CHAPTER 1 MOTIVATION AND EXAMPLES. COMPUTATIONAL CONSIDERATIONS

1.1 MOTIVATION AND OUTLINE. This course is intended as a first introduction to the methods of numerical linear algebra. Students are assumed to be already familiar with the basics of linear algebra, although some facts and properties are recalled here in an effort to make this document self-contained. Solution methods for solving (often very large) systems of linear equations are esential components of scientific computation in many areas, including model-based simulation of the response of physical systems, optimization, data analysis and statistics. Even though many readers will not be directly faced with designing or implementing such methods, understanding their underlying principles and conditions of applications is a necessary part of the background of anyone involved in mathematical modelling and scientific computation.

The topics presented in these lecture notes follow a natural order:

- Some general considerations relevant to matrix computation,
- Direct solution methods for linear systems and least-squares problems,
- Iterative solution methods for linear systems,
- Numerical solution of eigenvalue problems,
- Ill-posed linear systems and least-squares problems.

There is of course a huge literature on the topic of computational linear algebra, which is only very partially reflected by the sample of classical references given in the bibliography of this document without any claim of completeness or extensivity. In particular, the recent book [14] is recommmended as a very readable and student-friendly introduction to many of the important topics of computational linear algebra (including advanced topics such as direct solution methods for large sparse linear systems). That book moreover provides comprehensive examples of implementation (and implementation exercises) based on the Julia programming language, which is open-source and freely available.

To help put this course in context, we also mention (and provide references to) courses on related topics taught at ENSTA Paris.

1.2 EXAMPLES INVOLVING LINEAR SYSTEMS. In this section, we provide a motivation through a series of examples. Our background and field of activity being primarily concerned with the mathematical and computational modelling of systems arising in mechanics (deformable solids, fluids) and more generally in physics (e.g. acoustics, electromagnetism), these motivating exemples are certainly to some extent biased. Mechanics and physics usually rely on known mathematical models describing the underlying physics, often based on ODEs or PDEs, and in this context numerical linear algebra is heavily involved in solution methods for those models, based on discretization (finite elements, boundary elements, finite differences, discontinuous Galerkin methods...). Other situations involve models that are at least partially unknown, the missing knowledge on them being sought from experimental data: this in particular includes *inverse problems* and in many forms (e.g. data assimilation for weather forecast, seismic inversion). Finally, many other applications (e.g. image processing, statistical and data analysis) do not rely on *a priori* chosen mathematical models, but rather attempt to find *ad hoc* models, often without direct physical meaning, that correlate, explain, or allow predictions from, available data. All those tasks, and many more, rely on computation, and numerical linear

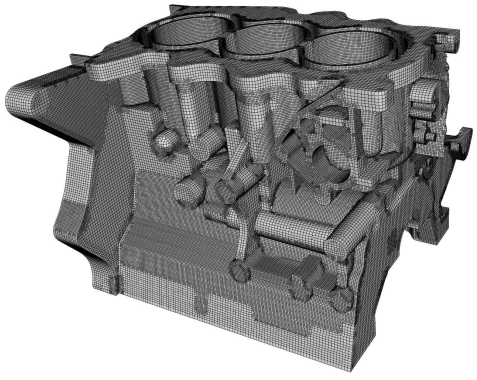

Figure 1.1: *Finite element mesh of automotive engine part (from the website of INRIA Gamma 3 team).*

algebra is among the key methodological ingredients allowing to achieve those varied goals. Data sets or physical unknowns often have very high dimension, and tasks such as solving large linear systems of equations often occur repeatedly in the overall process, hence the importance of computational efficiency.

1.2.1 Finite element analysis of mechanical structures. This example is both quite classical and very relevant to many sectors of industry (e.g. aerospace, automotive, energy production, civil engineering, biomechanics) where very large models may be solved. Upon finite element (FE) approximation [11, 3, 5] of the elastic equilibrium equations in weak form by means of the Galerkin method, and assuming usual small-strain linearly-elastic constitutive behavior, the unknown kinematic degrees of freedom gathered in a vector U solve the linear system

$$KU = F \tag{1.1}$$

where K is the *stiffness matrix* K while F is the vector of generalized loads obtained by evaluating the virtual work of the applied loads for each basis (test) function of the finite element approximation space.

- Here, the governing matrix K is (i) symmetric, (ii) positive definite[1] and (iii) banded. Therefore K is in particular *sparse*: for a very large FE model, the nonzero entries of K represent only a small fraction of its population.
- Well-known energy principles of mechanics show that (1.1) alternatively expresses that U minimizes the (FE-discretized) potential energy

  $$E_{\text{pot}}(U) := \tfrac{1}{2}U^{\mathrm{T}}KU - U^{\mathrm{T}}F, \tag{1.2}$$

  a remark that is strongly connected to certain *iterative solvers* applicable to the linear system (1.1).

1.2.2 Structural dynamics and modal analysis. Eigenvalue problems are another frequent component of engineering mechanics (in particular). For example, the free vibrations of an undamped elastic solid verify (using a finite element approximation)

$$KU - (2\pi f)^2 MU = 0$$

(where M is the *mass matrix* associated with kinetic energy), which is the *generalized eigenvalue problem* of finding frequencies f and nonzero displacements U (called vibration modes) solving the above equation. Those modes correspond to potential resonances (so need to be avoided under service loads); moreover they provide convenient basis functions for obtaining low-dimensional models of dynamic response (modal projections).

[1] Assuming the problem, or the chosen FE approximation, prevents rigid-body motions, failing which K is positive but not invertible and additional precautions must be taken to fix the rigid-body motion contribution to the solution.

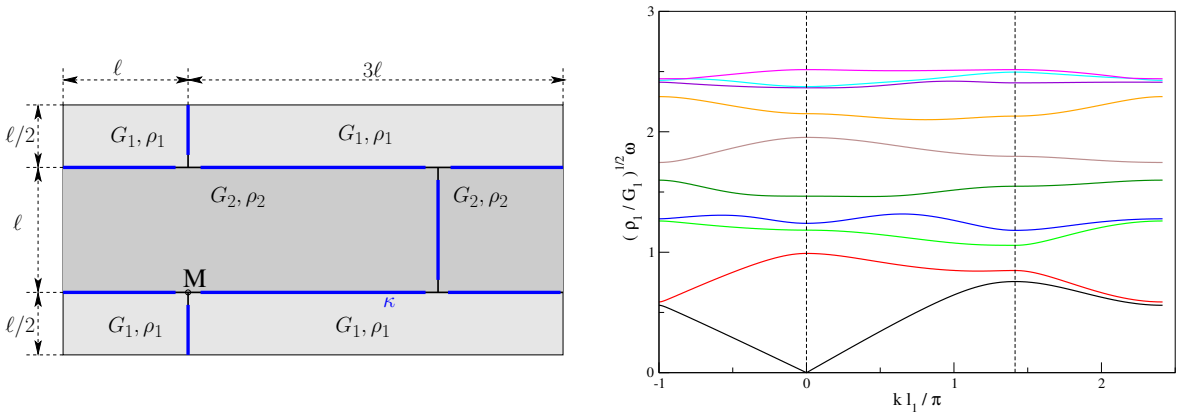

Figure 1.2: *Left: Periodicity cell Y for medium made of staggered bricks separated by compliant interfaces (in blue). The bricks are bonded by ligaments near the "triple points" such as M. Right: (wavenumber - frequency) computed dispersion diagram for that cell. Each line plot made of 601 points in wavenumber-frequency space, eachj point requires solving a separate eigenvalue problem. The FE model has 5690 fourth-order triangular finite elements (46426 nodal unknowns). Computations done in connection with research work presented in [21].*

- The mass matrix is a symmetric positive definite band matrix.

