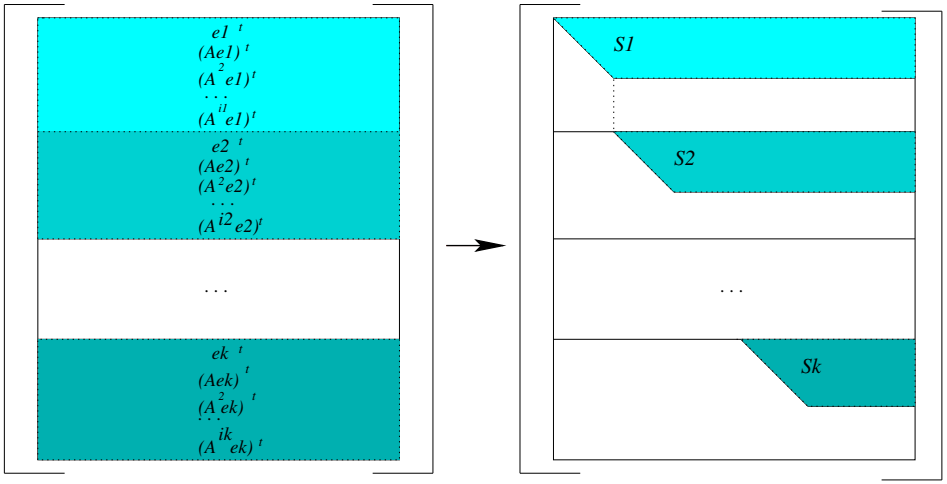
- Regroupe le calcul des itérés de Krylov de plusieurs vecteurs en un seul produit matriciel

- Complexité $O(n^w log(k_{max}))$ où k_{max} est la dimension du plus grand bloc compagnion. $\Rightarrow O(n^\omega log(n))$

- Influence faste du nombre de facteurs invariants de la matrice

- Calcule la forme de Hessenberg (ou polycyclique) de la matrice

Introduction de la factorisation LSP



Notre contribution: Besoin de déterminer le degré d'indépendance linéaire d'un bloc d'itérés

$\Rightarrow$ Keller-Gehrig utilise une élimination en escalier rapide

$\Rightarrow$ Nous l'avons remplacée par la factorisation LSP.

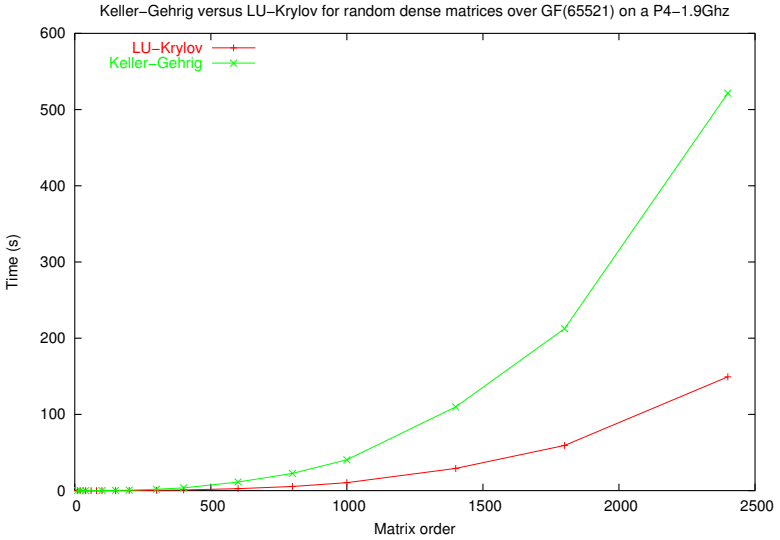
Avantages:

