Comprehensive LU Decomposition and True Path

Aishat Olagunju, Marc Moreno Maza, David Jeffrey

Western University

September, 2024

Ontario Research Centre for Computer Algebra

1. Overview
2. LU Decomposition
3. Constructible Sets
4. Comprehensive LU
5. Comprehensive Pivot Search
6. Experimentation
7. Conclusion
8. References

1. Overview
2. LU Decomposition
3. Constructible Sets
4. Comprehensive LU
5. Comprehensive Pivot Search
6. Experimentation
7. Conclusion
8. References

Overview

- We study the PLU decomposition of rectangular matrices with polynomial coefficients, depending on parameters, possibly subject to algebraic constraints.

Overview

- We study the PLU decomposition of rectangular matrices with polynomial coefficients, depending on parameters, possibly subject to algebraic constraints.
- Computing such a parametric decomposition typically requires to split computations into cases.

Overview

- We study the PLU decomposition of rectangular matrices with polynomial coefficients, depending on parameters, possibly subject to algebraic constraints.
- Computing such a parametric decomposition typically requires to split computations into cases.
- This yields various computational challenges, in particular (possibly too) many cases

Overview

- We study the PLU decomposition of rectangular matrices with polynomial coefficients, depending on parameters, possibly subject to algebraic constraints.
- Computing such a parametric decomposition typically requires to split computations into cases.
- This yields various computational challenges, in particular (possibly too) many cases
- We present heuristic methods which attempt to minimize the number of cases.

Overview

- We study the PLU decomposition of rectangular matrices with polynomial coefficients, depending on parameters, possibly subject to algebraic constraints.
- Computing such a parametric decomposition typically requires to split computations into cases.
- This yields various computational challenges, in particular (possibly too) many cases
- We present heuristic methods which attempt to minimize the number of cases.
- In particular, these methods try to avoid splitting the computations, if this is possible.

1 Overview

2 LU Decomposition

3 Constructible Sets

4 Comprehensive LU

5 Comprehensive Pivot Search

6 Experimentation

7 Conclusion

8 References

Definition

LU Decomposition

Given a matrix A, it can be decomposed (factorized) into a permutation matrix P, a lower triangular matrix L and an upper triangular matrix U.

$$A = PLU$$

Definition

LU Decomposition

Given a matrix A, it can be decomposed (factorized) into a permutation matrix P, a lower triangular matrix L and an upper triangular matrix U.

$$A = PLU$$

This is the most common method, where the L matrix ends up with $1's$ on the diagonal.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Recursive PLU

We utilize the Recursive PLU decomposition introduced by Cormen et al [1] in the book " Introduction to Algorithms".

Recursive PLU

We utilize the Recursive PLU decomposition introduced by Cormen et al [1] in the book " Introduction to Algorithms".

Recursive PLU

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$.

Recursive PLU

We utilize the Recursive PLU decomposition introduced by Cormen et al [1] in the book " Introduction to Algorithms".

Recursive PLU

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$.
- If $\mathbf{m}$ is 1 then RecursiveLU$(\mathbf{A})$ returns $(\mathbf{I}_1, \mathbf{I}_1, \mathbf{A})$

Recursive PLU

We utilize the Recursive PLU decomposition introduced by Cormen et al [1] in the book " Introduction to Algorithms".

Recursive PLU

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$.
- If $\mathbf{m}$ is 1 then RecursiveLU$(\mathbf{A})$ returns $(\mathbf{I}_1, \mathbf{I}_1, \mathbf{A})$
- If $\mathbf{m} > 1$ then RecursiveLU$(\mathbf{A})$ returns $(\mathbf{P}, \mathbf{L}, \mathbf{U})$ where:
  - $\mathbf{P}$ is an $\mathbf{m} \times \mathbf{m}$ row permutation matrix,
  - $\mathbf{L}$ is an $\mathbf{m} \times \mathbf{m}$ lower triangular matrix, and
  - $\mathbf{U}$ is $\mathbf{m} \times \mathbf{n}$ upper triangular matrix so that $\mathbf{A} = \mathbf{PLU}$ holds.

See next slide.