The computation of eigenvalues and eigenvectors can more generally have many motivations:

(i) Physical motivations: investigate the stability of dynamical responses under small perturbations (of systems or excitations), find the dispersion properties of complex materials under wave propagation (i.e. determine the wavenumber / frequency pairs that allow the propagation of free waves), predict potential structural instabilities such as buckling.

(ii) Computational and algorithmic motivations: evaluation of singular value decomposition of arbitrary matrices, of their spectral norm or condition number, estimation of convergence rates of iterative solvers.

To further illustrate (i), the question of whether a certain kind of free wave called *Bloch wave*, characterized by a vector-valued wavenumber $\boldsymbol{k} \in \mathbb{R}^3$, can propagate in a given complex (e.g. periodic) medium is answered by solving a generalized eigenvalue problem of the form (1.2) with the stiffness K replaced by a modified form $\hat{K}(\boldsymbol{k})$ that depends on the wavenumber. Then, solving $\hat{K}(\boldsymbol{k})U - (2\pi f)^2 MU = 0$ for all wavenumbers in a certain region of the $\boldsymbol{k}$-space (producing eigenfrequencies $f_1(\boldsymbol{k}), f_2(\boldsymbol{k}), \ldots$) yield *dispersion diagrams* (which plot $f_i(\boldsymbol{k})$ as functions of $\boldsymbol{k}$) that play an important role in the physics of wave propagation in complex media. See Fig. 1.2 for an illustration.

1.2.3 Boundary element solution of wave scattering. Wave propagation simulation problems often assume the idealization of an unbounded propagation medium exterior to the object that radiates or scatters (e.g. acoustic, electromagnetic or elastic) waves. Applications include noise generation or furtivity studies in the aerospace industry. Such computations are often performed on the bssis of a *boundary element method* (BEM) resulting from the discretization of a boundary integral equation (BIE) governing the (acoustic, electromagnetic, elastic) wave propagation problem. The latter has the typical form

$$\int_S G(\boldsymbol{x}, \boldsymbol{y}) u(\boldsymbol{y}) \, \mathrm{d}S(\boldsymbol{y}) = f(\boldsymbol{x}) \tag{1.3}$$

where u is the main unknown on the surface S (e.g. the acoustic pressure field), f is the data (e.g. related to the normal velocity of a pulsating surface) and $G(\boldsymbol{x}, \boldsymbol{y})$ is the (known) acoustic Green's function. We refer to e.g. the course [2] for a detailed exposition of BIE theory and BEMs.

- Upon discretization, the BIE (1.3) yields a linear system of the form $GU = F$ where $U \in \mathbb{C}^n$ (unknown), $F \in \mathbb{C}^n$ (data) and $G \in \mathbb{C}^{n\times n}$ (BEM influence matrix). The matrix G is *dense*, and is amenable to block-wise low-rank approximation (for example using the hierarchical matrix method, as in [9])

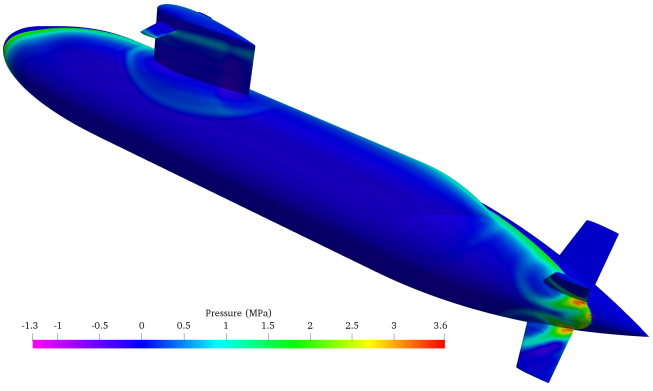

Figure 1.3: *Scattering of fluid pressure wave by a rigid motionless submarine: total pressure field on the surface, 5.4 milliseconds after the incident wave first hits the submarine. The boundary element mesh has about $3\,10^6$ pressure unknowns. From [25]*

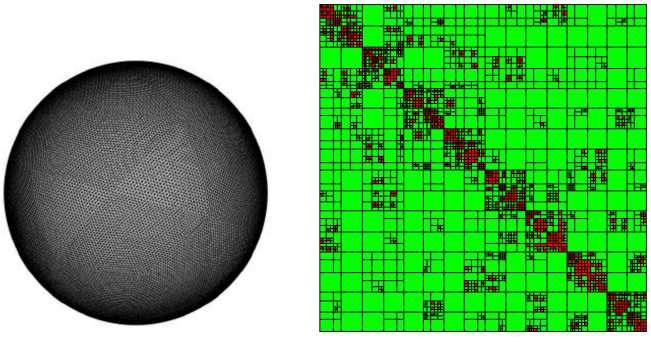

Figure 1.4: *Hierarchical matrix blockwise approximation of a BEM matrix. From [9]*

1.2.4 Solution of non-linear equations. Many models in mechanics and physics require (after a finite-dimensional approximation process) to solve non-linear equations of the general form:

$$\text{Find } U \in \mathbb{R}^n, \qquad \mathcal{F}(U) = 0$$

where (usually) $\mathcal{F}(U)$ is a $\mathbb{R}^n \to \mathbb{R}^n$ mapping[2]. If $\mathcal{F}$ is differentiable, many solution algorithms follow the iterative Newton-Raphson approach whereby a first-order Taylor expansion of $\mathcal{F}$ about the current iterate U_k allows to define the next iterate U_{k+1} from setting to zero the linearized approximation of $\mathcal{F}(U_{k+1})$:

$$\mathcal{F}(U_{k+1}) \approx \mathcal{F}(U_k) + K_k(U_{k+1} - U_k) \quad \Longrightarrow \quad \text{find } U_{k+1},\ K_k U_{k+1} = K_k U_k - \mathcal{F}(U_k). \tag{1.4}$$

where $K_k := \nabla\mathcal{F}(U_k)$ is often called the *tangent matrix* (or, in solid mechanics, the *tangent stiffness*). The above process is started from some initial guess U_0. Each iteration entails solving a linear system of equations governed by the tangent matrix K_k. For instance, with reference to (1.2), nonlinearly-elastic materials may be described by a non-quadratic potentiel energy of the form $E_{\text{pot}}(U) = \Phi(U) - U^{\mathrm{T}}F$,

[2] For example, we simply have $\mathcal{F}(U) = KU - F$ in the linear elastic FE case.

whose stationarity equation for a given load level E has the form $\mathcal{F}(U) := \nabla_U \Phi(U) - F = 0$ and is no longer linear in U.

