Exactly Solving Linear Systems via the Sparse Exact (SPEX) Factorization Framework

Towards truly exact optimization algorithms

Erick Moreno-Centeno, Adolfo Escobedo (ASU) Christopher Lourenco (USNA), Jinhao Chen(TAMU-CS), Timothy Davis (TAMU-CS)

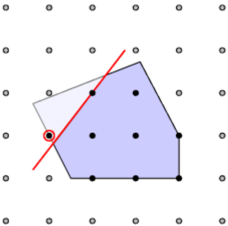
11/24/2020

INDUSTRIAL & SYSTEMS ENGINEERING
TEXAS A&M UNIVERSITY

Applications Requiring Exact LP/MIP

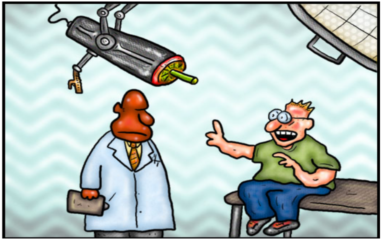
Theoretical

- Feasibility Problems
- Mathematical Proofs
- Mixed Integer Rounding Cuts



Practical

- Defense applications
- Chip design verification
- Healthcare delivery systems
- Numerically unstable problems
- Combinatorial Auctions (Billion$ settled [4])



Roundoff errors in Optimization (LP / IP)

Optimization solvers give certificates of optimality and optimality gaps, don't they?

The Big Deal

Commercial LP solvers **come with limited guarantees** [5].

- Klotz et. al [12, 13] show examples of failures with as few as **2 variables**
- Big-M method $\implies$ ill-conditioned SLEs

Bottom Line

Due to roundoff errors, our current LP and IP algorithms are nothing else but glorified heuristics.

Approaches to Exact Linear Programming

How are LPs solved exactly?

Exact LP

Full precision rational arithmetic LU factorization of promising LP bases.

$\Rightarrow$ Rational LU factorization of sparse SLE

- Worst case: $> 90\%$ of run time [9]

Research Goal

Expedite exact LP solvers by **eliminating** rational LU factorization.

Building from IPGE, these are our most important results

- (Dense) Factorizations Theory:
  - Roundoff Error Free (REF) LU and Cholesky Factorizations
  - REF Forward Substitution
  - REF Backward Substitution
  - REF Factorization (Column Replacement) Updates
- Sparse Exact (SPEX) Factorization Framework:
  - Left-looking LU Factorization**
  - Up-looking Cholesky Factorization**
  - Left-looking Cholesky Factorization**
- **Theory and professional-grade code

A different Gaussian Elimination

- Gaussian Elimination (GE):

$$a_{ij}^{k} = a_{ij}^{k-1} - \frac{a_{kj}^{k-1} a_{ik}^{k-1}}{a_{kk}^{k-1}} = \frac{a_{kk}^{k-1} a_{ij}^{k-1} - a_{kj}^{k-1} a_{ik}^{k-1}}{a_{kk}^{k-1}}$$

- Integer-preserving Gaussian Elimination (IPGE):

$$a_{ij}^{k} = \frac{a_{kk}^{k-1} a_{ij}^{k-1} - a_{kj}^{k-1} a_{ik}^{k-1}}{a_{k-1,k-1}^{k-2}} \tag{1}$$

- Key properties:
  1: All divisions in Equation (1) are exact [1, 6, 16].
  2: The maximum bit-length required to store any IPGE entry is bounded polynomially [2].

Worse Case Size of IPGE Entries

Let $\sigma = \max_{i,j} a^0_{i,j}$. Let β_{max} be the maximum bit-length of any IPGE entry. It can be upper bounded as [2]:

$$\beta_{max} \leq \lceil n \log(\sigma \sqrt{n}) \rceil.$$

Is there a tighter upper bound?