Recursive PLU Steps

If the first column of A contains a non-zero entry a, we write

$$P_1 A = \left[\begin{array}{c|c} a & w^T \\ \hline v & A' \end{array}\right]$$

and set $c = 1/a$, otherwise we set $c = 0$.

Recursive PLU Steps

If the first column of A contains a non-zero entry a, we write

$$P_1 A = \left[\begin{array}{c|c} a & w^T \\ \hline v & A' \end{array}\right]$$

and set $c = 1/a$, otherwise we set $c = 0$.
We make a recursive call on A' and write:

$$P_1 A = \left[\begin{array}{c|c} 1 & 0 \\ \hline cv & I_{n-1} \end{array}\right] \left[\begin{array}{c|c} a & w^T \\ \hline 0 & A' - cvw^T \end{array}\right]$$

where $P'(A' - cvw^T) = L'U'$.

Recursive PLU Steps

If the first column of A contains a non-zero entry a, we write

$$P_1 A = \left[\begin{array}{c|c} a & w^T \\ \hline v & A' \end{array}\right]$$

and set $c = 1/a$, otherwise we set $c = 0$.
We make a recursive call on A' and write:

$$P_1 A = \left[\begin{array}{c|c} 1 & 0 \\ \hline cv & I_{n-1} \end{array}\right] \left[\begin{array}{c|c} a & w^T \\ \hline 0 & A' - cvw^T \end{array}\right]$$

where $P'(A' - cvw^T) = L'U'$.
Let $v' = P'v$, thus

$$\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & P' \end{array}\right] P_1 A = \left[\begin{array}{c|c} 1 & 0 \\ \hline cv' & L' \end{array}\right] \left[\begin{array}{c|c} a & w^T \\ \hline 0 & U' \end{array}\right]$$

Parametric Examples

There are parametric matrices that are of interest in practice. So it is natural to adapt linear algebra algorithms like LU decomposition, rank computation, Jordan, Hermite and Smith normal forms .

Parametric Examples

There are parametric matrices that are of interest in practice. So it is natural to adapt linear algebra algorithms like LU decomposition, rank computation, Jordan, Hermite and Smith normal forms .

Moon's matrix about chaotic vibrations [2]

$$J = \begin{bmatrix} -a & -b & -d & 0 \\ b & -a & 0 & -d \\ d & 0 & -a & -b \\ 0 & d & b & -a \end{bmatrix}$$

Parametric Examples

There are parametric matrices that are of interest in practice. So it is natural to adapt linear algebra algorithms like LU decomposition, rank computation, Jordan, Hermite and Smith normal forms .

Moon's matrix about chaotic vibrations [2]

$$J = \begin{bmatrix} -a & -b & -d & 0 \\ b & -a & 0 & -d \\ d & 0 & -a & -b \\ 0 & d & b & -a \end{bmatrix}$$

5X5 Kac Murdock Szegö Matrix [3]

$$A5 = \begin{bmatrix} 1 & -p & 0 & 0 & 0 \\ -p & p^2+1 & -p & 0 & 0 \\ 0 & -p & p^2+1 & -p & 0 \\ 0 & 0 & -p & p^2+1 & -p \\ 0 & 0 & 0 & -p & 1 \end{bmatrix}$$

Example without Case Discussion

- One first idea to deal with polynomial matrices would be to do a LU decomposition over $\mathbb{Q}(a, b, c)$. For instance:

$$\begin{bmatrix} a & 2 & 1 \\ b & 4 & 3 \\ c & 6 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \frac{b}{a} & 1 & 0 \\ \frac{c}{a} & \frac{3a-c}{2a-b} & 1 \end{bmatrix} \begin{bmatrix} a & 2 & 1 \\ 0 & \frac{4a-2b}{a} & \frac{3a-b}{a} \\ 0 & 0 & \frac{a-2b+c}{2a-b} \end{bmatrix}$$

Example without Case Discussion

- One first idea to deal with polynomial matrices would be to do a LU decomposition over $\mathbb{Q}(a, b, c)$. For instance:

$$\begin{bmatrix} a & 2 & 1 \\ b & 4 & 3 \\ c & 6 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \frac{b}{a} & 1 & 0 \\ \frac{c}{a} & \frac{3a-c}{2a-b} & 1 \end{bmatrix} \begin{bmatrix} a & 2 & 1 \\ 0 & \frac{4a-2b}{a} & \frac{3a-b}{a} \\ 0 & 0 & \frac{a-2b+c}{2a-b} \end{bmatrix}$$

- But in practice, one may want to know which values of a, b or c this LU decomposition could be specialized.

