Introduction to Matrix Computations

Lecture notes support CME200 at Stanford

Margot Gerritsen
margot.gerritsen@stanford.edu

August 15, 2016

Contents

Preface

I love matrices, and matrix computations. I sometimes joke that matrices are so famous, they even created a blockbuster trilogy in their honor. If you've seen The Matrix, you may remember that Morpheus proclaims "The Matrix is everywhere. It's all around uneven. Even now, in this very room". He's right. Matrices are at the very core of nearly all engineering and applied sciences problems. No matter what your interests and future career plans, if you stay in the engineering, computing or science fields, you will encounter matrices. This course is therefore my very favorite course: it is incredibly useful.

This is particularly true for those of you interested, or already active in the broad area of data science. To understand data mining, machine learning and deep learning, as well as optimization, a solid grasp of matrix computations (also referred to as the field of linear algebra) is essential. Here are just a few examples that illustrate the importance of linear algebra and the italicized concepts discussed and motivated in these notes:

- Data are often represented in matrix form. For example, each row of a matrix may correspond to one item (data point, person, ...) and each column to a particular feature of the item. Data are analyzed through matrix and vector operations. Think for example about understanding relations between features, which means understanding how columns depend on each other. This can be done with a *QR factorization*.
- Dimensionality reduction, as in PCA, may sound different than performing a *Singular Value Decomposition*, but is really the same thing.
- The so-called L2 minimization is really just *least squares*
- Often machine learning problems are in the form of *overdetermined matrix-vector systems*, in which the matrix has many more rows than columns.
- Collaborative filtering and alternating least squares for recommender systems are in essence nonnegative *matrix factorization*
- Ranking algorithms, like the page rank algorithm, can be formulated as *eigenvalue*

problems. The algorithm that formed the basis of Google was really a smart form of the *Power method*, or *Jacobi iteration*.

- Search algorithms heavily use similarity measures (such as the cosine similarity) that are derived from vector operations such as the *inner products*.
- The behavior of the algorithms using inexact arithmetic (as on a computer with round-off errors), is studied extensively in the field of linear algebra, and concepts like the *conditioning* of matrices are critical.

These notes and exercises are available free of charge at my website. They were written to support CME200 at Stanford University, an upperclass/Masters level course in matrix computations with a focus on applications, as well as the ICME short courses in linear algebra. Exercises are at the end of each chapter. Comprehensive review questions are given in the last chapter. The book comes with answers to exercises. Your instructor may or may not include these. The book is in constant flux and will certainly still contain errors, both grammatical and otherwise. If you find any errors, please let me know and email me at margot.gerritsen@stanford.edu. The website will always have the latest version with a list of improvements since the last version.

You may also like the book "Linear Algebra and its Applications" by Gilbert Strang. It is my favorite upperclass book on linear algebra and covers a lot of the material discussed in these notes. Upon completion of this course, continue your studies with books by:

- Jim Demmel, Applied Numerical Linear Algebra
- Tim Davis, Direct Methods for Sparse Linear Systems
- Trefethen and Bau, Numerical Linear Algebra
- Golub and van Loan, Matrix Computations

I hope the notes are useful and good luck with your studies.

Best wishes, Margot Gerritsen

Notation

In these notes, we will use the following notation

$a, b, c, \ldots, z$	Scalars
$A, B, C, \ldots, Z$	Matrices of dimension $m \times n$
$\vec{a}, \vec{b}, \ldots, \vec{z}$	Column vectors of dimension $n \times 1$
$\vec{a}^T$	Row vector (transpose of column vector) of dimension $1 \times n$
$\vec{x} \cdot \vec{y}$, or $\vec{x}^T \vec{y}$	Inner product of vectors $\vec{x}$ and $\vec{y}$
A^T	Transpose of the matrix A
A^{-1}	Inverse of a nonsingular square matrix A
$\lambda(A)$	Eigenvalue(s) of the matrix A
$\vec{a}_j$	$\vec{a}_j$ is the j-th column of A
$\vec{r_i}^T$ or $\vec{r_{A_i}}^T$	$\vec{r_i}^T$ or $\vec{r_{A_i}}^T$ is the i-th row of A
$\sigma(A)$	Singular values of the matrix A
$A = LU$	LU-decomposition of the matrix A, with L unit lower triangular and U upper triangular
$A = QR$	QR-decomposition of the matrix A, with Q orthogonal and R upper triangular
$A = U\Sigma V^T$	Singular value decomposition of the matrix A, with U and V orthogonal and Σ diagonal (contains singular values)
$A = Y\Lambda Y^{-1}$	Canonical transform of the matrix A, with Λ diagonal (contains eigenvalues)

Chapter 1

Matrices, types and properties

Supporting material for this chapter:

- TedX 2014, "The Beauty I see in Algebra". This 13 minute talk gives a brief overview of why linear algebra is so incredible useful, and beautiful.
- Online mathematics modules at the Stanford Jolts pages Check these website from time to time as new modules are being added on a regular basis.

1.1 What are matrices?

The easy part of matrix computations is explaining what a matrix looks like. It is simply a square or rectangular table of numbers, which we typically enclose by square brackets, like these three:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 6 & 7 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & -1 \\ -1 & 3 \\ 0 & 4 \\ -1 & 0 \end{bmatrix}.$$

Here, A is a matrix with three rows and three columns, and we call it a 3×3 matrix. B has two rows and three columns and is 2×3, and C is a 4×2 matrix. In general, a matrix is $m \times n$ when it has m rows and n columns. Notice how we use capital letters to name matrices. This is standard practice to help distinguish them from scalars and we will use it throughout these notes. Check also the notation table in the preface of these notes.

A general $m \times n$ matrix A is written as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & & \ddots & & \vdots \\ a_{m-1,1} & a_{m-1,2} & \dots & a_{m-1,n-1} & a_{m-1,n} \\ a_{m1} & a_{m2} & \dots & a_{m,n-1} & a_{mn} \end{bmatrix}. \tag{1.1}$$

The element a_{ij} of the matrix A is the element in the i-th row and j-th column.

We often like to refer to the rows or columns of a matrix. In these notes, we will denote the j-th column of matrix A by $\vec{a}_j$, with $\vec{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}$. For a matrix B, we'd use $\vec{b}_j$, and so on.

We will denote the i-th row of a matrix A by $\vec{r}_i^T$, or, if we want to very clearly indicate which matrix we are referring to, by $\vec{r}_{A,i}^T$. Here, the transpose is used to indicate that this is indeed a *row* vector, so $\vec{r}_i^T = [a_{i1}, a_{i2}, \dots, a_{in}]$.

1.2 What are matrices used for?

Matrices are used throughout engineering and the sciences for many different reasons. In these notes, we will look at three main uses of matrices:

- store coefficients in systems of equations
- store connectivity information in graphs/networks
- represent operators on vectors, such as rotations and reflections

1.2.1 Representing systems of linear equations

The small system of linear equations

$$\begin{aligned} 2x_1 + 3x_2 - \ x_3 &= 4, \\ x_1 \qquad\quad + 2x_3 &= 3, \\ 2x_2 + 3x_3 &= 5, \end{aligned}$$

can also be written as

$$\begin{aligned} 2x_1 + 3x_2 - \ \ x_3 &= 4, \\ x_1 + 0x_2 + 2x_3 &= 3, \\ 0x_1 + 2x_2 + 3x_3 &= 5. \end{aligned}$$

This second notation may look a little unnecessary, but the nice thing about it is that the unknowns x_1, x_2 and x_3 are present in all equations and written in the exact same order, and that each unknown in each equation is explicitly multiplied by a coefficient (with some of the coefficients are zero). To store the system of equations, all we therefore have to remember are the order of the unknowns, the coefficients multiplying the unknowns and the values on the right hand sides of the equations. In other words, we need a table of coefficients, which we call A, a vector $\vec{x}$ with the unknowns in the proper order, and another vector $\vec{b}$ with all the right hand side values:

$$A = \begin{bmatrix} 2 & 3 & -1 \\ 1 & 0 & 2 \\ 0 & 2 & 3 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}.$$

We now simply write the system of equations as $A\vec{x} = \vec{b}$. Here, the matrix-vector multiplication $A\vec{x}$ must be defined so that it gives the vector $\begin{bmatrix} 2x_1 + 3x_2 - x_3 \\ x_1 + 0x_2 + 2x_3 \\ 0x_1 + 2x_2 + 3x_3 \end{bmatrix}$. Then $A\vec{x} = \vec{b}$ gives us the original three equations.

You can see that each element of this vector $A\vec{x}$ is simply the inner product of a row of A with $\vec{x}$:

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &= \vec{r}_1^T \vec{x}, \\ x_1 + 0x_2 + 2x_3 &= \vec{r}_2^T \vec{x}, \\ 0x_1 + 2x_2 + 3x_3 &= \vec{r}_3^T \vec{x}. \end{aligned}$$

In general notation, we have for a 3×3 matrix

$$A\vec{x} = \begin{bmatrix} \vec{r}_1^T \vec{x} \\ \vec{r}_2^T \vec{x} \\ \vec{r}_3^T \vec{x} \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \end{bmatrix}. \tag{1.2}$$

So, this is how we compute a matrix-vector product. Note that this is simply something that everyone agreed on: when you compute $A\vec{x}$ what you get is the vector whose elements are all inner products of rows of A with $\vec{x}$.

We can also write $A\vec{x}$ in terms of the columns of A. How? Observe that in (1.2), x_1 is multiplied by a coefficient that is stored in $\vec{a}_1$ (the first column of A), x_2 is always multiplied by a coefficient that is stored in $\vec{a}_2$ (the second column of A) and, yes, x_3 by a coefficient stored in $\vec{a}_3$, the third column of A. In other words, the system of equations $A\vec{x} = \vec{b}$ can be written as

$$A\vec{x} = x_1\vec{a}_1 + x_2\vec{a}_2 + x_3\vec{a}_3 = \vec{b}.$$

$A\vec{x}$ can thus be interpreted as a linear combination of the columns of A with coefficients equal to the elements of $\vec{x}$.

The general system of m equations in n unknowns

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 \ + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 \ + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned}$$

can be written as $A\vec{x} = \vec{b}$ with the $m \times n$ matrix A as given in (1.1). The vector $\vec{x}$ has n elements and the vector $\vec{b}$ has m elements. Note that the number of elements in $\vec{x}$ must be the same as the number of columns of A and the number of elements in $\vec{b}$ must be equal to the number of rows of A.

Each *row* $\vec{r}_i^T$ of the matrix corresponds to the i-th equation in the system. Each *column* $\vec{a}_j$ of the matrix corresponds to the *unknown* x_j because $\vec{a}_j$ contains all the coefficients that multiply the unknown $\vec{x}_j$ over all equations in the system.

Just as for the 3×3 example above, we can write this matrix-vector equation in two other ways:

$$A\vec{x} = \begin{bmatrix} \vec{r}_1^T\vec{x} \\ \vec{r}_2^T\vec{x} \\ \vdots \\ \vec{r}_m^T\vec{x} \end{bmatrix},$$

and also

$$A\vec{x} = x_1\vec{a}_1 + \ldots + x_n\vec{a}_n = \sum_{j=1}^{n} x_j\vec{a}_j.$$

Both of these interpretations of $A\vec{x}$ are very handy, and we'll use them throughout the notes.

1.2.2 Matrices representing networks or graphs

Another very common use of matrices is to represent networks and graphs. Here, we look at one example: the connectivity matrix. Suppose that you have a network that shows connections between m items. Imagine that you store these connections in a very straightforward fashion in a matrix A: if $a_{ij} = 1$ then there is a connection between item i and item j and if $a_{ij} = 0$ there is not. This will generate a $m \times m$ matrix A that may be very *sparse* (that is, contain many zeros) if the number of connections between items is limited. For example, the connectivity matrix

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

shows that items 1 and 2 are connected, as are items 1 and 3. However, there is no connection between items 2 and 3, and the zero diagonal shows that we do not consider, in this example, an item to be connected to itself (in another application, we may say that items are always self-connected and put a 1 everywhere on the diagonal).

This mapping from networks/graphs to matrices is very handy. Sometimes we go the other way around: we have matrix and display the corresponding graph. Not only does this often give useful insights in the matrix structure, but it also allows for fantastic visualizations (see, for example, the wonderful collection by Tim Davis at http://www.cise.ufl.edu/research/sparse/matrices/, as well as the LCSH galaxy at http://cads.stanford.edu/lcshgalaxy/. These visuals are also included in the TedX talk referred to at the start of this chapter).

Can you expect connectivity matrices to be symmetric?

Suppose that the network you are looking at is a friends network on Facebook. In that case, each time there is a connection from friend i to j, there is also a connection from friend j to i. In other words, $a_{ij} = a_{ji}$ for all i and j. The corresponding A is then symmetric. Another symmetric A would be a distance table between locations. The small example connectivity matrix given above is a symmetric connectivity matrix.

But symmetry is not always guaranteed. Let's say, for example, that we are storing connections between webpages. Our definition of a connection between webpage i and webpage j is that webpage i contains, somewhere on its page, a hyperlink to website j. Now it is not necessarily true that webpage j will have a hyperlink back to webpage i. For example, at ICME we link from our homepage to the Stanford homepage, but not the other way around (even though we wished this was the case!).

We will have a closer look at these types of matrices in later chapters. You will see them come up as an example in the chapter on eigenvalues and eigenvectors, for example, as eigenvalue computations of connectivity matrices form the core of the famous page rank algorithm.

1.2.3 Matrices as operators

The third common use of matrices is to represent operators on vectors.

Let's go back to the small example system we had earlier:

$$\begin{aligned} 2x_1 + 3x_2 - \ x_3 &= 4, \\ x_1 \qquad\quad + 2x_3 &= 3, \\ 2x_2 + 3x_3 &= 5, \end{aligned}$$

The unique solution to this system of equations is $\vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. In other words, we know that

$$\begin{bmatrix} 2 & 3 & -1 \\ 1 & 0 & 2 \\ 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}.$$

We can interpret this equation this way too: when A is applied to the vector $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, it changes this vector into $\begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}$. Viewed in this way, a matrix is an *operator* that takes one vector, and transforms it to another. And what is that transformation? A can change both the *direction* and the *length* of $\vec{x}$. We often think about matrices this way. It can give really nice insights and make some computations easier.

As example, let's look at a matrix Q that rotates a vector $\vec{x}$ in two dimensions over an angle θ. What would such a matrix look like? We can find it easily if we think about what the matrix Q should give for two simple vectors: the unit vectors $\vec{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\vec{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. The rotation matrix should rotate $\vec{e_1}$ to the vector $\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$, and $\vec{e_2}$ to the vector $\begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$. But remember that $Q\vec{x}$ is equal to $x_1\vec{q}_1 + x_2\vec{q}_2$, a linear combination of the columns of Q. We immediately see that $Q\vec{e}_1 = \vec{q}_1$ and $Q\vec{e}_2 = \vec{q}_2$. This then gives that

the *rotation matrix* $Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$. Check for yourself that this matrix rotates other vectors just the way you expect.

Another operator that is used often is a *projection matrix*. Suppose that in two dimensions, we want to find the operator matrix P that takes any vector $\vec{x}$ and projects it orthogonally onto the x-axis. In other words, P takes a vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and makes it into $\begin{bmatrix} x_1 \\ 0 \end{bmatrix}$. What would P look like? It's not hard to check that this projection matrix $P = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$.

As another example, let's look at what a reflector matrix R look like that reflects a vector in the y-axis. Such a matrix changes the vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ into the vector $\begin{bmatrix} -x_1 \\ x_2 \end{bmatrix}$. The matrix that achieves this is given by $R = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$. Check this for yourself.

1.3 What types of matrices will we encounter?

1.3.1 The simplest matrices: null and identity

The simplest matrix you can probably imagine is the matrix A with $a_{ij} = 0$ for all i, j. Such a matrix is referred to as the *null matrix*.

The next simplest is arguably the so-called *identity matrix*, which we denote by I. It is the matrix that, when applied to any vector $\vec{x}$, leaves the vector unchanged: $I\vec{x} = \vec{x}$. You can understand directly that this then must be a square $m \times m$ matrix, and it is also not hard to see it must look like this

$$I = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

.

1.3.2 Common matrix structures

In many applications, we encounter square *diagonal* matrices A that have all $a_{ij} = 0, i \neq j$. That is, only the diagonal elements $a_{ii}, i = 1, \dots, m$ are allowed to be nonzero (but they don't all have to be nonzero).

Other common matrices are the triangular matrices. In a square *lower triangular matrix* L, nonzero elements may only appear on the diagonal or below it. In other words, all elements above the diagonal are zero: $l_{ij} = 0, j > i$. Similarly, in a square *upper triangular matrix* U nonzero elements may only appear on the diagonal or above it, which means that $u_{ij} = 0, i > j$.

Yet another type of matrix is the *banded matrix*. Banded matrices have nonzero elements within a band around the diagonal. For example, a tridiagonal matrix is a matrix with nonzeros only on the main diagonal, and the two sub diagonals directly above and below it. We say that it has a bandwidth of 3. A bi-diagonal matrix has nonzeros only on the main diagonal and one of the sub diagonals directly above or below it. It has a bandwidth of 2.

1.3.3 Sparse matrices

In engineering problems, it is quite common to encounter matrices that have many more zeros than nonzeros. They are called *sparse matrices*. Sparse matrices are often stored differently than dense matrices: just the locations of the nonzeros are stored (two integer values - one for the row number and one for the column number) and the nonzero value. The zeros are simply not stored. MATLAB has functions developed specifically for dealing with sparse matrices that run very much faster on large sparse matrices than the general dense matrix functions. Whenever you have sparsity, use the sparse matrix storage scheme and the appropriate MATLAB functions.

1.3.4 Symmetric matrices

Apart from these matrices, that all have a special nonzero structure, we will also spend quite a bit of time thinking about the so-called *symmetric matrices*. A symmetric matrix A is a matrix for which $a_{i,j} = a_{j,i}$. One simple example is the matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 4 \\ 3 & 4 & 1 \end{bmatrix}$. The symmetry line is the main diagonal from the top left corner to the bottom right corner. When we know that a matrix is symmetric, we only have to store the diagonal and either the strictly lower, or strictly upper, triangular part. This saves in memory required. This does not seem like a big deal at all for small matrices, but it is when you work with matrices that have millions of rows and columns (as we often do in engineering or the applied sciences). You will learn in this course that symmetric matrices have all sorts of other interesting properties that make them quite a bit more attractive to work with than nonsymmetric matrices.

1.4 Matrix, vector and matrix-vector manipulations

Above, we already looked at matrix-vector multiplication and found that there are two ways to express a multiplication $A\vec{x}$: one in terms of the rows of A and one in terms of the columns. Here, we look at other important matrix-vector and matrix-matrix manipulations that you need throughout linear algebra. You need to be very fluent in these types of manipulations, so take some time to familiarize yourself with them.

1.4.1 Linear combinations

First, the easy stuff. It is not so hard to convince yourself that for any $m \times n$ matrix A, we have $A(\vec{x}+\vec{y}) = A\vec{x} + A\vec{y}$ for any two vectors $\vec{x}$ and $\vec{y}$ and $A(\alpha\vec{x}) = \alpha A\vec{x}$, for any constant α and vector $\vec{x}$. The first, for example, can be seen readily:

$$A(\vec{x}+\vec{y}) = \sum_{j=1}^{n}(x_j + y_j)\vec{a}_j = \sum_{j=1}^{n} x_j\vec{a}_j + \sum_{j=1}^{n} y_j\vec{a}_j = A\vec{x} + A\vec{y},$$

where we used the expression for the matrix-vector product in terms of the columns $\vec{a}_j$ of A derived above.

In general, we have

$$A\left(\sum_{k=1}^{N}\alpha_k\vec{z}_k\right) = \sum_{k=1}^{N}\alpha_k A\vec{z}_k.$$

Addition and subtraction of matrices are also easy. The matrix $A+B$, for example (assuming A and B have the same dimensions) is simply the matrix with elements $a_{ij}+b_{ij}$.

Multiplications like $(A+B)C$ can, as in the scalar case, be written as $(A+B)C = AC + BC$.

So far it works just like with scalars or vectors. Is that always the case? For scalars, we can do things like $(a+b)(a-b) = a^2 - b^2$, which follows because $ab = ba$ always and in the expansion these terms cancel out. But here we have to be very careful with matrices. Suppose, for example, that A is a projection matrix that takes any vector in two dimensions and projects it onto the y-axis. In other words, $A\vec{x} = \begin{bmatrix} 0 \\ x_2 \end{bmatrix}$. Take B the matrix that rotates a vector over $\pi/4$ radians. Can we now also say $AB = BA$? Let's pick a vector and see. We use $\vec{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. We first apply B and then A to $\vec{x}$, in other words, we find $AB\vec{x}$. So we first rotate the vector over 45 degrees and then project. The end result is the vector $\begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix}$. Now we compute $BA\vec{x}$, which means we first project and then rotate. We

get $\begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$. Clearly here $AB \neq BA$. If the matrices had both been rotation matrices, we would have had $AB = BA$. Can you see that? But the result does not apply generally. This means that if we compute something like $(A+B)(A-B) = A^2 - AB + BA - B^2$, we cannot simply say that $-AB + BA = 0$: it depends on the matrices.

When two matrices satisfy $AB = BA$ we say that they *commute*. Remember that this is not a given and must be checked carefully.

1.4.2 Matrix transpose and symmetric matrices

The transpose of a matrix A is the matrix A^T whose rows are the columns of A and whose columns are the rows of A. In other words, if A is the $m \times n$ matrix with elements $a_{ij}, 1 \leq i \leq m, 1 \leq j \leq n$, then A^T is the $n \times m$ matrix

$$A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} & \dots & a_{m1} \\ a_{12} & a_{22} & a_{32} & \dots & a_{m2} \\ \vdots & & \ddots & & \vdots \\ a_{1,n-1} & a_{2,n-1} & \dots & a_{m-1,n-1} & a_{m,n-1} \\ a_{1n} & a_{2n} & \dots & a_{m-1,n} & a_{mn} \end{bmatrix}.$$

If $A = A^T$, it follows that the matrix A is *symmetric*. A symmetric matrix A is necessarily square, and the i-th row of A is exactly equal to the i-th column, for all $1 \leq i \leq n$. If a matrix A satisfies $A = -A^T$, we say that A is *skew-symmetric*.

Both symmetric and skew-symmetric matrices occur often in engineering problems.

I really like symmetric matrices. First of all, only half (and a bit) of the matrix must be stored. Although memory has gotten a lot cheaper than in the days I started programming, it is still a big advantage. But the main reason that these matrices are so appealing is that they have very nice properties as you will see later in these notes. To name a few examples: the eigenvalues of a symmetric matrix are always real and the eigenvectors can always be chosen orthogonal, which implies that a symmetric matrix is always diagonalizable; specialized iterative methods can be designed that are very effective, such as the Conjugate Gradient algorithm; and the sensitivity of matrix computations to round-off or other possible errors is generally easier to analyze. In fact, symmetric matrices make life so much easier in the field of matrix computations that some books discuss only algorithms for symmetric matrices and leave out the typically nastier algorithms for non-symmetric matrices.

1.4.3 Matrix-matrix multiplication

How can we compute the elements of the matrix $C = AB$? That is, how do we actually perform matrix-matrix multiplication? It's relatively easy to understand how if we look at $AB\vec{x}$ first. Here, we first apply B to $\vec{x}$ and then A to the results $B\vec{x}$. Now, we know that

$$B\vec{x} = \sum_{j=1}^{n} x_j \vec{b}_j,$$

where $\vec{b}_j$ are the columns of B. Apply A to $B\vec{x}$ therefore gives

$$AB\vec{x} = \sum_{j=1}^{n} x_j A\vec{b}_j.$$

But

$$AB\vec{x} = \sum_{j=1}^{n} x_j A\vec{b}_j$$

can be written as

$$AB\vec{x} = \left[A\vec{b}_1\ A\vec{b}_2\ \ldots\ A\vec{b}_n\right]\vec{x},$$

which shows that

$$AB = \left[A\vec{b}_1\ A\vec{b}_2\ \ldots\ A\vec{b}_n\right].$$

Taking this a little further, we can look at each individual element of the product $C = AB$. We know that the j-th column of C is given by $\vec{c}_j = A\vec{b}_j$. Then, using what we know about matrix-vector multiplications, the i-th element in this column, c_{ij}, is given by $\vec{r_{A_i}}^T\vec{b}_j$. In other words c_{ij} is the inner product of the i-th row of A with the j-th column of B:

$$C = AB = \begin{bmatrix} \vec{r_{A_1}}^T\vec{b}_1 & \vec{r_{A_1}}^T\vec{b}_2 & \vec{r_{A_1}}^T\vec{b}_3 & \ldots & \vec{r_{A_1}}^T\vec{b}_n \\ \vec{r_{A_2}}^T\vec{b}_1 & \vec{r_{A_2}}^T\vec{b}_2 & \vec{r_{A_2}}^T\vec{b}_3 & \ldots & \vec{r_{A_2}}^T\vec{b}_n \\ \vdots & & \ddots & & \vdots \\ \vec{r_{A_{m-1}}}^T\vec{b}_1 & \vec{r_{A_{m-1}}}^T\vec{b}_2 & \ldots & \vec{r_{A_{m-1}}}^T\vec{b}_{n-1} & \vec{r_{A_{m-1}}}^T\vec{b}_n \\ \vec{r_{A_m}}^T\vec{b}_1 & \vec{r_{A_m}}^T\vec{b}_2 & \ldots & \vec{r_{A_m}}^T\vec{b}_{n-1} & \vec{r_{A_m}}^T\vec{b}_n \end{bmatrix}.$$

Note that these inner products also immediately tell us which size matrices we can actually multiply: the number of rows in B (that is, the number of elements in a column $\vec{b}_i$ of B) must be exactly the same as the number of columns of A (that is, the number of elements in a row $\vec{r_{A_j}}$ of A). We can multiply any $m \times n$ matrix A with any $n \times k$ matrix B, no matter what m, n and k.

We can also write this matrix vector multiplication as

$$\begin{bmatrix} \vec{r_{A1}}^T B \\ \vec{r_{A2}}^T B \\ \vdots \\ \vec{r_{Am}}^T B \end{bmatrix}.$$

Do you see that? To convince yourself, it's always easier to start with a wee example in just a few dimensions.

Pre- and post multiplication

A matrix A can be premultiplied by a matrix B, leading to BA, or post multiplied, leading to AB. Since matrices do not necessarily commute, the end result can be different. It's worth thinking about this a bit more. As example, we take the 2×2 matrices

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Here, B looks like a permuted identity matrix. In fact, we call it a *permutation matrix*. Let's find BA and AB:

$$BA = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix}, \quad AB = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}.$$

BA permutes the *rows* of A and AB permutes the *columns* of A.This is a good thing to remember: *premultiplying by a matrix can be used to manipulate rows, while post multiplying by a matrix can be used to manipulate columns.* We see permutation matrices in the following chapter on Gaussian elimination. In Gaussian elimination we also see matrices designed specifically to manipulate rows. In the chapter on orthogonalization (QR algorithm) we will encounter matrices that are specifically designed to manipulate columns.

1.5 Matrix and vector norms

A norm is a nonnegative measure of the size of an entity. For a scalar, we usually use its absolute value as norm. How do we measure the size of a matrix, or a vector? This section is all about such measures, or norms. The notation used for the norm of a matrix is $\|A\|$, and we use the same for vectors ($\|\vec{x}\|$).

1.5.1 Common vector norms

In these notes, we will use three different kinds of vector norms. The first is the standard Euclidean norm. We distinguish it from the other norms by giving it the subscript 2. We assume that we have a vector $\vec{x}$ with n elements $x_i, i = 1, \ldots, n$. Then, the Euclidean norm is defined as

$$\|\vec{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}.$$

Intuitively, this norm makes sense. It gives the Euclidean distance from the origin to the endpoint of the vector $\vec{x}$ and is a direct consequence of the Pythagorean theorem. Can you see that we can also write this as $\|\vec{x}\|_2 = \sqrt{\vec{x}^T \vec{x}}$?

Sometimes, we are not interested in this Euclidian, or 2-norm, but want to just measure, for example, the size of the largest element of $\vec{x}$. This is done with the *maximum norm* defined as

$$\|\vec{x}\|_\infty = \max_{1 \le i \le n} |x_i|.$$

Another norm that is often used is the 1-norm, defined by

$$\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|.$$

Which of these three norms is the optimal norm to use really depends on the application you are working on. Sometimes norms are related to physical properties. In fluid dynamics, for example, a 2-norm of velocities is linked to kinetic energy. Sometimes, the math works out better in one norm than another. The square of the 2-norm is a quadratic function, which can be differentiated easily, as compared to the 1-norm and infinity norm, which is often attractive. In applications where this is not important, the 1-norm may offer more attractive properties as we describe in the exercises related to the PageRank algorithm later in the book. The 1-norm is sometimes also preferred over the 2-norm as it puts less emphasis on outliers in the data.

1.5.2 Common matrix norms

How do we extend these vector norms to matrices? The simplest thing we could do is to define a straightforward extension of the Euclidean norm, called the *Frobenius norm*

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

Other matrix norms are intuitive when we think of A as an operator. For example, the so-called *induced* 2-*norm* of a matrix is defined as

$$\|A\|_2 = \max_{\vec{x}\neq 0} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2}.$$

This norm measures the maximum amplification of the length of a vector $\vec{x}$ when subjected to the operator A. It's a really nice way to look at a norm of a matrix. Note that this matrix norm is called an induced norm, because it is based on a vector norm.

We can do the same for the maximum norm and the 1-norm. These induced norms can be directly expressed in terms of the elements of A, which is very attractive. We do not show the derivations here, but the end results are that

$$\|A\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|, \tag{1.3}$$

which is just the maximum row sum, and

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|, \tag{1.4}$$

which is the maximum column sum. If you are keen, try to prove relations (1.3) and (1.4). Fun problems.

1.5.3 Important relations

The norms given in the previous subsections were not arbitrarily chosen. To be a valid and useful measure, all norms must satisfy the following important relations:

- $\|A\| \geq 0$, and for vectors $\|\vec{x}\| \geq 0$
- $\|A\| = 0$, if and only if $A = 0$, and for vectors $\|\vec{x}\| = 0$, if and only if $\vec{x} = \vec{0}$
- $\|\alpha A\| = |\alpha|\|A\|$, and for vectors $\|\alpha\vec{x}\| = |\alpha|\|\vec{x}\|$
- $\|A + B\| \leq \|A\| + \|B\|$, and for vectors $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$. This inequality is referred to as the *triangle inequality*
- $\|AB\| \leq \|A\|\|B\|$, and for vectors $|\vec{x}^T\vec{y}| \leq \|\vec{x}\|\ \|\vec{y}\|$. This inequality is referred to as the *Cauchy-Schwarz* inequality.

The first three relations are easy to verify for the norms that we defined above. The triangle inequality and Cauchy-Schwarz are a bit tricker to prove for some norms. Try to

prove them for the 2-norm, for example (proofs can be easily found online for these famous inequalities, but try yourself first).

We can also combine matrix and vector norms, provided that we use the same type of norm for both matrix and vector. So, we'd use $\|\vec{x}\|_1$ together with $\|A\|_1$, and $\|\vec{x}\|_\infty$ with $\|A\|_\infty$. If we use $\|\vec{x}\|_2$, we have a choice: we can either use the induced matrix 2-norm, or the Frobenius norm. With the correct match between vector and matrix norms, we can now also show that

$$\|A\vec{x}\| \leq \|A\|\|\vec{x}\|,$$

which will come in handy. Again, you may try to prove this for the 2-norm, for example. I like such proofs as it really forces me to understand the underlying concepts precisely.

