Scientific Computing: Dense Linear Systems

Aleksandar Donev
Courant Institute, NYU[1]
donev@courant.nyu.edu

[1]Course MATH-GA.2043 or CSCI-GA.2112, Spring 2012

February 9th, 2012

Outline

1. Linear Algebra Background
2. Conditioning of linear systems
3. Gauss elimination and LU factorization
4. Conclusions

Linear Spaces

- A **vector space** $\mathcal{V}$ is a set of elements called **vectors** $\mathbf{x} \in \mathcal{V}$ that may be multiplied by a **scalar** c and added, e.g.,

$$\mathbf{z} = \alpha\mathbf{x} + \beta\mathbf{y}$$

- I will denote scalars with lowercase letters and vectors with lowercase bold letters.
- Prominent examples of vector spaces are $\mathbb{R}^n$ (or more generally $\mathbb{C}^n$), but there are many others, for example, the set of polynomials in x.
- A **subspace** $\mathcal{V}' \subseteq \mathcal{V}$ of a vector space is a subset such that sums and multiples of elements of $\mathcal{V}'$ remain in $\mathcal{V}'$ (i.e., it is closed).
- An example is the set of vectors in $x \in \mathbb{R}^3$ such that $x_3 = 0$.

Image Space

- Consider a set of n vectors $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n \in \mathbb{R}^m$ and form a **matrix** by putting these vectors as columns

$$\mathbf{A} = [\mathbf{a}_1 \,|\, \mathbf{a}_2 \,|\, \cdots \,|\, \mathbf{a}_m] \in \mathbb{R}^{m,n}.$$

- I will denote matrices with bold capital letters, and sometimes write $\mathbf{A} = [m, n]$ to indicate dimensions.
- The **matrix-vector product** is defined as a **linear combination** of the columns:

$$\mathbf{b} = \mathbf{A}\mathbf{x} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n \in \mathbb{R}^m.$$

- The **image** $\text{im}(\mathbf{A})$ or **range** $\text{range}(\mathbf{A})$ of a matrix is the subspace of all linear combinations of its columns, i.e., the set of all $\mathbf{b}'s$.
  It is also sometimes called the **column space** of the matrix.

Dimension

- The set of vectors $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n$ are **linearly independent** or form a **basis** for $\mathbb{R}^m$ if $\mathbf{b} = \mathbf{A}\mathbf{x} = \mathbf{0}$ implies that $\mathbf{x} = \mathbf{0}$.
- The **dimension** $r = \dim\mathcal{V}$ of a vector (sub)space $\mathcal{V}$ is the number of elements in a basis. This is a property of $\mathcal{V}$ itself and *not* of the basis, for example,
$$\dim \mathbb{R}^n = n$$
- Given a basis $\mathbf{A}$ for a vector space $\mathcal{V}$ of dimension n, every vector of $\mathbf{b} \in \mathcal{V}$ can be uniquely represented as the vector of coefficients $\mathbf{x}$ in that particular basis,
$$\mathbf{b} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n.$$
- A simple and common basis for $\mathbb{R}^n$ is $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$, where $\mathbf{e}_k$ has all components zero except for a single 1 in position k.
With this choice of basis the coefficients are simply the entries in the vector, $\mathbf{b} \equiv \mathbf{x}$.

Kernel Space

- The dimension of the column space of a matrix is called the **rank** of the matrix $\mathbf{A} \in \mathbb{R}^{m,n}$,

$$r = \text{rank}\,\mathbf{A} \leq \min(m, n).$$

- If $r = \min(m, n)$ then the matrix is of **full rank**.
- The **nullspace** null($\mathbf{A}$) or **kernel** ker($\mathbf{A}$) of a matrix $\mathbf{A}$ is the subspace of vectors $\mathbf{x}$ for which

$$\mathbf{A}\mathbf{x} = \mathbf{0}.$$

- The dimension of the nullspace is called the **nullity** of the matrix.
- For a basis $\mathbf{A}$ the nullspace is null($\mathbf{A}$) $= \{\mathbf{0}\}$ and the nullity is zero.

Orthogonal Spaces

- An inner-product space is a vector space together with an **inner or dot product**, which must satisfy some properties.
- The standard dot-product in $\mathbb{R}^n$ is denoted with several different notations:

$$\mathbf{x} \cdot \mathbf{y} = (\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{n} x_i y_i.$$

- For $\mathbb{C}^n$ we need to add complex conjugates (here $\star$ denotes a complex conjugate transpose, or **adjoint**),

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^\star \mathbf{y} = \sum_{i=1}^{n} \bar{x}_i y_i.$$

- Two vectors $\mathbf{x}$ and $\mathbf{y}$ are **orthogonal** if $\mathbf{x} \cdot \mathbf{y} = 0$.