Example without Case Discussion

- One first idea to deal with polynomial matrices would be to do a LU decomposition over $\mathbb{Q}(a, b, c)$. For instance:

$$\begin{bmatrix} a & 2 & 1 \\ b & 4 & 3 \\ c & 6 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \frac{b}{a} & 1 & 0 \\ \frac{c}{a} & \frac{3a-c}{2a-b} & 1 \end{bmatrix} \begin{bmatrix} a & 2 & 1 \\ 0 & \frac{4a-2b}{a} & \frac{3a-b}{a} \\ 0 & 0 & \frac{a-2b+c}{2a-b} \end{bmatrix}$$

- But in practice, one may want to know which values of a, b or c this LU decomposition could be specialized.
- Specializing the above result at $b = 2a$ yields a division by 0.

Example without Case Discussion

- One first idea to deal with polynomial matrices would be to do a LU decomposition over $\mathbb{Q}(a, b, c)$. For instance:

$$\begin{bmatrix} a & 2 & 1 \\ b & 4 & 3 \\ c & 6 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \frac{b}{a} & 1 & 0 \\ \frac{c}{a} & \frac{3a-c}{2a-b} & 1 \end{bmatrix} \begin{bmatrix} a & 2 & 1 \\ 0 & \frac{4a-2b}{a} & \frac{3a-b}{a} \\ 0 & 0 & \frac{a-2b+c}{2a-b} \end{bmatrix}$$

- But in practice, one may want to know which values of a, b or c this LU decomposition could be specialized.
- Specializing the above result at $b = 2a$ yields a division by 0.
- Hence, to deal with parameters, we can not simply run LU decomposition over a field of rational functions.

1. Overview
2. LU Decomposition
3. Constructible Sets
4. Comprehensive LU
5. Comprehensive Pivot Search
6. Experimentation
7. Conclusion
8. References

Constructible Sets

- A *constructible set* is the solution set of a disjunction of conjunctions of polynomial equations and inequations.

Constructible Sets

- A *constructible set* is the solution set of a disjunction of conjunctions of polynomial equations and inequations.
- Let $\mathbb{K}[X_1, \ldots, X_n]$ be the ring of polynomials over the field $\mathbb{K}$ and with variables $X_1, \ldots, X_n$. [4]

Constructible Sets

- A *constructible set* is the solution set of a disjunction of conjunctions of polynomial equations and inequations.
- Let $\mathbb{K}[X_1, \ldots, X_n]$ be the ring of polynomials over the field $\mathbb{K}$ and with variables $X_1, \ldots, X_n$. [4]

Definition

- A *regular system* of $\mathbb{K}[X_1, \ldots, X_n]$ is a pair $S = [T, h]$ where $T \subset \mathbb{K}[X_1, \ldots, X_n]$ is a regular chain and $h \in \mathbb{K}[X_1, \ldots, X_n]$.

Constructible Sets

- A *constructible set* is the solution set of a disjunction of conjunctions of polynomial equations and inequations.
- Let $\mathbb{K}[X_1, \ldots, X_n]$ be the ring of polynomials over the field $\mathbb{K}$ and with variables $X_1, \ldots, X_n$. [4]

Definition

- A *regular system* of $\mathbb{K}[X_1, \ldots, X_n]$ is a pair $S = [T, h]$ where $T \subset \mathbb{K}[X_1, \ldots, X_n]$ is a regular chain and $h \in \mathbb{K}[X_1, \ldots, X_n]$.
- The zero set $Z(S)$ of S is the subset of the "zero set" $W(T)$ of T where h does not vanish.