1.6 Matrix inverses

The inverse A^{-1} of a square matrix A, if it exists, is the matrix that satisfies $AA^{-1} = I$ and $A^{-1}A = I$, where I is the identity matrix. Another way of viewing this is that A^{-1} is the matrix that undoes the operation of A, and vice versa. We will discuss inverses in more detail later in the notes, but here are some first thoughts.

The inverse of a matrix does not always exist. For example, the projection matrix $P = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ does not have an inverse. If it had, there would be a matrix C such that $PC = I$ and $CP = I$. But $PC = \begin{bmatrix} c_{11} & c_{12} \\ 0 & 0 \end{bmatrix}$, and we can never get that to equal the identity matrix. Of course, this makes sense: if I project vectors onto a plane, or line, many vectors end up in the same point and there is no way to invert this process.

For some matrices, we can very quickly find the inverse. For example, if the matrix Q is the matrix that rotates a vector over an angle θ, then naturally Q^{-1} should be the matrix that rotates a vector back over an angle $-\theta$.

If a matrix has an inverse, we call the matrix *nonsingular* or *regular*, and if it does not, the matrix is *singular*.

1.6.1 How do we know a matrix is nonsingular?

How do we know if A^{-1} exists? In a later chapter, we will discuss the determinant, which is a scalar property of the matrix and an indicator of singularity, and in the very last chapter of the book you will be introduced to singular values, which can also be used to assess

singularity. Here, we briefly discuss a link between non singularity of a matrix A and the system $A\vec{x} = \vec{b}$.

We can prove that A^{-1} exists if and only if each $\vec{b}$ in $A\vec{x} = \vec{b}$ has a unique solution $\vec{x}$. This is very similar to how inverses of functions are defined in calculus. Do you see that?

I like to give this proof because it also forces us to discuss the significance of an *if and only if* statement. Let's look at that first:

- *if*: The if part of this statements says A^{-1} *exists if each* $\vec{b}$ *has a unique* $\vec{x}$. In other words, starting from the knowledge that each $\vec{b}$ has a unique $\vec{x}$, we should be able to find a matrix A^{-1} that satisfies both $AA^{-1} = I$ and $A^{-1}A = I$.
- *only if*: The only if part of this statement (with the emphasis on *only*) tells us that if each $\vec{b}$ does *not* have a unique $\vec{x}$, the inverse A^{-1} can not exist. Another way to interpret this is that A^{-1} exist, we should be able to show that each $\vec{b}$ has a unique $\vec{x}$.

So, in other words:

- *if*: If each $\vec{b}$ has a unique $\vec{x}$, then A^{-1} exists
- *only if*: If A^{-1} exist, then each $\vec{b}$ has a unique $\vec{x}$.

In general, a claim of the form "statement D if and only if statement E" means that E always implies D *and* D always implies E.

The "only if "part of the proof is straightforward. We know $A\vec{x} = \vec{b}$, and we know that A^{-1} exists and satisfies $A^{-1}A = I$. Multiplying both sides of $A\vec{x} = \vec{b}$ with A^{-1} we find $\vec{x} = A^{-1}\vec{b}$. So, clearly each $\vec{b}$ has an associated $\vec{x}$. How do we know that this must be a unique $\vec{x}$? Any other solution $\vec{x}'$ to $A\vec{x} = \vec{b}$ will also satisfy $\vec{x}' = A^{-1}\vec{b}$.

The "if" part of the proof is a little more involved. How can we show that A^{-1} exists if each $\vec{b}$ has a unique $\vec{x}$? Note that we have to show a matrix A^{-1} exists that satisfies both $AA^{-1} = I$ and $A^{-1}A = I$.

We will start by showing that there is a matrix C that satisfies $AC = I$. We know that a matrix-matrix multiplication AC gives a matrix with columns $A\vec{c}_j, j = 1, \ldots, n$, where $\vec{c}_j$ is the j-th column of C. If $AC = I$, then we must have $A\vec{c}_j = \vec{e}_j$, for all j, where $\vec{e}_j$ is the j-th unit vector. Can we find such vectors $\vec{c}_j$? Sure. We know that for each $\vec{b}$ there is a unique $\vec{x}$ that satisfies $A\vec{x} = \vec{b}$. The equation $A\vec{c}_j = \vec{e}_j$ is exactly in this form. So, we know that each unit vector $\vec{e}_j$ has a corresponding and unique $\vec{c}_j$, and thus we know that a matrix C can be found for which $AC = I$.

Question now is if that same C also satisfies $CA = I$. Let's look at $A(CA) = (AC)A = IA = A$. We set $CA = D$ for ease of notation. $AD = A$ means that for each column $\vec{a}_j$ of

A, we have $A\vec{d}_j = \vec{a}_j$, where $\vec{d}_j$ is the j-th column of D. Again, this is in the form $A\vec{x} = \vec{b}$ and we know that for each column $\vec{a}_j$ there is a unique $\vec{d}_j$ such that $A\vec{d}_j = \vec{a}_j$. But, we also know that $A\vec{e}_j = \vec{a}_j$ as the matrix-vector multiplication $A\vec{e}_j$ picks out exactly the j-th column of A. Since we have uniqueness, this means that each column $\vec{d}_j$ is equal to the j-th unit vector, hence $D = CA = I$. Therefore, we can conclude that $C = A^{-1}$ and we have completed the proof.

OK, this was all a bit involved, but if you get this, then you really understand this material. Or should I be using an if and only if statement here?

1.7 An engineering example: numerical solution of the heat equation

Note: This section requires some background in numerical differentiation.

To illustrate the use of matrices in engineering, we will have a look at the numerical solution of the so-called steady heat equation

$$\frac{d^2T}{dx^2} = f(x), \qquad \text{for } 0 \leq x \leq 1. \tag{1.5}$$

We interpret this equation as modeling the steady state solution of a heat diffusion problem in a rod. T denotes temperature. The rod is one meter long and has constant thermal conductivity. The second order derivative in x models heat diffusion in the x-direction. Since no heat diffusion term is given for other coordinate directions, we conclude that the rod here is laterally insulated. The function $f(x)$ is a distributed heat source $f(x)$. To solve this equation, two boundary conditions are needed. We will set the temperatures at either end of the rod to a fixed value, that is, $T(0) = a$ and $T(1) = b$ are given.

For simple heat sources $f(x)$, the above differential equation may be straightforward to integrate. If we take $f(x) = 0$ for example, the answer will simply be

$$T(x) = a(1 - x) + bx,$$

a straight line between the boundary values. For complex $f(x)$, integration may be non-trivial and an exact analytical solution for $T(x)$ may not readily be found. In that case, we compromise: rather than asking for the exact solution $T(x)$ in all possible locations x, we are happy knowing, with decent accuracy, the solutions in a finite set of locations

$$x_i, \quad i = 0, \ldots, N, \text{ with } x_0 = 0, x_N = 1.$$

For simplicity we will here assume that the distance h between the points is fixed, that is

$$h = x_i - x_{i-1} = \frac{1}{N}.$$

The temperature values in the points x_i are denoted by T_i.

How do we find the desired T_i's? We know that the exact solution $T(x)$ satisfies (1.5). We now require the approximate solutions T_i to satisfy a discrete approximation to this differential equation. From numerical analysis classes, you may remember that the second order derivative $\frac{d^2T}{dx^2}$ in x_i can be approximated as

$$\frac{d^2T(x_i)}{dx^2} \approx \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2}. \tag{1.6}$$

There are various ways to derive this approximation. Perhaps the easiest one is to look at Taylor series expansions of T_{i+1} and T_{i-1} about the point x_i:

$$\begin{aligned} T_{i+1} &= T_i + h\frac{dT(x_i)}{dx} + \frac{h^2}{2!}\frac{d^2T(x_i)}{dx^2} + \frac{h^3}{3!}\frac{d^3T(x_i)}{dx^3} + \frac{h^4}{4!}\frac{d^4T(x_i)}{dx^4} + \text{h.o.t.}, \\ T_{i-1} &= T_i - h\frac{dT(x_i)}{dx} + \frac{h^2}{2!}\frac{d^2T(x_i)}{dx^2} - \frac{h^3}{3!}\frac{d^3T(x_i)}{dx^3} + \frac{h^4}{4!}\frac{d^4T(x_i)}{dx^4} + \text{h.o.t.}. \end{aligned}$$

Here, the abbreviation h.o.t. stands for higher order terms, that is, it contains all the remaining terms in the Taylor series that all have higher orders (higher than 4) in h as well as the derivatives.

Adding these two Taylor series together gives

$$T_{i+1} + T_{i-1} = 2T_i + h^2\frac{d^2T(x_i)}{dx^2} + \frac{h^4}{12}\frac{d^4T(x_i)}{dx^4} + \text{h.o.t.},$$

where the higher order terms here contain $h^6, h^8, \cdots$ and corresponding derivatives.

This equation can be rewritten in terms of the second order derivative as

$$\frac{d^2T(x_i)}{dx^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2} - \frac{h^2}{12}\frac{d^4T(x_i)}{dx^4} + \text{h.o.t.},$$

and clearly (1.6) is a reasonable approximation as long as h is sufficiently small. For $h \ll 1$ the error, which contains an infinite number of terms of the form $Ch^{2p-2}d^{2p}T/dx^{2p}$ with constant C and $p \geq 2$, is quite small. The leading error term is $\frac{h^2}{12}\frac{d^4T(x_i)}{dx^4}$ and we call the scheme a second order discretization because the leading error term depends on h^2.

We now plug the approximation into (1.5) and get the equations

$$T_{i+1} - 2T_i + T_{i-1} = h^2 f_i, \quad i = 1, \cdots, N-1,$$

with $T_0 = a$ and $T_N = b$ given by the boundary conditions. We have $N-1$ equations for the $N-1$ unknowns $T_i, i = 1, \cdots, N-1$.

For clarity, let's write this system out for a few cases. For $i = 1$, we get

$$-2T_1 + T_2 = h^2 f_1 - a,$$

and for $i = 2$, we have

$$T_1 - 2T_2 + T_3 = h^2 f_2.$$

For $i = N - 1$, we obtain

$$T_{N-2} - 2T_{N-1} = h^2 f_{N-1} - b.$$

In matrix-vector notation, we write $A\vec{T} = \vec{c}$, with

$$A = \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \vdots & & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -2 \end{bmatrix}, \quad \vec{T} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{N-2} \\ T_{N-1} \end{bmatrix}, \quad \vec{c} = \begin{bmatrix} h^2 f_1 - a \\ h^2 f_2 \\ h^2 f_3 \\ \vdots \\ h^2 f_{N-2} \\ h^2 f_{N-1} - b \end{bmatrix}. \tag{1.7}$$

The resulting matrix is a tridiagonal matrix. It is also symmetric. In later chapters, we will work with this system of equations and solve it for specific cases.

1.8 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled* $\star$ *are more challenging.*

Exercise 1.1

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

(a) If $A^2 + A = I$ then $A^{-1} = I + A$

(b) If all diagonal entries of A are zero, then A is singular (not invertible).

(c) $\|A\|_F^2 = \text{tr}(A^T A)$. Here $\text{tr}(A^T A)$ is the *trace* of $A^T A$. The trace of a matrix is the sum of its diagonal elements.

Exercise 1.2

The product of two n by n lower triangular matrices is again lower triangular (all its entries above the main diagonal are zero). Prove this in general and confirm it with a 3×3 example.

Exercise 1.3

If $A = A^T$ and $B = B^T$, which of these matrices are certainly symmetric?

(a) $A^2 - B^2$

(b) $(A + B)(A - B)$

(c) ABA

(d) $ABAB$

Exercise 1.4

A *skew-symmetric* matrix is a matrix that satisfies $A^T = -A$. Prove that if A is $n \times n$ and skew-symmetric, than for any $\vec{x}$ we must have that $\vec{x}^T A\vec{x} = 0$.

Exercise 1.5

Suppose A is an $n \times n$ invertible matrix, and you exchange its first two rows to create a new matrix B. Is the new matrix B necessarily invertible? If so, how could you find B^{-1} from A^{-1}? If not, why not?

Exercise 1.6

Let A be an invertible n-by-n matrix. Prove that A^m is also invertible and that

$$(A^m)^{-1} = \left(A^{-1}\right)^m$$

for $m = 1, 2, 3, \ldots$

Exercise 1.7

Let A be a 2×2 matrix $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ with $a_{11} \neq 0$ and let $\alpha = a_{21}/a_{11}$. Show that A can be factored into a product of the form

$$\begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ 0 & b \end{pmatrix}$$

What is the value of b?

* Exercise 1.8

(a) Show that if the $n \times n$ matrices A, B and $A + B$ are invertible, then the matrix $B^{-1} + A^{-1}$ is also invertible.

(b) Assume that C is a skew-symmetric matrix and that D is a matrix defined as

$$D = (I + C)(I - C)^{-1}$$

Prove that $D^T D = DD^T = I$.

* Exercise 1.9

Given an $n \times n$ matrix A with column vectors $\vec{a_1}, \vec{a_2}, \ldots, \vec{a_n}$, construct a matrix B such that the matrix AB has the columns $\vec{u_1}, \vec{u_2}, \ldots, \vec{u_n}$ with the following properties:

(a) $\vec{u_i} = \vec{a_i}, \quad i \neq j$

(b) $\vec{u_j} = \sum_{k=1}^{j} \alpha_k \vec{a_k}$,

where j is a *fixed* integer such that $1 \leq j \leq n$.

Exercise 1.10

The well-known Fibonacci number $0, 1, 1, 2, 3, 4, \ldots$ can be computed using the formula:

$$F_{n+2} = F_{n+1} + F_n, \quad F_0 = 0, \quad F_1 = 1.$$

Show that for $n > 1$:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} F_{n-1} \\ F_n \end{bmatrix}.$$

Exercise 1.11

If $A = LU$, where L is lower triangular with ones on the diagonal and U is upper triangular, show that L and U are unique.

** Exercise 1.12

For any $n \times n$ matrix A. Show that the 2-norm and the Frobenius norm of the matrix satisfy

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

Exercise 1.13

The matrix A is given by the factorization

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix}$$

Show (**without** multiplying these two factors) that A is invertible, symmetric, tridiagonal, and positive definite.

* Exercise 1.14

Suppose that the $n \times n$ matrix A has the property that the elements in each of its rows sum to 1. An example of such matrix is $\begin{bmatrix} 1/3 & 0 & 2/3 \\ 0 & 1 & 0 \\ 1/2 & 1/2 & 0 \end{bmatrix}$. Show that for any such $n \times n$ matrix A, the matrix $A - I$ (with I the $n \times n$ identity matrix) is singular.

* Exercise 1.15

(a) If $n \times n$ real matrix A is such that $A^2 = -I$, show that n must be even.

(b) If the $n \times n$ real matrix A is such that $A^2 = I$, show that A cannot be skew-symmetric.

Exercise 1.16

We define the 1-norm of a vector $\vec{x}$ as $\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|$, i.e. the sum of the magnitudes of the entries. The 1-norm of an $m \times n$ matrix A is defined as $\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|$, i.e. the largest column sum. It can be shown that for these norms, $\|A\vec{x}\|_1 \le \|A\|_1 \|\vec{x}\|_1$ (we will not prove this here, but you can use it as given, whenever needed). Compute the 1-norm of the following matrices and vectors

(a) $\vec{x} = (1, 2, 3, \ldots, n)^T$.

(b) $\vec{x} = \left(\frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)^T$.

(c) αI where I is the $n \times n$ identity matrix.

(d) $J - I$ where J is the $n \times n$ matrix filled with 1s and I is the $n \times n$ identity.

(e) $A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \\ & & & -1 & 2 \end{bmatrix}$, where A is an $n \times n$ matrix.

(f) $A = \begin{bmatrix} 1/n & \cdots & 1/n \\ \vdots & \ddots & \vdots \\ 1/n & \cdots & 1/n \end{bmatrix}$, where A is an $n \times n$ matrix.

Chapter 2

Solving matrix-vector equations with Gaussian Elimination

In this chapter, we will explain how matrix-vector equations $A\vec{x} = \vec{b}$ can be solved using a process called Gaussian Elimination. We'll start by explaining the general procedure, first on a small example system and then in general form. Then, we will formalize the procedure using the so-called LU-decomposition.

Solving $A\vec{x} = \vec{b}$ with Gaussian Elimination is a *direct* method: i we did not make any round-off errors during the operations involved in Gaussian Elimination, the solution $\vec{x}$ would be found exactly. This is in contrast to iterative method where an approximation to the solution $\vec{x}$ is found. Iterative methods are discussed in Chapter 7.

2.1 Elimination and back-substitution

Suppose that we are interested in finding the solution to the small system of equations

$$\begin{aligned} (1) \quad & 2x_1 + x_2 = 1 \\ (2) \quad & -2x_1 + 2x_2 = 2 \end{aligned}$$

You may remember from calculus that to solve this system, you could add the two equations together, which gives you $3x_2 = 3$, or $x_2 = 1$. We say that we *eliminated* x_1 from the system through this equation manipulation (the addition). Then, our next step would be to take $x_2 = 1$, plug it, for example, into equation 1, which gives $2x_1 + 1 = 1$, so $x_1 = 0$.

Typically with small systems like this, we stare at the equations for a while to figure out which manipulations will give us the solution easiest. We could, for example, also have

multiplied the first equation by 2 and then subtracted it from the second. Adding the equations seemed simpler.

As the size of the system grows, it gets harder and harder to just stare at the system and figure out what the best order is to manipulate the equations and successively eliminate unknowns. Therefore, we simply agree on the order in which we do it, and always do it this same way. Of course, fixing the order of manipulations and elimination also makes it a lot simpler to write a computer code for the algorithm that can be used for any matrix.

2.1.1 Gaussian Elimination: an agreed order of elimination

So, what is this order? Let's look at a slightly larger example:

$$\begin{aligned} (1) \quad & 2x_1 + x_2 + x_3 = 5 \\ (2) \quad & 4x_1 - 6x_2 + 3x_3 = 4 \\ (3) \quad & -2x_1 + 7x_2 + 2x_3 = 9 \end{aligned}$$

We now go through the following steps:

Step 1: Eliminate x_1 from equations (2) and (3) using equation (1)

To eliminate x_1 from equation 2 we multiply equation 1 by a factor of 2 and subtract it from equation 2. How do we get the factor 2? It is simply the coefficient 4 in front of x_1 in equation (2) divided by the coefficient 2 in front of x_1 in equation (1). In other words, equation (2) is replaced by the following expression

$$\text{equation (2)} - \frac{4}{2}\,\text{equation (1)} = \text{equation (2)} - 2\,\text{equation (1)}.$$

Following a similar procedure to eliminate x_1 from equation 3, we replace equation 3 by

$$\text{equation (3)} - \frac{-2}{2}\,\text{equation (1)} = \text{equation (3)} + \text{equation (1)}.$$

The new system of equation is given by

$$\begin{aligned} (1) \quad & 2x_1 + x_2 + x_3 = 5 \\ (2') \quad & -8x_2 + x_3 = -6 \\ (3') \quad & 8x_2 + 3x_3 = 14 \end{aligned}$$

Step 2: Eliminate x_2 from equation (3′) using equation (2′)

Our next step is to eliminate x_2 from equation $(3')$. We leave equations (1) and $(2')$ as is. To eliminate x_2 from equation $(3')$, we *must* use equation $(2')$: if we used equation $(1')$ instead, it would reintroduce x_1 in equation $(3')$. The elimination is similar to the eliminations in step 1. We replace equation $(3')$ by

$$\text{equation } (3') - \frac{8}{-8} \text{ equation } (2') = \text{equation } (3') + \text{equation } (2').$$

The resulting system is

$$\begin{array}{lrrrr} (1) & 2x_1 + & x_2 + & x_3 = & 5 \\ (2') & & -8x_2 + & x_3 = & -6 \\ (3'') & & & 4x_3 = & 8 \end{array}$$

Step 3: Use back substitution to find x_3, x_2 and then x_1

The new system is easy to solve. Equation $(3'')$ immediately gives $x_3 = 2$. Now that we have x_3, we can plug it in equation $(2')$ to find $-8x_2 = -8$, or $x_2 = 1$. Finally, we can plug the known values for x_3 and x_2 into equation (1) to find $x_1 = 1$. This process is called *back substitution*.

2.1.2 Augmented matrices

The original system can of course also be written as a matrix-vector equation

$$\begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 3 \\ -2 & 7 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 9 \end{bmatrix},$$

or $A\vec{x} = \vec{b}$.

In matrix-vector notation, the system we found after the elimination process is given by

$$\begin{bmatrix} 2 & 1 & 1 \\ 0 & -8 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ -6 \\ 8 \end{bmatrix},$$

or $U\vec{x} = \vec{c}$, where U is an upper triangular matrix. Solving the system $U\vec{x} = \vec{c}$ is easy because we can use back substitution, starting with the last row and working our way backwards.

So, Gaussian elimination can be seen as taking a matrix-vector system $A\vec{x} = \vec{b}$ and transforming it into an upper-triangular system $U\vec{x} = \vec{c}$. Note that the right hand side vector $\vec{b}$ also changes during Gaussian Elimination.

In the first step of Gaussian elimination, we eliminated x_1 from equations (2) and (3). In terms of the matrix A this means that we zeroed out all elements below the diagonal in the first column of A. In the second step, we also zero out the second column below the diagonal. This was equivalent to eliminating x_2 from equation (3′). After these two steps, the matrix is zero below the diagonal and we obtain an upper triangular matrix.

We don't have to perform Gaussian elimination on the system of equations. We can do it directly on the matrix, and the vector $\vec{b}$. To do this, we typically use the so-called augmented matrix $\left[A\ \vec{b}\right]$, or

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 4 & -6 & 3 & 4 \\ -2 & 7 & 2 & 9 \end{bmatrix},$$

Let's go through Gaussian Elimination again but now using this augmented matrix.

Step 1: Zero out the first column of A under the diagonal using row (1)

In step 1 of Gaussian Elimination, we remove x_1 from equations (2) and (3). In the matrix-notation, this means that we must zero out the first column below the diagonal. Similar to the equation manipulations above, we replace row (2) of the augmented matrix (so including the right hand side vector $\vec{b}$ with

$$\text{row } (2') = (\text{row } (2) - \frac{4}{2} \text{ row } (1))$$

and row 3 with

$$\text{row } (3') = (\text{row } (3) - \frac{-2}{2} \text{ row } (1).$$

Notice that both of the multiplication factors have 2 in the denominator, the element a_{11} in A. We refer to this element also as *pivot 1*. We now get the following augmented matrix $[A'\ \vec{b}']$.

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & 1 & -6 \\ 0 & 8 & 3 & 14 \end{bmatrix},$$

Step 2: Zero out the second column of A' under the diagonal

In step 2 of Gaussian Elimination, we remove x_2 from equation (3′). In the matrix-notation, this means that we must zero out the second column below the diagonal in A'. Similar to the equation manipulations above, we replace row (3′) of the augmented matrix with

$$\text{row } (3') - \frac{8}{-8} \text{ row } (2') = \text{row } (3') + \text{row } (2').$$

The multiplication factor has -8 in the denominator, the element a'_{22}, which we refer to as *pivot 2*. The new augmented matrix $\left[A''\ \vec{b}''\right]$ is

$$\begin{bmatrix} 2 & 1 & 1 & 5 \\ 0 & -8 & 1 & -6 \\ 0 & 0 & 4 & 8 \end{bmatrix}.$$

This augmented matrix represents the system of equations $U\vec{x} = \vec{c}$, where U is upper triangular (and equal to A'') and $\vec{c} = \vec{b}''$. The element $a''_{33} = u_{33}$ is referred to as *pivot 3*.

2.1.3 What's up with these pivots

Above, we defined the so-called pivots. Why the emphasis on these pivots? Well, if Gaussian Elimination encounters a zero pivot during the process, it breaks down as the corresponding multiplication factor is undefined. Of course, in the example, it would not matter if pivot 3 is zero as that final pivot is not used in subsequent multiplication factors.

What can we do if we encounter a troublesome zero pivot? Let's look at an example matrix

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 4 & 6 & 8 \end{bmatrix}$$

After step 1 of Gaussian Elimination, we get the matrix B' given by

$$B' = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 3 \\ 0 & 2 & 4 \end{bmatrix}$$

and we see that pivot 2 is equal to zero. We therefore can not continue as before with the elimination process. But here, the problem is solvable. We can simply swap rows $(2')$ and $(3')$. Remember that these rows represent equations, and the order in which we list equations has no impact at all on the solution. In other words, swapping is allowed (as long as we swap corresponding elements in the right hand side vector $\vec{b}'$ of course). After the swap, the matrix is equal to

$$B'_s = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{bmatrix},$$

and this is already in upper triangular form, so we are done.

We can not always recover from a zero pivot as easily as this. Look, for example at the matrix D given by

$$D = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 4 & 4 & 8 \end{bmatrix}.$$

After one step of the Gaussian Elimination process we get

$$D' = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 3 \\ 0 & 0 & 4 \end{bmatrix}.$$

Now, we cannot swap to restore the pivot. It does not seem to matter though as D' is already in upper triangular form, and therefore $U = D'$. However, it is clear that U is singular. This means that A is singular also and we will not be able to find a unique solution to $A\vec{x} = \vec{b}$. We will take a closer look.

After Gaussian Elimination we obtain the system $U\vec{x} = \vec{c}$. Here, $\vec{c} = [c_1,\ c_2,\ c_3]^T$. Note that $\vec{c}$ is a column vector. We write it as the transpose of a row vector to save vertical space. This is often done in linear algebra books and papers, so I thought I'd throw it in here to get you used to it.

Now, the second row of U gives the equation $3x_3 = c_2$, and the third row gives $4x_3 = c_3$. We can not solve these equations unless $c_2/3 = c_3/4$. If not, we have to conclude that no solution exists. If $c_2/3 = c_3/4$, we have many possible solutions. This is typical: if A is singular, then the system $A\vec{x} = \vec{b}$ will either have no solution or an infinite number of solutions. We will come back to this later in the book.

Finally, let's see what happens if the last pivot is equal to zero. For example, suppose that after Gaussian Elimination of some 3×3 system $A\vec{x} = \vec{b}$, we end up with $U\vec{x} = \vec{c}$, where

$$U = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}.$$

Naturally, c_3 should be zero for this system to be feasible. If it is not, we can not find any solutions to $U\vec{x} = \vec{c}$ and so also the original system $A\vec{x} = \vec{b}$ is unsolvable. If $c_3 = 0$, the equation represented by the third row is just not useful: it just states $0 = 0$. In other words, we really only have two equations for the three unknowns x_1, x_2 and x_3. This means that we will have an infinite number of solutions. We conclude that the matrix A is not invertible.

What does it really mean if we get a zero row during Gaussian Elimination? It means that that row was just a linear combination of the rows above it. Do you see that? Remember

that in Gaussian Elimination we are looking to zero out elements of a row, column by column. If row k turns into a zero row, we know that row k was equal to some linear combination of rows 1 through $k-1$ that we subtracted from row k in the Gaussian Elimination process. In turn this means that the equation corresponding to row k is a redundant equation, or a contradicting equation, depending on the elements in the right hand side vector as we discussed above.

2.1.4 Gaussian Elimination for a general matrix A

Because of the strict order in which we agree to eliminate unknowns, Gaussian Elimination can be easily extended to any $m \times n$ matrix A. Note that A does not have to be square. We take

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & & \ddots & & \vdots \\ a_{m-1,1} & a_{m-1,2} & \dots & a_{m-1,n-1} & a_{m-1,n} \\ a_{m1} & a_{m2} & \dots & a_{m,n-1} & a_{mn} \end{bmatrix}.$$

We start the Gaussian Elimination process by zeroing out all elements in the first column of A below the diagonal. We first check the pivot. In this case we have pivot $1 = a_{11}$. If the pivot is zero, we look for a row (k) with a nonzero element a_{k1} to swap with the first row before we proceed. Then, we replace rows (i), $i \geq 2$ with

$$\text{row (i}') = \text{row (i)} - \frac{a_{i1}}{\text{pivot 1}} \text{row (1)}.$$

We obtain the matrix

$$A' = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a'_{22} & a'_{23} & \dots & a'_{2n} \\ \vdots & & \ddots & & \vdots \\ 0 & a'_{m-1,2} & \dots & a'_{m-1,n-1} & a'_{m-1,n} \\ 0 & a'_{m2} & \dots & a'_{m,n-1} & a'_{mn} \end{bmatrix}.$$

We now take the $(m-1) \times (n-1)$ sub matrix

$$A'[2\!:\!m, 2\!:\!n] = \begin{bmatrix} a'_{22} & a'_{23} & \dots & a'_{2n} \\ \vdots & & \ddots & \vdots \\ a'_{m-1,2} & \dots & a'_{m-1,n-1} & a'_{m-1,n} \\ a'_{m2} & \dots & a'_{m,n-1} & a'_{mn} \end{bmatrix},$$

and apply Gaussian Elimination to it in the same way as above. We repeat this process until all elements below the diagonal are 0.

The only tricky part of Gaussian Elimination really is the row swapping when we encounter a zero pivot. This is often also referred to as *pivoting*. It would be very nice if we knew in advance which rows to swap to avoid any zero pivots. We can then perform all row swaps first and apply Gaussian Elimination without the pivot checks. Such look-ahead algorithms have been designed and are used in many software packages. We may also want to swap rows when a pivot is nonzero but very small compared to other elements of the matrix to avoid large round-off errors (see Chapter 3). This is referred to as *partial pivoting*. Also, instead of row pivoting, we could use column pivoting. Column pivoting (or column swapping) just changes the order of the unknowns. Again, this does not harm the underlying system of equations as long as we remember that we swapped.

2.1.5 Gaussian Elimination on a rectangular matrix

We finish this chapter with a quick example of Gaussian Elimination applied to $m \times n$ matrices with $m \neq n$. First, we look at a case where $m < n$. Let

$$A = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 2 & 2 & 4 & 2 \\ 0 & 1 & 3 & 1 \end{bmatrix}$$

Our first pivot is 2, so we can proceed. After one step of Gaussian Elimination we have

$$A' = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 3 & 1 \end{bmatrix}.$$

After the second step of Gaussian Elimination we get

$$A' = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 2 & -1 \end{bmatrix},$$

and we are done as all elements in the matrix below the diagonal are now zero. You may find it a bit funny to talk about a diagonal when we have a rectangular matrix, but we simply always define it as the set of elements a_{ii}, where $i = 1, \ldots, \min(m, n)$.

The next example is a case where $m > n$. Let

$$A = \begin{bmatrix} 2 & 1 \\ 2 & 1 \\ 0 & 1 \\ 4 & 2 \end{bmatrix}$$

Our first pivot is 2 so we can proceed. After one step of Gaussian Elimination we have

$$A' = \begin{bmatrix} 2 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Our second pivot (a'_{22}) is 0, and so we must first perform a row swap. Scanning down the second column for a nonzero, we see that we could swap rows 2 and 3. We get

$$A' = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

and we are done as all elements below the diagonal are already zero.

2.2 LU Decomposition

So far, we have always written the operations in Gaussian Elimination as equation, or row, manipulations. Is it also possible to write Gaussian Elimination as a matrix operation? The answer is yes and in this section we discuss how this can be done and how it leads to the well-known LU decomposition of a matrix. We will assume that the matrix under consideration is a square $n \times n$ matrix. In the first subsection we will also assume that pivoting is not required.