Theorem: Tighter β_{max}

If A is positive definite:

$$\beta_{max} \leq \lceil n \log(\sigma) \rceil$$

If A is sparse:

$$\beta_{max} \leq \lceil n \log(\sigma \sqrt{\gamma}) \rceil,$$

where γ is the smallest of two numbers: number nonzeros in most dense row of A, number nonzeros in most dense column of A.

Dense REF LU and Cholesky Factorizations

Theorem (Existence of the REF LU Factorization)

Any nonsingular square integral matrix A can be factored into the form $A = LDU$ ([7]):

$$\begin{bmatrix} \boldsymbol{a^0_{1,1}} & & & & \\ a^0_{2,1} & \boldsymbol{a^1_{2,2}} & & & \\ a^0_{3,1} & a^1_{3,2} & \boldsymbol{a^2_{3,3}} & & \\ \vdots & \vdots & \vdots & \ddots & \\ a^0_{n,1} & a^1_{n,2} & a^2_{n,3} & \dots & \boldsymbol{a^{n-1}_{n,n}} \end{bmatrix} \boldsymbol{D} \begin{bmatrix} \boldsymbol{a^0_{1,1}} & a^0_{1,2} & a^0_{1,3} & \dots & a^0_{1,n} \\ & \boldsymbol{a^1_{2,2}} & a^1_{2,3} & \dots & a^1_{2,n} \\ & & \boldsymbol{a^2_{3,3}} & \dots & a^2_{3,n} \\ & & & \ddots & \vdots \\ & & & & \boldsymbol{a^{n-1}_{n,n}} \end{bmatrix}$$

where $\boldsymbol{D} = diag\left(\boldsymbol{a^0_{1,1}}\ ,\ \boldsymbol{a^0_{1,1} a^1_{2,2}}\ ,\ \boldsymbol{a^1_{2,2} a^2_{3,3}}\ ,\ \dots\ ,\ \boldsymbol{a^{n-2}_{n-1,n-1} a^{n-1}_{n,n}}\right)^{-1}$

Theorem (Existence of the REF Cholesky Factorization)

Any symmetric positive-definite (SPD) integral matrix A can be factored into the form $(\boldsymbol{L}\sqrt{\boldsymbol{D}})(\boldsymbol{L}\sqrt{\boldsymbol{D}})^T = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L^T}$*:*

REF Forward Substitution - Triangular Solve

Theorem

$L^{(k-1)}D^{(k-1)}\mathbf{x} = A(:,k)$ can be solved exclusively in integer arithmetic as follows.

1. *Initialize $\mathbf{x} = A(:,k)$.*
2. *For $j = 2, \ldots n$, apply the following equation:*

$$x_j = \frac{\rho^i x_j - l_{j,i} x_i}{\rho^{i-1}} \quad \textit{for iteration } i = 1, \ldots, \min(j,k) - 1 \tag{2}$$

Computational complexity

Number of operations on SPD $A \in \mathbb{Z}^{n\times n}$:

- Construction of REF LU $\leq \frac{8n^3}{3}$ ($\frac{2n^3}{3}$ Traditional)
- Construction of REF Cholesky $\leq \frac{4n^3}{3}$ ($\frac{n^3}{3}$ Traditional)
- REF Forward and Backward Substitution $O(n^2)$

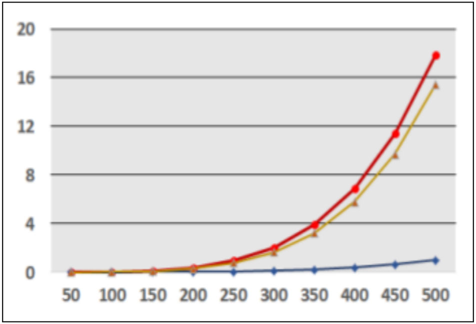
Worst-case computational complexity:

- REF factorizations $= O(n^3[\omega_{\max}\log\omega_{\max}\log\log\omega_{\max}])$ $= O\left(n^4\max(\log^2 n\log\log n, \log^2\sigma\log\log\sigma)\right)$
- REF substitution $= O(n^2[\omega_{\max}\log\omega_{\max}\log\log\omega_{\max}])$ $= O\left(n^3\max(\log^2 n\log\log n, \log^2\sigma\log\log\sigma)\right)$

Dense REF LU vs Rational Doolittle/Crout LU

Compare REF LU to a rational version of Crout and Doolittle LU factorization.

