1 20th of September 2018 — F. Poloni

1.1 A warm up

Before starting here is a small recap

- **Vector-Scalar product**:
  Let $x \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ we call **multiple** of vector x the following:

$$\lambda x = x\lambda = \begin{pmatrix} \lambda x_1 \\ \vdots \\ \lambda x_n \end{pmatrix}$$

  .

- **Vector-Vector product**:
  Let $x, y \in \mathbb{R}^n$. The product between those two vectors is computed as follows $x^T y = \sum_{i=1}^{n} x_i y_i$ and $x^T y \in \mathbb{R}$.

- **Scalar-Matrix product**:
  Let $A \in \mathbb{R}^{n \times m}$ and $\lambda \in \mathbb{R}$ we call the **scalar-matrix product** the following:

$$\lambda A = A\lambda = \begin{pmatrix} \lambda A_{11} & \lambda A_{12} & \dots \\ \lambda A_{21} & \lambda A_{22} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

  .

- **Matrix-Vector product**:
  Given a matrix $A \in M(n, m, \mathbb{R})$ and a vector $v \in \mathbb{R}^m$ the **matrix-vector product** $Av = w \in \mathbb{R}^n$ is computed as follows:

$$w = Av = \begin{pmatrix} A_1 v \\ A_2 v \\ \vdots \\ A_m v \end{pmatrix}, \; w_i = \sum_{j=1}^{m} A_{ij} v_j$$

  This is the simple way, just a row-by-column vector product, the computational complexity of this operation is $O(n^2)$.
  The smart way to compute it: **linear combinations** of columns of A, e.g.:

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

with **linear combinations** we have:

$$\begin{pmatrix} A_{11} \\ A_{21} \\ A_{31} \\ A_{41} \end{pmatrix} v_1 + \begin{pmatrix} A_{12} \\ A_{22} \\ A_{32} \\ A_{42} \end{pmatrix} v_2 + \begin{pmatrix} A_{13} \\ A_{23} \\ A_{33} \\ A_{43} \end{pmatrix} v_3 = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

- **Matrix-Matrix Product**:
  Given two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{m \times k}$ we call **matrix-matrix product** the following: $C = AB$ such that $C_{ij} = A_i B^j$, where ${A_i}^T \in \mathbb{R}^m$ is the i-th row of A, B^i is the i-th column of B ($B^i \in \mathbb{R}^m$) and $C \in M(n, k, \mathbb{R})$. Notice that this product is **not commutative**: $AB \neq BA$ might not even make sense dimension-wise.

  Computational Cost: multiplying $m \times n$ and $n \times p$ requires $m(2n-1)p$ floating point operations (flops) and two matrix, A and B, are both $n \times n$ is $O(n^2)$. Forget about fancier algorithms (e.g. Strassen)

  Order of operations

  Usual algebra properties hold, e.g.: A(B + C) = AB + AC, A(BC) = (AB)C, etc...
  Parenthesization matters a lot: if $A, B \in \mathbb{R}^{n \times n}$, $v \in \mathbb{R}^n$, then $(AB)v$ costs $O(n^3)$, but $A(Bv)$ costs $O(n^2)$. Programming languages usually do not rearrange parentheses to help.

- **Image** of a matrix A ($\mathrm{Im}(A)$): the set of vectors that can be obtained multiplying A by any vector in the domain of A.

- **Kernel** of a matrix A ($\ker(A)$): the set of vectors w in its domain such that $Aw = 0$.

- Given a matrix $A \in M(n, \mathbb{R})$ we call **inverse** of A the matrix A^{-1} such that:

$$A^{-1}A = AA^{-1} = I_n = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

  The **inverse of a product** (shoe-sock identity) is $(AB)^{-1} = B^{-1}A^{-1}$. Notice that this identity holds only for square matrices.

- The **transpose** of a matrix $A \in M(n, m, \mathbb{R})$ is A^T such that $A^T_{ij} = A_{ji}$. The **transpose of a product** (shoe-sock identity) is $(AB)^T = B^T A^T$. (This identity holds for square and rectangular matrices)

Definition 1.1. ***General linear group (GL)***: *the general linear group of degree n is the set of $n \times n$ invertible matrices, together with the operation of ordinary matrix multiplication*

Fact 1.1. *Let $A \in GL(n, \mathbb{R})$ (aka A is a real square matrix of size n and invertible), $B, C \in M(n, m, \mathbb{R})$ and we have the product $AB = AC$. If there is a matrix M such that $MA = I$:*

$$(MA)B = (MA)C \Longleftrightarrow B = C, \ M = A^{-1}$$

.

