Chapter 1

Introduction and basic tools

1. Course Description

This introductory course is designed to teach you fundamental topics in Numerical Linear Algebra. As described in the introduction to our textbook, numerical linear algebra is really a course in *applied* linear algebra, in which we learn (i) applied problems where techniques of linear algebra are used in real life (ii) and how to solve them numerically.

Over the course of this quarter, we will cover the following topics (not necessarily in this order):

- How to compute solutions of *well-posed* linear systems of equations $\mathbf{Ax} = \mathbf{b}$ using such as Gaussian elimination, LU factorization, and Cholesky factorization. We will also see a few examples of where such problems may arise in various applied mathematical research questions.
- How to compute approximate solutions of overdetermined linear system of equations $\mathbf{Ax} \approx \mathbf{b}$, such as one that may arise from linear least squares fitting problems. Primary topics include normal equations, orthogonal transformations, QR factorizations, and some popular orthogonalization methods such as Gram-Schmidt, and Householder transformations;
- How to find the eigenvector and eigenvalues in eigenvalue problems $\mathbf{Av} = \lambda\mathbf{v}$. Eigenvalue problems arise in a wide ranges of science and engineering fields and play a key role in the analysis of their stability for instance. Topics include numerical approaches that allow us to compute eigenvalues and eigenvectors (e.g., power iteration, Rayleigh quotient iteration, QR iteration).
- Singular value decomposition and its applications.
- Finally, how to solve linear systems of equations using iterative methods, which are usually more easily parallelizable than direct methods. This includes simple methods such as the Jacobi, Gauss-Seidel and Successive Over-relaxation methods, and more sophisticated Krylov methods including Conjugate Gradient, GMRES, Lanczos and Arnoldi methods.

In addition to these topics, we will also spend some time learning about stability issues that arise specifically from the numerical treatment of linear algebra problems, because numerical arithmetic (specifically, floating point arithmetic) is not exact. Finally, we will also have a mind towards the use of some of these algorithms in high-performance computing, and will discuss, when appropriate, issues such as storage, computing efficiency, and parallelization. For further information on that last topic, see the AMS 250 course.

In this first Chapter, we begin by reviewing some basics of linear algebra, and introduce some definitions that will be used throughout the course. Note that this is not meant to be a course on Linear Algebra, but rather, a very brief recap of material that you should already be familiar with and that is a prerequisite for the course (e.g. AMS 211 for instance, or any undergraduate Linear Algebra course). We will then continue by looking at the issues associated with floating point arithmetic and their potential effect on the stability of matrix operations.

2. Matrices and Vectors

See Chapter 1 of the textbook

2.1. Matrix and vector multiplications, component notations, inner and outer products

2.1.1. Basic operations and component notations Given an $m \times n$ matrix $\mathbf{A}$, it has m rows and n columns. Its components are expressed as $(\mathbf{A})_{ij} = a_{ij}$ where $i = 1, \dots, m$ spans the rows, and $j = 1, \dots, n$ spans the columns. The coefficients a_{ij} can be real or complex.

We can compute the product of $\mathbf{A}$ with a vector $\mathbf{x}$ as $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ has n entries and $\mathbf{b}$ has m entries. Written in component form, this is

$$\mathbf{Ax} = \mathbf{b} \rightarrow b_i = \sum_{j=1}^{n} a_{ij} x_j \text{ for } i = 1, \dots, m \tag{1.1}$$

This can also be interpreted as

$$\mathbf{b} = \begin{pmatrix} \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \\ \\ \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \dots + x_n\mathbf{a}_n \tag{1.2}$$

where the $\mathbf{a}_i$ are the column vectors of $\mathbf{A}$, showing that $\mathbf{Ax} = \mathbf{b}$ effectively expresses the vector $\mathbf{b}$ in the space spanned by the column vectors of $\mathbf{A}$.

Being a linear operation, multiplication of a vector by $\mathbf{A}$ has the following properties:

$$\mathbf{A}(\mathbf{x} + \mathbf{y}) = \mathbf{Ax} + \mathbf{Ay} \tag{1.3}$$

$$\mathbf{A}(\alpha\mathbf{x}) = \alpha\mathbf{Ax} \tag{1.4}$$

It is easy to show these properties using the component form (1.1) for instance.

We can also take the product of two suitably-sized matrices. Suppose $\mathbf{A}$ is an $m \times n$ matrix, and $\mathbf{B}$ is an $n \times l$ matrix, then their product $\mathbf{C} = \mathbf{AB}$ is a $m \times l$ matrix. Component-wise the product is expressed as

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \text{ for } i = 1, \dots, m \text{ and } j = 1, \dots, l \tag{1.5}$$

Another way of interpreting the matrix multiplication is column-by-column multiplication:

$$\mathbf{C} = \mathbf{AB} = \mathbf{A} \begin{pmatrix} \\ \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_l \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \mathbf{Ab}_1 & \mathbf{Ab}_2 & \dots & \mathbf{Ab}_l \\ \\ \end{pmatrix} \tag{1.6}$$

Recall that, in general, matrix multiplication does *not* commute so $\mathbf{AB} \neq \mathbf{BA}$.

2.1.2. Other basic matrix operations **The transpose** of a matrix $\mathbf{A}$ of size $m \times n$ is the $n \times m$ matrix $\mathbf{A}^T$ whose components are $(\mathbf{A}^T)_{ij} = a_{ji}$. Note the switch in indices, so the rows of $\mathbf{A}^T$ are the columns of $\mathbf{A}$, and vice versa.

For linear algebra with complex matrices, the notion of transpose must be replaced with that of the **adjoint**. **The adjoint** of a complex matrix $\mathbf{A}$ of size $m \times n$ is the $n \times m$ complex matrix $\mathbf{A}^*$, which is the *complex conjugate* of the transpose of $\mathbf{A}$, so that $(\mathbf{A}^*)_{ij} = a^*_{ji}$. Note that the following applies:

- The transpose and adjoint operations are nearly linear in the sense that $(\alpha\mathbf{A}+\beta\mathbf{B})^T = \alpha\mathbf{A}^T + \beta\mathbf{B}^T$ though one must be careful that $(\alpha\mathbf{A}+\beta\mathbf{B})^* = \alpha^*\mathbf{A}^* + \beta^*\mathbf{B}^*$.
- The transpose and adjoint of a product satisfy $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$ and $(\mathbf{AB})^* = \mathbf{B}^*\mathbf{A}^*$

A symmetric matrix satisfies $\mathbf{A} = \mathbf{A}^T$. The complex equivalent is a **Hermitian matrix**, which satisfies $\mathbf{A} = \mathbf{A}^*$.

2.1.3. Inner and outer products. It is sometimes useful to think of vectors as matrices. By default, vectors are thought of as column-vectors, so that a column-vector with m components is an $m \times 1$ matrix. However, it is also possible to consider row-vectors, where an m-component row-vector is a $1 \times m$ matrix. It is easy to see that we can create a row-vector by taking the transpose (or adjoint, for complex vectors) of a column vector.

With these definitions, we can then define the inner and outer products of column- and row-vectors as:

- The **inner product** is the product of a $1 \times m$ (row) vector $\mathbf{x}^T$ with a $m \times 1$ (column) vector $\mathbf{y}$, and the result is a scalar (a 1×1 matrix):

$$\mathbf{x}^T\mathbf{y} = \sum_{i=1}^{m} x_{1i}y_{i1} = \sum_{i=1}^{m} x_i y_i \tag{1.7}$$

  For complex vectors, this becomes

$$\mathbf{x}^*\mathbf{y} = \sum_{i=1}^{m} x_{1i}^* y_{i1} = \sum_{i=1}^{m} x_i^* y_i \tag{1.8}$$

- The **outer product** is the product of a $m \times 1$ (column) vector $\mathbf{x}$ with a $1 \times m$ (row) vector $\mathbf{y}^T$, and the result is an $m \times m$ matrix whose components are

$$(\mathbf{x}\mathbf{y}^T)_{ij} = x_{i1}y_{1j} = x_i y_j \tag{1.9}$$

  Again, for complex vectors this becomes

$$(\mathbf{x}\mathbf{y}^*)_{ij} = x_{i1}y_{1j}^* = x_i y_j^* \tag{1.10}$$

2.1.4. Range, Nullspace and Rank The matrix operation $\mathbf{x} \to \mathbf{A}\mathbf{x}$ is a function from $\mathbb{C}^n$ to $\mathbb{C}^m$, which, as we saw, returns a vector $\mathbf{A}\mathbf{x}$ that is a linear combination of the columns of $\mathbf{A}$. However, depending on $\mathbf{A}$, the operation $\mathbf{A}\mathbf{x}$ may generate vectors that only span a subspace of $\mathbb{C}^m$, rather than the whole of $\mathbb{C}^m$. We therefore define **the range** of the matrix $\mathbf{A}$ as the range of the function $\mathbf{A}\mathbf{x}$, namely the subspace of $\mathbb{C}^m$ spanned by all possible vectors $\mathbf{A}\mathbf{x}$ when $\mathbf{x}$ varies in the whole of $\mathbb{C}^n$. Having seen that $\mathbf{A}\mathbf{x}$ is necessarily a linear combination of the columns of $\mathbf{A}$, we have the theorem:

Theorem: *The range of* $\mathbf{A}$ *is the subspace spanned by the columns of* $\mathbf{A}$.

