Principles of Scientific Computing
Linear Algebra I, Theory and Conditioning

David Bindel and Jonathan Goodman

last revised February 9, 2006, printed February 25, 2009
1 Introduction

Linear algebra and calculus are the basic tools of quantitative science. The operations of linear algebra include solving systems of equations, finding subspaces, solving least squares problems, factoring matrices, and computing eigenvalues and eigenvectors. There is good, publicly available software to perform these computations, and in most cases this software is faster and more accurate than code you write yourself. Chapter ?? outlines some of the basic algorithms of computational linear algebra. This chapter discusses more basic material.

Conditioning is a major concern in many linear algebra computations. Easily available linear algebra software is backward stable, which essentially means that the results are as accurate as the conditioning of the problem allows. Even a backward stable method produces large errors if the condition number is of the order of $1/\epsilon_{\text{mach}}$. For example, if the condition number is $10^{18}$, even double precision calculations are likely to yield a completely wrong answer. Unfortunately, such condition numbers occur in problems that are not terribly large or rare.

If a computational method for a well-conditioned problem is unstable (much less accurate than its conditioning allows), it is likely because one of the subproblems is ill-conditioned. For example, the problem of computing the matrix exponential, $e^A$, may be well-conditioned while the problem of computing the eigenvectors of $A$ is ill-conditioned. A stable algorithm for computing $e^A$ (see Exercise 12) in that case must avoid using the eigenvectors of $A$.

The condition number of a problem (see Section ??) measures how small perturbations in the data affect the answer. This is called perturbation theory. Suppose $A$ is a matrix and $f(A)$ is the solution of a linear algebra problem involving $A$, such as $x$ that satisfies $Ax = b$, or $\lambda$ and $v$ that satisfy $Av = \lambda v$. Perturbation theory seeks to estimate $\Delta f = f(A + \Delta A) - f(A)$ when $\Delta A$ is small. Usually, this amounts to calculating the derivative of $f$ with respect to $A$.

We simplify the results of perturbation calculations using bounds that involve vector or matrix norms. For example, suppose we want to say that all the entries in $\Delta A$ or $\Delta v$ are small. For a vector, $v$, or a matrix, $A$, the norm, $\|v\|$ or $\|A\|$, is a number that characterizes the size of $v$ or $A$. Using norms, we can say that the relative size of a perturbation in $A$ is $\|\Delta A\|/\|A\|$.

The condition number of a problem involving $A$ depends on the problem as well as on $A$. The condition number of $f(A) = A^{-1}b$ (i.e. solving the system of linear equations $Ax = b$) is very different from the problem of finding the eigenvalues of $A$. There are matrices that have well conditioned eigenvalues but poorly conditioned eigenvectors. What is commonly called “the” condition number of $A$ is the worst case condition number of solving $Ax = b$, taking the worst possible $b$.

---

1 The precise definition of backward stability is in Chapter ??.
2 This notation replaces our earlier $A(x)$. In linear algebra, $A$ always is a matrix and $x$ never is a matrix.
2 Review of linear algebra

This section reviews some linear algebra that we will use later. It is not a substitute for a course on linear algebra. We assume that most of the topics are familiar to the reader. People come to scientific computing with vastly differing perspectives on linear algebra, and will know some of the concepts we describe by different names and notations. This section should give everyone a common language.

Much of the power of linear algebra comes from this interaction between the abstract and the concrete. Our review connects the abstract language of vector spaces and linear transformations to the concrete language of matrix algebra. There may be more than one concrete representation corresponding to any abstract linear algebra problem. We will find that different representations often lead to numerical methods with very different properties.

2.1 Vector spaces

A vector space is a set of elements that may be added and multiplied by scalars (either real or complex numbers, depending on the application). Vector addition is commutative \((u + v = v + u)\) and associative \(((u + v) + w = u + (v + w))\). Multiplication by scalars is distributive over vector addition \(a(u + v) = au + av\) and \((a + b)u = au + bu\) for scalars \(a\) and \(b\) and vectors \(u\) and \(v\). There is a unique zero vector, 0, with \(0 + u = u\) for any vector \(u\).

The standard vector spaces are \(\mathbb{R}^n\) (or \(\mathbb{C}^n\), consisting of column vectors

\[
\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix},
\]

where the components, \(u_k\), are arbitrary real (or complex) numbers. Vector addition and scalar multiplication are done componentwise.

A subset \(V'\) of a vector space \(V\) is a subspace of \(V\) if sums and scalar multiples of elements in \(V'\) remain in \(V'\). That is, \(V'\) is closed under vector addition and scalar multiplication. This means \(V'\) is also a vector space under the vector addition and scalar multiplication operations of \(V\). For example, suppose \(V = \mathbb{R}^n\) and \(V'\) consists of all vectors whose components sum to zero \((\sum_{k=1}^{n} u_k = 0)\). If we add two such vectors or multiply by a scalar, the result also has the zero sum property. On the other hand, the set of vectors whose components sum to one \((\sum_{k=1}^{n} u_k = 1)\) is not closed under vector addition or scalar multiplication.

---

3 Physicists use the word "scalar" in a different way. For them, a scalar is a number that is the same in any coordinate system. The components of a vector in a particular basis are not scalars in this sense.
The span of a set of vectors $\text{span}(f_1, f_2, \ldots, f_n) \subset V$ is the subspace of $V$ consisting of linear combination of the vectors $f_j$:

$$u = u_1 f_1 + \cdots + u_n f_n,$$

(1)

where $u_k$ are scalar coefficients. We say $f_1, \ldots, f_n$ are linearly independent if $u = 0$ implies that $u_k = 0$ for all $k$ in (1). Recall that the $f_j$ are linearly independent if and only if the representation (1) uniquely determines the expansion coefficients, $u_k$. A theorem of linear algebra states that if the $f_j$ are not linearly independent, then it is possible to find a subset of them with the same span. If $V = \text{span}(f_1, \ldots, f_n)$ and the $f_j$ are linearly independent, then the $f_j$ are a basis for $V$.

The standard vector spaces $\mathbb{R}^n$ and $\mathbb{C}^n$ have standard bases $\{e_1, \ldots, e_n\}$, where $e_k$ is the vector with all zero components but for a single 1 in position $k$. This is a basis because

$$u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = u_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + u_2 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + u_n \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = \sum_{k=1}^{n} u_k e_k.$$

In view of this, there is little distinction between coordinates, components, and expansion coefficients, all of which are denoted $u_k$. If $V$ has a basis with $n$ elements, we say the dimension of $V$ is $n$. It is possible to make this definition because of the theorem that states that every basis of $V$ has the same number of elements. A vector space that does not have a finite basis is called infinite-dimensional.

An inner product space is a vector space that has an inner product $\langle \cdot, \cdot \rangle$, which is a scalar function of two vector arguments with the following properties:

1. $\langle u, av_1 + bv_2 \rangle = a \langle u, v_1 \rangle + b \langle u, v_2 \rangle$;
2. $\langle u, v \rangle = \overline{\langle v, u \rangle}$, where $\overline{z}$ refers to the complex conjugate of $z$;
3. $\langle u, u \rangle \geq 0$;
4. $\langle u, u \rangle = 0$ if and only if $u = 0$.

When $u$ and $v$ are vectors with $\langle u, v \rangle = 0$, we say $u$ and $v$ are orthogonal. If $u$ and $v$ are $n$ component column vectors ($u \in \mathbb{C}^n$, $v \in \mathbb{C}^n$), their standard inner product (sometimes called the dot product) is

$$\langle u, v \rangle = \sum_{k=1}^{n} u_k v_k.$$

(2)

The complex conjugates are not needed when the entries of $u$ and $v$ are real.

\footnote{An infinite dimensional vector space might have an infinite basis.}
Hermite polynomials are useful in probability because if \( X \) \( \) is a linear combination of powers of \( x \), then they are uncorrelated:

\[
\text{power that it contains. The product } \frac{1}{3}(x-1)^2(x^3-3x)^6 \text{ has degree 20. The vector space } P_d \text{ is the set of all polynomials of degree at most } d. \text{ This space has a basis consisting of } d+1 \text{ elements:}
\]

\[
f_0 = 1, \quad f_1 = x, \quad \ldots, \quad f_d = x^d.
\] (3)

The power basis (3) is one basis for \( P_3 \) (with \( d = 3 \), so \( P_3 \) has dimension 4). Another basis consists of the first four Hermite polynomials

\[
H_0 = 1, \quad H_1 = x, \quad H_2 = x^2 - 1, \quad H_3 = x^3 - 3x.
\]

The Hermite polynomials are orthogonal with respect to a certain inner product\(^5\):

\[
\langle p, q \rangle = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p(x)q(x)e^{-x^2/2} \, dx.
\] (4)

Hermite polynomials are useful in probability because if \( X \) is a standard normal random variable, then they are uncorrelated:

\[
E[H_j(X)H_k(X)] = \langle H_j, H_k \rangle = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} H_j(x)H_k(x)e^{-x^2/2} \, dx = 0 \quad \text{if } j \neq k.
\]

Still another basis of \( P_3 \) consists of Lagrange interpolating polynomials for the points 1, 2, 3, and 4:

\[
\begin{align*}
l_1 &= \frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)}, & l_2 &= \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)}, \\
l_3 &= \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)}, & l_4 &= \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}.
\end{align*}
\]

These are useful for interpolation because, for example, \( l_1(1) = 1 \) while \( l_2(1) = l_3(1) = l_4(1) = 0 \). If we want \( u(x) \) to be a polynomial of degree 3 taking specified values \( u(1) = u_1, \ u(2) = u_2, \ u(3) = u_3, \text{ and } u(4) = u_4 \), the answer is

\[
u(x) = u_1 l_1(x) + u_2 l_2(x) + u_3 l_3(x) + u_4 l_4(x).
\]

The Lagrange interpolating polynomials are linearly independent because if 0 = \( u(x) = u_1 l_1(x) + u_2 l_2(x) + u_3 l_3(x) + u_4 l_4(x) \) for all \( x \) then in particular \( u(x) = 0 \) at \( x = 1, 2, 3, \text{ and } 4 \), so \( u_1 = u_2 = u_3 = u_4 = 0 \). Let \( V' \subset P_3 \) be the set of polynomials \( p \in P_3 \) that satisfy \( p(2) = 0 \) and \( p(3) \). This is a subspace of \( P_3 \). A basis for it consists of \( l_1 \) and \( l_4 \).

If \( V' \subset V \) is a subspace of dimension \( m \) of a vector space of dimension \( n \), then it is possible to find a basis of \( V \) consisting of vectors \( f_k \) so that the first \( m \)

\(^5\text{The reader can verify that the formula (4) defines an inner product on the vector space } P_3.\)
of the $f_k$ form a basis of $V'$. For example, if $V = P_3$ and $V'$ is the polynomials that vanish at $x = 2$ and $x = 3$, we can take
\[
    f_1 = l_1, \quad f_2 = l_3, \quad f_3 = l_2, \quad f_4 = l_3.
\]
In general, the dimension of a subspace $V'$ is the dimension of $V$ minus the number of linearly independent conditions that define $V'$. If there are any nontrivial constraints that define $V'$, then $V'$ is a proper subspace of $V$; that is, there is some $u \in V$ that is not in $V'$, $m < n$. One common task in computational linear algebra is finding a well-conditioned basis for a subspace.

2.2 Matrices and linear transformations

Suppose $V$ and $W$ are vector spaces. A function $L$ from $V$ to $W$ is linear if $L(v_1 + v_2) = L(v_1) + L(v_2)$ for any vectors $v_1, v_2 \in V$, and $L(av) = aL(v)$ for any scalar $a$ and vector $v \in V$. Linear functions are also called linear transformations. By convention, we write $Lv$ instead of $L(v)$, even though $L$ represents a function from $V$ to $W$. This makes algebra with linear transformations look just like matrix algebra, deliberately blurring the distinction between linear transformations and matrices. The simplest example is $V = \mathbb{R}^n$, $W = \mathbb{R}^m$, and $Lu = A \cdot u$ for some $m \times n$ matrix $A$. The notation $A \cdot u$ refers to the product of the matrix $A$ and the vector $u$. Most of the time we just write $Au$.