So $AB = AC$ does not usually imply $B = C$, A must be invertible!

Row and column vectors notation

$$v = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}, \, v^T = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$$

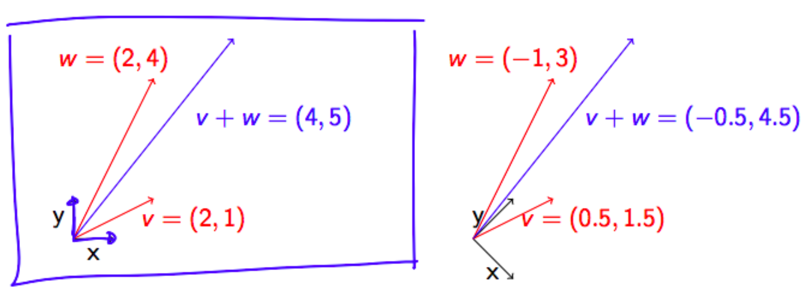
v is a column vector in $\mathbb{R}^3$ (or a matrix in $\mathbb{R}^{3\times 1}$) and v^T is a row vector (or a matrix in $\mathbb{R}^{1\times 3}$).

Definition 1.2. ***Basis***: *a set B of elements (vectors) in a vector space V is called a* ***basis****, if every element of V may be written in a unique way as a (finite) linear combination of elements of B. The coefficients of this linear combination are referred to as components or coordinates on B of the vector. The elements of a basis are called basis vectors.*

Canonical basis: $w = w_1 e_1 + w_2 e_2 + w_3 e_3 + w_4 e_4$, e.g. for $m = 4$

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad e_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The powerful idea behind linear algebra: many relations are true regardless of the basis we use. E.g. w, v and $w + v$ in two different bases.



1.2 Solving Linear Systems

The objective of this course, for the part concerning numerical methods, is solving linear systems efficiently.

Definition 1.3 (Linear system). *Let $A \in M(n, m, \mathbb{R})$, $b \in \mathbb{R}^n$ and $x \in \mathbb{R}^m$. We term **linear system** the following:*

$$Ax = b$$

Our goal is to approximate such vector x, hence resulting in solving a minimum problem:

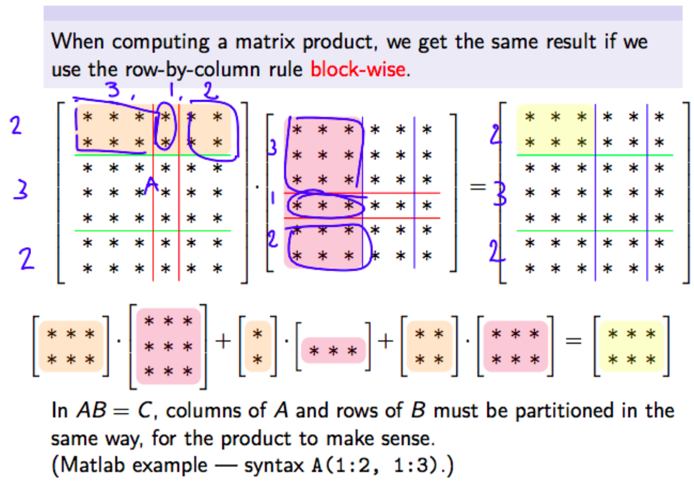
$$\min \|Ax - b\|$$

If we have a square and invertible A matrix solve a linear systems means: find coordinates $x_1, \ldots, x_m$ needed to write b as linear combinations of the columns of (square) $A \in \mathbb{R}^{m \times m}$ and in this case, the solution is given by: $x = A^{-1}b$.
Warning: this is not the best way to solve a linear system on a computer!

Something on Matlab . . .