- The structure and main characteristics of the tangent matrix K_k are problem-dependent. For many non-linear models arising from mechanical FE models, K_k can be considered as being the stiffness of a fictitious heterogeneous linear-elastic material whose local material parameters depend on the current solution iterate, so shares many of the properties (SPD mand matrix) of standard stiffness matrices.

Nonlinear solution algorithms based on Newton-Raphson iterations of the general form (1.4) are widely implemented in industry engineering mechanics FE codes (such as Abaqus or `code_aster`) for performing a wide variety of non-linear analyses involving material plasticity, frictional contact[3], buckling and many other phenomena.

1.2.5 Identification, inverse problems. Inverse problems are "indirect measurement" situations where quantitative information about some hidden physical variable is sought by exploiting measurements of other, related, physical variables. Classical examples include imaging subterranean media from surface accelerometry data and tumor detection by elastography.

By way of illustration, we briefly consider temperature reconstruction problems that are relevant in space industry: knowing the distribution of temperature on a vehicle after landing, infer temperatures of outer skin during atmospheric reentry phase. This is an instance of the *backwards heat conduction* problem (BHCP): knowing the temperature distribution $x \mapsto \theta(x,T)$ in a conducting body at time $T > 0$, can we infer the distribution $x \mapsto \theta(x,0)$ at an earlier time (say $t = 0$)? The measurement $\theta(\cdot,T)$ and the unknown $\theta(\cdot,0)$ can be shown to obey a linear relationship of the form

$$\theta(\cdot,T) = \mathcal{A}(T)\theta(\cdot,0) \quad \Longrightarrow \quad \text{(discretization) } \Theta(T) = A(T)\Theta(0) \tag{1.5}$$

where the linear operator $\mathcal{A}(T)$ results from the chosen linear heat diffusion physical model. Upon discretization, one obtains a linear system, where the matrix $A(T)$ is dense. The BHCP is known to be ill-posed [8], making its discretized counterpart ill-conditioned: temperature reconstructions with acceptable accuracy are very hard to obtain using "naive" linear algebra methods as soon as the data is even slightly noisy, as illustrated in Fig. 1.5 on a simple spatially 1D configuration. Less-naive approaches designed for ill-conditioned linear systems such as (1.5) will be presented in Chap. 6.

1.2.6 Image processing. A frequent issue in image processing is to "restore", or otherwise improve, images of insufficient quality. For instance, Earth-based telescopes take astronomical images of celestial objects which may be blurred by atmospheric turbulence (through which light propagates before reaching the telescope mirror) or local light pollution. Such problems can often be mathematically formulated as *deconvolution* problems, the imperfect image $\hat{f}$ being related to the perfect image f by

$$\hat{f}(x) = \int_Y k(x-y) f(y)\,\mathrm{d}y \qquad x \in Y$$

(Y being the set of all pixels and the function $y \mapsto f(y)$ representing for example a gray or color level at pixel y), where the *convolution kernel* k models measurement imperfections (due to e.g. the apparatus or environmental conditions). For instance, Gaussian kernels $k(z) = A\exp(-z^2/2b)$ are used to model atmospheric blurring (b is tuned according to the blur severity, and C is a normalization constant). It is well known that k converges (in the distributional limit sense) to the Dirac mass δ as $b \to 0$ (the no-blur limit), in which case the above convolution yields $\hat{f} = f$.

Upon discretization, deblurring a measured image translates into solving the system

$$\hat{F} = KF \qquad F = \{f(y_1),\dots,f(y_n)\}^{\mathrm{T}}, \quad \hat{F} = \{\hat{f}(y_1),\dots,\hat{f}(y_n)\}^{\mathrm{T}}, \quad K_{ij} = \Delta S_j k(x_i - y_j)$$

where ΔS_j is the (small) surface area of the j-th pixel. It turns out that such linear systems are highly sensitive to small perturbations in the data $\hat{F}$, which makes image deblurring a difficult computational problem (and a subject of intense current research). See Figure 1.6 for an illustrative example.

[3] where complications appear due to the non-smooth character of unilateral contact conditions.

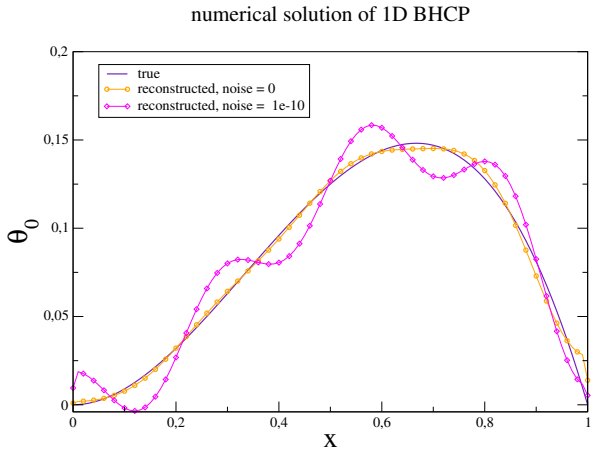

Figure 1.5: *1D reconstruction of initial temperature (at time $t = 0$) from noise-free or slightly noisy data. Observe the quite noticeable degradation in identification caused on the reconstruction by a very slight (simulated) data noise. Observation equation (1.5) solving by (unsuitably) naive methods. Data is temperature at times $t = O(5\,10^3)$s assuming material diffusivity is that of steel.*

Figure 1.6: *Example of image deblurring: image before degradation (left), degraded image (middle), deblurred image (right)*

- The matrix K is fully-populated but has a low numerical rank (it can be well approximated by a low-rank matrix). It is strictly speaking invertible, but very ill-conditioned. This topic will be revisited in Chap. 6

1.2.7 Correlation analysis. Consider a data set $D = \left\{ y_i, x_{i1}, \ldots, x_{in} \right\}_{i=1}^{m}$ (where each of the m individuals, numbered $i = 1$ to $i = m$, is described by a set of explanatory variables, or *predictors*, $x_1, \ldots, x_n$ and a dependent variable y). Assume that it is desired to find a simple (linear) mathematical relation allowing to predict, on the basis of this data, the value of the dependent variable y for any other individual knowing its predictors $x_1, \ldots, x_n$. Hence, we want to find the "best" model of the form

$$\hat{y} = a^{\mathrm{T}} x + b,$$

which requires finding optimal values for the parameters $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ of the linear approximation. A common approach is to seek a, b such that the model prediction $\hat{y}$ is closest to y in an average sense for the data set D, with the average taken as the quadratic mean. In other words:

$$\text{Find } a \in \mathbb{R}^n,\ b \in \mathbb{R} \quad \text{such that} \quad \sum_{i=1}^{n} |y_i - x_i^{\mathrm{T}} a - b|^2 \to \text{minimum}$$

Setting $X = [x_{ij}]$ $(1 \le i \le m,\ 1 \le j \le n)$, this can be recast in matrix terms as a *least squares problem*:

$$\text{Find } a \in \mathbb{R}^n,\ b \in \mathbb{R} \quad \text{such that} \quad \|Xa - (y-b)\|_2^2 \to \text{minimum}$$

Solving this type of problem also rests on numerical linear algebra methods, as we will see.

