5 Lecture 5, Jan 19

Announcements

- HW1 due this Thu @ 11:59PM.
- Quiz 1 this Thu, in class, closed-book.
- Demo

Last time

- Version control.

Today

- Version control (cont'd).
- Algorithm: general concepts.
- Numerical linear algebra: introduction, BLAS.

Algorithms

- *Algorithm* is loosely defined as a set of instructions for doing something. Input → Output.
- Knuth (2005): (1) finiteness, (2) definiteness, (3) input, (4) output, (5) effectiveness
- Basic unit for measuring efficiency is flop. A *flop* (floating point operation) consists of a floating point addition, subtraction, multiplication, division, and comparison, and the usually accompanying fetch and store. Some books count multiplication followed by an addition (fused multiply-add, FMA) as one flop.
- How to measure efficiency of an algorithm? Big O notation. If n is the size of a problem, an algorithm has order $O(f(n))$, where the leading term in the number of flops is $c \cdot f(n)$.

- E.g., matrix-vector multiplication `A %*% b`, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, takes $O(mn)$ flops. Matrix-matrix multiplication `A %*% B`, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, takes $O(mnp)$ flops.

- A hierarchy of computational complexity:
  Exponential order $O(b^n)$ (NP-hard="horrible")
  Polynomial order $O(n^q)$ (doable)
  $O(n \log n)$ (fast)
  Linear order $O(n)$ (fast)
  Log order $O(\log n)$ (super fast).

- One goal of this course is to get familiar with the flop counts for some common numerical tasks in statistics.

> The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.

- Compare flops of the following two R expressions:

```
G %*% Xt %*% y
G %*% (Xt %*% y)
```

  where `G` $\in \mathbb{R}^{p \times p}$, `Xt` $\in \mathbb{R}^{p \times n}$, and `y` $\in \mathbb{R}^n$. "Matrix multiplication is expensive."

- Hardware advancement, e.g., CPU clock rate, only affects constant c. Unfortunately, data size n is increasing too and often at a faster rate.

- Classification of data sets by Huber (1994, 1996).

Data Size	Bytes	Storage Mode
Tiny	10^2	Piece of paper
Small	10^4	A few pieces of paper
Medium	10^6 (megabyte)	A floppy disk
Large	10^8	Hard disk
Huge	10^9 (gigabytes)	Hard disk(s)
Massive	10^{12} (terabytes)	RAID storage

- Difference of $O(n^2)$ and $O(n \log n)$ on massive data. Suppose we have a teraflop supercomputer capable of doing 10^{12} flops per second. For a problem of size $n = 10^{12}$, $O(n \log n)$ algorithm takes about $10^{12} \log(10^{12})/10^{12} \approx 27$ seconds. $O(n^2)$ algorithm takes about 10^{12} seconds, which is approximately 31710 years!

- QuickSort and FFT are celebrated algorithms that turn $O(n^2)$ operations into $O(n \log n)$. *Divide-and-conquer* is a powerful technique. Another example is the Strassen's method, which turns $O(n^3)$ matrix multiplication into $O(n^{\log_2 7})$.

Numerical linear algebra

- The first big chunk of this course is numerical linear algebra.

- Topics in numerical algebra: BLAS, solve linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, regression computations $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}$, eigen-problems $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ and generalized eigen-problems $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{B}\boldsymbol{x}$, singular value decompositions $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, ...

- Please review the following linear algebra facts by yourself.

Linear algebra review

Vector and matrix norms

KL chapter 6. Norm measures the "size".

- Vector norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$. (a) $\|\boldsymbol{x}\| \geq 0$, (b) $\|\boldsymbol{x}\| = 0$ if and only if $\boldsymbol{x} = \boldsymbol{0}$, (c) $\|c\boldsymbol{x}\| = c\|\boldsymbol{x}\|$ (homogeneity), (d) $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (triangle inequality).

- ℓ_p norm. $\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$, $p \in [1, \infty]$. ℓ_1 is the Manhattan norm. ℓ_2 is the Euclidean norm. ℓ_∞ is the sup norm.

- For matrix norm $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$, we further require (e) $\|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.

- Frobenius norm $\|\boldsymbol{A}\|_{\mathrm{F}} = (\sum_i \sum_j a_{ij}^2)^{1/2}$. Properties of (e) is checked by Cauchy-Schwartz inequality.

- Induced matrix norm (or operator norm): $\|\boldsymbol{A}\| = \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|$ for any fixed vector norm. To check property (e), let $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x}$, then $\|\boldsymbol{A}\boldsymbol{B}\| = \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{y}\|}{\|\boldsymbol{y}\|} \frac{\|\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \|\boldsymbol{A}\| \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{B}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.

- Matrix-1 norm, $\|\boldsymbol{A}\|_1 = \max_j \sum_i |a_{ij}|$.
  Matrix-2 norm, $\|\boldsymbol{A}\|_2 = \sqrt{\rho(\boldsymbol{A}^\intercal \boldsymbol{A})} = \max_{\|\boldsymbol{u}\|_2=1, \|\boldsymbol{v}\|_2=1} \boldsymbol{u}^\intercal \boldsymbol{A} \boldsymbol{v}$, which reduces to $\rho(\boldsymbol{A})$ if $\boldsymbol{A}$ is symmetric. ρ is the spectral radius of a matrix, the absolute value of the dominant eigenvalue.
  Matrix-∞ norm, $\|\boldsymbol{A}\|_\infty = \max_i \sum_j |a_{ij}|$.

- When $\boldsymbol{A}$ is a column vector, these matrix induced norms reduce to the original vector norm.

- $\rho(\boldsymbol{A}) \leq \|\boldsymbol{A}\|$ for any induced matrix norm. For any $\boldsymbol{A}$ and $\epsilon > 0$, there exists an induced matrix norm such that $\|\boldsymbol{A}\| \leq \rho(\boldsymbol{A}) + \epsilon$.

Rank

Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

- $\text{rank}(\boldsymbol{A})$ is the maximum number of linearly independent rows (or columns) of a matrix.

- $\text{rank}(\boldsymbol{A}) \leq \min\{m, n\}$.

- A matrix is *full rank* if $\text{rank}(\boldsymbol{A}) = \min\{m, n\}$. It is *full row rank* if $\text{rank}(\boldsymbol{A}) = m$. It is *full column rank* if $\text{rank}(\boldsymbol{A}) = n$.

- A square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is *singular* if $\text{rank}(\boldsymbol{A}) < n$ and *non-singular* if $\text{rank}(\boldsymbol{A}) = n$.

- $\text{rank}(\boldsymbol{A}\boldsymbol{B}) \leq \min\{\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B})\}$. "Matrix multiplication cannot increase the rank."

- $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}^T) = \text{rank}(\boldsymbol{A}^T\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^T)$.

- $\text{rank}(\boldsymbol{A}\boldsymbol{B}) = \text{rank}(\boldsymbol{A})$ if $\boldsymbol{B}$ has full row rank.

- $\text{rank}(\boldsymbol{A}\boldsymbol{B}) = \text{rank}(\boldsymbol{B})$ if $\boldsymbol{A}$ has full column rank.

- $\text{rank}(\boldsymbol{A} + \boldsymbol{B}) \leq \text{rank}(\boldsymbol{A}) + \text{rank}(\boldsymbol{B})$.

Trace

$\boldsymbol{A} \in \mathbb{R}^{n \times n}$ a square matrix.