REF LU vs QLU (n,sec)



Sparsity Considerations

- **Theoretical:**
  - Right-looking vs Left-looking LU
  - Finding location of nonzeros in L and U
  - Skipping operations on entries that are zero
- **Computational:**
  - Column Preordering
  - Pivoting Schemes

Existence of Left Looking REF LU Factorization

Theorem: REF LU can be obtained in left-looking fashion by solving the SLE

$$L^{(k-1)}D^{(k-1)}\mathbf{x} = A(:,k),$$

formally:

$$L^{(k-1)}D^{(k-1)}\mathbf{x} = \left[\begin{array}{cccc|c} \frac{1}{\rho^0} & 0 & 0 & 0 & \\ \frac{l_{21}}{\rho^1} & \frac{1}{\rho^1} & 0 & 0 & \mathbf{0} \\ \vdots & & \ddots & & \\ \frac{l_{k-1,1}}{\rho^1} & \dots & \dots & \frac{1}{\rho^{k-2}} & \\ \hline \frac{l_{k1}}{\rho^1} & \dots & \dots & \frac{l_{k,k-1}}{\rho^{k-2}\rho^{k-1}} & \\ \vdots & & & \vdots & \frac{1}{\rho^{k-1}}\mathbf{I} \\ \frac{l_{n1}}{\rho^1} & \dots & \dots & \frac{l_{n,k-1}}{\rho^{k-2}\rho^{k-1}} & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1k} \\ a_{2k} \\ \vdots \\ a_{nk} \end{bmatrix} \tag{3}$$

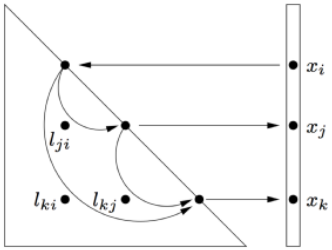
where $\mathbf{x}$ gives the kth column of the REF L and U matrices.

- With infinite precision, solving this system yields the correct $\mathbf{x}$.
- Some entries are fractional $\Rightarrow$ Would induce roundoff error!

Symbolic Analysis: Determine the Nonzeros in $\mathbf{x}$

1. a^0_{jk} is nonzero.

$$x_j = a^0_{jk}$$
$$x_j = \frac{\rho^i x_j - l_{ji} x_i}{\rho^{i-1}} \quad \text{for } i < \min(j,k) - 1$$



2. $a^0_{jk} = 0$ but $\exists i : i < j$ and $x_i \neq 0, l_{ji} \neq 0$.

$$x_j = a^0_{jk}$$
$$x_j = \frac{\rho^i x_j - l_{ji} x_i}{\rho^{i-1}} \quad \text{for } i < \min(j,k) - 1$$

These two conditions are identical to traditional Gaussian elimination.

Can find the nonzero pattern of **x**, denoted $\mathcal{X}$, via a graph traversal algorithm [8]:

$$\mathcal{X} = Reach_{G_L}(A(:,k))$$

Numeric Phase: Skipping Operations

$|\mathcal{X}| <<< n \implies$ Most IPGE operations involve zeros.

