Jim Lambers
MAT 610
Summer Session 2009-10
Lecture 2 Notes

These notes correspond to Sections 2.2-2.4 in the text.

Vector Norms

Given vectors $\mathbf{x}$ and $\mathbf{y}$ of length one, which are simply scalars x and y, the most natural notion of distance between x and y is obtained from the absolute value; we define the distance to be $|x - y|$. We therefore define a distance function for vectors that has similar properties.

A function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ is called a *vector norm* if it has the following properties:

1. $\|\mathbf{x}\| \geq 0$ for any vector $\mathbf{x} \in \mathbb{R}^n$, and $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$
2. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ for any vector $\mathbf{x} \in \mathbb{R}^n$ and any scalar $\alpha \in \mathbb{R}$
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for any vectors $\mathbf{x}$, $\mathbf{y} \in \mathbb{R}^n$.

The last property is called the *triangle inequality*. It should be noted that when $n = 1$, the absolute value function is a vector norm.

The most commonly used vector norms belong to the family of *p-norms*, or *ℓ_p-norms*, which are defined by

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

It can be shown that for any $p > 0$, $\|\cdot\|_p$ defines a vector norm. The following p-norms are of particular interest:

- $p = 1$: The *ℓ_1-norm*
$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$
- $p = 2$: The *ℓ_2-norm* or *Euclidean norm*
$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$$
- $p = \infty$: The *ℓ_∞-norm*
$$\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

It can be shown that the ℓ_2-norm satisfies the *Cauchy-Bunyakovsky-Schwarz* inequality

$$|\mathbf{x}^T\mathbf{y}| \leq \|\mathbf{x}\|_2\|\mathbf{y}\|_2$$

for any vectors $\mathbf{x}$, $\mathbf{y} \in \mathbb{R}^n$. This inequality is useful for showing that the ℓ_2-norm satisfies the triangle inequality. It is a special case of the *Holder* inequality

$$|\mathbf{x}^T\mathbf{y}| \leq \|\mathbf{x}\|_p\|\mathbf{y}\|_q, \quad \frac{1}{p} + \frac{1}{q} = 1.$$

Now that we have defined various notions of the size, or magnitude, of a vector, we can discuss distance and convergence. Given a vector norm $\|\cdot\|$, and vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we define the *distance* between $\mathbf{x}$ and $\mathbf{y}$, with respect to this norm, by $\|\mathbf{x}-\mathbf{y}\|$. Then, we say that a sequence of n-vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ *converges* to a vector $\mathbf{x}$ if

$$\lim_{k\to\infty} \|\mathbf{x}^{(k)} - \mathbf{x}\| = 0.$$

That is, the distance between $\mathbf{x}^{(k)}$ and $\mathbf{x}$ must approach zero. It can be shown that regardless of the choice of norm, $\mathbf{x}^{(k)} \to \mathbf{x}$ if and only if

$$\mathbf{x}_i^{(k)} \to x_i, \quad i = 1, 2, \ldots, n.$$

That is, each component of $\mathbf{x}^{(k)}$ must converge to the corresponding component of x. This is due to the fact that for any vector norm $\|\cdot\|$, $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x}$ is the zero vector.

Because we have defined convergence with respect to an arbitrary norm, it is important to know whether a sequence can converge to a limit with respect to one norm, while converging to a different limit in another norm, or perhaps not converging at all. Fortunately, for p-norms, this is never the case. We say that two vector norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ are *equivalent* if there exists constants C_1 and C_2, that are independent of $\mathbf{x}$, such that for any vector $\mathbf{x} \in \mathbb{R}^n$,

$$C_1\|\mathbf{x}\|_\alpha \leq \|\mathbf{x}\|_\beta \leq C_2\|\mathbf{x}\|_\alpha.$$

It follows that if two norms are equivalent, then a sequence of vectors that converges to a limit with respect to one norm will converge to the same limit in the other. It can be shown that all ℓ_p-norms are equivalent. In particular, if $\mathbf{x} \in \mathbb{R}^n$, then

$$\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n}\|\mathbf{x}\|_2,$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_\infty,$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n\|\mathbf{x}\|_\infty.$$

Matrix Norms

It is also very useful to be able to measure the magnitude of a matrix, or the distance between matrices. However, it is not sufficient to simply define the norm of an $m \times n$ matrix A as the norm of an mn-vector $\mathbf{x}$ whose components are the entries of A. We instead define a *matrix norm* to be a function $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ that has the following properties:

- $\|A\| \geq 0$ for any $A \in \mathbb{R}^{m\times n}$, and $\|A\| = 0$ if and only if $A = 0$
- $\|\alpha A\| = |\alpha|\|A\|$ for any $m \times n$ matrix A and scalar α
- $\|A + B\| \leq \|A\| + \|B\|$ for any $m \times n$ matrices A and B

Another property that is often, but not always, included in the definition of a matrix norm is the *submultiplicative property*: if A is $m \times n$ and B is $n \times p$, we require that

$$\|AB\| \leq \|A\|\|B\|.$$

This is particularly useful when A and B are square matrices.