- $\mathrm{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{ii}$
- $\mathrm{tr}(\boldsymbol{A} + \boldsymbol{B}) = \mathrm{tr}(\boldsymbol{A}) + \mathrm{tr}(\boldsymbol{B})$
- $\mathrm{tr}(\lambda \boldsymbol{A}) = \lambda \mathrm{tr}(\boldsymbol{A})$ where λ is a scalar
- $\mathrm{tr}(\boldsymbol{A}^{\intercal}) = \mathrm{tr}(\boldsymbol{A})$
- Invariance under cycle permutation: $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$. In general, $\mathrm{tr}(\boldsymbol{A}_1 \cdots \boldsymbol{A}_k) = \mathrm{tr}(\boldsymbol{A}_{j+1} \cdots \boldsymbol{A}_k \boldsymbol{A}_1 \cdots \boldsymbol{A}_j)$.

Orthogonality and orthogonalization

- $\boldsymbol{v}_1$ is *orthogonal* to $\boldsymbol{v}_2$, written $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$, if $\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle = \boldsymbol{v}_1^{\intercal} \boldsymbol{v}_2 = 0$. They are *orthonormal* if $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$ and $\|\boldsymbol{v}_i\|_2 = 1$, $i = 1, 2$.
- Gram-Schmidt transformation orthonormalizes two non-zero vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

$$\begin{aligned}
\tilde{\boldsymbol{x}}_1 &= \frac{1}{\|\boldsymbol{x}_1\|_2} \boldsymbol{x}_1 \\
\tilde{\boldsymbol{x}}_2 &= \frac{1}{\|\boldsymbol{x}_2 - \langle \tilde{\boldsymbol{x}}_1, \boldsymbol{x}_2 \rangle \tilde{\boldsymbol{x}}_1\|_2} (\boldsymbol{x}_2 - \langle \tilde{\boldsymbol{x}}_1, \boldsymbol{x}_2 \rangle \tilde{\boldsymbol{x}}_1)
\end{aligned}$$

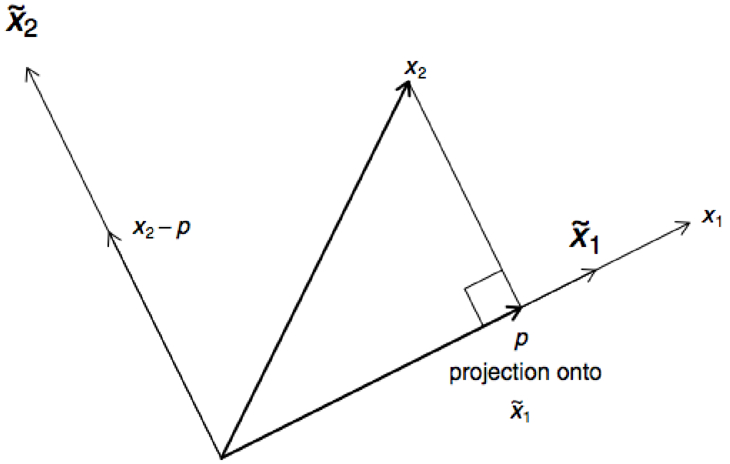


Fig. 2.2. Orthogonalization of x_1 and x_2

- A set of nonzero, mutually orthogonal vectors are linearly independent.

- A real square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is orthogonal if $\boldsymbol{A}^\intercal \boldsymbol{A} = \boldsymbol{I}_n$, i.e., its rows/columns are orthonormal. Orthogonal matrix is of full rank, thus $\boldsymbol{A}^\intercal = \boldsymbol{A}^{-1}$ and $\boldsymbol{A}\boldsymbol{A}^\intercal = \boldsymbol{I}_n$.

Positive (semi)definite matrix

Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is *symmetric.*

- A real symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is *positive semidefinite* (or *nonnegative definite*) if $\boldsymbol{x}^\intercal \boldsymbol{A} \boldsymbol{x} \ge 0$ for all $\boldsymbol{x}$. Notation: $\boldsymbol{A} \succeq \boldsymbol{0}_{n \times n}$.

- E.g., the *Gramian matrix* $\boldsymbol{X}^\intercal \boldsymbol{X}$ or $\boldsymbol{X}\boldsymbol{X}^T$.

- If inequality is strict for all $\boldsymbol{x} \neq \boldsymbol{0}$, then $\boldsymbol{A}$ is *positive definite.* Notation: $\boldsymbol{A} \succ \boldsymbol{0}_{n \times n}$.

- $\boldsymbol{A} \succeq \boldsymbol{B}$ means $\boldsymbol{A} - \boldsymbol{B} \succeq \boldsymbol{0}_{n \times n}$.

- If $\boldsymbol{A} \succeq \boldsymbol{B}$, then $\det(\boldsymbol{A}) \ge \det(\boldsymbol{B})$ with equality if and only if $\boldsymbol{A} = \boldsymbol{B}$.

- $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semidefinite if and only if $\boldsymbol{A}$ is a covariance matrix of a random vector in $\mathbb{R}^n$.

Matrix inverses

Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

- The *Moore-Penrose inverse* of $\boldsymbol{A}$ is a matrix $\boldsymbol{A}^+ \in \mathbb{R}^{n \times m}$ with following properties

  (a) $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}$. (*Generalized inverse*, g_1 *inverse*, or *inner pseudo-inverse*)

  (b) $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$. (*Outer pseudo-inverse.* Any g_1 inverse that satisfies this condition is called a g_2 *inverse*, or *reflexive generalized inverse* and is denoted by $\boldsymbol{A}^*$.)

  (c) $\boldsymbol{A}^+\boldsymbol{A}$ is symmetric.

  (d) $\boldsymbol{A}\boldsymbol{A}^+$ is symmetric.

- $\boldsymbol{A}^+$ exists and is unique for any matrix $\boldsymbol{A}$.
- *Generalized inverse* (or g_1 *inverse*, denoted by $\boldsymbol{A}^-$ or $\boldsymbol{A}^g$): property (a).
- g_2 *inverse* (denoted by $\boldsymbol{A}^*$): properties (a)+(b).
- *Moore-Penrose inverse* (denoted by $\boldsymbol{A}^+$): properties (a)+(b)+(c)+(d).
- If $\boldsymbol{A}$ is square and full rank, then the generalized inverse is unique and denoted by $\boldsymbol{A}^{-1}$ (inverse).
- In practice, the Moore-Penrose inverse $\boldsymbol{A}^+$ is easily computed from the singular value decomposition (SVD) of $\boldsymbol{A}$.
- $(\boldsymbol{A}^-)^T$ is a generalized inverse of $\boldsymbol{A}^T$.
- $\mathcal{C}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{A}\boldsymbol{A}^-)$ and $\mathcal{C}(\boldsymbol{A}^T) = \mathcal{C}((\boldsymbol{A}^-\boldsymbol{A})^T)$.
  $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^-) = \text{rank}(\boldsymbol{A}^-\boldsymbol{A})$.
  "Multiplication by generalized inverse does not change rank."
- $\text{rank}(\boldsymbol{A}^-) \geq \text{rank}(\boldsymbol{A})$. "Generalized inverse has equal or a larger rank than the original matrix."

System of linear equations

$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$.

- When is there a solution? The following statements are equivalent.

  1. The linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has a solution (*consistent*)
  2. $\boldsymbol{b} \in \mathcal{C}(\boldsymbol{A})$.
  3. $\text{rank}((\boldsymbol{A}, \boldsymbol{b})) = \text{rank}(\boldsymbol{A})$.
  4. $\boldsymbol{A}\boldsymbol{A}^-\boldsymbol{b} = \boldsymbol{b}$.

  The last equivalence gives intuition why $\boldsymbol{A}^-$ is called an inverse.