Any linear transformation between finite dimensional vector spaces may be represented by a matrix. Suppose $f_1, \ldots, f_n$ is a basis for $V$, and $g_1, \ldots, g_m$ is a basis for $W$. For each $k$, the linear transformation of $f_k$ is an element of $W$ and may be written as a linear combination of the $g_j$:
\[
    Lf_k = \sum_{j=1}^{m} a_{jk}g_j.
\]
Because the transformation is linear, we can calculate what happens to a vector $u \in V$ in terms of its expansion $u = \sum_k u_k f_k$. Let $w \in W$ be the image of $u$, $w = Lu$, written as $w = \sum_j w_j g_j$. We find
\[
    w_j = \sum_{k=1}^{n} a_{jk}u_k,
\]
which is ordinary matrix-vector multiplication.

The matrix that represents $L$ depends on the basis. For example, suppose $V = P_3$, $W = P_2$, and $L$ represents differentiation:
\[
    L \left( p_0 + p_1 x + p_2 x^2 + p_3 x^3 \right) = \frac{d}{dx} \left( p_0 + p_1 x + p_2 x^2 + p_3 x^3 \right) = p_1 + 2p_2 x + 3p_3 x^2.
\]
If we take the basis $1, x, x^2$ for $V$, and $1, x, x^2$ for $W$, then the matrix is
\[
    \begin{pmatrix}
        0 & 1 & 0 & 0 \\
        0 & 0 & 2 & 0 \\
        0 & 0 & 0 & 3
    \end{pmatrix}.
\]
The matrix would be different if we used the Hermite polynomial basis for $V$ (see Exercise 1).

Conversely, an $m \times n$ matrix, $A$, represents a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$ (or from $\mathbb{C}^n$ to $\mathbb{C}^m$). We denote this transformation also by $A$. If $v \in \mathbb{R}^n$ is an $n$-component column vector, then the matrix-vector product $w = Av$ is a column vector with $m$ components. As before, the notation deliberately is ambiguous. The matrix $A$ is the matrix that represents the linear transformation $A$ using standard bases of $\mathbb{R}^n$ and $\mathbb{R}^m$.

A matrix also may represent a change of basis within the same space $V$. If $f_1, \ldots, f_n$, and $g_1, \ldots, g_n$ are different bases of $V$, and $u$ is a vector with expansions $u = \sum_k v_k f_k$ and $u = \sum_j w_j g_j$, then we may write

\[
\begin{pmatrix}
  v_1 \\
  \vdots \\
  v_n
\end{pmatrix} =
\begin{pmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \cdots & \cdots & \ddots & \cdots \\
  a_{n1} & a_{n2} & \cdots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
  w_1 \\
  \vdots \\
  w_n
\end{pmatrix}.
\]

As before, the matrix elements $a_{jk}$ are the expansion coefficients of $g_j$ with respect to the $f_k$ basis. For example, suppose $u \in P_3$ is given in terms of Hermite polynomials or simple powers: $u = \sum_{j=0}^3 v_j H_j(x) = \sum_{k=0}^3 w_j x^j$, then

\[
\begin{pmatrix}
  v_0 \\
  v_1 \\
  v_2 \\
  v_3
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & -1 & 0 \\
  0 & 1 & 0 & -3 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  w_0 \\
  w_1 \\
  w_2 \\
  w_3
\end{pmatrix}.
\]

We may reverse the change of basis by using the inverse matrix:

\[
\begin{pmatrix}
  w_1 \\
  \vdots \\
  w_n
\end{pmatrix} =
\begin{pmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \cdots & \cdots & \ddots & \cdots \\
  a_{n1} & a_{n2} & \cdots & a_{nn}
\end{pmatrix}^{-1}
\begin{pmatrix}
  v_1 \\
  \vdots \\
  v_n
\end{pmatrix}.
\]

Two bases must have the same number of elements because only square matrices can be invertible.

Composition of linear transformations corresponds to matrix multiplication. If $L$ is a linear transformation from $V$ to $W$, and $M$ is a linear transformation from $W$ to a third vector space, $Z$, then $ML$ is the composite transformation that takes $V$ to $Z$. The composite of $L$ and $M$ is defined if the target (or range) of $L$, is the same as the source (or domain) of $M$, $W$ in this case. If $A$ is an $m \times n$ matrix and and $B$ is $p \times q$, then the target of $A$ is $W = \mathbb{R}^m$ and the source of $B$ is $\mathbb{R}^q$. Therefore, the composite $AB$ is defined if $n = p$. This is the condition for the matrix product $A \cdot B$ (usually written without the dot) to be defined. The result is a transformation from $V = \mathbb{R}^q$ to $Z = \mathbb{R}^m$, i.e., the $m \times q$ matrix $AB$.

\textsuperscript{6}We write $a_{jk}$ for the $(j, k)$ entry of $A$. 

7
For vector spaces $V$ and $W$, the set of linear transformations from $V$ to $W$ forms a vector space. We can add two linear transformations and multiply a linear transformation by a scalar. This applies in particular to $m \times n$ matrices, which represent linear transformations from $\mathbb{R}^n$ to $\mathbb{R}^m$. The entries of $A + B$ are $a_{jk} + b_{jk}$. An $n \times 1$ matrix has a single column and may be thought of as a column vector. The product $Au$ is the same whether we consider $u$ to be a column vector or an $n \times 1$ matrix. $A \times n$ matrix has a single row and may be thought of as a row vector. If $r$ is such a row vector, the product $rA$ makes sense, but $Ar$ does not. It is useful to distinguish between row and column vectors although both are determined by $n$ components. The product $ru$ is a $1 \times 1$ matrix ($(1 \times n) \cdot (n \times 1)$ gives $1 \times 1$), i.e., a scalar.

If the source and targets are the same, $V = W$, or $n = m = p = q$, then both composites $LM$ and $ML$ both are defined, but they probably are not equal. Similarly, if $A$ and $B$ are $n \times n$ matrices, then $AB$ and $BA$ are defined, but $AB \neq BA$ in general. That is, composition and matrix multiplication are noncommutative. If $A$, $B$, and $C$ are matrices so that the products $AB$ and $BC$ both are defined, then the products $A \cdot (BC)$ and $(AB) \cdot C$ also are defined. The associative property of matrix multiplication is the fact that these are equal: $A(BC) = (AB)C$. In practice, there may be good reasons for computing $BC$ then multiplying by $A$, rather than finding $AB$ and multiplying by $C$.

If $A$ is an $m \times n$ matrix with real entries $a_{jk}$, the transpose of $A$, written $A^T$, is the $n \times m$ matrix whose $(j, k)$ entry is $(a^T)_{jk} = a_{kj}$. If $A$ has complex entries, then $A^\star$, the adjoint of $A$, is the $n \times m$ matrix with entries $(a^\star)_{jk} = \overline{a}_{kj}$ ($\overline{a}$ is the complex conjugate of $a$). If $A$ is real, then $A^T = A^\star$. The transpose or adjoint of a column vector is a row vector with the same number of entries, and vice versa. A square matrix $(m = n)$ is symmetric if $A = A^T$, and self-adjoint (or Hermitian) if $A = A^\star$. In the case of real matrices, symmetry and self-adjointness are the same.

For a linear operator $L$ on an inner product space, the adjoint $L^\star$ is an operator that satisfies $\langle L^\star u, v \rangle = \langle u, Lv \rangle$ for all $u$ and $v$. The reader should check that the matrix adjoint described above satisfies the definition of an operator when the matrix $A$ is interpreted as an operator from $\mathbb{R}^m$ to $\mathbb{R}^n$ with the standard inner product.

### 2.3 Vector norms

The norm of a vector $u$, written $\|u\|$, is a single number that describes the magnitude of $u$. There are different ways to measure overall size and therefore different vector norms. We say $\|u\|$ (technically, a real number associated to the vector $u$), is a norm if it satisfies the axioms:

1. **Positive definiteness:** $\|u\| \geq 0$ with $\|u\| = 0$ only when $u = 0$;
2. **Homogeneity:** $\|au\| = |a|\|u\|$ for any scalar $a$;
3. **Triangle inequality:** $\|u + v\| \leq \|u\| + \|v\|$ for any vectors $u$ and $v$.  

8
There are several simple norms for $\mathbb{R}^n$ or $\mathbb{C}^n$ that are useful in scientific computing. One is the $l^1$ norm

$$\|u\|_1 = \|u\|_{l^1} = \sum_{k=1}^{n} |u_k| .$$

Another is the $l^\infty$ norm, also called the sup norm or the max norm$^7$:

$$\|u\|_\infty = \|u\|_{l^\infty} = \max_{k=1,\ldots,n} |u_k| .$$

Another is the $l^2$ norm, also called the Euclidean norm

$$\|u\|_2 = \|u\|_{l^2} = \left( \sum_{k=1}^{n} |u_k|^2 \right)^{1/2} = \langle u, u \rangle^{1/2} .$$

The $l^2$ norm is natural, for example, for vectors representing positions or velocities in three dimensional space. If the components of $u \in \mathbb{R}^n$ represent probabilities, the $l^1$ norm might be more appropriate. In some cases we may have a norm defined indirectly or with a definition that is hard to turn into a number. For example in the vector space $P_3$ of polynomials of degree 3, we can define a norm

$$\|p\| = \max_{a \leq x \leq b} |p(x)| . \quad (5)$$

There is no simple formula for $\|p\|$ in terms of the coefficients of $p$.

An appropriate choice of norms is not always obvious. For example, what norm should we use for the two-dimensional subspace of $P_3$ consisting of polynomials that vanish at $x = 2$ and $x = 3$? In other cases, we might be concerned with vectors whose components have very different magnitudes, perhaps because they are associated with measurements in different units. This might happen, for example, if the components of $u$ represent different factors (or variables) in a linear regression. The first factor, $u_1$, might be age of a person, the second, $u_2$, income, the third the number of children. In units of years and dollars, we might get

$$u = \begin{pmatrix} 45 \\ 50000 \\ 2 \end{pmatrix} . \quad (6)$$

However, most people would consider a five dollar difference in annual salary to be small, while a five-child difference in family size is significant. In situations like these we can define for example, a dimensionless version of the $l^1$ norm:

$$\|u\| = \sum_{k=1}^{n} \frac{1}{s_k} \cdot |u_k| ,$$

$^7$The name $l^\infty$ comes from a generalization of the $l^2$ norm below. The $l^p$ norm is $\|u\|_p = (\sum |u_k|^2)^{1/p}$. One can show that $\|u\|_p \rightarrow \|u\|_\infty$ as $p \rightarrow \infty$. 

9
where the scale factor $s_k^{-1}$ is a typical value of a quantity with the units of $u_k$ in the problem at hand. In the example above, we might use $s_1 = 40$ (years), $s_2 = 60000$ (dollars per year), and $s_3 = 2.3$ (children). This is equivalent to using the $l^1$ norm for the problem expressed in a different basis, $\{s_k^{-1}e_k\}$. In many computations, it makes sense to change to an appropriately scaled basis before turning to the computer.

2.4 Norms of matrices and linear transformations

Suppose $L$ is a linear transformation from $V$ to $W$. If we have norms for the spaces $V$ and $W$, we can define a corresponding norm of $L$, written $\|L\|$, as the largest amount by which it stretches a vector:

$$\|L\| = \max_{u \neq 0} \frac{\|Lu\|}{\|u\|}.$$  \hfill (7)

The norm definition (7) implies that for all $u$,

$$\|Lu\| \leq \|L\| \cdot \|u\|.$$  \hfill (8)

Moreover, $\|L\|$ is the sharp constant in the inequality (8) in the sense that if $\|Lu\| \leq C \cdot \|u\|$ for all $u$, then $C \geq \|L\|$. Thus, (7) is equivalent to saying that $\|L\|$ is the sharp constant in (8).

The different vector norms give rise to different matrix norms. The matrix norms corresponding to certain standard vector norms are written with corresponding subscripts, such as

$$\|L\|_{l^2} = \max_{u \neq 0} \frac{\|Lu\|_{l^2}}{\|u\|_{l^2}}.$$  \hfill (9)

For $V = W = \mathbb{R}^n$, it turns out that (for the linear transformation represented in the standard basis by $A$)

$$\|A\|_{l^1} = \max_k \sum_j |a_{jk}|,$$

and

$$\|A\|_{l^\infty} = \max_j \sum_k |a_{jk}|.$$

Thus, the $l^1$ matrix norm is the maximum column sum while the max norm is the maximum row sum. Other norms, such as the $l^2$ matrix norm, are hard to compute explicitly in terms of the entries of $A$.