Any vector norm *induces* a matrix norm. It can be shown that given a vector norm, defined appropriately for m-vectors and n-vectors, the function $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ defined by

$$\|A\| = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|$$

is a matrix norm. It is called the *natural*, or *induced*, matrix norm. Furthermore, if the vector norm is a ℓ_p-norm, then the induced matrix norm satisfies the submultiplicative property.

The following matrix norms are of particular interest:

- The ℓ_1-*norm*:

  $$\|A\|_1 = \max_{\|\mathbf{x}\|_1=1} \|A\mathbf{x}\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|.$$

  That is, the ℓ_1-norm of a matrix is its maximum column sum.

- The ℓ_∞-*norm*:

  $$\|A\|_\infty = \max_{\|\mathbf{x}\|_\infty=1} \|A\mathbf{x}\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|.$$

  That is, the ℓ_∞-norm of a matrix is its maximum *row* sum.

- The ℓ_2-*norm*:

  $$\|A\|_2 = \max_{\|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2.$$

To obtain a formula for this norm, we note that the function

$$g(\mathbf{x}) = \frac{\|A\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}$$

has a local maximium or minimum whenever $\mathbf{x}$ is a *unit* ℓ_2-norm vector (that is, $\|\mathbf{x}\|_2 = 1$) that satisfies

$$A^T A\mathbf{x} = \|A\mathbf{x}\|_2^2 \mathbf{x},$$

as can be shown by differentiation of $g(\mathbf{x})$. That is, $\mathbf{x}$ is an *eigenvector* of $A^T A$, with corresponding *eigenvalue* $\|A\mathbf{x}\|_2^2 = g(\mathbf{x})$. We conclude that

$$\|A\|_2 = \max_{1 \leq i \leq n} \sqrt{\lambda_i(A^T A)}.$$

That is, the ℓ_2-norm of a matrix is the square root of the largest eigenvalue of $A^T A$, which is guaranteed to be nonnegative, as can be shown using the vector 2-norm. We see that unlike the vector ℓ_2-norm, the matrix ℓ_2-norm is much more difficult to compute than the matrix ℓ_1-norm or ℓ_∞-norm.

- The *Frobenius norm*:

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

It should be noted that the Frobenius norm is *not* induced by any vector ℓ_p-norm, but it is equivalent to the vector ℓ_2-norm in the sense that $\|A\|_F = \|\mathbf{x}\|_2$ where $\mathbf{x}$ is obtained by reshaping A into a vector.

Like vector norms, matrix norms are equivalent. For example, if A is an $m \times n$ matrix, we have

$$\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2,$$

$$\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{m}\|A\|_\infty,$$

$$\frac{1}{\sqrt{m}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_\infty.$$

Eigenvalues and Eigenvectors

We have learned what it means for a sequence of vectors to converge to a limit. However, using the definition alone, it may still be difficult to determine, conclusively, whether a given sequence of

vectors converges. For example, suppose a sequence of vectors is defined as follows: we choose the initial vector $\mathbf{x}^{(0)}$ arbitrarily, and then define the rest of the sequence by

$$\mathbf{x}^{(k+1)} = A\mathbf{x}^{(k)}, \quad k = 0, 1, 2, \ldots$$

for some matrix A. Such a sequence actually arises in the study of the convergence of various iterative methods for solving systems of linear equations.

An important question is whether a sequence of this form converges to the zero vector. This will be the case if

$$\lim_{k\to\infty} \|\mathbf{x}^{(k)}\| = 0$$

in some vector norm. From the definition of $\mathbf{x}^{(k)}$, we must have

$$\lim_{k\to\infty} \|A^k\mathbf{x}^{(0)}\| = 0.$$

From the submultiplicative property of matrix norms,

$$\|A^k\mathbf{x}^{(0)}\| \leq \|A\|^k\|\mathbf{x}^{(0)}\|,$$

from which it follows that the sequence will converge to the zero vector if $\|A\| < 1$. However, this is only a *sufficient* condition; it is not *necessary*.