Part I of Fundamental Theorem

- One of the most important theorems in linear algebra is that the sum of rank and nullity is equal to the number of columns: For $\mathbf{A} \in \mathbb{R}^{m,n}$

$$\operatorname{rank} \mathbf{A} + \operatorname{nullity} \mathbf{A} = n.$$

- In addition to the range and kernel spaces of a matrix, two more important vector subspaces for a given matrix $\mathbf{A}$ are the:
  - **Row space** or **coimage** of a matrix is the column (image) space of its transpose, $\operatorname{im} \mathbf{A}^T$.
    Its dimension is also equal to the the rank.
  - **Left nullspace** or **cokernel** of a matrix is the nullspace or kernel of its transpose, $\ker \mathbf{A}^T$.

Part II of Fundamental Theorem

- The **orthogonal complement** $\mathcal{V}^\perp$ or orthogonal subspace of a subspace $\mathcal{V}$ is the set of all vectors that are orthogonal to every vector in $\mathcal{V}$.
- Let $\mathcal{V}$ be the set of vectors in $x \in \mathbb{R}^3$ such that $x_3 = 0$. Then $\mathcal{V}^\perp$ is the set of all vectors with $x_1 = x_2 = 0$.
- Second fundamental theorem in linear algebra:

$$\text{im}\, \mathbf{A}^T = (\ker \mathbf{A})^\perp$$

$$\ker \mathbf{A}^T = (\text{im}\, \mathbf{A})^\perp$$

Linear Transformation

- A function $L : \mathcal{V} \to \mathcal{W}$ mapping from a vector space $\mathcal{V}$ to a vector space $\mathcal{W}$ is a **linear function** or a **linear transformation** if

$$L(\alpha \mathbf{v}) = \alpha L(\mathbf{v}) \text{ and } L(\mathbf{v}_1 + \mathbf{v}_2) = L(\mathbf{v}_1) + L(\mathbf{v}_2).$$

- Any linear transformation L can be represented as a multiplication by a matrix $\mathbf{L}$

$$L(\mathbf{v}) = \mathbf{L}\mathbf{v}.$$

- For the common bases of $\mathcal{V} = \mathbb{R}^n$ and $\mathcal{W} = \mathbb{R}^m$, the product $\mathbf{w} = \mathbf{L}\mathbf{v}$ is simply the usual **matix-vector product**,

$$w_i = \sum_{k=1}^{n} L_{ik} v_k,$$

which is simply the dot-product between the i-th row of the matrix and the vector $\mathbf{v}$.

Matrix algebra

$$w_i = (\mathbf{Lv})_i = \sum_{k=1}^{n} L_{ik} v_k$$

- The composition of two linear transformations $\mathbf{A} = [m, p]$ and $\mathbf{B} = [p, n]$ is a **matrix-matrix product** $\mathbf{C} = \mathbf{AB} = [m, n]$:

$$\mathbf{z} = \mathbf{A}(\mathbf{Bx}) = \mathbf{Ay} = (\mathbf{AB})\,\mathbf{x}$$

$$z_i = \sum_{k=1}^{n} A_{ik} y_k = \sum_{k=1}^{p} A_{ik} \sum_{j=1}^{n} B_{kj} x_j = \sum_{j=1}^{n} \left(\sum_{k=1}^{p} A_{ik} B_{kj} \right) x_j = \sum_{j=1}^{n} C_{ij} x_j$$

$$C_{ij} = \sum_{k=1}^{p} A_{Ik} B_{kj}$$

- Matrix-matrix multiplication is **not commutative**, $\mathbf{AB} \neq \mathbf{BA}$ in general.

The Matrix Inverse

- A square matrix $\mathbf{A} = [n, n]$ is **invertible or nonsingular** if there exists a **matrix inverse** $\mathbf{A}^{-1} = \mathbf{B} = [n, n]$ such that:

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I},$$

  where $\mathbf{I}$ is the identity matrix (ones along diagonal, all the rest zeros).
- The following statements are equivalent for $\mathbf{A} \in \mathbb{R}^{n,n}$:
  - $\mathbf{A}$ is **invertible**.
  - $\mathbf{A}$ is **full-rank**, rank $\mathbf{A} = n$.
  - The columns and also the rows are linearly independent and form a **basis** for $\mathbb{R}^n$.
  - The **determinant** is nonzero, $\det \mathbf{A} \neq 0$.
  - Zero is not an eigenvalue of $\mathbf{A}$.