1.2.8 Learning. Learning methods aim in particular at defining *learning functions* $\mathcal{F}(x)$ that evaluate a classification $y = \mathcal{F}(x)$ associated with some "raw data" x. For example, x is a numerical vector encoding an image, and y is a classification output (e.g. $y = \mathcal{F}(x)$ encodes whether the image x shows a house, a boat, something else?). Such learning functions are for example constructed by composing affine maps $Lx := x \mapsto Ax + b$ and nonlinear functions N, with those steps applied several times (known as *layers*): $\mathcal{F}(x) = N\big(L\big(N(L(\dots Lx))\big)\big)$. Part of the available data (e.g. a set of images) is used as *training data*: A, b (and possibly other parameters) are tuned using optimization methods so that $\mathcal{F}$ computes on the training data the correct (known) classification output; this is the "learning" part. Then, $\mathcal{F}$ can be applied to new data to do actual classification.

Numerical linear algebra is involved in both the definition of the layers of the learning function and the computational methods (in particular optimization methods, which themselves rely heavily on linear algebra) used for tuning weights such as A, b. See [33] (and the references therein) for an introductory exposition together with the accompanying background on linear algebra, optimization, statistics.

1.3 FINITE-PRECISION COMPUTATION.

Despite its paramount importance, we will only briefly address the topic of finite-precision computation, as it is treated in much more detail in the companion course [24]. Numerical computing is based on the well-known *floating-point representation* of numbers, of the form

$$x = s\, m\, b^e$$

"Big-endian" and "Little-endian"

where s is the sign of x, m its mantissa, b the basis (normally $b = 2$) and e the exponent. Double-precision real numbers occupy 64 bits (1 for the sign of x, 52 for the mantissa, 11 for $|e|$). The mantissa is such that $1 \le m < 2$ (it has a leading implicit bit set to 1). A few observations immediately come to mind:

- Numbers are in general subject to *roundoff error*;
- Roundoff errors are relative to orders of magnitude; in particular, the numbers described by the floating point representation are not spaced evenly.
- For this reason, estimating *relative* errors on evaluations and algorithms makes more sense than estimating *absolute* errors.

The widely-used IEEE 754 norm for representing floating point numbers and computing with them ensures that

$$\text{for all } x \in \mathbb{R}, \text{there exists } \ \varepsilon,\ |\varepsilon| < \varepsilon_{\text{mach}} \quad \text{such that } \ \text{fl}(x) = x(1+\varepsilon)$$

where $x \mapsto \text{fl}(x)$ is the operator converting a number to its floating point representation and $\varepsilon_{\text{mach}}$, often called the "machine epsilon", is the maximum modulus of relative round-off error ($\varepsilon_{\text{mach}} \approx 10^{-16}$ for double-precision floating-point reals). That norm also guarantees that

$$\text{for all } x, y \in \mathbb{F}, \quad x \circledast y = \text{fl}(x \star y),$$

where $\star$ is one of the operations $+, -, \times, /, \sqrt{}$, $\circledast$ is the floating-point implementation of that operation and $\mathbb{F}$ is the set of all floating-point numbers, as well as

$$\text{for all } x, y \in \mathbb{F}, \text{there exists } \ \varepsilon,\ |\varepsilon| < \varepsilon_{\text{mach}} \quad \text{such that } \ x \circledast y = (x \star y)(1+\varepsilon) \tag{1.6}$$

Error analysis of computational algorithms (regarding round-off errors) basically consists in expressing each floating-point operation using (1.6) and keeping track of the effect of roundoff errors ε on the algorithm.

1.4 VECTORS, MATRICES, NORMS.

1.4.1 Matrices. Matrices are arrays of (real or complex) numbers that allow to express linear relationships between elements of finite-dimensional vector spaces.

Vector space. We begin by briefly recalling that vector spaces are sets of objects (called *vectors*) for which certain operations are defined, namely the addition of vectors and the multiplication of a vector by a scalar (i.e. a number). For all usual applications of numerical linear algebra, those scalars are either real or complex numbers, and we will use generically the notation $\mathbb{K}$ to refer to the set of scalars (with either $\mathbb{K}=\mathbb{R}$ or $\mathbb{K}=\mathbb{C}$). A set E is a vector space (also called a linear space) provided the following axioms are verified:

1. Addition of vectors is commutative:	$x+y=y+x$
2. Addition of vectors is associative:	$x+(y+z)=(x+y)+z$
3. There exists a zero vector:	$x+0=x$
4. Each vector has an opposite vector:	$x+(-x)=0$
5. Multiplication of scalars and with a vector are compatible:	$(\alpha\beta)x=\alpha(\beta x)$
6. Scalar multiplication has a unit element:	$1x=x$
7. Scalar multiplication is distributive w.r.t. scalar addition:	$(\alpha+\beta)x=\alpha x+\beta x$
8. Scalar multiplication is distributive w.r.t. vector addition:	$\alpha(x+y)=\alpha x+\alpha y$

To repeat, in this course we will only consider finite-dimensional vector spaces with $\mathbb{K}=\mathbb{R}$ or $\mathbb{K}=\mathbb{C}$. We will usually denote a n-dimensional vector space E on the scalars $\mathbb{K}$ by $\mathbb{K}^n$; this means that a basis of E is (perhaps implicitly) chosen, so that any vector x of E is given as n-uple of numbers: $x=(x_1,\ldots,x_n)\in\mathbb{K}^n$. In this representation, each basis vector e_k is the n-uple $e_k=(0,\ldots,0,1,0,\ldots,0)$ (with the unit coordinate at the k-th position), and any vector of E is given by

$$x=x_1e_1+\ldots+x_ne_n,\qquad (x_1,\ldots,x_n)\in\mathbb{K}^n$$

Matrices. A *matrix* $A\in\mathbb{K}^{m\times n}$ is a rectangular array of numbers a_{ij} $(1\le i\le m,\ 1\le j\le n)$, which in particular serves to represent the action of a linear mapping $\mathcal{A}:\mathbb{K}^n\to\mathbb{K}^m$ (after having chosen a basis in both spaces). Letting $x=(x_1,\ldots,x_n)\in\mathbb{K}^n$, its image $\mathcal{A}x=y=(y_1,\ldots,y_m)\in\mathbb{K}^m$ under $\mathcal{A}$ is given by

$$y_i=\sum_{i=1}^{n}a_{ij}x_j,\qquad \text{or}\qquad \begin{Bmatrix}y_1\\ \vdots\\ y_m\end{Bmatrix}=\begin{bmatrix}a_{11} & \ldots & a_{1n}\\ \vdots & & \vdots\\ a_{m1} & \ldots & a_{mn}\end{bmatrix}\begin{Bmatrix}x_1\\ \vdots\\ x_n\end{Bmatrix}\quad\text{i.e. }\boxed{y=Ax}\text{ in matrix notation}\qquad(1.7)$$

