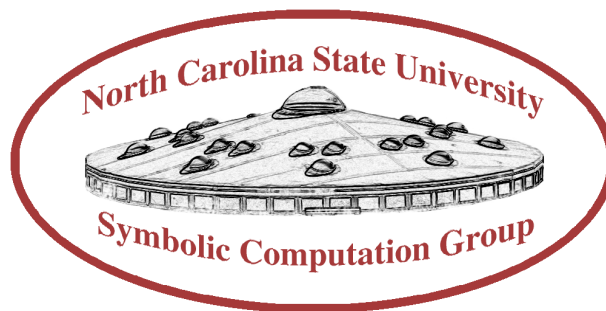


Black Box Linear Algebra with the LinBox Library

William J. Turner
North Carolina State University
www.math.ncsu.edu/~wjturner



Overview

1. Black Box Linear Algebra
2. Project LinBox
3. LinBox Objects
4. Black Box Algorithms
5. Black Box Preconditioners

Black Box Linear Algebra

Black box model of a matrix



Perform linear algebra operations, e.g., $A^{-1}b$ (Wiedemann, 1986)

Precondition to reduce matrix problems to computing minimum polynomials

Delayed matrix multiplication requires efficient matrix-vector products

Project LinBox

Approximately 20 researchers

Canada, France, and U.S.

Algorithms and software for symbolic linear algebra

Particularly black box matrix methods

40,000 lines of C++ code

Online documentation (Doc++)

LinBox Objects

Fields

Parameterized with encapsulated element and random element generator types

C++ types allowed to be elements; contain no information of field

Contain methods for element assignment, equality, arithmetic, IO:

$x = y$	:	<code>F.assign(x,y)</code>
$x == y$	:	<code>F.areEqual(x,y)</code>
$x = y + z$	:	<code>F.add(x,y,z)</code>
$x = x + y, x+ = y$	:	<code>F.addin(x,y)</code>
<code>cout << x</code>	:	<code>F.write(cout,x)</code>

Vectors

Three types of vectors:

dense vectors: “normal” vector,
e.g., STL `vector<Element>`

sparse sequence vectors: sequence of pairs of indices and nonzero elements,
e.g., STL `list<pair<integer,Element>>`

sparse associative vectors: association of indices to nonzero elements,
e.g., STL `map<integer,Element>`

Black Box Matrices

Templatized by vector class

Only applications to vector allowed:

$x = Ay$	:	<code>A.apply(x,y)</code> <code>x = A.apply(y)</code>
$x = Ax$	:	<code>A.applyin(x)</code>
$x = A^T y$	:	<code>A.transposeapply(x,y)</code> <code>x = A.transposeapply(y)</code>
$x = A^T x$	:	<code>A.transposeapplyin(x)</code>

Retrieve matrix dimensions:

`A.rowdim()`
`A.coldim()`

Archetypes

Three uses of archetypes:

1. To define the common object interface; i.e., specify what an explicitly designed class must have.
2. To distribute compiled code and prototype library components.
3. To control code bloat.

Used for all but the most basic objects such as floating point numbers and 10 by 10 matrices.

Not a Java interface; an archetype is an explicit class.