The nullspace of $\mathbf{A}$ is the set of all vectors $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0}$. Its dimension is often written null($\mathbf{A}$) and called the **nullity**.

And finally **the column rank** of $\mathbf{A}$ is the dimension of the space spanned by its column-vectors, and **the row rank** of a matrix is the space spanned by its row vectors. Even if the matrix is not square, *an important property of matrices is that the row rank and column ranks are always equal* (we will demonstrate this later), so we usually refer to them simply as the rank of the matrix. For a matrix $\mathbf{A}$ of size $m \times n$, we have that

$$\text{rank}(\mathbf{A}) + \text{null}(\mathbf{A}) = n, \tag{1.11}$$

i.e. the sum of the rank and the nullity is equal to the number of columns of $\mathbf{A}$.

A **full rank** matrix is a matrix that has maximal possible rank: if the matrix is $m \times n$, then the rank is equal to $\min(m, n)$. This implies that the nullity of a full rank matrix is $n - \min(m, n)$, and is 0 if the matrix has $m \geq n$ (square or tall matrices) but greater than 0 if the matrix has $m < n$ (wide matrices). Full rank matrices with $m \geq n$ therefore have an important property:

Theorem: *Given an $m \times n$ matrix $\mathbf{A}$ with $m \geq n$, it has full rank if and only if no two distinct vectors $\mathbf{x}$ and $\mathbf{y}$ exist such that $\mathbf{Ax} = \mathbf{Ay}$.*

The same is not true for matrices with $m < n$, however.

2.2. The inverse of a matrix

The notion of inverse of a matrix only makes sense when considering square matrices, which is what we now assume in this section.

A full rank square matrix of size $m \times m$ is invertible, and we denote the inverse of the matrix $\mathbf{A}$ as $\mathbf{A}^{-1}$. The inverse $\mathbf{A}^{-1}$ is also of size $m \times m$, and by definition, satisfies the following property:

$$\mathbf{Ax} = \mathbf{b} \Leftrightarrow \mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{1.12}$$

for any vector $\mathbf{x}$ in $\mathbb{C}^m$. By analogy with earlier, we therefore see that this expresses the vector $\mathbf{x}$ in the space spanned by the columns of $\mathbf{A}^{-1}$. Now, suppose we construct the m vectors $\mathbf{z}_i$ such that

$$\mathbf{z}_i = \mathbf{A}^{-1}\mathbf{e}_i \tag{1.13}$$

where the vectors $\mathbf{e}_i$ are the unit vectors in $\mathbb{C}^m$. Then by definition, $\mathbf{Az}_i = \mathbf{e}_i$. This implies

$$\begin{aligned}\mathbf{AZ} \quad &= \mathbf{A}\begin{pmatrix}\mathbf{z}_1 & \mathbf{z}_2 & \dots & \mathbf{z}_m\end{pmatrix} = \begin{pmatrix}\mathbf{Az}_1 & \mathbf{Az}_2 & \dots & \mathbf{Az}_m\end{pmatrix} \\ &= \begin{pmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_m\end{pmatrix} = \mathbf{I} \end{aligned} \tag{1.14}$$

where $\mathbf{I}$ is the identity matrix. In short, this proves that $\mathbf{AA}^{-1} = \mathbf{I}$, and it can easily be proved that this also implies $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$

There are many equivalent conditions for a matrix to be invertible (also called non-singular). These are, for instance:

Theorem: For any given square matrix of size $m \times m$, the following statements are equivalent:

- $\mathbf{A}$ has an inverse, $\mathbf{A}^{-1}$.
- $\text{rank}(\mathbf{A}) = m$ (the matrix has full rank)
- The range of $\mathbf{A}$ is $\mathbb{R}^m$ (or $\mathbb{C}^m$, for complex matrices)
- $\text{null}(\mathbf{A}) = 0$ (the nullspace is empty)
- 0 is not an eigenvalue of $\mathbf{A}$ (see later for definition)
- 0 is not a singular value of $\mathbf{A}$ (see later for definition)
- $\det(\mathbf{A}) \neq 0$

An important property of the inverse is that $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$. In combination with the transpose or the adjoint, we also have that $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$, and similarly $(\mathbf{A}^*)^{-1} = (\mathbf{A}^{-1})^*$. For this reason, you may sometimes also find the notations $\mathbf{A}^{-T}$ or $\mathbf{A}^{-*}$ to denote the successive application of the inverse and transpose/adjoint – the order does not matter.

3. Orthogonal vectors and matrices

See Chapter 2 of the textbook

3.1. Orthogonality of two vectors

The inner product defined in the previous section, when applied to two real vectors, is in fact the well known dot product. It therefore has the following properties:

- **The Eucledian length** of a vector is defined as $||\mathbf{x}|| = \sqrt{\mathbf{x}^T\mathbf{x}}$ (or by extension, $||\mathbf{x}|| = \sqrt{\mathbf{x}^*\mathbf{x}}$ for complex vectors)
- The cosine of the angle between two vectors $\mathbf{x}$ and $\mathbf{y}$ is given by

$$\cos\alpha = \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}^T||||\mathbf{y}||} \text{ for real vectors or } \cos\alpha = \frac{\mathbf{x}^*\mathbf{y}}{||\mathbf{x}^*||||\mathbf{y}||} \text{ otherwise} \quad (1.15)$$

- Two non-zero vectors $\mathbf{x}$ and $\mathbf{y}$ are therefore **orthogonal** provided their inner product $\mathbf{x}^*\mathbf{y} = 0$. If the vectors are real, this means that they lie at right-angles to one another in $\mathbb{R}^m$.

From this definition, we can now define an **orthogonal set** of non-zero vectors, as a set in which every possible pair of vectors is orthogonal (i.e. all members of the set are pair-wise orthogonal). An **orthonormal set** is a orthogonal set where all vectors have unit length, so $||\mathbf{x}|| = 1$ for all $\mathbf{x}$.

Orthogonal sets have a very important and very useful property:

Theorem: *The vectors in an orthogonal set are linearly independent.*

Linear independence means that it is not possible to write any vector in the set as a linear combination of other vectors in the set.

As a corollary, it is sufficient in $\mathbb{C}^m$ to find m orthogonal vectors to have a basis for the whole space $\mathbb{C}^m$. Using the orthogonality property, it is then very easy to express any vector in $\mathbb{C}^m$ as a linear combination of this new orthogonal basis. Indeed, suppose we have the orthonormal basis $\{\mathbf{q}_k\}_{k=1,\ldots,m}$, and we want to write the vector $\mathbf{b}$ in that basis. We know that it is possible, i.e. that there is a series of coefficients β_k such that

$$\mathbf{b} = \sum_{k=1}^{m} \beta_k \mathbf{q}_k \tag{1.16}$$

To find the β_k coefficient, we simply take the inner product of this expression with $\mathbf{q}_k$ (using the fact that the basis vectors have been normalized), to get

$$\beta_k = \mathbf{q}_k^* \mathbf{b} \tag{1.17}$$

Hence,

$$\mathbf{b} = \sum_{k=1}^{m} (\mathbf{q}_k^* \mathbf{b}) \mathbf{q}_k = \sum_{k=1}^{m} (\mathbf{q}_k \mathbf{q}_k^*) \mathbf{b} \tag{1.18}$$

where the second expression was obtained by commuting the scalar and the vector (which can always be done) and re-pairing the terms differently. While the first expression merely expresses $\mathbf{b}$ in the basis $\{\mathbf{q}_k\}_{k=1,\ldots,m}$, the second expression writes $\mathbf{b}$ as a sum of vectors that are each orthogonal projections of $\mathbf{b}$ on each of the $\mathbf{q}_k$, and identifies the projection operator onto $\mathbf{q}_k$ as the matrix $\mathbf{q}_k \mathbf{q}_k^*$. This property will be explored at length later in the course.