Notice that the machine precision is 10^{-16}, so we should pay attention when making computations, since we may incur in some error (proportional to the size of the operands).
In Matlab a matrix is written as `A=[1, 2, 3; 4, 5, 6];`, where `[1, 2, 3]` is the first row of the matrix A.
The transpose of a matrix or a vector is denoted by `A'`.
The inverse of a square matrix is denoted by `inv(A)`.
If we are interested in only a part of our matrix `A` we may write `A[1:2, 1:3]` and obtain only the rows of `A` that go from 1 to 2 and those columns from 1 to 3.

Definition 1.4 (Block multiplications). *Let $A \in M(n, m, \mathbb{R})$ and let $B \in M(m, k, \mathbb{R})$. We can compute the result of a block of the matrix AB as the product of the two blocks in A and B in the corresponding position.*



Note: Block operations usually give better performance: one matrix-matrix product performs faster than n matrix-vector products (even if they have the same number of flops). This is one of the reasons why library calls usually perform better than hand-coded loops (Blas/Lapack).

Fact 1.2 (Block triangular matrices)**.** *Let $M \in M(n, m, \mathbb{R})$ and $B \in M(m, k, \mathbb{R})$ such that they are* ***block triangular****. Their product is a block triangular matrix as well, block triangular matrices are closed under products:*

$$MB = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix} \begin{pmatrix} D & E \\ 0 & F \end{pmatrix} = \begin{pmatrix} AD & AE + BF \\ 0 & CF \end{pmatrix}$$

Fact 1.3 (Properties of triangular matrices)**.**
Let M be a block triangular matrix, with all A_{ii} square:

$$M = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1k} \\ 0 & A_{22} & \dots & A_{2k} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_{kk} \end{pmatrix}$$

1. *a block triangular matrix is invertible iff all diagonal blocks A_{ii} are invertible;*
2. *the eigenvalues of a block triangular matrix are the union of the eigenvalues of each A_{ii} block;*
3. *let $M \in GL(n, m, \mathbb{R})$ such that $M = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$ the inverse of M is*

   $$M^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{pmatrix};$$
4. *the product of two block (upper/lower) triangular matrices (with compatible block sizes) is still block triangular.*

Why are we interested in block triangular matrices? They depict a situation as shown in Figure 1.1.

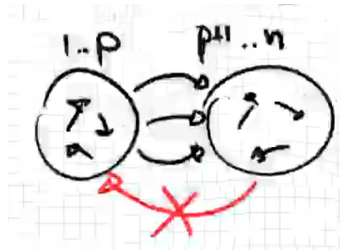


FIGURE 1.1: The adjacency matrix of a biparted graph has 0s in its bottom left part (Matlab syntax `A[p+1:n; 1:p]=0`).

General principle: matrix structures matter. Block triangular linear system has a cheaper system solution than a general system as shown in example 1.1.

Example 1.1. *2x2 block triangular linear system*

$$\begin{pmatrix} A & B \\ 0 & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} e \\ f \end{pmatrix}$$

(Again, diagonal blocks are square and all dimensions are compatible.)

$$\begin{pmatrix} Ax + By \\ Cy \end{pmatrix} = \begin{pmatrix} e \\ f \end{pmatrix} \implies y = C^{-1}f, x = A^{-1}(e - BC^{-1}f)$$

$$\begin{pmatrix} A & B \\ 0 & C \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{pmatrix}$$

Informal idea: we can start solving from the variables in C.

1.3 Orthogonality

Definition 1.5 (Norms)**.** *Let $x \in \mathbb{R}^n$. We "measure" their magnitude using so-called "norms".*

EUCLIDEAN: $\|x\|_2 = x^T x = \sqrt{\sum_{i=1}^{n} {x_i}^2}$;

NORM 1: $\|x\|_1 = \sum_{i=1}^{n} |x_i|$;

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

0-NORM: $\|x\|_0 = |\{i \ : \ |x_i| > 0\}|$;

∞-Norm: $\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$.

From now on in this part of th course we will refer to norm-2 only.

Definition 1.6 (Orthogonal matrix)**.** *Let $A \in M(n, \mathbb{R})$ a square matrix. A is orthogonal iff:*

- $A^T A = I_n$
- $AA^T = I_n$
- $A^{-1} = U^T$

where I_n is the identity matrix of size n (1 on the diagonal, 0 elsewhere).