LinBox Field Archetype

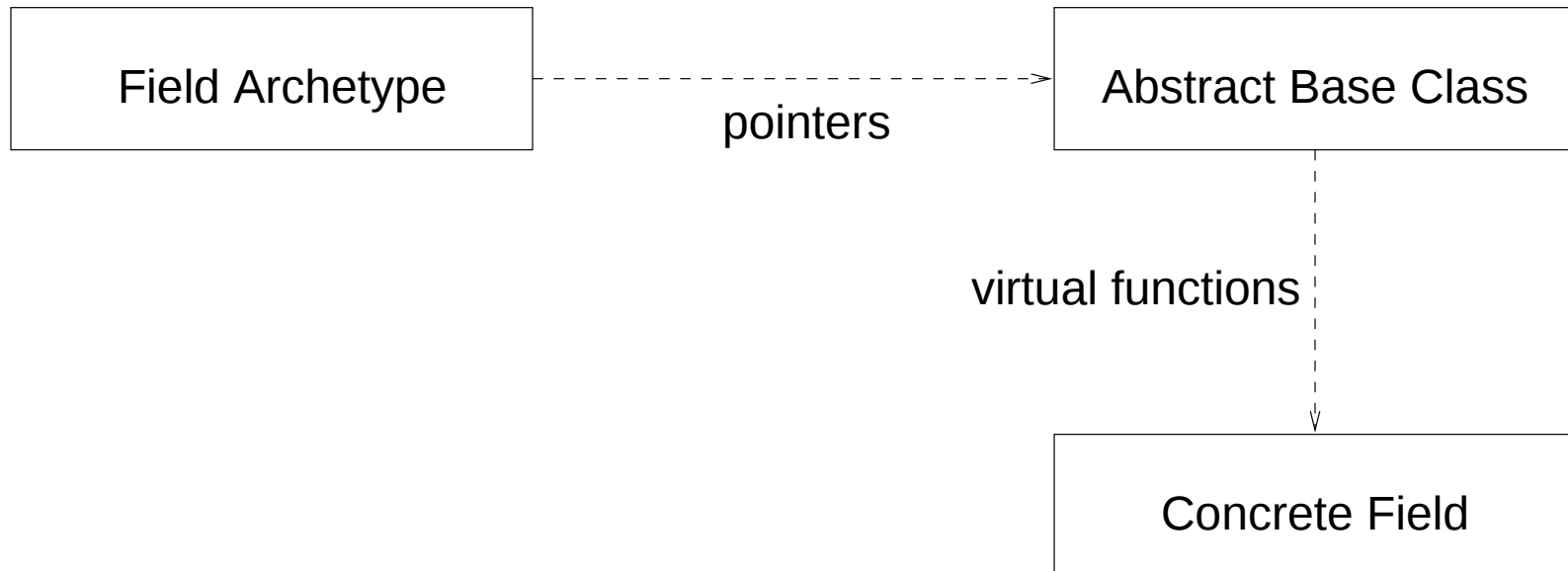
Allow C++ types such as doubles as field elements



Field elements must have (non-virtual) constructors



Field archetype $\neq$ Abstract base class



```
class Field_archetype
{
public:
    // Types
    typedef Element_archetype element;
    typedef RandIter_archetype randIter;

    // Object management
    element& init(element& x, const integer& y = 0 ) const;
    ...

    // Arithmetic
    element& add(element& x, const element& y, const element& z) const;
    ...

    // In-place arithmetic
    element& addin(element& x, const element& y) const;
    ...

    // I/O
    ostream& write(ostream& os) const;
    istream& read(istream& is);
    ostream& write(ostream& os, const element& x) const;
    istream& read(istream& is, element& x) const;
};
```

Black Box Matrix Archetype

No C++ types to be used as black box matrices



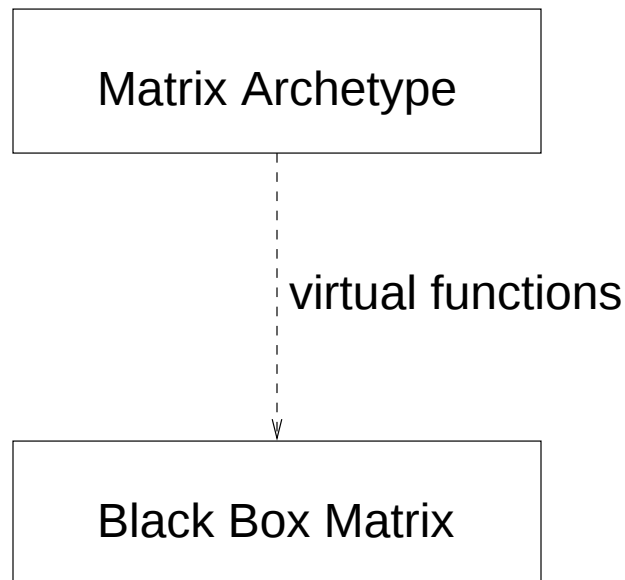
Black box matrices do not need (non-virtual) constructors