Some terminology and notation. We recall and list the definition of some important types of matrices, and the associated terminology:

- Matrices $A\in\mathbb{K}^{n\times n}$, having the same number n of rows and columns, are said to be *square*. Non-square matrices $A\in\mathbb{K}^{m\times n}$ $(m\neq n)$ are said to be *rectangular*.
- A matrix $A\in\mathbb{K}^{m\times n}$ with most of its entries equal to zero is said to be *sparse*[4]. A non-sparse matrix is called *dense*.
- The *transpose* A^{T} of a matrix $A\in\mathbb{K}^{m\times n}$ is such that $(A^{\mathrm{T}})_{ij}=A_{ji}$.
- The *conjugate transpose* A^{H} of a complex matrix $A\in\mathbb{C}^{m\times n}$ is such that $(A^{\mathrm{H}})_{ij}=\overline{A_{ji}}$, that is, $A^{\mathrm{H}}=\overline{A^{\mathrm{T}}}$. If A is real, $A^{\mathrm{T}}=A^{\mathrm{H}}$, of course.
- Square real matrices $A\in\mathbb{R}^{n\times n}$ that are equal to their transpose, i.e. verify $A^{\mathrm{T}}=A$, $a_{ji}=a_{ij}$, are said to be *symmetric*.

[4]There is no precise definition of "most". A frequent expectation is that the number of nonzero entries is at most of the same order as the number of rows or columns.

- Square complex matrices $A \in \mathbb{C}^{n\times n}$ that are equal to their conjugate transpose, i.e. verify $A^{\mathrm{H}} = A$, $a_{ji} = \overline{a_{ij}}$, are said to be *Hermitian*. Of course, Hermitian real matrices are simply symmetric matrices.
- Square matrices $A \in \mathbb{K}^{n\times n}$ are *symmetric positive definite* (SPD) if (i) $A^{\mathrm{H}} = A$, (ii) $x^{\mathrm{H}}Ax > 0$ for all $x \in \mathbb{K}^n$, $x \neq 0$ (Hermitian symmetry implies that $x^{\mathrm{H}}Ax \in \mathbb{R}$ for any $x \in \mathbb{K}^n$).
- Square matrices $A \in \mathbb{K}^{n\times n}$ such that $A^{\mathrm{H}}A = I$ are *unitary* (or *orthogonal* in $A \in \mathbb{R}^{n\times n}$). They verify $A^{-1} = A^{\mathrm{H}}$ and $(A^{\mathrm{H}})^{-1} = A$.

We also list here a few notation conventions that will be used throughout without much further warning:

- We adopt the usual convention that vectors are column vectors, their (conjugate) transpose being row vectors; this is consistent with the usual rules of matrix algebra such as the matrix-vector product $y = Ax$ appearing in (1.7), and also with conventions used in e.g. MATLAB. Scalar products, for example, are hence written as
$$(x, y) = x^{\mathrm{T}}y \quad \text{(for real vectors)}, \qquad (x, y) = x^{\mathrm{H}}y \quad \text{(for complex vectors)}.$$
while, if $x \in \mathbb{K}^m$, $y \in \mathbb{K}^n$, $xy^{\mathrm{H}} \in \mathbb{K}^{m\times n}$ is a rank-1 matrix.
- Vectors are denoted with lowercase letters, e.g. x, and (in context) x_i is the i-th entry of x.
- Matrices are denoted with uppercase letters, e.g. A, and (in context) a_{ij} is the (i, j)-th entry of A.
- A MATLAB-like colon ":" serves to define submatrices by their index ranges. For example, we write
$$A_{k:\ell,p:q} := [a_{ij}]_{k\leq i\leq \ell,\, p\leq j\leq q} \quad \text{(rectangular submatrix of } A\text{)},$$
$$A_{k:\ell,p} := [a_{ip}]_{k\leq i\leq \ell} \quad \text{(part of the } p\text{-th row of } A\text{)}$$

1.4.2 Vector norms.. In numerical linear algebra, as in many other areas, it is essential to measure the "magnitude" (and "smallness", "largeness"...) of objects such as vectors or matrices (for instance, the convergence of an algorithm may be defined in terms of errors on the solution becoming "increasingly small"). Such magnitudes are measured using (vector, matrix) norms. Regarding (for now) vectors, any vector norm $\|\cdot\|$ must satisfy the usual defining properties of a norm, namely:

$$\begin{aligned} &\text{zero norm:} && \|x\| = 0 \ \text{ if and only if } \ x = 0, \\ &\text{positive homogeneity:} && \|\lambda x\| = |\lambda|\|x\| \quad \text{for any } \lambda \in \mathbb{K} \qquad \text{for all } x \in \mathbb{K}^n \\ &\text{triangle inequality:} && \|x + y\| \leq \|x\| + \|y\| \end{aligned} \tag{1.8}$$

Common vector norms include

$$\|x\|_1 := \sum_{i=1}^{n} |x_i|, \qquad \|x\|_2 := \Big(\sum_{i=1}^{n} |x_i|^2\Big)^{1/2}, \qquad \|x\|_p := \Big(\sum_{i=1}^{n} |x_i|^p\Big)^{1/p}, \qquad \|x\|_\infty := \max_{1\leq i\leq n} |x_i|$$

in particular, $\|\cdot\|_2$ is the usual Euclidean norm. All vector norms are equivalent: for any pair $\|\cdot\|_\alpha, \|\cdot\|_\beta$ of norms equipping $\mathbb{K}^n$, there exist constants $C_1 \leq C_2$ (depending only on the choice of norms and on the dimension n) such that

$$C_1\|x\|_\alpha \leq \|x\|_\beta \leq C_2\|x\|_\alpha \quad \text{for all } x \in \mathbb{K}^n, \tag{1.9}$$

and, in particular, we have

$$\begin{aligned} &\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2 \\ &\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty \qquad \text{for all } x \in \mathbb{K}^n. \\ &\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty \end{aligned}$$

Notice that the upper equivalence constant blows up in all three cases as $n \to \infty$, reminding us that equivalence no longer holds in the infinite-dimensional limit (of e.g. linear spaces of sequences [7]).