- What are the solutions to a homogeneous system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$?
  $\mathcal{N}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{I}_n - \boldsymbol{A}^-\boldsymbol{A})$.

- If $\boldsymbol{Ax} = \boldsymbol{b}$ is consistent, then $\tilde{\boldsymbol{x}}$ is a solution to $\boldsymbol{Ax} = \boldsymbol{b}$ if and only if

  $$\tilde{\boldsymbol{x}} = \boldsymbol{A}^{-}\boldsymbol{b} + (\boldsymbol{I}_n - \boldsymbol{A}^{-}\boldsymbol{A})\boldsymbol{q}$$

  for some $\boldsymbol{q} \in \mathbb{R}^n$.
  Interpretation: "a specific solution" + "a vector in the null space of $\boldsymbol{A}$".

- $\boldsymbol{Ax} = \boldsymbol{b}$ is consistent for *all* $\boldsymbol{b}$ if and only if $\boldsymbol{A}$ has full row rank.

- If a system is consistent, its solution is unique if and only if $\boldsymbol{A}$ has full column rank.

- If $\boldsymbol{A}$ has full row and column rank, then $\boldsymbol{A}$ is non-singular and the unique solution is $\boldsymbol{A}^{-1}\boldsymbol{b}$.

Gramian matrix $\boldsymbol{A}^\intercal\boldsymbol{A}$

- $\boldsymbol{A}^\intercal\boldsymbol{A}$ is symmetric and positive semidefinite.

- $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}^\intercal) = \text{rank}(\boldsymbol{A}^\intercal\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^\intercal)$.

- $\boldsymbol{A}^\intercal\boldsymbol{A} = \boldsymbol{0}$ if and only if $\boldsymbol{A} = \boldsymbol{0}$.

- $\boldsymbol{B}\boldsymbol{A}^\intercal\boldsymbol{A} = \boldsymbol{C}\boldsymbol{A}^\intercal\boldsymbol{A}$ if and only if $\boldsymbol{B}\boldsymbol{A}^\intercal = \boldsymbol{C}\boldsymbol{A}^\intercal$.

- $\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{B} = \boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{C}$ if and only if $\boldsymbol{A}\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}$.

- For any generalized inverse $(\boldsymbol{A}^\intercal\boldsymbol{A})^{-}$, $[(\boldsymbol{A}^\intercal\boldsymbol{A})^{-}]^\intercal$ is also a generalized inverse of $\boldsymbol{A}^\intercal\boldsymbol{A}$. Note $(\boldsymbol{A}^\intercal\boldsymbol{A})^{-}$ is not necessarily symmetric.

- $(\boldsymbol{A}^\intercal\boldsymbol{A})^{-}\boldsymbol{A}^\intercal$ is a generalized inverse of $\boldsymbol{A}$.

- $\boldsymbol{A}\boldsymbol{A}^{+} = \boldsymbol{A}(\boldsymbol{A}^\intercal\boldsymbol{A})^{-}\boldsymbol{A}^\intercal$, where $\boldsymbol{A}^{+}$ is the Moore-Penrose inverse of $\boldsymbol{A}$.

- $\boldsymbol{P}_{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{A}^\intercal\boldsymbol{A})^{-}\boldsymbol{A}^\intercal$ is symmetric, idempotent, invariant to the choice of generalized inverse $(\boldsymbol{A}^\intercal\boldsymbol{A})^{-}$, and projects onto $\mathcal{C}(\boldsymbol{A})$.

Idempotent matrix and projection

Assume $\boldsymbol{P} \in \mathbb{R}^{n \times n}$.

- A matrix $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ is *idempotent* if and only if $\boldsymbol{P}^2 = \boldsymbol{P}$.
- A matrix $\boldsymbol{P}$ is a *projection* on a vector space $\mathcal{V}$ if (a) $\boldsymbol{P}$ is idempotent, (b) $\boldsymbol{P}\boldsymbol{x} \in \mathcal{V}$ for all $\boldsymbol{x}$, and (c) $\boldsymbol{P}\boldsymbol{z} = \boldsymbol{z}$ for all $\boldsymbol{z} \in \mathcal{V}$.
- An idempotent matrix $\boldsymbol{P}$ is a projection onto $\mathcal{C}(\boldsymbol{P})$.
- For a *general* matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, the matrices $\boldsymbol{A}\boldsymbol{A}^-$ are projections onto $\mathcal{C}(\boldsymbol{A})$ and $\boldsymbol{I}_n - \boldsymbol{A}^-\boldsymbol{A}$ are projections onto $\mathcal{N}(\boldsymbol{A})$.

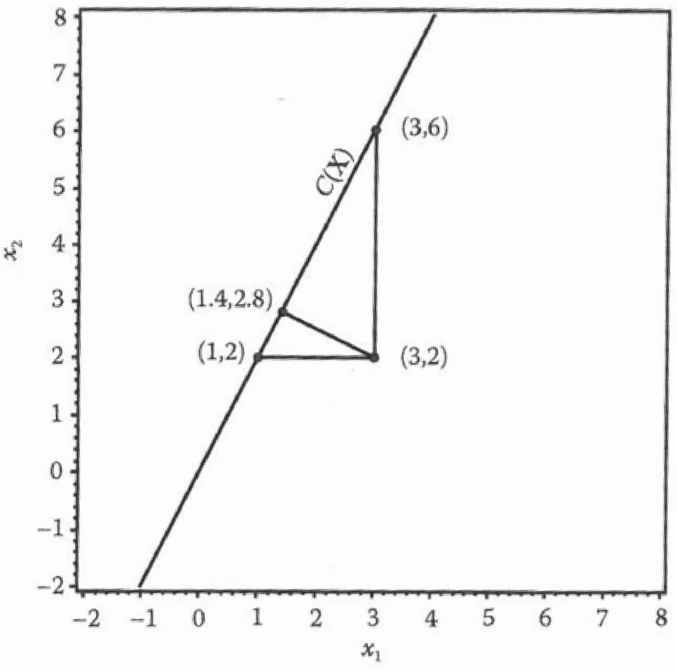


Figure A.1: Three projections onto a column space.

Symmetric idempotent matrix and orthogonal projection

Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$.

- A symmetric, idempotent matrix is called an *orthogonal projection*.

- An orthogonal projection $\boldsymbol{P}$ satisfies $\boldsymbol{y} - \boldsymbol{P}\boldsymbol{y} \perp \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathcal{C}(\boldsymbol{P})$.
- The orthogonal projection onto a vector space is unique.
- If a symmetric, idempotent matrix $\boldsymbol{P}$ projects onto $\mathcal{V}$, then $\boldsymbol{I} - \boldsymbol{P}$ projects onto the orthogonal complement $\mathcal{V}^{\perp}$.
- Pythagorean theorem: For $\boldsymbol{P}$ an orthogonal projection,

$$\|\boldsymbol{y}\|_2^2 = \|\boldsymbol{P}\boldsymbol{y}\|_2^2 + \|(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}\|_2^2.$$

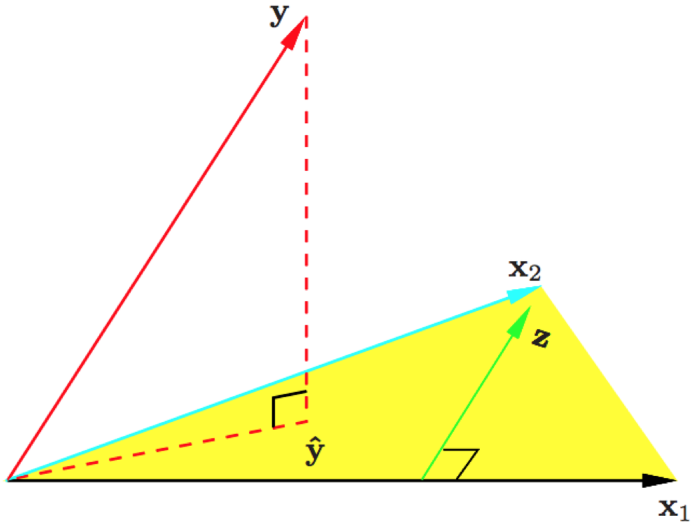
- Many books use the term "projection" in the sense of of orthogonal projection.