Any norm defined by (7) in terms of vector norms has several properties derived from corresponding properties of vector norms. One is homogeneity: $\|xL\| = |x| \|L\|$. Another is that $\|L\| \geq 0$ for all $L$, with $\|L\| = 0$ only for $L = 0$. The triangle inequality for vector norms implies that if $L$ and $M$ are two linear transformations from $V$ to $W$, then $\|L + M\| \leq \|L\| + \|M\|$. Finally, we have
\[ \|LM\| \leq \|L\| \|M\| \]. This is because the composite transformation stretches no more than the product of the individual maximum stretches:

\[ \|M(Lu)\| \leq \|M\| \|Lu\| \leq \|M\| \|L\| \|u\| \].

Of course, all these properties hold for matrices of the appropriate sizes.

All of these norms have uses in the theoretical parts of scientific computing, the \(l^1\) and \(l^\infty\) norms because they are easy to compute and the \(l^2\) norm because it is invariant under orthogonal transformations such as the discrete Fourier transform. The norms are not terribly different from each other. For example, \(\|A\|_{l^1} \leq n \|A\|_{l^\infty}\) and \(\|A\|_{l^\infty} \leq n \|A\|_{l^1}\). For \(n \leq 1000\), this factor of \(n\) may not be so important if we are asking, for example, about catastrophic ill-conditioning.

### 2.5 Eigenvalues and eigenvectors

Let \(A\) be an \(n \times n\) matrix, or a linear transformation from \(V\) to \(V\). If

\[ Ar = \lambda r \]

and \(r \neq 0\), we say that \(\lambda\) is an eigenvalue, and that \(r\) is the corresponding eigenvector\(^8\) of \(A\).

Eigenvalues and eigenvectors are useful in understanding dynamics related to \(A\). For example, the differential equation \(\frac{du}{dt} = \dot{u} = Au\) has solutions \(u(t) = e^{\lambda t}r\). If the differential equation describes something oscillating, \(A\) will have at least one complex eigenvalue. In general, eigenvalues and eigenvectors may be complex even though \(A\) is real. This is one reason people work with complex vectors in \(\mathbb{C}^n\), even for applications that seem to call for \(\mathbb{R}^n\).

The special case of the symmetric eigenvalue problem (\(A\) symmetric for real \(A\) or self-adjoint for complex \(A\)), is vastly different from the general, or unsymmetric problem. One of the differences is conditioning. The set of eigenvalues of a self-adjoint matrix is always a well-conditioned function of the matrix – a rare example of uniform good fortune. By contrast, the eigenvalues of an unsymmetric matrix may be so ill-conditioned, even for \(n\) as small as 20, that they are not computable in double precision arithmetic. Eigenvalues of unsymmetric matrices are too useful to ignore, but we can get into trouble if we forget their potential ill-conditioning. Eigenvectors, even for self-adjoint matrices, may be ill-conditioned.

We return to the unsymmetric eigenvalue problem. An \(n \times n\) matrix may have as many as \(n\) eigenvalues, denoted \(\lambda_k\), \(k = 1, \ldots, n\). If all the eigenvalues are distinct, the corresponding eigenvectors, denoted \(r_k\), with \(Ar_k = \lambda_k r_k\) must be linearly independent, and therefore form a basis for \(\mathbb{C}^n\). These \(n\) linearly independent vectors can be assembled to form the columns of an \(n \times n\) eigenvector

\(^8\) Note that “the” eigenvector is at best unique up to scaling: if \(r\) is an eigenvector, then so is \(ar\) for any scalar \(a\). Those who fail to understand this fact often complain needlessly when a computed eigenvector is scaled differently from the one they had in mind.
matrix that we call the *right eigenvector* matrix.

\[ R = \begin{pmatrix} \vdots & \cdots & \vdots \\ r_1 & \cdots & r_n \\ \vdots & \cdots & \vdots \end{pmatrix}. \] (10)

We also consider the diagonal eigenvalue matrix with the eigenvalues on the diagonal and zeros in all other entries:

\[ \Lambda = \begin{pmatrix} \lambda_1 & 0 \\ \vdots & \ddots \\ 0 & \cdots & \lambda_n \end{pmatrix}. \]

The eigenvalue – eigenvector relations may be expressed in terms of these matrices as

\[ AR = R\Lambda. \] (11)

To see this, check that multiplying \( R \) by \( A \) is the same as multiplying each of the columns of \( R \) by \( A \). Since these are eigenvectors, we get

\[
\begin{align*}
A \begin{pmatrix} \vdots & \cdots & \vdots \\ r_1 & \cdots & r_n \\ \vdots & \cdots & \vdots \end{pmatrix} &= \begin{pmatrix} \vdots & \cdots & \vdots \\ \lambda_1 r_1 & \cdots & \lambda_n r_n \\ \vdots & \cdots & \vdots \end{pmatrix} \\
&= \begin{pmatrix} \vdots & \cdots & \vdots \\ r_1 & \cdots & r_n \\ \vdots & \cdots & \vdots \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ \vdots & \ddots \\ 0 & \cdots & \lambda_n \end{pmatrix} = RA.
\end{align*}
\]

Since the columns of \( R \) are linearly independent, \( R \) is invertible, we can multiply (11) from the right and from the left by \( R^{-1} \) to get

\[ R^{-1}ARR^{-1} = R^{-1}R\Lambda R^{-1}, \]

then cancel the \( R^{-1}R \) and \( RR^{-1} \), and define \(^9 L = R^{-1} \) to get

\[ LA = \Lambda L. \]

This shows that the \( k^{th} \) row of \( L \) is an eigenvector of \( A \) if we put the \( A \) on the right:

\[ l_k A = \lambda_k l_k. \]

Of course, the \( \lambda_k \) are the same: there is no difference between “right” and “left” eigenvalues.

\(^9 \) Here \( L \) refers to a matrix, not a general linear transformation.
The matrix equation we used to define $L$, $LR = I$, gives useful relations between left and right eigenvectors. The $(j,k)$ entry of $LR$ is the product of row $j$ of $L$ with column $k$ of $R$. When $j = k$ this product should be a diagonal entry of $I$, namely one. When $j \neq k$, the product should be zero. That is

$$\begin{align*}
l_k r_k &= 1 \\
l_j r_k &= 0 \quad \text{if } j \neq k.
\end{align*}$$

These are called biorthogonality relations. For example, $r_1$ need not be orthogonal to $r_2$, but it is orthogonal to $l_2$. The set of vectors $r_k$ is not orthogonal, but the two sets $l_j$ and $r_k$ are biorthogonal. The left eigenvectors are sometimes called adjoint eigenvectors because their transposes form right eigenvectors for the adjoint of $A$:

$$A^* l_j^* = \lambda_j l_j^*.$$ 

Still supposing $n$ distinct eigenvalues, we may take the right eigenvectors to be a basis for $\mathbb{R}^n$ (or $\mathbb{C}^n$ if the entries are not real). As discussed in Section 2.2, we may express the action of $A$ in this basis. Since $Ar_j = \lambda_j r_j$, the matrix representing the linear transformation $A$ in this basis will be the diagonal matrix $\Lambda$. In the framework of Section 2.2, this is because if we expand a vector $v \in \mathbb{R}^n$ in the $r_k$ basis, $v = v_1 r_1 + \cdots + v_n r_n$, then $Av = \lambda_1 v_1 r_1 + \cdots + \lambda_n v_n r_n$. For this reason finding a complete set of eigenvectors and eigenvalues is called diagonalizing $A$. A matrix with $n$ linearly independent right eigenvectors is diagonalizable.

If $A$ does not have $n$ distinct eigenvalues then there may be no basis in which the action of $A$ is diagonal. For example, consider the matrix

$$J = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. $$

Clearly, $J \neq 0$ but $J^2 = 0$. A diagonal or diagonalizable matrix cannot have this property: if $A^2 = 0$ then $\Lambda = 0$, and if the relations $A \neq 0$, $A^2 = 0$ in one basis, they hold in any other basis. In general a Jordan block with eigenvalue $\lambda$ is a $k \times k$ matrix with $\lambda$ on the diagonal, 1 on the superdiagonal and zeros elsewhere:

$$\begin{pmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & 0 & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \lambda & 1 \\ 0 & 0 & \cdots & 0 & \lambda \end{pmatrix}. $$

If a matrix has fewer than $n$ distinct eigenvalues, it might or might not be diagonalizable. If $A$ is not diagonalizable, a theorem of linear algebra states that there is a basis in which $A$ has Jordan form. A matrix has Jordan form if it is block diagonal with Jordan blocks of various sizes and eigenvalues on the diagonal. It can be hard to compute the Jordan form of a matrix numerically, as we will see.
If $A$ has a Jordan block, a basis of eigenvectors will not exist; and even if $A$ is diagonalizable, transforming to an eigenvector basis may be very sensitive. For this reason, most software for the unsymmetric eigenvalue problem actually computes a Schur form:

$$AQ = QT,$$

where $T$ is an upper triangular matrix with the eigenvalues on the diagonal,

$$T = \begin{pmatrix}
\lambda_1 & t_{12} & t_{13} & \cdots & t_{1n} \\
0 & \lambda_2 & t_{23} & \cdots & t_{2n} \\
0 & 0 & \lambda_3 & \cdots & t_{3n} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & \lambda_n
\end{pmatrix},$$

and the columns $q_k$ of $Q$ are orthonormal, i.e.

$$q_j^\ast q_k = \begin{cases} 1, & j = k \\ 0, & j \neq k. \end{cases}$$

Equivalently, we can say that $Q$ is an orthogonal matrix, which means that $Q^\ast Q = I$. In many applications, the Schur form is an acceptable substitute for an eigenvalue–eigenvector decomposition. When eigenvectors are needed, they are computed from the Schur form.

The eigenvalue–eigenvector problem for self-adjoint matrices is different and in many ways simpler than the general nonsymmetric eigenvalue problem. The eigenvalues are real. The left eigenvectors are adjoints of the right eigenvectors: $l_k = r_k^\ast$. There are no Jordan blocks. Every self-adjoint matrix is diagonalizable even if the number of distinct eigenvalues is less than $n$. A complete set of eigenvectors of a symmetric matrix forms an orthonormal basis; that is, $R$ is orthogonal. The matrix form of the eigenvalue relation (11) may be written $R^\ast AR = \Lambda$, or $A = R\Lambda R^\ast$, or $R^\ast A = \Lambda R^\ast$. The latter shows (yet again) that the rows of $R^\ast$, which are $r_k^\ast$, are left eigenvectors of $A$.

2.6 Differentiation and perturbation theory

The main technique in the perturbation theory of Section 3 is implicit differentiation. We use the formalism of infinitesimal virtual perturbations, which is related to tangent vectors in differential geometry. It may seem roundabout at first, but it makes actual calculations quick.

Suppose $f(x)$ represents $m$ functions, $f_1(x), \ldots, f_m(x)$ of $n$ variables, $x_1, \ldots, x_n$. The Jacobian matrix$^{10}$, $f'(x)$, is the $m \times n$ matrix of partial derivatives $f_j'(x) = \partial_{x_k} f_j(x)$. If $f$ is differentiable (and $f'$ is Lipschitz continuous), then the first derivative approximation is (writing $x_0$ for $x$ to clarify some discussion below)

$$f(x_0 + \Delta x) - f(x_0) = \Delta f = f'(x_0)\Delta x + O(\|\Delta x\|^2).$$

\footnote{See any good book on multivariate calculus.}

14
Here $\Delta f$ and $\Delta x$ are column vectors.

Suppose $s$ is a scalar “parameter” and $x(s)$ is a differentiable curve in $\mathbb{R}^n$ with $x(0) = x_0$. The function $f(x)$ then defines a curve in $\mathbb{R}^m$ with $f(x(0)) = f(x_0)$. We define two vectors, called virtual perturbations,

$$\dot{x} = \frac{d}{ds} \bigg|_{s=0} x(s), \quad \dot{f} = \frac{d}{dx} \bigg|_{s=0} f(x(s)).$$

The multivariate calculus chain rule implies the virtual perturbation formula

$$\dot{f} = f'(x_0) \dot{x}. \quad (14)$$

This formula is nearly the same as (13). The virtual perturbation strategy is to calculate the linear relationship (14) between virtual perturbations and use it to find the approximate relation (13) between actual perturbations. For this, it is important that any $\dot{x} \in \mathbb{R}^n$ can be the virtual perturbation corresponding to some curve: just take the straight “curve” $x(s) = x_0 + sx$.