To obtain a sufficient *and* necessary condition, it is necessary to achieve a better understanding of the effect of matrix-vector multiplication on the magnitude of a vector. However, because matrix-vector multiplication is a complicated operation, this understanding can be difficult to acquire. Therefore, it is helpful to identify circumstances under which this operation can be simply described.

To that end, we say that a nonzero vector $\mathbf{x}$ is an *eigenvector* of an $n \times n$ matrix A if there exists a scalar λ such that

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

The scalar λ is called an *eigenvalue* of A corresponding to $\mathbf{x}$. Note that although $\mathbf{x}$ is required to be nonzero, it is possible that λ can be zero. It can also be complex, even if A is a real matrix.

If we rearrange the above equation, we have

$$(A - \lambda I)\mathbf{x} = \mathbf{0}.$$

That is, if λ is an eigenvalue of A, then $A - \lambda I$ is a singular matrix, and therefore $\det(A - \lambda I) = 0$. This equation is actually a polynomial in λ, which is called the *characteristic polynomial* of A. If A is an $n \times n$ matrix, then the characteristic polynomial is of degree n, which means that A has n eigenvalues, which may repeat.

The following properties of eigenvalues and eigenvectors are helpful to know:

- If λ is an eigenvalue of A, then there is at least one eigenvector of A corresponding to λ

- If there exists an invertible matrix P such that $B = PAP^{-1}$, then A and B have the same eigenvalues. We say that A and B are *similar*, and the transformation PAP^{-1} is called a *similarity transformation*.
- If A is a symmetric matrix, then its eigenvalues are real.
- If A is a *skew-symmetric* matrix, meaning that $A^T = -A$, then its eigenvalues are either equal to zero, or are purely imaginary.
- If A is a real matrix, and $\lambda = u + iv$ is a complex eigenvalue of A, then $\bar{\lambda} = u - iv$ is also an eigenvalue of A.
- If A is a triangular matrix, then its diagonal entries are the eigenvalues of A.
- $\det(A)$ is equal to the product of the eigenvalues of A.
- $\operatorname{tr}(A)$, the sum of the diagonal entries of A, is also equal to the sum of the eigenvalues of A.

It follows that any method for computing the roots of a polynomial can be used to obtain the eigenvalues of a matrix A. However, in practice, eigenvalues are normally computed using iterative methods that employ orthogonal similarity transformations to reduce A to upper triangular form, thus revealing the eigenvalues of A. In practice, such methods for computing eigenvalues are used to compute roots of polynomials, rather than using polynomial root-finding methods to compute eigenvalues, because they are much more robust with respect to roundoff error.

It can be shown that if each eigenvalue λ of a matrix A satisfies $|\lambda| < 1$, then, for any vector $\mathbf{x}$,

$$\lim_{k\to\infty} A^k\mathbf{x} = \mathbf{0}.$$

Furthermore, the converse of this statement is also true: if there exists a vector $\mathbf{x}$ such that $A^k\mathbf{x}$ does not approach $\mathbf{0}$ as $k \to \infty$, then at least one eigenvalue λ of A must satisfy $|\lambda| \geq 1$.

Therefore, it is through the eigenvalues of A that we can describe a necessary and sufficient condition for a sequence of vectors of the form $\mathbf{x}^{(k)} = A^k\mathbf{x}^{(0)}$ to converge to the zero vector. Specifically, we need only check if the magnitude of the largest eigenvalue is less than 1. For convenience, we define the *spectral radius* of A, denoted by $\rho(A)$, to be $\max|\lambda|$, where λ is an eigenvalue of A. We can then conclude that the sequence $\mathbf{x}^{(k)} = A^k\mathbf{x}^{(0)}$ converges to the zero vector if and only if $\rho(A) < 1$.

The spectral radius is closely related to natural (induced) matrix norms. Let λ be the largest eigenvalue of A, with $\mathbf{x}$ being a corresponding eigenvector. Then, for any natural matrix norm $\|\cdot\|$, we have

$$\rho(A)\|\mathbf{x}\| = |\lambda|\|\mathbf{x}\| = \|\lambda\mathbf{x}\| = \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|.$$

Therefore, we have $\rho(A) \leq \|A\|$. When A is symmetric, we also have

$$\|A\|_2 = \rho(A).$$

For a general matrix A, we have

$$\|A\|_2 = [\rho(A^T A)]^{1/2},$$

which can be seen to reduce to $\rho(A)$ when $A^T = A$, since, in general, $\rho(A^k) = \rho(A)^k$.

