# Towards an efficient implementation for the resolution of structured linear system

Benoît Lacelle – University of Western Ontario

ACA 2009 - Montréal

# Where are linear systems ?

- Set of equations: A.X=B
  - A is a matrix
  - X and B are vectors

- Example: Extended GCD of (p, q) in K[X]²

$$\begin{bmatrix} p_0 & & & q_0 & & \\ p_1 & \ddots & & q_1 & \ddots & \\ \vdots & \ddots & p_0 & \vdots & \ddots & q_0 \\ p_n & & p_1 & q_m & & q_1 \\ & \ddots & \vdots & & \ddots & \vdots \\ & & p_n & & & q_m \end{bmatrix} \times \begin{bmatrix} a_0 \\ \vdots \\ a_{m-1} \\ b_0 \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}$$

# The *displacement approach*

Matrices neither dense nor sparse

Kailath, Kung, Morf [1979]

*Displacement operator :* $\phi(.)$

*Displacement rank :* $\mathrm{rank}(\phi(.))$

- Low memory representation
- Fast elementary operations

# Example: Toeplitz matrices

- A is defined by its first row and first column

- *Displacement operator*: $\phi^{+}(A) = A - \searrow A$

$$\overset{A}{\begin{pmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ g & f & e & a \end{pmatrix}} - \overset{\searrow A}{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & a & b & c \\ 0 & e & a & b \\ 0 & f & e & a \end{pmatrix}} = \overset{\phi^{+}(A)}{\begin{pmatrix} a & b & c & d \\ e & 0 & 0 & 0 \\ f & 0 & 0 & 0 \\ g & 0 & 0 & 0 \end{pmatrix}}$$

- *Displacement rank*: $\mathrm{rank}(\phi^{+}(A)) \leq 2$
- Toeplitz-like: $\mathrm{rank}(\phi^{+}(A)) \ll \mathrm{Dim}(A)$

# Low memory consumption

- O(α.n) elements

$$\operatorname{rank}(\phi^+(A)) = \alpha \quad \Longrightarrow \quad \phi^+(A) = \sum_{j=1}^{\alpha} y_j z_j^{tr}$$

$$\Longrightarrow \quad A = \sum_{j=1}^{\alpha} L[y_j] U[z_j^{tr}]$$

where $L \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ b & a & 0 \\ c & b & a \end{bmatrix} \qquad U \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} a & b & c \\ 0 & a & b \\ 0 & 0 & a \end{bmatrix}$

# Fast Matrix-Vector Multiplication

- 2αM(n) operations $$A = \sum_{j=1}^{\alpha} L[y_j]U[z_j^{tr}]$$
  - M(n) is the cost of a polynomial product in degree n
  - FFT: M(n) = O(n log n loglog n)

$$\begin{bmatrix} a & & \\ b & a & \\ c & b & a \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} au \\ av + bu \\ aw + bv + cu \end{bmatrix}$$

$$(a + bX + cX^2) \times (u + vX + wX^2) \equiv \begin{bmatrix} au & \times X^0 \\ av + bu & \times X^1 \\ aw + bv + cu & \times X^2 \end{bmatrix} \bmod X^3$$

# Divide and Conquer à la Strassen

- *If A has generic rank profile (all north-west submatrices are invertible)*

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

- *Then $A_{1,1}$ is invertible and we have:*

$$A^{-1} = \begin{bmatrix} A_{1,1}^{-1} - A_{1,1}^{-1} A^{1,2} \Delta^{-1} A^{2,1} A_{1,1}^{-1} & -A_{1,1}^{-1} A^{1,2} \Delta^{-1} \\ -\Delta^{-1} A^{2,1} A_{1,1}^{-1} & \Delta^{-1} \end{bmatrix}$$

- With the Schur complement

$$\Delta = A_{2,2} - A_{2,1} A_{1,1}^{-1} A_{1,2}$$

# The MBA algorithm

- Morf [1980] and Bitmead-Anderson [1980]
  - Inversion in $O(\alpha^2 M(n) \log n)$
  - FFT: $O(\alpha^2 n (\log n)^2 \log\log n)$

  - In the dense case: $O(n^\omega)$ with $2 < \omega \leq 3$

- *Input*: A given by generators
- *Output*: $A^{-1}$ given by generators

1. Inverse $A_{1,1}$
2. Compute $\Delta$
3. Inverse $\Delta$
4. Recompose $A^{-1}$
5. Reduce the number of generators defining $A^{-1}$

# Our improvements

- Previous work
  - Kaltofen [1994]
    - Use of randomization
    - Remove the *generic rank profile* condition
- Our improvements
  - Divide the complexity by a constant
    - 4 to 8 times faster