2.2.1 In search of L

In the first step of Gaussian Elimination of an $m \times n$ matrix A we replace row (i), $i = 2, \ldots, m$ with

$$\text{row (i}') = \text{row (i)} - \frac{a_{i1}}{\text{pivot 1}} \text{ row (1)}. \tag{2.1}$$

We now look for a matrix C_1 such that these row operations can be written as $C_1 A$. We use premultiplication here because that leads to manipulations of the rows of A, as we discussed in the previous chapter.

What does this C_1 look like? Recall that the element in the i-th and j-th column of $C_1 A$ is given by $\vec{r}_i^T \vec{a}_j$, where $\vec{r}_i^T$ is the i-th row of C_1.

The first row of A should not be affected by C_1, that is, the first row of C should be equal to

$$\vec{r_{C_1}} = [1, 0, 0, \ldots, 0].$$

In other words, the inner product of $\vec{r_{C1}}$ and each column $\vec{a}_j = [a_{1j}, a_{2j}, \ldots, a_{mj}]^T$ should just pick out the first element of this column.

How about the second row? From equation 2.1, we know that element a'_{2j} is given by

$$a'_{2j} = a_{2j} - \frac{a_{21}}{a_{11}} a_{1j} = \left(-\frac{a_{21}}{a_{11}}\right) a_{1j} + 1\, a_{2j} + 0\, a_{3j} + \ldots 0\, a_{mj}.$$

This must be equal to $\vec{r_{C2}}^T \vec{a}_j$, so $\vec{r_{C2}}^T$ must be given by

$$\vec{r_{C2}}^T = [-\frac{a_{21}}{a_{11}}, 1, 0, \ldots, 0].$$

Using a very similar argument, we can see that

$$\vec{r_{C3}}^T = [-\frac{a_{31}}{a_{11}}, 0, 1, 0, \ldots, 0],$$

and so on, until we find

$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & 0 & 0 & \ldots & 0 \\ -\frac{a_{31}}{a_{11}} & 0 & 1 & 0 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ -\frac{a_{n-1,1}}{a_{11}} & 0 & \ldots & 0 & 1 & 0 \\ -\frac{a_{n1}}{a_{11}} & 0 & \ldots & 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

Thus, the first step of Gaussian Elimination can simply we written as the operation $A' = C_1 A$, with C_1 given by equation 2.2.

Following the same arguments, the next step in Gaussian Elimination can be written as $A'' = C_2 A' = C_2 C_1 A$, where

$$C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & -\frac{a'_{32}}{a'_{22}} & 1 & 0 & 0 & \ldots & 0 \\ 0 & -\frac{a'_{42}}{a'_{22}} & 0 & 1 & 0 & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \\ 0 & -\frac{a'_{n-1,2}}{a'_{22}} & 0 & & 0 & 1 & 0 \\ 0 & -\frac{a'_{n2}}{a'_{22}} & 0 & \ldots & 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

Note that the column with multiplication factors has shifted to the right, and that now the first two rows of C_2 are the same as in the identity matrix. This pattern will repeat itself for each step in Gaussian Elimination. Also note that for an $n \times n$ matrix without pivoting we have $n-1$ elimination steps. At the end, we will have transformed A into the upper triangular matrix U which is obtained through a sequence of matrix operations:

$$U = C_{n-1}C_{n-2}\dots C_2C_1A. \tag{2.4}$$

Now, we can multiply both sides with the inverses of the C_i matrices to get

$$A = C_1^{-1}C_2^{-1}\dots C_{n-2}^{-1}C_{n-1}^{-1}U.$$

These inverses exist and that is clear from the structure of the matrices (can you convince yourself?).

You may wonder where we are going with this. After all, the expression 2.7 does not look all that attractive. But, there are two very nice things about these matrices C_k^{-1}. First, they are extremely easy to find. You may not immediately see it from the expressions 2.2 and 2.3, but it becomes clear when you think about the C_k as operators. Each matrix C_k takes rows and *subtracts* a constant times another row. For example, C_1 takes each row (i) with $2 \le i \le n$ and *subtracts* a constant times row (1). Since the inverse of C_1 should undo whatever C_1 does, it should take each row (i) and *add back* that same constant times row (1). In other words,

$$C_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ \frac{a_{21}}{a_{11}} & 1 & 0 & 0 & \dots & 0 \\ \frac{a_{31}}{a_{11}} & 0 & 1 & 0 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \frac{a_{n-1,1}}{a_{11}} & 0 & \dots & 0 & 1 & 0 \\ \frac{a_{n1}}{a_{11}} & 0 & \dots & 0 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

The only difference between C_1 and its inverse C_1^{-1} is the sign in the first column below the diagonal. The same simple change holds for C_2^{-1} and all others.

The second nice thing about these inverses is that when multiplying them out as in $A = C_1^{-1}C_2^{-1}\dots C_{n-2}^{-1}C_{n-1}^{-1}U$, the result is very straightforward. Check for yourself, for example,

that

$$C_1^{-1}C_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ \frac{a_{21}}{a_{11}} & 1 & 0 & 0 & \dots & 0 \\ \frac{a_{31}}{a_{11}} & \frac{a'_{32}}{a'_{22}} & 1 & 0 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \frac{a_{n-1,1}}{a_{11}} & \frac{a'_{n-1,2}}{a'_{22}} & \dots & 0 & 1 & 0 \\ \frac{a_{n1}}{a_{11}} & \frac{a'_{n2}}{a'_{22}} & \dots & 0 & 0 & 1 \end{bmatrix}. \tag{2.6}$$

The multiplication factors are preserved at their original locations in the matrix!!! This is quite wonderful. Note that when we multiply C_2C_1, the result is not so tidy at all: these matrices do not commute. Check this as well.

We can continue these multiplications to form $C_1^{-1}C_2^{-1}\dots C_{n-2}^{-1}C_{n-1}^{-1}$ and at each step, we will see the same thing: the multiplication factors retain their original positions. Because the inverse of a lower triangular matrix is again lower triangular and the product of lower triangular matrices is also lower triangular, we know that the resulting matrix, which we will call L, will be lower triangular also. So,

$$A = C_1^{-1}C_2^{-1}\dots C_{n-2}^{-1}C_{n-1}^{-1}U = LU,$$

and we have our LU decomposition. Keep in mind here that L always has 1s on the diagonal. We call L a *unit lower triangular matrix*. The diagonal of U contains the pivots.

2.2.2 LU decomposition with pivoting

We have seen in the previous subsection that when pivoting is not required we can find the LU decomposition of a matrix A. What changes when pivoting is necessary? Let's look at a very general case

$$U = C_{n-1}P_{n-1}C_{n-2}P_{n-2}\dots P_3C_2P_2C_1P_1A, \tag{2.7}$$

where P_k is a permutation matrix (and may be the identity). We saw permutation matrices in the previous chapter and we know that permutation matrices generally do not commute with other matrices. That means that we cannot pull all the different P_is together. However, if we somehow figured out before the Gaussian Elimination process what all necessary permutations would need to be, we could do the permutations before we started GE. Then we'd get $PA = LU$, where P is the permutation matrix that takes care of all necessary row swaps.

2.2.3 LDLT and Cholesky

When A is a symmetric matrix, the LU decomposition is typically written differently. Since A is symmetric, it is clear that the strictly upper triangular part of U must be symmetric to the strictly lower triangular part of L. We cannot write $U = L^T$, however, because the diagonals of L and U are not the same: L is a unit lower triangular matrix, with all ones on the diagonal, and U has the pivots on the diagonal. The idea is now to take the pivots out and store them in a separate diagonal matrix D. Then, U can be written as $U = DL^T$, and the LU decomposition is $A = LDL^T$. This is simply referred to as the LDL^T (pronounce LDLT) decomposition.

If all the diagonal elements are nonnegative, we can also write this as $A = LD^{1/2}D^{1/2}L^T$, or $A = (LD^{1/2})(LD^{1/2})^T = L_1L_1^T$, where $L_1 = LD^{1/2}$ is a lower triangular (but not unit lower triangular) matrix. This is referred to as the *Cholesky* decomposition and you may see this in the literature. In practice, I tend to use the LDLT decomposition so that I do not have to compute square roots.

2.3 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled ⋆ are more challenging.*

Exercise 2.1

Compute the LU decomposition without pivoting (that is, without row exchanges) of the 4×4 matrix

$$A = \begin{bmatrix} 4 & 2 & -1 & 4 \\ 3 & 2 & 1 & 1 \\ -1 & 2 & 4 & 3 \\ 1 & 3 & -2 & 4 \end{bmatrix}.$$

Clearly show the intermediate matrices A', A'', A''', and C_1, C_2, C_3.

Exercise 2.2

Solve $A\vec{x} = \vec{b}$ for the matrix from problem 2.1 and $\vec{b} = \begin{bmatrix} 1 \\ 1 \\ 6 \\ 13 \end{bmatrix}$.

Exercise 2.3

Compute the LU decomposition of the matrix from problem 2.1 using partial pivoting. Partial pivoting is a strategy in which rows are swapped to create the largest possible pivot at each stage of Gaussian Elimination.

* Exercise 2.4

The minor diagonal of a matrix is the diagonal from the lower left corner of the matrix to the upper right corner. If a matrix has only zeros above this diagonal (so in the left corner of the matrix), the matrix is called a *reverse lower triangular* matrix. A 4×4 example is the matrix

$$R = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 3 \\ 0 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 \end{bmatrix}.$$

Let T be an arbitrary $n \times n$ reverse lower triangular matrix with non-zero elements on its minor diagonal ($n > 1$).

(a) Prove that T of this form is non-singular (prove this for any n).

(b) Explain why the LU factorization of T (for any n) can **not** be found if row exchanges are not permitted. This shows that even though a matrix is non-singular, a LU decomposition without row re-ordering may not exist.

(c) Prove, for any n, that if row exchanges are permitted, the LU factorization of PT can be computed, where P is the permutation matrix that performs the appropriate row exchanges. This shows that, with pivoting, it is always possible to find LU for a re-ordered form of the matrix. Note that perhaps the simplest way of proving this is to actually find the LU factorization.

Exercise 2.5

MATLAB Exercise. We will solve 1D heat equation, discussed in section 1.7, numerically for different boundary conditions. Set the source term $f(x) = 0$ with boundary conditions $T(0) = 0$ and $T(1) = 2$.

(a) Give the exact solution to this differential equation.

(b) Discretize the 1D equation using the second order discretization scheme with $h = 1/N$. Write an expression for each equation in the system.

(c) Formulate the resulting matrix-vector equation $A\vec{T} = \vec{c}$ for $N = 5$, and use Gaussian Elimination (by hand) to check that the matrix A is nonsingular.

(d) Now set the source $f(x) = -10\sin(3\pi x/2)$. Keep the boundary conditions as before.

  (i) Verify that $T(x) = (2 + 40/(9\pi^2))x + 40/(9\pi^2)\sin(3\pi x/2)$ is the exact solution that solves the differential equation with this source and boundary conditions.

  (ii) For this source function, solve the system $A\vec{T} = \vec{c}$ using MATLAB for $N = 5$, $N = 10$, $N = 25$ and $N = 200$. Plot all computed solutions together in one figure with the exact solution. Clearly label your graph. Note that the curves may overlap. Use a legend. *Note: We want you to arrive at a discrete solution for N=200 points here not because we need it for accuracy, but because it will make it hard to construct the matrix in MATLAB element-by-element, which we do not want you to do. Use some of MATLAB's built in functions for creating these matrices. Use the diag command to construct tridiagonal matrices (consult help diag for usage). If you are ambitious, construct A as a sparse matrix using sp-diags, since the sparse representation is more efficient for matrices with few nonzero elements.*

Exercise 2.6

In the first chapter, we discretized the 1-dimensional heat equation with Dirichlet boundary conditions:

$$\frac{d^2T}{dx^2} = 0, 0 \leq x \leq 1$$
$$T(0) = 0, T(1) = 2$$

The discretization leads to the matrix-vector equation $A\vec{T} = \vec{b}$, with

$$A = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix}, \vec{b} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -2 \end{pmatrix} \tag{2.8}$$

Here A is an $(N-1) \times (N-1)$ matrix.

(a) Find the LU factorization of A for $N = 10$ using `Matlab`. Is `Matlab` using any pivoting to find the LU decomposition? Find the inverse of A also. As you can see the inverse of A is a dense matrix.
Note: The attractive sparsity of A has been lost when computing its inverse, but L and U are sparse. Generally speaking, banded matrices have L and U with similar band structure. Naturally we then prefer to use the L and U matrices to compute the solution, and not the inverse. Finding L and U matrices with least "fill-in" ("fill-in" refers to nonzeros appearing at locations in the matrix where A has a zero element) is an active research area, and generally involves sophisticated matrix re-ordering algorithms.

(b) Compute the determinants of L and U for $N = 1000$ using `Matlab`'s determinant command. Why does the fact that they are both nonzero imply that A is non-singular? How could you have computed these determinants really quickly yourself without `Matlab`'s determinant command?

Exercise 2.7

In Section 2.2, we introduced the LU decomposition of A, where L is unit lower triangular, in that it has ones along the diagonal, and U is upper triangular. However, in the case of symmetric matrices, such as the discretization matrix, it is possible to decompose A as LDL^T, where L is still unit-lower triangular and D is diagonal. This decomposition clearly shows off the symmetric nature of A.

(a) Find the LDL^T decomposition for the matrix given in Exercise 2.6. Show that L is bidiagonal. How do D and L relate to the matrix U in the LU decomposition of A?
Hint: Think about how D and L^T relate to U.
Note: Computing LDL^T this way does not work out for any symmetric matrix, it only happens to work for this matrix in particular.

*(b) (i) To solve $A\vec{x} = \vec{b}$ we can exploit this new decomposition. We get $LDL^T\vec{x} = \vec{b}$ which we can now break into three parts: Solve $L\vec{y} = \vec{b}$ using forward substitution, now solve $D\vec{z} = \vec{y}$, and then solve $L^T\vec{x} = \vec{z}$ using back substitution. Write a `Matlab` code that does exactly this for arbitrary N for the A in Exercise 2.6.

(ii) Solve a system of the same form as Exercise 2.6 for A of size 10 and of size 1000 with $\vec{b}$ having all zeros except 2 as the last entry in both cases, and verify the correctness of your solution using `Matlab`'s `A\b` operator and the `norm` command.

Exercise 2.8

For tridiagonal matrices, such as those given in Exercise 2.6, a fast algorithm was developed called the Thomas algorithm. Go here for a description of this algorithm:

http://en.wikipedia.org/wiki/Tridiagonal matrix algorithm

(a) Implement the algorithm in `Matlab` for the system $A\vec{t} = \vec{b}$ given as in equation (2.8) with comments to show your understanding of the algorithm. Use your implementation to compute solutions for the system given in Exercise 2.6 for $N = 400$ and $N = 1000$. Compare the 2-norm of the solutions found using Thomas and backslash `(A\b)` in `Matlab` for both $N = 400$ and $N = 1000$.

*(b) A $n \times n$ matrix A is said to be strictly diagonally dominant if

$$|A_{ii}| > \sum_{j=1}^{i-1} |A_{ij}| + \sum_{j=i+1}^{n} |A_{ij}| \quad \text{for all } i \text{ between 1 and } n$$

where A_{ij} denotes the entry in the i^{th} row and j^{th} column. Show that strictly diagonally dominance guarantees the Thomas algorithm to work.

*(c) The matrix in equation (2.8) is not strictly diagonally dominant. Nevertheless, the Thomas algorithm works, which we hope you found out in part 1 of this problem. How can you explain that it does?

* Exercise 2.9

The column vectors $\vec{u}$ and $\vec{v}$ have n elements. We form the $n \times n$ matrix A as $A = \vec{u}\vec{v}^T$. We now perform Gaussian elimination on A to find the matrix U. Show that U has either $n-1$ or n zero rows.

Exercise 2.10

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

If the matrix A is symmetric and tridiagonal, pivoting is never required in Gaussian elimination.

Chapter 3

Conditioning and perturbation studies for $A\vec{x} = \vec{b}$

In many applications (engineering or elsewhere), the matrix A as well as the vector $\vec{b}$ may have elements that are inexact and/or uncertain. For example, when solving a fluid flow problem, the matrix elements will contain physical parameters, such as viscosity, that are available from experimentation and will not be exact. The vector $\vec{b}$ typically contains information about boundary conditions and again these can be uncertain or approximations of reality. This is quite typical for many matrix-vector systems: they are derived from physical or engineering processes that inherently have inexact or uncertain data. And even if we did have perfect parameters, the actual computer calculations will introduce roundoff during the solution process. What this all means is that we must understand how perturbations in A as well as $\vec{b}$ may affect the quality of the solution to $A\vec{x} = \vec{b}$.

In this section, we ask ourselves two questions:

- How reliable are the results we get taking into account that the data in A and $\vec{b}$ likely contain errors?
- If the results of the computation are indeed not reliable, what, if anything, can we do about it?

We can reformulate the first question also in this way: will small perturbations in the data (which we usually cannot avoid) and/or round-off errors (which we also cannot avoid) lead to small perturbations in the solution $\vec{x}$, as we hope they will, or can small perturbations have a large effect on the solution (in which case we are not so happy)?

For the discussion, we will need to define vector and matrix norms, see 1.5. Here, we will use the generic notation $\|\vec{x}\|$ and $\|A\|$, and assume that the vector and matrix norms chosen

are compatible so that they satisfy the inequality $\|A\vec{x}\| \leq \|A\|\|\vec{x}\|$. We assume in all of this section that $A\vec{x} = \vec{b}$ has a unique solution. That is A is an invertible $n \times n$ matrix and A^{-1} exists.

3.1 Effect on $\vec{x}$ because of a perturbation in $\vec{b}$

First, we will look at the effect on the solution vector $\vec{x}$ resulting from a perturbation in the vector $\vec{b}$. We call the perturbation $\partial\vec{b}$ (note here $\partial\vec{b}$ is one particular perturbation vector, not some variable called ∂ times $\vec{b}$). This perturbation in $\vec{b}$ will change the solution to the matrix-vector system. We will get $A\vec{y} = \partial\vec{b}$, where $\vec{y}$ is the new solution. We assume that this $\vec{y}$ is close to the old solution $\vec{x}$. Then, we can write is as $\vec{y} = \vec{x} + \partial\vec{x}$, given the equality

$$A(\vec{x} + \partial\vec{x}) = \vec{b} + \partial\vec{b}. \tag{3.1}$$

What we are interested in is how large $\partial\vec{x}$ is for a given $\partial\vec{b}$. So, we may be tempted to look at $\|\partial\vec{x}\|$ as compared to $\|\partial\vec{b}\|$. But, is this the right question to ask? Not really, as we are not so interested in the absolute norm of $\partial\vec{x}$ but more in how big this is relative to the solution $\vec{x}$. So it would be better to ask: how big is the relative change in $\vec{x}$ as a result of some given relative change in $\vec{b}$. In other words, how is
$\|\partial\vec{x}\|/\|\vec{x}\|$ related to $\|\partial\vec{b}\|/\|\vec{b}\|$?

So, how do we find this relation? We have the two equations

$$A(\vec{x} + \partial\vec{x}) = \vec{b} + \partial\vec{b},$$
$$A\vec{x} = \vec{b}.$$

We can subtract these two equations, which gives us $A\partial\vec{x} = \partial\vec{b}$, or $\partial\vec{x} = A^{-1}\partial\vec{b}$ so that

$$\|\partial\vec{x}\| = \|A^{-1}\partial\vec{b}\| \leq \|A^{-1}\|\|\partial\vec{b}\|. \tag{3.2}$$

At the same time $A\vec{x} = \vec{b}$ gives

$$\|\vec{b}\| \leq \|A\|\|\vec{x}\|, \quad \text{or } \frac{1}{\|\vec{x}\|} \leq \frac{\|A\|}{\|\vec{b}\|}. \tag{3.3}$$

Combining the inequalities in 3.2 and 3.3 gives

$$\frac{\|\partial\vec{x}\|}{\|\vec{x}\|} \leq \|A^{-1}\|\|A\|\frac{\|\partial\vec{b}\|}{\|\vec{b}\|}. \tag{3.4}$$

An upper bound like that given in inequality 3.4 tells us that in the worst possible case, the relative change in $\vec{x}$ will be $\|A\|\|A^{-1}\|$ times the given relative change in $\vec{b}$. If $\vec{b}$ is

perturbed by 1 percent, $\vec{x}$ could be perturbed quite a lot more if $\|A\|\|A^{-1}\|$ is very large. This number is therefore quite an important one and we give it the name *condition number*. In the book by Strang (Linear Algebra and its Applications) and some other textbooks, the term *amplification factor* is used, but we will preserve that for later use in iterative methods. We often denote the condition number by $\kappa(A)$ and will use $\kappa(A) = \|A\|\|A^{-1}\|$ in these notes.

Because $AA^{-1} = I$, the smallest value $\kappa(A)$ can have is $\|I\|$ (do you see this?). For the 2-norm, for example, $\|I\| = 1$ and we get $\kappa \geq 1$. For some matrices, $\kappa(A) = \infty$. Which matrices would these be?

Let's look at an example. Take

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \text{ with exact solution } \vec{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}. \tag{3.5}$$

We can find that

$$A^{-1} = \begin{bmatrix} 10001 & -10000 \\ -10000 & 10000 \end{bmatrix},$$

which leads to $\kappa(A) \approx 40,000$ in the Frobenius norm. That is quite a large condition number for such a small matrix, so we expect some trouble.

We now perturb the vector $\vec{b}$ a little with the perturbation $\partial\vec{b} = \begin{bmatrix} 0 \\ 0.0001 \end{bmatrix}$. This gives $\vec{x} + \partial\vec{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, which is a very large change compared to $\vec{x}$. In the 2-norm, we have that

$$\frac{\|\partial\vec{x}\|}{\|\vec{x}\|} = \frac{\sqrt{2}}{2} \leq \kappa(A)\frac{\|\partial\vec{b}\|}{\|\vec{b}\|} = \sqrt{2}.$$

In this case, the worst case estimate is a good indicator of the error in $\vec{x}$. A small change in $\vec{b}$ leads to a very large change in $\vec{x}$, which is undesirable. It means that if we suspect that the vector $\vec{b}$ we work with might have such perturbations, we can not trust our solution. We call such a system *ill conditioned*. Could we have suspected problems without computing the condition number of A? Probably so, as A is quite close to being singular: it's second row is very close to its first row.

Of course, this case is quite extreme. If we instead take the system with

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \text{ with exact solution } \vec{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix},$$

then the same perturbation in $\vec{b}$ gives $\vec{x} + \partial\vec{x} = \begin{bmatrix} 2.00005 \\ -0.00005 \end{bmatrix}$. So here, a small change in $\vec{b}$ leads to only a small change in $\vec{x}$, which is what we would like to see. To compute the

condition number, which we expect to be small, we need to know the inverse of A. It can be readily be found as

$$A^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The condition number of A, measured using the Frobenius norm, is given by $\kappa(A) = 2$.

3.2 Effect on $\vec{x}$ because of a perturbation in $\vec{b}$ and in A

In case both the vector $\vec{b}$ and the matrix A are perturbed, we can derive the following bound on the relative error in $\vec{x}$:

$$\frac{\|\partial\vec{x}\|}{\|\vec{x}\|} \leq C, \text{ with } C \approx \kappa(A)\left(\frac{\|\partial\vec{b}\|}{\|\vec{b}\|} + \frac{\|\partial A\|}{\|A\|}\right).$$

This is the topic of one of the exercises in this chapter.

3.3 More on ill-conditioning

3.3.1 What causes ill-conditioning

We have seen that *ill-conditioning is a property of the matrix* A. So if the matrix-vector system represent a physical or engineering process, the ill-conditioning is really a reflection of the nature of this physics or process. An example of a process that can lead to ill-conditioning is one with disparate scales. Suppose, for example, that we are trying to accurately simulate flow in the Atlantic Ocean. The ocean has some very large scale flow patterns, such as the Gulf Stream, that change very slowly, as well as local patterns (say small eddies in coastal regions) that change very rapidly. When creating the matrix-vector systems that represent the relations between velocities, density and pressure in such systems, we will find that the matrices are ill-conditioned. The same thing happens when modeling climate models, or combustion engines, and even social patterns with large scale fluctuations in behavior as well as localized behaviors.

Sometimes, we can spot ill-conditioning because the matrix in question is clearly nearly singular, as for the example in 3.5. The two rows of this matrix are almost identical, which renders the matrix almost singular.

The main thing to remember is that ill-conditioning is a property of the underlying system. We can not rectify it by simply re-ordering the rows or columns in the matrix, only by changing the underlying physics or process model.

3.3.2 What to do when we have an ill-conditioned system?

When we are asked to solve a matrix-vector equation it is very prudent to check the condition number of the system. If this condition number is large, we can expect some problems with robustness. But mind that the bounds that we have derived in this chapter thus far are all upper bounds. They are worst case scenarios. The errors in the solution may or may not be that big. Typically then what you do is a perturbation analysis: you purposely put in perturbations in your matrix and your right hand side vector $\vec{b}$ to check what happens to the solution. If the solution is very sensitive and you conclude that you cannot trust your solutions, then perhaps you need to find different ways to model the processes you are interested in.

3.3.3 What is a large condition number?

A question you may ask at this stage is what constitutes a large condition number? Unfortunately, there are no sharp definitions. As a practicing engineer, you will routinely do perturbation studies to develop a good feel for systems and their behaviors. What you also need to realize is that often condition numbers increase as the size of your system increases. So, that means that a condition number of $\kappa = 10000$ would be pretty bad for a 2×2 matrix, but perfectly ok for a 1000000×1000000 matrix.

If you are not sure whether or not a condition number is too large, simply run perturbation studies to see the impact on the computed solutions of a small change in the data. If this impact is larger than you like, you have to take some action.

3.4 Round-off error accumulation

In the examples so far, we did not introduce any round-off errors: all computations were performed exactly. This means that the way that we compute the result does not have any impact on the results. When we compute large systems on a computer, round-off errors will be made and their accumulation will depend on the order in which we perform computations.

What is important is to realize that round-off error accumulation can lead to large errors in our solution of the matrix-vector system $A\vec{x} = \vec{b}$ even if the matrix A is well-conditioned. The good news is that for well-conditioned systems it is generally possible to avoid round-off error accumulation by choosing better algorithms, that is, by being more careful about the order in which we perform calculations.

To make this all a bit clearer, let's look at an example. Round-off errors made by one addition, multiplication or division are typically small and dictated by the machine precision of the computer used. It typically takes quite a few operations to see their accummulated impact. In many realistic simulations, the systems are indeed very large and so we have to take care. It is not really practical to look at very very large systems in these notes. So instead we look at a small system but strongly exaggerate the round-off error. That is, we artificially increase the floating point errors we make to allow the illustration.

Take the system

$$A\vec{x} = \vec{b}, \text{ with } A = \begin{bmatrix} 0.01 & 1 \\ 1 & -1 \end{bmatrix}, \vec{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{3.6}$$

The condition number of A is $\kappa(A) \approx 3$. The matrix is therefore well-conditioned and we do not expect any problems.

In exact arithmetic, Gaussian Elimination leads to

$$U\vec{x} = \vec{c}, \text{ with } U = \begin{bmatrix} 0.01 & 1 \\ 0 & -101 \end{bmatrix}, \vec{c} = \begin{bmatrix} 1 \\ -100 \end{bmatrix},$$

with solution $\vec{x} = \begin{bmatrix} 0.9900990099 \\ 0.9900990099 \end{bmatrix}$.

Now we introduce round-off error in our computations. We assume that we can only work with two-point floating point arithmetic. This means that the number -101 is truncated to -100. Then, we find that $x_2 = 1$ and so $x_1 = 0$. This is a huge change in the solution.

Could we have avoided this? It turns out that the problem is really caused by the small pivot 0.01. This leads to a large multiplication factor (100) and that causes the information stored in the element $a_{22} = -1$ to be overwhelmed and truncated away by the large -100 subtraction. When this happens we talk about *catastrophic cancellation*.

To avoid catastrophic cancellation, we could try changing the system so we get another, larger, pivot. For example, we can try swapping rows before applying Gaussian Elimination. This gives the system

$$\begin{bmatrix} 1 & -1 \\ 0.01 & 1 \end{bmatrix} \vec{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

which leads (with rounding errors)

$$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \vec{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

with the solution $x_1 = x_2 = 1$, which is only a small perturbation of the exact solution.

Row swapping, or column swapping, to avoid small pivots is very common practice and is typically referred to as *partial pivoting*. We discussed partial pivoting already in Section 2.1.4. As you can see, partial pivoting is highly recommended to reduce the risk of round-off error related problems.

3.5 In summary

In summary:

- The condition number $\kappa(A)$ of a matrix is defined as $\kappa(A) = \|A\|\|A^{-1}\|$. It gives information about the conditioning of the system $A\vec{x} = \vec{b}$, that is, about the sensitivity of the system to perturbations in the coefficients of A, the elements of $\vec{b}$ and round-off error introduced during the solution process.
- If a matrix is *ill-conditioned* it is very important to conduct perturbation analyses in which the input data are purposely perturbed and the effect on the solution $\vec{x}$ measured. If $\vec{x}$ is found to indeed be very sensitive, the solutions must be interpreted carefully. Many input data are by nature uncertain and so we simply cannot trust solutions in such cases.
- If a matrix is *well-conditioned* we still have to be careful during the solution process because a poor choice of algorithm can lead to severe round-off error accumulation. It is often possible to avoid such round-off error accumulations by reorganizing the computations used in the solution process. In Gaussian Elimination the standard procedure is to avoid small pivots by row (or column) swapping.

3.6 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled $\star$ are more challenging.*

Exercise 3.1

Assume that the vector $\vec{b}$ in the system $A\vec{x} = \vec{b}$ is perturbed by $\overrightarrow{\delta b}$. The solution will now change to $\vec{x} + \overrightarrow{\delta x}$. In the note, we derived an upper bound for the relative error of $\vec{x}$ in terms of the condition number of A and the relative error in $\vec{b}$:

$$\frac{\|\overrightarrow{\delta x}\|_2}{\vec{x}} \leq \|A\|_F \|A^{-1}\|_F \frac{\|\overrightarrow{\delta b}\|_2}{\vec{b}}.$$

Derive a **lower** bound for the relative error of $\vec{x}$.

* Exercise 3.2

In the notes, we discussed how a perturbation in the matrix A can lead to perturbations in the solution $\vec{x}$, and separately, how a perturbation in the vector $\vec{b}$ can perturb $\vec{x}$. How might the solution $\vec{x}$ to the linear system $A\vec{x} = \vec{b}$ change when we perturb both A and $\vec{b}$?

Exercise 3.3

The system $A\vec{x} = \vec{b}$, with $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \vec{b} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$ has the solution $\vec{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The vector $\vec{b}$ is now perturbed. The new vector is equal to $\vec{b} + \vec{\partial b}$ with $\vec{\partial b} = \begin{bmatrix} 0.001 \\ 0 \end{bmatrix}$. Estimate the maximum perturbation we can expect in the solution $\vec{x}$, measured in the vector 2-norm. For matrix-norms, you may use the Frobenius norm.

Exercise 3.4

In exercise 2.5, we solved a heat equation using numerical discretization for the nonuniform source term given by $f(x) = -10\sin(3\pi x/2)$.

(a) Find the condition number of the matrix A using MATLAB for $N = 5$, $N = 10$, $N = 25$ and $N = 200$. Use the Frobenius norm to compute the condition number, that is, set $\kappa(A) = \|A\|_F \|A^{-1}\|_F$ (use the MATLAB function cond(A,'fro')).