Because the condition $\rho(A) < 1$ is necessary and sufficient to ensure that $\lim_{k\to\infty} A^k \mathbf{x} = \mathbf{0}$, it is possible that such convergence may occur even if $\|A\| \geq 1$ for some natural norm $\|\cdot\|$. However, if $\rho(A) < 1$, we can conclude that

$$\lim_{k\to\infty} \|A^k\| = 0,$$

even though $\lim_{k\to\infty} \|A\|^k$ may not even exist.

In view of the definition of a matrix norm, that $\|A\| = 0$ if and only if $A = 0$, we can conclude that if $\rho(A) < 1$, then A^k converges to the zero matrix as $k \to \infty$. In summary, the following statements are all equivalent:

1. $\rho(A) < 1$
2. $\lim_{k\to\infty} \|A^k\| = 0$, for any natural norm $\|\cdot\|$
3. $\lim_{k\to\infty} (A^k)_{ij} = 0$, $i, j = 1, 2, \ldots, n$
4. $\lim_{k\to\infty} A^k \mathbf{x} = \mathbf{0}$

We will see that these results are very useful for analyzing the convergence behavior of various iterative methods for solving systems of linear equations.

Perturbations and the Inverse

Using what we have learned about matrix norms and convergence of sequences of matrices, we can quantify the change in the inverse of a matrix A in terms of the change in A. Suppose that an $n \times n$ matrix F satisfies $\|F\|_p < 1$, for some p. Then, from our previous discussion, $\lim_{k\to\infty} F^k = 0$. It follows that, by convergence of telescoping series,

$$\left(\lim_{k\to\infty} \sum_{i=0}^{k} F^i\right)(I - F) = \lim_{k\to\infty} I - F^{k+1} = I,$$

and therefore $(I - F)$ is nonsingular, with inverse $(I - F)^{-1} = \sum_{i=0}^{\infty} F^i$. By the properties of matrix norms, and convergence of geometric series, we then obtain

$$\|(I - F)^{-1}\|_p \leq \sum_{i=0}^{\infty} \|F\|_p^i = \frac{1}{1 - \|F\|_p}.$$

Now, let A be a nonsingular matrix, and let E be a perturbation of A such that $r \equiv \|A^{-1}E\|_p < 1$. Because $A + E = A(I - F)$ where $F = -A^{-1}E$, with $\|F\|_p = r < 1$, $I - F$ is nonsingular, and therefore so is $A + E$. We then have

$$\begin{aligned}
\|(A+E)^{-1} - A^{-1}\|_p &= \| - A^{-1}E(A+E)^{-1}\|_p \\
&= \| - A^{-1}E(I-F)^{-1}A^{-1}\|_p \\
&\leq \|A^{-1}\|_p^2\|E\|_p\|(I-F)^{-1}\|_p \\
&\leq \frac{\|A^{-1}\|_p^2\|E\|_p}{1-r},
\end{aligned}$$

from the formula for the difference of inverses, and the submultiplicative property of matrix norms.

Roundoff Errors and Computer Arithmetic

In computing the solution to any mathematical problem, there are many sources of error that can impair the accuracy of the computed solution. The study of these sources of error is called *error analysis*, which will be discussed later in this lecture. First, we will focus on one type of error that occurs in all computation, whether performed by hand or on a computer: *roundoff error*.

This error is due to the fact that in computation, real numbers can only be represented using a finite number of digits. In general, it is not possible to represent real numbers exactly with this limitation, and therefore they must be approximated by real numbers that *can* be represented using a fixed number of digits, which is called the *precision*. Furthermore, as we shall see, arithmetic operations applied to numbers that can be represented exactly using a given precision do not necessarily produce a result that can be represented using the same precision. It follows that if a fixed precision is used, then *every* arithmetic operation introduces error into a computation.

Given that scientific computations can have several sources of error, one would think that it would be foolish to compound the problem by performing arithmetic using fixed precision. However, using a fixed precision is actually far more practical than other options, and, as long as computations are performed carefully, sufficient accuracy can still be achieved.

Floating-Point Numbers

We now describe a typical system for representing real numbers on a computer.

