Classical LU Decomposition in ACL2

Carl Kwan

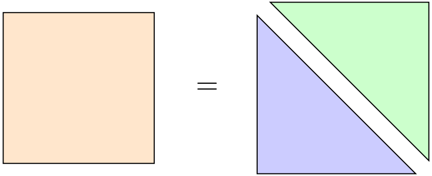
The University of Texas at Austin

2023 November 13

Introduction

LU decomposition:

- Factor a matrix into **L**ower and **U**pper triangular parts
- Fundamental in scientific computing, solves linear systems, many other applications



Motivation:

- Numerical linear algebra used in numerous critical applications
- Very few verification efforts for matrix algorithms, many possible reasons, one major reason is indexing

Today:

- First theorem prover verification of LU decomposition (to our knowledge)
- ACL2 approach to formalization modifies a systematic approach to deriving matrix algorithms via partitioning, can be applied to other classes of algorithms

Indexing Examples

Typical examples of matrix algorithms in literature:

Algorithm 1 $C := C + AB$ (Golub)

> **for** $i = 1 : m$ **do**
>   **for** $j = 1 : n$ **do**
>   **for** $k = 1 : r$ **do**
> $C(i,j) = C(i,j) + A(i,k)B(k,j)$

Above is not too bad to think about, but what about below?

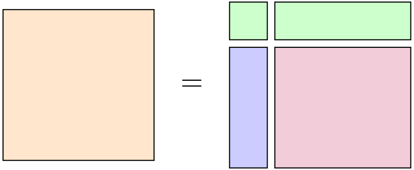
Algorithm 2 $A = LU$ (Stewart)

> **for** $k = 1 : n - 1$ **do**
>   **if** A[k,k] $= 0$ **then** Error
> $A[k+1:n, k] = A[k+1:n, k]/A[k,k]$
> $A[k+1:n, k+1:n] = A[k+1:n, k+1:n] - A[k+1:n, k]A[k, k+1:n]$

Implement? Sure. But what about verification? In ACL2?

LU Decomposition

An LU decomposition factors a matrix A into an upper triangular matrix U and a unit lower triangular matrix L



Want: 0s above diagonal of L, 1s on diagonal of L, 0s below diagonal of U

$$\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) = A = LU = \left(\begin{array}{c|c} 1 & \\ \hline \ell_{21} & L_{22} \end{array}\right) \left(\begin{array}{c|c} \upsilon_{11} & u_{12}^T \\ \hline & U_{22} \end{array}\right)$$

or, equivalently,

$$\alpha_{11} = \upsilon_{11} \; , \qquad a_{21} = \upsilon_{11}\ell_{21} \; , \qquad a_{12}^T = u_{12}^T \; , \qquad A_{22} = \ell_{21} u_{21}^T + L_{22} U_{22} \; .$$

This forces

$$\ell_{21} = a_{21}\alpha_{11}^{-1} \; , \qquad\qquad L_{22}U_{22} = A_{22} - a_{21}\alpha_{11}^{-1}a_{12}^T \; .$$

Reduce LU for A to LU for a smaller matrix $A_{22} - a_{21}\alpha_{11}^{-1}a_{12}^T$. Formalize this.

LU Decomposition in ACL2

Algorithm 3 LU decomposition (recursive)

procedure $\text{LU}(A \in \mathbb{R}^{m\times n})$

Partition $A = \begin{pmatrix} \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{pmatrix}$

if $m = 0$ or $n = 0$ **then return** $(\;)$

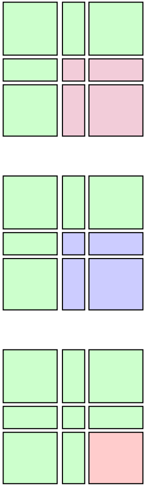
else if $n = 1$ **then return** $\begin{pmatrix} \alpha_{11} \\ a_{21}\alpha_{11}^{-1} \end{pmatrix}$

else if $m = 1$ **then return** A

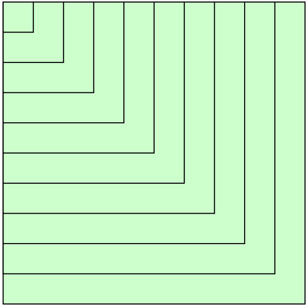
else

$a_{21} := a_{21}\alpha_{11}^{-1}$
$A_{22} := A_{22} - a_{21}a_{12}^T$

return $\begin{pmatrix} \alpha_{11} & a_{12}^T \\ a_{21} & \text{LU}(A_{22}) \end{pmatrix}$