Matrix Algebra

- Matrix-vector multiplication is just a special case of matrix-matrix multiplication. Note $\mathbf{x}^T\mathbf{y}$ is a scalar (dot product).

$$\mathbf{C}(\mathbf{A}+\mathbf{B}) = \mathbf{CA}+\mathbf{CB} \text{ and } \mathbf{ABC} = (\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$$

$$\left(\mathbf{A}^T\right)^T = \mathbf{A} \text{ and } (\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$$

$$\left(\mathbf{A}^{-1}\right)^{-1} = \mathbf{A} \text{ and } (\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \text{ and } \left(\mathbf{A}^T\right)^{-1} = \left(\mathbf{A}^{-1}\right)^T$$

- Instead of **matrix division**, think of multiplication by an inverse:

$$\mathbf{AB} = \mathbf{C} \quad \Rightarrow \quad \left(\mathbf{A}^{-1}\mathbf{A}\right)\mathbf{B} = \mathbf{A}^{-1}\mathbf{C} \quad \Rightarrow \quad \begin{cases} \mathbf{B} &= \mathbf{A}^{-1}\mathbf{C} \\ \mathbf{A} &= \mathbf{CB}^{-1} \end{cases}$$

Vector norms

- Norms are the abstraction for the notion of a length or **magnitude**.
- For a vector $\mathbf{x} \in \mathbb{R}^n$, the p-norm is

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

and special cases of interest are:

1. The 1-norm (L^1 norm or Manhattan distance), $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$
2. The 2-norm (L^2 norm, **Euclidian distance**), $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{\sum_{i=1}^{n} |x_i|^2}$
3. The ∞-norm (L^∞ or maximum norm), $\|\mathbf{x}\|_\infty = \max_{1 \le i \le n} |x_i|$

1. Note that all of these norms are inter-related in a finite-dimensional setting.

Matrix norms

- Matrix norm **induced** by a given vector norm:

$$\|\mathbf{A}\| = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} \quad \Rightarrow \|\mathbf{Ax}\| \leq \|\mathbf{A}\| \, \|\mathbf{x}\|$$

- The last bound holds for matrices as well, $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$.
- Special cases of interest are:
  1. The 1-norm or **column sum norm**, $\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$
  2. The ∞-norm or **row sum norm**, $\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$
  3. The 2-norm or **spectral norm**, $\|\mathbf{A}\|_2 = \sigma_1$ (largest singular value)
  4. The Euclidian or **Frobenius norm**, $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$
     (note this is not an induced norm)

Matrices and linear systems

- It is said that 70% or more of applied mathematics research involves solving systems of m linear equations for n unknowns:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, \cdots, m.$$

- Linear systems arise directly from **discrete models**, e.g., traffic flow in a city. Or, they may come through representing or more abstract **linear operators** in some finite basis (representation).
  Common abstraction:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

- Special case: Square invertible matrices, $m = n$, $\det \mathbf{A} \neq 0$:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.$$

- The goal: Calculate solution $\mathbf{x}$ given data $\mathbf{A}, \mathbf{b}$ in the most numerically stable and also efficient way.

Stability analysis: rhs perturbations

Perturbations on right hand side (rhs) only:

$$\mathbf{A}(\mathbf{x}+\delta\mathbf{x}) = \mathbf{b}+\delta\mathbf{b} \quad \Rightarrow \mathbf{b}+\mathbf{A}\delta\mathbf{x} = \mathbf{b}+\delta\mathbf{b}$$

$$\delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{b} \quad \Rightarrow \|\delta\mathbf{x}\| \le \left\|\mathbf{A}^{-1}\right\| \|\delta\mathbf{b}\|$$

Using the bounds

$$\|\mathbf{b}\| \le \|\mathbf{A}\| \|\mathbf{x}\| \quad \Rightarrow \|\mathbf{x}\| \ge \|\mathbf{b}\| / \|\mathbf{A}\|$$

the relative error in the solution can be bounded by

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \frac{\left\|\mathbf{A}^{-1}\right\| \|\delta\mathbf{b}\|}{\|\mathbf{x}\|} \le \frac{\left\|\mathbf{A}^{-1}\right\| \|\delta\mathbf{b}\|}{\|\mathbf{b}\| / \|\mathbf{A}\|} = \kappa(\mathbf{A}) \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

where the **conditioning number** $\kappa(\mathbf{A})$ depends on the matrix norm used:

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\| \ge 1.$$

Stability analysis: matrix perturbations

- Perturbations of the matrix only:

$$(\mathbf{A}+\delta\mathbf{A})(\mathbf{x}+\delta\mathbf{x})=\mathbf{b} \quad \Rightarrow \delta\mathbf{x}=-\mathbf{A}^{-1}(\delta\mathbf{A})(\mathbf{x}+\delta\mathbf{x})$$

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}+\delta\mathbf{x}\|}\leq\|\mathbf{A}^{-1}\|\,\|\delta\mathbf{A}\|=\kappa(\mathbf{A})\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}.$$

- Conclusion: The **worst-case conditioning** of the linear system is determined by

$$\kappa(\mathbf{A})=\|\mathbf{A}\|\,\|\mathbf{A}^{-1}\|\geq 1$$

- Solving an ill-conditioned system, $\kappa(\mathbf{A})\gg 1$, should only be done if something is known about $\mathbf{x}$ or $\mathbf{b}$ (e.g., smooth solutions in PDEs).
- The conditioning number can only be **estimated** in practice since $\mathbf{A}^{-1}$ is not available (see MATLAB's *rcond* function).