(b) Plot the condition numbers and comment on any pattern that you might find.

(c) Perturb the matrix A by ∂A for $\partial A = 0.1A$. Solve the perturbed system of equations $(A + \partial A)\vec{T} = \vec{c}$ and plot the solution. Comment on what you see. Can you relate it to the condition number of A?

Exercise 3.5

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

(a) $\kappa(\alpha A) = \alpha \cdot \kappa(A)$, where $\alpha > 0$

(b) If the determinant of an $n \times n$ matrix A is nearly 0, then the matrix is ill-conditioned

(c) Catastrophic cancellation can only occur during Gaussian Elimination of a matrix if that matrix is ill-conditioned.

Exercise 3.6

If a matrix is close to being singular, its condition number is very high. Give an example of a 4×4 matrix for which this is the case.

Exercise 3.7

Show that any $n \times n$ permutation matrix P has a condition number (computed with the Frobenius norm) that is equal to n.

Chapter 4

11 concepts underlying matrix theory

In this chapter, we succinctly present 11 concepts that are frequently used in linear algebra. They are all important. The chapter may be a little less exciting than chapters in which new algorithms are introduced, but when you master these 11 concepts, you have a great basis for the remainder of the notes.

Each section has one or more problems to help consolidate the material.

4.1 Linear combination of vectors

A *linear combination* of a set of vectors $\vec{v}_j, j = 1, \ldots, n$ is an expression of the form

$$\alpha_1\vec{v}_1 + \alpha_2\vec{v}_2 + \ldots + \alpha_n\vec{v}_n = \sum_{j=1}^{n} \alpha_j\vec{v}_j, \quad \alpha_i \in \mathcal{R}. \tag{4.1}$$

Question: What is $A\vec{x}$ a linear combination of?

Answer: We have seen that

$$A\vec{x} = [\vec{a}_1\ \vec{a}_2\ \ldots \vec{a}_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{j=1}^{n} x_j \vec{a}_j.$$

That is, the matrix-vector product $A\vec{x}$ is a linear combination of the columns of A. The coefficients of the linear combination are the vector elements x_j.

The problem above shows that we can find a solution to the system $A\vec{x} = \vec{b}$ if and only if $\vec{b}$ is a linear combination of the columns of A. For example, if we have the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}, \vec{b}_1 = \begin{bmatrix} 6 \\ 0 \end{bmatrix}, \vec{b}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

then $A\vec{x} = \vec{b}_1$ can be solved, whereas $A\vec{x} = \vec{b}_2$ can not.

4.2 Linear (in)dependence

A set of vectors $\vec{v}_j, j = 1, \ldots, n$ is linearly *independent* if none of the vectors $\vec{v}_j$ can be expressed as a linear combination of the others. The set of vectors is *dependent* if one or more of the vectors $\vec{v}_j$ can be expressed as a linear combination of the others.

Question: Are the vectors $\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ dependent or independent?

Answer: Because the third vector in the set is the difference of the second and the first, it is clear that the vectors are dependent.

If one of the vectors in a set is a linear combination of the others, then we know that for some integer k, $\vec{v}_k = \sum_{j=1, j\neq k}^{n} \alpha_j \vec{v}_j$, which we can rewrite as $\sum_{j=1}^{n} \beta_j \vec{v}_j = \vec{0}$, for a set of coefficients β_j not all equal to zero. So, we could also say that a set of vectors $\vec{v}_j, j = 1, \ldots n$ is linearly *independent* is

$$\sum_{j=1}^{n} \beta_j \vec{v}_j = \vec{0}, \text{ only if all } \beta_j = 0.$$

Question: are the vectors $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ independent?

Answer: If the vectors are independent then $\beta_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \beta_2 \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} + \beta_3 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ only for $\beta_1 = \beta_2 = \beta_3 = 0$. So, $\vec{\beta} = \vec{0}$ is the only solution to $A\vec{\beta} = \vec{0}$, where A is the matrix whose columns are the three vectors under study. This is the case if A is nonsingular. To find out whether A is nonsingular, we perform Gaussian Elimination, which gives

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow U = \begin{bmatrix} 1 & 2 & 1 \\ 0 & -1 & -1 \\ 0 & 0 & 1 \end{bmatrix},$$

and we see that none of the pivots in U are zero. Hence, A is nonsingular and we find that the given vectors are linearly independent.

4.3 Vector space

A *vector space* is a collection of vectors that is closed under multiplication by a constant (including 0) and addition/subtraction. In other words, *any* linear combination of *any* set of vectors of a vector space is again in the vector space.

We immediately understand that $\vec{0}$ should be in *all* vector spaces. And, conversely, if a collection of vectors does not contain $\vec{0}$, it can not be a vector space.

Let's look at a few examples:

- $\mathbb{R}$, or in general $\mathbb{R}^n$ with n a positive integer, is a vector space.
- Any line in $\mathbb{R}^2$ of the form $x_2 = \alpha x_1$ is a vector space. Since this line is a subset of $\mathbb{R}^2$ we refer to it as a *vector subspace*. Note that all vectors in this subspace live in the space $\mathbb{R}^2$, but they are constrained to this line. Since a line only has one degree of freedom (a vector can move up and down the line, but cannot leave it), we refer to this as a one-dimensional subspace in the two-dimensional space $\mathbb{R}^2$.
- A line in $\mathbb{R}^2$ of the form $x_2 = x_1 + b$, with $b \neq 0$, can not be a vector space as $x_1 = x_2 = 0$ is not on this line.
- The set of all vectors $\vec{x}$ that satisfy $A\vec{x} = \vec{0}$ is a vector space. To check, assume that the set of vectors $\vec{x}_i, i = 1, \ldots, p$ all satisfy $A\vec{x}_i = \vec{0}$. We should check if a linear

combination of these vectors $\vec{y} = \sum_i^p \alpha_i \vec{x}_i$ also satisfies $A\vec{y} = \vec{0}$. Let's see:

$$A\vec{y} = A\sum_i^p \alpha_i \vec{x}_i = \sum_i^p \alpha_i A\vec{x}_i,$$

and since $A\vec{x}_i = \vec{0}$, we see that $A\vec{y} = \vec{0}$ also.

- The set of all vectors $\vec{x}$ that satisfy $A\vec{x} = \vec{b}$, $\vec{b} \neq \vec{0}$, is not a vector space. We know this because $\vec{0}$ is not in the set.

4.4 Spanning set

A set of vectors $\vec{v}_i, i = 1, \ldots, m$ spans a vector space, or is a *spanning set*, if every vector in the space can be expressed as a linear combination of these $\vec{v}_i's$.

Question: Can you give a few spanning sets for $\mathbb{R}^2$?

Answer: Simple examples are the sets

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}, \text{ and } \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}.$$

The second set is a so called *redundant spanning set*. To span $\mathbb{R}^2$ we need only two independent vectors, not three. Indeed, you can find quickly that each of the vectors in this redundant spanning set is a linear combination of the others. See that?

4.5 Basis

A *basis* of a vector space, or vector subspace is a spanning set with independent vectors.

Any redundant spanning set can be reduced to a basis. In the last section, we gave two examples of spanning sets for $\mathbb{R}^2$. The first is a basis, the second is not. In the exercises, we will ask you to prove that the number of vectors needed in a basis is always the same.

4.6 Dimension

All bases for a vector space, or vector subspace must contain the same number of vectors (see exercises). This number is called the *dimension* of the space or subspace.

For example, we know that the line $x_2 = x_1$ is a subspace of $\mathbb{R}^2$. To find a basis, we observe that each vector on this line is of the form

$$\vec{x} = \begin{bmatrix} x_1 \\ x_1 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Here, x_1 can be any number in $\mathbb{R}$. So, we see that all vectors on the line are simply a linear combination of the vector $[1, 1]^T$. This vector therefore forms the basis of the line, and we see again that a line itself has one dimension.

We now give several questions that cover the concepts introduced in sections 1 through 6. They all relate to the set of vectors $\vec{x}$ in $\mathbb{R}^3$ that all satisfy $x_1 + x_2 + x_3 = 0$.

Question: Is this set a vector space?

Answer: Yep. Look at any number of vectors $\vec{c}_j, j = 1, \ldots, p$ in this set, where p is some positive integer. Each of these vectors therefore satisfy $c_{j,1} + c_{j,2} + c_{j,3} = 0$. Now, take a linear combination $\sum_{j=1}^{p} \alpha_j \vec{c}_j$. We need to now check whether the elements of this linear combination again sum to zero. This is certainly the case:

$$\sum_{j=1}^{n} \alpha_j c_{j,1} + \sum_{j=1}^{n} \alpha_j c_{j,2} + \sum_{j=1}^{n} \alpha_j c_{j,3} = \sum_{j=1}^{n} \alpha_j (c_{j,1} + c_{j,2} + c_{j,3}) = 0.$$

Question: Find a basis for this vector space.

Answer: First, let's think about how many basis vectors we expect. We know that vectors in the subspace satisfy $x_1 + x_2 + x_3 = 0$. In other words, we have 1 constraint for 3 vector elements. This means that there are only $3 - 1 = 2$ degrees of freedom left. In $\mathbb{R}^3$ a subspace with two degrees of freedom, or as we say with two dimensions, is a plane. We need two basis vectors to span it. How do we get them? Here's a common approach. We know that $x_1 = -x_2 - x_3$, so for each vector in the subspace we have

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -x_2 - x_3 \\ x_2 \\ x_3 \end{bmatrix} = x_2 \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix},$$

where both x_2 and x_3 can take any value in $\mathbb{R}$. Clearly, any vector in the subspace is a linear combination of the vectors in the set

$$\left\{ \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \right\},$$

and this is therefore the basis for the subspace. Wait a minute. Should we say this is *the* basis for the subspace or *a* basis for the subspace? The latter, of course, as bases are certainly not unique.

Question: Find a vector that is orthogonal to this vector space.

Answer: We can actually see this quite easily. We know that $x_1 + x_2 + x_3 = 0$, but we can interpret this expression as

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0,$$

which shows that the vector $\vec{n} = [1, 1, 1]^T$ is orthogonal to any vector in the subspace. We can readily see that this vector is indeed orthogonal to both of the basis vectors and from that we can also conclude that it must be orthogonal to any linear combination of the basis vectors and hence to any vector in the subspace.

4.7 Complement sets

In the above example, we found a vector $\vec{c}$ orthogonal to a two-dimensional plane in $\mathbb{R}^3$. If we took the two basis vectors of the plane and added to this collection the vector $\vec{c}$, we

would have three vectors that were all independent. These would therefore form a basis for the whole of $\mathbb{R}^3$. We say that $\vec{c}$ complements the basis of the plane. In more general terms:

Two sets of vectors in $\mathbb{R}^n$ are called *complement sets* if each set is a basis for a subspace of $\mathbb{R}^n$ and together the vectors of both sets form a basis for $\mathbb{R}^n$. The sets are orthogonal complement sets if all vectors of one set are orthogonal to all vectors of the second set.

4.8 Column space

The *column space* $\mathcal{R}(\mathcal{A})$ of a matrix A is the space spanned by the columns of A. The columns of an $m \times n$ matrix A all live in the space $\mathbb{R}^m$, so the column space is a subspace of $\mathbb{R}^m$. If there are m independent columns, then the column space is actually the space $\mathbb{R}^m$ itself.

We have seen before that the matrix-vector equation $A\vec{x} = \vec{b}$ only has a solution if $\vec{b}$ is a linear combination of the columns of A. What does really means is that $\vec{b}$ must be in $\mathcal{R}(\mathcal{A})$.

4.9 Row space

The *row space* of a matrix A is the space spanned by the rows of A. Because the rows of A are the columns of the matrix A^T, this space is typically denoted by $\mathcal{R}(\mathcal{A}^\mathcal{T})$ and does not have its own symbol. Each year, we decide what symbol to use in CME200. In 2013, we used the tree symbol, in honor of the Stanford tree. Latex, which I am using to write these notes, does not have a tree symbol and so here I will be using the symbol $\clubsuit(A)$ that comes closest.

The rows of the $m \times n$ matrix A live in the space $\mathbb{R}^n$, so the row space is a subspace of $\mathbb{R}^n$. If there are n independent rows, then the row space is actually the space $\mathbb{R}^n$ itself.

Question: Why do A and the corresponding U found in Gaussian Elimination have the same row space?

Answer: Remember how we got U: to create U we take a sequence of linear combination of rows of A. In other words, all rows of U are linear combinations of the rows of A. The rows of U are by design created such that they are either independent or zero. Do you see this?

4.10 Matrix rank

The *rank* of a matrix A, which is denoted by $r(A)$, is the dimension of the row space of A.

Since the row spaces of A and U are the same, the ranks of A and U are also the same. The rank of U can easily be found: it is simply equal to the number of nonzero rows in U. Convince yourself of this.

It turns out the $r(A)$ is also the dimension of $\mathcal{R}(\mathcal{A})$. We will see this in a worked out problem below.

4.11 Null space

The *nullspace* of A, denoted by $\mathcal{N}(\mathcal{A})$ is the collection of all vectors $\vec{x}$ that satisfy $A\vec{x} = \vec{0}$. The vectors in the nullspace all live in the space $\mathbb{R}^n$.

All vectors in the nullspace of a matrix are orthogonal to all vectors in the row space of that same matrix. This is easy to see if we write the matrix-vector equation $A\vec{x}$ as

$$A\vec{x} = \begin{bmatrix} \vec{r}_1^T\vec{x} \\ \vdots \\ \vec{r}_m^T\vec{x} \end{bmatrix},$$

which shows that if $A\vec{x} = \vec{0}$, the vector $\vec{x}$ is orthogonal to all row vectors of A.

The following question covers sections 8 through 11.

Question: The 3×4 matrix A is given by

$$A = \begin{bmatrix} 1 & 2 & 0 & 2 \\ -1 & -2 & 1 & 1 \\ 1 & 2 & -3 & -7 \end{bmatrix}.$$

Find bases for the row space, column space and null space of A. What is $r(A)$?

Answer: We start with Gaussian Elimination, which can give us information about dependencies of the rows (and columns) of A. We find that the matrix U is equal to

$$U = \begin{bmatrix} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

We have not used any pivoting in this process.
The rows of U are formed by taking linear combinations of the rows of A. A zero row in U therefore indicates that this same row in A was linearly dependent on the others. Here, we see two nonzero rows in U. These rows can be used as a basis for $\clubsuit(A)$. We can also use the first two rows of the original A.
We know that $\mathcal{N}(\mathcal{A}) = \mathcal{N}(\mathcal{U})$. Because U is sparser, it is easiest to find a basis for the nullspace of U. Setting $U\vec{x} = \vec{0}$ gives $x_1 + 2x_2 + 2x_4 = 0$ and $x_3 + 3x_4 = 0$. These are two equations in four unknowns. We therefore know that we have two degrees of freedom. We pick any two variables and express them in the others. Here, I write $x_3 = -3x_4$ and $x_1 = -2x_2 - 2x_4$, which gives

$$\vec{x} = \begin{bmatrix} -2x_2 - 2x_4 \\ x_2 \\ -3x_4 \\ x_4 \end{bmatrix} = x_2 \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -2 \\ 0 \\ -3 \\ 1 \end{bmatrix}.$$

With $x_2, x_4 \in \mathbb{R}$, this just states that the nullspace contains all linear combinations of the two vectors in this last expression, which are therefore base vectors of $\mathcal{N}(\mathcal{A})$.
For the column space, we first observe that the first and third column of U are independent. They are the columns with the nonzero pivots and these must be independent. Do you see this? These columns then form a basis for the column space of U. They do *not* form a basis for the column space of A as Gaussian elimination messes the columns about. What we can do, however, is conclude that the first and third column of A also must be independent and hence they can chosen as base vectors for $\mathcal{R}(\mathcal{A})$. That this is true comes from the following argument: the nullspaces of A and U are the same; for each base vector $\vec{y}$ of the nullspaces, we know that $A\vec{y} = \vec{0}$ and $U\vec{y} = \vec{0}$; $A\vec{y}$ is a linear combination of the columns of A; because the nullspaces are the same, the same linear combinations of the columns of A and U are zero; the columns therefore have the same dependency relations.

4.12 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled ⋆ are more challenging.*

Exercise 4.1

Let L be the set of vectors $\vec{x}$ in $\mathbb{R}^4$ for which

$$x_1 + x_2 + x_3 = 0. \tag{4.2}$$

Find a basis for L. What is the dimension of L?

Exercise 4.2

Find a basis for the subspace of $\mathbb{R}^4$ spanned by the vectors

$$\left\{ \begin{bmatrix} 0 \\ 2 \\ 3 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \right\}.$$

Exercise 4.3

(a) Find a basis for the column space and row space of matrix A given by

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 4 & 6 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

(b) Construct a matrix B such that the null space of B is identical to the row space of A.

Exercise 4.4

We are given the LU decomposition of a matrix A as

$$A = LU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 & 1 & 2 & 1 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

(a) What is the rank $r(A)$ of matrix A?

(b) Find a basis for the null space of A.

(c) Find a basis for the column space of A.

(d) For $\vec{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ do solution(s) to $A\vec{x} = \vec{b}$ exist? Clearly motivate your answer. Note that it is not necessary to compute the solution(s).

Exercise 4.5

Consider the matrix

$$A = \begin{bmatrix} -2 & -1 & 0 \\ -1 & -1 & -1 \\ 0 & -1 & -2 \end{bmatrix}.$$

(a) Find a basis for the null space of A. What is the rank of A?

(b) Find the solution(s), if any can be found, to the equation $A\vec{x} = \vec{b}$ with $\vec{b} = \begin{bmatrix} 0 \\ -1/2 \\ -1 \end{bmatrix}$.

(c) Prove that A^k is singular for all integer $k > 0$.

Exercise 4.6

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

(a) If the vectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m$ in $\mathbb{R}^n$ span a subspace S in $\mathbb{R}^n$, the dimension of S is m.

(b) If A and B have the same row space, the same column space, and the same nullspace, then A must be equal to B.

*(c) If for the $m \times n$ matrix A, the equation $A\vec{x} = \vec{b}$ always has at least one solution for every choice of $\vec{b}$, then the only solution to $A^T\vec{y} = \vec{0}$ is $\vec{y} = \vec{0}$.

*(d) If the m vectors $\vec{x}_i, i = 1, \ldots, m$ are orthogonal, they are also independent. Note here that $\vec{x}_i \in \mathbb{R}^n$ and $n > m$.

Exercise 4.7

A projector P is an $n \times n$ matrix that satisfies $P = P^2$. Show that $I - P$, with I the identity matrix, is also a projector, and that the range of $I - P$ exactly equals the null space of P.

* Exercise 4.8

An $n \times n$ matrix A has a property that the elements in each of its rows sum to 1. Let P be any $n \times n$ permutation matrix. Prove that $(P - A)$ is singular.

* Exercise 4.9

(a) The nonzero column vectors $\vec{u}$ and $\vec{v}$ have n elements. An $n \times n$ matrix A is given by $A = \vec{u}\vec{v}^T$ (Note: this is different from the innerproduct, which we would write as $\vec{v}^T\vec{u}$). Show that the rank of A is 1.

(b) Show that the converse is true also. That is, if the rank of a matrix A is 1, then we can find two vectors $\vec{u}$ and $\vec{v}$, such that $A = \vec{u}\vec{v}^T$.

* Exercise 4.10

Prove that all bases of a given vector subspace must have the same number of vectors.

Exercise 4.11

Let U and V be subspaces of $\mathbb{R}^n$.

(a) The *intersection* of U and V is the set

$$U \cap V := \{x \in \mathbb{R}^n \mid x \in U \text{ and } x \in V\}.$$

Is $U \cap V$ a subspace for any U and V?

(b) The *union* of U and V is the set

$$U \cup V := \{x \in \mathbb{R}^n \mid x \in U \text{ or } x \in V\}.$$

Is $U \cup V$ a subspace for any U and V?

Exercise 4.12

Let A be an $m \times n$ matrix with rank $r \leq \min\{m, n\}$. Depending on $m,\ n$ and r, a system $A\vec{x} = \vec{b}$ can have none, one, or infinitely many solutions.
For what choices of $m,\ n$ and r do each of the following cases hold? If no such $m,\ n$ and r can be found explain why not.

(a) $A\vec{x} = \vec{b}$ has no solutions, regardless of $\vec{b}$

(b) $A\vec{x} = \vec{b}$ has exactly 1 solution for any $\vec{b}$

(c) $A\vec{x} = \vec{b}$ has infinitely many solutions for any $\vec{b}$

(**Hint**: think of what conditions the column vectors and column space of A should satisfy.)

Exercise 4.13

Consider a matrix product AB, where A is $m \times n$ and B is $n \times p$. Show that the column space of AB is contained in the column space of A. Give an example of matrices $A,\ B$ such that those two spaces are not identical.
Definition. A vector space U is *contained* in another vector space V (denoted as $U \subseteq V$) if every vector $\vec{u} \in U$ ($\vec{u}$ in vector space U) is also in V.
Definition. We say that two vector spaces are *identical* (equal) if $U \subseteq V$ **and** $V \subseteq U$.
(e.g. V is identical to itself since $V \subseteq V$ and $V \subseteq V$.)

Exercise 4.14

Let V and W be 3 dimensional subspaces of $\mathbb{R}^5$. Show that V and W must have at least one nonzero vector in common.

Exercise 4.15

Show that the two sets of vectors $S_1 = \left\{ \begin{bmatrix} 3 \\ 1 \\ -2 \end{bmatrix}, \begin{bmatrix} -1 \\ 3 \\ -4 \end{bmatrix} \right\}$ and $S_2 = \left\{ \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix} \right\}$ span the same subspace.

Exercise 4.16

The 4×3 matrix A is given by $A = \begin{bmatrix} 1 & 2 & -3 \\ -1 & 1 & 1 \\ -1 & 0 & 2 \\ 3 & 8 & -10 \end{bmatrix}$, and the vector $\vec{d}$ by $\vec{d} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$.

(a) Find the basis for the row space, null space, and column space of A.

(b) Is it possible to find a vector $\vec{b}$ for which $A\vec{x} = \vec{b}$ has an infinite number of solutions? Explain. If it is possible, find such a $\vec{b}$.

(c) Show that $A\vec{x} = \vec{d}$, with $\vec{d}$ as given above, cannot be solved exactly. Then explain clearly how a solution may be found that minimizes $\|\vec{d} - A\vec{x}\|_2$, and give the general expression of this solution in terms of A and $\vec{d}$. **Note:** you need not calculate this solution exactly.

* Exercise 4.17

Indicate whether the following if-statement is TRUE or FALSE and motivate your answers clearly. To show a statement is false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

Let $\vec{x}$ and $\vec{y}$ be vectors in $\mathbb{R}^n$, and P be the projection matrix that projects a vector onto some subspace of $\mathbb{R}^n$. If $\vec{x}$ and $\vec{y}$ are orthogonal, then $P\vec{x}$ and $P\vec{y}$ are also orthogonal.

Exercise 4.18

We are given the following matrix A:

$$A = \begin{bmatrix} c & 4 & 0 \\ 4 & c & 3 \\ 0 & 3 & c \end{bmatrix}$$

(a) For $c = 0$, find a basis for the row space of A.

(b) Find all vectors $\vec{b}$ for which $A\vec{x} = \vec{b}$ does NOT have a solution. These vectors will be dependent on c.

Exercise 4.19

We are given the following matrix A:

$$A = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 1 & 2 & -1 & 1 \\ -1 & -4 & 5 & 1 \end{bmatrix}$$

(a) Find the LU decomposition of A. Give both L and U.

(b) Find all $\vec{x}$ for which $A\vec{x} = \vec{0}$.

(c) Does $A\vec{x} = \vec{b}$ have a solution for $\vec{b} = \begin{bmatrix} 2 \\ 3 \\ -5 \end{bmatrix}$?

Chapter 5

Orthogonalization

I work with orthogonal bases and matrices whenever possible. We will discuss orthogonality in this chapter and will see that orthogonal matrices and bases are well-conditioned and very easy to work with.

In this chapter, we will use inner products often. Just as a refresher remember that the inner product of two vectors $\vec{x}$ and $\vec{y}$ is given by $\vec{x}^T\vec{y}$. Other books may also use the notations $\vec{x}\cdot\vec{y}$ or $(\vec{x},\vec{y})$. Also remember from vector calculus that $\cos\theta = \frac{\vec{x}^T\vec{y}}{\|\vec{x}\|\|\vec{y}\|}$, with θ the angle between the vectors.

5.1 Orthogonal things

We start with a few definitions:

Orthogonal set of vectors A set of vectors $\vec{q}_j, j = 1,\dots m$ is *orthogonal* if each vector is orthogonal to any other, that is if $\vec{q}_j^T\vec{q}_k = 0, j \neq k$.

Orthonormal set of vectors The set is called *orthonormal* if besides from being orthogonal to each other, the vectors are also all of unit length.

Orthogonal basis for a vector space A basis is an *orthogonal basis* if the set of base vectors is orthogonal. Very often, orthogonal bases are also normalized (the length of all basis vectors are set to 1). Strictly speaking, we would refer to such bases as orthonormal, not orthogonal.

An orthogonal basis for a vector space is quite attractive. Remember that any vector in a vector space can be expressed as a linear combination of the basis vectors. Suppose that an m-dimensional vector space has a non-orthogonal basis $\vec{a}_1, \ldots, \vec{a}_m$. Then, a vector $\vec{x}$ in the vector space can be expressed as

$$\vec{x} = \sum_{i=1}^{m} \alpha_i \vec{a}_i,$$

or

$$\vec{x} = [\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_m]\, \vec{\alpha}.$$

To find the coefficients in the linear combination, we solve this last equation for α, which is an expensive operation if the matrix is large. But, now let's look at what it would involve to find the coefficients if the basis was orthogonal. We will denote the orthogonal basis vectors by $\vec{q}_i$. We again have

$$\vec{x} = \sum_{i=1}^{m} \beta_i \vec{q}_i.$$

but now we can easily find a coefficient β_k, for any k between 1 and m. We simply take the inner product of this equation with $\vec{q}_k$. We have

$$\vec{q}_k^T \vec{x} = \vec{q}_k^T \sum_{i=1}^{m} \beta_i \vec{q}_i = \sum_{i=1}^{m} \beta_i \vec{q}_k^T \vec{q}_i = \beta_k \vec{q}_k^T \vec{q}_k = \beta_k \|\vec{q}_k\|_2^2.$$

The last-but-one equality follows directly from the orthogonality of $\vec{q}_k$ to any other vector $\vec{q}_i, i \neq k$. We see that we can very quickly "pick out" each coefficient β_k just by performing an inner product (well, two inner products actually because we also need to compute the norm of the base vectors - if the basis is orthonormal this is not necessary).

Orthogonal matrix A $n \times n$ matrix Q is set to be an *orthogonal matrix* if its columns are orthonormal. It's perhaps a little strange that such a matrix is not called orthonormal, rather than orthogonal, but that's the way it's done. It's actually very unusual in applications to work with matrices whose columns are orthogonal but not orthonormal because it is very attractive from a computational point of view to normalize vectors (it reduces the risk for round-off error accumulation, amongst others).

An orthogonal $n \times n$ matrix Q has the wonderful property that $Q^T Q = I$, where I is the identity matrix. Do you see this? It is also true that $QQ^T = I$, which follows directly from

$$QQ^T = QQ^T(QQ^{-1}) = Q(Q^T Q)Q^{-1} = QIQ^{-1} = I.$$

This means that $Q^T = Q^{-1}$. The inverse of an orthogonal matrix is therefore very easy to find. An orthogonal $m \times n$ matrix, with $m \neq n$ still satisfies $Q^T Q = I$. Check this for yourself. However, now, it is not true that $QQ^T = I$. Convince yourself of this too.

Another nice property of an orthogonal matrix is that it preserves the 2−norm of a vector, that is $\|Q\vec{x}\|_2 = \|\vec{x}\|_2$. This can be seen easily if you remember that $\|\vec{x}\|_2 = \sqrt{\vec{x}^T\vec{x}}$. Then, $\|Q\vec{x}\|_2^2 = \vec{x}^T Q^T Q\vec{x} = \vec{x}^T\vec{x} = \|\vec{x}\|_2^2$.

5.2 Orthogonalization

Suppose that we are given a non-orthogonal basis $\vec{a}_i, i = 1, \ldots, p$ of a vector space in $\mathbb{R}^n$. How could we construct an orthonormal basis for this subspace? We do this through a process called *orthogonalization*. There are various approaches to orthogonalization. In this class we focus only on the so-called *Gram-Schmidt* process.

Note: all norms in this section are 2-norms.

5.2.1 Gram-Schmidt process

In the Gram-Schmidt process, we build the orthonormal basis one vector at a time. You will see that we use linear combinations of the base vectors $\vec{a}_i$ to find the orthonormal base vectors $\vec{q}_i$.

Step 1 The start is easy: we simply set $\vec{q}_1 = \vec{a}_1/\|\vec{a}_1\|$. In other words, $\vec{q}_1$ is just a normalized $\vec{a}_1$.

Step 2 We now consider $\vec{q}_2$. We will construct it from $\vec{a}_2$. Because $\vec{a}_1$ and $\vec{a}_2$ (and hence $\vec{q}_1$ and $\vec{a}_2$) are not necessarily orthogonal, we are only interested in the component of $\vec{a}_2$ that is orthogonal to $\vec{q}_1$, not in the component of $\vec{a}_2$ that is parallel to $\vec{q}_1$.

It is not so hard to find the parallel component using a bit of vector calculus. We refer to the figure. We know that

$$\cos\alpha = (\vec{a}_2^T\vec{q}_1)/(\|\vec{a}_2\|\|\vec{q}_1\|).$$

Of course, $\|\vec{q}_1\| = 1$, so we have

$$\cos\alpha = (\vec{a}_2^T\vec{q}_1)/\|\vec{a}_2\|.$$

At the same time, we know that this cosine is $b/\|\vec{a}_2\|$. Comparing the two expressions for the cosine, clearly $b = \cos\alpha\|\vec{a}_2\|$, which is equal to $b = \vec{a}_2^T\vec{q}_1$. Now, the component of $\vec{a}_2$ parallel to $\vec{q}_1$ is a vector. We know its length (which is given by b) and its direction (which is given by $\vec{q}_1$), so this vector is $(\vec{a}_2^T\vec{q}_1)\vec{q}_1$.

We now find that the component of $\vec{a}_2$ that is orthogonal to $\vec{q}_1$ is given by

$$\vec{w}_2 = \vec{a}_2 - \left(\vec{a}_2^T \vec{q}_1\right) \vec{q}_1.$$

This vector will not necessarily have unit length, so we normalize it to find $\vec{q}_2 = \vec{w}_2/\|\vec{w}_2\|$.

Can you see that $\vec{q}_1$ and $\vec{q}_2$ together span the same subspace as $\vec{a}_1$ and $\vec{a}_2$? Also, we see that $\vec{q}_2$ is found by taking a multiple of $\vec{a}_1$ and subtracting it from $\vec{a}_2$ (and normalizing along the way).