GE	**IPGE**
$a_{ij}^{k} = a_{ij}^{k-1} - \frac{a_{ik}^{k-1} a_{kj}^{k-1}}{\rho^{k}}$	$a_{ij}^{k} = \frac{\rho^{k} a_{ij}^{k-1} - a_{ik}^{k-1} a_{kj}^{k-1}}{\rho^{k-1}}$
$a_{ij}^{k} = a_{ij}^{k-1}$	$a_{ij}^{k} = \frac{\rho^{k} a_{ij}^{k-1}}{\rho^{k-1}}$

- Suppose a sequence of iterations $\{l, \ldots, k\}$ exist where this occurs.

$$a_{ij}^{k} = a_{ij}^{l-1} \frac{\rho^{l} \rho^{l+1} \cdots \rho^{k}}{\rho^{l-1} \rho^{l} \cdots \rho^{k-1}} \tag{4}$$

- Lee and Saunders [14]: Using a **"history" matrix**, this expression can be condensed into a single **"History" update**:

$$a_{ij}^{k} = \frac{a_{ij}^{l-1} \rho^{k}}{\rho^{l-1}}$$

Impact of SLIP LU

SLIP LU: Exactly solves sparse SLEs exclusively in integer-arithmetic

Theorem

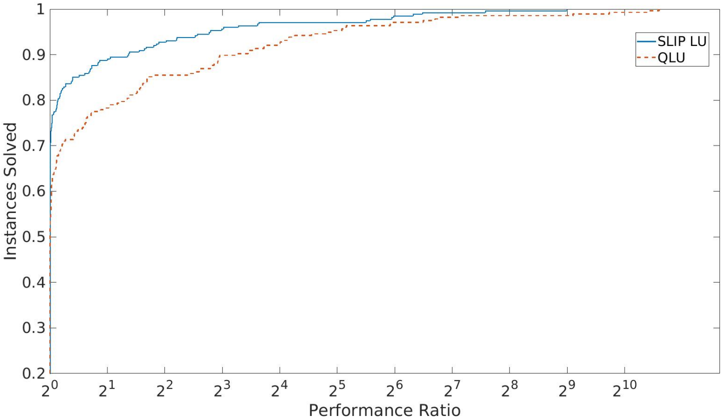
SLIP LU exactly solves the sparse linear system $A\mathbf{x} = \mathbf{b}$ in time proportional to arithmetic work.

$$\mathcal{O}(I(\beta_{max} \log \beta_{max} \log\log \beta_{max})$$

SLIP LU is the **only exact factorization** with this asymptotically efficient bound.

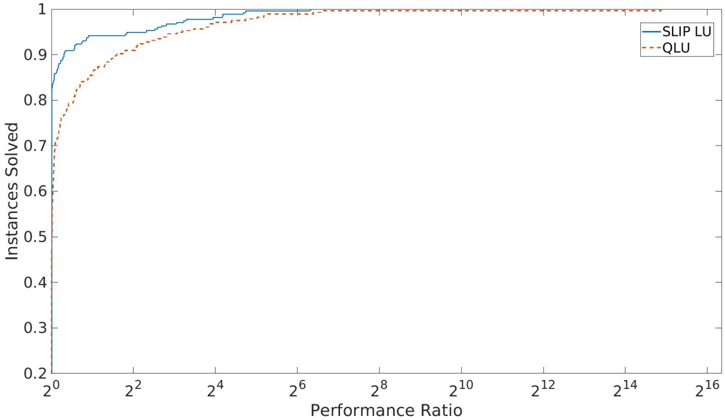
SLIP LU dominates QLU in factorization time (276 real world LP bases)

	AVG	GMean	StDev	Better
SLIP LU	145.64	0.14	865.38	164
QLU	888.47	0.21	5343.53	112



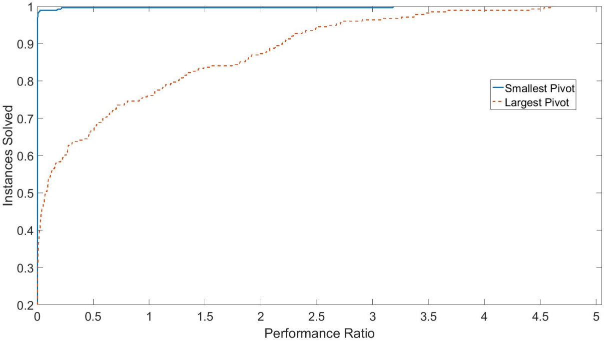
SLIP LU dominates QLU in solution time (276 real world LP bases)

