Sparse matrices in computer algebra when using distributed memory: theory and applications

G. Malaschonok, E. Ilchenko

Jerusalem, ACA-2017, 17-21 July, 2017

1National University Kiev-Mogila Academy, Kiev, Ukraine and Tambov State University, Tambov, Russia
History
Outline of the talk

History

Applications
Outline of the talk

- History
- Applications
- Algorithms
- Experiments
The generalization of Strassen’s matrix inversion algorithm (1969) with additional permutations of rows and columns by J. Bunch and J. Hopkroft (1974) is not a block-recursive algorithm.
History: block-recursive matrix algorithms

The generalization of Strassen’s matrix inversion algorithm (1969) with additional permutations of rows and columns by J. Bunch and J. Hopcroft (1974) is not a block-recursive algorithm.

Block-recursive algorithms were not so important as long as the calculations were performed on computers with shared memory...
The generalization of Strassen’s matrix inversion algorithm (1969) with additional permutations of rows and columns by J. Bunch and J. Hopcroft (1974) is not a block-recursive algorithm.

Block-recursive algorithms were not so important as long as the calculations were performed on computers with shared memory...

... but very important for sparse super large matrices on a supercomputer with distributed memory.
Generalisation of Strassen’s inversion and algorithm for the solution of systems in commutative domains was described at 1997-2006 ([7], [8], [10], [9]) with strong restriction: the leading minors should not be zero.
Algorithms for solution of a system of linear equations of size $n$ in an integral domain, which served as the basis for creating recursive algorithms:

- **(1983)** Forward and backward algorithm ($\sim n^3$) [4].
- **(1989)** One pass algorithm ($\sim \frac{2}{3} n^3$) [5].
- **(1995)** Combined algorithm with upper left block of size $r$ ($\sim \frac{7}{12} n^3$ for $r = \frac{n}{2}$) [6].
Recursive algorithms for solution of a system of linear equations and for adjoint matrix computation in an integral domain without permutations:

(1997) Recursive algorithm for solution of a system of linear equations [7].
(2000) Adjoint matrix computation (with 6 levels) [8].
(2006) Adjoint matrix computation alternative algorithm (with 5 levels) [10].
This restriction was removed at (2008-2015):

**Main recursive algorithms for sparse matrices:**

(2008) The algorithm that computes the adjoint matrix, the echelon form, and the kernel of the matrix operator for the commutative domains was described in [11].

(2010) The block-recursive algorithm for the Bruhat decomposition and the LEU decomposition for the matrix over the field was obtained in [12].

(2013, 2015) and these algorithms were generalized to the LDU and Bruhat decomposition for the matrices over commutative domains in [14], [15].
New achievements:

It is proved that the LEU algorithm has the complexity $O(n^2 r^{\beta-2})$ for matrices of rank $r$. [19] (2013).

It is proved that the LEU algorithm has the complexity $O(n^2 s^{\beta-2})$ for quasiseparable matrix, if any it’s submatrix which entirely below or above the main diagonal has small rank $s$ [20] (2017).
1) Applications: Calculation of electronic circuits.

The behavior of electronic circuits can be described by Kirchhoff’s laws. The three basic approaches in this theory are direct current, constant frequency current and a current that varies with time. All these cases require the compilation and solution of sparse systems of equations (numerical, polynomial or differential). The solution of such differential equations by the Laplace method also leads to the solution of polynomial systems of equations [16].
2) Applications: Control systems.

In 1967 Howard H. Rosenbrock introduced a useful state-space representation and transfer function matrix form for control systems, which is known as the Rosenbrock System Matrix [17]. Since that time, the properties of the matrix of polynomials being intensively studied in the literature of linear control systems.
3). Applications: Computation of Groebner basis.