Method of least squares

- Goal: Approximate $\boldsymbol{y} \in \mathbb{R}^n$ by a linear combination of columns of $\boldsymbol{X} \in \mathbb{R}^{n \times p}$.
- Least squares criterion: $\min Q(\boldsymbol{b}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|_2^2$.
- Any solution to the normal equation $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{X}^T\boldsymbol{y}$ (always consistent) is a minimizer of the least squares criterion $Q(\boldsymbol{b})$.
- Solutions to the normal equation:

$$\hat{\boldsymbol{b}} = (\boldsymbol{X}^T\boldsymbol{X})^{-}\boldsymbol{X}^T\boldsymbol{y} + (\boldsymbol{I}_p - (\boldsymbol{X}^T\boldsymbol{X})^{-}\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{q},$$

  where $\boldsymbol{q} \in \mathbb{R}^q$ is arbitrary.
- $(\boldsymbol{X}^T\boldsymbol{X})^{-}\boldsymbol{X}^T$ is a generalized inverse of $\boldsymbol{X}$. Therefore the least squares solution applies even when the system $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{y}$ is consistent.
- Least squares solution is unique if and only if $\boldsymbol{X}$ has full column rank.
- $\boldsymbol{P}_{\boldsymbol{X}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-}\boldsymbol{X}^T$ is *the* orthogonal projection onto $\mathcal{C}(\boldsymbol{X})$.
- Geometry: The fitted value from the least squares solution $\hat{\boldsymbol{y}} = \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y}$ is the orthogonal projection of the response vector $\boldsymbol{y}$ onto the column space $\mathcal{C}(\boldsymbol{X})$.



- $\boldsymbol{I}_n - \boldsymbol{P}_{\boldsymbol{X}}$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{X}^T)$.

- Decomposition of $\boldsymbol{y}$:

$$\boldsymbol{y} = \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y} + (\boldsymbol{I}_n - \boldsymbol{P}_{\boldsymbol{X}})\boldsymbol{y} = \hat{\boldsymbol{y}} + \hat{\boldsymbol{e}},$$

  where $\hat{\boldsymbol{y}} \perp \hat{\boldsymbol{e}}$ and

$$\|\boldsymbol{y}\|_2^2 = \|\hat{\boldsymbol{y}}\|_2^2 + \|\hat{\boldsymbol{e}}\|_2^2.$$

Eigenvalues and eigenvectors

KL chapter 8. Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ a square matrix.

- *Eigenvalues* are defined as roots of the characteristic equation $\det(\lambda\boldsymbol{I}_n - \boldsymbol{A}) = 0$.

- If λ is an eigenvalue of $\boldsymbol{A}$, then there exist non-zero $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ such that $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ and $\boldsymbol{y}^T\boldsymbol{A} = \lambda\boldsymbol{y}^T$. $\boldsymbol{x}$ and $\boldsymbol{y}$ are called the (column) *eigenvector* and *row eigenvector* of $\boldsymbol{A}$ associated with the eigenvalue λ.

- $\boldsymbol{A}$ is singular if and only if it has at least one 0 eigenvalue.

- Eigenvectors associated with distinct eigenvalues are linearly independent.

- Eigenvalues of an upper or lower triangular matrix are its diagonal entries: $\lambda_i = a_{ii}$.

- Eigenvalues of an idempotent matrix are either 0 or 1.

- Eigenvalues of an orthogonal matrix have complex modulus 1.

- In most statistical applications, we deal with eigenvalues/eigenvectors of symmetric matrices.

  The eigenvalues and eigenvectors of a real *symmetric* matrix are real.

- Eigenvectors associated with distinct eigenvalues of a symmetry matrix are orthogonal.

- Eigen-decompostion of a symmetric matrix: $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal$, where

  - $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$
  - columns of $\boldsymbol{U}$ are the eigenvectors which are (or can be chosen to be) mutually orthonormal

- A real symmetric matrix is positive semidefinite (positive definite) if and only if all eigenvalues are nonnegative (positive).

- *Spectral radius* $\rho(\boldsymbol{A}) = \max_i |\lambda_i|$.

- $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ a square matrix (not required to be symmetric), then $\text{tr}(\boldsymbol{A}) = \sum_i \lambda_i$ and $|\boldsymbol{A}| = \prod_i \lambda_i$.

Singular value decomposition

KL chapter 9. Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $p = \min\{m, n\}$.

- Singular value decomposition (SVD): $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$, where

  - $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in \mathbb{R}^{m \times m}$ is orthogonal
  - $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \mathbb{R}^{n \times n}$ is orthgonal
  - $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$, $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_p \ge 0$.

  σ_i are called the *singular values*, $\boldsymbol{u}_i$ are the *left singular vectors*, and $\boldsymbol{v}_i$ are the *right singular vectors.*

- $\boldsymbol{A}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$ and $\boldsymbol{A}^T \boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i$ for $i = 1, \ldots, p$.

- Thin SVD. Assume $m \geq n$. $\boldsymbol{A}$ can be factored as $\boldsymbol{A} = \boldsymbol{U\Sigma V}^{\mathsf{T}}$, where
  - $\boldsymbol{U} \in \mathbb{R}^{m \times n}$, $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{U} = \boldsymbol{I}_n$
  - $\boldsymbol{V} \in \mathbb{R}^{n \times n}$, $\boldsymbol{V}^{\mathsf{T}}\boldsymbol{V} = \boldsymbol{I}_n$
  - $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$

- Relation to eigen-decomposition. Using thin SVD,

$$\begin{aligned} \boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} &= \boldsymbol{V\Sigma U}^{\mathsf{T}}\boldsymbol{U\Sigma V}^{\mathsf{T}} = \boldsymbol{V\Sigma}^2\boldsymbol{V}^{\mathsf{T}} \\ \boldsymbol{AA}^{\mathsf{T}} &= \boldsymbol{U\Sigma V}^{\mathsf{T}}\boldsymbol{V\Sigma U}^{\mathsf{T}} = \boldsymbol{U\Sigma}^2\boldsymbol{U}^{\mathsf{T}}. \end{aligned}$$

- Another relation to eigen-decomposition. Using thin SVD,

$$\begin{pmatrix} \boldsymbol{0}_{n\times n} & \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A} & \boldsymbol{0}_{m\times m} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V} & \boldsymbol{V} \\ \boldsymbol{U} & -\boldsymbol{U} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{0}_{n\times n} \\ \boldsymbol{0}_{n\times n} & -\boldsymbol{\Sigma} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V}^{\mathsf{T}} & \boldsymbol{U}^{\mathsf{T}} \\ \boldsymbol{V}^{\mathsf{T}} & -\boldsymbol{U}^{\mathsf{T}} \end{pmatrix}.$$

  Hence any symmetric eigen-solver can produce the SVD of a matrix $\boldsymbol{A}$ without forming $\boldsymbol{AA}^{\mathsf{T}}$ or $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$.