Norms are often used to evaluate estimates (usually upper bounds), for example to show that a vector is "small", or remains bounded, under some computational process. Such uses often rely on classical inequalityes, chiefly the Cauchy-Schwarz inequality

$$|x^{\mathrm{H}}y| \le \|x\|_2\|y\|_2,$$

which is a special case of the more-general Hölder inequality

$$|x^{\mathrm{H}}y| \le \|x\|_p\|y\|_q \quad \text{with} \quad \frac{1}{p}+\frac{1}{q}=1,\ 1<p,q<\infty.$$

1.4.3 Matrix norms. Matrix norms may be defined as norms induced by vector norms[5]: for any $p>0$, the p norm $\|A\|_p$ of $A\in\mathbb{K}^{m\times n}$ is defined by

$$\|A\|_p := \max_{x\in\mathbb{K}^n, x\neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p=1} \|Ax\|_p, \tag{1.10}$$

and thus provides the best upper bound on matrix-vector products: for any matrix A and vector x, we have

$$\|Ax\|_p \le \|A\|_p\|x\|_p,$$

with at least one vector x achieving equality in the above inequality[6]. In fact, one may equally well define the q-norm of a matrix induced by the vector p-norm:

$$\|A\|_{p,q} := \max_{x\in\mathbb{K}^n, x\neq 0} \frac{\|Ax\|_q}{\|x\|_p} = \max_{\|x\|_p=1} \|Ax\|_q,$$

Another frequently used norm is the Frobenius norm $\|\cdot\|_{\mathrm{F}}$ defined by

$$\|A\|_{\mathrm{F}} := \Big(\sum_{i,j}|a_{ij}|^2\Big)^{1/2} = \sqrt{\mathrm{Tr}(AA^{\mathrm{H}})} \tag{1.11}$$

(where A^{H} is the Hermitian transpose of A). The Frobenius norm extends to matrices the Euclidean norm of vectors[7]; it is not an induced norm. Why?

The Frobenius norm (1.11) and all induced matrix p-norms (1.10) are *sub-multiplicative*: they all satisfy

$$\|AB\| \le \|A\|\,\|B\| \tag{1.12}$$

The terminology "matrix norm" is sometimes restricted to norms of matrices that do verify the sub-multiplicativity requirement (1.12). While not part of the basic defining properties (1.8) of a norm, sub-multiplicativity is very useful as it permits to derive estimates (upper bounds) of quantities involving matrix multiplications.

Matrix norms are all equivalent, i.e. obey inequalities of the form (1.9); in particular we have[8]

$$\begin{aligned} \tfrac{1}{\sqrt{m}}\|A\|_1 \le \|A\|_2 \le \sqrt{n}\|A\|_1 \\ \tfrac{1}{\sqrt{n}}\|A\|_\infty \le \|A\|_2 \le \sqrt{m}\|A\|_\infty \\ \tfrac{1}{\sqrt{\min(m,n)}}\|A\|_{\mathrm{F}} \le \|A\|_2 \le \|A\|_{\mathrm{F}} \end{aligned} \qquad \text{for all } A\in\mathbb{K}^{m\times n}$$

Notation convention for norms. The generic norm symbol $\|\cdot\|$ will stand for any unspecified vector norm, and indicate the *induced* matrix norm if evaluated on a matrix (e.g. $\|Ax\| \le \|A\|\,\|x\|$); in particular, a Frobenius norm will always be indicated explicitly (e.g. $\|A\|_{\mathrm{F}}$).

[5] In infinite-dimensional linear spaces, induced norms are usually called *operator norms*, see e.g. [7]

[6] This is true because the range of A, as a finite-dimensional subspace of $\mathbb{K}^m$, is closed, implying the existence of x maximizing $\|Ax\|_p$. If the matrix A is *normal*, i.e. satisfies $AA^{\mathrm{H}} = A^{\mathrm{H}}A$, such x is in fact any eigenvector for the eigenvalue of A having largest modulus, see Sec. 5.

[7] Its infinite-dimensional counterpart in Hilbert spaces is known as the Hilbert-Schmidt norm

[8] noticing again the loss of equivalence in the infinite-dimensional limit

1.5 ACCURACY AND STABILITY. Generally speaking, many scientific computing tasks may be viewed as evaluating a function $\mathcal{F}$ on a given datum $x \in \mathcal{X}$ to obtain a desired result $y = \mathcal{F}(x)$, $y \in \mathcal{Y}$. The data space $\mathcal{X}$ and the solution space $\mathcal{Y}$ may be very diverse and complex (for instance, function spaces for which some partial differential equation is well-posed, see [7, 6]). The function $\mathcal{F}$, which we may call the solution operator, may itself be complicated (with nonlinear components, tests...) and defined only implicitly as a sequence of simpler operations (an algorithm).

In this course, we limit ourselves to the case where the data and solution spaces are finite-dimensional, and $\mathcal{F}$ may symbolize diverse computational tasks such as solving the linear system $Ay = b$ (A, b being the data and y the sought solution) or diagonalizing a symmetric matrix A (A being the data, and $x = (\lambda_i, x_i)$ the sought eigenvalue-eigenvector pairs). We will see for example (Sec. 5) that the latter problem cannot be solved by applying a predefined function to the data, and instead requires an iterative algorithm.

We assume that there is an exact problem to be solved (for example, find the solution of $Ay = b$ with A, b known exactly, i.e. without errors induced by e.g. finite machine precision). Exactness here does *not* pertain to a possible prior approximation methodology leading to the finite-dimensional problem being solved (for example, we treat the finite element approximation of a PDE as exact if the stiffness matrix A and load vector b are exact by virtue of prior computation not suffering from any numerical errors), and $y = \mathcal{F}(x)$ symbolizes this exact problem, where an "exact" solution y is found by applying an "exact" method $\mathcal{F}$ to an exact datum x.

In practice, only an imperfect version of the solution operator $\widetilde{\mathcal{F}}$ can be achieved[9], whereas the data x is approximated by floating-point numbers and incurs roundoff errors in the process. We let $x \mapsto \widetilde{\mathcal{F}}(x)$ denote the process of rounding off the data x, then applying to it the (imperfect) numerical solution method. To assess the reliability of the algorithm $\widetilde{\mathcal{F}}$ available in practice, it is natural to consider questions such as

$$\text{How close to } y = \mathcal{F}(x) \text{ is the approximation } \widetilde{y} := \widetilde{\mathcal{F}}(x)\,?$$

Accuracy. Since a good algorithm must approximate the exact problem well, one may for example consider the relative solution accuracy

$$e_{\text{rel}} = \frac{\|\widetilde{\mathcal{F}}(x) - \mathcal{F}(x)\|}{\|\mathcal{F}(x)\|}. \tag{1.13}$$

Under optimal conditions, the best possible accuracy is $e_{\text{rel}} \approx \varepsilon_{\text{mach}}$. Individual floating-point arithmetic operations do meet this goal.