The Leibniz rule (product rule) for matrix multiplication makes virtual perturbations handy in linear algebra. Suppose $A(s)$ and $B(s)$ are differentiable curves of matrices and compatible for matrix multiplication. Then the virtual perturbation of $AB$ is given by the product rule

$$\frac{d}{ds} \bigg|_{s=0} AB = \dot{A}B + A\dot{B}. \quad (15)$$

To see this, note that the $(jk)$ entry of $A(s)B(s)$ is $\sum_l a_{jl}(s)b_{lk}(s)$. Differentiating this using the ordinary product rule then setting $s = 0$ yields

$$\sum_l \dot{a}_{jl}b_{lk} + \sum_l a_{jl}\dot{b}_{lk}.$$  

These terms correspond to the terms on the right side of (15). We can differentiate products of more than two matrices in this way.

As an example, consider perturbations of the inverse of a matrix, $B = A^{-1}$. The variable $x$ in (13) now is the matrix $A$, and $f(A) = A^{-1}$. Apply implicit differentiation to the formula $AB = I$, use the fact that $I$ is constant, and we get $\dot{A}B + A\dot{B} = 0$. Then solve for $\dot{B}$ and use $A^{-1} = B$, and get

$$\dot{B} = -A^{-1}\dot{A}A^{-1}.$$  

The corresponding actual perturbation formula is

$$\Delta (A^{-1}) = -A^{-1} (\Delta A) A^{-1} + O (\|\Delta A\|^2) \quad . \quad (16)$$

This is a generalization of the fact that the derivative of $1/x$ is $-1/x^2$, so $\Delta (1/x) \approx (-1/x^2)\Delta x$. When $x$ is replaced by $A$ and $\Delta A$ does not commute with $A$, we have to worry about the order of the factors. The correct order is (16), and not, for example, $\dot{A}A^{-1}A^{-1} = A\dot{A}^{-2}$.
For future reference we comment on the case \( m = 1 \), which is the case of one function of \( n \) variables. The \( 1 \times n \) Jacobian matrix may be thought of as a row vector. We often write this as \( \nabla f \), and calculate it from the fact that \( \dot{f} = \nabla f(x) \cdot \dot{x} \) for all \( \dot{x} \). In particular, \( x \) is a stationary point of \( f \) if \( \nabla f(x) = 0 \), which is the same as \( \dot{f} = 0 \) for all \( \dot{x} \). For example, suppose \( f(x) = x^*Ax \) for some \( n \times n \) matrix \( A \). This is a product of the \( 1 \times n \) matrix \( x^* \) with \( A \) with the \( n \times 1 \) matrix \( x \). The Leibniz rule \((15)\) gives, if \( A \) is constant,

\[
\dot{f} = \dot{x}^*Ax + x^*A\dot{x}.
\]

Since the \( 1 \times 1 \) real matrix \( \dot{x}^*Ax \) is equal to its transpose, this is

\[
\dot{f} = x^*(A + A^*)\dot{x}.
\]

This implies that (both sides are row vectors)

\[
\nabla (x^*Ax) = x^*(A + A^*).
\]

(17)

If \( A^* = A \), we recognize this as a generalization of the \( n = 1 \) formula \( \frac{d}{dx}(ax^2) = 2ax \).

### 2.7 Variational principles for the symmetric eigenvalue problem

A variational principle is a way to find something by solving a maximization or minimization problem. The Rayleigh quotient for an \( n \times n \) matrix is

\[
Q(x) = \frac{x^*Ax}{x^*x} = \frac{\langle x, Ax \rangle}{\langle x, x \rangle}.
\]

(18)

If \( x \) is complex, \( x^* \) is the adjoint. In either case, the denominator is \( x^*x = \sum_{k=1}^{n} |x_k|^2 = \|x\|^2 \). The Rayleigh quotient is defined for \( x \neq 0 \). A vector \( r \) is a stationary point if \( \nabla Q(r) = 0 \). If \( r \) is a stationary point, the corresponding value \( \lambda = Q(r) \) is a stationary value.

**Theorem 1** Suppose \( A \) is self-adjoint. A vector \( x \neq 0 \) is an eigenvector if and only if it is a stationary point of the Rayleigh quotient; and if \( x \) is an eigenvector, the Rayleigh quotient is the corresponding eigenvalue.

**Proof:** The underlying principle is the calculation \((17)\). If \( A^* = A \) (this is where symmetry matters) then \( \nabla (x^*Ax) = 2x^*A \). Since \((18)\) is a quotient, we differentiate using the quotient rule. We already know \( \nabla(x^*Ax) = 2x^*A \). Also, \( \nabla x^*x = 2x^* \) (the rule with \( A = I \)). The result is

\[
\nabla Q = 2 \left( \frac{1}{x^*x} \right) x^*A - 2 \left( \frac{x^*Ax}{(x^*x)^2} \right) x^* = \frac{2}{x^*x} (x^*A - x^*Q(x)) x^*.
\]

If \( x \) is a stationary point (\( \nabla Q = 0 \)), then \( x^*A = \left( \frac{x^*Ax}{x^*x} \right) x^* \), or, taking the adjoint,

\[
Ax = \left( \frac{x^*Ax}{x^*x} \right) x.
\]
This shows that $x$ is an eigenvector with
\[ \lambda = \frac{x^*Ax}{x^*x} = Q(x) \]
as the corresponding eigenvalue. Conversely if $Ar = \lambda r$, then $Q(r) = \lambda$ and the calculations above show that $\nabla Q(r) = 0$. This proves the theorem. \[ \square \]

A simple observation shows that there is at least one stationary point of $Q$ for Theorem 1 to apply to. If $\alpha$ is a real number, then $Q(\alpha x) = Q(x)$. We may choose $\alpha$ so that
\[ \|\alpha x\| = 1. \]
This shows that
\[ \max_{x \neq 0} Q(x) = \max_{\|x\|=1} Q(x) = \max_{\|x\|=1} x^*Ax. \]

A theorem of analysis states that if $Q(x)$ is a continuous function on a compact set, then there is an $r$ so that $Q(r) = \max_x Q(x)$ (the max is attained). The set of $x$ with $\|x\| = 1$ (the unit sphere) is compact and $Q$ is continuous. Clearly if $Q(r) = \max_x Q(x)$, then $\nabla Q(r) = 0$, so $r$ is a stationary point of $Q$ and an eigenvector of $A$.

Now suppose we have found $m$ orthogonal eigenvectors $r_1, \ldots, r_m$. If $x$ is orthogonal to $r_j$, i.e. $r_j^*x = 0$, then so is $Ax$:
\[ r_j^*(Ax) = (r_j^*A)x = \lambda_j r_j^*x = 0. \]

Therefore, the subspace $V_m \subset C^n$ of all $x$ that are orthogonal to $r_1, \ldots, r_m$ is an invariant subspace: if $x \in V_m$, then $Ax \in V_m$. Thus $A$ defines a linear transformation from $V_m$ to $V_m$, which we call $A_m$. Chapter ?? gives a proof that $A_m$ is symmetric in a suitable basis. Therefore, Theorem 1 implies that $A_m$ has at least one eigenvector, $r_{m+1}$, with eigenvalue $\lambda_{m+1}$. Since $r_{m+1} \in V_m$, the action of $A$ and $A_m$ on $r_{m+1}$ is the same, which means that $Ar_{m+1} = \lambda_{m+1}r_{m+1}$. After finding $r_{m+1}$, we can repeat the procedure to find $r_{m+2}$, and continue until we eventually find a full set of $n$ orthogonal eigenvectors.

### 2.8 Least squares

Suppose $A$ is an $m \times n$ matrix representing a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$, and we have a vector $b \in \mathbb{R}^m$. If $n < m$ the linear equation system $Ax = b$ is \textit{overdetermined} in the sense that there are more equations than variables to satisfy them. If there is no $x$ with $Ax = b$, we may seek an $x$ that minimizes the \textit{residual}
\[ r = Ax - b. \]

This \textit{linear least squares} problem
\[ \min_x \|Ax - b\|_2, \]
is the same as finding \( x \) to minimize the sum of squares
\[
\|r\|^2 = SS = \sum_{j=1}^{n} r_j^2.
\]

Linear least squares problems arise through linear regression in statistics. A linear regression model models the response, \( b \), as a linear combination of \( m \) explanatory vectors, \( a_k \), each weighted by a regression coefficient, \( x_k \). The residual, \( R = (\sum_{k=1}^{m} a_k x_k) - b \), is modeled as a Gaussian random variable\(^{12} \) with mean zero and variance \( \sigma^2 \). We do \( n \) experiments. The explanatory variables and response for experiment \( j \) are \( a_{jk} \), for \( k = 1, \ldots, m \), and \( b_j \), and the residual (for given regression coefficients) is \( r_j = \sum_{k=1}^{m} a_{jk} x_k - b_j \). The log likelihood function is \( f(x) = -\sigma^2 \sum_{j=1}^{n} r_j^2 \). Finding regression coefficients to maximize the log likelihood function leads to (20).

The normal equations give one approach to least squares problems. We calculate:
\[
\|r\|^2 = r^* r = (Ax - b)^* (Ax - b) = x^* A^* A x - 2x^* A^* b + b^* b.
\]

Setting the gradient to zero as in the proof of Theorem 1 leads to the normal equations
\[
A^* A x = A^* b,
\]
which can be solved by
\[
x = (A^* A)^{-1} A^* b.
\]

The matrix \( M = A^* A \) is the moment matrix or the Gram matrix. It is symmetric, and positive definite if \( A \) has rank \( m \), so the Choleski decomposition of \( M \) (see Chapter ??) is a good way to solve (21). The matrix \((A^* A)^{-1} A^* \) is the pseudoinverse of \( A \). If \( A \) were square and invertible, it would be \( A^{-1} \) (check this). The normal equation approach is the fastest way to solve dense linear least squares problems, but it is not suitable for the subtle ill-conditioned problems that arise often in practice.

The singular value decomposition in Section 2.9 and the QR decomposition from Section ?? give better ways to solve ill-conditioned linear least squares problems.

### 2.9 Singular values and principal components

Let \( A \) be an \( m \times n \) matrix that represents a linear transformation from \( \mathbb{R}^n \) to \( \mathbb{R}^m \). The right singular vectors, \( v_k \in \mathbb{R}^n \) form an orthonormal basis for \( \mathbb{R}^n \). The left

\(^{12} \)See any good book on statistics for definitions of Gaussian random variable and the log likelihood function. What is important here is that a systematic statistical procedure, the maximum likelihood method, tells us to minimize the sum of squares of residuals.
singular vectors, \( u_k \in \mathbb{R}^m \), form an orthonormal basis for \( \mathbb{R}^m \). Corresponding to each \( v_k \) and \( u_k \) pair is a non-negative singular value, \( \sigma_k \) with

\[
Av_k = \sigma_k u_k .
\]  

(23)

By convention these are ordered so that \( \sigma_1 \geq \sigma_2 \geq \cdots \geq 0 \). If \( n < m \) we interpret (23) as saying that \( \sigma_k = 0 \) for \( k > n \). If \( n > m \) we say \( \sigma_k = 0 \) for \( k > m \).

The non-zero singular values and corresponding singular vectors may be found one by one using variational and orthogonality principles similar to those in Section 2.7. We suppose \( A \) is not the zero matrix (not all entries equal to zero). The first step is the variational principle:

\[
\sigma_1 = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} .
\]  

(24)

As in Section 2.7, the maximum is achieved, and \( \sigma_1 > 0 \). Let \( v_1 \in \mathbb{R}^n \) be a maximizer, normalized to have \( \|v_1\| = 1 \). Because \( \|Av_1\| = \sigma_1 \), we may write \( Av_1 = \sigma_1 u_1 \) with \( \|u_1\| = 1 \). This is the relation (23) with \( k = 1 \).

The optimality condition calculated as in the proof of Theorem 1 implies that

\[
u_1^* A = \sigma_1 v_1^* .
\]  

(25)

Indeed, since \( \sigma_1 > 0 \), (24) is equivalent to

\[
\sigma_1^2 = \max_{x \neq 0} \frac{\|Ax\|^2}{\|x\|^2} = \max_{x \neq 0} \frac{(Ax)^*(Ax)}{x^*x} = \max_{x \neq 0} \frac{x^*(A^*A)x}{x^*x} .
\]  

(26)

Theorem 1 implies that the solution to the maximization problem (26), which is \( v_1 \), satisfies \( \sigma_1^2 v_1 = A^*Av_1 \). Since \( Av_1 = \sigma_1 u_1 \), this implies \( \sigma_1 v_1 = A^*u_1 \), which is the same as (25).