- Yet another relation to eigen-decomposition: If the eigendecomposition of a real symmetric matrix is $\boldsymbol{A} = \boldsymbol{W\Lambda W}^T = \boldsymbol{W}\operatorname{diag}(\lambda_1, \ldots, \lambda_n)\boldsymbol{W}^T$, then

$$\boldsymbol{A} = \boldsymbol{W\Lambda W}^T = \boldsymbol{W} \begin{pmatrix} |\lambda_1| & & \\ & \ddots & \\ & & |\lambda_n| \end{pmatrix} \begin{pmatrix} \operatorname{sgn}(\lambda_1) & & \\ & \ddots & \\ & & \operatorname{sgn}(\lambda_n) \end{pmatrix} \boldsymbol{W}^T$$

  is the SVD of $\boldsymbol{A}$.

- Relation to the Moore-Penrose (MP) inverse: Using thin SVD,

$$\boldsymbol{A}^+ = \boldsymbol{V\Sigma}^+\boldsymbol{U}^{\mathsf{T}},$$

  where $\boldsymbol{\Sigma}^+ = \operatorname{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)$, $r = \operatorname{rank}(\boldsymbol{A})$.

- Denote $\sigma(\boldsymbol{A}) = (\sigma_1, \ldots, \sigma_p)$. Then
  - $\operatorname{rank}(\boldsymbol{A}) = \#$ nonzero singular values $= \|\sigma(\boldsymbol{A})\|_0$
  - $\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T = \sum_{i=1}^r \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$
  - $\|\boldsymbol{A}\|_{\mathrm{F}} = (\sum_{i=1}^p \sigma_i^2)^{1/2} = \|\sigma(\boldsymbol{A})\|_2$

  - $\|\boldsymbol{A}\|_2 = \sigma_1 = \|\sigma(\boldsymbol{A})\|_\infty$

- Assume $\text{rank}(\boldsymbol{A}) = r$ and partition $\boldsymbol{U} = (\boldsymbol{U}_r, \tilde{\boldsymbol{U}}_r) \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} = (\boldsymbol{V}_r, \tilde{\boldsymbol{V}}_r) \in \mathbb{R}^{n\times n}$, then

  - $\mathcal{C}(\boldsymbol{A}) = \text{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$, $\mathcal{N}(\boldsymbol{A}^T) = \text{span}\{\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_m\}$
  - $\mathcal{N}(\boldsymbol{A}) = \text{span}\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n\}$, $\mathcal{C}(\boldsymbol{A}^T) = \text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\}$
  - $\boldsymbol{U}_r\boldsymbol{U}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A})$
  - $\tilde{\boldsymbol{U}}_r\tilde{\boldsymbol{U}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A}^T)$
  - $\boldsymbol{V}_r\boldsymbol{V}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A}^T)$
  - $\tilde{\boldsymbol{V}}_r\tilde{\boldsymbol{V}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A})$

Preliminaries of numerical linear algebra

Numerical linear algebra concerns how matrix/vector computations are done in computer. We first look at some basic linear algebra operations.

Flop counts of some basic linear algebra subroutines (BLAS)

See `http://www.netlib.org/blas/` for a complete listing of BLAS functions.

Level	Example Operation	Name	Dimension	Flops
1	$\alpha \leftarrow \boldsymbol{x}^T\boldsymbol{y}$	dot product	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	n
	$\boldsymbol{y} \leftarrow \boldsymbol{y} + a\boldsymbol{x}$	saxpy	$a \in \mathbb{R}$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	n
2	$\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{A}\boldsymbol{x}$	gaxpy	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$	mn
	$\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{y}\boldsymbol{x}^T$	rank one update	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$	mn
3	$\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$	matrix multiplication	$\boldsymbol{A} \in \mathbb{R}^{m\times p}$, $\boldsymbol{B} \in \mathbb{R}^{p\times n}$, $\boldsymbol{C} \in \mathbb{R}^{m\times n}$	mnp
	$\boldsymbol{A} \leftarrow \boldsymbol{A}\boldsymbol{D}$	column scaling	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_n)$	mn
	$\boldsymbol{A} \leftarrow \boldsymbol{D}\boldsymbol{A}$	row scaling	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_m)$	mn

- Go over the "mxmult" session in R.
  `http://hua-zhou.github.io/teaching/biostatm280-2016winter/mxmult.html`
  Different ways to compute $\boldsymbol{X}^T\boldsymbol{W}^{-1}\boldsymbol{X}$, such as in the weighted least squares.
  $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, $\boldsymbol{W} = \text{diag}(w_1, \ldots, w_n) \in \mathbb{R}^{n\times n}$.

1. `t(X) %*% solve(W) %*% X`: $O(n^3 + n^2p + np^2)$ flops, why need to do the expensive matrix inversion?
2. `t(X) %*% diag(1 / w) %*% X`: $O(n^2p + np^2)$ flops, why need to save a diagonal matrix and do an extra matrix multiplication?
3. `(t(X) / w)  %*% X`: wrong! $\boldsymbol{w}$ recycled incorrectly
4. `t(X) %*% (X / w)`: $O(np^2 + np) = O(np^2)$ flops
5. `crossprod(X, X / w)`: same as 4, skip the transpose operation

- Another example: Fisher information matrix of a generalized linear model (GLM): $\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}$, where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{W} = \text{diag}(w_1, \ldots, w_n)$ are the observation weights.

- Bottom line: Always be flop-aware when writing code.

> The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.

- But, for high-performance matrix commutations, it is *not* enough to minimize flops. Pipelining, effective use of memory hierarchy, data layout in memory, ... play important role too.

Vector computer

- Most modern computers are vector machines, which perform vector calculations (saxpy, inner product) fast.

- Vector processing by *pipelining*. E.g., vector addition $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y}$

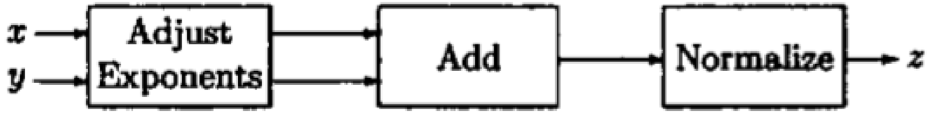


FIG. 1.4.1 ***A 3-Cycle Adder***

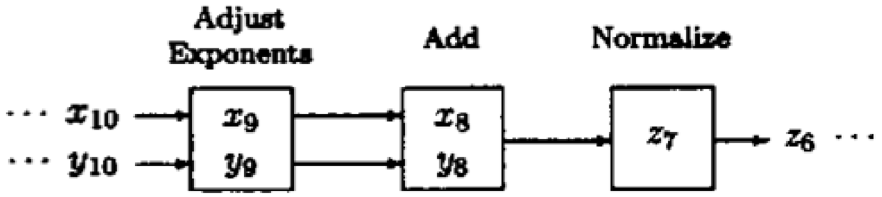
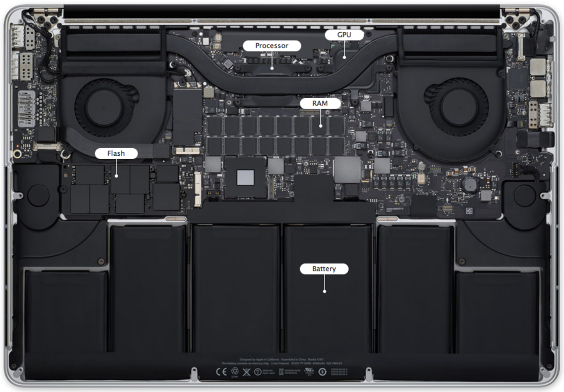