Constructible Sets

- A *constructible set* is the solution set of a disjunction of conjunctions of polynomial equations and inequations.
- Let $\mathbb{K}[X_1, \ldots, X_n]$ be the ring of polynomials over the field $\mathbb{K}$ and with variables $X_1, \ldots, X_n$. [4]

Definition

- A *regular system* of $\mathbb{K}[X_1, \ldots, X_n]$ is a pair $S = [T, h]$ where $T \subset \mathbb{K}[X_1, \ldots, X_n]$ is a regular chain and $h \in \mathbb{K}[X_1, \ldots, X_n]$.
- The zero set $Z(S)$ of S is the subset of the "zero set" $W(T)$ of T where h does not vanish.

- For every constructible set C one can compute regular systems $S_1, \ldots, S_e$ so that we have

$$C = Z(S_1) \cup \cdots \cup Z(S_e).$$

Constructible Sets

- A *constructible set* is the solution set of a disjunction of conjunctions of polynomial equations and inequations.
- Let $\mathbb{K}[X_1, \ldots, X_n]$ be the ring of polynomials over the field $\mathbb{K}$ and with variables $X_1, \ldots, X_n$. [4]

Definition

- A *regular system* of $\mathbb{K}[X_1, \ldots, X_n]$ is a pair $S = [T, h]$ where $T \subset \mathbb{K}[X_1, \ldots, X_n]$ is a regular chain and $h \in \mathbb{K}[X_1, \ldots, X_n]$.
- The zero set $Z(S)$ of S is the subset of the "zero set" $W(T)$ of T where h does not vanish.

- For every constructible set C one can compute regular systems $S_1, \ldots, S_e$ so that we have
$$C = Z(S_1) \cup \cdots \cup Z(S_e).$$
- Given two constructible sets C_1, C_2 represented by regular systems, one can deduce a regular system representation for sets
$$C_1 \setminus C_2,\ C_1 \cap C_2 \text{ and } C_1 \cup C_2.$$

1 Overview

2 LU Decomposition

3 Constructible Sets

4 Comprehensive LU

5 Comprehensive Pivot Search

6 Experimentation

7 Conclusion

8 References

ComprehensiveLU is an adaptation of RecursiveLU to matrices depending on parameters. The main enhancements are:

Specification

ComprehensiveLU is an adaptation of RecursiveLU to matrices depending on parameters. The main enhancements are:

- the search for a pivot, is comprehensive and this is where computations may split,

Specification

ComprehensiveLU is an adaptation of RecursiveLU to matrices depending on parameters. The main enhancements are:

- the search for a pivot, is comprehensive and this is where computations may split,
- the recursive calls may return several branches.

Specification

ComprehensiveLU is an adaptation of RecursiveLU to matrices depending on parameters. The main enhancements are:

- the search for a pivot, is comprehensive and this is where computations may split,
- the recursive calls may return several branches.

Specification

Input: $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.

ComprehensiveLU is an adaptation of RecursiveLU to matrices depending on parameters. The main enhancements are:

- the search for a pivot, is comprehensive and this is where computations may split,
- the recursive calls may return several branches.

Specification

Input: $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.

Output: ComprehensiveLU$(\mathbf{A}, W)$ returns a list of tuples $[\mathbf{P}_i, \mathbf{Q}_i, \mathbf{L}_i, \mathbf{U}_i, W_i]$, called *branches*, for $1 \leq i \leq e$:

ComprehensiveLU is an adaptation of RecursiveLU to matrices depending on parameters. The main enhancements are:

- the search for a pivot, is comprehensive and this is where computations may split,
- the recursive calls may return several branches.

Specification

Input: $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.

Output: ComprehensiveLU$(\mathbf{A}, W)$ returns a list of tuples $[\mathbf{P}_i, \mathbf{Q}_i, \mathbf{L}_i, \mathbf{U}_i, W_i]$, called *branches*, for $1 \leq i \leq e$:

1. the W_i's form a partition of W.

ComprehensiveLU is an adaptation of RecursiveLU to matrices depending on parameters. The main enhancements are:

- the search for a pivot, is comprehensive and this is where computations may split,
- the recursive calls may return several branches.

Specification

Input: $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.

Output: ComprehensiveLU$(\mathbf{A}, W)$ returns a list of tuples $[\mathbf{P}_i, \mathbf{Q}_i, \mathbf{L}_i, \mathbf{U}_i, W_i]$, called *branches*, for $1 \leq i \leq e$:

1. the W_i's form a partition of W.
2. $\mathbf{A} = \mathbf{P}_i \mathbf{L}_i \mathbf{U}_i \mathbf{Q}_i$ holds at every point of W_i.

