Note Taker Checklist Form -*MSRI*

Name : Rob Stapleton

E-mail Address/ Phone #: jrstaple@ncsu.edu

Talk Title and Workshop assigned to:
Interactive Parallel Computation in Support of Research in Algebra, Geometry, and Number Theory

Lecturer (Full name): Clément Pernet

Date & Time of Event: 9:00 a.m. Jan 29, 2007

Check List:

(✓) Introduce yourself to the lecturer prior to lecture. Tell them that you will be the note taker, and that you will need to make copies of their own notes, if any.

(✓) Obtain all presentation materials from lecturer (i.e. Power Point files, etc). This can be done either before the lecture is to begin or after the lecture; please make arrangements with the lecturer as to when you can do this.

(✓) Take down all notes from media provided (blackboard, overhead, etc.)

(✓) Gather all other lecture materials (i.e. Handouts, etc.)

(✓) Scan all materials on PDF scanner in 2nd floor lab (assistance can be provided by Computing Staff) – Scan this sheet first, then materials. In the subject heading, enter the name of the speaker and date of their talk.

Please do **NOT** use **pencil** or colored pens other than black when taking notes as the scanner has a difficult time scanning pencil and other colors.

Please fill in the following after the lecture is done:

1. List 6-12 lecture keywords: parallel, LinBox, Linear Algebra, blackbox, block algorithms, Wiedemann, triangular systems

2. Please summarize the lecture in 5 or less sentences.
LinBox is generic middleware developed for efficient implementation of Linear Algebra in a variety of situations, with many tasks currently and soon to be implemented in a parallel manner.

Once the materials on check list above are gathered, please scan ALL materials and send to the Computing Department. Return this form to Larry Patague, Head of Computing (rm 214)

Talk: Parallel Perspectives for the LinBox Library
Clement Pernet

Exact Linear Algebra:
Building block in exact computation. Topology: Smith form. Graph Theory: Characteristic Polynomial. Rep. Theory: Null space
Cryptography: sparse system resolution.
The matricies involved can get very very large. etc.

Software Libraries for Exact Computations:
finite fields: NTL, GMP, Lidia, Pari, ...
polynomials: NTL, ...
integers: GMP, ...

Global Solutions:
Maple
Magma
Sage

Linear Algebra? It falls somewhere in between. In global solutions, it's not always as efficient as it could be. This is where LinBox tries to intervene, linking the global solutions with the libraries.

LinBox is genereic middleware.

Maple, GAP, Sage ----> LinBox ---> Finite Fields (NTL, Givaro, ...) , BLAS (ATLAS, GOTO, ...), GMP

Joint NSF-NSERC-CNRS project.
- -U Delaware, NC State
- -U Waterloo, U Calgary
- -Laboratoires, ... (missed)

Solutions: rank, det, minpoly, charpoly, system solve, positive definiteness over domains: finite fields, ZZ, QQ for matricies: sparse, structured; dense

A design for genericity:
Field/Ring interface:
- -Shared interace with Givaro
- -Wraps NTL, Lidia, Givaro implementations using archetype or envelopes
- -Proper implementations suited for dense computations (mainly rely on FLOp arithmetic) Matrix interface
- -Missed

Structure of the Library:
Solutions (det, rank) - spcify the method, domain -> Algorithms -damnit

Several levels of use:
- Web Servers: http://www.linalg.org/
- Executables:$ charpoly MyMatrix 65521
- Call to a soltuion:

```
NTL::ZZp F(65521);
Toeplitz<NTL::ZZp> A(F);
Polynomial<NTL::ZZp> P;
charpoly (P, A);
```

- Calls to specific algorithms

Dense computations:

Bilding block: matrix multiplication over word-size finite field
Principle:
- Delayed modular reduction
- Floating Point arithmetic (fused-mac, SSE2, ...)
- BLAS cache management.
- Sub-cubic algorithm (Winograd)

Design of other dense routines:
- Reduction to matrix multiplication
- Bounds for delayed modular operations
  - Block algorithm with multiple cascade.

Char Poly:
Fact: O(n^omega) Las Vegas probabilistic algorithm for the computation of the char poly over a Field.
This new algorithm is also practical. Virtually always beats the LU-Krylov for n>100

BlackBox Compuations:
Goal: computation with very large sparse or structured matricies.
- No explicit rep of matrix
- Only operation: application of a vector
- Efficient algorithms
- Efficient preconditioners: Toeplitz, Hankel, Butterfly, ...
...