Definition *(Floating-point Number System) Given integers $\beta > 1$, $p \geq 1$, L, and $U \geq L$, a* **floating-point number system** $\mathbb{F}$ *is defined to be the set of all real numbers of the form*

$$x = \pm m\beta^E.$$

The number m is the **mantissa** *of x, and has the form*

$$m = \left(\sum_{j=0}^{p-1} d_j \beta^{-j} \right),$$

where each digit d_j, $j = 0, \ldots, p-1$ is an integer satisfying $0 \leq d_i \leq \beta - 1$. The number E is called the **exponent** *or the* **characteristic** *of x, and it is an integer satisfying $L \leq E \leq U$. The integer p is called the* **precision** *of $\mathbb{F}$, and β is called the* **base** *of $\mathbb{F}$.*

The term "floating-point" comes from the fact that as a number $x \in \mathbb{F}$ is multiplied by or divided by a power of β, the mantissa does not change, only the exponent. As a result, the decimal point shifts, or "floats," to account for the changing exponent.

Nearly all computers use a binary floating-point system, in which $\beta = 2$. In fact, most computers conform to the *IEEE standard* for floating-point arithmetic. The standard specifies, among other things, how floating-point numbers are to be represented in memory. Two representations are given, one for *single-precision* and one for *double-precision*. Under the standard, single-precision floating-point numbers occupy 4 bytes in memory, with 23 bits used for the mantissa, 8 for the exponent, and one for the sign. IEEE double-precision floating-point numbers occupy eight bytes in memory, with 52 bits used for the mantissa, 11 for the exponent, and one for the sign.

Example Let $x = -117$. Then, in a floating-point number system with base $\beta = 10$, x is represented as

$$x = -(1.17)10^2,$$

where 1.17 is the mantissa and 2 is the exponent. If the base $\beta = 2$, then we have

$$x = -(1.110101)2^6,$$

where 1.110101 is the mantissa and 6 is the exponent. The mantissa should be interpreted as a string of binary digits, rather than decimal digits; that is,

$$\begin{aligned} 1.110101 &= 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} \\ &= 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} \\ &= \frac{117}{64} \\ &= \frac{117}{2^6}. \end{aligned}$$

□

Properties of Floating-Point Systems

A floating-point system $\mathbb{F}$ can only represent a finite subset of the real numbers. As such, it is important to know how large in magnitude a number can be and still be represented, at least approximately, by a number in $\mathbb{F}$. Similarly, it is important to know how small in magnitude a number can be and still be represented by a *nonzero* number in $\mathbb{F}$; if its magnitude is too small, then it is most accurately represented by zero.

Definition *(Underflow, Overflow) Let $\mathbb{F}$ be a floating-point number system. The smallest positive number in $\mathbb{F}$ is called the* **underflow level***, and it has the value*

$$UFL = m_{\min}\,\beta^L,$$

where L is the smallest valid exponent and $m_{\min}$ is the smallest mantissa. The largest positive number in $\mathbb{F}$ is called the overflow level*, and it has the value*

$$OFL = \beta^{U+1}(1 - \beta^{-p}).$$

The value of $m_{\min}$ depends on whether floating-point numbers are *normalized* in $\mathbb{F}$; this point will be discussed later. The overflow level is the value obtained by setting each digit in the mantissa to $\beta - 1$ and using the largest possible value, U, for the exponent.

It is important to note that the real numbers that can be represented in $\mathbb{F}$ are not equally spaced along the real line. Numbers having the same exponent are equally spaced, and the spacing between numbers in $\mathbb{F}$ decreases as their magnitude decreases.

Normalization

It is common to *normalize* floating-point numbers by specifying that the leading digit d_0 of the mantissa be nonzero. In a binary system, with $\beta = 2$, this implies that the leading digit is equal to 1, and therefore need not be stored. Therefore, in the IEEE floating-point standard, $p = 24$ for single precision, and $p = 53$ for double precision, even though only 23 and 52 bits, respectively, are used to store mantissas. In addition to the benefit of gaining one additional bit of precision, normalization also ensures that each floating-point number has a unique representation.

One drawback of normalization is that fewer numbers near zero can be represented exactly than if normalization is not used. Therefore, the IEEE standard provides for a practice called *gradual underflow*, in which the leading digit of the mantissa is allowed to be zero when the exponent is equal to L, thus allowing smaller values of the mantissa. In such a system, the number UFL is equal to β^{L-p+1}, whereas in a normalized system, UFL $= \beta^L$.

Rounding

A number that can be represented exactly in a floating-point system is called a *machine number*. Since only finitely many real numbers are machine numbers, it is necessary to determine how non-

machine numbers are to be approximated by machine numbers. The process of choosing a machine number to approximate a non-machine number is called *rounding*, and the error introduced by such an approximation is called *roundoff error*. Given a real number x, the machine number obtained by rounding x is denoted by $\text{fl}(x)$.