3.2. Unitary matrices

An orthogonal matrix is a real square matrix $\mathbf{Q}$ of size $m \times m$, satisfying the property $\mathbf{Q}^T = \mathbf{Q}^{-1}$, or equivalently $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$. Extending this to complex matrices, a **unitary matrix** is a complex square matrix $\mathbf{Q}$ of size $m \times m$, satisfying the property $\mathbf{Q}^* = \mathbf{Q}^{-1}$ or equivalently $\mathbf{Q}^* \mathbf{Q} = \mathbf{I}$.

Unitary (or orthogonal) matrices have the interesting property that their columns form an orthonormal basis. To see this, simply note that the ij coefficient of the product of two matrices is the inner product of the i-th row vector of the first matrix, with the $j-$th column vector of the second one. In order words, in the case of the product $\mathbf{Q}^*\mathbf{Q}$, and calling $\mathbf{q}_j$ the $j-$th column of $\mathbf{Q}$,

$$(\mathbf{Q}^*\mathbf{Q})_{ij} = \mathbf{q}_i^* \mathbf{q}_j = \mathbf{I}_{ij} = \delta_{ij} \tag{1.19}$$

where δ_{ij} is the Kronecker symbol. This proves that the pairwise product of any two column-vectors of $\mathbf{Q}$ is 0 if the vectors are different and 1 is the vectors are identical – the set $\{\mathbf{q}\}_{k=1,\ldots,m}$ is an orthonormal set.

Another interesting property of unitary matrices is that their effect preserves lengths and angles (modulo the sign) between vectors, and as such they can be viewed either as **rotations** or **reflections**. To see why, note how the inner product between two vectors is preserved by the action of $\mathbf{Q}$ on both vectors:

$$(\mathbf{Qx})^*(\mathbf{Qy}) = \mathbf{x}^*\mathbf{Q}^*\mathbf{Qy} = \mathbf{x}^*\mathbf{y} \tag{1.20}$$

This in turn means that the Euclidean norm of a vector is preserved by the action of $\mathbf{Q}$, and this together with the interpretation of the inner product as a dot product, also shows that the cosines of the angles are preserved (with possible change of sign of the actual angle)

4. Eigenvalues and singular values

See Chapter 24 and Chapter 4 of the textbook

4.1. Eigenvalues and eigenvectors

4.1.1. Definition Let us now restrict our attention to square matrices $\mathbf{A}$ of size $m \times m$. If a vector $\mathbf{v}$ satisfies the property that

$$\mathbf{Av} = \lambda\mathbf{v} \tag{1.21}$$

for any scalar $\lambda \in \mathbb{C}$, then this vector is called an **eigenvector of A** and λ is the **eigenvalue** associated with that vector. Eigenvectors represent special directions along which the action of the matrix $\mathbf{A}$ reduces to a simple multiplication. Eigenvectors and eigenvalues have very general uses in many branches of applied mathematics, from the solution of PDEs, to the analysis of the stability of physical problems, etc. We shall see some of these examples later in the course.

Note that eigenvectors are not unique but they can be scaled infinitely different ways: if $\mathbf{Av} = \lambda\mathbf{v}$, then for any nonzero scalar γ, $\gamma\mathbf{v}$ is also an eigenvector corresponding to λ because $\mathbf{A}(\gamma\mathbf{v}) = \lambda(\gamma\mathbf{v})$. For this reason, we usually consider eigenvectors to be normalized. This tells us that the fundamental object of interest is not really any particular choice of eigenvector, but rather the *direction(s)* spanned by the eigenvector(s).

A few definitions: the space spanned by all eigenvectors associated with the same eigenvalue λ is called the **eigenspace** corresponding to λ. Within that eigenspace, multiplication by the matrix $\mathbf{A}$ reduces to a scalar multiplication. The set consisting of all the eigenvalues of $\mathbf{A}$ is called the **spectrum** of $\mathbf{A}$. Finally, the largest possible value of $|\lambda|$ over all the eigenvalues in the spectrum of $\mathbf{A}$ is called the **spectral radius** of $\mathbf{A}$, and is denoted by $\rho(\mathbf{A})$.

4.1.2. Characteristic Polynomials The equation $\mathbf{Av} = \lambda\mathbf{v}$ is equivalent to considering a homogeneous system of linear equations

$$\Big(\mathbf{A} - \lambda\mathbf{I}\Big)\mathbf{v} = \mathbf{0}. \tag{1.22}$$

Recall that this system has a nonzero solution $\mathbf{v}$ if and only if $\mathbf{A} - \lambda\mathbf{I}$ is singular, which is further equivalent to,

$$\det\Big(\mathbf{A} - \lambda\mathbf{I}\Big) = 0. \tag{1.23}$$

The relation in Eq. (1.23) is a polynomial of degree m in λ, and is called the **characteristic polynomial** $p_A(\lambda)$ of $\mathbf{A}$, which can be written as,

$$\begin{aligned} p_A(\lambda) &= \lambda^m + \cdots + c_1\lambda + c_0 && (1.24)\\ &= (\lambda-\lambda_1)(\lambda-\lambda_2)\cdots(\lambda-\lambda_m) && (1.25)\end{aligned}$$

with $c_k \in \mathbb{C}$ (or $c_k \in \mathbb{R}$ for real matrices). The roots of $p_A(\lambda)$ are the eigenvalues of $\mathbf{A}$. Note that this implies that even if a matrix is real, its eigenvalues may be complex. Finally, to find the eigenvectors, we then solve for each λ_k,

$$\mathbf{A}\mathbf{v}_k = \lambda_k\mathbf{v}_k. \tag{1.26}$$

Example: Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{1.27}$$

The characteristic polynomial of this matrix is

$$(\lambda-2)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda-3)(\lambda-1) = 0, \tag{1.28}$$

therefore has two distinct roots $\lambda_1 = 1$ and $\lambda_2 = 3$, which are the eigenvalues of $\mathbf{A}$. To find the eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$, we solve

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda_k \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{1.29}$$

which tells us that $2x_1 + x_2 = x_1$ for $\mathbf{v}_1$ and $2x_1 + x_2 = 3x_1$ for $\mathbf{v}_2$. This then shows that

$$\mathbf{v}_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \text{ and } \mathbf{v}_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \tag{1.30}$$

where both eigenvectors have been normalized. □

Note: This illustrates that when seeking eigenvalues and eigenvectors *analytically*, one usually first solves for the characteristic polynomial to find the eigenvalues λ_k, and then find the eigenvectors. As we shall see in this course the numerical approach to finding eigenvectors and eigvenvalues is very different. This is because the characteristic polynomial turns out not to be too useful as a means of actually computing eigenvalues for matrices of nontrivial size. Several issues that can arise in numerical computing include:

- computing the coefficients of $p_A(\lambda)$ for a given matrix $\mathbf{A}$ is, in general, a substantial task,

- the coefficients of $p_A(\lambda)$ can be highly sensitive to perturbations in the matrix $\mathbf{A}$, and hence their computation is unstable,

- rounding error incurred in forming $p_A(\lambda)$ can destroy the accuracy of the roots subsequently computed,

- computing the roots of any polynomial of high degree numerically is another substantial task.

4.1.3. Multiplicity, Defectiveness, and Diagonalizability Since the characteristic polynomial can be written as $p_A(\lambda) = \Pi_{i=1}^{m}(\lambda - \lambda_i)$, we can define the **algebraic multiplicity** of a particular eigenvalue λ_k simply as the multiplicity of the corresponding root of $p_A(\lambda)$, i.e., how many times the factor $(\lambda - \lambda_k)$ appears in $p_A(\lambda)$.