Block Projection Algorithms:
- Wiedemann algorithm: scalar projection sof A^i for i=1..2d
- Block Wiedemann: kxk dense projections of A^i for i=1..2d/k
  - balance between blackbox and dense applications

Data Containers/Iterators:
Distinction etween computation and access to the data:
Example: Iterates (u^TA^iv)_{i=1..k} used for system resolution can be
- Precomputed and stored
- computed on the fly
- computed in parallel
Solution: Solver defined using generic iterators, independantly from the method to compute the data.

Existing ocntainers.iterators:
Scalar projections: (v^TA^iu)_{i=1..k} --> Wiedemann's algorithm
Block projections: (AV_i)_{i=1..k} --> Block Wiedemann
Modular homomorphic imaging: (Algortihm(A mod p_i))_{i=1..k} --> Chinese Remainder Algorithm
No modification of high-level algorithms for paralleliztion

Parallel tools:
Until now, fer paralleliztaions:
- Attempts with MPI and POSIX threads
- Higher level systems: Athapascan-1, KAAPI
  - Full design compatibility
  - missed

Example: rank computations:
[Dumas & urbanska]
- Parallel block Wiedemann algorithm: [u_1,...,u_k]^T(GG^T)u_i, i=1..k
  - Only rank(g)/k iterations

-Combined with sigma basis algorithm
Matrix: GL7d17, n=1,548,650 m=955,128 rank=626910
Time estimation: T_{iter} 0.46875 min. T_{seq} 621.8 days. T_{par}(50) 12.4 days. T_{par}(50,ET) 8.16 days

TURBO triangular elimination:
[Roch and Dumas 02]: recursive block algorithm for triangularization
- divide both rows and columns
  - better memory management
  - Enables to use recursive data structures
- 5 recursive calls (U,V,C,D,Z), including 2 being paralle (C,D)

Principle of Workstealing
[Arora, Blumofe, Plaxton01], [Acar, Belloche, Blumofe02]
-2 algorithms to complete a task f: f_seq and f_par
-When a processor becomes idle, ExtractPar steals the work to f_seq

Application to multiple triangular system solving:
TRSM : Compute <<U_1,0>|<<U_2,U_3>>^(-1) <<B_1,B_2>> x_2 = TRSM(U_3,B_2), B_1 = B_1 - U_2X_2, X_1 = TRSM(U_1,B_1)
f_seq: TRSM(U,B) --> T_1 = n^3, t_{infinity} = O(n)
f_par: Compute V = U^(-1); TRMM(V,B); --> t_1 = 4/3 n^3. T_{infinity} = O(log(n))

When sequential TRSM and parallel Inverse join: Computer X_1 = A_1^(-1) B_1 in parallel (TRMM)
BOX(Top down inverse going down) (Bottom-up TRSM coming up)

Multi-adic lifting: Solving Ax = b over ZZ
Standard p-adic Lifting [Dixon 82]

```
Compute A^(-1) mod p
r=b
for i=0..n do
      x_i = A^(-1)r mod p
      r = (r-Ax_i)/p
end for
z = x_0 + px_1 + ... + x_np^n
x = RatReconst(z)
end
```

Multi-adic lifting:

```
for all j=1..k do
      compute A^(-1) mod p_j
      r=b
      for i=0..n/k do
            x_i = A^(-1)r mod p_i
            r = (r-Ax_i)/p_j
      end for
      z_j = x_0 + p_jx_1 + ... + p^{n/k}_jx_{n/k}
end for
Z = ChineseRemainderAlg((z_j,p^{n/k}_j)_{j=1..k})
X= RatReconst(Z)
end
```

Complexity of this algorithm is worse, but can be made faster in practice (??)
-Used for sequential computation [Chen and Storjohann 05], to balance fficiency between BLAS levels 02 and 03 (?)

Conclusion:

Large Grain parallelism:
- Chinese Remaindering
- Multi-adic lifting
- Block Wiedermann

Fine Grain Adaptive Parallelism:
- Work Stealing

Perspectives:
- Development of simple paralle containers
- Parallel distribution of LinBox, based on Kaapi.

LinBox does not use "Greasing" techniques over finite fields
Multiple organizations worry about standards, esp. concerning matrix multiplication over small prime fields. There will be more talk about this later in the week.

The problem comes in that there are many different arithmetics to choose from.

Parallel Perspectives for the LinBox library

Clément PERNET

Symbolic Computation Group
University of Waterloo

January 29, 2007

Exact linear algebra

Building block in exact computation:

Cryptography : sparse system resolution

Representation theory : null space

Topology : Smith form

Graph theory : characteristic polynomial

...