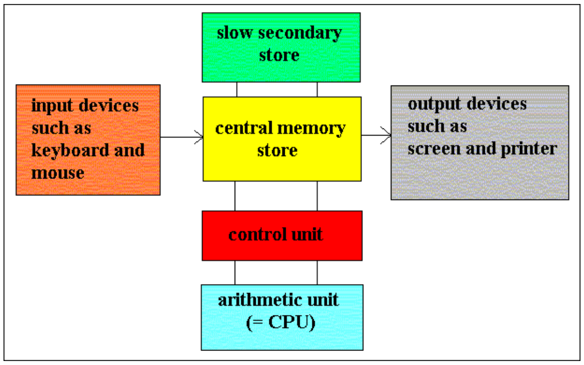
FIG. 1.4.2 *Pipelined Addition*

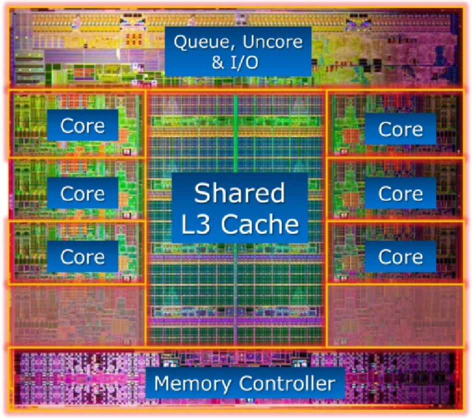
- For example, for Intel Sandy Bridge/Ivy Bridge:
  - 8 DP FLOPs/cycle: 4-wide AVX addition + 4-wide AVX multiplication
  - 16 SP FLOPs/cycle: 8-wide AVX addition + 8-wide AVX multiplication

  and for Intel Haswell/Broadwell/Skylake:
  - 16 DP FLOPs/cycle: two 4-wide FMA (fused multiply-add) instructions
  - 32 SP FLOPs/cycle: two 8-wide FMA (fused multiply-add) instructions
- One implication of the pipelining technology is that we need to ship vectors to the pipeline fast enough to keep the arithmetic units (ALU) busy and maintain high throughput.

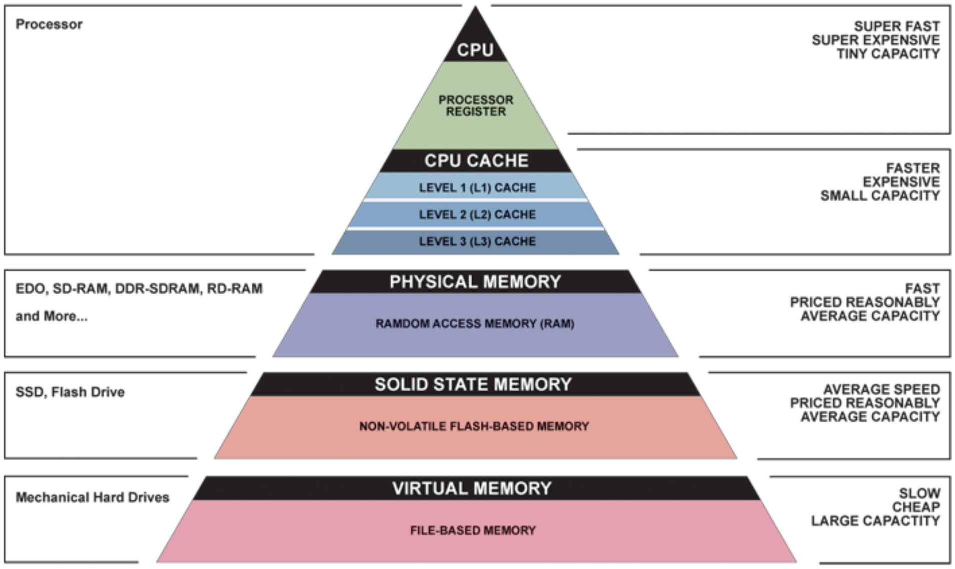
Computer architecture and BLAS




slow secondary store
input devices such as keyboard and mouse
central memory store
output devices such as screen and printer
control unit
arithmetic unit (= CPU)



- Memory hierarchy:



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Upper the hierarchy, faster the memory accessing speed, and more expensive the memory units.

> Key to high performance is effective use of memory hierarchy. True on all architectures.

- Can we keep the super fast arithmetic units busy with enough deliveries of matrix data and ship the results to memory fast enough to avoid backlog?
  Answer: use high-level BLAS as much as possible

- Why high-level BLAS?

BLAS	Dimension	Mem Refs	Flops	Ratio
Level 1: $\boldsymbol{y} \leftarrow \boldsymbol{y} + a\boldsymbol{x}$	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	$3n$	n	3:1
Level 2: $\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{A}\boldsymbol{x}$	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{A} \in \mathbb{R}^{n \times n}$	n^2	n^2	1:1
Level 3: $\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$	$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$	$4n^2$	n^3	4:n

- BLAS 1 tend to be *memory bandwidth-limited*. E.g., Xeon X5650 CPU has a theoretical throughput of 128 DP GFLOPS but a max memory bandwidth of 32GB/s.

- Higher level BLAS (3 or 2) make more effective use of arithmetic logic units (ALU) by keeping them busy.

- Message: Although we state many algorithms (solving linear equations, least squares, eigen-decomposition, SVD, ...) in terms of inner product and saxpy, the actual implementation may be quite different.

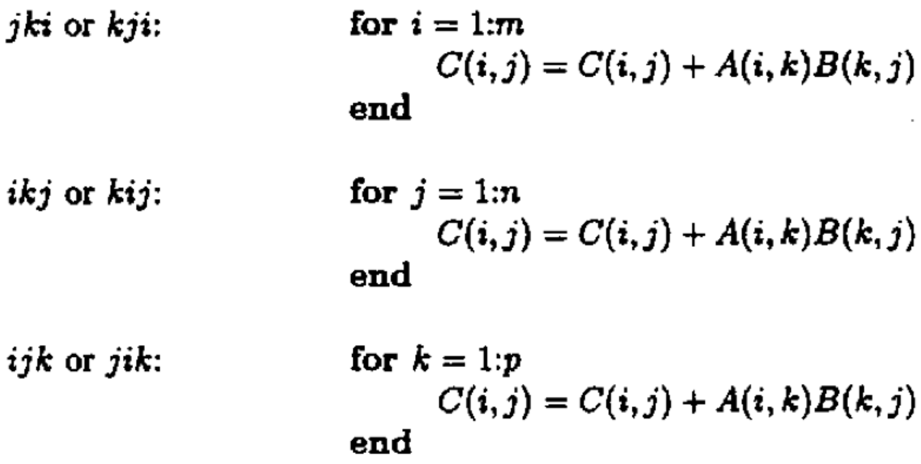
- A distinction between LAPACK and LINPACK is that LAPACK makes use of higher level BLAS as much as possible (usually by smart partitioning) to increase the so-called *level-3 fraction*.

Effect of data layout

- Data layout in memory effects execution speed too. It is much faster to move chunks of data in memory than retrieving/writing scattered data.

- Storage mode: column-major (Fortran, Matlab, R) vs row-major (C/C++).