The analogue of the eigenvalue orthogonality principle is that if \( x^*v_1 = 0 \), then \((Ax)^* u_1 = 0 \). This is true because

\[
(Ax)^* u_1 = x^* (A^*u_1) = x^*\sigma_1 v_1 = 0 .
\]

Therefore, if we define \( V_1 \subset \mathbb{R}^n \) by \( x \in V_1 \) if \( x^*v_1 = 0 \), and \( U_1 \subset \mathbb{R}^m \) by \( y \in U_1 \) if \( y^*u_1 = 0 \), then \( A \) also defines a linear transformation (called \( A_1 \)) from \( V_1 \) to \( U_1 \). If \( A_1 \) is not identically zero, we can define

\[
\sigma_2 = \max_{x \neq 0} \frac{\|Ax\|^2}{\|x\|^2} = \max_{x \neq 0} \frac{\|Ax\|^2}{\|x\|^2} ,
\]

These calculations make constant use of the associativity of matrix multiplication, even when one of the matrices is a row or column vector.
and get $A v_2 = \sigma_2 u_2$ with $v_2^* v_1 = 0$ and $u_2^* u_1 = 0$. This is the second step in constructing orthonormal bases satisfying (23). Continuing in this way, we can continue finding orthonormal vectors $v_k$ and $u_k$ that satisfy (23) until we reach $A_k = 0$ or $k = m$ or $k = n$. After that point, we may complete the $v$ and $u$ bases arbitrarily as in Chapter ?? with remaining singular values being zero.

The singular value decomposition (SVD) is a matrix expression of the relations (23). Let $U$ be the $m \times m$ matrix whose columns are the left singular vectors $u_k$ (as in (10)). The orthonormality relations$^{14}$ $u_j^* u_k = \delta_{jk}$ are equivalent to $U$ being an orthogonal matrix: $U^* U = I$. Similarly, we can form the orthogonal $n \times n$ matrix, $V$, whose columns are the right singular vectors $v_k$.

Finally, the $m \times n$ matrix, $\Sigma$, has the singular values on its diagonal (with somewhat unfortunate notation), $\sigma_{jj} = \sigma_j$, and zeros off the diagonal, $\sigma_{jk} = 0$ if $j \neq k$. With these definitions, the relations (23) are equivalent to $AV = U\Sigma$, which more often is written

$$A = U\Sigma V^*.$$  

(27)

This the singular value decomposition. Any matrix may be factored, or decomposed, into the product of the form (27) where $U$ is an $m \times m$ orthogonal matrix, $\Sigma$ is an $m \times n$ diagonal matrix with nonnegative entries, and $V$ is an $n \times n$ orthogonal matrix.

A calculation shows that $A^* A = V \Sigma^* \Sigma V^* = V A V^*$. This implies that the eigenvalues of $A^* A$ are given by $\lambda_j = \sigma_j^2$ and the right singular vectors of $A$ are the eigenvectors of $A^* A$. Section 3 explains that the condition number associated with solving linear systems may be taken to be $\kappa_{\ell^2}(A) = \sigma_1(A)/\sigma_n(A)$. This implies that $\kappa_{\ell^2}(A^* A) = \kappa_{\ell^2}(A)^2$. This means that the condition number of solving the normal equations (21) is the square of the condition number of the original least squares problem (20). If the condition number of a least squares problem is $\kappa_{\ell^2}(A) = 10^5$, even the best solution algorithm can amplify errors by a factor of $10^5$. Solving using the normal equations can amplify rounding errors by a factor of $10^{10}$.

Many people call singular vectors $u_k$ and $v_k$ principal components. They refer to the singular value decomposition as principal component analysis, or PCA. One application is clustering, in which you have $n$ objects, each with $m$ measurements, and you want to separate them into two clusters, say “girls” and “boys”. You assemble the data into a matrix, $A$, and compute, say, the largest two singular values and corresponding left singular vectors, $u_1 \in \mathbb{R}^m$ and $u_2 \in \mathbb{R}^m$. The data for object $k$ is $a_k \in \mathbb{R}^m$, and you compute $z_k \in \mathbb{R}^2$ by $z_{k1} = u_1^* a_k$ and $z_{k2} = u_2^* a_k$, the components of $a_k$ in the principal component directions. You then plot the $n$ points $z_k$ in the plane and look for clusters, or maybe just a line that separates one group of points from another. Surprising as may seem, this simple procedure does identify clusters in practical problems.

$^{14}$ Here $\delta_{jk}$ is the Kronecker delta, which is equal to one when $j = k$ and zero otherwise.
3 Condition number

Ill-conditioning can be a serious problem in numerical solution of problems in linear algebra. We take into account possible ill-conditioning when we choose computational strategies. For example, the matrix exponential $\exp(A)$ (see Exercise 12) can be computed using the eigenvectors and eigenvalues of $A$. We will see in Section 3.3 that the eigenvalue problem may be ill conditioned even when the problem of computing $\exp(A)$ is fine. In such cases we need a way to compute $\exp(A)$ that does not use the eigenvectors and eigenvalues of $A$.

As we said in Section ?? (in slightly different notation), the condition number is the ratio of the change in the answer to the change in the problem data, with (i) both changes measured in relative terms, and (ii) the change in the problem data being small. Norms provide a way to extend this definition to functions and data with more than one component. Let $f(x)$ represent $m$ functions of $n$ variables, with $\Delta x$ being a change in $x$ and $\Delta f = f(x + \Delta x) - f(x)$ the corresponding change in $f$. The size of $\Delta x$, relative to $x$, is $\|\Delta x\| / \|x\|$, and similarly for $\Delta f$. In the multivariate case, the size of $\Delta f$ depends not only on the size of $\Delta x$, but also on the direction. The norm-based condition number seeks the worst case $\Delta x$, which leads to

$$\kappa(x) = \lim_{\epsilon \to 0} \max_{\|\Delta x\| = \epsilon} \frac{\|f(x + \Delta x) - f(x)\| / \|f(x)\|}{\|\Delta x\| / \|x\|}. \quad (28)$$

Except for the maximization over directions $\Delta x$ with $\|\Delta x\| = \epsilon$, this is the same as the earlier definition ??.

Still following Section ??, we express (28) in terms of derivatives of $f$. We let $f'(x)$ represent the $m \times n$ Jacobian matrix of first partial derivatives of $f$, as in Section 2.6, so that, $\Delta f = f'(x)\Delta x + O\left(\|\Delta x\|^2\right)$. Since $O\left(\|\Delta x\|^2\right) / \|\Delta x\| = O\left(\|\Delta x\|\right)$, the ratio in (28) may be written

$$\frac{\|\Delta f\|}{\|\Delta x\|} \cdot \frac{\|x\|}{\|f\|} = \frac{\|f'(x)\Delta x\|}{\|\Delta x\|} \cdot \frac{\|x\|}{\|f\|} + O\left(\|\Delta x\|\right).$$

The second term on the right disappears as $\Delta x \to 0$. Maximizing the first term on the right over $\Delta x$ yields the norm of the matrix $f'(x)$. Altogether, we have

$$\kappa(x) = \|f'(x)\| \cdot \frac{\|x\|}{\|f(x)\|}. \quad (29)$$

This differs from the earlier condition number definition ?? in that it uses norms and maximizes over $\Delta x$ with $\|\Delta x\| = \epsilon$.

In specific calculations we often use a slightly more complicated way of stating the definition (28). Suppose that $P$ and $Q$ are two positive quantities and there is a $C$ so that $P \leq C \cdot Q$. We say that $C$ is the sharp constant if there is no $C'$ with $C' < C$ so that $P \leq C' \cdot Q$. For example, we have the inequality $\sin(2\epsilon) \leq 3\epsilon$ for all $\epsilon > 0$. But $C = 3$ is not the sharp constant because the inequality also is true with $C' = 2$, which is sharp.
This sharp constant idea is not exactly what we want because it is required to hold for all \( \epsilon \) (large or small), and because the inequality you might want for small \( \epsilon \) is not exactly true. For example, there is no inequality \( \tan(\epsilon) \leq C\epsilon \) that applies for all \( \epsilon > 0 \). As \( \epsilon \to 0 \), the constant seems to be \( C = 1 \), but this is not strictly true, since \( \tan(\epsilon) > \epsilon \) for \( 0 < \epsilon < \frac{\pi}{2} \). Therefore we write

\[
P(\epsilon) \lesssim CQ(\epsilon) \quad \text{as } \epsilon \to 0 ,
\]

if \( P(\epsilon) > 0, Q(\epsilon) > 0, \) and

\[
\lim_{\epsilon \to 0} \frac{P(\epsilon)}{Q(\epsilon)} \leq C .
\]

This is the sharp constant, the smallest \( C \) so that

\[
P(\epsilon) \leq C \cdot Q(\epsilon) + O(\epsilon) \quad \text{as } \epsilon \to 0.
\]

The definition (28) is precisely that \( \kappa(x) \) is the sharp constant in the inequality

\[
\|\Delta f\| \lesssim \|\Delta x\| \quad \text{as } \|x\| \to 0.
\]

### 3.1 Linear systems, direct estimates

We start with the condition number of calculating \( b = Au \) in terms of \( u \) with \( A \) fixed. This fits into the general framework above, with \( u \) playing the role of \( x \), and \( Au \) of \( f(x) \). Of course, \( A \) is the Jacobian of the function \( u \to Au \), so (29) gives

\[
\kappa(A, u) = \|A\| \cdot \|u\|/\|Au\| .
\]

This shows that computing \( Au \) is ill-conditioned if \( \|A\| \) is much larger than the ratio \( \|Au\|/\|u\| \). The norm of \( A \) is the maximum \( A \) can stretch any vector \( \max \|Av\|/\|v\| \). Computing \( Au \) is ill-conditioned if it stretches some vector \( v \) much more than it stretches \( u \).

The condition number of solving a linear system \( Au = b \) (finding \( u \) as a function of \( b \)) is the same as the condition number of computing \( u = A^{-1}b \). The formula (32) gives this as

\[
\kappa(A^{-1}, b) = \|A^{-1}\| \cdot \|b\|/\|A^{-1}b\| = \|A^{-1}\| \cdot \|Au\|/\|u\| .
\]

This is large if there is some vector that is stretched much less than \( u \). Of course, “stretching” factors can be less than one. For future reference, note that \( \kappa(A^{-1}, b) \) is not the same as (32).

The traditional definition of the condition number of the \( Au \) computation takes the worst case relative error not only over perturbations \( \Delta u \) but also over vectors \( u \). Taking the maximum over \( \Delta u \) led to (32), so we need only maximize it over \( u \):

\[
\kappa(A) = \|A\| \cdot \max_{u \neq 0} \|u\|/\|Au\| .
\]

22
Since \( A(u + \Delta u) - Au = A\Delta u, \) and \( u \) and \( \Delta u \) are independent variables, this is the same as
\[
\kappa(A) = \max_{u \neq 0} \frac{\|u\|}{\|Au\|} \cdot \max_{\Delta u \neq 0} \frac{\|A\Delta u\|}{\|\Delta u\|}.
\]
(34)

To evaluate the maximum, we suppose \( A^{-1} \) exists. Substituting \( Au = v, \) \( u = A^{-1}v, \) gives
\[
\max_{u \neq 0} \frac{\|u\|}{\|Au\|} = \max_{v \neq 0} \frac{\|A^{-1}v\|}{\|v\|} = \|A^{-1}\|.
\]
Thus, (33) leads to
\[
\kappa(A) = \|A\| \|A^{-1}\|
\]
as the worst case condition number of the forward problem.

The computation \( b = Au \) with
\[
A = \begin{pmatrix} 1000 & 0 \\ 0 & 10 \end{pmatrix}
\]
illustrates this discussion. The error amplification relates \( \|\Delta b\| / \|b\| \) to \( \|\Delta u\| / \|u\| \).