Software solutions for exact computations

Libraries

finite fields: NTL, GMP, Lidia, Pari, ...

polynomials: NTL, ...

integers: GMP, ...

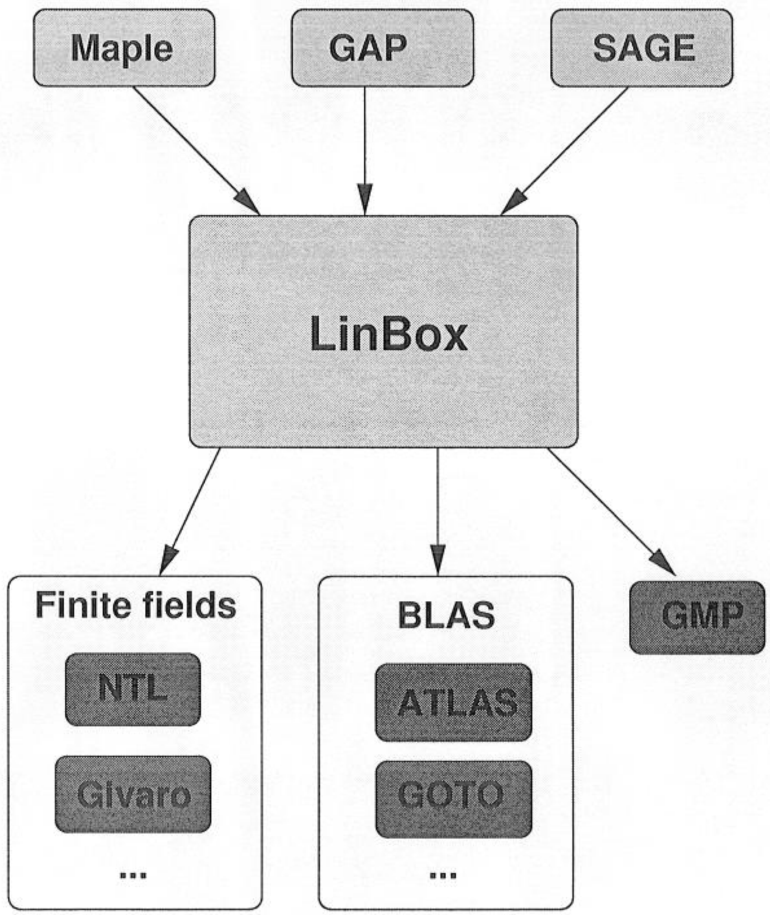
Global solutions

- Maple
- Magma
- Sage

Linear Algebra ?

Outline

A generic middleware



ˉhe LinBox project, facts

Joint NFS-NSERC-CNRS project.

- U. of Delaware, North Carolina State U.
- U. of Waterloo, U. of Calgary,
- Laboratoires LJK, ID (Grenoble), LIP (Lyon)

A LGPL source library:

- 122 000 lines of C++ code
- 5-10 active developpers

.inBox-1.0

;olutions

- rank
- det
- minpoly
- charpoly
- system solve
- positive definiteness

Domains of computation

- Finite fields
- $\mathbb{Z}, \mathbb{Q}$

Matrices

- Sparse, structured
- Dense

\ design for genericity

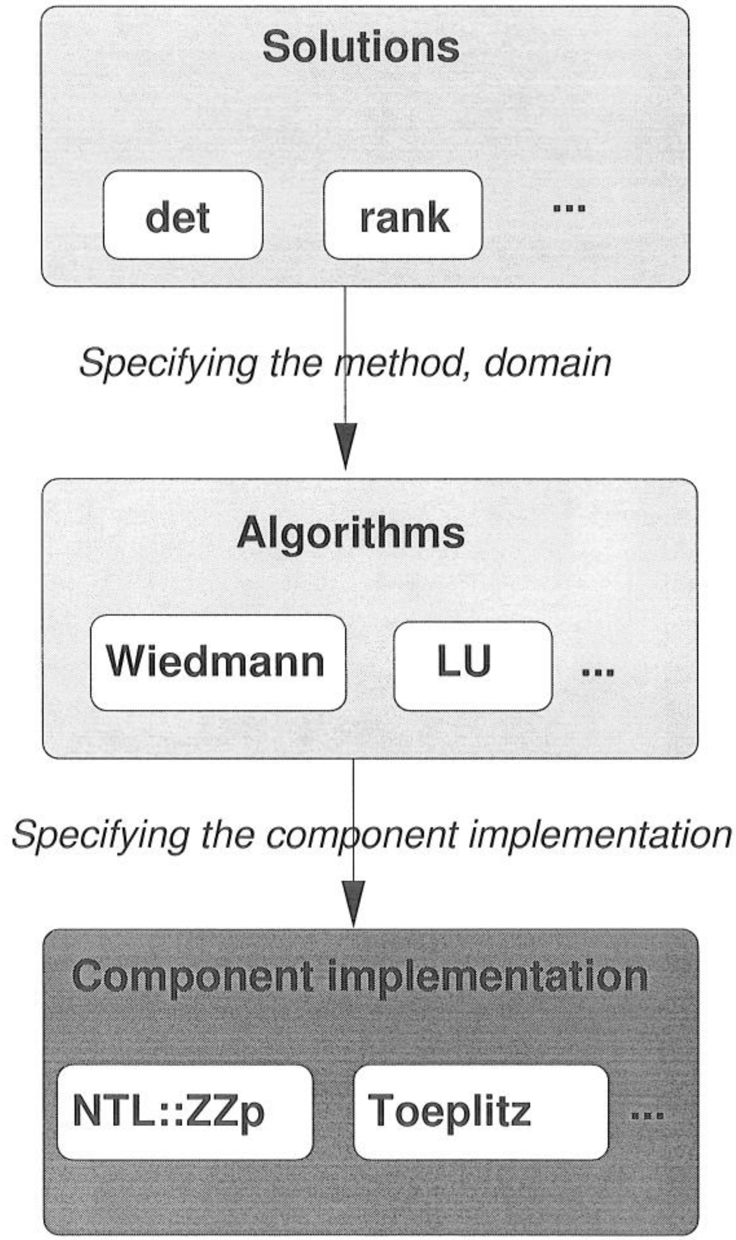
Field/Ring interface

- Shared interface with Givaro
- Wraps NTL, Lidia, Givaro implementations, using archetype or envelopes
- Proper implementations, suited for dense computations

Matrix interface

- Sparse, Dense: BlackBox `apply`
- Dense matrix interface: several levels of abstraction

Structure of the library



Several levels of use

- Web servers: `http://www.linalg.org`
- Executables: `$ charpoly MyMatrix 65521`
- Call to a solution:

```
NTL::ZZp F(65521);
Toeplitz<NTL::ZZp> A(F);
Polynomial<NTL::ZZp> P;
charpoly (P, A);
```

- Calls to specific algorithms

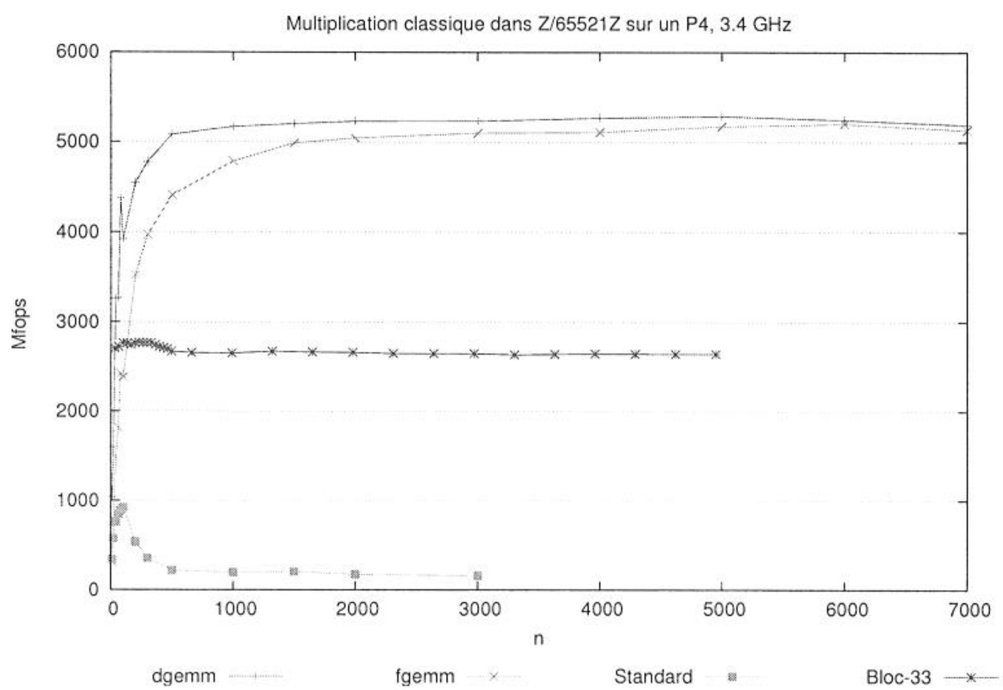
)ense computations

Building block:

matrix multiplication over word-size finite field

Principle:

- Delayed modular reduction
- Floating point arithmetic (fused-mac, SSE2, ...)
- BLAS cache management



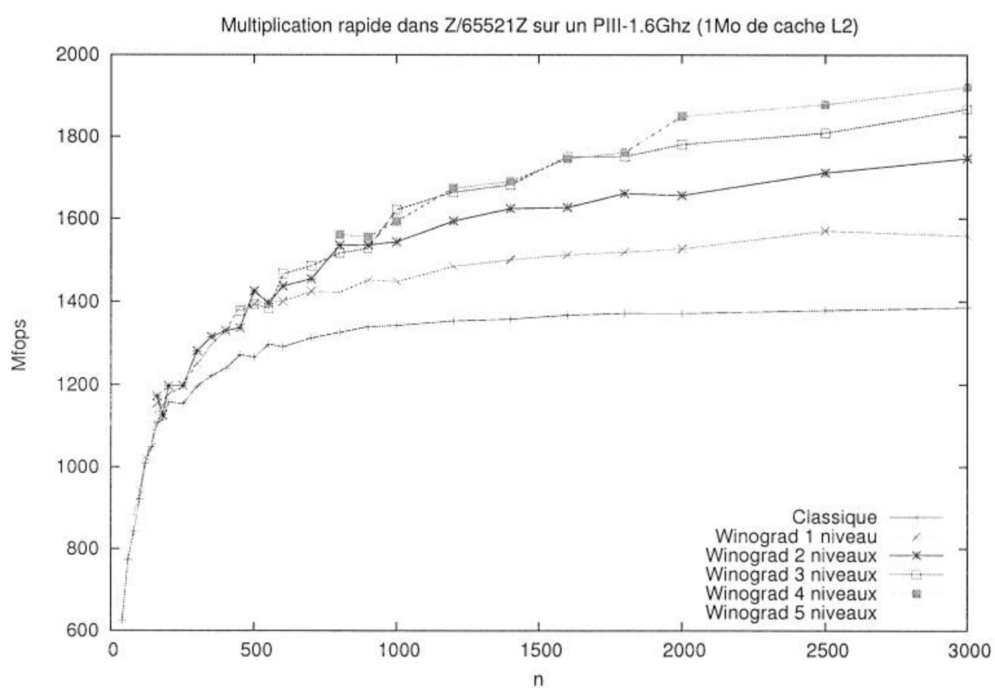
)ense computations

Building block:

matrix multiplication over word-size finite field

Principle:

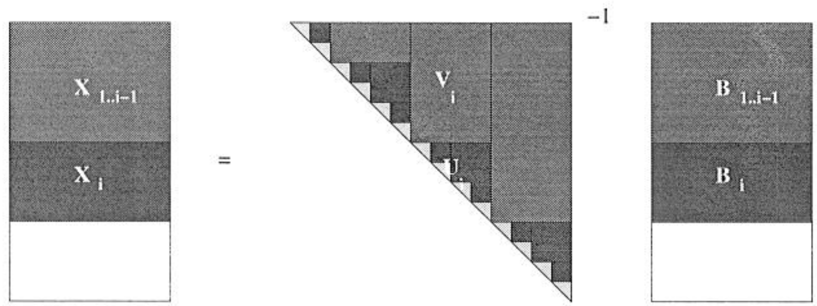
- Delayed modular reduction
- Floating point arithmetic (fused-mac, SSE2, ...)
- BLAS cache management
- Sub-cubic algorithm (Winograd)



)esign of other dense routines

- Reduction to matrix multiplication
- Bounds for delayed modular operations.