Mixed perturbations

- Now consider general perturbations of the data:

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$$

- The full derivation, not given here, estimates the uncertainty or perturbation in the solution:

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\kappa(\mathbf{A})}{1 - \kappa(\mathbf{A})\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}} \left(\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} \right)$$

- Best possible error with rounding unit u $u \approx 10^{-16}$:

$$\frac{\|\delta\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \lesssim 2u\kappa(\mathbf{A}),$$

  which is close to what the MATLAB linear solvers achieve.
- It certainly makes no sense to try to solve systems with $\kappa(\mathbf{A}) > 10^{15}$!

GEM: Eliminating x_1

Step 1:

$$Ax = b$$

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix}$$

Multiply FIRST row by

$\leftarrow l_{21} = \dfrac{a_{21}^{(1)}}{a_{11}^{(1)}}$

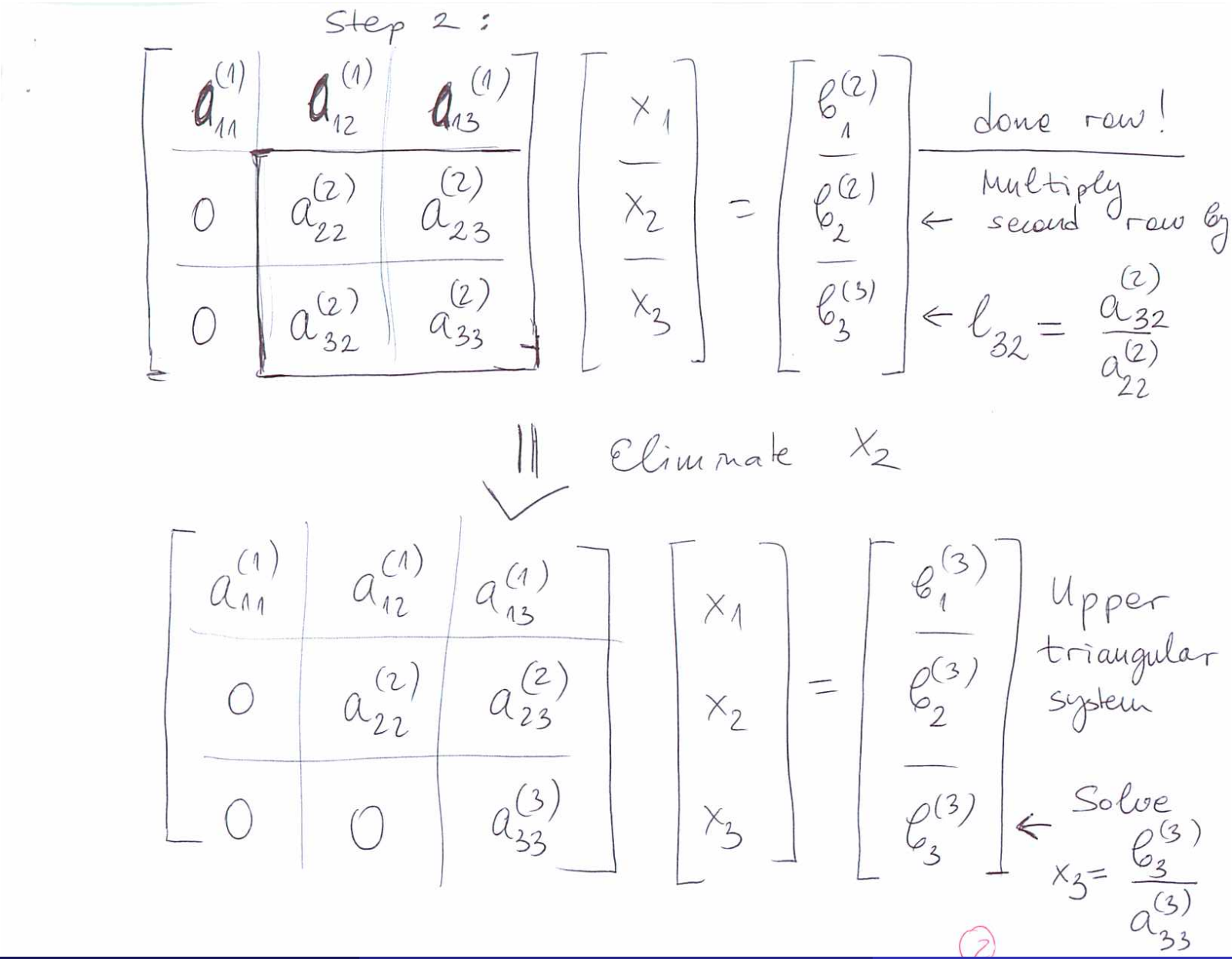
$\leftarrow l_{31} = \dfrac{a_{31}^{(1)}}{a_{11}^{(1)}}$