In most floating-point systems, rounding is achieved by one of two strategies:

- *chopping*, or *rounding to zero*, is the simplest strategy, in which the base-β expansion of a number is truncated after the first p digits. As a result, $\text{fl}(x)$ is the unique machine number between 0 and x that is nearest to x.
- *rounding to nearest* sets $\text{fl}(x)$ to be the machine number that is closest to x in absolute value; if two numbers satisfy this property, then an appropriate tie-breaking rule must be used, such as setting $\text{fl}(x)$ equal to the choice whose last digit is even.

Example Suppose we are using a floating-point system with $\beta = 10$ (decimal), with $p = 4$ significant digits. Then, if we use chopping, or rounding to zero, we have $\text{fl}(2/3) = 0.6666$, whereas if we use rounding to nearest, then we have $\text{fl}(2/3) = 0.6667$. □

Machine Precision

In error analysis, it is necessary to estimate error incurred in each step of a computation. As such, it is desirable to know an upper bound for the relative error introduced by rounding. This leads to the following definition.

Definition *(Machine Precision) Let $\mathbb{F}$ be a floating-point number system. The* **unit roundoff** *or* **machine precision***, denoted by* $\mathbf{u}$*, is the real number that satisfies*

$$\left|\frac{\text{fl}(x) - x}{x}\right| \leq \mathbf{u}$$

for any real number x such that $\text{UFL} < x < \text{OFL}$.

An intuitive definition of $\mathbf{u}$ is that it is the smallest positive number such that

$$\text{fl}\left(1 + \mathbf{u}\right) > 1.$$

The value of $\mathbf{u}$ depends on the rounding strategy that is used. If rounding toward zero is used, then $\mathbf{u} = \beta^{1-p}$, whereas if rounding to nearest is used, $\mathbf{u} = \frac{1}{2}\beta^{1-p}$.

It is important to avoid confusing $\mathbf{u}$ with the underflow level UFL. The unit roundoff is determined by the number of digits in the mantissa, whereas the underflow level is determined by the range of allowed exponents. However, we do have the relation that $0 < \text{UFL} < \mathbf{u}$.

Example The following table summarizes the main aspects of a general floating-point system and a double-precision floating-point system that uses a 52-bit mantissa and 11-bit exponent. For both systems, we assume that rounding to nearest is used, and that normalization is used. □

	General	Double Precision
Form of machine number	$\pm m\beta^E$	$\pm 1.d_1d_2\cdots d_{52}2^E$
Precision	p	53
Exponent range	$L \leq E \leq U$	$-1023 \leq E \leq 1024$
UFL (Underflow Level)	β^L	2^{-1023}
OFL (Overflow Level)	$\beta^{U+1}(1-\beta^{-p})$	$2^{1025}(1-2^{-53})$
u	$\frac{1}{2}\beta^{1-p}$	2^{-53}

Floating-Point Arithmetic

We now discuss the various issues that arise when performing *floating-point arithmetic*, or *finite-precision arithmetic*, which approximates arithmetic operations on real numbers.

When adding or subtracting floating-point numbers, it is necessary to shift one of the operands so that both operands have the same exponent, before adding or subtracting the mantissas. As a result, digits of precision are lost in the operand that is smaller in magnitude, and the result of the operation cannot be represented using a machine number. In fact, if x is the smaller operand and y is the larger operand, and $|x| < |y|\mathbf{u}$, then the result of the operation will simply be y (or $-y$, if y is to be subtracted from x), since the entire value of x is lost in rounding the result.

In multiplication or division, the operands need not be shifted, but the mantissas, when multiplied or divided, cannot necessarily be represented using only p digits of precision. The product of two mantissas requires $2p$ digits to be represented exactly, while the quotient of two mantissas could conceivably require infinitely many digits. Furthermore, overflow or underflow may occur depending on the exponents of the operands, since their sum or difference may lie outside of the interval $[L, U]$.

Because floating-point arithmetic operations are not exact, they do not follow all of the laws of real arithmetic. In particular, floating-point arithmetic is not associative; i.e., $x+(y+z) \neq (x+y)+z$ in floating-point arithmetic.

Cancellation

Subtraction of floating-point numbers presents a unique difficulty, in addition to the rounding error previously discussed. If the operands, after shifting exponents as needed, have leading digits in common, then these digits cancel and the first digit in which the operands do not match becomes the leading digit. However, since each operand is represented using only p digits, it follows that the result contains only $p - m$ correct digits, where m is the number of leading digits that cancel.