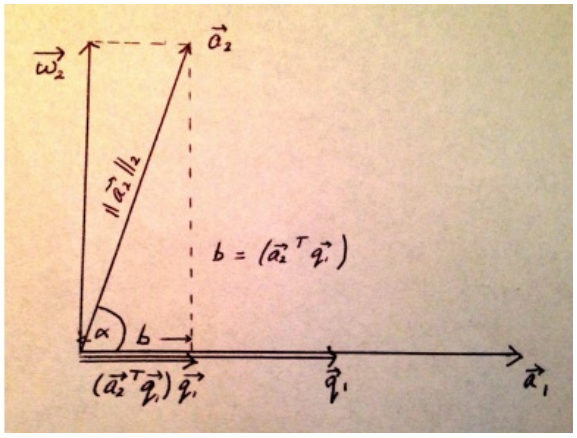


Figure 5.1: Illustration for step 2 of Gram-Schmidt process

Step 3 To find $\vec{q}_3$, we now follow a very similar process: we take $\vec{a}_3$ and subtract from it the components parallel to both $\vec{q}_1$ and $\vec{q}_2$. This gives us a vector $\vec{w}_3$ that is orthogonal to both $\vec{q}_1$ and $\vec{q}_2$. We now set $\vec{q}_3 = \vec{w}_3/\|\vec{w}_3\|$. We find

$$\vec{w}_3 = \vec{a}_3 - (\vec{a}_3^T \vec{q}_1)\vec{q}_1 - (\vec{a}_3^T \vec{q}_2)\vec{q}_2,$$
$$\vec{q}_3 = \frac{\vec{w}_3}{\|\vec{w}_3\|}.$$

Steps 4 through p All subsequent steps are done in the same way. To find $\vec{q}_k$ for $4 \leq k \leq p$, we set

$$\vec{w}_k = \vec{a}_k - \sum_{j=1}^{k-1} (\vec{a}_k^T \vec{q}_j)\vec{q}_j,$$
$$\vec{q}_k = \frac{\vec{w}_k}{\|\vec{w}_k\|}.$$

5.2.2 QR decomposition

In Gaussian Elimination (GE), we created a LU decomposition of A by taking linear combinations of rows. GE could be interpreted as a matrix-matrix operation: we found a matrix C that gives $CA = U$, and then wrote $L = C^{-1}$ to get $A = LU$. We premultiplied A by C because GE involves row manipulations.

In the Gram-Schmidt (GS) process we perform column manipulations, not row manipulations. It is natural to ask ourselves if we perhaps can express GS as a matrix-matrix operation that involves post-multiplication of A by some matrix S such that $AS = Q$. Do you see why we are now looking at post multiplication, not premultiplication as in GE? It's because we are taking linear combinations of columns, not of rows.

Nonsingular A We will first look at the case for A square and nonsingular. Then, $AS = Q$, with both A and Q nonsingular, means that S is nonsingular (check this). We set $R = S^{-1}$, and get $A = QR$. We can find an expression for R explicitly: $Q^T A = Q^T QR = R$. We used here that $Q^{-1} = Q^T$. So, it seems that the elements of R can readily be found as

$$r_{ij} = \vec{q}_i^T \vec{a}_j.$$

This does not look so very interesting, until you realize that R is actually an upper triangular matrix. The elements r_{ij} of the matrix R are given by $r_{ij} = \vec{q}_i^T \vec{a}_j$. In the previous subsection, we derived the relations between the vectors $\vec{q}_i$ and the vectors $\vec{a}_j$. We know, for example, that $\vec{a}_1$ is a multiple of $\vec{q}_1$ and that $\vec{a}_2$ is a linear combination of $\vec{q}_2$ and $\vec{q}_1$. In general, $\vec{a}_j$ is a linear combination of $\vec{q}_1$ through $\vec{q}_j$. The lower triangular elements of R are those for which $i > j$. For these elements, we have $r_{ij} = \vec{q}_i^T \vec{a}_j = 0$ because each $\vec{q}_i$ with $i > j$ is orthogonal to all vectors $\vec{q}_j, j = 1, \cdots, j$.

The diagonal elements of R are equal to

$$r_{ii} = \vec{q}_i^T \vec{a}_i,$$

so that

$$r_{11} = \vec{q}_1^T \vec{a}_1 = \|\vec{a}_1\|,$$
$$r_{22} = \vec{q}_2^T (\vec{w}_2 + (\vec{a}_2^T \vec{q}_1)\vec{q}_1) = \vec{q}_2^T \vec{w}_2 = \|\vec{w}_2\|,$$

and we find that in general

$$r_{ii} = \|\vec{w}_i\|, \quad \text{for all } i.$$

In other words, the diagonal of R contains the normalization constants.

The upper triangular elements of R are all the coefficients that appear in the expressions for $\vec{w}_i$ that we have looked at earlier. Isn't this very reminiscent of the matrix L in GE, which contains the coefficients of the GE process?

We have now found the QR decomposition of A: $A = QR$ with Q orthogonal and R upper triangular. Like LU this can be used to effectively compute solutions to $A\vec{x} = \vec{b}$. We set $QR\vec{x} = \vec{b}$ and first solve $Q\vec{z} = \vec{b}$ using $\vec{z} = Q^T\vec{b}$, and then $R\vec{x} = \vec{z}$ using back-substitution.

Singular or rectangular A What would we get in the GS process if the columns of the $n \times n$ matrix were not independent to begin with? We will illustrate this with an example. Take A equal to

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Applying GS will give

$$\vec{q}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \vec{q}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \vec{q}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

It is no surprise that $\vec{q}_3 = \vec{0}$ because the third column of A is dependent on the first two (it's just the summation of these columns). The matrix R is found as

$$R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

As before we could write $A = QR$ with $Q = [\vec{q}_1, \vec{q}_2, \vec{q}_3]$, but does this make any sense? Not really as this matrix Q is not orthogonal. We could instead just write A as

$$A = Q_{skinny}R_{skinny} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix},$$

which is the so-called *skinny QR* decomposition of A. Note that we left out the last row of R when forming R_{skinny} as we left out the irrelevant third vector $\vec{q}_3$ from Q when forming Q_{skinny}. We could also leave the third row of R in and create a new third column for Q to make Q truly orthogonal again as in

$$A = Q_F R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

We call this the *full QR decomposition*. In this decomposition, you have to be careful of course when interpreting Q_F, which seems to signal that A is nonsingular. When you look at R, however, you notice a zero row which indicates that A was not nonsingular in fact and that the third column of Q_F is not in the column space of A. See that?

5.3 Modified GS

In textbooks and other references, you often read about the modified GS algorithm. This is a version of the GS algorithm that is less sensitive to round-off error accummulation. One of these days, I will write about that here. If you are currently enrolled in CME200, it may come up in the lectures.

5.4 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled $\star$ are more challenging.*

Exercise 5.1

Indicate whether the following statement is TRUE or FALSE and motivate your answer clearly. To show a statement is false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

If Q is an orthogonal matrix, then $\|Q\vec{x}\|_2 = \|\vec{x}\|_2$ for any $\vec{x}$ (you may assume that Q and $\vec{x}$ are real).

Exercise 5.2

(a) Using Gram-Schmidt orthogonalization, create an orthonormal basis for $\mathbb{R}^2$ from the vectors

$$\vec{a}_1 = \begin{bmatrix} 3 \\ -4 \end{bmatrix}, \quad \text{and } \vec{a}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

(b) Find the QR decomposition of the matrix

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -4 & 1 & -1 \end{bmatrix},$$

where Q is a 2×2 matrix and R is 2×3.

Exercise 5.3

(a) (i) Use `Matlab` command `A=rand(4)` to generate a random 4-by-4 matrix and then use the function `qr` to find an orthogonal matrix Q and an upper triangular matrix R such that $A = QR$. Compute the determinants of A, Q and R.

(ii) Set `A=rand(n)` for at least 5 different n's in `Matlab` for computing the determinant of Q where Q is the orthogonal matrix generated by `qr(A)`. What do you observe about the determinants of the matrices Q? Show, with a mathematical proof, that the determinant of any orthogonal matrix is either 1 or -1.

(iii) For a square $n \times n$ matrix matrix B, suppose there is an orthogonal matrix Q and an upper-triangular matrix R such that $B = QR$. Show that if a vector x is a linear combination of the first k column vectors of B with $k \leq n$, then it can also be expressed as a linear combination of the first k columns of Q.

*(b) (i) Assume $\{\vec{v}_1, \vec{v}_2 \cdots \vec{v}_n\}$ is an orthonormal basis of $\mathbb{R}^n$. Suppose there exists a unit vector $\vec{u}$ such that $\vec{u}^T \vec{v}_k = 0$ for all $k = 2, 3 \cdots n$, show that $\vec{u} = \vec{v}_1$ or $\vec{u} = -\vec{v}_1$.

(ii) Prove that if C is non-singular and $C = QR$, where Q is an orthogonal matrix and R is an upper-triangular matrix with diagonal elements all positive, then the Q and R are unique.
Hint: Use proof by contradiction.

* Exercise 5.4

Suppose that $A = QR$ is the QR-factorization of the $m \times n$ matrix A, with $m > n$, and $r(A) = n$. Show that the projection $\vec{x}$ of a vector $\vec{b}$ onto the column space of A is given by $R\vec{x} = Q^T\vec{b}$.

Exercise 5.5

The matrix A is given by $A = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix}$. Find a 2×2 orthogonal matrix Q and a 2×3 upper triangular matrix R such that $A = QR$.

Exercise 5.6

Find an orthonormal basis for the column space of the following matrix:

$$A = \begin{bmatrix} 1 & -6 \\ 3 & 6 \\ 4 & 8 \\ 5 & 0 \\ 7 & 8 \end{bmatrix}$$

Chapter 6

Determinants

A determinant is a scalar function of the elements of a matrix A. It can be used to assess singularity of matrix. Determinants also have relevance in coordinate transformations. For small systems, determinants can also be used to find solutions to $A\vec{x} = \vec{b}$ through Cramer's rule (not discussed in these notes).

For us, the most important application of the determinant is for assessment of singularity. The symbol used for the determinant of A is either $|A|$, or $\det(A)$.

6.1 How to compute the determinant

The determinant is a scalar function of the elements of a matrix that helps us decide whether or not a matrix is singular. This logically means that we limit the discussion to square matrices.

We start by looking at the smallest matrix possible: a 1×1 matrix, which is nothing but a scalar a. A scalar is singular if it is equal to 0. The determinant of this 1×1 matrix is therefore just a. If $\det(A) = a = 0$, then this little matrix is declared to be singular.

For a 2×2 matrix A it is a little more involved. What scalar function can we think of now? Let's look at the matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

We know that the matrix is singular if one row is a multiple of the other (or one column a multiple of the other). This is the case if $\frac{a}{c} = \frac{b}{d}$, in other words if $ad = bc$, or $ad - bc = 0$. Note that we could not divide by c and d if they were zero, but we can still use the end result

$ad - bc = 0$ as a test for singularity even if c and d are zero (do you see why?). This scalar function $ad - bc$ of the elements of A is the determinant we are after: $|A| = ad - bc$.

What is interesting is that the factors ad and bc of this determinant each contain exactly one element from each row and one element from each column. We'll see that this is true for the determinant of any $n \times n$ matrix.

For a 3×3 matrix, the determinant calculation is a bit more complex. Take

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & k \end{bmatrix},$$

The determinant of A is now defined as

$$|A| = a \begin{vmatrix} e & f \\ h & k \end{vmatrix} - b \begin{vmatrix} d & f \\ g & k \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}.$$

We see here that each first row element appears separately, multiplied by $+1$ or -1 according to a checkerboard pattern. For the first row element in the j-th column, the 2×2 determinants are determinants of the matrix left after removing the first row and j-th column from A. Again, you can see that $|A|$ is a combination of factors, each of which contains exactly one element from each row and one element from each column of A.

We could have found the determinant also by looping over the second row, or the third row, in a similar matter, and even by looping over columns. The 2×2 determinants are determined in the same way by removing a row and column from A. The only thing we have to be careful of is the checkerboard pattern. For the first row it was $+ - +$. For the second row it is $- + -$ and for the third row again $+ - +$.

For general $n \times n$ matrices, we can do a similar thing. We loop over one of the rows and take each element in turn, multiplied by $+1$ or -1 according to the checkerboard pattern. We multiply each first row element then by the determinant of the $(n-1) \times (n-1)$ matrix left after deleting the row and column to which this element belongs. The $(n-1) \times (n-1)$ determinants can now be applied using the same general rule. This will introduce $(n-2) \times (n-2)$ determinants and we can keep going until we have reached 1×1 determinants.

This general rule can be written as

$$|A| = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} |M_{ij}|,$$

where a_{ij} is the element of A in row i and column j, and M_{ij} is the matrix formed from A by deleting row i and column j.

6.1.1 Properties of the determinant

Determinants have lots of wonderful and useful properties. We discuss the main ones in this section. We illustrate them with 2×2 determinants, but they all apply generally as well.

Property 1

$$\begin{vmatrix} ta & tb \\ c & d \end{vmatrix} = t \begin{vmatrix} a & b \\ c & d \end{vmatrix}.$$

If elements of any row, or column, are multiplied by a constant t, then $|A|$ is multiplied by the same constant.

This property results directly form the fact that each factor of $|A|$ contains exactly one element of each row and each column of A.

Question: What is $|A|$ if A has a zero row or column?

Answer: $|A| = 0$, which follows directly from this property with $t = 0$.

Property 2

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = - \begin{vmatrix} c & d \\ a & b \end{vmatrix}.$$

The determinant chances sign is any 2 rows, or columns, are interchanged.

Question: What is $|A|$ if A has 2 identical rows or columns?

Answer: $|A| = 0$ because this property shows that $|A| = -|A|$ if 2 rows or columns are identical (swap the identical rows or columns, which causes the determinant to change sign, but the matrix is unchanged).

Property 3

$$\begin{vmatrix} a + a' & b + b' \\ c & d \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} + \begin{vmatrix} a' & b' \\ c & d \end{vmatrix}.$$

It follows immediately that in general $|A + B| \neq |A| + |B|$. This is easy to check for yourself with a couple of simple examples.

Property 4

$$\begin{vmatrix} a-\alpha c & b-\alpha d \\ c & d \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} + \begin{vmatrix} -\alpha c & -\alpha d \\ c & d \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix}.$$

The last equality follows from properties 1 and 3.

So, if any multiple of one row, or column, is added to or subtracted from any other row, or column, the determinant is unchanged. This immediately implies that $|A| = \pm|U|$, where U is the matrix formed from A by Gaussian Elimination. The $\pm$ comes from any pivoting: each time a row swap takes place, the determinant changes sign (property 2).

Property 5

$$\begin{vmatrix} a & 0 & 0 \\ d & e & 0 \\ g & h & k \end{vmatrix} = aek.$$

If a matrix is lower, or upper, triangular, the determinant of the matrix is simply the product of its diagonal elements, which can be readily checked from the definition of the determinant.

This immediately implies that the determinant of a matrix A is equal to $\pm \prod$ pivots, as the diagonal of U contains the pivots.

Property 6

$$|AB| = |A||B|.$$

The determinant of a product of matrices, is the product of the determinants of the individual matrices.

We use this without proof.

6.2 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled ⋆ are more challenging.*

Exercise 6.1

Let P be a $n \times n$ projection matrix. Prove that $|P| = \pm 1$.

Exercise 6.2

A is an $n \times n$ matrix. Prove that $|A| = |A^T|$.

Exercise 6.3

Prove that the determinant of an orthogonal matrix Q is given by $|Q| = \pm 1$.

Exercise 6.4

In this exercise, we look at a geometric interpretation of the determinant.

Let $\vec{u} \in \mathbb{R}^2$ and $\vec{v} \in \mathbb{R}^2$ be the columns of a 2×2 matrix A, that is,

$$A = \begin{bmatrix} \vec{u} & \vec{v} \end{bmatrix} = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}.$$

Prove that $|\det(A)|$ is the area of the parallelogram whose vertices are the origin, $\vec{u}$, $\vec{v}$, and $\vec{u} + \vec{v}$ (see Figure 6.1).

* Exercise 6.5

The $m \times m$ tridiagonal matrix A_m is given by

$$\begin{bmatrix} b & c & & & \\ a & b & c & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & c \\ & & & a & b \end{bmatrix}$$

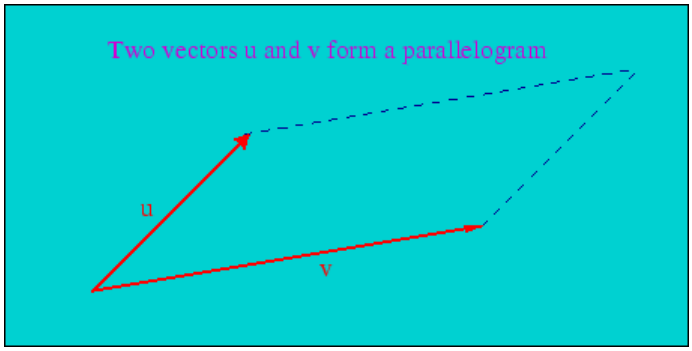


Figure 6.1: The parallelogram formed by two vectors $\vec{u}$ and $\vec{v}$.

We write the determinant of the matrix as $D_m = \det(A_m)$.

(a) Show that $D_m = bD_{m-1} - acD_{m-2}$.

(b) Write the difference equation in 1. as a first-order system of two equations of the form $x^{(m)} = Ax^{(m-1)}$, where $x^{(m)} = \begin{bmatrix} D_m \\ D_{m-1} \end{bmatrix}$.

* Exercise 6.6

(a) Using the system found in part 2 of Exercise 6.5, show that

$$D_m = \frac{r^m}{\sin q} \sin\left((m+1)q\right),$$

where $r = \sqrt{ac}$ and $2r\cos q = b$, given that $ac > 0$ and $\left|\frac{b}{2r}\right| \leq 1$.

(b) If $a_j = \frac{j\pi}{m+1}$ for $j = 1, \ldots, m$, show that

$$I_j = b + 2\sqrt{ac}\cos a_j$$

are the eigenvalues of A_m. Note that you do not need the solution to part 2 of Exercise 6.5 to answer this question.

Exercise 6.7

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false it is sufficient to give one counter example. To show a statement is true, provide a general proof.

(a) If the determinant of an $n \times n$ matrix A is nearly 0, then the matrix is ill-conditioned

(b) For any $n \times n$ unitary matrix U, $\det(U) = \det(U^*)$.

(c) Every $n \times n$ permutation matrix P has a determinant that is equal to either -1 or $+1$.

Chapter 7

Iterative methods

Many problems in engineering and the applied sciences require the solution of one or more systems of linear equations given by

$$A\vec{x} = \vec{b},$$

where A is a $n \times n$ nonsingular matrix with constant coefficients and the unique solution $\vec{x}$ and righthand side vector $\vec{b}$ have n elements. Just in my own work, I've seen such matrix-vector systems come up when simulating fluid flows, when running recommender systems, computing page ranks, searching and browsing, solving optimization problems, running statistical analyses, and many other areas. Often, solving linear system(s) is one of the most expensive part of the solution process. Therefore it is very important to design efficient linear solvers.

You are probably familiar with direct methods. In beginning matrix computation courses, you may have used the inverse A^{-1} of A to find the solution $\vec{x}$: if $A\vec{x} = \vec{b}$ then $\vec{x} = A^{-1}\vec{b}$. You will also have seen Gaussian Elimination (the LU decomposition) and perhaps the QR decomposition. For example, if we know the LU decomposition of A, then we can solve $A\vec{x} = LU\vec{x} = \vec{b}$ in two steps: first for $L\vec{z} = \vec{b}$ using forward substitution, then for $U\vec{x} = \vec{z}$ using back substitution. If we prefer to use QR, then we first solve for $Q\vec{z} = \vec{b}$ using the fact that the inverse of the orthogonal matrix Q is just its transpose, and then $R\vec{x} = \vec{z}$ using back substitution. In either case, we obtain the exact solution $\vec{x}$, that is, we would have the exact solution if rounding errors did not occur.

In these notes we focus on an alternative approach: the iterative methods. In such methods, we seek a decent solution through a series of approximations

$$\vec{x}^{(0)}, \vec{x}^{(1)}, \vec{x}^{(2)}, \vec{x}^{(3)}, \ldots,$$

for which $\vec{x}^{(k)}$ gets closer and closer to the exact solution $\vec{x}$ as k grows. At some point, the approximation is good enough for our purposes and we stop the process. In designing the iterative method you try to minimize the work required to find a reasonable approximation for the problem at hand. The work depends on the number of iteration steps you must perform as well as the work per iteration step. Generally, iterative methods require matrix-vector products and/or solving simple systems of equations in each iteration step.

There are many iterative solvers around. We typically divide them into two classes. The first class is one that relies on a *splitting of the matrix A*. These are what this document is all about. They are also referred to as stationary methods. The second group is the group of search methods (also known as non stationary methods), which are discussed in courses like CME302 (for those of you on Stanford campus).

These notes are by no means complete, but will help you understand the general ideas behind stationary iterative solvers and give you a head start when studying journal papers or advanced books in this area. If you want to read more about the theory of iterative methods, the bible in the field of matrix computations is Golub & van Loan (Matrix Computations), available in most university libraries and online bookstores. This is especially suitable for advanced students. It is not an introductory text book. Two other books that are helpful are Trefethen & Bau (Numerical Linear Algebra) and Demmel (Applied Numerical Linear Algebra). And finally, when you are interested in sparse solvers mostly, check out Davis (Direct Methods for Sparse Linear Systems) and Saad (Iterative Methods for Sparse Linear Systems). This last book is out of print, but is freely available on the internet at http://www-users.cs.umn.edu/ saad/books.html . If you are just about to implement iterative methods, read Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods by Barrett et al. first. It is a SIAM (Society for Industrial and Applied Mathematics) book that is available from the SIAM website (http://www.siam.org) and is very reasonably priced, especially when you are a SIAM member.

Supporting videos can also be found online. Check out the Stationary Iterative Methods jolts on the Mathematics channel of the Jolts pages.

7.1 Stationary Methods - general ideas

Stationary iterative methods, or matrix-split methods, are based on a pretty clever idea. We start by splitting the matrix A as $A = M - N$, with M nonsingular [1]. Now, we take the splitting and substitute it into the equation $A\vec{x} = \vec{b}$:

$$A\vec{x} = \vec{b}, \text{ or } (M - N)\vec{x} = \vec{b}, \text{ or } M\vec{x} = N\vec{x} + \vec{b}.$$

Now the thought occurs that we could try an iteration based on this last equation by simply setting

$$M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b}, \;\; k = 0, 1, \ldots. \tag{7.1}$$

We start the iteration with an initial guess $\vec{x}^{(0)}$ based perhaps on some intuition about the solution. We can then compute the next approximation $\vec{x}^{(1)}$ by solving the system $M\vec{x}^{(1)} = N\vec{x}^{(0)} + \vec{b}$. Note that all terms on the right-hand side of this equation are known. With the new $\vec{x}^{(1)}$, we now compute $\vec{x}^{(2)}$ from the system $M\vec{x}^{(2)} = N\vec{x}^{(1)} + \vec{b}$. Again, the right hand side of this equation is known we solve this matrix-vector equation for the unknown $\vec{x}^{(2)}$. We continue this process until (hopefully) the approximations converge to our desired solution $\vec{x}$. It is clear that we have to require M to be nonsingular, otherwise the solution to the iteration equation 7.1 does not exist.

At each iteration step, we must compute a matrix-vector product $N\vec{x}^{(k)}$ and solve a matrix-vector equation $M\vec{x}^{(k+1)} = \vec{c}$, with $\vec{c} = N\vec{x}^{(k)} + \vec{b}$. We expect to have to perform several iteration steps before we can find a decent approximation. So, this matrix-vector equation should be much easier to solve than the original matrix-vector equation $A\vec{x} = \vec{b}$. But, this is not the only consideration when choosing M. Let's look at an example. To solve the matrix-vector system very easily, we could just choose $M = I$, which gives $\vec{x}^{(k+1)} = (I - A)\vec{x}^{(k)} + \vec{b}$. Cannot be simpler, but unfortunately this does not always lead to a useful series of approximations. Here, we try it on the matrix-vector equation

$$A\vec{x} = \vec{b}, \text{ where } A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \text{ and } \vec{b} = \begin{bmatrix} 7 \\ 5 \end{bmatrix}, \text{ with solution } \vec{x} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

The iteration now gives us

$$\vec{x}^{(k+1)} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} \vec{x}^{(k)} + \begin{bmatrix} 7 \\ 5 \end{bmatrix}.$$

As initial guess $\vec{x}^{(0)}$ we take $\vec{x}^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. The iteration leads to the following series of successive iterates:

$$\vec{x}^{(1)} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}, \; \vec{x}^{(2)} = \begin{bmatrix} -3 \\ -5 \end{bmatrix}, \; \vec{x}^{(3)} = \begin{bmatrix} 15 \\ 13 \end{bmatrix}, \; \vec{x}^{(4)} = \begin{bmatrix} -21 \\ -23 \end{bmatrix}, \ldots$$

and we see that the series of approximations quickly diverges away from the true solution.

[1] It may seem a little strange to write a split as $M - N$ and not $M + N$, but this is simply done so we can write $M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b}$ instead of $M\vec{x}^{(k+1)} = -N\vec{x}^{(k)} + \vec{b}$, which is a bit more involved with the negative sign.

Clearly, care must be taken in choosing the splitting and we immediately have to ask ourselves the following questions:

1. If this iteration converges, will it converge to the solution we are after? In other words, do we need to worry about converging to a wrong solution?
2. For what choices of M and N does the iteration converge?
3. If the system converges, how fast does it converge? In other words, how many iteration steps do we need to take before we have an approximation that is good enough?
4. How do we test for convergence? In other words, how do we check that an approximation is good enough?
5. How do we find an optimal M so that 7.1 is easy to solve and the iteration converges fast?

We address all of these questions below.

7.2 About convergence

7.2.1 If we converge, do we converge to the correct solution?

Before we look at convergence criteria, let's put our mind at ease regarding the second question in the above list: if the iteration converges, do we get the solution we are after? The answer is yes. If we converge to a solution $\vec{x}^*$, then the difference between $\vec{x}^{(k+1)}$ and $\vec{x}^{(k)}$ decreases as k increases and both will get closer and closer to $\vec{x}^*$. In other words,

$$\vec{x}^{(k+1)} \to \vec{x}^* \text{ and } \vec{x}^{(k)} \to \vec{x}^* \text{ as } k \to \infty,$$

so that

$$M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b} \;\to\; M\vec{x}^* = N\vec{x}^* + \vec{b}.$$

This gives $(M-N)\vec{x}^* = \vec{b}$ or $A\vec{x}^* = \vec{b}$. Thus, $\vec{x}^*$ solves the original matrix-vector equation and therefore must be the unique solution $\vec{x}$.

7.2.2 For which choices of M and N do we converge?

To study convergence properties, we typically look at the behavior of the error vector $\vec{e}^k$, which is the difference between the approximation $\vec{x}^k$ and the solution $\vec{x}$, that is $\vec{e}^{(k)} = \vec{x}^{(k)} - \vec{x}$. Behavior of this error is hidden in the iteration equation $M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b}$.

We will rewrite this equation so that instead of $\vec{x}^{(k+1)}$ and $\vec{x}^{(k)}$ we expose $\vec{e}^{(k+1)}$ and $\vec{e}^{(k)}$. It goes as follows:

$$\begin{aligned} M\vec{x}^{(k+1)} &= N\vec{x}^{(k)} + \vec{b}, \\ M\vec{x}^{(k+1)} - M\vec{x} + M\vec{x} &= N\vec{x}^{(k)} - N\vec{x} + N\vec{x} + \vec{b}, \\ M(\vec{x}^{(k+1)} - \vec{x}) &= N(\vec{x}^{(k)} - \vec{x}) + N\vec{x} - M\vec{x} + \vec{b} \\ M\vec{e}^{(k+1)} &= N\vec{e}^{(k)} - A\vec{x} + \vec{b} \\ M\vec{e}^{(k+1)} &= N\vec{e}^{(k)} \\ \vec{e}^{(k+1)} &= M^{-1}N\vec{e}^{(k)} \end{aligned}$$

In the last but one step, we used the fact that the exact solution satisfies $A\vec{x} = \vec{b}$.

The equation for the error is therefore very simple: $\vec{e}(k+1) = M^{-1}N\vec{e}^{(k)}$. We call the matrix $G = M^{-1}N$ the amplification matrix because the new error is simply the previous error amplified by G. Starting with the initial error $\vec{e}^{(0)} = \vec{x}^{(0)} - \vec{x}$, we see that $\vec{e}^{(1)} = G\vec{e}^{(0)}$ and $\vec{e}^{(2)} = G\vec{e}^{(1)} = GG\vec{e}^{(0)} = G^2\vec{e}^{(0)}$. Similarly $\vec{e}^{(3)} = G\vec{e}^{(2)} = GG\vec{e}^{(1)} = GGG\vec{e}^{(0)} = G^3\vec{e}^{(0)}$ and in general

$$\vec{e}^{(k)} = G^k\vec{e}^{(0)},\ k = 1, 2, \ldots. \tag{7.2}$$

The error we make with our initial guess is amplified over and over again by the matrix G during our iteration process. We would like our iteration to converge (our errors to go to zero as the iteration count k goes to infinity) independent of the initial guess $\vec{x}^{(0)}$, that is, independent of the initial error $\vec{e}^{(0)}$. The amplification matrix G controls this convergence.

In section 10.10, we will see that a sufficient and necessary condition for convergence is that the spectral radius $\rho(G)$ of G satisfies $\rho(G) < 1$. The spectral radius of a matrix is its largest eigenvalue in absolute value. What the necessity and sufficiency of this condition indicates is that if the condition is met, we will always converge from any possible initial guess, and if it is not met, we may not. In the small example above in which we chose $M = I$, the spectral radius of G was equal to $\rho(G) = 2$, and so it is no surprise that we were not converging. But don't worry about that now, we will make this clear in later chapters.

We can find other conditions for convergence (that are sufficient, but not necessary). Let's take the norm of 7.2 on both sides. We get

$$\|\vec{e}^{(k)}\| = \|G^k\vec{e}^{(0)}\| \leq \|G^k\|\|\vec{e}^{(0)}\| \leq \|G\|^k\|\vec{e}^{(0)}\|.$$

From this we see that a sufficient criterion for the error to converge to zero is $\|G\| < 1$. Depending on the norm we use this may be quicker to ascertain than the condition on the spectral radius. Note that this matrix norm condition is not a necessary condition. All it says that if we can find a norm for which $\|G\| < 1$, we will have convergence. If we happen to pick a norm for which $\|G\| \geq 1$, we simply can not tell whether we converge or not.

7.2.3 How fast do we converge?

Knowing what we know now about the behavior of the error, this question is relatively simple to answer. If we know that the iteration converges from any initial condition we have either found a norm for which $\|G\| < 1$, or we know that $\rho(G) < 1$. In either case, the size of the norm or the spectral radius also indicates the rate of convergence. The smaller, the faster. Now, how does this translate to correct digits in our solution?

We typically define the rate of convergence as the increase in the number of correct decimal places in the solution per iteration step. We know that

$$\|\vec{e}^{(k+1)}\| \leq \rho(G)\|\vec{e}^{(k)}\|. \text{ or}$$

$$\frac{\|\vec{e}^{(k)}\|}{\|\vec{e}^{(k+1)}\|} \geq (\rho(G))^{-1}, \text{ so}$$

$$\log_{10}\|\vec{x}^{(k)} - \vec{x}\| - \log_{10}\|\vec{x}^{(k+1)} - \vec{x}\| \geq -\log_{10}\rho(G).$$

This equation shows that the increase in the number of correct decimal places when going from step k to $k+1$ is at least as big as $-\log_{10}\rho(G)$. This is what we define to be the rate of convergence $r(G)$. Thus, the smaller $\rho(G)$ the more significant digits we have at each time. If $\rho(G) = 0.1$, we gain at least one significant digit at each iteration, for example.