By contrast, the **geometric multiplicity** of λ_k is the dimension of the eigenspace associated with λ_k. In other words, it is the maximal number of linearly independent eigenvectors corresponding to λ_k.

In general, the algebraic multiplicity is always greater or equal to the geometric multiplicity. If the algebraic multiplicity is strictly greater than the geometric multiplicity then the eigenvalue λ_k is said to be *defective*. By extension, an $m \times m$ matrix that has fewer than m linearly independent eigenvectors is said to be **defective**.

Example: The matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ -1 & 3 \end{bmatrix} \tag{1.31}$$

has $p_A(\lambda) = (\lambda-1)(\lambda-3)+1 = (\lambda-2)^2 = 0$, hence has an eigenvalue $\lambda = 2$ as a double root. Therefore the **algebraic multiplicity** of $\lambda = 2$ is 2. Let's assume that there are two linearly independent eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$. If $\mathbf{v}_1 = (x_1, x_2)^T$ then x_1 and x_2 must satisfy $x_1 + x_2 = 2x_1$, or in other words $x_1 = x_2$ so $\mathbf{v}_1 = (1/\sqrt{2}, 1/\sqrt{2})^T$. Similarly we seek $\mathbf{v}_2 = (x_1, x_2)^T$, and its components must satisfy $-x_1 + 3x_2 = 2x_2$, or equivalently $x_1 = x_2$ again. This shows that $\mathbf{v}_1$ and $\mathbf{v}_2$ are actually the same vector, which implies that the **geometric multiplicity** of $\lambda = 2$ is only equal to one. This eigenvalue is therefore **defective**, and so is the matrix $\mathbf{A}$.

Example: $\lambda = 1$ is the only eigenvalue with algebraic multiplicity two for both matrices:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \text{ and } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{1.32}$$

Its geometric multiplicity, however, is one for the first and two for the latter. □

If $m \times m$ matrix $\mathbf{A}$ is *nondefective*, then it has a full set of linearly independent eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_m$ corresponding to the eigenvalues $\lambda_1, \cdots, \lambda_m$. If we let $\mathbf{D}$ be the diagonal matrix formed with all the eigenvalues λ_1 to λ_m, and $\mathbf{V}$

is the matrix formed by the column vectors $\mathbf{v}_1$ to $\mathbf{v}_m$ (in the same order), then $\mathbf{V}$ is nonsingular since the vectors are linearly independent and we have

$$\mathbf{AV} = \mathbf{VD} \tag{1.33}$$

To see why, note that

$$\mathbf{AV} = \begin{pmatrix} \mathbf{Av}_1 & \mathbf{Av}_2 & \dots & \mathbf{Av}_m \end{pmatrix} = \begin{pmatrix} \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \dots & \lambda_m\mathbf{v}_m \end{pmatrix} \tag{1.34}$$

while

$$\mathbf{VD} = \begin{pmatrix} \mathbf{V}\lambda_1\mathbf{e}_1 & \mathbf{V}\lambda_2\mathbf{e}_2 & \dots & \mathbf{V}\lambda_m\mathbf{e}_m \end{pmatrix} = \begin{pmatrix} \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \dots & \lambda_m\mathbf{v}_m \end{pmatrix} \tag{1.35}$$

Multiplying on both sides by $\mathbf{V}^{-1}$ yields

$$\mathbf{V}^{-1}\mathbf{AV} = \mathbf{D}. \tag{1.36}$$

This shows that $\mathbf{A}$ is **diagonalizable**, i.e., can be put into a diagonal form using a **similarity transformation**.

4.1.4. Similarity transformation, change of base Let $\mathbf{A}$ and $\mathbf{B}$ be two $m \times m$ square matrices. Then $\mathbf{A}$ is **similar** to $\mathbf{B}$ if there is a nonsingular matrix $\mathbf{P}$ for which

$$\mathbf{B} = \mathbf{P}^{-1}\mathbf{AP}. \tag{1.37}$$

The operation $\mathbf{P}^{-1}\mathbf{AP}$ is called a **similarity transformation** of $\mathbf{A}$. Note that this is a symmetric relation (i.e., $\mathbf{B}$ is *similar* to $\mathbf{A}$), since

$$\mathbf{A} = \mathbf{Q}^{-1}\mathbf{BQ}, \text{ with } \mathbf{Q} = \mathbf{P}^{-1}. \tag{1.38}$$

A similarity transformation is simply a change of base for matrices, i.e. $\mathbf{B}$ can be interpreted as being the matrix $\mathbf{A}$ expressed in the basis formed by the column vectors of $\mathbf{P}$. As such, many of the geometric properties of $\mathbf{A}$ are preserved. To be specific, if $\mathbf{A}$ and $\mathbf{B}$ are similar, then the followings statements are true.

1. $p_A(\lambda) = p_B(\lambda)$.

   Proof: Then

$$\begin{aligned} p_B(\lambda) &= \det(\mathbf{B} - \lambda\mathbf{I}) \\ &= \det(\mathbf{P}^{-1}(\mathbf{A} - \lambda\mathbf{I})\mathbf{P}) \\ &= \det(\mathbf{P}^{-1})\det(\mathbf{A} - \lambda\mathbf{I})\det(\mathbf{P}) \\ &= \det(\mathbf{A} - \lambda\mathbf{I}) \\ &= p_A(\lambda). \end{aligned} \tag{1.39}$$

   since $\det(\mathbf{P}^{-1}) = 1/\det(\mathbf{P})$ for any nonsingular matrix. $\square$

2. The eigenvalues of $\mathbf{A}$ and $\mathbf{B}$ are exactly the same, $\lambda(\mathbf{A}) = \lambda(\mathbf{B})$, and there is a one-to-one correspondence of the eigenvectors.

   Proof: Let $\mathbf{Av} = \lambda\mathbf{v}$. Then
   $$\mathbf{P}^{-1}\mathbf{AP}(\mathbf{P}^{-1}\mathbf{v}) = \lambda\mathbf{P}^{-1}\mathbf{v}, \tag{1.40}$$
   or equivalently,
   $$\mathbf{Bu} = \lambda\mathbf{u}, \text{ with } \mathbf{u} = \mathbf{P}^{-1}\mathbf{v}. \tag{1.41}$$
   Also the one-to-one correspondence between $\mathbf{v}$ and $\mathbf{u}$ is trivial with the relationship $\mathbf{u} = \mathbf{P}^{-1}\mathbf{v}$, or $\mathbf{v} = \mathbf{Pu}$. $\square$

3. The trace and determinant are unchanged. This can easily be shown from the fact that the characteristic polynomial remains the same:
   $$\begin{aligned} \text{trace}(\mathbf{A}) &= \text{trace}(\mathbf{B}), & (1.42)\\ \det(\mathbf{A}) &= \det(\mathbf{B}). & (1.43) \end{aligned}$$

4.2. Singular values and singular vectors

By contrast with eigenvectors and eigenvalues, singular vectors and singular values have a very simple geometric interpretation and do not require the matrix to be square in order to be computed. We therefore consider here any matrix $\mathbf{A}$ of size $m \times n$. The notion of singular values and singular vectors can be understood easily if one notes first that the image of the unit ball in $\mathbb{R}^n$ (or more generally $\mathbb{C}^n$) is a hyperellipse in $\mathbb{R}^m$ (or more generally $\mathbb{C}^m$).

Example: Consider the matrix $\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$ then if $\mathbf{y} = \mathbf{Ax}$, and we only consider vectors $\mathbf{x} = (x_1, x_2)^T$ such that $x_1^2 + x_2^2 = 1$, their image $\mathbf{y} = (y_1, y_2)^T$ satisfies $y_1 = 2x_1 + x_2$ and $y_2 = x_2$ so that $x_2 = y_2$, $x_1 = (y_1 - y_2)/2$ and therefore
$$\frac{(y_1 - y_2)^2}{4} + y_2^2 = 1 \tag{1.44}$$
which is indeed the equation of a slanted ellipse (see Figure 1). $\square$

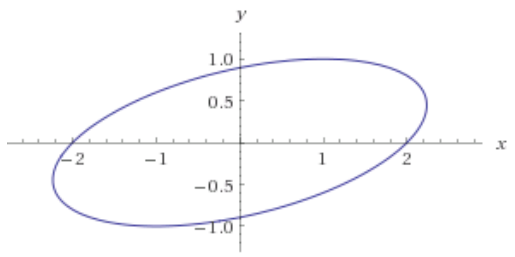

Figure 1. Image of the unit circle after application of the matrix $\mathbf{A}$.