In an extreme case, if the two operands differ by less than **u**, then the result contains no correct digits; it consists entirely of roundoff error from previous computations. This phenomenon is known as *catastrophic cancellation*. Because of the highly detrimental effect of this cancellation, it is important to ensure that no steps in a computation compute small values from relatively large operands. Often, computations can be rearranged to avoid this risky practice.

Example Consider the quadratic equation

$$ax^2 + bx + c = 0,$$

which has the solutions

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Suppose that $b > 0$. Then, in computing x_1, we encounter catastrophic cancellation if b is much larger than a and c, because this implies that $\sqrt{b^2 - 4ac} \approx b$ and as a result we are subtracting two numbers that are nearly equal in computing the numerator. On the other hand, if $b < 0$, we encounter this same difficulty in computing x_2.

Suppose that we use 4-digit rounding arithmetic to compute the roots of the equation

$$x^2 + 10,000x + 1 = 0.$$

Then, we obtain $x_1 = 0$ and $x_2 = -10,000$. Clearly, x_1 is incorrect because if we substitute $x = 0$ into the equation then we obtain the contradiction $1 = 0$. In fact, if we use 7-digit rounding arithmetic then we obtain the same result. Only if we use at least 8 digits of precision do we obtain roots that are reasonably correct,

$$x_1 \approx -1 \times 10^{-4}, \quad x_2 \approx -9.9999999 \times 10^3.$$

A similar result is obtained if we use 4-digit rounding arithmetic but compute x_1 using the formula

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}},$$

which can be obtained by multiplying and dividing the original formula for x_1 by the *conjugate* of the numerator, $-b - \sqrt{b^2 - 4ac}$. The resulting formula is not susceptible to catastrophic cancellation, because an addition is performed instead of a subtraction. □

Basics of Error Analysis

Intuitively, it is not difficult to conclude that any scientific computation can include several approximations, each of which introduces error in the computed solution. Therefore, it is necessary to understand the effects of these approximations on accuracy. The study of these effects is known as *error analysis*.

Error analysis will be a recurring theme in this course. For this reason, we will introduce some basic concepts that will play a role in error analyses of specific algorithms in later lectures.

Absolute Error and Relative Error

Now that we have been introduced to some specific errors that can occur during computation, we introduce useful terminology for discussing such errors. Suppose that a real number $\hat{y}$ is an approximation to some real number y. For instance, $\hat{y}$ may be the closest number to y that can be represented using finite precision, or $\hat{y}$ may be the result of a sequence of arithmetic operations performed using finite-precision arithmetic, where y is the result of the same operations performed using exact arithmetic.

Definition *(Absolute Error, Relative Error) Let $\hat{y}$ be a real number that is an approximation to the real number y. The* **absolute error** *in $\hat{y}$ is*

$$E_{abs} = \hat{y} - y.$$

The **relative error** *in $\hat{y}$ is*

$$E_{rel} = \frac{\hat{y} - y}{y},$$

provided that y is nonzero.

The absolute error is the most natural measure of the accuracy of an approximation, but it can be misleading. Even if the absolute error is small in magnitude, the approximation may still be grossly inaccurate if the exact value y is even smaller in magnitude. For this reason, it is preferable to measure accuracy in terms of the relative error.

The magnitude of the relative error in $\hat{y}$ can be interpreted as a percentage of $|y|$. For example, if the relative error is greater than 1 in magnitude, then $\hat{y}$ can be considered completely erroneous, since the error is larger in magnitude as the exact value. Another useful interpretation of the relative error concerns *significant digits*, which are all digits excluding leading zeros. Specifically, if the relative error is at most β^{-p}, where β is an integer greater than 1, then the representation of $\hat{y}$ in base β has at least p correct significant digits.

It should be noted that the absolute error and relative error are often defined using absolute value; that is,

$$E_{abs} = |\hat{y} - y|, \quad E_{rel} = \left| \frac{\hat{y} - y}{y} \right|.$$

This definition is preferable when one is only interested in the magnitude of the error, which is often the case. If the sign, or direction, of the error is also of interest, then the first definition must be used.

We also note that if the quantities y and $\hat{y}$ are vectors or matrices, rather than numbers, then appropriate norms can instead be used to measure absolute and relative error. For example, given an exact value $\mathbf{y}$ and approximation $\hat{\mathbf{y}}$, that are both vectors in $\mathbb{R}^n$, we can define the absolute error and relative error by

$$E_{abs} = \|\hat{\mathbf{y}} - \mathbf{y}\|, \quad E_{rel} = \frac{\|\hat{\mathbf{y}} - \mathbf{y}\|}{\|\mathbf{y}\|},$$

where $\|\cdot\|$ is any vector norm.