	AVG	GMean	StDev	Better
SLIP LU	2.01	0.02	25.86	208
QLU	17.19	0.08	121.19	68



Pivot Selection Schemes

- Traditional left-looking approach: Select the largest pivot
  - Maintains numerical stability
- IPGE Approach: "no clear choice is evident" between largest and smallest pivot on sparse matrices (Lee and Saunders [14])
- Our approach: select the **smallest** pivot at each iteration



Benchmark MATLAB Backslash' accuracy on 276 LP Bases

Across the set of 276 LP bases:

Relative Error Threshold	Percentage of Instances
$< 10^{-12}$	95.65%
$< 10^{-6}$	96.74%
$< 10^{-2}$	97.10%

- Completely incorrect for $\approx 3\%$ of instances

Current Status of SLIP LU

The SLIP LU software package:

- Submitted to ACM TOMs
- **Commercial quality software package**
- Will be distributed within SuiteSparse to all Linux Distros

SLIP LU is publically available at:

- `https://github.com/clouren/SLIP_LU`
- Distributed as a component of SuiteSparse [3] `www.suitesparse.com`

Sparse Integer-Preserving Cholesky Factorizations

Up-looking and Left-looking Integer-Preserving Cholesky Factorizations.

- Works on **ALL SPD matrices**
- Form $A = (L\sqrt{D})(L\sqrt{D})^T = LDL^T$
- Differ from each other: Way they compute L
- Similar to SLIP LU by using REF triangular solve
- Differ from SLIP LU in:
  - More efficient symbolic analysis (elimination tree)
  - Exploits the symmetry of A to require only $1/2$ number of operations.

Computational Complexities

Theorem

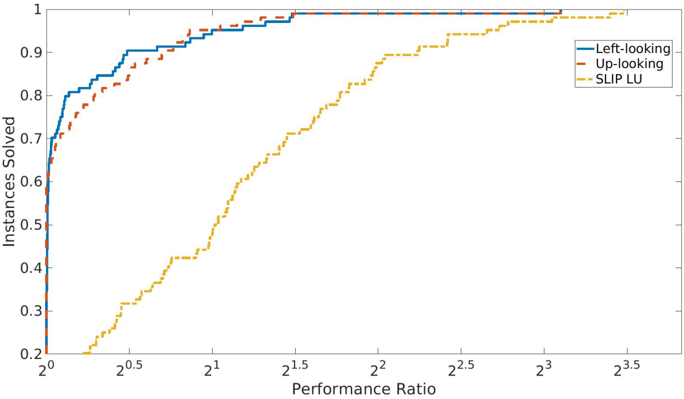
The Up-looking and Left-looking integer-preserving Cholesky factorizations exactly solves the SPD sparse linear system $A\mathbf{x} = \mathbf{b}$ in time proportional to arithmetic work.

$$\mathcal{O}(I_C(\beta_{max} \log \beta_{max} \log \log \beta_{max})$$

Asymptotically efficient & 1/2 operations of SLIP LU

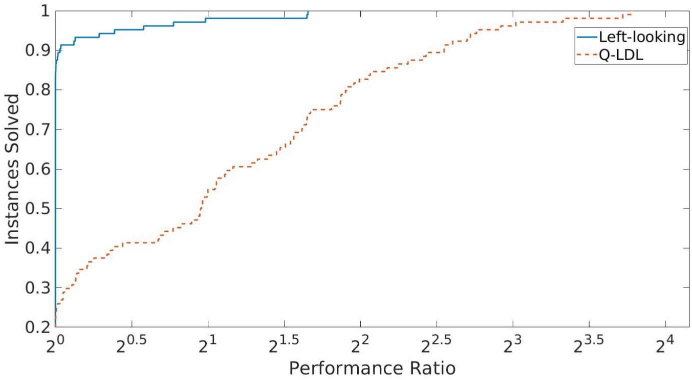
IP Cholesky Dominates SLIP LU on 103 real-world SPD matrices