A matrix composed of Buchberger S-polynomials is a strongly sparse matrix. Reduction of the polynomial system is performed when calculating the echelon and diagonal forms of this matrix. The algorithm F4 [18] was the first such matrix algorithm.
4) Applications: Solving ODE’s and PDEs.
Solving ODE’s and PDE’s is often based on solution of linear systems with sparse matrices over numbers or over polynomials. One of the important class of sparse matrix is called quasiseparable. Any submatrix of quasiseparable matrix entirely below or above the main diagonal has small rank. These quasiseparable matrices arise naturally in solving PDEs for particle interaction with the Fast Multi-pole Method (FMM). The efficiency of application of the block-recursive algorithm of the Bruhat decomposition to the quasiseparable matrices is studied in [20].
Recursive matrix multiplication for tree trunk and branches

\[
\begin{pmatrix}
A_0 & A_1 \\
A_2 & A_3
\end{pmatrix} \times \begin{pmatrix}
B_0 & B_1 \\
B_2 & B_3
\end{pmatrix} + \begin{pmatrix}
C_0 & C_1 \\
C_2 & C_3
\end{pmatrix} = \begin{pmatrix}
D_0 & D_1 \\
D_2 & D_3
\end{pmatrix}
\]

\[D = AB + C\]
Recursive sparse matrix multiplication on the leaf’s block

A:

\[ \begin{array}{ccc}
2_{20,91}, & 3_{20,93} \\
-4_{21,90}, & 7_{21,92}, & 1_{21,93} \\
5_{22,91} \\
6_{23,90}, & -3_{23,91}
\end{array} \]

B:

\[ \begin{array}{ccc}
4_{90,51}, & 1_{90,52}, & -2_{90,53} \\
7_{91,50}, & 8_{91,51} \\
2_{92,50}, & 5_{92,53} \\
1_{93,50}, & 4_{93,51}, & -4_{93,53}
\end{array} \]

Storage of sparse matrices without zero elements
If $A = \begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix}$, $\det(A) \neq 0$ and $\det(A_0) \neq 0$ then

$$A^{-1} = \begin{pmatrix} I & -A_0^{-1}A_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & (A_3 - A_2A_0^{-1}A_1)^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -A_2 & I \end{pmatrix} \begin{pmatrix} A_0^{-1} & 0 \\ 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} A_0^{-1} + A_0^{-1}A_1(A_3 - A_2A_0^{-1}A_1)^{-1}A_2A_0^{-1} & -A_0^{-1}A_1(A_3 - A_2A_0^{-1}A_1)^{-1} \\ -(A_3 - A_2A_0^{-1}A_1)^{-1}A_2A_0^{-1} & (A_3 - A_2A_0^{-1}A_1)^{-1} \end{pmatrix}.$$

If $M_0 = -A_0^{-1}$, $M_1 = M_0A_1$, $M_2 = A_2M_0$, $M_3 = M_2A_1$, $M_4 = (A_3 + M_3)^{-1}$, $M_5 = -M_4M_2$, then

$$A^{-1} = \begin{pmatrix} M_1M_5 - M_0 & M_1M_4 \\ M_5 & M_4 \end{pmatrix}.$$
Recursive computation of the adjoint and kernel: 1 of 2

\[ M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \]

\[ A_{\text{ext}}(M_{11}, d_0) = (A_{11}, S_{11}, E_{11}, d_{11}). \]

\[ M_{12}^1 = \frac{A_{11} M_{12}}{d_0} , \quad M_{21}^1 = -\frac{M_{21} Y_{11}}{d_0} , \quad M_{22}^1 = \frac{M_{22} d_{11} - M_{21} E_{11}^T M_{12}^1}{d_0}. \]

\[ A_{\text{ext}}(\bar{I}_{11} M_{12}^1, d_{11}) = (A_{12}, S_{12}, E_{12}, d_{12}), \quad A_{\text{ext}}(M_{21}^1, d_{11}) = (A_{21}, S_{21}, E_{21}, d_{21}). \]

\[ M_{22}^2 = -\frac{A_{21} M_{22}^1 Y_{12}}{(d_{11})^2} , \quad d_s = \frac{d_{21} d_{12}}{d_{11}}. \]

\[ A_{\text{ext}}(\bar{I}_{21} M_{22}^2, d_s) = (A_{22}, S_{22}, E_{22}, d_{22}). \]