# Where is the cost ?

Matrix-Matrix products:

- Rank($\phi^+(A)$) = $\alpha$ and Rank($\phi^+(B)$)= $\beta$
  - $(2\alpha\beta+2)\, M(n)$
- Return A.B defined by $\alpha+\beta+1$ generators

Reduction of generators:

- Rank($\phi^+(A)$) = $\alpha$ but defined by $\beta$ generators
  - $O(\alpha\, \beta\, n)$
- Return A defined by $\alpha$ generators

# Optimized Schur Inversion

- Reduce the number of multiplications
  - 10 to 6 matrix-matrix multiplications

$$\gamma = A_{1,1}^{-1} A_{1,2}$$

$$\Delta = A_{2,2} - A_{2,1}\gamma$$

$$A^{-1} = \begin{bmatrix} A_{1,1}^{-1} + \gamma B_{2,1} & -\gamma\Delta^{-1} \\ B_{2,1} = -\Delta^{-1} A_{2,1} A_{1,1}^{-1} & \Delta^{-1} \end{bmatrix}$$

- Supplementary reductions
  - Pay for the reduction
  - Save in following matrix-matrix products

# Hybrid algorithm

Quadratic inversion:

- Heinig, Rost [1984]
  - Inversion in $O(\alpha n^2)$ with the solution of $\alpha$ structured systems $A.X=B$ with the same A and different B

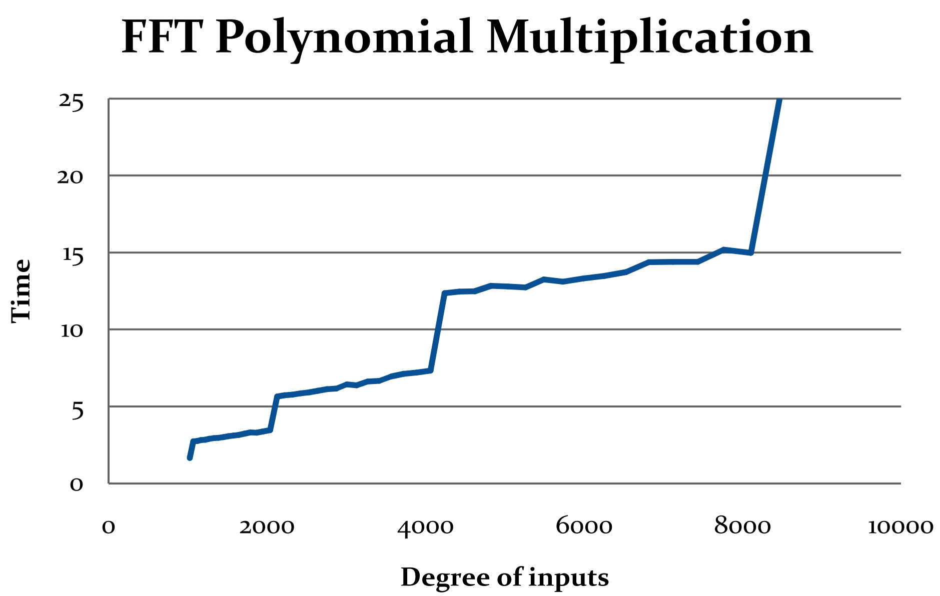
- Gohberg, Kailath, Koltracht [1986]
  - Resolution in $O(\alpha n^2)$ plus $O(n^2)$ per systems (same A, different B)

- Lead to inversion in $O(\alpha n^2)$
- MBA with FFT: $O(\alpha^2\ n\ (\log n)^2 \log\log n)$

# No Staircase Behaviour

<u>Fast Fourier Transformation</u>

- ~ constant complexity between powers of 2




- Recursive algorithm based on FFT
  - Shall have a staircase behavior? No!
  - Unbalanced subproblems