Based on this geometrical interpretation of $\mathbf{A}$, we define **the singular values** of $\mathbf{A}$ as the lengths of the principal axes of the hyperellipse that is the image of the unit ball. They are denoted as $\{\sigma_i\}_{i=1,\dots,k}$, and are usually ordered, such that $\sigma_1 \geq \sigma_2 \geq \sigma_3 \cdots \geq \sigma_k > 0$. Note that k could be smaller than m. Then we define the **left singular vectors** of $\mathbf{A}$, $\{\mathbf{u}_i\}_{i=1,\dots,k}$, which are the directions of the principal axes of the hyperellipse corresponding to σ_i. Finally, we define the **right singular vectors** of $\mathbf{A}$, $\{\mathbf{w}_i\}_{i=1,\dots,k}$, such that $\mathbf{A}\mathbf{w}_i = \sigma_i \mathbf{u}_i$ for $i = 1, \dots, k$.

Further properties of the singular values and singular vectors, together with how to compute them, will be discussed later in the course.

5. Norms

See Chapter 3 of the textbook

In much of what we will learn in this course and in AMS 213B, we will need to have tools to measure things such as the quality of a numerical solution in comparison with the true solution (when it is known), or the rate of convergence of an iterative numerical algorithm to a solution. To do so usually involves measuring the size (loosely speaking) of a vector or a matrix, and in this respect we have several options that all fall under the general description of norms.

5.1. Vector norms

5.1.1. Definitions of vector norms A norm is a function that assigns a real valued number to each vector in $\mathbb{C}^m$. There are many possible definitions of vector norms, but they must all satisfy the following conditions:

- A norm must be positive for all non-zero vectors: $||\mathbf{x}|| > 0$ unless $\mathbf{x} = \mathbf{0}$, and $||\mathbf{0}|| = 0$.
- $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$ for any two vectors $\mathbf{x}$ and $\mathbf{y}$. This is called the **triangle inequality**
- $||\alpha\mathbf{x}|| = |\alpha|||\mathbf{x}||$ for any vectors $\mathbf{x}$ and any real α.

A large class of norms consists of the ***p*-norms**. The p-norm (or l^p-norm) of an n-vector $\mathbf{x}$ is defined by

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{n} \left| x_i \right|^p \right)^{\frac{1}{p}}. \qquad (1.45)$$

Important special cases are:

- 1-norm:

$$||\mathbf{x}||_1 = \sum_{i=1}^{n} \left| x_i \right| \qquad (1.46)$$

- 2-norm: (also called the Euclidean length, defined in the previous section)

$$||\mathbf{x}||_2 = \left(\sum_{i=1}^{n} \left|x_i\right|^2\right)^{\frac{1}{2}} = \sqrt{\mathbf{x}^T\mathbf{x}} \tag{1.47}$$

- ∞-norm (or max-norm):

$$||\mathbf{x}||_\infty = \max_{1\le i\le n} \left|x_i\right| \tag{1.48}$$

As we shall see, different norms come in handy in different kinds of applications although in practice the most useful ones are the $1-$norm, $2-$norm and ∞-norms.

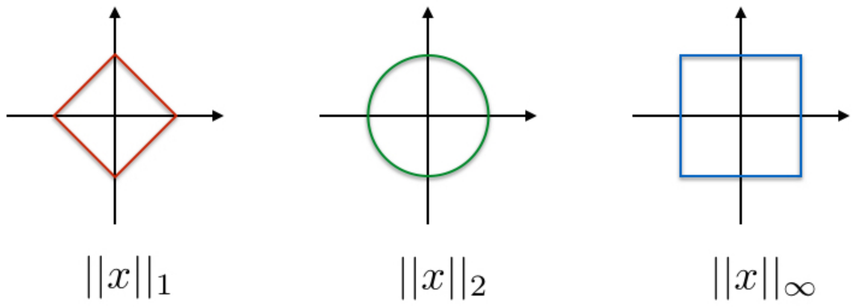

Figure 2. Illustrations of a unit sphere in $\mathbb{R}^2$, $||x|| = 1$, in three different norms: 1-norm, 2-norm and ∞-norm.

Example: For the vector $\mathbf{x} = (-1.6, 1.2)^T$, we get

$$||\mathbf{x}||_1 = 2.8, ||\mathbf{x}||_2 = 2.0, ||\mathbf{x}||_\infty = 1.6. \tag{1.49}$$

$\square$

5.1.2. The Cauchy-Schwarz and Hölder inequalities When proving various theorems on stability or convergence of numerical algorithms, we shall sometimes use the Hölder inequality.

Theorem: *For any two vectors* $\mathbf{x}$ *and* $\mathbf{y}$*,*

$$|\mathbf{x}^*\mathbf{y}| \le ||\mathbf{x}||_p||\mathbf{y}||_q, \tag{1.50}$$

where p and q satisfy $\frac{1}{p} + \frac{1}{q} = 1$ with $1 \le p, q \le \infty$.

The bound is tight in the sense that, if $\mathbf{x}$ and $\mathbf{y}$ are parallel vectors (such that $\mathbf{x} = \alpha\mathbf{y}$ for some α), the inequality becomes an equality. When applied to the $2-$norm, the inequality is called the Cauchy-Schwarz inequality.

5.2. Matrix norms

5.2.1. Definitions of matrix norms Similar to vector norms, matrix norms are functions that take a matrix and return a positive scalar, with the following properties:

- $||\mathbf{A}|| > 0$ unless $\mathbf{A} = \mathbf{0}$ in which case $||\mathbf{0}|| = 0$.
- $||\mathbf{A} + \mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}||$
- $||\alpha \mathbf{A}|| = |\alpha| ||\mathbf{A}||$.

We could, in principle, define matrix norms exactly as we defined vector norms, by summing over all components a_{ij} of the matrix. In fact, a commonly used norm is the **Hilbert-Schmidt norm**, also called the **Frobenius norm**:

$$||\mathbf{A}||_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \tag{1.51}$$

which is analogous to the $2-$norm for vectors.

An interesting property of this norm is that it can be written compactly as $||\mathbf{A}||_F = \sqrt{\text{Tr}(\mathbf{A}\mathbf{A}^*)} = \sqrt{\text{Tr}(\mathbf{A}^*\mathbf{A})}$. However, aside from this norm, other more useful definitions of norms can be obtained by defining a norm that measures the *effect* of a matrix on a vector.

Definition: The matrix p-norm of $m \times n$ matrix $\mathbf{A}$ can be defined by

$$||\mathbf{A}||_p = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{||\mathbf{A}\mathbf{x}||_p}{||\mathbf{x}||_p}. \tag{1.52}$$

In practice, what this does is to estimate the maximum amount by which the $p-$norm of a vector $\mathbf{x}$ can be *stretched* by the application of the matrix. And since any vector can be written as $\mathbf{x} = \alpha \hat{\mathbf{x}}$ where $\hat{\mathbf{x}}$ is a unit vector in the direction of $\mathbf{x}$, it suffices to look at the action of $\mathbf{A}$ on all possible unit vectors:

$$||\mathbf{A}||_p = \sup_{\hat{\mathbf{x}}} ||\mathbf{A}\hat{\mathbf{x}}||_p \tag{1.53}$$

Some matrix norms are easier to compute than others. For instance, by this definition, and by the definition of singular values, we simply have

$$||\mathbf{A}||_2 = \sigma_1 \tag{1.54}$$

that is, the length of the largest principal axis of the image of the unit ball.