	AVG	GM
Left-Looking	5694.71	6.85
Up-Looking	5671.15	6.94
SLIP LU	9676.05	12.76



IP Cholesky Dominates Q-LDL on 103 real-world SPD matrices

	AVG	GM
Left-Looking	5694.71	6.85
Q-LDL	11618.63	15.24



Benchmarking MATLAB CHOLMOD's accuracy

Cholesky factorization is generally considered stable[15, 18]

- May fail for highly ill-conditioned instances [10, 11].
- It is unknown how often Cholesky fails

Relative Error Threshold	Percentage of Instances
$\leq 10^{-12}$	70.87%
$\leq 10^{-6}$	93.20%
$\leq 10^{-2}$	98.06%

- Completely incorrect for $\approx 2\%$ of instances
- Surprisingly less accurate in general than unsymmetric LU.

Traditional Approaches For Column Pre-ordering

Total work performed in floating point factorizations is a function of:

- The number & structure of nonzeros in A
- The number of new nonzeros induced in L and U (i.e., fill-in)

Total work performed in integer-preserving factorizations is a function of:

- The number, structure, and bit-lengths of nonzeros in A
- The number and bit-lengths of new nonzeros induced in L and U (i.e., fill-in)

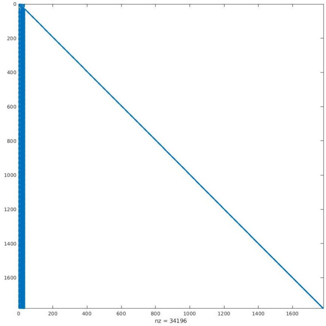
Conjecture

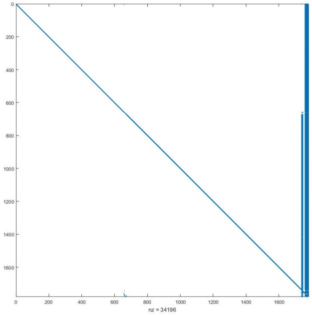
It is NP-Hard to find matrices P and Q which minimize work in integer-preserving factorization.

Idea of New Pre-ordering Algorithm

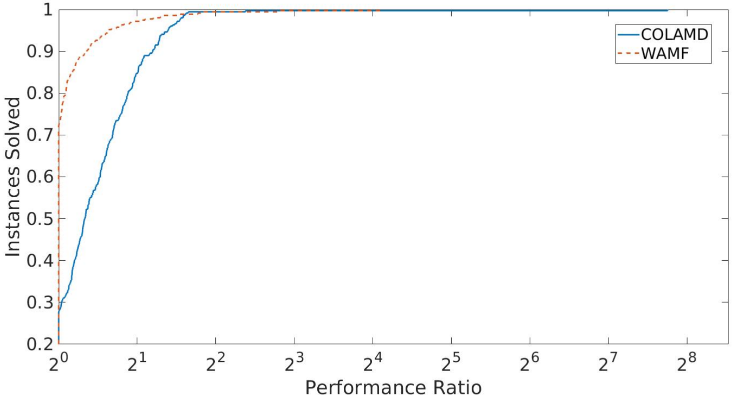
Goal: balance sparsity and bit-length

- Nodes eliminated based on a combination of sparsity and entry size criterion
  - Sparsity: Based on AMF fill-in estimates [17]
  - Bit-length: Based on symbolic IPGE
  - Referred to as **W**eighted **A**pproximate **M**inimum **F**ill (WAMF)
- The sequence of eliminated nodes is the columns for the matrix Q.