7.2.4 How do we test for convergence and guide the iteration?

To be effective, a method must know when to stop. A good stopping or convergence criterion should

- identify when the error is small enough to stop
- stop if the error is no longer decreasing or is decreasing too slowly
- limit the maximum amount of time iterating (or the maximum number of iteration steps)

The last two are not so hard to implement. We often put an explicit limit to the number of iteration steps, and typically check to see if the difference between consecutive approximations ($\|\vec{x}^{(k+1)} - \vec{x}^{(k)}\|$) is decreasing. If we see this difference start to increase, we often

stop the iteration and try something else. The tricky part is the first item: identify when the error is small enough.

We have to be a little careful when asking for an error to be "small enough". The absolute value of an error does not mean much. We always have to measure it compared to the norm of the solution. For example, if the exact solution has a norm of 10^{-7}, then getting the error below 10^{-6} does not really mean that much. In other words, we want $\frac{\|\vec{e}^{(k)}\|}{\|\vec{x}\|} < \epsilon$, where ϵ is some error tolerance, not $\|\vec{e}^{(k)}\| < \epsilon$.

There is one snag (well, maybe more than one) when designing stopping criteria: we do not know the error $\vec{e}^{(k)} = \vec{x}^{(k)} - \vec{x}$ because we do not know $\vec{x}$. This is why typically we instead look for convergence in terms of the residual $\vec{r}^{(k)} = \vec{b} - A\vec{x}^{(k)}$. As we converge, this residual will get smaller and smaller. The question is how small should we make it to guarantee that the error in the solution itself is small enough?

Since $A^{-1}\vec{r}(k) = A^{-1}\vec{b} - \vec{x}^{(k)} = \vec{x} - \vec{x}^{(k)} = -\vec{e}(k)$, we know that $\|\vec{e}(k)\| \leq \|A^{-1}\|\|\vec{r}^{(k)}\|$. If a good estimate of $\|A^{-1}\|$ is around, we could use the criterion

$$\|\vec{r}^{(k)}\| \leq \epsilon \|\vec{x}^{(k)}\| / \|A^{-1}\|$$

because then we have that $\|\vec{e}^{(k)}\| \leq \|A^{-1}\|\|\vec{r}^{(k)}\| \leq \epsilon\|\vec{x}^{(k)}\|$, and so $\frac{\|\vec{e}^{(k)}\|}{\|\vec{x}\|} < \epsilon$.

If we cannot use the above criterion because an estimate of $\|A^{-1}\|$ is not available or not sufficiently accurate, we could iterate until the residual is smaller than some tolerance. Again, we need to do this relatively. In this case, we should set the tolerance relative to the norms of A and $\vec{b}$. Typically, what is done therefore is to ask for the residual to satisfy

$$\|\vec{b} - A\vec{x}^{(k)}\| \leq \epsilon \left(\|A\|\|x^{(k)}\| + \|\vec{b}\| \right),$$

where $1 > \epsilon > 0$ is some tolerance set tolerance. If an estimate of $\|A\|$ is not available, we need to simply this even further and often use something like $\|\vec{r}^{(k)}\| \leq \epsilon\|\vec{b}\|$.

How small or large to make ϵ will depend on the accuracy you desire. When I set tolerances, I usually think about errors that are already existent in the model that I am solving. A linear system is typically part of a larger engineering or science problem that in itself was created using mathematical models of reality. Input parameters to the model may be uncertain, some physics may have been left out when formulating the model and other approximations may have been made already in the overall solution process. It does not make much sense to ask for the linear solver to be any more accurate than other errors

that are already made. So, setting ϵ is always problem dependent and it's good to think carefully before you decide as this tolerance controls the number of iteration steps you will perform and therefore the costs of the linear solve.

In summary, a relatively simple control of the iterative method looks something like this:

Algorithm 1 Iterative stationary method with simple control

[2]

k=0
Choose $\vec{x}^{(0)}$ and compute $\vec{r}^{(0)}) = \vec{b} - A\vec{x}^{(0)}$
while $k < step\_limit$ and $\|\vec{r}^{(k)}\| \geq \epsilon \left(\|A\| \|x^{(k)}\| + \|\vec{b}\| \right)$ **do**
  Solve the system $M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b}$ for $\vec{x}^{(k+1)}$
  Compute the residual $r^{(k+1)} = \vec{b} - A\vec{x}^{(k)}$
  Compute $\|\vec{x}^{(k+1)}\|$ and $\|r^{(k+1)}\|$
  k = k+1
end while

Above, we used generic norms. The type of norm you want to use depends often on the problem you are solving. In some of the problems, for example, that I work on, I really like to control $\| \|_2$. In other problems, I'm more interested in $\| \|_\infty$, and other $\| \|_1$. Let's quickly recap these and other norms that were discussed in Section 1.5. For vectors, the norms are defined as

$$\|\vec{x}\|_\infty = \max_i |x_i|,$$
$$\|\vec{x}\|_1 = \sum_i |x_i|,$$
$$\|\vec{x}\|_2 = \left(\sum_i |x_i|^2 \right),$$

[2] In actual implementation you would of course overwrite vectors during the iteration process to limit memory usage

where x_i is the $i-$th element of the vector, with $i = 1, \ldots, n$. Matrix norms we typically consider are

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|, \text{ maximum row-sum}$$
$$\|A\|_1 = \max_j \sum_i |a_{ij}|, \text{ maximum column-sum}$$
$$\|A\|_F = \left(\sum_i \sum_j |a_{ij}|^2 \right), \text{ Frobenius norm}$$
$$\|A\|_2 = \max_{\vec{x} \neq \vec{0}} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2},$$

where a_{ij} is the elements of A in row i and column j, and $i, j = 1, \ldots, n$. All these norms satisfy the triangle inequality ($\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$ and $\|A + B\| \leq \|A\| + \|B\|$ as well as the inequality $\|A\vec{x}\| \leq \|A\|\|\vec{x}\|$, for consistent pairs of vector and matrix norms. The vector 2-norm is consistent with both the matrix 2-norm and the matrix Frobenius norm.

An important difference between these norms is their dependency on the size of the vector or matrix. As n grows we may see the 2-norm grow like $\sqrt{n}$ and the 1-norm like n. This should be taken into account of course when setting tolerances in the convergence criterion.

7.3 How to choose M

We've collected several criteria already for the choice of M. Let's summarize what we have:

- M must be nonsingular
- M must be chosen such that solving the iteration equation $M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b}$ is cheap
- M must be chosen such that $\|G\| = \|M^{-1}N\| < 1$ for some norm. We can also use the condition $\rho(G) < 1$
- M must be chosen such that convergence is as fast as possible.

Choosing the optimal M is really an optimization problem. If we choose M so that the iteration system $M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b}$ is very cheap to solve at each step we often find that the convergence rate is relatively slow. That is, each iteration step is fast, but the total number of iterations required to reach a certain accuracy in the solution is high.

Conversely, we can get matrices M for which only a small number of iteration steps are required, but each step is quite expensive. This trade-off means that it is not always easy to find the optimal splitting. And of course, we also need to satisfy the other criteria on M.

Intuitively, it makes sense to think that the more M resembles A, the fewer iteration steps would be required. The extreme case is taking $M = A$ itself in which case we converge in one iteration (as we simply get back the original system!). Of course this is not very helpful. So, typically we look for a matrix M that contains important information of A but for which the matrix-vector solve is (much) faster than for A.

In the next sections we will see some well-known iterative methods, namely the Jacobi, Gauss-Seidel and SOR methods. These are still being used, but not widely as they only converge for relatively small classes of problems. Nevertheless, they are fun to look at, easy to understand and may give you some ideas on how to invent your own methods.

7.4 The Jacobi Method

For Jacobi, as well as Gauss-Seidel and SOR, we write the matrix A as the sum of three parts: $A = D+L+U$, where D contains the diagonal of A, L is the strictly lower triangular part of A and U the strictly upper triangular part. Beware that the L and U used here are not the same L and U as in Gaussian Elimination.

In the Jacobi method, we take $M = D$. Clearly, we assume here that the diagonal of A does not contain any zeroes, otherwise M would not be invertible and this whole idea would not work. But, certainly this M is easy to invert because M^{-1} can be quickly found as the diagonal matrix with diagonal elements $1/a_{ii}, 1 = 1, \ldots, n$, where a_{ii} are the elements of A.

The iteration equation is then

$$D\vec{x}^{(k+1)} = -(L + U)\vec{x}^{(k)} + \vec{b},$$

or

$$\vec{x}^{(k+1)} = -D^{-1}(L + U)\vec{x}^{(k)} + D^{-1}\vec{b} = D^{-1}\left(\vec{b} - (L + U)\vec{x}^{(k)}\right).$$

Here, $G = -D^{-1}(L + U) = -D^{-1}(A - D) = I - D^{-1}A$.

We could also write this iteration as

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \; i = 1, \ldots, n,$$

where $x_i^{(k)}$ is the i-th element of $\vec{x}^{(k)}$. What is immediately clear from this alternative expression is that Jacobi has one major attraction: all updates $x_i^{(k+1)}$ can be computed independently of each other. $x_i(k+1)$ does not depend on any of the other $x_j^{(k+1)}, j \neq i$ values. This is very attractive for parallel computing as all calculations can be easily divided across compute units/ threads. We call such methods embarrassingly parallel. Parallelization makes each iteration step super fast and even in cases where the convergence rate of Jacobi is low, the total wall clock time spent on the problem could still be attractive.

Let's look at a Jacobi example. We solve

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \vec{x} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \text{ with exact solution } \vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

The Jacobi iteration gives

$$\vec{x}^{(k+1)} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}^{-1} \left(\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \vec{x}^{(k)} \right) = \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} + \begin{bmatrix} 0 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{bmatrix} \vec{x}^{(k)}.$$

In the alternative notation, we have

$$x_1^{(k+1)} = \frac{1}{2} \left(1 + x_2^{(k)} \right), \tag{7.3}$$

$$x_2^{(k+1)} = \frac{1}{2} \left(x_1^{(k)} + x_3^{(k)} \right), \tag{7.4}$$

$$x_3^{(k+1)} = \frac{1}{2} \left(1 + x_2^{(k)} \right). \tag{7.5}$$

If we start with the initial guess $\vec{x}^{(0)} = \vec{b}$, for instance, we get the following sequence of approximations:

$$\vec{x}^{(1)} = \begin{bmatrix} 1/2 \\ 1 \\ 1/2 \end{bmatrix}, \vec{x}^{(2)} = \begin{bmatrix} 1 \\ 1/2 \\ 1 \end{bmatrix}, \vec{x}^{(3)} = \begin{bmatrix} 3/4 \\ 1 \\ 3/4 \end{bmatrix}, \vec{x}^{(4)} = \begin{bmatrix} 1 \\ 3/4 \\ 1 \end{bmatrix}, \vec{x}^{(5)} = \begin{bmatrix} 7/8 \\ 1 \\ 7/8 \end{bmatrix}, \text{ etc.}$$

The pattern is clear and the sequence seems to converge to the exact solution.

In this case, perhaps it is intuitive that the solution converges. The diagonal of A is quite heavy in that the largest elements of A can be found there. In other words, it seems that M really has a significant part of A in it. We can use the General Convergence Theorem also to check for convergence. The spectral radius of the amplification matrix is $\rho(G) = \sqrt{(1/2)}$, which is smaller than 1. So indeed, convergence is guaranteed.

Jacobi does not always converge though, and this is clear from this next example. Now, we take

$$\begin{bmatrix} -1 & 2 & 0 \\ 2 & -1 & 2 \\ 0 & 2 & -1 \end{bmatrix} \vec{x} = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}, \text{ with exact solution } \vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

If we again start with the initial guess $\vec{x}^{(0)} = \vec{b}$, we get the following sequence of approximations:

$$\vec{x}^{(1)} = \begin{bmatrix} 5 \\ 1 \\ 5 \end{bmatrix}, \vec{x}^{(2)} = \begin{bmatrix} 1 \\ 17 \\ 1 \end{bmatrix}, \vec{x}^{(3)} = \begin{bmatrix} 33 \\ 1 \\ 33 \end{bmatrix}, \vec{x}^{(4)} = \begin{bmatrix} 1 \\ 129 \\ 1 \end{bmatrix}, \ldots,$$

which quickly diverges. Choosing a different initial guess does not help much. For example, $\vec{x}^{(0)} = [1, 0, 0]^T$ gives

$$\vec{x}^{(1)} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \vec{x}^{(2)} = \begin{bmatrix} -3 \\ -7 \\ -3 \end{bmatrix}, \vec{x}^{(3)} = \begin{bmatrix} -15 \\ -15 \\ -15 \end{bmatrix}, \ldots,$$

and again this diverges.

We should be able to see this from the amplification matrix. For this last example, the amplification matrix is given by

$$G = \begin{bmatrix} 0 & 2 & 0 \\ 2 & 0 & 2 \\ 0 & 2 & 0 \end{bmatrix},$$

with a spectral radius of $\rho(G) = \sqrt{8}$, which does not bode well for convergence.

7.4.1 Proof of convergence for strictly diagonally dominant matrices

We've reasoned that it makes sense to choose M so that it contains a hefty part of A. It may not be very surprising then to hear that Jacobi converges if A is diagonally dominant,

that is, if the elements of the diagonal of A are larger in absolute value than the combined sum of the off-diagonal elements in either the rows or the columns. More precisily it can be shown that the Jacobi method always converges if the matrix A is strictly diagonally dominant.

Let's first look at the situation where A is row strictly diagonally dominant, that is, when

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|,\ i = 1, \ldots, n.$$

What this says is that each diagonal element a_{ii} of A is larger in absolute value than the combined sum of all other elements in the $i-$th row. The amplification matrix G is given by $G = I - D^{-1}A$. This matrix has zeros on the diagonal and its elements on the off-diagonal are given $a_{ij}/a_{ii}, j \neq i$ (all off-diagonal elements in each row are scaled by the corresponding diagonal element). Now, when we try to show convergence, we either look at the spectral radius condition $\rho(G) < 1$, or we find some norm for which $\|G\| < 1$. Because of the given relation for the elements on each row, it makes a lot of sense to try the maximum norm of G, which is the maximum row sum:

$$\|G\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |g_{ij}| = \max_{1\leq i\leq n} \sum_{j=1, j\neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|}.$$

We see that because A is row strictly diagonally dominant, $\|G\|_\infty < 1$, and we have proved convergence. Note that this works so nicely because the elements of G are given just in the right way with a_{ii} in the denominator, making all elements of G nice and small.

When A is column strictly diagonally dominant, it is a little trickier. Now we now that

$$|a_{jj}| > \sum_{i=1, i\neq j}^{n} |a_{ij}|,\ j = 1, \ldots, n.$$

You may be tempted to simply change the norm to the 1-norm of G, which is given by the maximum column sum. However, as you will quickly find out when you try this, this won't work. Remember that in G the off-diagonal elements $g_{ij}, i \neq j$ are the off-diagonal elements of A, $a_{ij}, i \neq j$, scaled by the diagonal element a_{ii}. In other words, the scaling is row-wise, not column wise. For example, in the second column, we would have elements $g_{12} = a_{12}/a_{11}$ and $g_{32} = a_{32}/a_{33}$ and because these are divided by different diagonal elements, we can not go through a similar reasoning as above. See this?
It would have been much easier if G was given instead by $G = I - AD^{-1}$ which would have scaled all columns of A by its diagonal elements instead of all rows. So, somehow, we'd

like to work with the $1-$norm of this matrix. We can do that if instead of $\|G\|_1$, we use the induced norm $\|G\|_D$ that we define as

$$\|G\|_D = \max_{\vec{x}\neq 0} \frac{\|DG\vec{x}\|_1}{\|D\vec{x}\|_1}.$$

Now $DG = DGD^{-1}D$, with $DGD^{-1} = I - AD^{-1}$, the matrix we liked so much. Setting $D\vec{x} = \vec{y}$, we get

$$\|G\|_D = \max_{\vec{y}\neq 0} \frac{\|(I - AD^{-1})\vec{y}\|_1}{\|\vec{y}\|_1} = \|I - AD^{-1}\|_1 = \max_{1\leq j\leq n} \sum_{i=1, i\neq j}^{n} \frac{|a_{ij}|}{|a_{jj}|}.$$

Now, the fact that A is column strictly diagonally dominant gives us the desired result $\|G\|_D < 1$, and we have convergence. You see from this proof that sometimes you have to play a bit with norms. It's a fun proof.

7.5 The Gauss–Seidel method

In the Jacobi iteration, we chose $M = D$ and $N = -(L + U)$, where $A = D + L + U$. The first example in the previous subsection gives for Jacobi:

$$x_1^{(k+1)} = \frac{1}{2}\left(1 + x_2^{(k)}\right), \tag{7.6}$$

$$x_2^{(k+1)} = \frac{1}{2}\left(x_1^{(k)} + x_3^{(k)}\right), \tag{7.7}$$

$$x_3^{(k+1)} = \frac{1}{2}\left(1 + x_2^{(k)}\right). \tag{7.8}$$

Let's have a closer look at this. If we computed the elements of $\vec{x}^{(k+1)}$ sequentially with $x_1^{(k+1)}$ first, then $x_2^{(k+1)}$ and then $x_3^{(k+1)}$, we could immediately use $x_1^{(k+1)}$ instead of $x_1^{(k)}$ in the computation of $x_2^{(k+1)}$, and similarly $x_2^{(k+1)}$ instead of $x_2^{(k)}$ in the calculation of $x_3^{(k+1)}$. In other words, as soon as we compute a new element, we could use it to find the next element. Then, the process would look like

$$x_1^{(k+1)} = \frac{1}{2}\left(1 + x_2^{(k)}\right), \tag{7.9}$$

$$x_2^{(k+1)} = \frac{1}{2}\left(x_1^{(k+1)} + x_3^{(k)}\right), \tag{7.10}$$

$$x_3^{(k+1)} = \frac{1}{2}\left(1 + x_2^{(k+1)}\right). \tag{7.11}$$

Now, of course this would mean that we loose the easy parallelism of Jacobi, but it is hoped that the extra time needed per iteration step is off-set by a higher convergence rate.

In general form, Jacobi is written as

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \; i = 1, \ldots, n,$$

or

$$D\vec{x}^{(k+1)} = -(L+U)\vec{x}^{(k)} = \vec{b}.$$

In the new approach we get

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right), \; i = 1, \ldots, n,$$

or

$$(D+L)\vec{x}^{(k+1)} = -U\vec{x}^{(k)} + \vec{b}.$$

Do you see why on the right hand side of the equation, the element $a_{ij}, j < i$ are associated with $x_j^{(k+1)}$?

This new approach is known as the Gauss-Seidel iteration. Clearly in Gauss-Seidel $M = D + L$ and $N = -U$. Even though the Gauss-Seidel iteration equation $(D+L)\vec{x}^{(k+1)} = -U\vec{x}^{(k)} + \vec{b}$ is harder to solve than the Jacobi iteration equation $D\vec{x}^{(k+1)} = -(L+U)\vec{x}^{(k)} + \vec{b}$, it is still relatively easy as it just requires a forward substitution. Also, it is hoped that this extra expense at each step of Gauss-Seidel is offset by a higher convergence rate so that the total computational costs associated with the iteration are lower than for Jacobi.

Looking at the example

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \vec{x} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \text{ with exact solution } \vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

we get for Gauss-Seidel with initial guess chosen equal to $\vec{b}$

$$\vec{x}^{(1)} = \begin{bmatrix} 1/2 \\ 3/4 \\ 7/8 \end{bmatrix}, \vec{x}^{(2)} = \begin{bmatrix} 7/8 \\ 7/8 \\ 15/16 \end{bmatrix}, \vec{x}^{(3)} = \begin{bmatrix} 15/16 \\ 29/32 \\ 61/64 \end{bmatrix}, \text{ etc.}$$

This sequence clearly converges to the desired solution. it is easy to check that the norm of the amplification matrix in this case is, as desired, smaller than 1.

It can be shown that the Gauss-Seidel method also converges for diagonally dominant matrices, as well as for symmetric matrices that are also positive definite, a property we will learn about later.

7.6 The Successive Over Relaxation (SOR) method

We could be a bit more creative and not take all of D, L, or U in either M or N. One possibility is to take

$$M = \alpha D + L,\ N = -(1-\alpha)D - U, \quad \alpha \in \mathbb{R},$$

with the standard iterative scheme $M\vec{x}^{k+1} = N\vec{x}^k + \vec{b}$. The corresponding amplification matrix is once again $G = M^{-1}N$.

In most textbooks, the iterative scheme is given in the slightly reworked version

$$(D + \omega L)\vec{x}^{k+1} = \left[(1-\omega)D - \omega U\right]\vec{x}^k + \omega\vec{b},$$

with $\omega = 1/\alpha$.

The idea is that you can "twiggle" α to improve the convergence rate of the method. This idea of introducing a free parameter that than can be optimized for is very common in applied mathematics and engineering.

In Section 10.10, you will be introduced to the General Convergence Theorem. It gives a useful criterion for convergence: the convergence rate is given by the spectral radius of $G = M^{-1}N$. The spectral is the largest eigenvalue (in absolute value) of G. In SOR, G will depend on α. The idea is now that α is chosen to minimize this spectral radius. It can be shown that ω must lie in the interval $(0, 2)$ (so α must be larger than 0.5). In most applications, ω is chosen between 1 and 2. In many engineering applications, the optimal ω lies somewhere between 1.7 and 1.9.

To illustrate SOR, let's apply it to the simple system

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\vec{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \text{ with exact solution } \vec{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

The Jacobi iteration gives

$$\vec{x}^{k+1} = \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix}\vec{x}^k + \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}.$$

Gauss-Seidel gives

$$\vec{x}^{k+1} = \begin{bmatrix} 0 & 1/2 \\ 0 & 1/4 \end{bmatrix} \vec{x}^k + \begin{bmatrix} 1/2 \\ 3/4 \end{bmatrix}.$$

SOR leads to the system

$$\begin{bmatrix} 2 & 0 \\ -\omega & 2 \end{bmatrix} \vec{x}^{k+1} = \begin{bmatrix} 2(1-\omega) & \omega \\ 0 & 2(1-\omega) \end{bmatrix} \vec{x}^k + \begin{bmatrix} \omega \\ \omega \end{bmatrix}.$$

If we start with the initial guess $\vec{x}^0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, we will see that the Gauss-Seidel method is about 1.5 times faster than Jacobi and SOR with a value of $\omega = 1.8$ a bit faster still. Implement this quickly in MATLAB to check. When I apply these methods, I often see SOR converge about twice as fast as Gauss-Seidel, and three or four times faster than Jacobi. This is of course attractive, but there is a price to pay. Do you see which one? There is no such thing as a free lunch!

7.7 Search Methods

Non-stationary methods, also known as search methods, try to improve the approximate solution $\vec{x}^{(k)}$ by adding a suitable correction. The correction vector is usually written as $\alpha^{(k)}\vec{p}^{(k)}$, where $\vec{p}^{(k)}$ is referred to as the search direction, and $\alpha^{(k)}$ is a constant. In this way, we split the search for the correction: we first find a general direction for this correction ($\vec{p}^{(k)}$), and then we figure out how far we want to go in that direction (which is determined by $\alpha^{(k)}$). So, we find a sequence of approximations

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \alpha^{(k)}\vec{p}^{(k)},\ k = 0, 1, 2, \ldots$$

It is a little difficult to find good search directions just be staring at the matrix-vector equations $A\vec{x} = \vec{b}$. It is a lot easier to do this if we had a minimization problem. So, the next idea in search methods is to replace the matrix-vector equation $A\vec{x} = \vec{b}$ with an *equivalent minimization problem*:

$$\min_{\vec{x}} F(\vec{x}),$$

for an appropriate function $F(\vec{x})$.

It is not so hard to find such an equivalent minimization problem if you realize that $\vec{x}$ that solves $A\vec{x} = \vec{b}$ also minimizes the norm of the residual $\|\vec{b} - A\vec{x}\|$ (it makes it zero). When deciding which norm to minimize for, the 2-norm comes to mind because it is just

a quadratic function of $\vec{x}$. Of course, we will square it so that we do not have the square root to think about. So, we could take

$$F(\vec{x}) = \|\vec{b} - A\vec{x}\|_2^2 = (\vec{b} - A\vec{x})^T(\vec{b} - A\vec{x}) = \vec{b}^T\vec{b} - 2\vec{x}^T A^T\vec{b} + \vec{x}^T A^T A\vec{x}.$$

There are quite a few search methods that are based on minimization of the residual.

Another minimization problem that is used for symmetric and positive definite matrices is

$$F(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{x}^T\vec{b},$$

which is a bit simpler. A positive definite matrix will be discussed in more detail in the chapter on eigenvalues and eigenvectors. Here, we just state that for a positive definite matrix, we always have $\vec{x}^T A\vec{x} > 0, \vec{x} \neq \vec{0}$. The minimum of the function F is indeed given by the solution to $A\vec{x} = \vec{b}$. You can convince yourself by computing the gradient of F which is $A\vec{x} - \vec{b}$. Setting this to zero would give the extremum and this is indeed a minimum because of the positive definite character of A. Note that for scalar A this is all very easy to see.

Of course, what we would like for all choices of minimization function, α and search direction $\vec{p}$ is that $F(\vec{x}^{(k+1)}) < F(\vec{x}^{(k)})$ at each iteration step. We need to check this when designing a new method.

Now, suppose that we have, somehow, chosen a search direction $\vec{p}^{(k)}$, and now need to decide on $\alpha^{(k)}$. We have $F(\vec{x}^{(k+1)}) = F(\vec{x}^{(k)} + \alpha^{(k)}\vec{p}^{(k)})$, and we want to make this as small as possible, naturally. We realize that $F(\vec{x}^{(k)} + \alpha^{(k)}\vec{p}^{(k)})$ at this stage is just a function of the unknown $\alpha^{(k)}$, because we assumed that $\vec{p}^{(k)}$ was already chosen. So, to make $F(\vec{x}^{(k+1)})$ as small as possible, we compute $\frac{\partial F(\vec{x}^{(k+1)})}{\partial \alpha^{(k)}}$, and set it to 0 to find the optimal $\alpha^{(k)}$. Assuming that the function F is a reasonable function to differentiate, it is clear that finding $\alpha^{(k)}$ is not so hard once we know $\vec{p}^{(k)}$. Finding $\vec{p}^{(k)}$ is really the biggest challenge. We will look at one method, steepest descent, to illustrate how this could be done.

Overall, I really really really like search methods. They are based on such a clever idea: replacing the matrix-vector equation with an equivalent minimization problem. This opens many doors for new methods and this area of linear algebra has therefore been extremely active in the last decades (and a lot of fun).

7.7.1 Steepest Descent

The steepest descent method is one of the simplest of the search methods. It can be used for symmetric and positive definite matrices and it uses the minimization function

$$F(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{x}^T\vec{b}.$$

We will illustrate this type of function using a simple example. We take the symmetric and positive definite matrix A given by $A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$. We take $\vec{b} = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$. The function F can now be seen to be

$$F(\vec{x}) = \frac{1}{2}\left(3x_1^2 + 6x_2^2 + 4x_1x_2\right) - 2x_1 + 8x_2.$$

A sketch of this function and a sketch of its contour plot is given in the figure.

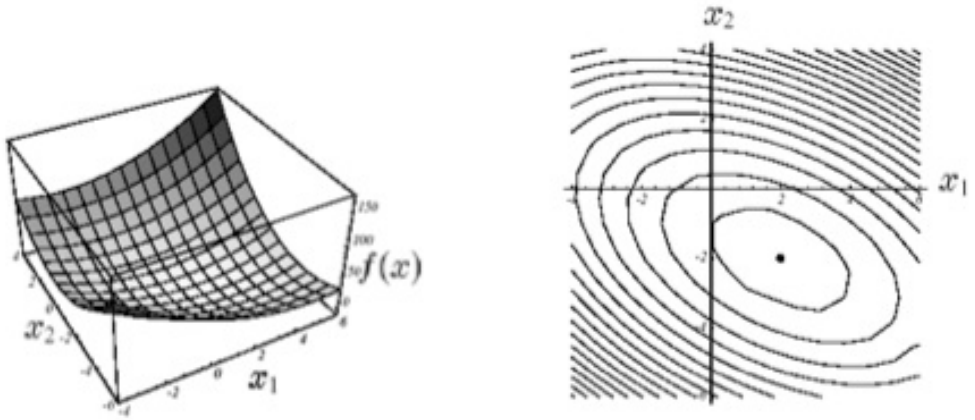


Figure 7.1: A sketch of F and its contour lines

F has its minimum in the point $(2, -2)$, and its shape is a bowl, or more precise a paraboloid. For symmetric positive definite matrices, F always looks like that.

Before we find $\vec{p}^{(k)}$ for each iteration step, we first compute what $\alpha^{(k)}$ will be once we have found $\vec{p}^{(k)}$. We have seen that for that, we must compute $\frac{\partial F(\vec{x}^{(k+1)})}{\partial \alpha^{(k)}}$. We can do this for the example, or in general. Let's do the latter so we can use it for other A as well. We

get

$$\frac{\partial F(\vec{x}^{(k+1)})}{\partial \alpha^{(k)}} = \frac{\partial}{\partial \alpha^{(k)}} \left(\frac{1}{2} (\vec{x}^{(k)} + \alpha^{(k)} \vec{p}^{(k)})^T A (\vec{x}^{(k)} + \alpha^{(k)} \vec{p}^{(k)}) - (\vec{x}^{(k)} + \alpha^{(k)} \vec{p}^{(k)})^T \vec{b} \right) = \alpha^{(k)} (\vec{p}^{(k)})^T A \vec{p}^{(k)} + (\vec{x}^{(k)})^T A \vec{p}^{(k)} - (\vec{p}^{(k)})^T \vec{b}.$$

When we set this to zero, we therefore get

$$\alpha^{(k)} = \frac{(\vec{p}^{(k)})^T (\vec{b} - A\vec{x}^{(k)})}{(\vec{p}^{(k)})^T A \vec{p}^{(k)}} = \frac{(\vec{p}^{(k)})^T \vec{r}^{(k)}}{(\vec{p}^{(k)})^T A \vec{p}^{(k)}},$$

where $\vec{r}$ is the residual. So, no matter what we choose for $\vec{p}^{(k)}$, we will always find that this $\alpha^{(k)}$ gives us the smallest possible $F(\vec{x}^{(k+1)})$. Substituting this value for $\alpha^{(k)}$ back into $F(\vec{x}^{(k+1)}) = F(\vec{x}^{(k)} + \alpha^{(k)} \vec{p}^{(k)})$ gives

$$F(\vec{x}^{(k+1)}) = \frac{1}{2} (\vec{x}^{(k)})^T A \vec{x}^{(k)} - (\vec{x}^{(k)})^T \vec{b} - \frac{1}{2} \frac{((\vec{p}^{(k)})^T \vec{r})^2}{(\vec{p}^{(k)})^T A \vec{p}^{(k)}} = F(\vec{x}^{(k)}) - \frac{1}{2} \frac{((\vec{p}^{(k)})^T \vec{r}^{(k)})^2}{(\vec{p}^{(k)})^T A \vec{p}^{(k)}}.$$

From this, we conclude that indeed $F(\vec{x}^{(k+1)}) < F(\vec{x}^{(k)})$ unless $(\vec{p}^{(k)})^T \vec{r}^{(k)} = 0$. So, we make a note to selves that we should not choose the search direction orthogonal to the residual. By the way, we know that it is always smaller because the term subtracted has a positive numerator, as well as a positive denominator. See that?