1 Overview

2 LU Decomposition

3 Constructible Sets

4 Comprehensive LU

5 Comprehensive Pivot Search

6 Experimentation

7 Conclusion

8 References

The term *true path* comes from the fact that, in MAPLE's `Regularchains` library, the constructible set corresponding to the entire affine space $\mathbf{K}^n$ is displayed as **true**.

The term *true path* comes from the fact that, in MAPLE's `Regularchains` library, the constructible set corresponding to the entire affine space $\mathbf{K}^n$ is displayed as **true**.

True Path

We say that the pair $(\mathbf{A}, W)$ has a *true path* whenever there exists an algorithm satisfying **ComprehensiveLU** and returning a single branch when applied to $(\mathbf{A}, W)$.

The term *true path* comes from the fact that, in MAPLE's `Regularchains` library, the constructible set corresponding to the entire affine space $\mathbf{K}^n$ is displayed as **true**.

True Path

We say that the pair $(\mathbf{A}, W)$ has a *true path* whenever there exists an algorithm satisfying **ComprehensiveLU** and returning a single branch when applied to $(\mathbf{A}, W)$.

True path pivot

Given a row index r and a column index c of $\mathbf{A}$, we say that the coefficient $\mathbf{A}[r, c]$ of $\mathbf{A}$ is a *true path pivot* whenever $\mathbf{A}[r, c]$ vanishes nowhere on W.

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.
- Let full be a Boolean parameter wth **false** as default value.

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.
- Let full be a Boolean parameter wth **false** as default value.
- Then ComprehensivePivotSearch(**A**, W, full) returns a list of tuples $[\mathtt{flag}_i, r_i, c_i, W_i]$, for $1 \leq i \leq e$, where $\mathtt{flag}_i$ is a Boolean flag, thus either **true** or **false**, so that

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.
- Let full be a Boolean parameter wth **false** as default value.
- Then ComprehensivePivotSearch(**A**, W, full) returns a list of tuples $[\mathtt{flag}_i, r_i, c_i, W_i]$, for $1 \leq i \leq e$, where $\mathtt{flag}_i$ is a Boolean flag, thus either **true** or **false**, so that
  - if $\mathtt{flag}_i = \mathbf{true}$ holds, then $A[r_i, c_i]$ is a true path pivot for W_i, that is, $A[r_i, c_i]$ does not vanish on W_i,

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.
- Let full be a Boolean parameter wth **false** as default value.
- Then ComprehensivePivotSearch(**A**, `W`, full) returns a list of tuples $[\texttt{flag}_i, r_i, c_i, W_i]$, for $1 \leq i \leq e$, where $\texttt{flag}_i$ is a Boolean flag, thus either **true** or **false**, so that
  - if $\texttt{flag}_i = \textbf{true}$ holds, then $A[r_i, c_i]$ is a true path pivot for W_i, that is, $A[r_i, c_i]$ does not vanish on W_i,
  - if $\texttt{flag}_i = \textbf{false}$ holds, then no pivot for W_i could be found in the first column of $\mathbf{A}$, and

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.
- Let full be a Boolean parameter wth **false** as default value.
- Then ComprehensivePivotSearch(**A**, `W`, full) returns a list of tuples $[\texttt{flag}_i, r_i, c_i, W_i]$, for $1 \leq i \leq e$, where $\texttt{flag}_i$ is a Boolean flag, thus either **true** or **false**, so that
  - if $\texttt{flag}_i = \mathbf{true}$ holds, then $A[r_i, c_i]$ is a true path pivot for W_i, that is, $A[r_i, c_i]$ does not vanish on W_i,
  - if $\texttt{flag}_i = \mathbf{false}$ holds, then no pivot for W_i could be found in the first column of $\mathbf{A}$, and
- $\{W_1, \ldots, W_e\}$ is a partition of W.