The worst case would make \( \|b\| \) small relative to \( \|u\| \) and \( \|\Delta b\| \) large relative to \( \|\Delta u\| \): amplify \( u \) the least and \( \Delta u \) the most. This is achieved by taking
\[
u = \begin{pmatrix} 0 \\ 1 \end{pmatrix}
\]
such that \( Au = \begin{pmatrix} 0 \\ 10 \end{pmatrix} \) with amplification factor 10, and \( \Delta u = \begin{pmatrix} \epsilon \\ 0 \end{pmatrix} \)
with \( A\Delta u = \begin{pmatrix} 1000\epsilon \\ 0 \end{pmatrix} \) and amplification factor 1000. This makes \( \|\Delta b\| / \|b\| \) 100 times larger than \( \|\Delta u\| / \|u\| \). For the condition number of calculating \( u = A^{-1}b, \) the worst case is \( b = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \) and \( \Delta b = \begin{pmatrix} \epsilon \\ 0 \end{pmatrix} \), which amplifies the error by the same factor of \( \kappa(A) = 100. \)

The informal condition number (33) has advantages and disadvantages over the more formal one (32). At the time we design a computational strategy, it may be easier to estimate the informal condition number than the formal one, as we may know more about \( A \) than \( u. \) If we have no idea what \( u \) will come up, we have a reasonable chance of getting something like the worst one. Also, by coincidence, \( \kappa(A) \) defined by (33) determines the convergence rate of some iterative methods for solving linear systems involving \( A. \) On the other hand, in solving differential equations \( \kappa(A) \) often is much larger than \( \kappa(A, u). \) In such cases, the error in solving \( Au = b \) is much less than the condition number \( \kappa(A) \) would suggest. For example, in Exercise 11, \( \kappa(A) \) is on the order of \( n^2, \) where \( n \) is the number of unknowns. The truncation error for the second order discretization is on the order of \( 1/n^2. \) A naive estimate using (33) might suggest that solving the system amplifies the \( O(n^{-2}) \) truncation error by a factor of \( n^2 \) to be on the same order as the solution itself. This does not happen because the \( u \) we seek is smooth, and not like the worst case.

\textsuperscript{15}See Exercise 8 for a the \( l^2 \) condition number of the \( u \to Au \) problem with singular or non-square \( A. \)
The informal condition number \((35)\) also has the strange feature than \(\kappa(A) = \kappa(A^{-1})\), since \((A^{-1})^{-1} = A\). This gives the mistaken problem that solving a forward problem (computing \(Au\) from \(u\)) has the same stability properties as solving the inverse problem (computing \(u\) from \(b = Au\)). For example, solving the heat equation forward in time is far different from the inverse problem of solving it backward in time.\(^{16}\) Again, the more precise definition \((32)\) does not have this drawback.

### 3.2 Linear systems, perturbation theory

If \(Au = b\), we can study the dependence of \(u\) on \(A\) through perturbation theory. The starting point is the perturbation formula \((16)\). Taking norms gives

\[ \|\Delta u\| \lesssim \|A^{-1}\| \|A\| \|\Delta A\| \|u\|, \quad \text{(for small } \Delta A\text{),} \tag{36} \]

so

\[ \frac{\|\Delta u\|}{\|u\|} \lesssim \|A^{-1}\| \|A\| \|\Delta A\| \|A\|. \tag{37} \]

This shows that the condition number satisfies \(\kappa \leq \|A^{-1}\| \|A\|\). The condition number is equal to \(\|A^{-1}\| \|A\|\) if the inequality \((36)\) is (approximately for small \(\Delta A\)) sharp, which is because we can take \(\Delta A = \epsilon I\) and \(u\) to be a maximum stretch vector for \(A^{-1}\). Note that the condition number formula \((35)\) applies to the problem of solving the linear system \(Au = b\) both when we consider perturbations in \(b\) and in \(A\), though the derivations here are different.

### 3.3 Eigenvalues and eigenvectors

The eigenvalue relationship is \(Ar_j = \lambda_j r_j\). Perturbation theory allows to estimate the changes in \(\lambda_j\) and \(r_j\) that result from a small \(\Delta A\). These perturbation results are used throughout science and engineering. The symmetric or self-adjoint case is often is called Rayleigh-Schrödinger perturbation theory.\(^\text{17}\) Using the virtual perturbation method of Section 2.6, differentiating the eigenvalue relation using the product rule yields

\[ \dot{A}r_j + Ar_j = \dot{\lambda}_j r_j + \lambda_j \dot{r}_j. \tag{38} \]

Multiply this from the left by \(r_j^*\) and use the fact that \(r_j^*\) is a left eigenvector gives

\[ r_j^* \dot{A}j r_j = \dot{\lambda}_j r_j^* r_j. \]

If \(r_j\) is normalized so that \(r_j^* r_j = \|r_j\|^2 = 1\), then the right side is just \(\dot{\lambda}_j\). Trading virtual perturbations for actual small perturbations turns this into the famous formula

\[ \Delta \lambda_j = r_j^* \Delta A r_j + O(\|\Delta A\|) \text{.} \tag{39} \]

\(^{16}\)See, for example, the book by Fritz John on partial differential equations.

\(^{17}\)Lord Rayleigh used it to study vibrational frequencies of plates and shells. Later Erwin Schrödinger used it to compute energies (which are eigenvalues) in quantum mechanics.
We get a condition number estimate by recasting this in terms of relative errors on both sides. The important observation is that \( \| r_j \|_2 = 1 \), so 
\[
\| \Delta A \cdot r_j \|_2 \leq \| \Delta A \|_2 
\]
and finally
\[
\| r^* \Delta \lambda \|_2 \approx \| \Delta A \|_2.
\]
This inequality is sharp because we can take \( \Delta A = \epsilon r_j r_j^* \), which is an \( n \times n \) matrix with (see Exercise 7) \( \| \epsilon r_j r_j^* \|_2 = |\epsilon| \). Putting this into (39) gives the also sharp inequality,
\[
\left| \frac{\Delta \lambda_j}{\lambda_j} \right| \leq \frac{\| \lambda \|_2}{|\lambda_j|} \| \Delta A \|_2.
\]
We can put this directly into the abstract condition number definition (28) to get the conditioning of \( \lambda_j \):
\[
\kappa_j(A) = \frac{\| \lambda \|_2}{|\lambda_j|} = \frac{|\lambda|_{\text{max}}}{|\lambda_j|}.
\]
(40)
Here, \( \kappa_j(A) \) denotes the condition number of the problem of computing \( \lambda_j \), which is a function of the matrix \( A \), and \( \| \lambda \|_2 = |\lambda|_{\text{max}} \) refers to the eigenvalue of largest absolute value.

The condition number formula (40) predicts the sizes of errors we get in practice. Presumably \( \lambda_j \) depends in some way on all the entries of \( A \) and the perturbations due to roundoff will be on the order of the entries themselves, multiplied by the machine precision, \( \epsilon_{\text{mach}} \), which are on the order of \( \| A \|_2 \). Only if \( \lambda_j \) is very close to zero, by comparison with \( |\lambda|_{\text{max}} \), will it be hard to compute with high relative accuracy. All of the other eigenvalue and eigenvector problems have much worse condition number difficulties.

The eigenvalue problem for non-symmetric matrices can be much more sensitive. To derive the analogue of (39) for non-symmetric matrices we start with (38) and multiply from the left with the corresponding left eigenvector, \( l_j \), and using the normalization condition \( l_j r_j = 1 \). After simplifying, the result is
\[
\hat{\lambda}_j = l_j \hat{A} r_j, \quad \Delta \lambda_j = l_j \Delta A r_j + O \left( \| \Delta A \|_2 \right).
\]
(41)
In the non-symmetric case, the eigenvectors need not be orthogonal and the eigenvector matrix \( R \) need not be well conditioned. For this reason, it is possible that \( l_k \), which is a row of \( R^{-1} \) is very large. Working from (41) as we did for the symmetric case leads to
\[
\left| \frac{\Delta \lambda_j}{\lambda_j} \right| \leq \| l_j^* \|_2 \| r_j \|_2 \left| \frac{\Delta \lambda_j}{\lambda_j} \right|.
\]
Therefore, the condition number of the non-symmetric eigenvalue problem is (again because the inequalities are sharp)
\[
\kappa_j(A) = \| l_j \|_2 \| r_j \|_2 \| A \|_2.
\]
(42)
A useful upper bound is $\|l_j\| \|r_j\| \leq \kappa_{LS}(R)$, where $\kappa_{LS}(R) = \|R^{-1}\| \|R\|$ is the linear systems condition number of the right eigenvector matrix. Since $A$ is not symmetric, we cannot replace $\|A\|$ by $|\lambda|_{\text{max}}$ as we did for (40). In the symmetric case, the only reason for ill-conditioning is that we are looking for a (relatively) tiny number. For non-symmetric matrices, it is also possible that the eigenvector matrix is ill-conditioned. It is possible to show that if a family of matrices approaches a matrix with a Jordan block, the condition number of $R$ approaches infinity. For a symmetric or self-adjoint matrix, $R$ is orthogonal or unitary, so that $\|R\|_2 = \|R^*\|_2 = 1$ and $\kappa_{LS}(R) = 1$.

The eigenvector perturbation theory uses the same ideas, with the extra trick of expanding the derivatives of the eigenvectors in terms of the eigenvectors themselves. We expand the virtual perturbation $\dot{r}_j$ in terms of the eigenvectors $r_k$. Call the expansion coefficients $m_{jk}$, and we have

$$\dot{r}_j = \sum_{l=1}^n m_{jl} r_l.$$  

For the symmetric eigenvalue problem, if all the eigenvalues are distinct, the formula follows from multiplying (38) from the left by $r_k^*$:

$$m_{jk} = \frac{r_k^* \dot{r}_j}{\lambda_j - \lambda_k},$$

so that

$$\Delta r_j = \sum_{k \neq j} \frac{r_k^* \Delta r_j}{\lambda_j - \lambda_k} + O \left( \|\Delta A\|^2 \right).$$

(The term $j = k$ is omitted because $m_{jj} = 0$: differentiating $r_j^* r_j = 1$ gives $r_j^* \dot{r}_j = 0$.) This shows that the eigenvectors have condition number “issues” even when the eigenvalues are well-conditioned, if the eigenvalues are close to each other. Since the eigenvectors are not uniquely determined when eigenvalues are equal, this seems plausible. The unsymmetric matrix perturbation formula is

$$m_{kj} = \frac{l_j \dot{r}_k}{\lambda_j - \lambda_k}.$$  

Again, we have the potential problem of an ill-conditioned eigenvector basis, combined with the possibility of close eigenvalues. The conclusion is that the eigenvector conditioning can be problematic, even though the eigenvalues are fine, for closely spaced eigenvalues.

4 Software

4.1 Software for numerical linear algebra

There have been major government-funded efforts to produce high quality software for numerical linear algebra. This culminated in the public domain software package LAPACK. LAPACK is a combination and extension of earlier
packages *EISPACK*, for solving eigenvalue problems, and *LINPACK*, for solving systems of equations. LAPACK is used in many mathematical packages, including *MATLAB*.

Our advice is to use LAPACK for dense linear algebra computations whenever possible\(^\text{18}\), either directly or through an environment like *MATLAB*. These routines are extensively tested, and are much less likely to have subtle bugs than codes you developed yourself or copied from a textbook. The LAPACK software also includes condition estimators that are often more sophisticated than the basic algorithms themselves, and includes methods such as *equilibration* to improve the conditioning of problems.

LAPACK is built upon a library of *Basic Linear Algebra Subroutines* (BLAS). A BLAS library includes routines that perform such tasks as computing dot products and multiplying matrices. Different systems often have their own specially-tuned BLAS libraries, and these tuned libraries have much better performance than the reference implementation. Computations using LAPACK with an optimized BLAS can be faster than computations with the reference BLAS by an order of magnitude or more.

On many platforms, LAPACK and a tuned BLAS are packaged as a pre-compiled library. On machines running OS X, for example, LAPACK and a tuned BLAS are provided as part of the vecLib framework. On Windows and Linux machines, optimized LAPACK and BLAS implementations are part of Intel’s Math Kernel Libraries (MKL), which are sold commercially but with a free version under Windows, and as part of the AMD Core Math Library (ACML), which is freely available from AMD. The LAPACK Frequently Asked Questions list from [http://www.netlib.org/lapack](http://www.netlib.org/lapack) includes as its first item a list of places where you can get LAPACK and the BLAS pre-packaged as part of a vendor library.

LAPACK is written in Fortran 90\(^\text{19}\), not C++. It is not too difficult to call Fortran routines from C++, but the details vary from platform to platform. One alternative to mixed-language programming is to use CLAPACK, which is an automatic translation of LAPACK into the C language. CLAPACK provides a Fortran-style interface to LAPACK, but because it is written entirely in C, it is often easier to link. As of this writing, there is a standardized C interface to the BLAS (the CBLAS), but there is no standard C interface to LAPACK. One of the advantages of using LAPACK via a vendor-provided library is that many of these libraries provide a native C-style interface to the LAPACK routines.