Use virtual `clone()` to construct derived classes



Black box matrix archetype = Abstract base class



```

template <class Vector> class Blackbox_archetype
{
public:
    virtual ~Blackbox_archetype() {}
    virtual Blackbox_archetype* clone() const = 0;
    virtual Vector& apply(const Vector& x) const = 0;
    virtual Vector& apply(Vector& y, const Vector& x) const
    { return y = apply(x); }
    virtual Vector& applyin(Vector& x) const
    { Vector y(x); return x = apply(y); }
    virtual Vector& applyTranspose(const Vector& x) const = 0;
    virtual Vector& applyTranspose(Vector& y, const Vector& x) const
    { return y = applyTranspose(x); }
    virtual Vector& applyinTranspose(Vector& x) const
    { Vector y(x); return x = applyTranspose(y); }
    virtual size_t rowdim(void) const = 0;
    virtual size_t coldim(void) const = 0;

protected:

    Blackbox_archetype(void) {}

};

```

Black Box Algorithms

Wiedemann Method

For $u, b \in \mathbb{F}^n$ and $A \in \mathbb{F}^{n \times n}$, let

- χ^A = characteristic polynomial of $A = \det(\lambda I - A)$
- f^A = minimal polynomial of A
- $f^{A,b}$ = minimal recurrence polynomial of $\{A^i b\}_{i=0,1,\dots}$
- $f_u^{A,b}$ = minimal recurrence polynomial of $\{u^\top A^i b\}_{i=0,1,\dots}$

Then,

$$f_u^{A,b} \mid f^{A,b} \mid f^A \mid \chi^A$$

Berlekamp-Massey algorithm (1967) computes minimal linear generator of $\{u^\top A^i b\}_{i=0,1,\dots}$

Let $f_u^{A,b} = \lambda^m + c_{m-1}\lambda^{m-1} + \dots + c_0$

Wiedemann (1986):

Linear Solver: Choose random u so $f_u^{A,b} = f^{A,b}$ and a solution to $Ax = b$ is

$$x = -\frac{1}{c_0}(A^{m-1}b + c_{m-1}A^{m-2}b + \dots + c_1b)$$

Minimal Polynomial: Choose random u and b so

$$f_u^{A,b} = f^{A,b} = f^A$$

Determinant: Randomly perturb A and choose random u and b so

$$f_u^{A,b} = f^{A,b} = f^A = \chi^A$$

and

$$\det A = (-1)^n c_0$$

Baby Steps/Giant Steps Determinant Algorithm

For $A \in \mathbb{F}^{n \times n}$ dense, can compute $a_i = u^T A^i b$ for $i = 0, 1, \dots, 2n - 1$ efficiently by baby steps/giant steps method (Kaltofen & Villard, 2001):

For some c , let $r = \lceil \sqrt{cn} \rceil$ and $s = \lceil 2n/r \rceil$

1. (Baby Steps) For $j = 1, 2, \dots, r - 1$ Do $v^{[j]} \leftarrow A^j v$;
2. $Z \leftarrow A^r$;
3. (Giant Steps) For $k = 1, 2, \dots, s$ Do $(u^{[k]})^T \leftarrow u^T Z^k$
4. For $k = 0, 1, 2, \dots, s$ Do
 For $j = 0, 1, 2, \dots, r - 1$ Do $a_{kr+j} \leftarrow \langle u^{[k]}, v^{[j]} \rangle$;

If $A \in \mathbb{Z}^{n \times n}$, compute $\det A \pmod{p}$ for several primes and Chinese remainder to recover $\det A$.

Kaltofen-Saunders Algorithms

Let $A \in \mathbb{F}^{n \times n}$ have rank $r < n$.

Then, Kaltofen & Saunders (1991) gives us:

Rank:

If

- $f^A = \lambda g(\lambda)$
- $\chi^A = \lambda^k g(\lambda)$
- $g(0) \neq 0$

Then,

$$r = \deg(f^A) - 1$$

Linear Solver:

Let

- A have leading principle minors nonzero up to A_r
- $A' \in \mathbb{F}^{r \times n}$ be the matrix formed by first r rows of A
- $b \in \mathbb{F}^n$ be such that $Ax = b$ is solvable in $x \in \mathbb{F}^n$
- $x_0 \in \mathbb{F}^n$ be such that $Ax_0 = b$
- $v \in \mathbb{F}^n$ be random

Then,

$$\begin{pmatrix} A_r^{-1} A'(x_0 + v) \\ 0 \end{pmatrix} - v$$

uniformly samples the solution manifold of $Ax = b$

Preconditioners

Schwartz-Zippel Lemma (Zippel, 1979; Schwartz, 1980):