- Take matrix multiplication as an example. Assume the storage is column-major, such as in Fortran. $\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, $\boldsymbol{C} \in \mathbb{R}^{m \times p}$. There are 6 variants of the algorithms according to the order in the triple loops. We pay attention to the innermost loop, where the vector calculation occurs,

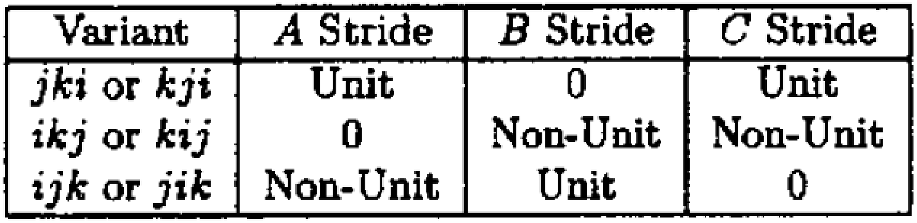


```
jki or kji:     for i = 1:m
                    C(i,j) = C(i,j) + A(i,k)B(k,j)
                end

ikj or kij:     for j = 1:n
                    C(i,j) = C(i,j) + A(i,k)B(k,j)
                end

ijk or jik:     for k = 1:p
                    C(i,j) = C(i,j) + A(i,k)B(k,j)
                end
```

and the associated *stride* when accessing the three matrices in memory (assuming column-major storage)



Variant	A Stride	B Stride	C Stride
jki or kji	Unit	0	Unit
ikj or kij	0	Non-Unit	Non-Unit
ijk or jik	Non-Unit	Unit	0

Apparently the variants jki or kji are preferred.

- Message: data storage mode effects algorithm implementation too.

Solving linear equations

We consider algorithms for solving linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, a ubiquitous task in statistics. Idea: turning original problem into an "easy" one, e.g., triangular system.

Triangular system

- *Forward substitution* to solve $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{L} \in \mathbb{R}^{n \times n}$ is lower triangular

$$\begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$\begin{aligned} x_1 &= b_1/a_{11} \\ x_2 &= (b_2 - a_{21}x_1)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \\ &\vdots \\ x_m &= b_m - a_{m1}x_1 - a_{m2}x_2 - \dots - a_{m,m-1}x_{m-1})/a_{mm} \end{aligned}$$

$n^2/2$ flops ($n^2/2$ multiplications/divisions and $n^2/2$ additions/substractions) and $\boldsymbol{L}$ is accessed by row.

- *Back substitution* to solve $\boldsymbol{Ux} = \boldsymbol{b}$ where $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ is upper triangular

$$\begin{bmatrix} a_{11} & \dots & a_{1,m-1} & a_{1m} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & a_{m-1,m-1} & a_{m-1,m} \\ 0 & \dots & 0 & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{m-1} \\ b_m \end{bmatrix}$$

$$\begin{aligned} x_m &= b_m/a_{mm} \\ x_{m-1} &= (b_{m-1} - a_{m-1,m}x_m)/a_{m-1,m-1} \\ x_{m-2} &= (b_{m-2} - a_{m-2,m-1}x_{m-1} - a_{m-2,m}x_m)/a_{m-2,m-2} \\ &\vdots \\ x_1 &= b_1 - a_{12}x_2 - a_{13}x_3 - \dots - a_{1m}x_m)/a_{11} \end{aligned}$$

$n^2/2$ flops ($n^2/2$ multiplications/divisions and $n^2/2$ additions/substractions) and $\boldsymbol{U}$ is accessed by row.

- Column version: reverse the order of looping.
- BLAS level 2 function: `?trsv` (triangular solve with one right hand side)
- BLAS level 3 function: `?trsm` (matrix triangular solve, i.e., multiple right hand sides)

- In R, `forwardsolve()` and `backsolve()` (wrappers of `dtrsm`).

- In JULIA, `A \b` uses forward or backward substitution when $\boldsymbol{A}$ is a triangular matrix. Or we can simply call BLAS functions directly.

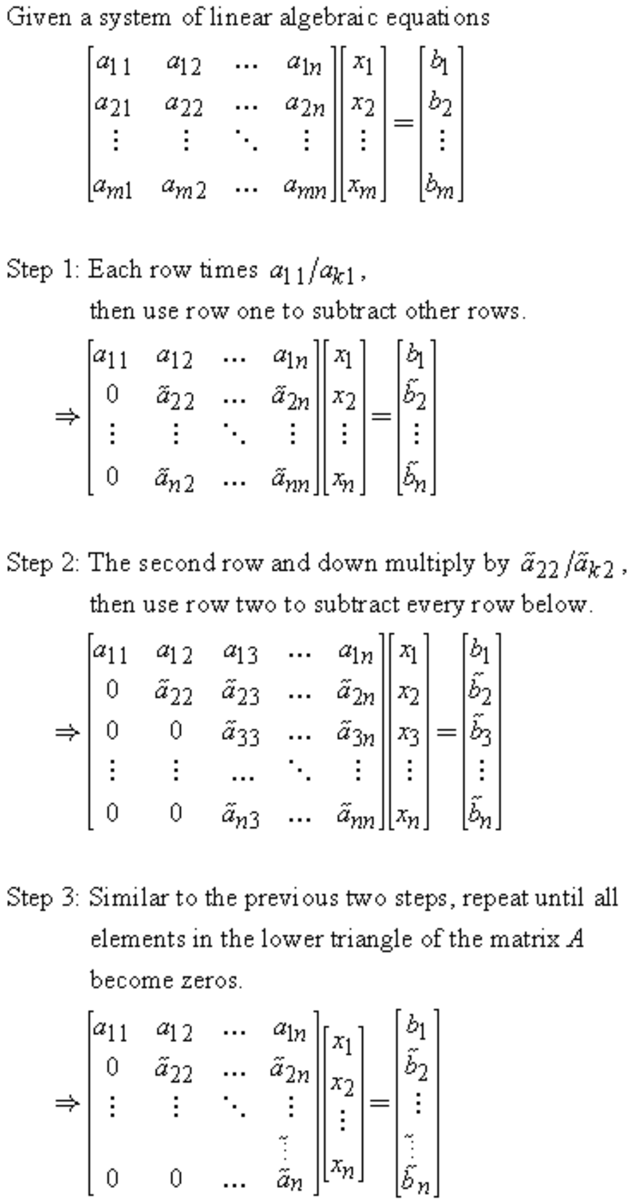
- Eigenvalues of $\boldsymbol{L}$ are diagonal entries $\lambda_i = \ell_{ii}$. $\det(\boldsymbol{L}) = \prod_i \ell_{ii}$.

- A *unit triangular matrix* is a triangular matrix with all diagonal entries being 1.

- The algebra of triangular matrices (HW2)

  - The product of two upper (lower) triangular matrices is upper (lower) triangular.
  - The inverse of an upper (lower) triangular matrix is upper (lower) triangular.
  - The product of two unit upper (lower) triangular matrices is unit upper (lower) triangular.
  - The inverse of a unit upper (lower) triangular matrix is unit upper (lower) triangular.

Gaussian elimination and LU decomposition



Given a system of linear algebraic equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Step 1: Each row times a_{11}/a_{k1},
then use row one to subtract other rows.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 2: The second row and down multiply by $\tilde{a}_{22}/\tilde{a}_{k2}$,
then use row two to subtract every row below.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ 0 & 0 & \tilde{a}_{33} & \cdots & \tilde{a}_{3n} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \tilde{a}_{n3} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 3: Similar to the previous two steps, repeat until all
elements in the lower triangle of the matrix A
become zeros.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ & & & \tilde{\vdots} \\ 0 & 0 & \cdots & \tilde{a}_{n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{\vdots} \\ \tilde{b}_n \end{bmatrix}$$

- History: It is one by-product of Gauss's efforts to re-discover the dwarf planet Ceres using method of least squares. No linear algebra in 1800!