$\Downarrow$ Eliminate x_1

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 = a_{21}^{(1)} - l_{21} \cdot a_{11} & a_{22}^{(1)} - l_{21} \cdot a_{12}^{(1)} & a_{23}^{(1)} - l_{21} \cdot a_{13}^{(1)} \\ 0 & a_{32}^{(1)} - l_{31} \cdot a_{12}^{(1)} & a_{33}^{(1)} - l_{31} \cdot a_{13}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - l_{21} \cdot b_1 \\ b_3 - l_{31} \cdot b_1 \end{bmatrix}$$

(1)

GEM: Eliminating x_2



GEM: Backward substitution

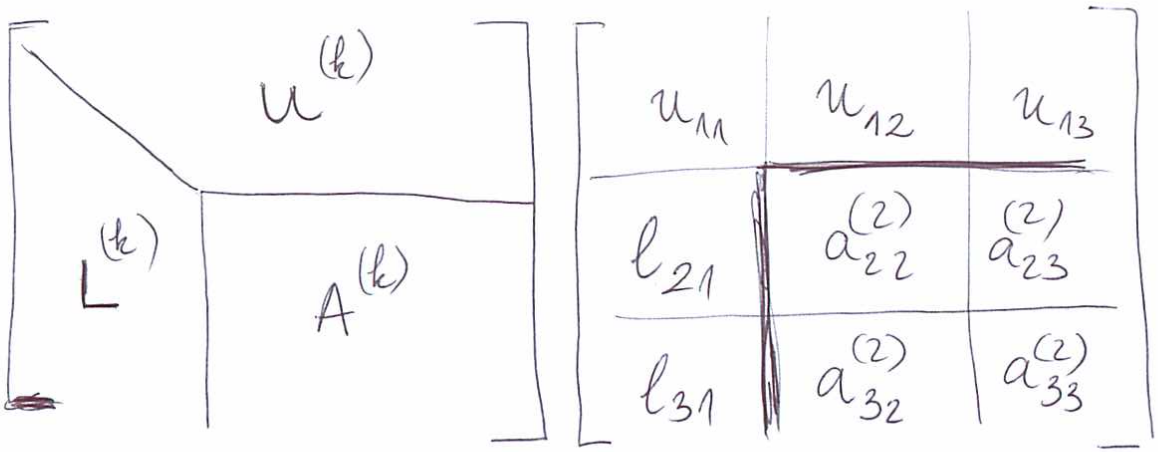
Eliminate x_3 entirely $\rightarrow$

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} \\ 0 & a_{22}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1^{(3)} - a_{13}^{(1)} x_3 \\ b_2^{(3)} - a_{23}^{(2)} x_3 \end{bmatrix} = \tilde{b}$$

solve for $x_2 = \dfrac{\tilde{b}}{a_{22}^{(2)}}$, then x_1, and done!

IDEA: Store the multipliers in the lower triangle of A:

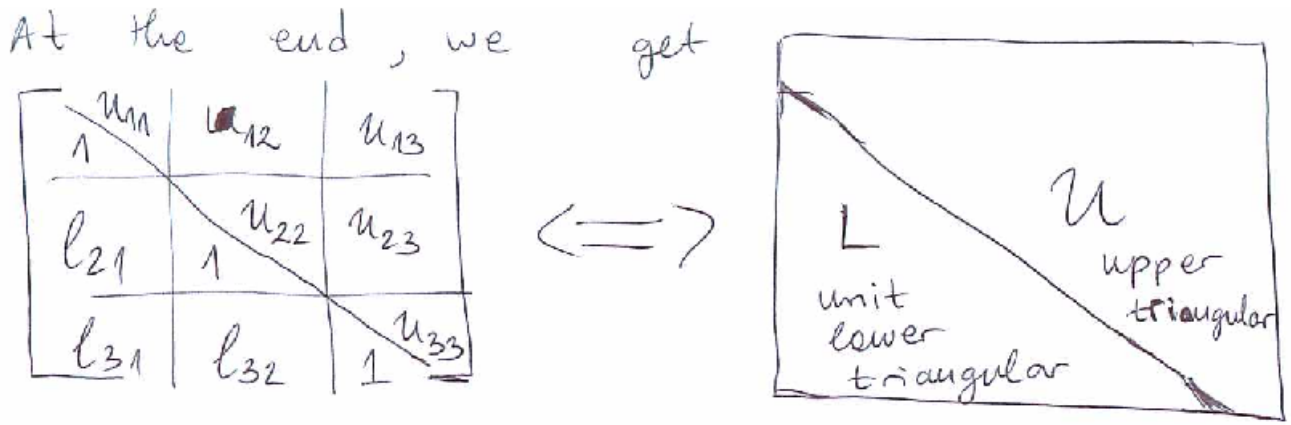
Matrix at Step k:



Example step 2

3

GEM as an LU factorization tool



- We have actually **factorized A** as

$$\mathbf{A} = \mathbf{LU},$$

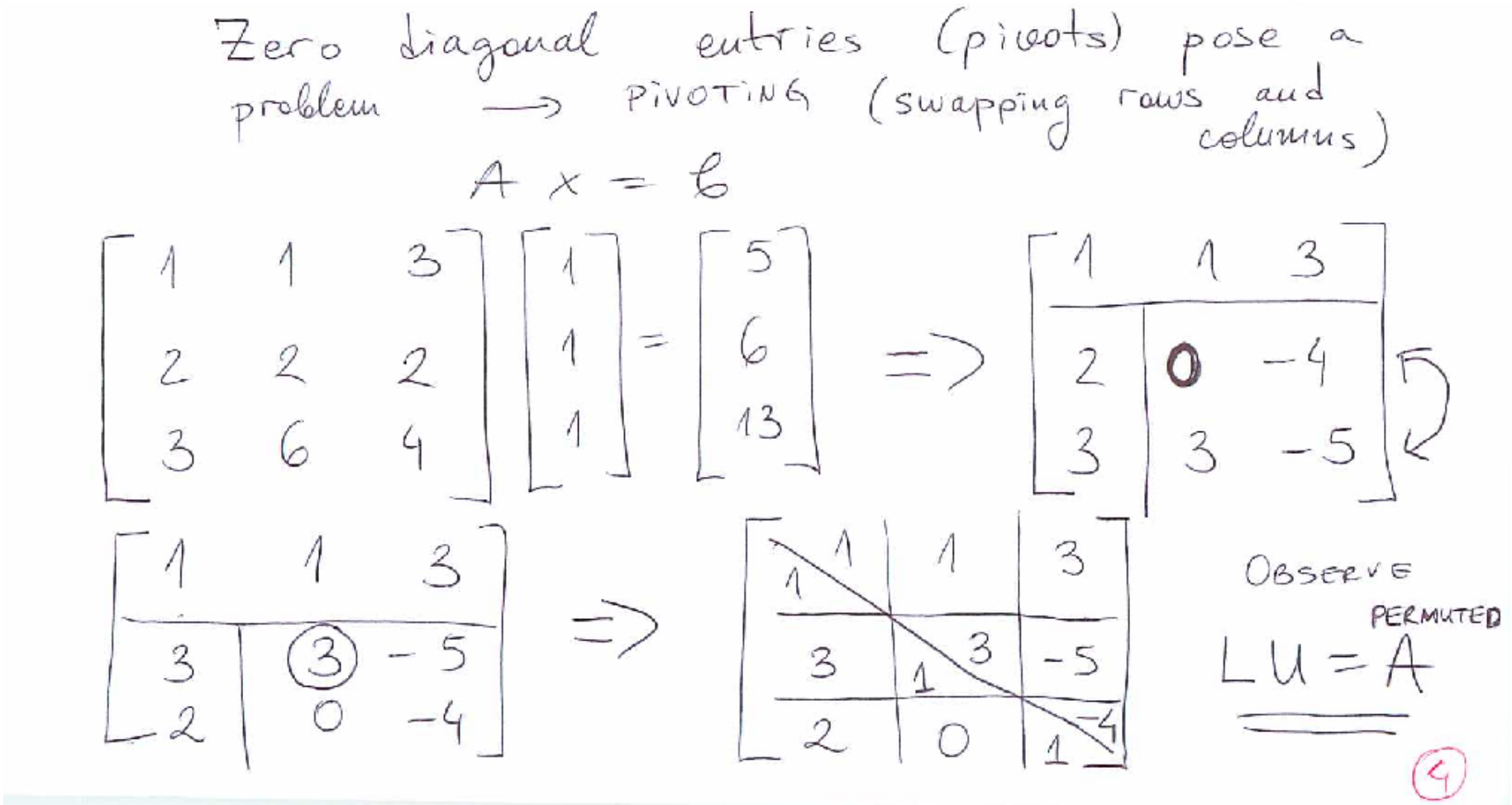
  L is **unit lower triangular** ($l_{ii} = 1$ on diagonal), and **U** is **upper triangular**.
- GEM is thus essentially the same as the LU **factorization method**.