Example Assume that we are using a floating-point system that uses base $\beta = 10$ (decimal), with 4 digits of precision and rounding to nearest. Then, if we add the numbers 0.4567×10^0 and 0.8580×10^{-2}, we obtain the exact result

$$x = 0.4567 \times 10^0 + 0.008530 \times 10^0 = 0.46523 \times 10^0,$$

which is rounded to

$$\text{fl}(x) = 0.4652 \times 10^0.$$

The absolute error in this computation is

$$E_{abs} = \text{fl}(x) - x = 0.4652 - 0.46523 = -0.00003,$$

while the relative error is

$$E_{rel} = \frac{\text{fl}(x) - x}{x} = \frac{0.4652 - 0.46523}{0.46523} \approx -0.000064484.$$

Using the same floating-point system, suppose that we multiply 0.4567×10^4 and 0.8530×10^{-2}. The exact result is

$$x = (0.4567 \times 10^4) \times (0.8530 \times 10^{-2}) = 0.3895651 \times 10^2 = 38.95651,$$

which is rounded to

$$\text{fl}(x) = 0.3896 \times 10^2 = 38.96.$$

The absolute error in this computation is

$$E_{abs} = \text{fl}(x) - x = 38.96 - 38.95651 = 0.00349,$$

while the relative error is

$$E_{rel} = \frac{\text{fl}(x) - x}{x} = \frac{38.96 - 38.95651}{38.95651} \approx 0.000089587.$$

We see that in this case, the relative error is smaller than the absolute error, because the exact result is larger than 1, whereas in the previous operation, the relative error was larger in magnitude, because the exact result is smaller than 1. □

Forward Error and Backward Error

Suppose that we compute an approximation $\hat{y} = \hat{f}(x)$ of the value $y = f(x)$ for a given function f and given problem data x. Before we can analyze the accuracy of this approximation, we must have a precisely defined notion of error in such an approximation. We now provide this precise definition.

Definition *(Forward Error) Let x be a real number and let $f : \mathbb{R} \to \mathbb{R}$ be a function. If $\hat{y}$ is a real number that is an approximation to $y = f(x)$, then the* **forward error** *in $\hat{y}$ is the difference $\Delta y = \hat{y} - y$. If $y \neq 0$, then the* **relative forward error** *in $\hat{y}$ is defined by*

$$\frac{\Delta y}{y} = \frac{\hat{y} - y}{y}.$$

Clearly, our primary goal in error analysis is to obtain an estimate of the forward error Δy. Unfortunately, it can be difficult to obtain this estimate directly.

An alternative approach is to instead view the computed value $\hat{y}$ as the *exact* solution of a problem with modified data; i.e., $\hat{y} = f(\hat{x})$ where $\hat{x}$ is a perturbation of x.

Definition *(Backward Error) Let x be a real number and let $f : \mathbb{R} \to \mathbb{R}$ be a function. Suppose that the real number $\hat{y}$ is an approximation to $y = f(x)$, and that $\hat{y}$ is in the range of f; that is, $\hat{y} = f(\hat{x})$ for some real number $\hat{x}$. Then, the quantity $\Delta x = \hat{x} - x$ is the* **backward error** *in $\hat{y}$. If $x \neq 0$, then the* **relative backward error** *in $\hat{y}$ is defined by*

$$\frac{\Delta x}{x} = \frac{\hat{x} - x}{x}.$$

The process of estimating Δx is known as *backward error analysis*. As we will see, this estimate of the backward error, in conjunction with knowledge of f, can be used to estimate the forward error.

As discussed previously, floating-point arithmetic does not follow the laws of real arithmetic. This tends to make forward error analysis difficult. In backward error analysis, however, real arithmetic is employed, since it is assumed that the computed result is the exact solution to a modified problem. This is one reason why backward error analysis is sometimes preferred.

In analysis of roundoff error, it is assumed that $\text{fl}(x \texttt{ op } y) = (x \texttt{ op } y)(1 + \delta)$, where `op` is an arithmetic operation and δ is an unknown constant satisfying $|\delta| \leq \mathbf{u}$. From this assumption, it can be seen that the relative error in $\text{fl}(x \texttt{ op } y)$ is $|\delta|$. In the case of addition, the relative backward error in each operand is also $|\delta|$.