From BLAS routines to finite field exact linear algebra solutions

Pascal Giorgi

Laboratoire de l’Informatique du Parallélisme (Arenaire team)
ENS Lyon - CNRS - INRIA - UCBL
France
Main goals

Solve Linear Algebra problems exactly using BLAS routines

Implementation in LinBox library

- Focus on finite fields
- Use matrix multiplication and BLAS routines as kernel
- Fast exact linear algebra routines (triangular solver, LSP factorization)
Finite field computations via BLAS routines

Main idea [Dumas-Gautier-Pernet 02]

Convert data from finite field to double
Direct call to BLAS
Convert back the result

- Only one reduction

- Certify data hold over 53 bits

- Use division-free BLAS routines only
Illustration with matrix multiplication

- \((m \text{ by } k \text{ matrix}) \times (k \text{ by } n \text{ matrix}) \text{ over } \mathbb{GF}(p).\)

**Certificate:** \(k(p - 1)^2 < 2^{53}\)

- Performances with \(m = n = k\) and \(p = 19\):

![Matrix multiplication performance graph]

- Even better with Strassen-Winograd algorithm [Dumas-Gautier-Pernet 02]

P. Giorgi
Our extension to a triangular solver with matrix r.h.s.

Problem:
- Certificate is $p^k < 2^{53}$ instead of $kp^2 < 2^{53}$ for matrix multiplication
  Direct call to BLAS only is too restrictive (only small matrices)

Solution:
- Use a block recursive algorithm
- Decrease matrix size until certification
- Use BLAS-based matrix multiplication routine to reconstruct the solution
Block recursive algorithm

- Solve \( AX = B \) over \( \mathbb{GF}(p) \)

- While no certification

\[
\begin{align*}
\begin{array}{c}
A_1 & A_2 & \times & X_1 & = & B_1 \\
A_2 & A_3 & & X_2 & & B_2
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{cases}
\text{solve } & A_3X_2 = B_2 & \text{recursive call} \\
& B_1 \leftarrow B_1 - A_2X_2 & \text{BLAS-based MM} \\
\text{solve } & A_1X_1 = B_1 & \text{recursive call}
\end{cases}
\end{align*}
\]

- Now, how to solve small (certified) linear systems?
Solving a certified triangular linear system

- Certified as soon as \((p - 1)p^{m-1} < 2^{53}\) (\(m = \) row dimension of small system \(AX = B\))

- Let \(A = UD\) over \(GF(p)\)
  
  with \(D\) a diagonal matrix and \(U\) a unit upper triangular matrix

  Solve \(UY = B\) using the \(dtrsm\) BLAS routine

- Return \(X = D^{-1}Y\) over \(GF(p)\)
Performances over $\text{GF}(19)$ on Intel Itanium

Field TRSM over $\text{GF}(19)$ on a IA64-750Mhz

- LinBox FTRSM using GivZpz
- LinBox FTRSM using dble
- BLAS DTRSM through ATLAS
- LinBox FTRSM using GivZpz without BLAS

Matrix size $n=m$

Mops

P.Giorgi
In summary, we have just seen

- BLAS-based matrix multiplication
- BLAS-based triangular solver with matrix r.h.s.

Now, let us see

- LSP factorization
What is LSP factorization?

- **LSP** matrix factorization [Bini-Pan 94]
  
  \(L\) - lower triangular matrix with 1’s on the main diagonal
  
  \(S\) - reduces to an u.t. matrix with nonzero diagonal entries when zero rows deleted
  
  \(P\) - permutation matrix

- Exemple over \(GF(7)\):

\[
\begin{bmatrix}
1 & 3 & 5 & 2 & 4 & 6 \\
3 & 2 & 1 & 6 & 5 & 4 \\
3 & 3 & 6 & 0 & 1 & 2 \\
5 & 3 & 3 & 6 & 0 & 1
\end{bmatrix}
= \begin{bmatrix}
1 \\
3 \\
1 \\
1
\end{bmatrix}
\times \begin{bmatrix}
3 & 1 & 5 & 6 & 2 & 4 \\
2 & 1 & 3 & 5 & 4 \\
2 & 1 & 5 & 3
\end{bmatrix}
\times \begin{bmatrix}
1 \\
1 \\
1 \\
1
\end{bmatrix}
\]
LSP factorization via matrix multiplication