GEM in MATLAB

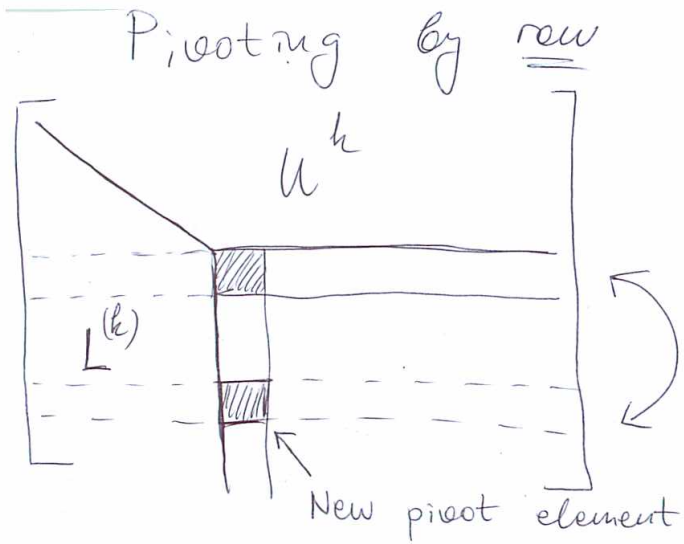
Sample MATLAB code (for learning purposes only, not real computing!):

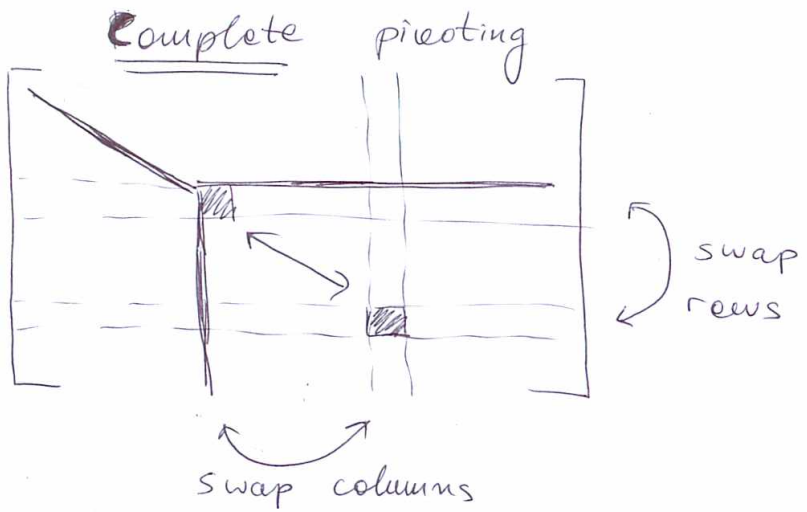
```
function A = MyLU(A)
% LU factorization in-place (overwrite A)
[n,m]=size(A);
if (n ~= m); error('Matrix not square'); end
for k=1:(n-1) % For variable x(k)
    % Calculate multipliers in column k:
    A((k+1):n,k) = A((k+1):n,k) / A(k,k);
    % Note: Pivot element A(k,k) assumed nonzero!
    for j=(k+1):n
        % Eliminate variable x(k):
        A((k+1):n,j) = A((k+1):n,j) - ...
            A((k+1):n,k) * A(k,j);
    end
end
end
```

Pivoting


Zero diagonal entries (pivots) pose a problem → PIVOTING (swapping rows and columns)
Ax = b
OBSERVE
PERMUTED
LU = A

Pivoting during **LU** factorization





- **Partial (row) pivoting** permutes the rows (equations) of **A** in order to ensure sufficiently large pivots and thus numerical stability:

$$\mathbf{PA} = \mathbf{LU}$$

- Here **P** is a **permutation matrix**, meaning a matrix obtained by permuting rows and/or columns of the identity matrix.
- **Complete pivoting** also permutes columns, $\mathbf{PAQ} = \mathbf{LU}$.

Gauss Elimination Method (GEM)

- GEM is a **general** method for **dense matrices** and is commonly used.
- Implementing GEM efficiently and stably is difficult and we will not discuss it here, since others have done it for you!
- The **LAPACK** public-domain library is the main repository for excellent implementations of dense linear solvers.
- MATLAB uses a highly-optimized variant of GEM by default, mostly based on LAPACK.
- MATLAB does have **specialized solvers** for special cases of matrices, so always look at the help pages!