- Solve $\boldsymbol{Ax} = \boldsymbol{b}$ for a general matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

- Idea: a series of *elementary operations* that turn $\boldsymbol{A}$ into a triangular system.

- We consider the square $\boldsymbol{A}$ case first.

- *Elementary operator matrix* $\boldsymbol{E}_{jk}(c)$ is the identity with the 0 in position (j, k) replaced by c. For any vector $\boldsymbol{x}$,

  $$\boldsymbol{E}_{jk}(c)\boldsymbol{x} = (x_1, \ldots, x_{j-1}, x_j + cx_k, x_{j+1}, \ldots, x_n)^\intercal.$$

  Applying $\boldsymbol{E}_{jk}(c)$ to both sides of the system $\boldsymbol{Ax} = \boldsymbol{b}$ replaces the j-th equation $\boldsymbol{a}_{j*}^\intercal \boldsymbol{x} = b_j$ by $\boldsymbol{a}_{j*}^\intercal \boldsymbol{x} + c\boldsymbol{a}_{k*}^\intercal \boldsymbol{x} = b_j + cb_k$. For $j > k$, $\boldsymbol{E}_{jk}(c) = \boldsymbol{I} + c\boldsymbol{e}_j\boldsymbol{e}_k^\intercal$ is unit lower triangular and full rank. $\boldsymbol{E}_{jk}^{-1}(c) = \boldsymbol{E}_{jk}(-c)$.

- Zeroing the first column

  $$\begin{aligned}
  \boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b} \\
  \boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b} \\
  &\vdots \\
  \boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b}
  \end{aligned}$$

  where $c_i^{(1)} = -a_{i1}/a_{11}$. Denote $\boldsymbol{M}_1 = \boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})$.

- Then zero the k-th column for $k = 2, \ldots, n-1$ sequentially. This results in a transformed linear system $\boldsymbol{Ux} = \tilde{\boldsymbol{b}}$, where $\boldsymbol{U} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{A}$ is upper triangular and $\tilde{\boldsymbol{b}} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{b}$. $\boldsymbol{M}_k$ has the shape

  $$\boldsymbol{M}_k = \boldsymbol{E}_{n,k}^{(k)}\cdots\boldsymbol{E}_{k+1,k}^{(k)} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & c_{k+1}^{(k)} & 1 & & \\ & & \vdots & & \ddots & \\ & & c_n^{(k)} & & & 1 \end{pmatrix},$$

  where $c_i^{(k)} = -\tilde{a}_{ik}^{(k-1)}/\tilde{a}_{kk}^{(k-1)}$. $\boldsymbol{M}_k$ is unit lower triangular and full rank. $\boldsymbol{M}_k$ are called the *Gauss transformations*.

- Let $\boldsymbol{L} = \boldsymbol{M}_1^{-1}\cdots\boldsymbol{M}_{n-1}^{-1}$. We have the decomposition

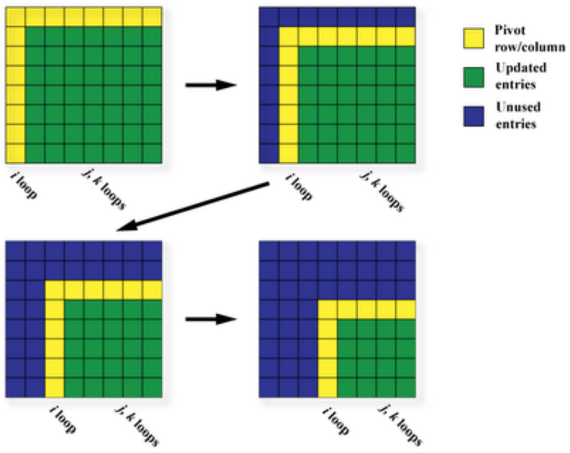
  $$\boldsymbol{A} = \boldsymbol{LU}.$$

  $\boldsymbol{M}_k$ is unit upper triangular, so $\boldsymbol{M}_k^{-1}$ and thus $\boldsymbol{L}$ is unit lower triangular.

- Where is $\boldsymbol{L}$? Note $\boldsymbol{M}_k = \boldsymbol{I} + (0, \ldots, 0, c_{k+1}^{(k)}, \ldots, c_n^{(k)})^{\mathsf{T}} \boldsymbol{e}_k^{\mathsf{T}}$. By Sherman-Morrison, $\boldsymbol{M}_k^{-1} = \boldsymbol{I} - (0, \ldots, 0, c_{k+1}^{(k)}, \ldots, c_n^{(k)})^{\mathsf{T}} \boldsymbol{e}_k^{\mathsf{T}}$. So the entries of $\boldsymbol{L}$ are simply $\ell_{ik} = -c_i^{(k)}$, $i > k$, the negative of multipliers in GE.

- The whole LU procedure is done in place, i.e., $\boldsymbol{A}$ is overwritten by $\boldsymbol{L}$ and $\boldsymbol{U}$.

- Implementation: outer product LU (kij loop), block outer product LU (higher level-3 fraction), Crout's algorithm (jki loop), ...



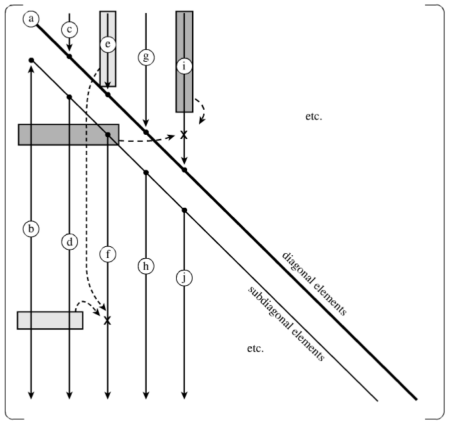


Figure 2.3.1. Crout's algorithm for *LU* decomposition of a matrix. Elements of the original matrix are modified in the order indicated by lower case letters: a, b, c, etc. Shaded boxes show the previously modified elements that are used in modifying two typical elements, each indicated by an "x".

- LU decomposition exists if the principal sub-matrix $A(1 : k, 1 : k)$ is non-singular for $k = 1, \ldots, n-1$. If the LU decomposition exists and $\boldsymbol{A}$ is non-singular, then the LU decomposition is unique and $\det(\boldsymbol{A}) = \prod_{i=1}^n u_{ii}$.

- This *forward elimination* or *LU decomposition* costs $(n-1)^2 + (n-2)^2 + \cdots + 1^2 \approx \frac{1}{3}n^3$ flops ($n^3/3$ multiplications and $n^3/3$ additions).

- Given LU, one right hand side costs n^2 flops (one backward substitution and then forward substitution).

- For inversion, there are n right hand sides $\boldsymbol{e}_i$. However, taking advantage of zeros reduces n^3 flops to $\frac{2}{3}n^3$ (see JM exercise 3.2). So matrix inversion costs $\frac{1}{3}n^3 + \frac{2}{3}n^3 = n^3$ flops in total.

- We do *not* compute matrix inverse unless (i) it is absolutely necessary to compute s.e., (2) number of right hand sides is much larger than n, (3) n is small.

- LU decomposition of a rectangular matrix $\boldsymbol{A} \in \mathbf{R}^{m\times n}$ exists if $\boldsymbol{A}(1:k, 1:k)$ is non-singular for $k = \min\{m, n\}$. For example,

  $$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 3 & 1 \\ 5 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & -2 \end{pmatrix}$$

  and

  $$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{pmatrix}.$$

  Slight modification to the algorithm.