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.
- Let full be a Boolean parameter wth **false** as default value.
- Then ComprehensivePivotSearch(**A**, W, full) returns a list of tuples $[\mathtt{flag}_i, r_i, c_i, W_i]$, for $1 \leq i \leq e$, where $\mathtt{flag}_i$ is a Boolean flag, thus either **true** or **false**, so that
  - if $\mathtt{flag}_i = \mathbf{true}$ holds, then $A[r_i, c_i]$ is a true path pivot for W_i, that is, $A[r_i, c_i]$ does not vanish on W_i,
  - if $\mathtt{flag}_i = \mathbf{false}$ holds, then no pivot for W_i could be found in the first column of $\mathbf{A}$, and
- $\{W_1, \ldots, W_e\}$ is a partition of W.
- Moreover, if full is **false** then the search for a pivot or a true path is limited to the first column.

- Let $\mathbf{A}$ be an $\mathbf{m} \times \mathbf{n}$-matrix over $\mathbb{K}[X_1, \ldots, X_v]$ and let W be a constructible set given by polynomials of $\mathbb{K}[X_1, \ldots, X_v]$.
- Let full be a Boolean parameter wth **false** as default value.
- Then ComprehensivePivotSearch(**A**, `W`, full) returns a list of tuples $[\mathtt{flag}_i, r_i, c_i, W_i]$, for $1 \leq i \leq e$, where $\mathtt{flag}_i$ is a Boolean flag, thus either **true** or **false**, so that
  - if $\mathtt{flag}_i = \mathbf{true}$ holds, then $A[r_i, c_i]$ is a true path pivot for W_i, that is, $A[r_i, c_i]$ does not vanish on W_i,
  - if $\mathtt{flag}_i = \mathbf{false}$ holds, then no pivot for W_i could be found in the first column of $\mathbf{A}$, and
- $\{W_1, \ldots, W_e\}$ is a partition of W.
- Moreover, if full is **false** then the search for a pivot or a true path is limited to the first column.
- Otherwise, that is if full is **true**, then the search for a true path is performed in the entire matrix.

1 Overview

2 LU Decomposition

3 Constructible Sets

4 Comprehensive LU

5 Comprehensive Pivot Search

6 Experimentation

7 Conclusion

8 References

We implemented in MAPLE the algorithms, CLU refers to ComprehensiveLU(A,W) with partial pivoting and FLU refers to ComprehensiveLU(A,W, full) with full pivoting.The algorithms output a list of cases along with their corresponding constraints. To illustrate these methods, we selected examples from various papers, and ran these examples over $\mathbb{K}[X_1, \ldots, X_v]$ and an initially empty constructible set W in MAPLE 2024 using an HP Pavilion x360 PC with `Windows` 11.

Comparison between the metrics for Comprehensive LU without full pivoting **CLU** and with full pivoting **FLU**.

	CLU			**FLU**		
Example	Time	Cases	Size	Time	Cases	Size
E_2	0.522	57	27648	0.138	16	6440
E_5	0.861	54	18458	0.533	29	9028
E_7	8.205	326	192658	0.143	9	3527
E_8	1.198	106	43590	0.030	5	1439
E_9	0.183	21	5156	0.010	1	208
E_{13}	0.023	2	1150	0.016	1	596
E_{16}	0.540	5	2255	0.154	5	2255
E_{17}	2.894	34	16143	0.147	5	2251

Comparison between Dimension and Degree of Cases for **CLU** and **FLU**

		Total Degree	
Example	Dimension of Cases	**CLU**	**FLU**
E_2	0	83	15
	1	22	6
E_5	2	44	19
	3	16	9
E_7	4	118	3
	5	60	3
E_{17}	3	28	3
	4	20	2

Example

To visually demonstrate with example E_{13}, the input matrix[5] was

Example

To visually demonstrate with example E_{13}, the input matrix[5] was

$$\mathrm{E}_{13} = \begin{bmatrix} -1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 1 & 0 \\ 0 & ca & 0 & a & 0 & 0 & 0 \\ 0 & 0 & -ca & 0 & -a & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

with **CLU** we have, **P**, **L**, **U**, W_i

$$\begin{bmatrix}1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\end{bmatrix},\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\-1&4&1&1&0&0&0\\0&0&1&0&1&0&0\\0&0&-ca&0&-ac+a&1&0\\0&2ca&0&ac-a&0&0&1\end{bmatrix},$$