⇒Block algorithm with multiple cascade



	n	*1000*	*2000*	*3000*	*5000*	*10 000*
TRSM	$\frac{ftrsm}{dtrsm}$	1,66	1,33	1,24	1,12	1,01
LQUP	$\frac{lqup}{dgetrf}$	2,00	1,56	1,43	1,18	1,07
INVERSE	$\frac{inverse}{dgetrf+dgetri}$	1.62	1.32	1.15	0.86	0.76

;haracteristic polynomial

Fact

$\mathcal{O}(n^{\omega})$ *Las Vegas probabilistic algorithm for the computation of the characteristic polynomial over a Field.*

Practical algorithm :

n	magma-2.11	LU-Krylov	New algorithm
100	0.010s	0.005s	0.006s
300	0.830s	0.294s	0.105s
500	3.810s	1.316s	0.387s
1000	29.96s	10.21s	2.755s
3000	802.0s	258.4s	61.09s
5000	3793s	1177s	273.4s
7500	MT	4209s	991.4s
10 000	MT	8847s	2080s

Computation time for 1 Frobenius block matrices, on a Athlon 2200, 1.8Ghz, 2Gb

MT: Memory thrashing

BlackBox computations

Goal: computation with very large sparse or structured matrices.

- No explicit representation of the matrix,
- Only operation: application of a vector
- Efficient algorithms
- Efficient preconditionners: Toeplitz, Hankel, Butterfly, ...

3lock projection algorithms

- Wiedemann algorithm: scalar projections of A^i for $i = 1..2d$
- Block Wiedemann: $k \times k$ dense projections of A^i for $i = 1..2d/k$

$\Rightarrow$Balance efficiency between BlackBox and dense compations

)ata Containers/Iterators

Distinction between computation and access to the data:

Example

Iterates $(u^T A^i v)_{i=1..k}$ *used for system resolution can be*

- *precomputed and stored*
- *computed on the fly*
- *computed in parallel*

Solution: solver defined using generic iterators, independently from the method to compute the data

Example: A parallel data flow iterator

```
const iterator& iterator::operator++(){
    if (++current>launched) {
        ...
        for (int i=0; i<n; ++i)
            Fork<launch>(i,...);
        launched += n;
        }
    return *this;
}
const value_type& iterator::operator*(){
    return _d[current].read();
}
```

:xisting containers/iterators

- Scalar projections: $(v^T A^i u)_{i=1..k}$

  $\Rightarrow$Wiedemann's algorithm
- Block projections: $(Av_i)_{i=1..k}$

  $\Rightarrow$Block Wiedemann algorithm
- Modular homomorphic imaging: $(\texttt{Algorithm}(A \mod p_i))_{i=1..k}$

  $\Rightarrow$Chinese Remainder Algorithm

$\Rightarrow$no modifications to the high level algorithms for the parallelization.

ᴾarallelization tools

Until now, few parallelization:

- attempts with MPI, and POSIX threads
- Higher level systems: Athapascan-1, KAAPI
  $\Rightarrow$Full design compatibility
  $\Rightarrow$Provides efficient schedulers; work stealing abilities

Example: rank computations

[Dumas & urbanska]

- parallel block Wiedemann algorithm:
  $[u_1, .., u_k]^T (GG^T) u_i, i = 1..k$
  $\Rightarrow$Only $\frac{rank(G)}{k}$ iterations
- combined with sigma basis algorithm.

matrix	n	m	rank
GL7d17	1,548,650	955,128	626,910
GL7d20	1,437,547	1,911,130	877,562
GL7d21	822,922	1,437,547	559,985

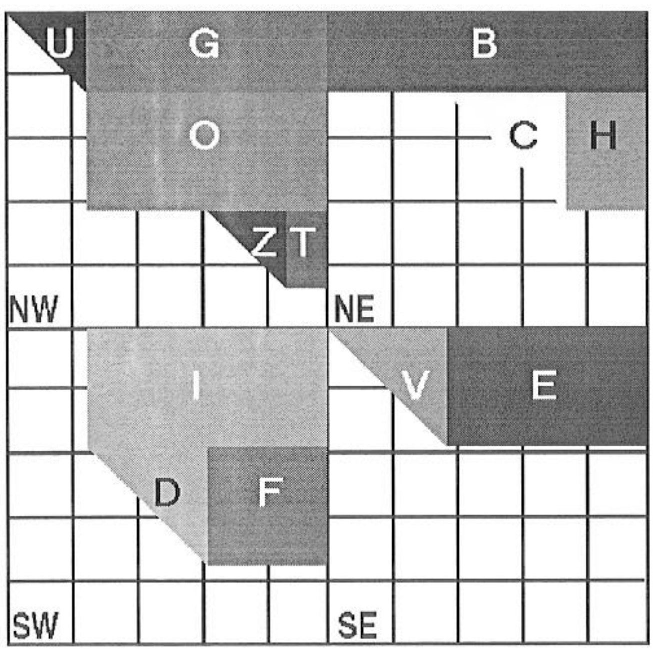
Timings estimations [in days]