Let's also look for instance at the case of the $1-$norm, for an $m \times n$ matrix $\mathbf{A}$. We begin by computing

$$||\mathbf{A}\hat{\mathbf{x}}||_1 = ||\sum_{i=1}^{n} \mathbf{a}_i \hat{x}_i||_1 \leq \sum_{i=1}^{n} |\hat{x}_i| ||\mathbf{a}_i||_1 \tag{1.55}$$

using the column vectors $\mathbf{a}_i$ of $\mathbf{A}$, and the triangle inequality. Then, using the fact that $||\hat{\mathbf{x}}||_1 = \sum_{i=1}^n |\hat{x}_i| = 1$, and noting that $||\mathbf{a}_i||_1 \leq \max_i ||\mathbf{a}_i||_1$ for any i, we have

$$||\mathbf{A}\hat{\mathbf{x}}||_1 \leq \sum_{i=1}^n |\hat{x}_i| ||\mathbf{a}_i||_1 \leq \max_i ||\mathbf{a}_i||_1 \sum_{i=1}^n |\hat{x}_i| = \max_i ||\mathbf{a}_i||_1 \tag{1.56}$$

so $||\mathbf{A}\hat{\mathbf{x}}||_1 \leq \max_i ||\mathbf{a}_i||_1$. As it turns out, the inequality becomes an equality when $\hat{\mathbf{x}} = \mathbf{e}_I$, where I is the index i for which $||\mathbf{a}_i||_1$ is largest, so the maximum possible value of $||\mathbf{A}\hat{\mathbf{x}}||_1$ over all possible unit vectors $\hat{\mathbf{x}}$ is indeed $\max_i ||\mathbf{a}_i||_1$. In short

$$||\mathbf{A}||_1 = \max_j \sum_{i=1}^m \left|a_{ij}\right| \tag{1.57}$$

It can similarly be shown that

$$||\mathbf{A}||_\infty = \max_i \sum_{j=1}^n \left|a_{ij}\right| \tag{1.58}$$

Finally, the $p-$ norm of *diagonal matrices* is easy to compute: $||\mathbf{D}||_p = \max_i |d_i|$, where d_i is the i-th diagonal component of the matrix $\mathbf{D}$.

5.2.2. Properties of matrix norms Similar to the Hölder inequality, it is possible to bound the norm of a product of two matrices. For any one of the $p-$norms associated with a vector norm, or for the Hilbert-Schmidt/Frobenius norms, we have

$$||\mathbf{AB}|| \leq ||\mathbf{A}|| ||\mathbf{B}|| \tag{1.59}$$

In general, the inequality is strict. However, in the special case where one of the matrices is unitary (orthogonal), and where we use the $2-$norm or the Hilbert-Schmidt/Frobenius norms, we have

$$||\mathbf{QA}||_2 = ||\mathbf{A}||_2 \text{ and } ||\mathbf{QA}||_F = ||\mathbf{A}||_F \tag{1.60}$$

6. Properties of Machine Arithmetics

See Chapter 13 of textbook + additional material on computer representation of numbers from various sources

So far, we have merely summarized the most salient results of Linear Algebra that will come in handy when trying to design numerical algorithms to solve linear algebra problems. Up to now, every result discussed was exact. At this point in time, however, we must now learn about one of the most important issues related to the numerical implementation of numerical algorithms, namely, the fact that numbers are not represented *exactly*, but rather, *approximately*, in the computer.

Most computers have different modes for representing integers and real numbers, *integer mode* and *floating-point mode*, respectively. Let us now take a look at these modes.

6.1. Integer mode

The representation of an integer number is (usually) *exact.* Recall that we can represent an integer as a sequence of numbers from 0 to 9, for instance, as an expansion in base 10:

$$a_n a_{n-1} \cdots a_0 = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_0 \times 10^0. \tag{1.61}$$

Example: Base 10

$$159 = 1 \times 10^2 + 5 \times 10^1 + 9 \times 10^0. \tag{1.62}$$

□

However, the number base used in computers is seldom decimal (e.g., base 10), but instead binary (e.g., base 2) where one "bit" is either 0 or 1. In binary form, any positive integers can be written as

$$a_n a_{n-1} \cdots a_0 = a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_0 \times 2^0. \tag{1.63}$$

Example: Base 2

$$\begin{aligned} 159 &= 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 10011111_2. \end{aligned} \tag{1.64}$$

□

A **signed integer** is typically stored on a finite number of bytes (recall, 1 byte = 8 bits), usually using 1-bit for the sign (though other conventions also exist).

In Fortran there are two common ways to represent integers, *normal* and *long* types. The normal integers are stored on 4 bytes, or equivalently 32 bits where one bit is reserved for the sign and the rest 31 bits for the value itself. On the other hand, the long integers are stored on 8 bytes which are equivalent to 64 bits, where one bit for the sign and the rest 63 bits for the value.

As a consequence for a given integer type there are only finite numbers of integers which can be used in programing:

- for normal 4-byte integers: between -2^{31} and 2^{31},
- for normal 8-byte integers: between -2^{63} and 2^{63}.

This means that any attempts to reach numbers beyond these values will cause problems. Note that we have $2^{31} \approx 2.1$ billion which is not so big a number.

6.2. Floating-point mode

The base 10 notation (decimal) for real numbers can be written as

$$\begin{aligned}&a_n a_{n-1} \cdots a_0 . b_1 b_2 \cdots b_m \\ &= a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_0 \times 10^0 \\ &+ b_1 \times 10^{-1} + b_2 \times 10^{-2} + \cdots + b_m \times 10^{-m},\end{aligned} \tag{1.65}$$

and by analogy we write a real number in base 2 (binary) as

$$\begin{aligned}&a_n a_{n-1} \cdots a_0 . b_1 b_2 \cdots b_m \\ &= a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_0 \times 2^0 \\ &+ b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots + b_m \times 2^{-m},\end{aligned} \tag{1.66}$$

Definition: We note that we can only store finite numbers of a_i and b_j as every computer has finite storage limit. This implies that there are cases when real numbers can only be *approximated* with finitely many combinations of a_i and b_j. The error associated with this approximation is called *roundoff errors.*

6.2.1. Standard notations In fact, the numbers are not stored as written above. Rather, they are stored as

$$2^n \Big(a_n + a_{n-1} 2^{-1} + \cdots + a_0 2^{-n} + b_1 2^{-n-1} + \cdots + b_m 2^{-n-m} \Big), \tag{1.67}$$

or

$$2^{n+1} \Big(a_n 2^{-1} + a_{n-1} 2^{-2} + \cdots + a_0 2^{-n-1} + b_1 2^{-n-2} + \cdots + b_m 2^{-n-m-1} \Big). \tag{1.68}$$

In the first case a_n can be chosen to be nonzero by assumption, which necessarily gives $a_n = 1$. The first is referred to as *IEEE standard* and the second as *DEC standard.*

Example: The representation of 27.25 in base 2 becomes

$$\begin{aligned}&27.25 \\ &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &= 11011.01_2,\end{aligned} \tag{1.69}$$

which can be written in two ways as just shown above:

$$11011.01_2 = \begin{cases} 2^4 \Big(1.1011101_2 \Big) \text{ IEEE standard,} \\ \\ 2^5 \Big(0.11011101_2 \Big) \text{ DEC standard.} \end{cases} \tag{1.70}$$

□

Definition: In general, this takes of the form of

$$x = 2^k \times f, \tag{1.71}$$

where k and f are called the *exponent* and *mantissa*, respectively.

6.2.2. Standard Fortran storage types: double vs. single precisions There are two standard storage types available in Fortran. In addition to them, one can define any new type as needed in Fortran 90 and above. The two standard storage types are

- single precision : `type REAL(SP)`. Storage is on 4 bytes (i.e., 32 bits = 1 bit for sign + 8 bits for the exponent + 23 bits for the mantissa),
- double precision : `type REAL(DP)`. Storage on 8 bytes (i.e., 64 bits = 1 bit for sign + 11 bits for the exponent + 52 bits for the mantissa).

Note: The bits in the exponent store integers from L to U, where usually, L is the lowest exponent, a negative integer; and U is the highest exponent, a positive integer, with

$$U - L \approx \begin{cases} 2^8 \text{ for single precision,} \\ \\ 2^{11} \text{ for double precision,} \end{cases} \tag{1.72}$$

where

$$L = \begin{cases} -126 \text{ for single precision,} \\ \\ -1022 \text{ for double precision,} \end{cases} \tag{1.73}$$

and

$$U = \begin{cases} 127 \text{ for single precision,} \\ \\ 1023 \text{ for double precision.} \end{cases} \tag{1.74}$$

□

6.2.3. Floating-point arithmetic and roundoff errors Arithmetic using floating-point numbers is *quite* different from real arithmetic. In order to understand this let's work in base 10 for simplicity. If we represent π in, say, DEC standard, we get

- a representation up to 2 decimal places (significant digits) is 0.31, and
- a representation up to 6 decimal places (significant digits) is 0.314159.