Now, what shall we do about $\vec{p}^{(k)}$? From calculus you will remember that the function $F(\vec{x})$ decreases most rapidly in the direction of the negative gradient and that this direction is orthogonal to the contour line at the point under consideration. We already saw that the gradient of $F(\vec{x})$ is equal to $A\vec{x} - \vec{b}$, and so the negative gradient is just the residual $\vec{b} - A\vec{x}$. It seems that a pretty good choice for $\vec{p}^{(k)}$ would be $\vec{p}^{(k)} = \vec{r}^{(k)}$. For sure then the search direction is not orthogonal to the residual, which is what we wanted to avoid. And locally, this is definitely giving the fastest descent. This is of course where the name *steepest descent* comes from.

If we use the residual at each iteration step as the search direction, and α as each step as we have derived originally, and if we start at $\vec{x}(0) = [-2, -2]^T$, then we get the sequence of search paths as given in the figure:

What we see is that we continue along a search direction as long as we keep crossing contour lines with decreasing values of F. At some point, we will hit a contour line tangent, however and then we should stop. If we were to continue, we would start crossing contour lines with increasing values of F. So, at the tangent point, we turn and start searching in the direction orthogonal to the contour line (the negative gradient). Hence the 90 degree angles. In this 2D case, we see that we zigzag quite a few times. Every other search direction is a direction that we already have seen. There is therefore quite a bit of redundancy. If we

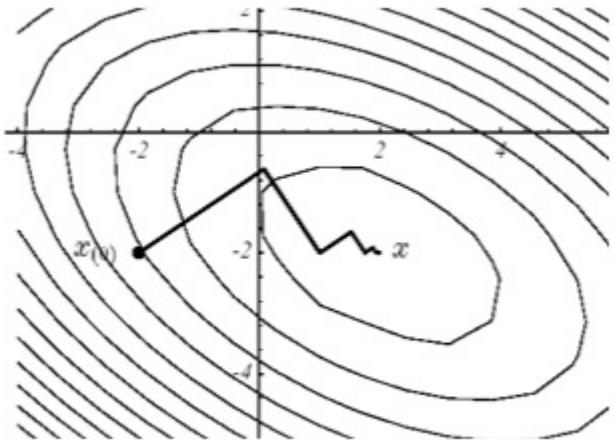


Figure 7.2: A sketch of F and its contour lines

had only moved a bit further along the first search direction, we would not have had to revisit it later. But in steepest descent, we only use local information and there is no way of knowing what the optimal path length in one particular search direction would be in a global sense.

The zigzagging is typically worse as the contour ellipses have a larger ratio between the maximum and minimum radii. We will see in later chapters that this is directly related to the ratio of the largest and smallest eigenvalue of A.

A more robust method that avoids this redundancy is the Conjugate Gradient method, also designed for symmetric and positive definite matrices. Unfortunately, engineering applications do not often lead to symmetric and positive definite matrices. For non symmetric matrices, it is common to minimize the norm of the residual instead of the function used in steepest descent. This then leads to methods like the Generalized Minimum Residual methods (GMRES),which is very popular. We do not discuss these methods in this course, but they are covered well in CME302.

7.8 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled $\star$ are more challenging.*

Exercise 7.1

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

(a) For a non-singular $n \times n$ matrix A, the *Jacobi* method for solving $A\vec{x} = \vec{b}$ will always converge to the correct solution, if it converges.

(b) The *Gauss-Seidel* iteration used for solving $A\vec{x} = \vec{b}$ will always converge for any $n \times n$ invertible matrix A.

Exercise 7.2

(a) Find an invertible 2×2 matrix for which the *Jacobi* method does not converge.

(b) Find an invertible 10×10 non-diagonal matrix for which the *Jacobi* method converges very quickly.

Exercise 7.3

The matrix A given by

$$A = \begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}.$$

Show that neither the Jacobi method nor the Gauss-Seidel method converges when used to solve the equation $A\vec{x} = \vec{b}$, with $\vec{b}$ an arbitrary vector in $\mathbb{R}^2$.

Exercise 7.4

Show that both the Jacobi and Gauss-Seidel iterations converge for the matrix $A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$. Which iteration converges faster, and by what factor?

* Exercise 7.5

One of your friends invented a new iterative scheme for solving the system of equations $A\vec{x} = \vec{b}$ for real and nonsingular $n \times n$ matrices A. The scheme is given by

$$\vec{x}^{(k+1)} = (I + \beta A)\vec{x}^{(k)} - \beta\vec{b}, \qquad \text{with } \beta > 0.$$

(a) Show that **if** this scheme converges, it converges to the desired solution of the system of equations. In other words, your friend seems to be on to something.

(b) Derive an equation for the error $\vec{e}^{(k)} = \vec{x}^{(k)} - \vec{x}^{*}$, where $\vec{x}^{*}$ is the exact solution, for each iteration step k.

Chapter 8

Least Squares Methods

So far, we've mostly looked at solving $A\vec{x} = \vec{b}$ when $\vec{b} \in \mathcal{R}(\mathcal{A})$.

In this chapter, we will look at how to solve $A\vec{x} = \vec{b}$ when $\vec{b} \notin \mathcal{R}(\mathcal{A})$. We assume that A is an $m \times n$ matrix. In this chapter, we will make the assumption that $m \geq n$. We know that the rank $r(A)$ is at most n. Since $\vec{b} \in \mathbb{R}^m$, this means immediately that for $m > n$ there will always be many vectors $\vec{b}$ that are not in $\mathcal{R}(\mathcal{A})$, and for $m = n$ we will similarly have problems if $r(A) < n$. We do not require that A is full column rank in any of the derivations below.

If $\vec{b} \notin \mathcal{R}(\mathcal{A})$, then clearly we cannot solve $A\vec{x} = \vec{b}$ exactly. But, that won't deter us! Instead, we think about how we can find the vector $\vec{x} \in \mathbb{R}^n$ for which $A\vec{x}$ is closest to $\vec{b}$. How should we define closeness? It makes sense to look at $\|\vec{b} - A\vec{x}\|$ as a measure, that is, the norm of the residual. What norm? Well, this is not so hard to answer when we realize that we are trying to *minimize* the norm of the residual, and minimizations are always easiest to do with a norm that is easily differentiable, like the $2-$norm. With this choice, the minimization method is referred to as *least squares*.

So, our goal is to find the vector $\vec{x}$ that minimizes $\|\vec{b} - A\vec{x}\|_2^2$, for a given vector $\vec{b}$. Note that we square the norm here so we do not have to worry about the square root in the definition of the 2-norm. The minimization is just the same.

Where do such problems occur? A very well known application is polynomial fitting of a series of observations. Suppose that we have run several experiments to find out the relation between the variables z_i and w_i. A plot showing the measured data points is given below. The data indicates a linear relation between the two variables. We now try to find the line given by $z = \beta + \alpha w$ that fits the data best. In order to do this, we write

$z_i = \beta + \alpha w_i$, for all observations $i = 1, \ldots, m$. This can also be written in the form

$$\begin{bmatrix} 1 & w_1 \\ 1 & w_2 \\ \vdots & \vdots \\ 1 & w_m \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

The matrix in this example is an $m \times 2$ matrix, with $m > 2$. The columns of A are independent, so A is full rank in this case.

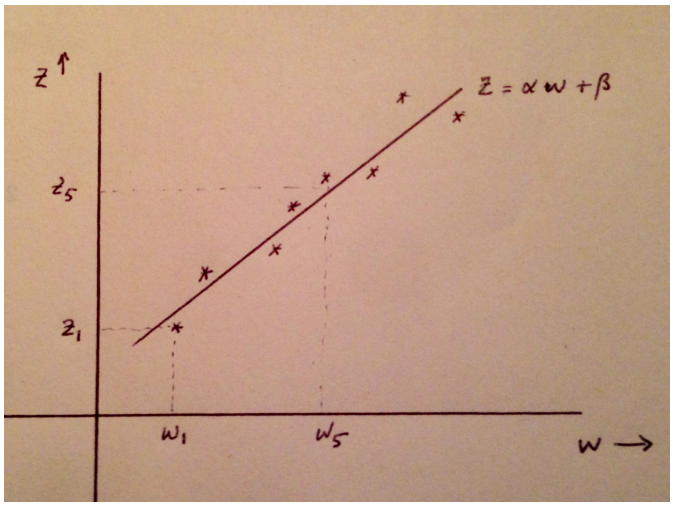


Figure 8.1: Fitting a linear function to a series of data points. Observe the sheer beauty of this picture and the quality of the photographer

We will show that minimization the residual leads to the so-called *normal equations* $A^T A\vec{x} = A^T\vec{b}$. There are two ways we can derive these equations. The first uses algebra, the second geometrical insights.

8.1 Derivation of the normal equations using algebra

We know that

$$\|\vec{b} - A\vec{x}\|_2^2 = (\vec{b} - A\vec{x})^T(\vec{b} - A\vec{x}) = \vec{b}^T\vec{b} - 2\vec{x}^T A^T\vec{b} + \vec{x}^T A^T A\vec{x}.$$

This last expression is simply a function of $\vec{x}$, and a quadratic function at that. We will call it $G(\vec{x})$. We now minimize $G(\vec{x})$ by computing the gradient with respect to $\vec{x}$ and

setting all elements of this gradient to zero. In other words, we find $\vec{x}$ such that

$$\nabla G(\vec{x}) = \begin{bmatrix} \frac{\partial G}{\partial x_1} \\ \vdots \\ \frac{\partial G}{\partial x_m} \end{bmatrix} = \vec{0},$$

with

$$\frac{\partial G}{\partial x_i} = \frac{\partial}{\partial x_i}(\vec{b}^T\vec{b} - 2\vec{x}^T A^T\vec{b} + \vec{x}^T A^T A\vec{x}) = -2(A^T\vec{b})_i + 2(A^T A\vec{x})_i, \quad i = 1, \ldots n.$$

Do you see why we get the given partial derivatives? Note that here $(A^T\vec{b})_i$ is the ith element of the vector $A^T\vec{b}$, and similar for $(A^T A\vec{x})_i$.

Setting all partial derivatives to zero, now gives us

$$-2A^T\vec{b} + 2A^T A\vec{x} = \vec{0}, \quad \text{or} \quad A^T A\vec{x} = A^T\vec{b},$$

which are the desired normal equations.

So, we started with $A\vec{x} = \vec{b}$, which we could not solve because $\vec{b} \notin \mathcal{R}(\mathcal{A})$, and we ended up with $A^T A\vec{x} = A^T\vec{b}$, which apparently we *can* solve. Just multiplying the original unsolvable system with A^T did the trick. Quite remarkable. Even more so when you realize that we did not ask anything special of A: it can be rectangular and it does not have to be full rank. You may be tempted to say: "But these equations are exactly the same. Look, we just need to multiply by the inverse of A^T". But, don't forget that A^T may not have an inverse at all. It may be rectangular, and if it is square it may not be invertible.

The downside of these normal equations is that it requires us to work with the matrix $A^T A$. At first glance, this seems like quite a nice matrix. It is symmetric, and $A^T A\vec{x} = A^T\vec{b}$ could be solved with the symmetric version of LU decomposition known as Cholesky, or with appropriate iterative methods. But typically the condition number of this matrix is not great. For a square A we know that the condition number of $A^T A$ is the square of the condition number of A. If the condition number of A is large then we're in real trouble.

Instead of solving the normal equations directly, we can also use the skinny QR factorization of A. In this factorization R is an $r \times r$ invertible matrix, with $r = r(A)$ the rank of A. Plugging this directly into the normal equations gives us

$$A^T A\vec{x} = R^T Q^T QR\vec{x} = R^T R\vec{x}$$

on the left hand side, and

$$R^T Q^T\vec{b}$$

on the right hand side. Remember that $Q^TQ = I$. So, we get

$$R^TR\vec{x} = R^TQ^T\vec{b},$$

and since R is nonsingular, we can write this as

$$R\vec{x} = Q^T\vec{b},$$

which can be solved by back substitution.

What is really intriguing is that the normal equations are always solvable. That is, we can always find at least one solution to them. This means in turn that the vector $A^T\vec{b}$ must be in the column space of A^TA. Always. This is quite something. $\vec{b}$ was not in the column space of A, but $A^T\vec{b}$ is guaranteed to be in the column space of A^TA, even if A^TA is singular. The proof of this is not straightforward. You can find proofs involving the so-called Fredholm Alternative online (which is not covered in this course). An alternative proof is given in the solutions to the exercises in this chapter.

If A is full column rank, then A^TA will be a nonsingular matrix (this is actually a nice thing to try to prove - try it - I will provide the answer below). Then, the solution can be found as

$$\vec{x} = (A^TA)^{-1}A^T\vec{b}.$$

This matrix $(A^TA)^{-1}A^T$ is referred to as a pseudo-inverse of A. If A is not full column rank, then the normal equations are expected to have many solutions. Do you see why?

And now the wee proof: Show that if A is full column rank, A^TA is nonsingular. First, observe that $r(A) = n$ and that the nullspace of A therefore has dimension 0. Now, we can show that the nullspace of A and of A^TA are exactly the same. Of course, $\mathcal{N}(\mathcal{A}) \in \mathcal{N}(\mathcal{A}^\mathcal{T}\mathcal{A})$, because when $A\vec{x} = \vec{0}$, then also $A^TA\vec{x} = \vec{0}$. If $A^TA\vec{x} = \vec{0}$, can we conclude that $A\vec{x} = \vec{0}$? Note that we cannot simply remove A^T because its inverse does not exist for $m > n$. But, what we can say is that $A^TA\vec{x} = \vec{0}$ gives $\vec{x}^TA^TA\vec{x} = 0$ and this is equal to $\|A\vec{x}\|_2^2 = 0$, and indeed we see that $A\vec{x} = \vec{0}$. So, the nullspaces are the same, and both have dimension 0. But that means that the $n \times n$ matrix A^TA has rank $r = n - 0 = n$ and is therefore nonsingular.

8.2 Derivation of the normal equations using projections

We can derive the normal equations also using projections. I like this approach, because it links several fundamental concepts together, as you will see.

Remember that we are in trouble with $A\vec{x} = \vec{b}$ because $\vec{b}$ is not in $\mathcal{R}(\mathcal{A})$. In the above section, we decided that instead we would solve a minimization problem: if we cannot

get $A\vec{x} = \vec{b}$ exactly, let's get $A\vec{x}$ as close as possible to $\vec{b}$. Hence the minimization of the residual.

We can also look at this another way. The vector $\vec{b}$ is not in the column space $\mathcal{R}(\mathcal{A})$, but we could find the vector $\vec{b}^*$ in $\mathcal{R}(\mathcal{A})$ that is as close as possible to $\vec{b}$ and then solve $A\vec{x} = \vec{b}^*$. That vector $\vec{b}^*$ can be found by projecting $\vec{b}$ orthogonally onto $\mathcal{R}(\mathcal{A})$. May sound strange to project onto a column space, but a column space is just some sort of hyperplane and so we can project on it as we can on any hyperplane.

The figure shows a 3D illustration. We have some 2-dimensional column space, and project $\vec{b}$ onto it. This gives us $\vec{b}^*$, which we set $A\vec{x}$ equal to. The vector $A\vec{x}(= \vec{b}^*)$ is drawn in the figure as well, and also the vector $\vec{b} - A\vec{x}$, which is the difference vector (and equal to the residual). Using an orthogonal projection means that we are minimizing this difference vector, hence minimizing the residual, just as we did above in an algebraic way.

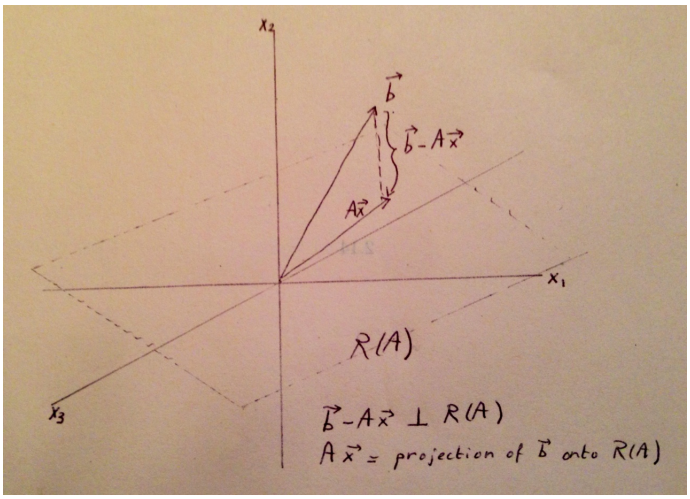


Figure 8.2: Fitting a linear function to a series of data points. Another smashing illustration by yours truly.

Now we reason. The difference vector $\vec{b} - A\vec{x}$ is orthogonal to the column space of A. It is therefore orthogonal to the row space of A^T because the column space of A and the row space of A^T are exactly the same. Now, we remember that every row of a matrix is orthogonal to every vector in the nullspace of that matrix. Do you recall? We discussed this in section 4.11. So, what we now know is that $\vec{b} - A\vec{x}$ is in the nullspace of A^T, hence $A^T(\vec{b} - A\vec{x}) = \vec{0}$, which gives us the normal equations as before.

8.3 A few examples

8.3.1 A small linear fit

We have the three points $(0,0),(1,2),(2,3)$ that are not collinear, and look for the best fit $z = \alpha w + \beta$. We formulate the problem as above, with

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix},$$

with the unknowns

$$\vec{x} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}.$$

Note, that compared to the earlier discussion, we swapped α and β around, so now the first column of A contains all the 1s. We find the QR decomposition of A as

$$A = QR = \begin{bmatrix} \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{3} & \sqrt{3} \\ 0 & \sqrt{2} \end{bmatrix}.$$

We now solve $R\vec{x} = Q^T\vec{b}$ and get $\alpha = 3/2$ and $\beta = 1/6$ (if I did not make any algebra mistakes).

8.3.2 Minimizing the residual

In this example, we find the solution by directly minimizing the residual. We take

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Now, A is not square but it most certainly is not invertible. Also we can see directly that $\vec{b} \notin \mathcal{R}(\mathcal{A})$. We have

$$\|\vec{b} - A\vec{x}\|_2^2 = 5x_1^2 + 5x_2^2 + 10x_1x_2 - 6x_1 - 6x_2 + 2,$$

which you could verify if you wanted to. We now compute

$$\frac{\partial}{\partial x_1}\left(5x_1^2 + 5x_2^2 + 10x_1x_2 - 6x_1 - 6x_2 + 2\right) = 10x_1 + 10x_2 - 6,$$
$$\frac{\partial}{\partial x_2}\left(5x_1^2 + 5x_2^2 + 10x_1x_2 - 6x_1 - 6x_2 + 2\right) = 10x_2 + 10x_1 - 6,$$

and we set these equal to zero. You can see that this will give an infinite number of solutions, all satisfying $x_1 = -x_2 + 3/5$.

8.4 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled ⋆ are more challenging.*

Exercise 8.1

(a) Find the best straight-line fit to the following measurements, and graph your solution:

$$y_1 = 2 \text{ at } t_1 = -1, \quad y_2 = 0 \text{ at } t_2 = 0,$$
$$y_3 = -3 \text{ at } t_1 = 1, \quad y_4 = -5 \text{ at } t_4 = 2,$$

What is the norm of the residual?

(b) Suppose that instead of a straight line, we fit the data above by the parabolic function:

$$y_i = a_2 x_i^2 + a_1 x_i + a_0$$

Derive the over-determined system $A\vec{x} = \vec{b}$ to which least squres could be applied to find this quadratic fit.

(c) Let's look at the general problem of making n observations $y_i, i = 1, 2, \ldots, n$ at n different times t_i. You can extend what you did in the last two parts to find polynomial fits of degree k ($y_i = a_k t_i^k + a_{k-1} t_i^{k-1} + \cdots + a_1 t_i + a_0$) by using least squares. If $k < n-1$, what would the over-determined system $A\vec{x} = \vec{b}$ look like for this general case?

*(d) Prove that for $k = n - 1$, the system $A\vec{x} = \vec{b}$ will no longer be over-determined and we can find a unique fit by solving $A\vec{x} = \vec{b}$ instead of the normal equations.

(e) Consider the systems you solved for in part c and d. For $0 < k < n$, how does the norm of the residual change as we increase k?

Exercise 8.2

(a) From the notes, we know that $P = A\left(A^T A\right)^{-1} A^T$ is an orthogonal projection matrix that projects onto the column space of A. Prove that P is symmetric and $P^2 = P^T P = P$.

*(b) In general, a matrix which satisfies $P^2 = P$ is a projector. Show that if a projector matrix is symmetric, then it is an orthogonal projector.

**(c) Show that regardless of rank of A, the equation $A^T A\vec{x} = A^T\vec{b}$ always has at least one solution

Exercise 8.3

We measured the temperature T below the ground on an unusually cold day a few weeks ago. The temperature was measured as a function of the distance from the ground surface. The outside temperature was a balmy 2 deg. Celsius. The data from the measurements are given in the table below:

Distance (m)	Temperature (C)
0	2.0
5	2.2
10	5.8
15	10.4
20	11.0
25	13.8
30	22.4
35	28.4
40	33.3

(a) Write a matlab function that fits the data to a polynomial of degree n using the method of least squares. Make sure that your function allows you to specify n. (Do not use matlab built-in functions `polyfit` or `polyval` except perhaps to check that your code is correct.) Plot the data. On the same axes, plot the polynomial fit for $n = 1$ and $n = 2$. Be sure to clearly label your fit curves.

(b) Calculate the residual error in your fitted values and the observed data for n ranging from 0 to 8.

Plot the 2-norm of this residual error against n.

Comment on what does this result says about how to choose the order of your polynomial fit.

(c) We improve our drilling and sensing methodology and decide that we can drill to 45m and 50m below ground with minimal effort. We want to estimate the temperature at this new data point.

(i) Provide a table of n versus the predicted temperature at these new data points. It turns out that the temperatures are 48.9 deg. Celsius at 45m and 57.9 deg. Celsius at 50m below ground, respectively.

(ii) Plot the 2-norm of the prediction error only at these two points versus n.

(iii) Comment on what does this result says about how to choose the order of your polynomial fit? Be sure to use what you learned from the previous problem.

* Exercise 8.4

We know that the normal equations for the $m \times n$ matrix A given by $A^T A\vec{x} = A^T\vec{b}$ are solved by the vector $\vec{x}$ for which $\|\vec{b} - A\vec{x}\|_2$ is minimal. Show that if A has full column rank, the normal equations give a unique solution $\vec{x}$.

Chapter 9

Nonlinear Systems of Equations

In this course we mostly focus on solving linear systems of equations. However, many problems in engineering involve the solution of a (large) set of nonlinear equations. These are typically solved using an iterative procedure and the most common one is Newton-Raphson for systems. At each iteration step, this method requires the solution of a linear system. Hence, it is quite appropriate to include nonlinear systems in this class.

An ICME alumnus, Daniel Korenblum, recently wrote a couple of very nice visualization of the Newton-Raphson method in 1D and 2D. You can find these visualizations on LinkedIn: 1D visualization and 2D visualization. If these links fail to work at any one time, please let me know.

9.1 Illustrating example

We will start by giving a small example that we will use as illustration throughout this chapter. In one of the first lectures of the course we talked about a discretization of the 1D steady state heat equation that is a simple model of heat flow in a rod that is laterally insulated (that is, heat can only flow through the rod, in the longitudinal direction). We assumed that the diffusion coefficient for heat was 1 and that the temperatures at each end of the rod were fixed to 0 and 1 respectively. The equation describing this is

$$\frac{d^2T}{dx^2} = 0, \quad x \in [0,1], \quad T(0) = 0, T(1) = 1,$$

To solve this numerically, we divided the domain into N equally spaced intervals $[x_{i-1}, x_i]$, where $i = 1, \ldots, N$. The interval width, also referred to as the grid step size, is equal

to $h = 1/N$. We then replaced the second order derivative in the heat equation with an approximation that was derived using Taylor series:

$$\frac{d^2T(x_i)}{dx^2} \approx \frac{T_{i-1} - 2T_i + T_{i+1}}{h^2}.$$

Here, T_i is the approximated temperature in the point x_i. The unknowns here are T_1 through T_{N-1} because T_0 and T_N are known from the boundary conditions. The resulting system of equations is

$$\begin{aligned} T_0 - 2T_1 + T_2 &= 0, \\ T_1 - 2T_2 + T_3 &= 0, \\ &\vdots \\ T_{N-2} - 2T_{N-1} + T_N &= 0, \end{aligned}$$

or, substituting the known boundary conditions in the first and last equation,

$$\begin{aligned} -2T_1 + T_2 &= 0, \\ T_1 - 2T_2 + T_3 &= 0, \\ &\vdots \\ T_{N-2} - 2T_{N-1} &= -1, \end{aligned}$$

which can be written as $A\vec{T} = \vec{b}$, with

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ 0 & \dots & 0 & 1 & -2 & 1 \\ 0 & \dots & 0 & 0 & 1 & -2 \end{bmatrix}, \vec{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix}.$$

Now, we make the equation a bit more exciting and include a diffusion coefficient K that is a function of temperature itself. We choose $K(T) = T$, so that heat flows faster as the temperature increases. The equation now changes. It is not important to understand this derivation for this class, so we just give it directly. It is

$$\frac{d}{dx}K(T)\frac{dT}{dx} = 0, \quad x \in [0,1], \quad T(0) = 0, T(1) = 1,$$

A standard discretization of this equation is of the form

$$\frac{d}{dx}K(T)\frac{dT}{dx} \approx \frac{-K(T_i)(T_i - T_{i-1}) + K(T_{i+1})(T_{i+1} - T_i)}{h^2}.$$

For constant K this reduces to the approximation we had before.

We will take $N = 4$, and we get the following equations (note that we immediately substitute in the given boundary conditions)

$$\begin{aligned} -T_1^2 + T_2(T_2 - T_1) &= 0, \\ -T_2(T_2 - T_1) + T_3(T_3 - T_2) &= 0, \\ -T_3(T_3 - T_2) - T_3 &= -1, \end{aligned}$$

or

$$\begin{aligned} -T_1^2 + T_2^2 - T_1T_2 &= 0, \\ -T_2^2 + T_3^2 + T_1T_2 - T_2T_3 &= 0, \\ -T_3^2 + T_2T_3 - T_3 &= -1. \end{aligned}$$

This is a system of three equations in the three unknowns T_1, T_2 and T_3, but the equations are now nonlinear, not linear. We will solve this system with the most commonly used nonlinear solver: The Newton-Raphson method. Other nonlinear solvers are available (such as Wolfe's method), but seldom used. Newton-Raphson (NR) is the method of choice.

Before we introduce NR for systems, we will revisit NR for scalar nonlinear equations.

9.2 Newton-Raphson for a scalar nonlinear equation

Suppose that we are asked to solve the nonlinear equation $f(x) = 0$ with solution x, which is referred to as the root of the function. We will try to find an approximate solution to this problem using an iterative method. The initial guess is some $x^{(0)}$. In $x^{(0)}$, we approximate the function $f(x)$ by its tangent, that is, by the line that goes through $(x^{(0)}, f(x^{(0)}))$, and has slope equal to $f'(x^{(0)})$. This line is given by

$$h^{(0)}(x) = f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)}).$$

The linear function $h^{(0)}(x)$ is exactly equal to the first two terms of the Taylor series of $f(x)$ about $x^{(0)}$. The idea is now that instead of solving $f(x) = 0$, we solve $h^{(0)}(x) = 0$. The x for which this equation holds is our next approximation $x^{(1)}$. It is given by

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})}.$$

We now repeat the process: we approximate the function by its tangent line $h^{(1)}(x)$ through the new point $(x^{(1)}, f(x^{(1)}))$, find the x for which $h^{(1)}(x)$ crosses the x-axis and use this

as a next approximation $x^{(2)}$, and we keep going. So, what we really do is "find an approximation to the root of a function through a sequence of roots of local approximations to the function" if you catch my drift. The figure gives an illustration.

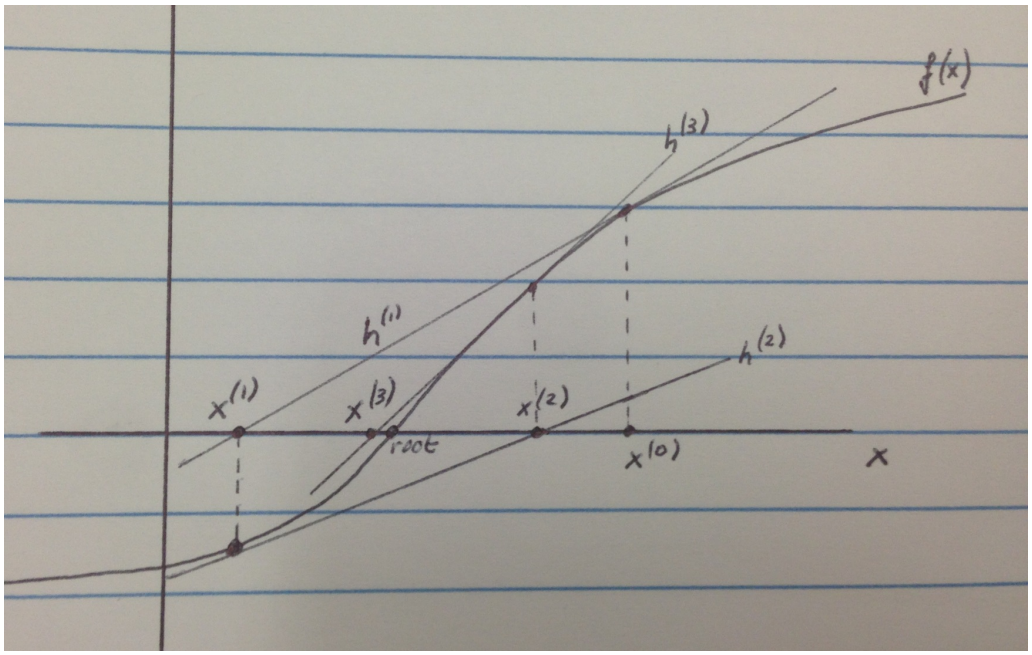


Figure 9.1: Illustration of a few steps of the NR process for a nonlinear scalar equation $f(x) = 0$

We have created the iteration

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}. \tag{9.1}$$

The iterative algorithm is given below.

Algorithm 2 Newton-Raphson for a scalar nonlinear equation $f(x) = 0$

k=0
Choose $x^{(0)}$
while not converged **do**
  Compute $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$.
  Check for convergence
  k = k+1
end while

As you (hopefully) will remember from previous math courses, convergence of this NR iteration is not guaranteed and depends strongly on the initial guess $x^{(0)}$. If the NR iteration does converge, the convergence is quadratic sufficiently close to the root. It is common in the case of scalar equations to first use a more robust, but slower iterative method, such as bisection, to get closer to the root and then switch to NR.