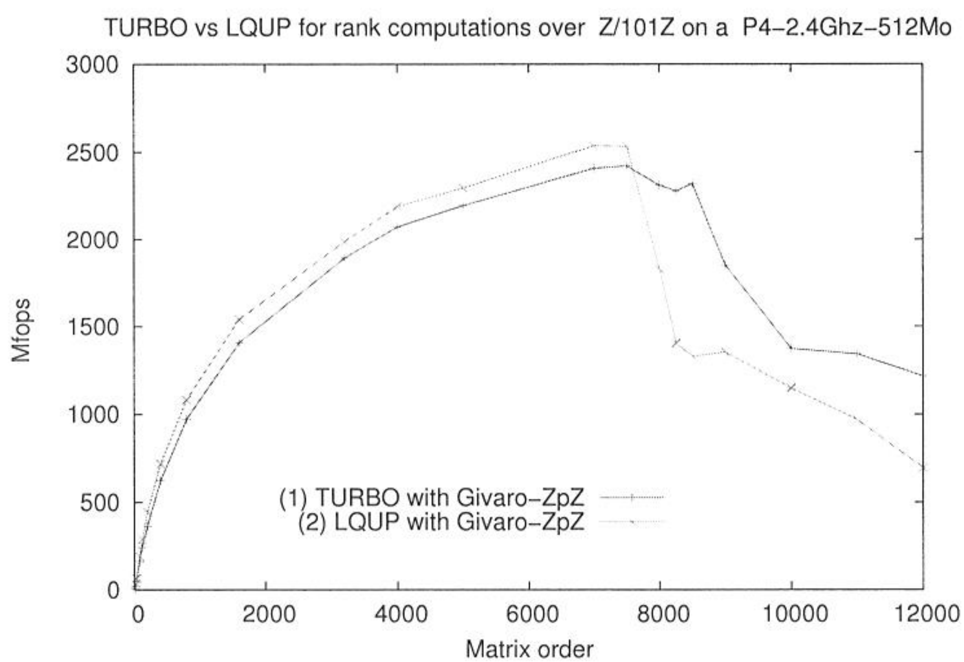
matrix	T_{iter} [min]	T_{seq}	T_{par} (50)	T_{par} (50, ET)
GL7d17	0.46875	621.8	12.4	8.16
GL7d20	0.68182	1361.31	27.2272	16.6214
GL7d21	0.35714	408.196	8.1644	5.5559

‾URBO triangular elimination

[Roch & Dumas 02]: recursive block algorithm for triangularization

- divide both rows and columns
  - ⇒Better memory management
  - ⇒Enables to use recursive data structures
- 5 recursive calls (U, V, C, D, Z), including 2 being parallel (C, D)

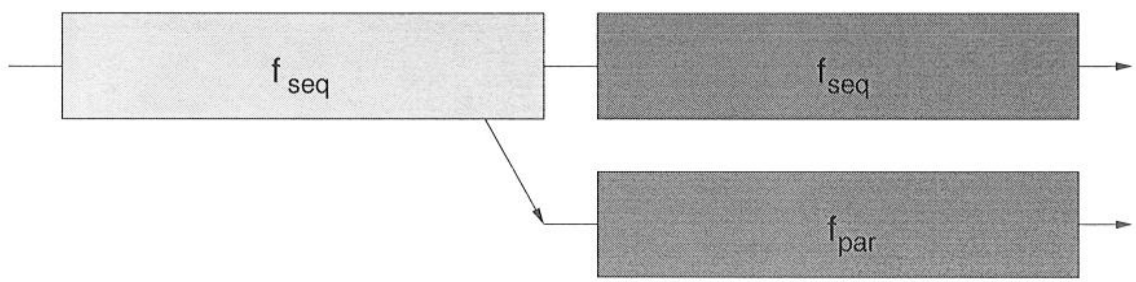




Principle of Workstealing

[Arora, Blumofe, Plaxton01], [Acar, Blelloche, Blumofe02]

- 2 algorithms to complete a task f: f_{seq} and f_{par}
- When a processor becomes idle, `ExtractPar` *steals* the work to f_{seq}.