Stability. To achieve $e_{\text{rel}} \approx \varepsilon_{\text{mach}}$ in (1.13) is however unrealistically demanding for most algorithms due to factors like large-scale computations or ill-conditioned problems. For these reasons, it is more appropriate to aim for *stability*: the algorithm $\widetilde{\mathcal{F}}$ for a problem $\mathcal{F}$ is deemed stable if

$$\text{for each } x \in \mathcal{X},\ \frac{\|\widetilde{\mathcal{F}}(x) - \mathcal{F}(\widetilde{x})\|}{\|\mathcal{F}(\widetilde{x})\|} = O(\varepsilon_{\text{mach}}) \qquad \text{for some } \widetilde{x} \text{ such that } \frac{\|x - \widetilde{x}\|}{\|x\|} = O(\varepsilon_{\text{mach}}). \tag{1.14}$$

This can be stated informally as: a stable algorithm must yield nearly the right answer if given a nearly correct data (with "nearly" meant in the relative sense). It turns out that many algorithms of numerical linear algebra satisfy a stability condition that is stronger than (1.14), known as *backward stability*:

$$\text{for each } x \in \mathcal{X},\ \widetilde{\mathcal{F}}(x) = \mathcal{F}(\widetilde{x}) \qquad \text{for some } \widetilde{x} \text{ such that } \frac{\|x - \widetilde{x}\|}{\|x\|} = O(\varepsilon_{\text{mach}}). \tag{1.15}$$

Definition (1.15) is stronger than (1.14) in that the first $O(\varepsilon_{\text{mach}})$ is replaced by zero. Stated informally, (1.15) says that a backward stable algorithm must yield exactly the right answer for some nearly

[9] For instance, $\widetilde{\mathcal{F}}$ is a finite-precision method for solving $Ay = b$, yielding an approximate solution y even for exact data A, b.

correct data. As we will see, algorithms of numerical linear algebra are often assessed in terms of their backward stability.

The notation $O(\varepsilon_{\text{mach}})$ in (1.14) and (1.15) (and similarly in the remainder of these course notes) means that the quantity on the left-hand side is less than $C\varepsilon_{\text{mach}}$, for some fixed constant $C > 0$, whenever $\varepsilon_{\text{mach}}$ is small enough. A given machine has a fixed (small) value for $\varepsilon_{\text{mach}}$ while the $O(\cdot)$ notation is mathematically defined with reference to a limiting process. The two aspects can be reconciled by envisioning a (virtual) family of computers with smaller and smaller $\varepsilon_{\text{mach}}$.

1.6 CONDITIONING, CONDITION NUMBER. The forward and backward errors may be connected by considering the *relative sensitivity* $\varrho(\mathcal{F}; x, \widetilde{x})$, which is the ratio of forward and backward errors:

$$\varrho(\mathcal{F}; x, \widetilde{x}) := \frac{\|\widetilde{\mathcal{F}}(x) - \mathcal{F}(x)\|}{\|\mathcal{F}(x)\|} \Big(\frac{\|\widetilde{x} - x\|}{\|x\|} \Big)^{-1} = \frac{\|\mathcal{F}(\widetilde{x}) - \mathcal{F}(x)\|}{\|\mathcal{F}(x)\|} \frac{\|x\|}{\|\widetilde{x} - x\|}$$

(note that we have used the defining equality $\widetilde{\mathcal{F}}(x) = \mathcal{F}(\widetilde{x})$ of backward stability, see (1.15)). The *condition number* $\kappa = \kappa(\mathcal{F}; x)$ of $\mathcal{F}$ at $x \in \mathcal{X}$ is then defined as the limiting value of the relative sensitivity $\varrho(\mathcal{F}; x, \widetilde{x})$ in the limit of vanishing data perturbations:

$$\kappa(\mathcal{F}; x) := \lim_{\delta \to 0} \sup_{\|\widetilde{x} - x\| \le \delta} \frac{\|\mathcal{F}(\widetilde{x}) - \mathcal{F}(x)\|}{\|\mathcal{F}(x)\|} \frac{\|x\|}{\|\widetilde{x} - x\|}. \tag{1.16}$$

If the solution operator $\mathcal{F}$ is regular enough, the condition number (1.16) is given by the more-explicit formula

$$\kappa(\mathcal{F}, x) = \frac{\|\mathcal{F}'(x)\| \, \|x\|}{\|\mathcal{F}(x)\|},$$

where $\mathcal{F}'(x)$ is the Fréchet derivative[10] of $\mathcal{F}$ at x (which is assumed to exist). Most often no ambiguity arises as to the solution operator $\mathcal{F}$ and data point x being considered, and we will for short write κ rather than $\kappa(\mathcal{F}, x)$.

Interpretation. As said before, κ is the (limit for small data perturbations of the) ratio between relative solution errors (a.k.a. forward errors) and relative data errors (a.k.a. backward errors). Notice in particular that, by construction, κ is a *dimensionless* number (i.e. it has no physical units, even though the data or solution, or both, may have physical units); this implies that κ is deemed small (resp. large) if $\kappa \ll 1$ (resp. $\kappa \gg 1$). A solution process $\mathcal{F}$ is said to be *well-conditioned* if $\kappa = O(1)$ (relative solution error of the same order as relative data error), and *ill-conditioned* if $\kappa(\mathcal{F}) \gg 1$ (relative solution error much larger than relative data error). To be sound and reliable, computational methods must therefore be well-conditioned.

1.7 CONDITION NUMBER OF LINEAR SYSTEMS. We now derive and illustrate the condition number of the computational task of fundamental interest for this course, namely solving a linear system of equations. We focus on the sensitivity of the solution to data errors, the exposition of solution methods being deferred to Chapter 2 and 4. Accordingly, consider the linear system

$$Ay = b, \tag{1.17}$$

with data $x = (A, b)$ and (unique) solution y, A being assumed to be a square invertible matrix. Consider also a perturbed version

$$(A + E)(y + z) = b + f \tag{1.18}$$

[10] A function $\mathcal{F} : \mathcal{X} \to \mathbb{R}$ is *differentiable* at $x \in \mathcal{X}$ if there exists a continuous linear functional $\mathcal{F}'(x) \in \mathcal{X}'$, called the Fréchet derivative of $\mathcal{F}$ at x, such that $\mathcal{F}(\widetilde{x}) - \mathcal{F}(x) = \langle \mathcal{F}'(x), \widetilde{x} - x \rangle + o(\|\widetilde{x} - x\|)$ for any $\widetilde{x}$ in a neighborhood of x in $\mathcal{X}$; $\mathcal{X}$ may be any normed vector space (with $\mathcal{X}'$ its topological dual). In the case of present interest where $\mathcal{X}$ is finite-dimensional, the Fréchet derivative can be represented as a vector (the gradient $\nabla \mathcal{F}(x)$ of $\mathcal{F}$ at x) and we have $\mathcal{F}(\widetilde{x}) - \mathcal{F}(x) = [\nabla \mathcal{F}(x)]^{\mathrm{T}}(\widetilde{x} - x) + o(\|\widetilde{x} - x\|)$