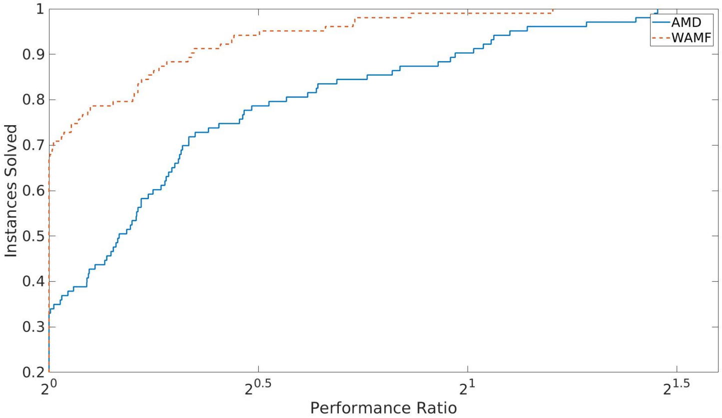




WAMF dominates COLAMD on Unsymmetric matrices



WAMF dominates COLAMD on SPD matrices



All Done!

Thank you!

Questions?

[1] Erwin H Bareiss. Sylvester's identity and multistep integer-preserving gaussian elimination. *Mathematics of computation*, 22(103):565–578, 1968.

[2] Erwin H Bareiss. Computational solutions of matrix problems over an integral domain. *IMA Journal of Applied Mathematics*, 10(1):68–104, 1972.

[3] Tim Davis, WW Hager, and IS Duff. Suitesparse. *htt p://faculty. cse. tamu. edu/davis/suitesparse. html*, 2014.

[4] Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309, 2003.

[5] Marcel Dhiflaoui, Stefan Funke, Carsten Kwappik, Kurt Mehlhorn, Michael Seel, Elmar Schömer, Ralph Schulte, and Dennis Weber. Certifying and repairing solutions to large lps how good are lp-solvers? In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 255–256. Society for Industrial and Applied Mathematics, 2003.

[6] Jack Edmonds. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards Sect. B*, 71:241–245, 1967.

[7] Adolfo R Escobedo and Erick Moreno-Centeno. Roundoff-error-free algorithms for solving linear systems via cholesky and lu factorizations. *INFORMS Journal on Computing*, 27(4):677–689, 2015.

[8] John R Gilbert and Tim Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM Journal on Scientific and Statistical Computing*, 9(5):862–874, 1988.

[9] Ambros M Gleixner. Exact and fast algorithms for mixed-integer nonlinear programming. 2015.

[10] Nicholas J Higham. The accuracy of solutions to triangular systems. *SIAM Journal on Numerical Analysis*, 26(5):1252–1265, 1989.

[11] Nicholas J Higham. *Accuracy and stability of numerical algorithms*, volume 80. Siam, 2002.

[12] Ed Klotz. Identification, assessment, and correction of ill-conditioning and numerical instability in linear and integer programs. In *Bridging Data and Decisions*, pages 54–108. INFORMS, 2014.

[13] Ed Klotz and Alexandra M Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1-2):18–32, 2013.

[14] Hong R Lee and B David Saunders. Fraction free gaussian elimination for sparse matrices. *Journal of symbolic computation*, 19 (5):393–402, 1995.

[15] Jean Meinguet. Refined error analyses of cholesky factorization. *SIAM journal on numerical analysis*, 20(6):1243–1250, 1983.

[16] René M Montante-Pardo and Marco A Méndez-Cavazos. Un método númerico para cálculo matricial. *Revista Técnico-Científica de Divulgación*, 2:1–24, 1977.

[17] Edward Rothberg and Stanley C Eisenstat. Node selection strategies for bottom-up sparse matrix ordering. *SIAM Journal on Matrix Analysis and Applications*, 19(3):682–695, 1998.

[18] James H Wilkinson. A priori error analysis of algebraic processes. In *Intern. Congress Math*, volume 19, pages 629–639, 1968.