### 4.2 Linear algebra in Matlab

The MATLAB system uses LAPACK and BLAS for most of its dense linear algebra operation. This includes the * command to multiply matrices, the

---

\(^\text{18}\) LAPACK has routines for *dense* linear algebra. *Sparse* matrices, such as those with only a few nonzero entries, should generally be solved using other packages. We will discuss sparse matrices briefly in Chapter ??

\(^\text{19}\) Most of LAPACK was developed in Fortran 77. However, the most recent version (LAPACK 3.2) uses some Fortran 90 features to provide extended precision routines.
N = 500;
A = rand(N);
B = rand(N);

tic; C = A*B; t1 = toc;

tic;
C2 = zeros(N);
for i = 1:N
    for j = 1:N
        for k = 1:N
            C2(i,j) = C2(i,j) + A(i,k)*B(k,j);
        end
    end
end
t2 = toc;

fprintf('For N = %d: built-in = %f sec; ours = %f sec\n', ...
        N, t1, t2);

Figure 1: Example MATLAB program that multiplies two matrices with a built-in primitive based on the BLAS and with a hand-written loop.

The backslash command to solve linear systems and least squares problems (\), the eig command to compute eigenvalues and eigenvectors, and the svd command to compute singular values and vectors.

Where MATLAB provides a built-in routine based on LAPACK and the BLAS, it will usually be much more efficient than a corresponding routine that you would write yourself. For example, consider the code shown in Figure 1 to multiply two matrices using the built-in * operator and a hand-written loop. On a desktop Pentium 4 system running version 7.6 of MATLAB, we can see the difference between the two calls clearly:

For N = 500: built-in = 0.061936 sec; ours = 5.681781 sec

Using the MATLAB built-in functions that call the BLAS is almost a hundred times faster than writing our own matrix multiply routine! In general, the most efficient MATLAB programs make heavy use of linear algebra primitives that call BLAS and LAPACK routines.

Where LAPACK provides very different routines for different matrix structures (e.g. different algorithms for symmetric and unsymmetric eigenvalue problems), MATLAB tests the matrix structure and automatically chooses an appropriate routine. It’s possible to see this indirectly by looking at the timing of the
A = rand(600); % Make a medium-sized random matrix
B = (A+A')/2; % Take the symmetric part of A
Bhat = B+eps*A; % Make B just slightly nonsymmetric

tic; eig(B); t1 = toc; % Time eig on a symmetric problem
tic; eig(Bhat); t2 = toc; % Time eig on a nonsymmetric problem
fprintf('Symmetric: %f sec; Nonsymmetric: %f sec\n', t1, t2);

Figure 2: Example MATLAB program that times the computation of eigenvalues for a symmetric matrix and a slightly nonsymmetric matrix.

following MATLAB script in Figure 2. On a desktop Pentium 4 system running version 7.6 of MATLAB, we can see the difference between the two calls clearly:

Symmetric: 0.157029 sec; Nonsymmetric: 1.926543 sec

4.3 Mixing C++ and Fortran

Like an enormous amount of scientific software, LAPACK is written in Fortran. Because of this, it is useful to call Fortran code from C. Unfortunately, many systems do not include a Fortran compiler; and even when there is a Fortran compiler, it may be difficult to figure out which libraries are needed to link together the Fortran and C codes. The f2c translator, available from http://www.netlib.org/f2c, translates Fortran codes into C. In this section, we will describe how to use f2c-translated codes in a C++ program. As an example, we will translate and use the reference version of the BLAS dot product routine ddot, which is available at http://www.netlib.org/blas/ddot.f. In our description, we assume a command line shell of the sort provided by OS X, Linux, or the Cygwin environment under Windows. We also assume that f2c has already been installed somewhere on the system.

The first step is to actually translate the code from Fortran to C:

f2c -C++ ddot.f

This command generates a function ddot.c with the translated routine. The -C++ option says that the translation should be done to C++-compatible C. In Fortran, ddot has the signature

DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY, INCY)
    * .. Scalar Arguments ..
    INTEGER INCX, INCY, N
    * ..
    * .. Array Arguments..
    DOUBLE PRECISION DX(*), DY(*)
```cpp
#include <iostream>
#include "f2c.h"

extern "C"
double ddot_(const integer& N, const double* dx,
             const integer& incX, const double* dy,
             const integer& incY);

int main()
{
    using namespace std;
    double xy[] = {1, 2};
    double result = ddot_(2, xy, 1, xy, 1);
    cout << "Result = " << result << endl; // Result = 5
    return 0;
}
```

Figure 3: Example C++ program that calls an f2c-translated routine.

The corresponding C function ddot_ has the signature

```c
double ddot_(integer* n, double* dx, integer* incx,
             double* dy, integer* incy);
```

We note three things about the signature for this translated routine:

- The C name of the routine is all lower case, with an underscore appended at the end: ddot_.
- Fortran uses call-by-reference semantics; that is, all arguments are passed by reference (through a pointer in C) rather than by value.
- An integer is a typedef defined in the f2c.h header file. Depending on the system, a Fortran integer may correspond to a C int or long.

Once we have translated the function to C, we need to be able to use it in another program. We will use as an example the code in Figure 3, which computes the dot product of a two-element vector with itself. We note a few things about this program:

- The extern "C" directive tells the C++ compiler to use C-style linkage\[^{20}\], which is what f2c requires. This is important when using C++ with any other language.

\[^{20}\] C++ linkage involves name mangling, which means that the compiler adds type information into its internal representation for a function name. In C-style linkage, we keep only the function name.

30
Rather than using C-style pointers to pass the vector length \( N \) and the parameters \( \text{incX} \) and \( \text{incY} \), we use C++-style references\(^{21}\). We tell the compiler that the \( \text{ddot} \) routine uses these values purely as input parameters by declaring that these are references to constant values. If we do this, the C++ compiler will allow us to pass in literal constants like 2 and 1 when we call \( \text{ddot} \), rather than requiring that we write:

\[
// If we had \text{ddot}(_{\text{integer* N, ...}})
\text{integer two = 2;}
\text{integer one = 1;}
\text{ddot(_{\&two, xy, &one, xy, &one});}
\]

Finally, we compile the program:

```
c++ -o example.exe -I f2cdir example.cc ddot.c -L f2cdir -lf2c
```

Here -I \( f2cdir \) tells the compiler to search the directory \( f2cdir \) for the header file \( f2c.h \); and the flag -L \( f2cdir \) tells the compiler where to find the support library \( \text{libf}2c \).

The steps we used to incorporate the Fortran \( \text{ddot} \) routine in our C++ code using \( f2c \) are similar to the steps we would use to incorporate a Fortran library into any C++ code:

1. Follow the directions to compile the library (or to translate it using \( f2c \), in our case). If you can find a pre-built version of the library, you may be able to skip this step.

2. Write a C++ program that includes a prototype for the library routine. Remember that the C name will probably be slightly different from the Fortran name (a trailing underscore is common), and that all arguments in Fortran are passed by reference.

3. Compile your C++ program and link it against the Fortran library and possibly some Fortran support libraries.

## 5 Resources and further reading

For a practical introduction to linear algebra written by a numerical analyst, try the books by Gilbert Strang \([9, 10]\). More theoretical treatments may be found in the book by Peter Lax \([5]\) or the one by Paul Halmos \([3]\). The linear algebra book by Peter Lax also has a beautiful discussion of eigenvalue perturbation theory and some of its applications. More applications may be found in the book *Theory of Sound* by Lord Rayleigh (reprinted by Dover Press) and in any book on quantum mechanics \([7]\).

\(^{21}\) The C++ compiler would complain at us if it were keeping type information on the arguments to \( \text{ddot} \), but those are discarded because the compiler is using C linkage. References and pointers are implemented nearly identically, so we can get away with this in practice.

Beresford Parlett has a book on the theory and computational methods for the symmetric eigenvalue problem [6]; though it does not include some of the most recent methods, it remains an excellent and highly readable book. G.W. Stewart has devoted the second volume of his *Matrix Algorithms* book entirely to eigenvalue problems, including both the symmetric and the nonsymmetric case [8].

The software repository *Netlib*, http://netlib.org, is a source for LAPACK and for many other numerical codes.

6 Exercises

1. Let $L$ be the differentiation operator that takes $P_3$ to $P_2$ described in Section 2.2. Let $f_k = H_k(x)$ for $k = 0, 1, 2, 3$ be the Hermite polynomial basis of $P_3$ and $g_k = H_k(x)$ for $k = 0, 1, 2$ be the Hermite basis of $P_2$. What is the matrix, $A$, that represents this $L$ in these bases?

2. Suppose $L$ is a linear transformation from $V$ to $V$ and that $f_1, \ldots, f_n$, and $g_1, \ldots, g_n$ are two bases of $V$. Any $u \in V$ may be written in a unique way as $u = \sum_{k=1}^n v_k f_k$, or as $u = \sum_{k=1}^n w_k g_k$. There is an $n \times n$ matrix, $R$ that relates the $f_k$ expansion coefficients $v_k$ to the $g_k$ coefficients $w_k$ by $v_j = \sum_{k=1}^n r_{jk} w_k$. If $v$ and $w$ are the column vectors with components $v_k$ and $w_k$ respectively, then $v = Rw$. Let $A$ represent $L$ in the $f_k$ basis and $B$ represent $L$ in the $g_k$ basis.

   (a) Show that $B = R^{-1}AR$.

   (b) For $V = P_3$, and $f_k = x^k$, and $g_k = H_k$, find $R$.

   (c) Let $L$ be the linear transformation $Lp = q$ with $q(x) = \partial_x (xp(x))$. Find the matrix, $A$, that represents $L$ in the monomial basis $f_k$.

   (d) Find the matrix, $B$, that represents $L$ in the Hermite polynomial basis $H_k$.

   (e) Multiply the matrices to check explicitly that $B = R^{-1}AR$ in this case.

3. If $A$ is an $n \times m$ matrix and $B$ is an $m \times l$ matrix, then $AB$ is an $n \times l$ matrix. Show that $(AB)^* = B^*A^*$. Note that the incorrect suggestion $A^*B^*$ in general is not compatible for matrix multiplication.
4. Let $V = \mathbb{R}^n$ and $M$ be an $n \times n$ real matrix. This exercise shows that $\|u\| = (u^*Mu)^{1/2}$ is a vector norm whenever $M$ is positive definite (defined below).

(a) Show that $u^*Mu = u^*M^*u = u^* \left( \frac{1}{2}(M + M^*) \right) u$ for all $u \in V$. This means that as long as we consider functions of the form $f(u) = u^*Mu$, we may assume $M$ is symmetric. For the rest of this question, assume $M$ is symmetric. Hint: $u^*Mu$ is a $1 \times 1$ matrix and therefore equal to its transpose.

(b) Show that the function $\|u\| = (u^*Mu)^{1/2}$ is homogeneous: $\|au\| = |a| \|u\|$.

(c) We say $M$ is positive definite if $u^*Mu > 0$ whenever $u \neq 0$. Show that if $M$ is positive definite, then $\|u\| \geq 0$ for all $u$ and $\|u\| = 0$ only for $u = 0$.

(d) Show that if $M$ is symmetric and positive definite (SPD), then $|u^*Mv| \leq \|u\| \|v\|$. This is the Cauchy–Schwarz inequality. Hint (a famous old trick): $\phi(t) = (u + tv)^*M(u + tv)$ is a quadratic function of $t$ that is non-negative for all $t$ if $M$ is positive definite. The Cauchy–Schwarz inequality follows from requiring that the minimum value of $\phi$ is not negative, assuming $M^* = M$.

(e) Use the Cauchy–Schwarz inequality to verify the triangle inequality in its squared form $\|u + v\|^2 \leq \|u\|^2 + 2 \|u\| \|v\| + \|v\|^2$.

(f) Show that if $M = I$ then $\|u\|$ is the $l^2$ norm of $u$.

5. Verify that $\|p\|$ defined by (5) on $V = P_3$ is a norm as long as $a < b$.

6. Suppose $A$ is the $n \times n$ matrix that represents a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$ in the standard basis $e_k$. Let $B$ be the matrix of the same linear transformation in the scaled basis $e_k$.\n
(a) Find a formula for the entries $b_{jk}$ in terms of the $a_{jk}$ and $\pi_k$.

(b) Find a matrix formula for $B$ in terms of $A$ and the diagonal scaling matrix $W = \text{diag}(\pi_k)$ (defined by $w_{kk} = \pi_k$, $w_{jk} = 0$ if $j \neq k$) and $W^{-1}$.