# Unbalanced subproblems

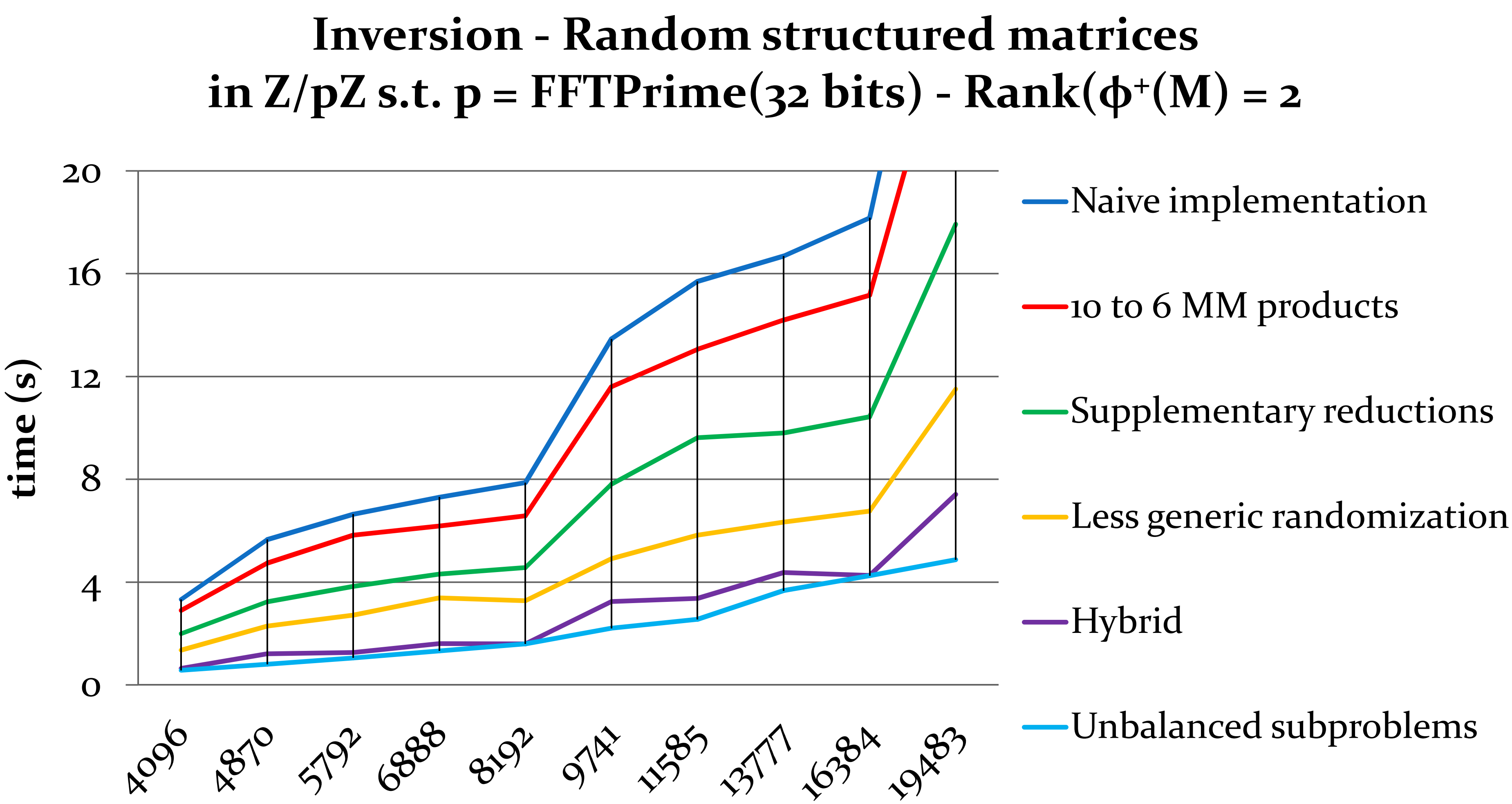
- Balanced policy

$$
\begin{aligned}
C(10) &= C(\lceil 10/2 \rceil) + C(\lfloor 10/2 \rfloor) + T(10) \\
C(10) &= 2C(5) + T(10) \\
C(10) &= 2C(2) + 2C(3) + T(10) + 2T(5) \\
C(10) &= 4C(2) + 2C(1) + T(10) + 2T(5) + 2T(3) \\
C(10) &= 10C(1) + T(10) + 2T(5) + 2T(3) + 4T(2) \\
C(10) &= 10C(1) + T(10) + 2T(8) + 2T(4) + 4T(2)
\end{aligned}
$$

- Unbalanced policy

$$
\begin{aligned}
C(10) &= C(2^{\lfloor log_2 10-1 \rfloor}) + C(10 - 2^{\lfloor log_2 10-1 \rfloor}) + T(10) \\
C(10) &= C(8) + C(2) + T(10) \\
C(10) &= 2C(4) + C(2) + T(10) + T(8) \\
C(10) &= 5C(2) + T(10) + T(8) + 2T(4) \\
C(10) &= 10C(1) + T(10) + T(8) + 2T(4) + 5T(2)
\end{aligned}
$$

# C++ implementation based on NTL



# Coming improvements

- Newton iteration adapted to structured matrices
  - Inversion of rational matrices

- C++
  - Generic Programming
  - Handle other structures