Solving linear systems

- Once an LU factorization is available, solving a linear system is simple:

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{L}\left(\mathbf{Ux}\right) = \mathbf{Ly} = \mathbf{b}$$

  so solve for $\mathbf{y}$ using **forward substitution**.
  This was implicitly done in the example above by overwriting $\mathbf{b}$ to become $\mathbf{y}$ during the factorization.
- Then, solve for $\mathbf{x}$ using **backward substitution**

$$\mathbf{Ux} = \mathbf{y}.$$

- If row pivoting is necessary, the same applies but $\mathbf{L}$ or $\mathbf{U}$ may be permuted upper/lower triangular matrices,

$$\mathbf{A} = \widetilde{\mathbf{L}}\mathbf{U} = \left(\mathbf{P}^T\mathbf{L}\right)\mathbf{U}.$$

In MATLAB

- In MATLAB, the **backslash operator** (see help on *mldivide*)

$$x = A\backslash b \approx A^{-1}b,$$

  solves the linear system $\mathbf{Ax} = \mathbf{b}$ using the LAPACK library.
  Never use matrix inverse to do this, even if written as such on paper.
- Doing $x = A\backslash b$ is **equivalent** to performing an LU factorization and doing two **triangular solves** (backward and forward substitution):

$$\begin{aligned}[\tilde{L}, U] &= lu(A)\\ y &= \tilde{L}\backslash b\\ x &= U\backslash y\end{aligned}$$

- This is a carefully implemented **backward stable** pivoted LU factorization, meaning that the returned solution is as accurate as the conditioning number allows.

GEM Matlab example (1)

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 0 ];
>> b=[2 1 -1]';

>> x=A^(-1)*b; x' % Don't do this!
ans =    -2.5556    2.1111    0.1111

>> x = A\b; x' % Do this instead
ans =    -2.5556    2.1111    0.1111

>> linsolve(A,b)' % Even more control
ans =    -2.5556    2.1111    0.1111
```

GEM Matlab example (2)

```
>> [L,U] = lu(A) % Even better if resolving

L =    0.1429    1.0000         0
       0.5714    0.5000    1.0000
       1.0000         0         0
U =    7.0000    8.0000         0
            0    0.8571    3.0000
            0         0    4.5000

>> norm(L*U-A,inf)
ans =     0

>> y = L\b;
>> x = U\y; x'
ans =   -2.5556    2.1111    0.1111
```

Cost estimates for GEM

- For forward or backward substitution, at step k there are $\sim (n-k)$ multiplications and subtractions, plus a few divisions.
  The total over all n steps is

$$\sum_{k=1}^{n}(n-k) = \frac{n(n-1)}{2} \approx \frac{n^2}{2}$$

  subtractions and multiplications, giving a total of $O(n^2)$ **floating-point operations** (FLOPs).
- The LU factorization itself costs a lot more, $O(n^3)$,

$$\text{FLOPS} \approx \frac{2n^3}{3},$$

  and the triangular solves are negligible for large systems.
- When many linear systems need to be solved with the same **A** the **factorization can be reused**.

Matrix Rescaling

- Pivoting is not always sufficient to ensure lack of roundoff problems. In particular, **large variations** among the entries in **A should be avoided**.
- This can usually be remedied by changing the physical units for **x** and **b** to be the **natural units** $\mathbf{x}_0$ and $\mathbf{b}_0$.
- **Rescaling** the unknowns and the equations is generally a good idea even if not necessary:

$$\mathbf{x} = \mathbf{D}_x \tilde{\mathbf{x}} = \text{Diag}\{\mathbf{x}_0\}\,\tilde{\mathbf{x}} \text{ and } \mathbf{b} = \mathbf{D}_b \tilde{\mathbf{b}} = \text{Diag}\{\mathbf{b}_0\}\,\tilde{\mathbf{b}}.$$

$$\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{D}_x\tilde{\mathbf{x}} = \mathbf{D}_b\tilde{\mathbf{b}} \quad \Rightarrow \quad \left(\mathbf{D}_b^{-1}\mathbf{A}\mathbf{D}_x\right)\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$$

- The **rescaled matrix** $\widetilde{\mathbf{A}} = \mathbf{D}_b^{-1}\mathbf{A}\mathbf{D}_x$ should have a better conditioning.
- Also note that **reordering the variables** from most important to least important may also help.

Conclusions/Summary

- The conditioning of a linear system $\mathbf{Ax} = \mathbf{b}$ is determined by the condition number
$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\| \geq 1$$
- Gauss elimination can be used to solve general square linear systems and also produces a factorization $\mathbf{A} = \mathbf{LU}$.
- Partial pivoting is often necessary to ensure numerical stability during GEM and leads to $\mathbf{PA} = \mathbf{LU}$ or $\mathbf{A} = \widetilde{\mathbf{L}}\mathbf{U}$.
- MATLAB has excellent linear solvers based on well-known public domain libraries like LAPACK. Use them!