Application to mutiple triangular system solving

$\texttt{TRSM}$: Compute $\begin{bmatrix} U_1 & U_2 \\ & U_3 \end{bmatrix}^{-1} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$

$$X_2 = \texttt{TRSM}(U_3, B_2)$$
$$B_1 = B_1 - U_2 X_2$$
$$X_1 = \texttt{TRSM}(U_1, B_1)$$

f_{seq}

$\texttt{TRSM}(U, B)$

$\Rightarrow T_1 = n^3,\ T_\infty = \mathcal{O}(n)$

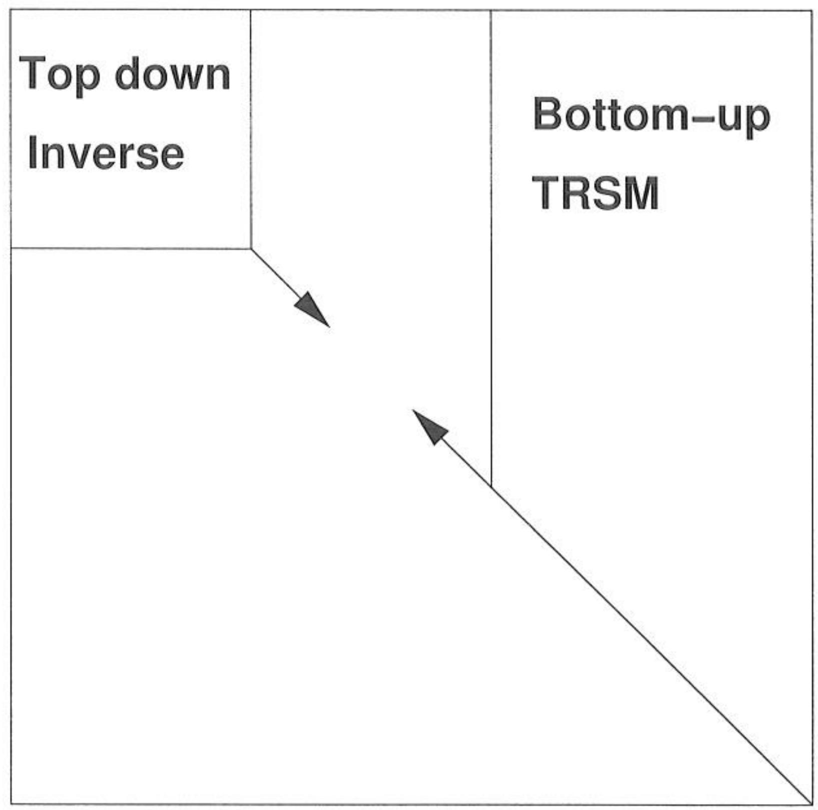
f_{par}

Compute $V = U^{-1}$;

$\texttt{TRMM}(V, B)$;

$\Rightarrow T_1 = \frac{4}{3} n^3,\ T_\infty = \mathcal{O}(\log n)$

\pplication to multiple triangular system ;olving



When sequential `TRSM` and parallel `Inverse` join:
Compute $X_1 = A_1^{-1} B_1$ in parallel (`TRMM`).

/ulti-adic lifting

Solving $Ax = b$ over $\mathbb{Z}$

Standard p-adic Lifting [Dixon82]

Compute $A^{-1} \mod p$
$r = b$
for $i = 0..n$ **do**
  $x_i = A^{-1} r \mod p$
  $r = (r - Ax_i)/p$
end for
$z = x_0 + px_1 + p^2 x_2 + \cdots + x_n p^n$
$x = \texttt{RatReconst}(z)$

/ulti-adic lifting

Solving $Ax = b$ over $\mathbb{Z}$

multi-adic lifting:

for all j=1..k **do**
  Compute $A^{-1} \mod p_j$
  $r = b$
  for $i = 0..n/k$ **do**
    $x_i = A^{-1} r \mod p_j$
    $r = (r - Ax_i)/p_j$
  end for
  $z_j = x_0 + p_j x_1 + \cdots + p_j^{n/k} x_{n/k}$
end for
$z = \text{ChineseRemainderAlg}((z_j, p_j^{n/k})_{j=1..k})$
$x = \texttt{RatReconst}(z)$

/ulti-adic lifting

- Used in sequential computation [Chen & Storjohann 05], to balance efficiency between BLAS level 2 and 3
- Divides a sequential loop into several parallel tasks
- Work stealing perspectives...

Conclusion

Large grain parallelism:

- Chinese remaindering
- Multi-adic lifting
- Block Wiedemann

Fine grain adaptive parallelsim:

$\Rightarrow$ Work stealing

Perspectives

- Development of simple parallel containers
- Parallel distribution of LinBox, based on Kaapi