If

- $f \in \mathbb{F}[x_1, \dots, x_m]$
- $f \neq 0$
- $n = \deg f$
- $S \subseteq \mathbb{F}$, S finite
- $a_1, \dots, a_m$ selected uniformly and independently from S

Then

$$\text{Prob}(f(a_1, \dots, a_m) \neq 0) \geq 1 - \frac{n}{|S|}$$

Determinant-Preserving Preconditioner

To compute determinant, want A' nonsingular and

$$f^{A'} = \chi^{A'}$$

Use preconditioners

Wiedemann (1986): Permute columns so all leading principle minors are nonzero and multiply by diagonal matrix

Chen *et al.* (2001): Multiply by diagonal matrix

These increase determinant

Baby steps/giant steps algorithm uses more primes for Chinese remaindering

Theorem 1. Let $\mathbb{F}$ be a field, $A \in \mathbb{F}^{n \times n}$ be nonsingular, and S be a finite subset of $\mathbb{F}$. If

$$U = \begin{bmatrix} 1 & a_1 & & \\ & \ddots & \ddots & \\ & & 1 & a_{n-1} \\ & & & 1 \end{bmatrix}$$

where $a_1, \dots, a_{n-1}$ are chosen uniformly and independently from S , then AU is nonsingular and the minimal polynomial of AU is the characteristic polynomial of AU with probability at least $1 - n(n-1)/(2|S|)$.

Can also use $AU^\top$, UA , and $U^\top A$.

Idea behind proof:

$$\mathcal{U} = \begin{bmatrix} 1 & \alpha_1 & & & \\ & \ddots & \ddots & & \\ & & 1 & \alpha_{n-1} & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$$

Smith form of $\lambda I - A\mathcal{U}$ is

$$\begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & f^{A\mathcal{U}} & \end{bmatrix}$$

There exists y such that

$$y, (AU)y, \dots, (AU)^{n-1}y$$

are linearly independent.

The determinant of the matrix with these columns is a nonzero polynomial of degree no more than $n(n-1)/2$ in $\alpha_1, \dots, \alpha_{n-1}$.

Apply Schwartz-Zippel to find probability that determinant of matrix with columns

$$y, (AU)y, \dots, (AU)^{n-1}y$$

is nonzero.

Rank Algorithm Preconditioner

For Kaltofen-Saunders rank algorithm, mix columns and rows so all leading principle minors up to rank are nonzero and multiply by diagonal matrix

Wiedemann permutes columns to also obtain all leading principle minors up to rank are nonzero

Theorem 2. Let $\mathbb{F}$ be a field, $A \in \mathbb{F}^{n \times n}$ have rank $r \leq n - 1$, and S be a finite subset of $\mathbb{F}$. If $D = \text{diag}(d_1, \dots, d_n)$ where $d_1, \dots, d_n$ are chosen uniformly and independently from S , then the degree of the minimal polynomial of AD is $r + 1$ with probability at least $1 - n(n - 1)/(2|S|)$.

Can also use DA .

Idea behind proof:

$$\mathcal{D} = \text{diag}(\delta_1, \dots, \delta_n)$$

Smith form of $\lambda I - A\mathcal{D}$ is

$$\begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \lambda & & \\ & & & & \ddots & \\ & & & & & \lambda \\ & & & & & & f^{A\mathcal{U}} \end{bmatrix}$$

There exists y such that

$$y, (AU)y, \dots, (AU)^r y$$

are linearly independent.

The determinant of the matrix with these columns is a nonzero polynomial of degree no more than $r(r+1)/2 \leq n(n-1)/2$ in $\delta_1, \dots, \delta_n$.

Apply Schwartz-Zippel to find probability that determinant of matrix with columns

$$y, (AU)y, \dots, (AU)^r y$$

is nonzero.