\[ M_{11}^2 = -\frac{S_{11} Y_{21}}{d_{11}} , \quad M_{12}^2 = \left( \frac{S_{11} E_{21}^T A_{21}}{d_{11}} \frac{M_{22}^1 - I_{11} M_{12} d_{21}}{d_{11}} \right) Y_{12} + S_{12} d_{21} , \quad M_{12}^3 = -\frac{M_{12}^2 Y_{22}}{d_s} , \]
Recursive computation of the adjoint and kernel: 2 of 2

\[
M_{22}^3 = S_{22} - \frac{l_{21} M_{22}^2 Y_{22}}{d_s}, \quad A^1 = A_{12} A_{11}, \quad A^2 = A_{22} A_{21},
\]

\[
L = \left( A^1 - \frac{l_{11} M_{12}^1 E_{12}^T A^1}{d_{11}} \right) d_{22}, \quad P = \frac{A^2 - \frac{l_{21} M_{22}^2 E_{22}^T A^2}{d_s}}{d_{21}},
\]

\[
F = - \frac{\left( S_{11} E_{21}^T A_{21} \right) d_{22} + \frac{M_{12}^2 E_{22}^T A^2}{d_s}}{d_{21}}, \quad G = - \frac{\left( M_{21} E_{11}^T \right) d_{12} + \frac{M_{22}^1 E_{12}^T A^1}{d_{11}}}{d_{11}},
\]

\[
A = \begin{pmatrix} \frac{L+FG}{d_{12}} & F \\ \frac{PG}{d_{12}} & P \end{pmatrix}, \quad S = \begin{pmatrix} \frac{M_{11}^2 d_{22}}{d_{21}} & M_{12}^3 \\ \frac{S_{21} d_{22}}{d_{21}} & M_{22}^3 \end{pmatrix}, \quad E = \begin{pmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{pmatrix}, \quad d = d_{22}.
\]

Then

\[
A_{\text{ext}}(M, d_0) = (A, S, E, d_{22}).
\]

\[
l_{ij} = E_{ij} E_{ij}^T, \quad \bar{l}_{ij} = I - l_{ij}, \quad Y_{ij} = E_{ij}^T S_{ij} - d_{ij} I, \quad i, j \in 1, 2.
\]
Recursive adjoint matrix computation

G. Malaschonok, E. Ilchenko

ACA 2017, Jerusalem
The block-recursive matrix algorithms for sparse matrix require a special approach to managing parallel programs. One approach to the cluster computations management is a scheme with one dispatcher (or one master).

We consider another scheme of cluster management. It is a scheme with multidispatching, when each involved computing module has its own dispatch thread and several processing threads [21], [22]. We demonstrate the results of experiments with parallel programs on the base of multidispatching.
Recursive matrix multiplication (dense, n=8000, Z, 15b)
Recursive matrix multiplication (dense, n=12000, Z, 15b)

G. Malaschonok, E. Ilchenko

ACA 2017, Jerusalem
Recursive matrix multiplication (dense, n=14000, Z, 15b)

![Graph showing efficiency of different cores. The y-axis represents efficiency in %, and the x-axis represents cores (N). The graph shows a decreasing trend as the number of cores increases, with a 50% efficiency mark.]
Recursive inversion Strassen (dense, n=8000, double)
Recursive inversion Strassen (dense, n=16000, double)

![Graph showing efficiency vs. number of cores (N)].

50% efficiency
Recursive adjoint and kernel (dence, n=8000, Z)
Recursive adjoint and kernel \((d=100\%, n=100, Z, 15b, \text{CRT}+\text{P})\)
Recursive adjoint and kernel (d=1%, n=10000, Z, 15b, CRT+P)
Recursive adjoint and kernel (d=1%, n=10000, Z, CRT+P)

G. Malaschonok, E. Ilchenko

ACA 2017, Jerusalem
Comparing sequential program with Mathematica and MAPLE

G. Malaschonok, E. Ilchenko

ACA 2017, Jerusalem

*Strassen V.* *Gaussian Elimination is not optimal.* Numerische Mathematik. V. 13, Issue 4, 354–356 (1969)