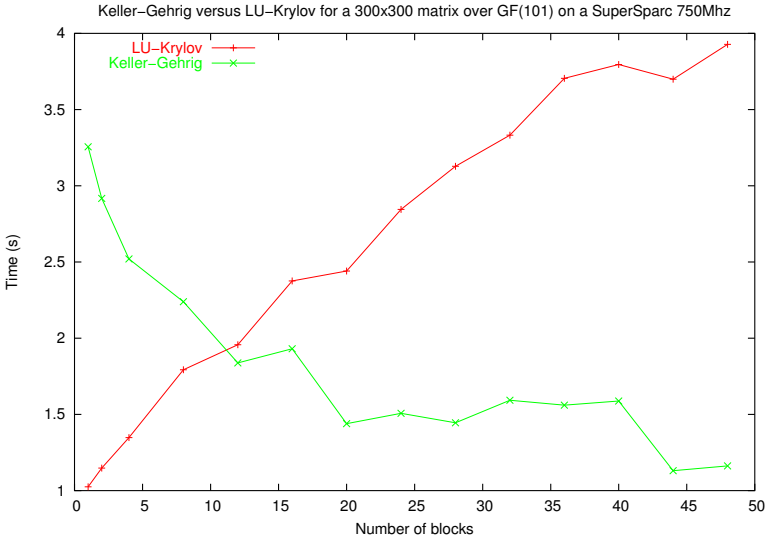
- Simplification de l'algorithme
- Permet de déduire directement les polynômes minimaux.

L'algorithme `LSP-Keller-Gehrig`

1. $U \leftarrow Id_n, V = B \leftarrow A^t, r_i \leftarrow 1 \forall i$
2. Tant que $\exists\, i_0, r_{i_0} = 2^l$
3. $X \leftarrow \begin{bmatrix} U_1 \\ \hline V_1 \\ \hline \dots \\ \hline U_k \\ \hline V_k \end{bmatrix}$ où U_i et V_i sont $r_i \times n$
4. Calculer $(L, S, P) = LSP(X)$
5. Mettre à jour $r_i\ \forall i$
6. Mettre à jour $U = (v_1, \dots, A^{r_1-1}v_1, \dots, v_i, \dots, A^{r_i-1}v_i, \dots)^t$
7. $B \leftarrow B^2, l \leftarrow 2l, V \leftarrow UB$
8. fin tant que
9. Pour chaque bloc, calculer $M_i(x)$ par l'algorithme `minpoly(A,v)`
10. Retourner $CharPoly(A) = \prod_i M_i(x)$

Résultats expérimentaux





- Gain supérieur à 100 par rapport à `Maple8` (meilleur des cas pour `Maple`) , et même au moins 5 pour `Magma2.9`
- Comportement complémentaire selon le nombre de blocs:

  `LU-Krylov` pour peu de blocs, de grande taille

  `LSP-Keller-Gehrig` pour beaucoup de petits blocs

  $\Rightarrow$ Approche hybride nécessaire

Conclusions

- Succès de l'approche FFLAS appliquée au calcul du polynôme caractéristique.
- Deux nouveaux algorithmes
  - Sans amélioration théorique de la complexité
  - Mais très efficaces en pratique
- Debouchant sur une implémentation hybride cumulant tous les avantages:
  - Complexité proche de l'optimal (n^3 ou $n^\omega log(n)$)
  - Des vitesses de loin supérieures à tous les programmes actuels
  - Des performances garanties, quelle que soit la matrice traitée
- Disponibles au sein de la bibliothèque `LinBox`

Perspectives

- Calcul de la forme de Frobenius à partir de la forme de Hessenberg:
  - différentes méthodes: Storjohann, Eberly, Giesbrecht
  - avec calcul de la matrice de transition
- Heuristiques pour le cas des matrices creuses:
  - Abandonnent les techniques d'élimination (coûteuses en mémoire)
  - Basées sur des perturbations de matrice et des calculs de polynômes minimaux.
  - Recherche combinatoire sur les multiplicités
  - Possibilité d'algorithmes hybrides dense-creux