of system (1.17), where E, f are perturbations of the matrix and right-hand side (i.e. the absolute backward errors) and z is the induced perturbation on the solution (i.e. the absolute forward error). To examine how z is linked to (E, f), we use (1.17) and (1.18) and find that z solves the system

$$(A+E)z = f - Ey \qquad \text{with} \quad y = A^{-1}b$$

We assume the matrix perturbation to be small enough to have $\|A^{-1}\|\|E\| < 1$; by a Neumann series argument, this guarantees in particular that $A+E$ is invertible, and we moreover have

$$(A+E)^{-1} = A^{-1}(I + EA^{-1})^{-1}, \qquad \|(A+E)^{-1}\| \le \frac{\|A^{-1}\|}{1 - \|EA^{-1}\|} \le \frac{\|A^{-1}\|}{1 - \|E\|\|A^{-1}\|},$$

so that the solution error may be estimated as

$$\|z\| = \|(A+E)^{-1}(f - Ey)\| \le \frac{\|A^{-1}\|}{1 - \|E\|\|A^{-1}\|}\Big(\|f\| + \|E\|\|y\| \Big).$$

We reformulate the above estimate in terms of the forward relative error $\|z\|/\|y\|$ and the backward relative errors $\|E\|/\|A\|$, $\|f\|/\|b\|$, to obtain

$$\frac{\|z\|}{\|y\|} \le \frac{\|A^{-1}\|\|A\|}{1 - \|E\|\|A^{-1}\|}\Big(\frac{\|f\|}{\|b\|}\frac{\|b\|}{\|A\|\|y\|} + \frac{\|E\|}{\|A\|} \Big) \le \frac{\|A^{-1}\|\|A\|}{1 - \|E\|\|A^{-1}\|}\Big(\frac{\|f\|}{\|b\|} + \frac{\|E\|}{\|A\|} \Big),$$

the last inequality resulting from $\|b\| \le \|A\|\|y\|$ as a consequence of $Ay = b$. Upper bounds on the relative sensitivity and its limiting value for small relative data errors $\|E\|/\|A\|$ and $\|f\|/\|b\|$ are therefore obtained as

$$\frac{\|z\|}{\|y\|}\Big(\frac{\|f\|}{\|b\|} + \frac{\|E\|}{\|A\|} \Big)^{-1} \le \frac{\kappa(A)}{1 - \kappa(A)\frac{\|E\|}{\|A\|}} = \kappa(A) + O\Big(\frac{\|E\|}{\|A\|} \Big), \qquad \text{with} \quad \kappa(A) := \|A^{-1}\|\|A\|.$$

Recalling the definition (1.16) of the condition number, $\kappa(A) = \|A^{-1}\|\|A\|$ is an upper bound of the condition number associated with solving $Ay = b$; it is called the *condition number of the matrix* A. Loosely speaking, $\kappa(A)$ is the multiplicative coefficient which, applied to the relative matrix error or the relative right-hand side error (or both), estimates the induced relative solution error. This concept is, obviously, absolutely essential in numerical linear algebra.

Properties. The following properties of $\kappa(A)$ and remarks must be kept in mind:

- We always have $\kappa(A) \ge 1$ (because $\|A^{-1}\|\|A\| \ge \|\|A^{-1}A\| = \|I\| = 1$ for any induced norm).
- The value of $\kappa(A)$ depends on the choice of (matrix) norm; if defined for the induced p-norm, we sometimes write it $\kappa_p(A)$.
- If the matrix A is normal (i.e. verifies $AA^{\mathrm{H}} = A^{\mathrm{H}}A$), it is unitarily diagonalizable (i.e. $A = Q\Lambda Q^{\mathrm{H}}$ for some unitary matrix Q). In this case, and for the choice $\|\cdot\| = \|\cdot\|_2$ of norm, we have $\|A\|_2 = |\lambda_{\max}|$ and $\|A^{-1}\|_2 = 1/|\lambda_{\min}|$, and hence

  $$\kappa_2(A) = |\lambda_{\max}|/|\lambda_{\min}|,$$

  where $\lambda_{\max}$ and $\lambda_{\min}$ are the eigenvalues of A with largest and smallest modulus (noting that invertibility of A requires $|\lambda_{\min}| > 0$). For instance, for the 2×2 (normal) matrix $A = \mathrm{diag}(1, \varepsilon)$, we have $\kappa(A) = 1/|\varepsilon|$ and A is ill-conditioned for small ε.
- For any orthogonal or unitary matrix Q, we have $\|Qx\|_2 = \|x\|_2$ for all $x \in \mathbb{K}^n$, implying that $\|Q\|_2 = 1$. Similarly, $\|Q^{-1}\|_2 = \|Q^{\mathrm{H}}\|_2 = 1$. Consequently, $\kappa_2(Q) = 1$.
- The condition number $\kappa_2(A)$ of an arbitrary matrix $A \in \mathbb{K}^{m\times n}$ is given in terms of either its *pseudo-inverse* or its *singular values*, see Secs. 3.4, 3.5 and Theorem 3.6.

A simple numerical example. Consider the following (symmetric) matrix A, whose inverse (as given below) is exact:

$$A = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix}, \qquad A^{-1} = \begin{bmatrix} 25 & 41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{bmatrix}$$

We consider perturbations f of b, or E of A, and evaluate the induced perturbations z on the solution y of $Ay = b$:

$$b = \begin{Bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{Bmatrix} \implies y = \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix},$$

$$f = \begin{Bmatrix} 0.1 \\ -0.1 \\ 0.1 \\ -0.1 \end{Bmatrix} \implies z = \begin{Bmatrix} 8.2 \\ -13.6 \\ 3.5 \\ -2.1 \end{Bmatrix}, \qquad \begin{matrix} E_{23} = 0.1 \\ E_{ij} = 0 \text{ otherwise} \end{matrix} \implies z \approx \begin{Bmatrix} -5.86 \\ -11.7 \\ -2.43 \\ -3.43 \end{Bmatrix}$$

In fact, the eigenvalues of A, numerically computed (see Sec. 5) as listed below in order of decreasing magnitude, allow (since A is symmetric) to evaluate its condition number:

$$(\lambda_1, \lambda_2, \lambda_3, \lambda_4) \approx (30.29,\ 3.858,\ 0.8431,\ 0.01015),, \qquad \kappa_2(A) = |\lambda_1|/|\lambda_4| \approx 2.98\,10^3$$

This makes A rather ill-conditioned (relative data errors being amplified about 3000-fold), especially given that its size is only 4.