Kaltofen & Saunders (1991) uses random unit Toeplitz matrices for mixing

	Kaltofen-Saunders	New
Random elements	$5n - 2$	n
Matrix-vector prods	$O(n)$	$O(n)$
Additional field ops	$O(n^2 \log(n) \log \log(n))$	$O(n^2)$

Butterfly Network Preconditioner

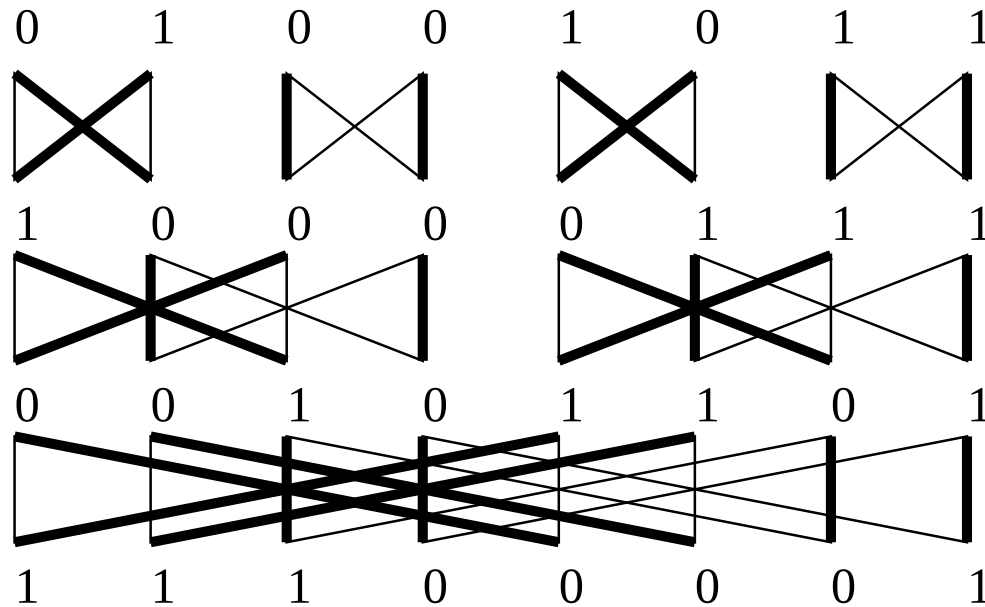
For Kaltofen-Saunders linear solver, need to permute or mix rows and columns so leading principle minor A_r nonzero

Wiedemann (1986): Extend A so $n = 2^k$ and use Beneš network

Kaltofen & Saunders (1991): Pre- and postmultiply by random unit upper and lower triangular Toeplitz matrices

Chen *et al.* (2001): Do not extend A and use two butterfly networks

For $n = 2^l$:

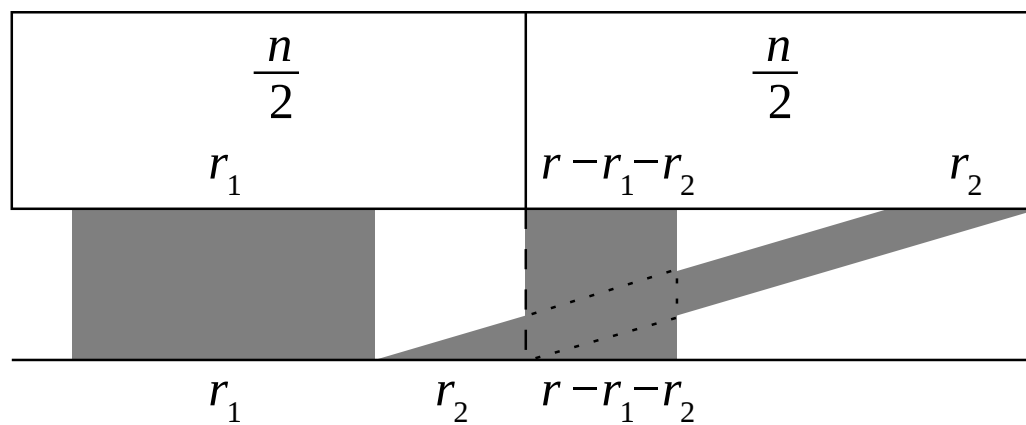
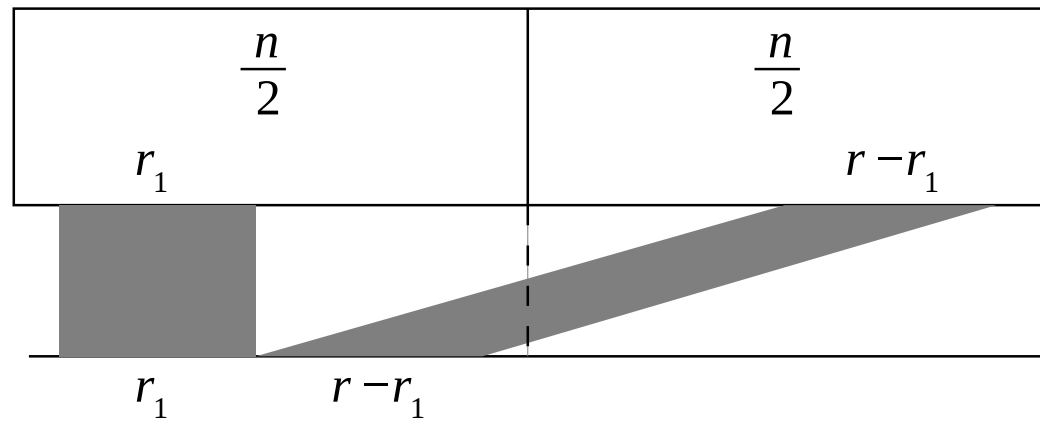