$$\begin{bmatrix}-1&1&1&1&1&0&1\\0&\frac{1}{2}&0&\frac{1}{2}&0&1&0\\0&0&1&0&1&0&0\\0&0&0&-1&0&-4&1\\0&0&0&0&-1&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&0&-ca+a\end{bmatrix},\left\{a=0\quad or\left\{c-2=0\right.\right.$$

with **CLU** we have, **P**, **L**, **U**, W_i

$$\begin{bmatrix}1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\end{bmatrix}, \begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\-1&4&1&1&0&0&0\\0&0&1&0&1&0&0\\0&0&-ca&0&-ac+a&1&0\\0&2ca&0&ac-a&0&0&1\end{bmatrix},$$

$$\begin{bmatrix}-1&1&1&1&1&0&1\\0&\frac{1}{2}&0&\frac{1}{2}&0&1&0\\0&0&1&0&1&0&0\\0&0&0&-1&0&-4&1\\0&0&0&0&-1&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&0&-ca+a\end{bmatrix}, \left\{a=0 \quad or \left\{c-2=0\right.\right.$$

and

$$\begin{bmatrix}1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\end{bmatrix}, \begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\-1&4&1&1&0&0&0\\0&0&1&0&1&0&0\\0&2ca&0&ac-a&0&1&0\\0&0&-ca&0&-ac+a&0&1\end{bmatrix},$$

$$\begin{bmatrix}-1&1&1&1&1&0&1\\0&\frac{1}{2}&0&\frac{1}{2}&0&1&0\\0&0&1&0&1&0&0\\0&0&0&-1&0&-4&1\\0&0&0&0&-1&0&0\\0&0&0&0&0&2ca-4a&-ca+a\\0&0&0&0&0&0&1\end{bmatrix}, \begin{cases}a\neq 0\\c-2\neq 0\end{cases}$$

But with **FLU** we have **P**, **Q**, **L**, **U**, W_i

$$\begin{bmatrix}1&0&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\end{bmatrix}, \begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\end{bmatrix},$$

$$\begin{bmatrix}1&0&0&0&0&0&0\\-1&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&\frac{1}{4}&-\frac{1}{4}&1&0&0&0\\0&0&1&0&1&0&0\\0&0&-ca&0&-a&1&0\\0&\frac{ca}{2}&-\frac{ca}{2}&-2ca+4a&-ca+a&-ca+a&1\end{bmatrix},$$

$$\begin{bmatrix}-1&1&1&1&1&0&1\\0&2&1&1&1&0&1\\0&0&1&0&0&0&0\\0&0&0&\frac{1}{4}&-\frac{1}{4}&1&-\frac{1}{4}\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&2ca-4a\end{bmatrix}, \left\{ true \right.$$

1 Overview

2 LU Decomposition

3 Constructible Sets

4 Comprehensive LU

5 Comprehensive Pivot Search

6 Experimentation

7 Conclusion

8 References

Conclusion

Our goal was to provide a procedure which

- comprehensively computes the LU decomposition of parametric matrices with constraints, and
- offers heuristic methods, with negligible overheads, in attempt to minimize the number of branches generated by case discussion.

Our experimental results suggest that our proposed methods achieve our goals in the vast majority of text examples.

1 Overview

2 LU Decomposition

3 Constructible Sets

4 Comprehensive LU

5 Comprehensive Pivot Search

6 Experimentation

7 Conclusion

8 References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and S. Clifford, *Introduction to Algorithms*.
MIT Press and McGraw-Hill, 2001.

[2] R. Corless, M. Moreno Maza, and S. E. Thornton, "Jordan Canonical Form with Parameters from Frobenius Form with Parameters," *Mathematical Aspects of Computer and Information Sciences*, 2017.

[3] M. F. C., *Chaotic Vibrations: An Introduction for Applied Scientists and Engineers*.
Spherical Pendulum, 1987.

[4] P. Aubry, D. Lazard, and M. Moreno Maza, "On the theories of triangular sets," *J. Symbolic Computation*, vol. 28, pp. 105–124, 1999.

[5] S. Dietz, C. Scherer, and W. Huygen, "Linear parameter-varying controller synthesis using matrix sum-of-squares relaxations," *Brazilian Automation Conf*, 2006.