7. Show that if $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ and $A = uv^*$, then $\|A\|_{l2} = \|u\|_{l2} \cdot \|v\|_{l2}$. Hint: Note that $Aw = bu$ where $b$ is a scalar, so $\|Aw\|_{l2} = |b| \cdot \|u\|_{l2}$. Also, be aware of the Cauchy–Schwarz inequality: $|v^*w| \leq \|v\|_{l2} \cdot \|w\|_{l2}$.

8. Suppose that $A$ is an $n \times n$ invertible matrix. Show that

$$\|A^{-1}\| = \max_{u \neq 0} \frac{\|u\|}{\|Au\|} = \left( \min_{u \neq 0} \frac{\|Au\|}{\|u\|} \right)^{-1}.$$ 

9. The symmetric part of the real $n \times n$ matrix is $M = \frac{1}{2} (A + A^*)$. Show that $\nabla \left( \frac{1}{2} x^*Ax \right) = Mx$. 

33
10. The informal condition number of the problem of computing the action of $A$ is

$$\kappa(A) = \max_{x \neq 0, \Delta x \neq 0} \frac{\|A(x+\Delta x) - Ax\|}{\|Ax\|} \frac{\|x + \Delta x\|}{\|x\|}.$$ 

Alternatively, it is the sharp constant in the estimate

$$\frac{\|A(x + \Delta x) - Ax\|}{\|Ax\|} \leq C \cdot \frac{\|x + \Delta x\|}{\|x\|},$$

which bounds the worst case relative change in the answer in terms of the relative change in the data. Show that for the $l^2$ norm,

$$\kappa = \sigma_{\text{max}}/\sigma_{\text{min}},$$

the ratio of the largest to smallest singular value of $A$. Show that this formula holds even when $A$ is not square.

11. We wish to solve the boundary value problem for the differential equation

$$\frac{1}{2} \partial_x^2 u = f(x) \text{ for } 0 < x < 1,$$

with boundary conditions

$$u(0) = u(1) = 0.$$  

We discretize the interval $[0, 1]$ using a uniform grid of points $x_j = j\Delta x$ with $n\Delta x = 1$. The $n - 1$ unknowns, $U_j$, are approximations to $u(x_j)$, for $j = 1, \ldots, n - 1$. If we use a second order approximation to $\frac{1}{2} \partial_x^2 u$, we get discrete equations

$$\frac{1}{2} \frac{1}{\Delta x^2} (U_{j+1} - 2U_j + U_{j-1}) = f(x_j) = F_j.$$  

Together with boundary conditions $U_0 = U_n = 0$, this is a system of $n - 1$ linear equations for the vector $U = (U_1, \ldots, U_{n-1})^*$ that we write as $AU = F$.

(a) Check that there are $n - 1$ distinct eigenvectors of $A$ having the form $r_{kj} = \sin(k\pi x_j)$. Here $r_{kj}$ is the $j$ component of eigenvector $r_k$. Note that $r_{kj+1} = \sin(k\pi x_{j+1}) = \sin(k\pi(x_j + \Delta x))$, which can be evaluated in terms of $r_{kj}$ using trigonometric identities.

(b) Use the eigenvalue information from part (a) to show that $\|A^{-1}\| \rightarrow 2/\pi^2$ as $n \rightarrow \infty$ and $\kappa(A) = O(n^2)$ (in the informal sense) as $n \rightarrow \infty$. All norms are $l^2$.

(c) Suppose $\bar{U}_j = u(x_j)$ where $u(x)$ is the exact but unknown solution of (43), (44). Show that if $u(x)$ is smooth then the residual\textsuperscript{22}, $R = \|AU - F\| = \sum_j 2(1 - \sin(k\pi x_j) - f(x_j))$ (in the informal sense) as $n \rightarrow \infty$. All norms are $l^2$.

\textsuperscript{22}Residual refers to the extent to which equations are not satisfied. Here, the equation is $AU = F$, which $\bar{U}$ does not satisfy, so $R = A\bar{U} - F$ is the residual.
$A \tilde{U} - F$, satisfies $\|R\| = O(\Delta x^2) = O(1/n^2)$. For this to be true we have to adjust the definition of $\|U\|$ to be consistent with the $L^2$ integral $\|u\|^2_{L^2} = \int_{x=0}^1 u^2(x)dx$. The discrete approximation is $\|U\|^2 = \Delta x \sum_{k=1}^n U_j^2$.

(d) Show that $A \left( U - \tilde{U} \right) = R$. Use part (b) to show that $\|U - \tilde{U}\| = O(\Delta x^2)$ (with the $\Delta x$ modified $\|\|$).

(e) (harder) Create a fourth order five point central difference approximation to $\partial_x^2 u$. You can do this using Richardson extrapolation from the second order three point formula. Use this to find an $A$ so that solving $AU = F$ leads to a fourth order accurate $U$. The hard part is what to do at $j = 1$ and $j = n - 1$. At $j = 1$ the five point approximation to $\partial_x^2 u$ involves $U_0$ and $U_{n-1}$. It is fine to take $U_0 = u(0) = 0$. Is it OK to take $U_{n-1} = -U_1$?

(f) Write a program in Matlab to solve $AU = F$ for the second order method. The matrix $A$ is symmetric and tridiagonal (has nonzeros only on three diagonals, the main diagonal, and the immediate sub and super diagonals). Use the Matlab matrix operation appropriate for symmetric positive definite tridiagonal matrices. Do a convergence study to show that the results are second order accurate.

(g) (extra credit) Program the fourth order method and check that the results are fourth order when $f(x) = \sin(\pi x)$ but not when $f(x) = \max(0, .15 - (x - .5)^2)$. Why are the results different?

12. This exercise explores conditioning of the non-symmetric eigenvalue problem. It shows that although the problem of computing the fundamental solution is well-conditioned, computing it using eigenvalues and eigenvectors can be an unstable algorithm because the problem of computing eigenvalues and eigenvectors is ill-conditioned. For parameters $0 < \lambda < \mu$, there is a Markov chain transition rate matrix, $A$, whose entries are $a_{jk}$ if $|j - k| > 1$ if $1 \leq j \leq n - 2$, $a_{j,j-1} = \mu$, $a_{j,j} = -(\lambda + \mu)$, and $a_{j,j+1} = \lambda$ (taking $j$ and $k$ to run from 0 to $n - 1$). The other cases are $a_{00} = -\lambda$, $a_{01} = \lambda$, $a_{n-1,n-1} = -\mu$, and $a_{n-1,n-2} = \mu$. This matrix describes a continuous time Markov process with a random walker whose position at time $t$ is the integer $X(t)$. Transitions $X \rightarrow X + 1$ happen with rate $\lambda$ and transitions $X \rightarrow X - 1$ have rate $\mu$. The transitions $0 \rightarrow -1$ and $n - 1 \rightarrow n$ are not allowed. This is the $M/M/1$ queue used in operations research to model queues ($X(t)$ is the number of customers in the queue at time $t$, $\lambda$ is the rate of arrival of new customers, $\mu$ is the service rate. A customer arrival is an $X \rightarrow X + 1$ transition.). For each $t$, we can consider the row vector $p(t) = (p_1(t), \ldots, p_n(t))$ where $p_j(t) = \text{Prob}(X(t) = j)$. These probabilities satisfy the differential equation $\dot{p} = \frac{d}{dt} p = pA$. The solution can be written in terms of the fundamental solution, $S(t)$, which in an $n \times n$ matrix that satisfies $\dot{S} = SA$, $S(0) = I$.

(a) Show that if $\dot{S} = SA$, $S(0) = I$, then $p(t) = p(0)S(t)$.
(b) The matrix exponential may be defined through the Taylor series
\[ \exp(B) = \sum_{k=0}^{\infty} \frac{1}{k!} B^k. \]
Use matrix norms and the fact that \( \|B^k\| \leq \|B\|^k \) to show that the infinite sum of matrices converges.

(c) Show that the fundamental solution is given by \( S(t) = \exp(tA) \). To do this, it is enough to show that \( \exp(tA) \) satisfies the differential equation \( \frac{d}{dt} \exp(tA) = \exp(tA)A \) using the infinite series, and show \( \exp(0A) = I \).

(d) Suppose \( A = LR \) is the eigenvalue and eigenvector decomposition of \( A \), show that \( \exp(tA) = L \exp(t\Lambda)R \), and that \( \exp(t\Lambda) \) is the obvious diagonal matrix.

(e) Use the Matlab function \([R, \text{Lam}] = \text{eig}(A)\) to calculate the eigenvalues and right eigenvector matrix of \( A \). Let \( r_k \) be the \( k \)th column of \( R \). For \( k = 1, \ldots, n \), print \( r_k, Ar_k, \lambda_k r_k \), and \( \|\lambda_k - Ar_k\| \) (you choose the norm). Mathematically, one of the eigenvectors is a multiple of the vector \( 1 \) defined in part h. The corresponding eigenvalue is \( \lambda = 0 \). The computed eigenvalue is not exactly zero. Take \( n = 4 \) for this, but do not hard wire \( n = 4 \) into the Matlab code.

(f) Let \( L = R^{-1} \), which can be computed in Matlab using \( \text{L} = \text{R}^{*-1} \). Let \( l_k \) be the \( k \)th row of \( L \), check that the \( l_k \) are left eigenvectors of \( A \) as in part e. Corresponding to \( \lambda = 0 \) is a left eigenvector that is a multiple of \( p_\infty \) from part h. Check this.

(g) Write a program in Matlab to calculate \( S(t) \) using the eigenvalues and eigenvectors of \( A \) as above. Compare the results to those obtained using the Matlab built in function \( S = \text{expm}(t*A) \). Use the values \( \lambda = 1, \mu = 4, t = 1 \), and \( n \) ranging from \( n = 4 \) to \( n = 80 \). Compare the two computed \( S(t) \) (one using eigenvalues, the other just using \( \text{expm} \)) using the \( l_1 \) matrix norm. Use the Matlab routine \( \text{cond}(R) \) to compute the condition number of the eigenvector matrix, \( R \). Print three columns of numbers, \( n \), error, condition number. Comment on the quantitative relation between the error and the condition number.

(h) Here we figure out which of the answers is correct. To do this, use the known fact that \( \lim_{t \to -\infty} S(t) = S_\infty \) has the simple form \( S_\infty = 1 p_\infty \), where \( 1 \) is the column vector with all ones, and \( p_\infty \) is the row vector with \( p_{\infty,j} = ((1 - r)/(1 - r^n)) r^j \), with \( r = \lambda/\mu \). Take \( t = 3 * n \) (which is close enough to \( t = \infty \) for this purpose) and the same values of \( n \) and see which version of \( S(t) \) is correct. What can you say about the stability of computing the matrix exponential using the ill conditioned eigenvalue/eigenvector problem?

13. This exercise explores eigenvalue and eigenvector perturbation theory for the matrix \( A \) defined in exercise 12. Let \( B \) be the \( n \times n \) matrix with \( b_{jk} = 0 \) for all \( (j,k) \) except \( b_{00} = -1 \) and \( b_{1,n-1} = 1 \) (as in exercise 12, indices run from \( j = 0 \) to \( j = n - 1 \)). Define \( A(s) = A + sB \), so that \( A(0) = A \) and \( \frac{dA(s)}{ds} = B \) when \( s = 0 \).
(a) For \( n = 20 \), print the eigenvalues of \( A(s) \) for \( s = 0 \) and \( s = .1 \). What does this say about the condition number of the eigenvalue eigenvector problem? All the eigenvalues of a real tridiagonal matrix are real\(^{23} \) but that \( A(s = .1) \) is not tridiagonal and its eigenvalues are not real.

(b) Use first order eigenvalue perturbation theory to calculate \( \dot{\lambda}_k = \frac{d}{ds} \lambda_k \) when \( s = 0 \). What size \( s \) do you need for \( \Delta \lambda_k \) to be accurately approximated by \( s \dot{\lambda}_k \)? Try \( n = 5 \) and \( n = 20 \). Note that first order perturbation theory always predicts that eigenvalues stay real, so \( s = .1 \) is much too large for \( n = 20 \).

\(^{23}\)It is easy to see that if \( A \) is tridiagonal then there is a diagonal matrix, \( W \), so that \( WAW^{-1} \) is symmetric. Therefore, \( A \) has the same eigenvalues as the symmetric matrix \( WAW^{-1} \).
References