For a given number of significant digits (i.e., a given length of mantissa), the "distance" between two consecutive numbers is dependent on the value of the exponent. Let us consider the following example.

Example: Suppose we have 3 possible values of the exponent, $k = -2, -1$ and 0, and 2 digits for mantissa. Positive numbers we can possibly create from this condition are

$$\left.\begin{array}{c} 0.10\times 10^{-2} \\ 0.11\times 10^{-2} \\ \vdots \\ 0.98\times 10^{-2} \\ 0.99\times 10^{-2} \end{array}\right\} \text{all separated by } 10^{-4} \tag{1.75}$$

$$\left.\begin{array}{c} 0.10\times 10^{-1} \\ 0.11\times 10^{-1} \\ \vdots \\ 0.98\times 10^{-1} \\ 0.99\times 10^{-1} \end{array}\right\} \text{all separated by } 10^{-3} \tag{1.76}$$

$$\left.\begin{array}{c} 0.10\times 10^{0} \\ 0.11\times 10^{0} \\ \vdots \\ 0.98\times 10^{0} \\ 0.99\times 10^{0} \end{array}\right\} \text{all separated by } 10^{-2} \tag{1.77}$$

Here we note that the continuous real line has been discretized which inevitably introduces roundoff errors. Also, the discretization does not produce equidistance uniform spacing, instead the non-uniform spacing depends on the absolute value of the numbers considered. □

Note: The floating-point arithmetic operation is not associative as a result of roundoff errors. To see this, let's consider a case with 6-digit mantissa. Let's take three real numbers,

$$a = 472635 = 0.472635\times 10^6, \tag{1.78}$$
$$b = 472630 = 0.472630\times 10^6, \tag{1.79}$$
$$c = 27.5013 = 0.275013\times 10^2. \tag{1.80}$$

We see that the floating-point operation fails to preserve associative rule,

$$(a-b)+c \neq a-(b-c). \tag{1.81}$$

In the first case, we have

$$\begin{aligned} (a-b)+c &= (472635-472630)+27.5013 \\ &= 5.00000+27.5013 \\ &= 32.5013, \end{aligned} \tag{1.82}$$

whereas the second case gives

$$\begin{aligned} a-(b-c) &= 472635-(472630-27.5013) \\ &= 472635-\underbrace{472602.4987}_{\text{more than 6 digits hence must be rounded off}} \\ &= 472635-472602 \\ &= 33.0000. \end{aligned} \tag{1.83}$$

As can be seen the error on the calculation is huge! It is of the order of the discretization error for the largest of the numbers considered (i.e., a and b in this case). □

6.2.4. Machine accuracy ϵ It is a similar concept – now with the question of what is the *largest* number ϵ that can be added to 1 such that in floating-point arithmetic, one gets

$$1+\epsilon=1? \tag{1.84}$$

Let's consider the following example:

Example: Consider 6-digit mantissa. Then we have

$$1=0.100000\times 10^1, \tag{1.85}$$

and then

$$1+10^{-7}=0.1000001\times 10^1. \tag{1.86}$$

However, the last representation exceeds 6-digit limit and hence needs to be rounded *down* to 0.100000×10^1, resulting

$$1+10^{-7}=0.100000\times 10^1. \tag{1.87}$$

This implies that the machine accuracy is $\epsilon \approx 10^{-7}$. □

Note: For floating-point arithmetic in base 2, with mantissa of size m, we have

$$\epsilon \approx 2^{-m} = \begin{cases} 2^{-23}\approx 10^{-7} \text{ in real single precision,} \\ 2^{-52}\approx 10^{-16} \text{ in real double precision.} \end{cases} \tag{1.88}$$

□

6.2.5. Overflow and underflow problems There exists a smallest and a largest number (in absolute value) that can be represented in floating-point notation. For instance, let us suppose that the exponent k ranges from -4 to 4, and the mantissa has 8 significant digits. This gives us that the smallest possible number in base 10 is

$$x_{\min}=0.10000000\times 10^{-4}=10^{-4-1}, \tag{1.89}$$

and the largest possible number is

$$x_{\max} = 0.9999999 \times 10^4 \approx 10^4. \tag{1.90}$$

Therefore in general, in base 2, we have

$$x_{\min} = 2^{L-1} = \begin{cases} 2^{-127} \approx 10^{-38} \text{ in real single precision,} \\ 2^{1023} \approx 10^{-308} \text{ in real double precision.} \end{cases} \tag{1.91}$$

$$x_{\max} = 2^{U} = \begin{cases} 2^{127} \approx 10^{38} \text{ in real single precision,} \\ 2^{1023} \approx 10^{308} \text{ in real double precision.} \end{cases} \tag{1.92}$$

If the outcome of a floating-point operation yields $|x| < x_{\min}$ then an underflow error occurs. In this case x will be set to be zero usually and computation will continue. In contrast, if $|x| > x_{\max}$ then an overflow error occurs, causing a fatal termination of program.

7. Conditioning and condition number

See Chapter 12 of the textbook

Having seen that numbers are not represented exactly in the numerical world, one may well begin to worry about the effects of this approximate representation on simple things such as matrix operations. In other words, what happens (for instance) to $\mathbf{Ax}$ if the entries of $\mathbf{x}$ are only known approximately, or if the entries of $\mathbf{A}$ are only known approximately? Does this have a significant effect or a small effect on $\mathbf{Ax}$?

As it turns out, this notion is quite general and does not necessarily only apply to matrices. It is called **conditioning** and is more generally applied to any function of $\mathbf{x}$.

Loose definition: A function $\mathbf{f}(\mathbf{x})$ (where $\mathbf{f}$ and $\mathbf{x}$ are either scalars or vectors) is *well-conditioned* near the point $\mathbf{x}_0$ provided small perturbation $\delta\mathbf{x}$ around $\mathbf{x}_0$ only lead to small perturbations $\delta\mathbf{f} = \mathbf{f}(\mathbf{x}_0 + \delta\mathbf{x}) - \mathbf{f}(\mathbf{x}_0)$. A problem is *ill-conditioned* if a small $\delta\mathbf{x}$ leads to a large $\delta\mathbf{f}$.

What small and large mean, in this definition, can depend on the application of interest. We would also like to create a number (the **condition number**) that can become a diagnostic of the conditioning properties of a problem. To do so, we now introduce the following more mathematical definitions.

7.1. Absolute condition number

For a fixed $\mathbf{x}_0$, we define the **absolute condition number** at $\mathbf{x} = \mathbf{x}_0$ as

$$\hat{\kappa}(\mathbf{x}_0) = \lim_{\epsilon \to 0} \sup_{||\delta \mathbf{x}|| \leq \epsilon} \frac{||\delta \mathbf{f}||}{||\delta \mathbf{x}||} \tag{1.93}$$

where $\delta \mathbf{f}$ was defined above. This is therefore the limit, when ϵ tends to 0, of the maximum possible value of $||\delta \mathbf{f}||/||\delta \mathbf{x}||$ over all possible $\delta \mathbf{x}$ whose norm is less than or equal to ϵ.