Lemma 3. Let $n = 2^l$. The l -dimensional butterfly network discussed above can switch any r indices

$$1 \leq i_1 < \dots < i_r \leq n$$

into any desired contiguous block of indices; wrap around outside, for our purposes, shall preserve contiguity. Furthermore, the network contains a total of $n \log_2(n)/2$ switches.

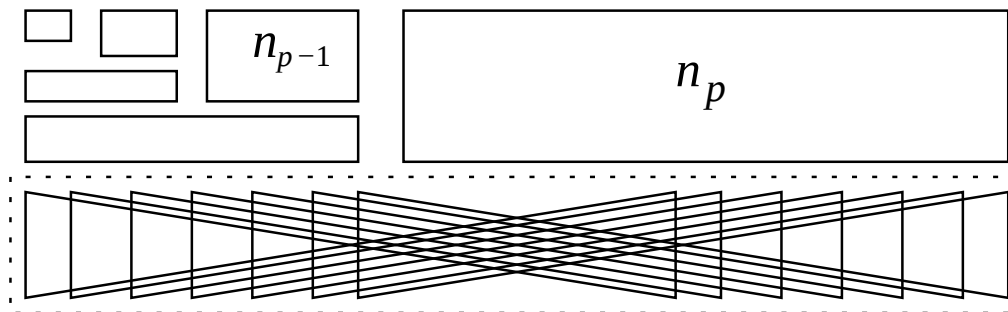
Idea behind proof:



If $n \neq 2^l$,

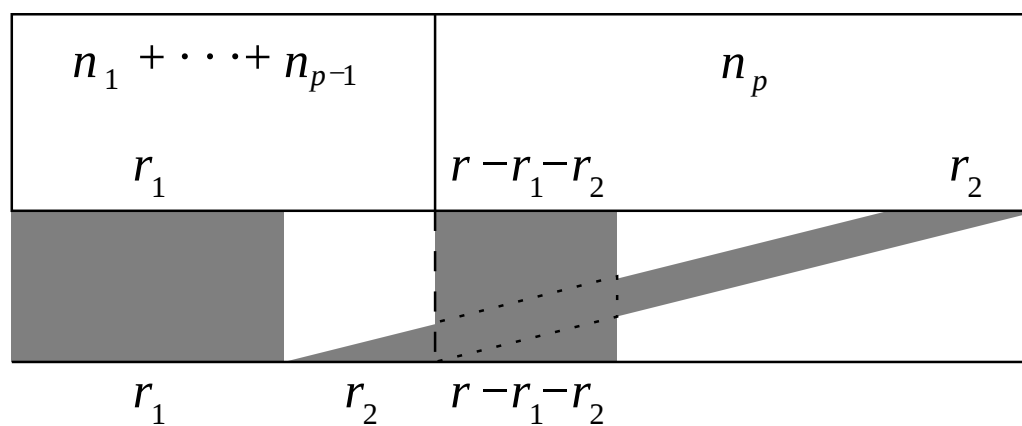
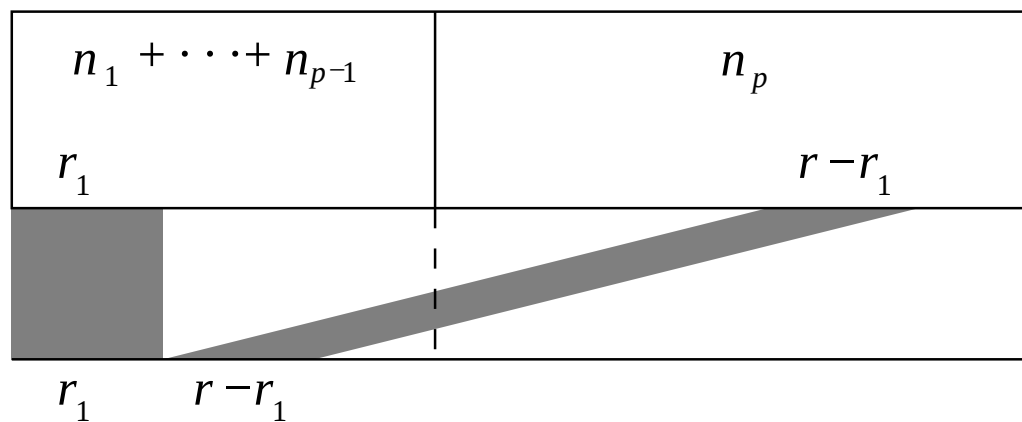
$$n = \sum_{i=1}^k 2^{l_i} \text{ where } l_1 < l_2 < \dots < l_p; \text{ let } n_i = 2^{l_i}$$

1. Lay out butterfly network for each n_i block
2. Connecting by butterfly switches recursively such that $\sum_{i=1}^{k-1} n_i$ merged with far right nodes of n_k



Theorem 4. *The generalized butterfly network discussed above can switch any r indices $1 \leq i_1 < \dots < i_r \leq n$ into the contiguous block $1, 2, \dots, r$. Furthermore, it has a depth of $\lceil \log_2(n) \rceil$ and a total of no more than $n \lceil \log_2(n) \rceil / 2$ butterfly switches.*

Idea behind proof:



If each boolean switch is replaced by a random switch

$$\begin{pmatrix} 1 & a \\ 1 & 1 + a \end{pmatrix}$$

then network switches indices $1 \leq i_1 < \dots < i_r \leq n$ to contiguous block $1, 2, \dots, r$ with probability no less than $1 - n^{\lceil \log_2(n) \rceil} / |S|$.

References

- E. R. BERLEKAMP (1968). *Algebraic Coding Theory*. McGraw-Hill, New York.
- L. CHEN, W. EBERLY, E. KALTOFEN, B. D. SAUNDERS, W. J. TURNER, & G. VILLARD (2001). Efficient Matrix Preconditioners for Black Box Linear Algebra. *Linear Algebra and Applications*. Special issue on *Infinite Systems of Linear Equations Finitely Specified*.
- E. KALTOFEN & B. D. SAUNDERS (1991). On Wiedemann's Method of Solving Sparse Linear Systems. In H. F. MATTSON, T. MORA, & T. R. N. RAO (eds.), *Proc. AAEECC-9*, vol. 539 of *Lect. Notes Comput. Sci.*, 29–38. Springer Verlag, Heidelberg, Germany.
- E. KALTOFEN & G. VILLARD (2001). On the complexity of computing determinants. In *Proc. Fifth Asian Symposium on Computer Mathematics ASCM 2001*. World Scientific Publishing Company. Invited contribution; extended abstract.
- J. L. MASSEY (1969). Shift-Register Synthesis and BCH Decoding. *IEEE Trans. Inf. Theory*, **IT-15**: 122–127.
- J. T. SCHWARTZ (1980). Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, **27**: 701–717.
- D. H. WIEDEMANN (1986). Solving Sparse Linear Equations Over Finite Fields. *IEEE Trans. Inf. Theory*, **IT-32**: 54–62.
- R. ZIPPEL (1979). Probabilistic Algorithms for Sparse Polynomials. In *Proc. EUROSAM '79*, vol. 72 of *Lect. Notes Comput. Sci.*, 216–226. Springer Verlag, Heidelberg, Germany.