- Recursive algorithm [Ibarra-Moran-Hui 82]:
  - Partition \( A = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \) and factor \( A_1 = L_1 S_1 P_1 \)
  - Partition \( S_1 = [S'_1 \ B] \) and \( A_2 P_1^{-1} = [C \ D] \)
  - Solve \( G S'_1 = C \) and factor \( D - G B = L_2 S_2 P_2 \)
  - Reconstruct with formula:

\[
A = \begin{bmatrix}
L_1 \\
G \\
L_2
\end{bmatrix} \times \begin{bmatrix}
S'_1 \\
B P_2^{-1} \\
S_2
\end{bmatrix} \times \begin{bmatrix}
I \\
P_2 \\
\times P_1
\end{bmatrix}
\]
Our implementation

- Solve $\mathbf{G} \mathbf{S}_1' = \mathbf{C}$ by using BLAS-based triangular solver
- Compute $\mathbf{D} - \mathbf{G} \mathbf{B}$ by using BLAS-based matrix multiplication
- Performances over $\mathbb{GF}(19)$ on Intel Itanium:

```
LSP factorization over GF(19), IA64-750Mhz
LinBox LSP using GivZpz
LinBox LSP using double Zpz
LinBox LSP using GivZpz without BLAS
```

P.Giorgi
In summary, we have just seen

- BLAS-based matrix multiplication
- BLAS-based triangular solver with matrix r.h.s.
- BLAS-based LSP factorization

Now, let us see

- Availability in LinBox and application to minimal matrix polynomial
Integration in LinBox library

- **FFLAS package** [Dumas-Gautier-Pernet 02]

- **Fast BLAS-based triangular solver (BLAS-like interface)**
  
  ```cpp
template < class Field >
void Field_trsm( const Field& F,
                int m, int n,
                typename Field::Element * B, int ldb,
                typename Field::Element * A, int lda,
                typename Field::Element * X, int ldx,
                Triangular tr,
                Unitary un,
                Side si);
```

- **Fast BLAS-based LSP factorization**

- **Genericity over the domain** $\text{GF}(p)$ **and dense matrices (with BLAS-like storage)**
Features of our LinBox implementation routines

- Easier to implement higher level algorithms based on matrix multiplication
- Speeding up matrix multiplication $\Rightarrow$ faster routines

- Example: computation of minimal matrix polynomial
Minimal matrix polynomial $\Pi_A(x)$ of a square matrix $A$

- Important algorithm in LinBox (Krylov/Lanczos approach)

Two main steps of the algorithm:

- Compute the first terms of $UV, UAV, UA^2V, ...$

- Deduce $\Pi_A(x)$ by computing a matrix approximant of

$$\begin{bmatrix}
\sum_i (UA^iV)x^i & 0 \\
0 & I
\end{bmatrix}$$

[Turner’s PhD Thesis 02]
Matrix approximant algorithm

• Beckermann-Labahn’s algorithm via matrix multiplication [Giorgi-Jeannerod-Villard 03]

• Iterative algorithm which computes approximant $M(x)$ s.t.

$$M(x)F(x) = O(x^\sigma) \text{ in } \sigma \text{ steps}$$

• Main operations involved at step $k$
  - $k$ calls to matrix multiplication
  - 1 call to LSP factorization
  - $k$ calls to triangular system solving

• We have used our LinBox BLAS-based routines to implement this algorithm
First performances over $\text{GF}(19)$ on Intel Itanium
Conclusion and future work

• Significant improvement for some linear algebra problems over $\mathbf{GF}(p)$

• Implementation in LinBox library [www.linalg.org]

• Extension to sparse matrices

• Extension to other algorithms using matrix multiplication