If the function $\mathbf{f}(\mathbf{x})$ is differentiable, then we can write $\delta \mathbf{f} = \mathbf{J}_0 \delta \mathbf{x}$ in the limit $\delta \mathbf{x} \to \mathbf{0}$, where $\mathbf{J}_0$ is the Jacobian of $\mathbf{f}$ (i.e. the matrix of partial derivatives) at $\mathbf{x} = \mathbf{x}_0$. In that case

$$\hat{\kappa}(\mathbf{x}_0) = \lim_{\epsilon \to 0} \sup_{||\delta \mathbf{x}|| \leq \epsilon} \frac{||\delta \mathbf{f}||}{||\delta \mathbf{x}||} = \lim_{\epsilon \to 0} \sup_{||\delta \mathbf{x}|| \leq \epsilon} \frac{||\mathbf{J}_0 \delta \mathbf{x}||}{||\delta \mathbf{x}||} = ||\mathbf{J}(\mathbf{x}_0)|| \tag{1.94}$$

for any $p-$norm. In short,

$$\hat{\kappa}(\mathbf{x}_0) = ||\mathbf{J}(\mathbf{x}_0)|| \tag{1.95}$$

7.2. Relative condition number

In many cases, especially when working with floating point arithmetic, it makes more sense to establish the conditioning of relative changes $||\delta \mathbf{f}||/||\mathbf{f}||$ rather than of $||\delta \mathbf{f}||$ itself. To do so, we consider instead the **relative condition number** at $\mathbf{x} = \mathbf{x}_0$, defined as

$$\kappa(\mathbf{x}_0) = \lim_{\epsilon \to 0} \sup_{||\delta \mathbf{x}|| \leq \epsilon} \frac{||\delta \mathbf{f}||/||\mathbf{f}||}{||\delta \mathbf{x}||/||\mathbf{x}||} \tag{1.96}$$

Using the same trick as above for differentiable functions $\mathbf{f}(\mathbf{x})$, we then have

$$\kappa(\mathbf{x}_0) = \frac{||\mathbf{J}_0|| ||\mathbf{x}_0||}{||\mathbf{f}(\mathbf{x}_0)||} \tag{1.97}$$

The relative condition number is more commonly used in numerical linear algebra because the rounding errors (which can be the cause of $\delta \mathbf{x}$ and therefore $\delta \mathbf{f}$) are relative to a given $\mathbf{x}_0$, see the previous sections.

Example 1: Consider the function $f(x) = \sqrt{x}$, whose derivative (Jacobian) at any point $x > 0$ is $0.5x^{-1/2}$. The relative condition number

$$\kappa = \left| \frac{f'(x)x}{f(x)} \right| = \left| \frac{x}{2\sqrt{x}\sqrt{x}} \right| = \frac{1}{2} \tag{1.98}$$

is finite and small, suggesting a well-conditioned problem. Note, however, that the absolute condition number

$$\hat{\kappa} = \left| f'(x) \right| = \frac{1}{2\sqrt{x}} \to \infty \text{ as } x \to 0 \tag{1.99}$$

suggesting ill-conditioning as $x \to 0$. Should we worry about it? The answer is no – this is a specific case where we care more about the relative changes than the absolute changes in δx, since it doesn't make sense to take the limit $x \to 0$ unless $|\delta x|$ also goes to 0. □

Example 2: Consider the function $f(x_1, x_2) = x_2 - x_1$. The Jacobian of this function is the row-vector $\mathbf{J} = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{pmatrix} = (-1 \;\; 1)$. The ∞-norm is the sum of the absolute values of the components of that row (see previous section), namely $||\mathbf{J}||_\infty = 2$, so

$$\kappa(x_1, x_2) = \frac{||\mathbf{J}||_\infty ||\mathbf{x}||_\infty}{|f(x_1, x_2)|} = \frac{2 \max(|x_1|, |x_2|)}{|x_2 - x_1|} \tag{1.100}$$

In this case, we see that there can be serious ill-conditioning in terms of relative errors when $x_1 \to x_2$. This recovers our earlier findings for floating point arithmetic that the truncation errors are always of the order of the largest terms (here x_1 and x_2), so the relative error on $x_2 - x_1$ can be huge compared with $x_2 - x_1$. □

7.3. Condition number of Matrix-Vector multiplications

Suppose we consider the function that takes a vector $\mathbf{x}$ and multiplies the matrix $\mathbf{A}$ to it: $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$. In this case, by definition the Jacobian matrix $\mathbf{J}$ *is* the matrix $\mathbf{A}$, so we can immediately construct the relative condition number as

$$\kappa(\mathbf{x}_0) = \frac{||\mathbf{A}||||\mathbf{x}_0||}{||\mathbf{A}\mathbf{x}_0||} \tag{1.101}$$

Generally speaking, we would have to stop here, and evaluate κ. However, suppose we know that $\mathbf{A}$ is an invertible square matrix. Then if we write $\mathbf{x}_0 = \mathbf{A}^{-1}\mathbf{A}\mathbf{x}_0$, we have, by this equality, that $||\mathbf{x}_0|| = ||\mathbf{A}^{-1}\mathbf{A}\mathbf{x}_0|| < ||\mathbf{A}^{-1}||||\mathbf{A}\mathbf{x}_0||$ using the matrix multiplication bound discussed in the previous section. So

$$\kappa(\mathbf{x}_0) = \frac{||\mathbf{A}||||\mathbf{x}_0||}{||\mathbf{A}\mathbf{x}_0||} \leq ||\mathbf{A}||||\mathbf{A}^{-1}|| \tag{1.102}$$

regardless of $\mathbf{x}_0$, for any non-singular matrix $\mathbf{A}$.

7.4. Condition number of the solution of a set of linear equations

Suppose we now want to solve the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{b}$ is known exactly, but some uncertainty exists in the entries of $\mathbf{A}$. What are the impacts of this uncertainty on the numerical evaluation of the solution $\mathbf{x}$? Since solving this problem requires computing the function $\mathbf{x} = \mathbf{f}(\mathbf{b}) = \mathbf{A}^{-1}\mathbf{b}$, which is a matrix multiplication by $\mathbf{A}^{-1}$, we can use all of the results of the previous section to show that the condition number of this problem is bounded by

$$\kappa(\mathbf{b}) \leq ||\mathbf{A}^{-1}||||\mathbf{A}|| \tag{1.103}$$

as before.

7.5. Condition number of a matrix

The quantity $||\mathbf{A}||||\mathbf{A}^{-1}||$ is so important and comes up so often that it is often referred to simply as **the condition number** of the matrix $\mathbf{A}$, $\text{cond}(\mathbf{A})$. If we use the 2−norm to compute it, then $||\mathbf{A}||_2 = \sigma_1$, and it can also be shown that $||\mathbf{A}^{-1}|| = 1/\sigma_m$. In that case,

$$\text{cond}(\mathbf{A}) = ||\mathbf{A}||||\mathbf{A}^{-1}|| = \frac{\sigma_1}{\sigma_m} \tag{1.104}$$

and the condition number of $\mathbf{A}$ can be re-interpreted as being related to the eccentricity of the hyperellipse image of the unit ball.

The following important properties of the condition number are easily derived from the definition using the 2−norm, and in fact hold for any norm:

1. For any matrix $\mathbf{A}$, $\text{cond}(\mathbf{A}) \geq 1$.
2. For the identity matrix, $\text{cond}(\mathbf{I}) = 1$.
3. For any matrix $\mathbf{A}$ and nonzero γ, $\text{cond}(\gamma\mathbf{A}) = \text{cond}(\mathbf{A})$.
4. For any diagonal matrix $\mathbf{D}$, $\text{cond}(\mathbf{D}) = \frac{\max_i |d_{ii}|}{\min_i |d_{ii}|}$ or in other words, is the ratio of the largest to the smallest eigenvalue (in absolute value).

Final remarks:

- These results shows that if $\mathbf{A}$ is ill-conditioned, then numerical algorithms that are prone to round-off errors will have problems both with simple matrix multiplications by $\mathbf{A}$, and with matrix multiplications by $\mathbf{A}^{-1}$ (or equivalently, with solving any linear problem of the kind $\mathbf{Ax} = \mathbf{b}$).
- The reason for this is that the condition number effectively measures how close a matrix is to being singular: a matrix with a large condition number has its smallest singular value very close to zero, which means that the matrix operation is fairly insensitive to anything that happens along that singular direction. Another way of seing this is that this direction is very close to being part of the nullspace. A matrix with a condition number close to 1 on the other hand is far from being singular.
- Notice that the determinant of a matrix is *not* a good indicator of near singularity. In other words, the magnitude of $\det(\mathbf{A})$ has no information on how close to singular the matrix $\mathbf{A}$ may be. For example, $\det(\alpha\mathbf{I}_n) = \alpha^n$. If $|\alpha| < 1$ the determinant can be very small, yet the matrix $\alpha\mathbf{I}_n$ is perfectly well-conditioned for any nonzero α.
- The usefulness of the condition number is in accessing the accuracy of solutions to linear system. However, the calculation of the condition number is not trivial as it involves the inverse of the matrix. Therefore, in practice, one often seeks for a good estimated approach to approximate condition numbers.