Conditions for Success



Textbook condition for success: Leading principal submatrices of order 1, ..., $n-1$ must be nonsingular

Proof (sketch):

Let $A = \begin{pmatrix} \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{pmatrix}$ and look at $S := A_{22} - a_{21}\alpha_{11}^{-1}a_{12}^T$.

Note if α_{11} nonsingular, then A is nonsingular iff S is nonsingular. Induct.

Upshot: If A is $n \times n$, then S is $(n-1) \times (n-1)$. Reduce the condition for a matrix to be LU decomposable into the same condition for a smaller matrix.

ACL2 approach: Write a function that checks $\alpha_{11} \neq 0$ and recurse on S – this recognizes nonsingular leading principal submatrices. ACL2 automatically inducts according to a scheme suggested by this function and proves LU correctness without any user-provided hints. Induction step is satisfied by the formal derivation.

Conditions for Success

Program 1 ACL2 recognizer for matrices with nonsingular leading principal submatrices and ACL2 theorem for LU correctness

```
(define nonsingular-leading-principal-submatrices-p ((A matrixp))
 :measure (and (row-count A) (col-count A))
 (b* (((unless (matrixp A)) nil)
      ((if (m-emptyp A)) t)
      (alph (car (col-car A)))
      ((if (zerop alph)) nil)
      ((if (or (m-emptyp (row-cdr A))
               (m-emptyp (col-cdr A))))
       t)
      ;; Compute S = A22 - out-*(a21/alph,a12)
      (a21     (col-car (row-cdr A)))
      (a12     (row-car (col-cdr A)))
      (A22     (col-cdr (row-cdr A)))
      (a21/a   (sv* (/ alph) a21))
      (S       (m+ A22 (sm* -1 (out-* a21/a a12)))))
     (nonsingular-leading-principal-submatrices-p S))
 ///
 (defthm lu-correctness
  (b* ((LU (lu A))
       (L  (get-L LU))
       (U  (get-U LU)))
      (implies (and (equal (col-count A) (row-count A))
                    (nonsingular-leading-principal-submatrices-p A))
               (equal (m* L U) A)))))
```

Conclusion

- Casting linear algebra algorithms in terms of partitioned matrices makes verification easier
- Partition, prove the derivation, define the recognizer, and ACL2 handles the rest
- Works for other algorithms, e.g. Cholesky verified
- QR next – LU, Cholesky, and QR form the "three amigos"

Program 2 LU correctness

```
(defthm lu-correctness
 (b* ((LU (lu A))
      (L  (get-L LU))
      (U  (get-U LU)))
     (implies (and (equal (col-count A) (row-count A))
                   (nonsingular-leading-principal-submatrices-p A))
              (equal (m* L U) A)))))
```

Thank you!

References

1. Robert van de Geijn & Margaret Myers (2023): Advanced Linear Algebra: Foundations to Frontiers. Available at `https://www.cs.utexas.edu/users/flame/laff/alaff/frontmatter.html`.
2. Gene H. Golub & Charles F. Van Loan (2013): Matrix Computations - 4th Edition. Johns Hopkins University Press, Philadelphia, PA, doi:10.56021/9781421407944.
3. G. W. Stewart (1998): Matrix Algorithms. Society for Industrial and Applied Mathematics, doi:10.1137/1.9781611971408.
4. Joe Hendrix (2003): Matrices in ACL2. Available at `https://www.cs.utexas.edu/users/moore/acl2/workshop-2003/contrib/hendrix/hendrix.pdf`.