9.3 NR for systems of nonlinear equations

The NR method can be generalized to systems of nonlinear equations quite easily. We will use the system

$$\begin{aligned} -T_1^2 + T_2^2 - T_1T_2 &= 0, \\ -T_2^2 + T_3^2 + T_1T_2 - T_2T_3 &= 0, \\ -T_3^2 + T_2T_3 - T_3 &= -1. \end{aligned}$$

as a guiding example. We first rewrite this system as

$$\vec{f}(\vec{T}) = \begin{bmatrix} f_1(\vec{T}) \\ f_2(\vec{T}) \\ f_3(\vec{T}) \end{bmatrix} = \vec{0}, \text{ with } \vec{T} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix},$$

and with

$$\begin{aligned} f_1(\vec{T}) &= -T_1^2 + T_2^2 - T_1T_2, \\ f_2(\vec{T}) &= -T_2^2 + T_3^2 + T_1T_2 - T_2T_3, \\ f_3(\vec{T}) &= -T_3^2 + T_2T_3 - T_3 + 1. \end{aligned}$$

We will once again, just as in the scalar case, use the first two terms of the Taylor expansion to approximate the function $\vec{f}(\vec{T})$. We now need to use the multi-dimensional Taylor series expansion. For each $f_i(\vec{T})$, the first two terms of the Taylor series about $\vec{T}^{(k)}$ are given by

$$h_i^{(k)} = f_i(\vec{T}^{(k)}) + (T_1 - T_1^{(k)})\frac{\partial f_i(\vec{T}^{(k)})}{\partial T_1} + (T_2 - T_2^{(k)})\frac{\partial f_i(\vec{T}^{(k)})}{\partial T_2} + (T_3 - T_3^{(k)})\frac{\partial f_i(\vec{T}^{(k)})}{\partial T_3}.$$

As before, we set these approximations to zero. In other words, we solve $\vec{h}^{(k)}(\vec{T}) = \vec{0}$, rather than $\vec{f}(\vec{T}) = \vec{0}$. The solution gives us $\vec{T}^{(k+1)}$. After a bit of reshuffling, we can write this system in matrix-vector notation as :

$$\begin{bmatrix} \partial f_1/\partial T_1 & \partial f_1/\partial T_2 & \partial f_1/\partial T_3 \\ \partial f_2/\partial T_1 & \partial f_2/\partial T_2 & \partial f_2/\partial T_3 \\ \partial f_3/\partial T_1 & \partial f_3/\partial T_2 & \partial f_3/\partial T_3 \end{bmatrix} \begin{bmatrix} (T_1^{(k+1)} - T_1^{(k)}) \\ (T_2^{(k+1)} - T_2^{(k)}) \\ (T_3^{(k+1)} - T_3^{(k)}) \end{bmatrix} = \begin{bmatrix} f_1(\vec{T}^{(k)} \\ f_2(\vec{T}^{(k)} \\ f_3(\vec{T}^{(k)} \end{bmatrix}.$$

The matrix in this expression is referred to as the Jacobian matrix J. It changes at each iteration step, so we denote it by $J^{(k)}$. Plugging in the partial derivatives for our example, we get

$$J^{(k)} = \begin{bmatrix} -2T_1^{(k)} - T_2^{(k)} & 2T_2^{(k)} - T_1^{(k)} & 0 \\ T_2^{(k)} & -2T_2^{(k)} + T_1^{(k)} - T_3^{(k)} & 2T_3^{(k)} - T_2^{(k)} \\ 0 & T_3^{(k)} & -2T_3^{(k)} + T_2^{(k)} - 1 \end{bmatrix}$$

In short hand notation, we therefore have the following iteration

$$J^{(k)}(\vec{T}^{(k+1)} - \vec{T}^{(k)}) = -\vec{f}^{(k)}. \tag{9.2}$$

This may not look similar to the scalar iteration 9.1 in this form, but we can rewrite this matrix-vector equation as

$$\vec{T}^{(k+1)} = \vec{T}^{(k)} - (J^{(k)})^{-1}\vec{f}^{(k)},$$

which is very similar. In the scalar case, we divide by the derivative and in the system case we multiply by the inverse of the Jacobian. We have assumed here of course that the Jacobian is invertible.

In actual applications, we do not compute the inverse of the Jacobian, but solve the iteration 9.2 using our favorite direct or iterative method. In many applications, the Jacobian matrix is sparse. It is certainly not always symmetric and may also be ill-conditioned.

The algorithm for Newton-Raphson for systems of nonlinear equations is given below.

Algorithm 3 Newton-Raphson for a system of nonlinear equations $\vec{f}(\vec{x}) = \vec{0}$

```
k=0
Choose x^(0)
while not converged do
    Construct J^(k)
    Solve J^(k) y = -f^(k)
    Set x^(k+1) = x^(k) + y
    Check for convergence
    k = k+1
end while
```

Some remarks:

- Again, the convergence criterion used must be designed carefully. There is no point in requiring the error in the NR solution to be smaller than errors made elsewhere. In our example, we would certainly use a tolerance that is not any smaller than the discretization errors we are already making in solving the given differential equation.

- Again, convergence depends on the initial guess. In many applications, NR is used in solving time-dependent problems. Then, a good initial guess is generally provided by the solution at the old time value, and convergence is generally not an issue provided that the time step between states is chosen sufficiently small. If not, the iteration may diverge. Generally NR implementations have a convergence check that stops the program if the solution starts to diverge during the NR iteration.
- Computing the Jacobians (a new one at each iteration step!) is often an expensive part of the NR iteration. It is not always possible to derive Jacobians analytically as we did in our example. Sometimes the partial derivatives in the Jacobian must be found approximately using, for example, a finite difference discretization. In many applications, it also works well to leave the Jacobian unchanged for a couple of iteration steps. Convergence is than a little slower, but a lot of Jacobian evaluations can be avoided, which makes it computationally attractive. This is often referred to as *modified NR* in the literature.
- If an iterative method is used to solve for the Jacobian equation $J^{(k)}\vec{y} = -\vec{f}^{(k)}$, it is very common to keep the error tolerance for that iterative method relatively large. This reduces the computational cost per NR iteration step. It may increase the total number of NR steps required, but again, the total costs are generally lower because the average work load per NR step is reduced. Really, finding an efficient NR algorithm is an optimization problem in its own right.

9.4 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled $\star$ are more challenging.*

Exercise 9.1

We are interested in finding the fixed points (the points at which the time derivatives are zero) of the following system of equations:

$$\begin{aligned}\frac{dx_1}{dt} &= x_1(a - bx_2)\\ \frac{dx_2}{dt} &= -x_2(c - dx_1)\end{aligned}$$

for $a = 3,\ b = 1,\ c = 2,\ d = 1$. We can use the Newton-Raphson method to find these fixed points, simply by setting the derivatives zero in the given system of equations.

(a) In the scalar case, Newton-Raphson breaks down at points at which the derivative of the nonlinear function is zero. In general, where can it break down for systems of nonlinear equations? For the system given above, find the troublesome points.

(b) Find the fixed points of the above system analytically.

(c) Find all fixed points using repeated application of the Newton-Raphson method. You will have to judiciously choose your starting points (but of course, you are not allowed to use the known roots as starting points!). You may use MATLAB to program the method if you like.

Exercise 9.2

Higher order Taylor expansions.

(a) Assuming that a single-variable function is twice-differentiable you can approximate it in a point x with its second order Taylor expansion at a near-by point x_0

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0)$$

Based on this, approximate $f'(x)$ with a linear function in the proximity of x_0. Then explain how you could apply Newton-Raphson to find the critical points of a scalar function ($f'(x) = 0$).

(b) Now extend this idea to derive an algorithm for finding extrema of functions $f(\vec{x}) = f(x_1, x_2, \ldots, x_n)$ of more than one variable by exploiting the second order Taylor expansion, which for such multi-variable functions is given by

$$f(\vec{x}) = f(\vec{x}_0) + (\nabla f(\vec{x}_0))^T (\vec{x} - \vec{x}_0) + \frac{1}{2} (\vec{x} - \vec{x}_0)^T H(x_0) (\vec{x} - \vec{x}_0),$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ and $\nabla f(\vec{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$.
To find what the Hessian is, see http://en.wikipedia.org/wiki/Hessian˙matrix .

*(c) Newton-Raphson is a method for solving a system of nonlinear equations:

$$f(x_1, x_2, \ldots, x_n) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

If you recall from class, we used linear approximations of f_i to derive this method. Now use the quadratic approximation introduced in the last section for $f_i(\vec{x}) = f_i(x_1, x_2, \ldots, x_n)$ to reduce these equations f_i to a quadratic form and then design a general algorithm for solving the nonlinear system of equations based on this approximation. (no need to program this up!)

Exercise 9.3

The vector function $\vec{f}(\vec{x})$ is given by

$$\vec{f}(\vec{x}) = \begin{bmatrix} x_1^2 - x_1 x_2 \\ x_1 x_2 - x_2^2 \end{bmatrix}.$$

Give the Newton-Raphson algorithm for solving $\vec{f}(\vec{x}) = \vec{0}$. Perform two steps in this algorithm with the start vector $\vec{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. If you were to continue this process, what convergence criterion or criteria would you use?

Chapter 10

Eigenvalues and eigenvectors

In this chapter we discuss eigenvalues and eigenvectors. We motivate eigenvalues and eigenvectors them by looking for an effective way to analyze the behavior of solutions to systems of ordinary differential equations. We then continue to investigate various properties of eigenvalues and eigenvectors, introduce a few more decompositions (why not!) and also discuss iterative methods to approximate eigenvalues and eigenvectors when they cannot be found analytically.

Eigenvalues and eigenvectors are only used for square matrices. So, we assume that all matrices in this chapter are $n \times n$ matrices.

10.1 Exponential matrix

In your murky calculus past, you surely looked at the equation

$$\frac{du}{dt} = au, \quad t \geq 0,\ u(0) = u_0,$$

which is a linear ordinary differential equation with initial condition u_0. The solution to it is given by $u(t) = u_0 e^{at}$. You may have also seen examples of some systems of ordinary differential equations

$$\frac{d\vec{u}}{dt} = A\vec{u}, \quad t \geq 0,\ \vec{u}(0) = \vec{u}_0. \tag{10.1}$$

A well-known example is a predator-prey system (see exercises).

What would be the solution to this system? As a joke, you may suggest it is simply

$$\vec{u}(t) = e^{At}\vec{u}_0, \tag{10.2}$$

using a direct analogy to the scalar case. And you would be right! We do need to understand, of course, what this *exponential matrix* e^{At} really is. For this purpose, we first go back to the scalar case, where we used e^{at}. The Taylor series expansion of this exponential function is given by

$$e^{at} = 1 + ta + \frac{1}{2!}t^2a^2 + \frac{1}{3!}t^3a^3 + \dots$$

The matrix version is defined through this Taylor series, simply as

$$e^{At} = I + tA + \frac{1}{2!}t^2A^2 + \frac{1}{3!}t^3A^3 + \dots \tag{10.3}$$

Surely, this exponential matrix is really a matrix now. That is good. But is 10.2 really the solution we are after if this definition is used? Let's check. We would get

$$\vec{u}(t) = e^{At}\vec{u}_0 = \left(I + tA + \frac{1}{2!}t^2A^2 + \frac{1}{3!}t^3A^3 + \dots\right)\vec{u}_0.$$

This solution will have a time derivative equal to

$$\frac{d\vec{u}}{dt} = \left(0 + A + tA^2 + \frac{1}{2!}t^2A^3 + \dots\right)\vec{u}_0,$$

which can also be written as

$$\frac{d\vec{u}}{dt} = A\left(I + tA + \frac{1}{2!}t^2A^2 + \dots\right)\vec{u}_0,$$

which is clearly showing that

$$\frac{d\vec{u}}{dt} = Ae^{At}\vec{u}_0 = A\vec{u}.$$

Also, at $t = 0$, the initial condition is correctly satisfied. So, we have shown that this is indeed the desired solution. Very fun.

We see that the solution to this system of ODEs is completely controlled by powers of A. In order to understand the solution, we must understand these powers of A, and of course also this infinite series.

System of ODEs are extremely common in engineering. For example, every time I solve a fluid flow problem, I formulate a very large system of ODEs along the way. Systems also come up all the time in control theory, in optimization problems, in population models, in models for the spread of diseases, and so on.

No wonder that people thought long and hard about how to understand powers of matrices. The trick is to study the eigenvalues of the matrix, which we will look at next.

10.2 Introduction to eigenvalues and eigenvectors

In the very beginning of the class we talked about matrices as operators. A matrix applied to a vector can change both the direction and amplitude of a vector. It would be interesting then to ask if there were any vectors $\vec{y}$ were only affected by A in amplitude, that is, their direction would not be changed when multiplied by A? What this means is that we are interested in finding out if there are vectors $\vec{y}$ such that

$$A\vec{y} \sim \vec{y},$$

which we can also write as

$$A\vec{y} = \lambda\vec{y}, \tag{10.4}$$

with λ the proportionality constant. For now, we will assume $\lambda \in \mathbb{R}$.

Why are such vectors interesting? Suppose that $\vec{y}$ represents a state of a system, and A is an outside influence that impacts this state continuously. That is, the state is constantly multiplied by this matrix (for example, the matrix could represent a wind force that is impacting a swaying building, or bridge, continuously). If $\vec{y}$ is one of these special vectors and the corresponding λ is bigger than one, the state would get continuously amplified. In many cases, we really want to know when this happens.

We have also seen examples in this class. Remember stationary iterative methods? There the initial error was constantly hit by the amplification matrix G. What if the initial error was one of these special vectors and the corresponding λ was bigger than 1? The iterative method would diverge.

Vectors $\vec{y}$ that satisfy the equation 10.4 are called *eigenvectors* and the corresponding proportionality constants λ are called *eigenvalues*.

How do we find eigenvalues and eigenvectors? $A\vec{y} = \lambda\vec{y}$ is the same as $A\vec{y} - \lambda\vec{y} = \vec{0}$, or

$$(A - \lambda I)\,\vec{y} = \vec{0}.$$

We are of course only interested in nonzero vectors $\vec{y}$. The above equation shows that if indeed there are such $\vec{y}$, they will be in the nullspace of the matrix $A - \lambda I$. This matrix must then be singular and its determinant must be zero. But this gives us a really nice way to find them. We just find λ for which

$$\det(A - \lambda I) = 0.$$

Will we find any solutions to this? Absolutely. It's not hard to see that this determinant condition gives us an n-th degree polynomial in λ. To make this clearer, let's look at an example. We take

$$A = \begin{bmatrix} 1 & 4 \\ 1 & 1 \end{bmatrix},$$

which gives the equation

$$\det(A - \lambda I) = \begin{vmatrix} 1-\lambda & 4 \\ 1 & 1-\lambda \end{vmatrix} = (1-\lambda)^2 - 4 = 0.$$

So indeed we find a quadratic ($n = 2$) equation in λ that we can easily solve. We have $1 - \lambda = 2$ or $1 - \lambda = -2$, leading to $\lambda = -1$, or $\lambda = 3$.

If the matrix is $n \times n$, then we will find an n-th degree polynomial in λ when solving for the determinant. Importantly, this means that an $n \times n$ matrix will have n eigenvalues $\lambda_i, i = 1, \ldots, n$, because an n-th degree polynomial has n roots. The eigenvalues do not all have to be distinct, of course. Polynomials can have roots that have a multiplicity larger than 1. The polynomial used to find the eigenvalues is called the *characteristic polynomial*, and the equation that sets this polynomial to zero to find the eigenvalues is called the *characteristic equation*.

Once we have each $\lambda_i, i = 1, \ldots n$, we can find the eigenvector(s) corresponding to each λ_i by finding the nullspace of $A - \lambda_i I$. If this nullspace has dimension 1, we will find one base vector, which we set the eigenvector to be. We will denote eigenvectors by $\vec{y}_i, i = 1, \ldots, n$. Note that the eigenvector is not unique: every multiple is again an eigenvector. We often choose to normalize the eigenvector.

If the nullspace has dimension higher than 1, we do not find one eigenvector, but an *eigenspace*. Any basis for this eigenspace can be chosen for the eigenvectors.

Let's go back to our example. $\lambda_1 = -1$, gives

$$\begin{bmatrix} 2 & 4 \\ 1 & 2 \end{bmatrix} \vec{y}_1 = \vec{0}.$$

See that this matrix is singular? To find $\vec{y}_1$ we must simply find the nullspace of the given matrix. We see that this is given by the vectors $\vec{x}$ that satisfy $x_1 + 2x_2 = 0$, or $x_1 = -2x_2$. This means that $\vec{y}_1 = [-2, 1]^T$, or any multiple thereof. The length of an eigenvector is not set by the equation 10.4. This is not a surprise. Multiplying 10.4 by a constant gives the same result. Only the *direction* of the eigenvector is important. We often normalize eigenvectors but in this case, we'll just use the vector $[-2, 1]^T$. Similarly, for $\lambda_2 = 3$, we get

$$\begin{bmatrix} -2 & 4 \\ 1 & -2 \end{bmatrix} \vec{y}_2 = \vec{0}.$$

Again, the matrix is singular, as we designed it to be. The nullspace is now spanned by the vector $[2, 1]^T$, which is what we set $\vec{y}_2$ equal to.

10.3 Canonical transform of A

In the above example, we have two distinct eigenvalues and two distinct eigenvectors. They satisfy

$$A\vec{y}_1 = \lambda_1\vec{y}_1,\ A\vec{y}_2 = \lambda_2\vec{y}_2.$$

We can write this as $AY = Y\Lambda$ with

$$Y = [\vec{y}_1, \vec{y}_2], \quad \text{and } \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}.$$

This means that

$$A = Y\Lambda Y^{-1}.$$

We found another decomposition. Hurrah! It is called the *canonical form* of A, sometimes also referred to as the *eigenvalue decomposition*. We also say that if we can find such a decomposition, the matrix A is *diagonalizable*.

The canonical form is a very useful decomposition. First, if $A = Y\Lambda Y^{-1}$, then $A^2 = Y\Lambda Y^{-1}Y\Lambda Y^{-1} = Y\Lambda^2 Y^{-1}$. We can take this further and get, for all positive integer powers of A

$$A^k = Y\Lambda^k Y^{-1}.$$

This is a very powerful result because it tells you that powers of A are really controlled by the powers of the eigenvalues. In fact, plugging this into the equation for the system of ODEs, we get

$$\vec{u} = e^{At}\vec{u}_0 = Y\left(I + \Lambda t + \frac{1}{2}\Lambda t^2 + \ldots\right)Y^{-1}\vec{u}_0.$$

The matrix in the middle (containing the powers of Λ) is a diagonal matrix whose diagonal elements are equal to

$$1 + \lambda_i t + \frac{1}{2}(\lambda_i t)^2 + \frac{1}{3!}(\lambda_i t)^3 + \ldots = e^{\lambda_i t}.$$

We see that we get a relatively simple expression for the solution to this system of ODEs, and also clearly see that its behavior in time is fully controlled by the eigenvalues. If all eigenvalues are smaller than zero, the solution will go to 0 as time increases. If one of the eigenvalues is positive, the solution will continually increase.

We could also plug the decomposition into the system of ODEs itself. We have

$$\frac{d\vec{u}}{dt} = A\vec{u} = Y\Lambda Y^{-1}\vec{u},$$

or

$$\frac{dY^{-1}\vec{u}}{dt} = \Lambda(Y^{-1}\vec{u}).$$

We set $\vec{w} = Y^{-1}\vec{u}$, and see that

$$\frac{d\vec{w}}{dt} = \Lambda\vec{w}.$$

Because Λ is diagonal matrix, we simply get the m differential equations

$$\frac{dw_i}{dt} = \lambda_i w_i,$$

whose solutions are $w_i(t) = e^{\lambda_i t}w_i(0)$, and $\vec{w}(0) = Y^{-1}\vec{u}(0)$.

10.4 Diagonalizable matrices

We have seen in the previous section that diagonalizability is a very nice property. When is a matrix diagonalizable, in other words, when can we find a decomposition $A = Y\Lambda Y^{-1}$ for an $n \times n$ matrix A? From the way we derived the decomposition, it is clear that *we need n independent eigenvectors*. We have talked about a set of independent vectors mostly in the context of spaces and bases. It is really important to understand that having n independent eigenvectors, or not, has absolutely nothing to do with the rank of A. Again, in bold **there is no correlation between diagonalizability and singularity**. Also, if eigenvalues are repeated, it does not mean that we cannot find n independent eigenvectors. Let's look at some examples that show this clearly.

Example of a nonsingular matrix with repeated eigenvalues that is diagonalizable First, let's find the eigenvalues of the identity matrix. The matrix $I - \lambda I$ has $1 - \lambda$ on each diagonal element. Setting the determinant to zero gives $\lambda = 1$. This is the only solution, but it is repeated m times. Thus, $\Lambda = I$. To find the corresponding eigenvector(s), we investigate the nullspace of $I - \lambda I$. But this matrix is just the null matrix. What is the nullspace of the $n \times n$ null matrix? It's $\mathbb{R}^n$! So, any set of n linearly independent vectors can be chosen as the set of eigenvectors (remember that eigenvector(s) corresponding to a given λ span the nullspace of $(A - \lambda I)$). We now know that I is diagonalizable. This is also easy to see as if I is diagonalizable, we must be able to write $I = Y\Lambda Y^{-1}$. With $\Lambda = I$, we get $I = YY^{-1}$ and that's of course true for any nonsingular Y. So, the identity matrix is an example of a nonsingular matrix with repeated eigenvalues that is diagonalizable.

Example of a singular matrix that is diagonalizable We find in a similar way that the null matrix is also diagonalizable. It has n eigenvalues $\lambda = 0$, but again the nullspace of $A - \lambda I$, which is again the null matrix, is the whole of $\mathbb{R}^n$. This is an example of a completely singular matrix with repeated eigenvalues that is diagonalizable.

Example of a nonsingular matrix that is not diagonalizable Let's look at the matrix

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

This is a triangular matrix and that means that $A-\lambda I$ is also triangular. The determinant of a triangular matrix is simply the product of the diagonal elements. So, here $|A - \lambda I|= (1 - \lambda)^2$, giving us $\lambda = 1$ (double root). The matrix $A - \lambda I$ is then equal to

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

We can immediately see that the rank of this matrix is 1, and so its nullspace has only dimension 1. We can not find two base vectors for the nullspace and hence we do not have a full set of eigenvectors. A is not diagonalizable.

Hopefully these examples convinced you that there is no relation between singularity, repeated eigenvalues and diagonalizability.

It would be great if we could spot diagonalizability just by looking at some properties of a matrix A. This is not easy to do, but it is true that symmetric matrices are always diagonalizable. We will prove this a few sections below and look at some other classes of matrices that are always diagonalizable.

10.4.1 Diagonalizability and stationary iterative methods

Let's go back to the stationary iterative methods. There, we found that the error $\vec{e}^{(k)}$ is related to the initial error $\vec{e}^{(0)}$ through $\vec{e}^{(k)} = G^k\vec{e}^{(0)}$, where G is the amplification matrix. If G is diagonalizable, we can see that $\vec{e}^{(k)} = (Y\Lambda Y^{-1})^k\vec{e}^{(0)} = Y\Lambda^k Y^{-1}\vec{e}^{(0)}$. The change in the error is therefore completely controlled by the powers of the eigenvalues. Nice. If all eigenvalues are smaller than 1 in absolute value, the iteration will converge.

This leads to another convergence criterion: The iterative method converges if the largest eigenvalue, in absolute value, is smaller than 1. The largest eigenvalue, in absolute value, of a matrix has a special name. It's called the *spectral radius* and is usually denoted by $\rho(A)$. So, here, the convergence criterion is $\rho(A) < 1$. Section 10.10 dives into convergence and spectral radius a bit deeper.

10.5 Examples

Example 1 - exponential matrix Let's look at

$$\frac{d\vec{x}}{dt} = A\vec{x},$$

with

$$A = \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix}.$$

We know the solution to this equation is $\vec{x} = e^{At}\vec{x}_0$, with $\vec{x}_0$ the initial condition. If we simply develop the Taylor series for the exponential matrix, we get

$$\begin{aligned} e^{At} &= I + t\begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} + \frac{1}{2}t^2\begin{bmatrix} 1 & -1 \\ 0 & 4 \end{bmatrix} + \frac{1}{6}t^3\begin{bmatrix} 1 & 3 \\ 0 & -8 \end{bmatrix} + \dots \\ &= \begin{bmatrix} 1 + t + \frac{1}{2}t^2 + \frac{1}{6}t^3 + \dots & t - \frac{1}{2}t^2 + \frac{1}{6}t^3 3 + \dots \\ 0 & 1 - 2t + \frac{1}{2}t^2 4 - \frac{1}{6}t^3 8 + \dots \end{bmatrix} \\ &= \begin{bmatrix} e^t & \frac{1}{3}e^t - \frac{1}{3}e^{-2t} \\ 0 & e^{-2t} \end{bmatrix}. \end{aligned}$$

In this case, it was not so hard to find the actual expression of the exponential matrix. We were greatly helped by the upper triangular character of A in this expansion. We see that the behavior of the solution is completely determined by exponentials. We will certainly see growth in the solution because not all the exponentials have negative exponents.

Note, this example also shows clearly that for a matrix A given by $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, the exponential matrix is not simply equal to $e^{At} = \begin{bmatrix} e^{a_{11}t} & e^{a_{12}t} \\ e^{a_{21}t} & e^{a_{22}t} \end{bmatrix}$. That is only true in exceptional cases (e.g. when the matrix is diagonal).

We can also use the canonical transform to solve this problem provided A is diagonalizable. Here,

$$Y = \begin{bmatrix} 1 & 1 \\ 0 & -3 \end{bmatrix}, \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix},$$

which gives

$$e^{At} = Ye^{\Lambda t}Y^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & -3 \end{bmatrix}\begin{bmatrix} e^t & 0 \\ 0 & e^{-2t} \end{bmatrix}\begin{bmatrix} 1 & 1/3 \\ 0 & -1/3 \end{bmatrix} = \begin{bmatrix} e^t & \frac{1}{3}e^t - \frac{1}{3}e^{-2t} \\ 0 & e^{-2t} \end{bmatrix},$$

exactly as we found before.

Example 2 - repeated eigenvalues The eigenvalues of the identity matrix are readily found. The identity matrix is diagonal, and so the eigenvalues can simply be read off the diagonal. For an $n \times n$ matrix I, we find n eigenvalues all equal to 1. To find eigenvectors, we find base vectors for the nullspace of $(A - \lambda I)$, which for $A = I, \lambda = 1$ is just the null matrix. The nullspace of the null matrix is the whole of $\mathbb{R}^n$. There are an infinite number of bases for the whole of $\mathbb{R}^n$ and each can be considered as the set of eigenvectors. It is no

surprise that any vector is an eigenvector of I. We already knew that for all $\vec{x}$, $I\vec{x} = \vec{x}$, so each vector is an eigenvector with $\lambda = 1$.

We saw another matrix with repeated eigenvalues above: the matrix $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. For this matrix, again the two eigenvalues are 1. Now, the null space of $(A - \lambda I)$ only has rank 1. So there is not a complete set of eigenvectors and A is not diagonalizable.

Example 3 - distinct eigenvalues Take

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The e-values are given by the characteristic equation $(-\lambda)(-\lambda) - 1 = 0$, which leads to $\lambda = \pm 1$. $\lambda_1 = 1$ gives $A - \lambda I = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$. The corresponding eigenvectors $\vec{y}_1$ spans the nullspace of this matrix, and can be found as $\vec{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. For $\lambda_2 = -1$, we get $\vec{y}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Note, again, that these eigenvectors are not unique. Instead of $\vec{y}_1$ we could have chosen $\begin{bmatrix} -200 \\ -200 \end{bmatrix}$ if we had wanted to. As long as it is a base vector of the nullspace. Often, we choose to normalize the base vector, but even then we have two choices (we have two unit base vectors that are each others opposite). With this information, we know that A is diagonalizable and should be equal to

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{-1},$$

which I leave to you to check.

10.6 Similarity transforms and Schur decomposition

There are two major questions left in this chapter:

- When is a matrix diagonalizable? Can we find classes of matrices that are diagonalizable?
- How can we compute eigenvalues for very large matrices for which solving the characteristic equation is really not tractable?[1]

To answer the first question, we really need some more machinery. We need to know about similarity transforms and Schur decomposition. These are discussed in this section. In this section we'll also introduce some complex arithmetic. We will tackle the second question in the next section.

10.6.1 Similarity transform

We start this section simply with the definition of similarity:
If $A = VBV^{-1}$ for some nonsingular matrix V then A and B are said to be similar.

This mathematical notion of being similar/similarity, is different than the normal use of this word. So, when we say two matrices are similar, we do not mean to say that they look like each other. The transform $A = VBV^{-1}$ is called a *similarity transform.*

All fine and dandy, but why is this useful? Well, it turns out that if two matrices A and B are similar in this mathematical sense, then their eigenvalues are the same!. This can be shown easily: to find the eigenvalues of A we would compute $|A - \lambda I| = 0$. But

$$|A - \lambda I| = \left|VBV^{-1} - \lambda I\right| = \left|V(B - \lambda I)V^{-1}\right| = |B - \lambda I|\,.$$

Do you see how we did each step in this manipulation?

So, if A and B are similar, they have the same eigenvalues. This also means that if B happens to be triangular, we can read off the eigenvalues of A from the diagonal of B.

A special similarity transform is the diagonalization of a matrix. There $V = Y$, $B = \Lambda$.

Example 1 - similar matrices, then same eigenvalues The matrices $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} -1 & 0 \\ 2 & 3 \end{bmatrix}$ are similar. We have $A = VBV^{-1}$, with $V = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. The eigenvalues of A and B are $\lambda_1 = 3$ and $\lambda_2 = -1$.

You may wonder if the eigenvectors are the same as well. This is not necessarily so. It's not hard to convince yourself. Let $\vec{y}$ be an eigenvector of A. Then $A\vec{y} = \lambda\vec{y}$, so $VBV^{-1}\vec{y} = \lambda\vec{y}$, which means $B(V^{-1}\vec{y}) = \lambda(V^{-1}\vec{y})$. We see therefore that $V^{-1}\vec{y}$ is an eigenvector of B, not $\vec{y}$ itself.

[1] For large n, we can no longer find a closed form expression for the roots of the polynomial. Therefore, we have to resort to some kind of iterative scheme. It is typically very tricky to find all roots accurately in a reasonable amount of computational time

Example 2 - same eigenvalues, not similar You may wonder if it is also true that if two matrices have the same eigenvalues they must be similar? This is *not* true. Here's a counter example. We take $A = I$ and $B = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Both A and B have the repeated eigenvalue $\lambda = 1$. But, $A = VBV^{-1}$ can never be true. If it were then $V^{-1}AV = B$, but with $A = I$ would give $V^{-1}V = B$, or $I = B$, which is obviously not the case.

10.6.2 Schur form - real matrices with real eigenvalues

The Schur form is an extraordinarily handy decomposition of a matrix. In the optional subsection 10.6.6, a proof of existence is outlined. Instead of the *Schur form*, we also can refer to this as the *Schur decomposition*.

We discuss several different versions of the Schur form, starting with real matrices with real eigenvalues.

Schur form Version 1 - real matrices with real eigenvalues [2] *If the matrix A is real and has real eigenvalues then there is always a real orthogonal matrix U such that $A = USU^{-1} = USU^T$, with S real and upper triangular.*

The eigenvalues of A can be found on the diagonal of S. Why? Because A and S are similar, so they have the same eigenvalues, and S is upper triangular, so its eigenvalues are the same as its diagonal elements. But, note that U does *not* hold the e-vectors of A, unless S is diagonal.

10.6.3 Working with complex matrices and vectors

In all our examples thus far, the eigenvalues were real. However, a real matrix may very well have complex eigenvalues, and of course complex matrices can have complex eigenvalues. For example, $A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ has the characteristic equation $(1-\lambda)^2 + 1 = 0$, which gives $\lambda = 1 \pm i$. There is a version of the Schur form for complex eigenvalues and for complex matrices. But if we extend our matrix classes to those involving complex numbers, we do need to understand how to work with complex numbers in the context of matrices and vectors. Let's have a look at that first.

Let $\vec{x} \in C^n$ and $A, U \in C^{n\times n}$. Then:

$$\|\vec{x}\|_2^2 = |x_1|^2 + \ldots + |x_n|^2, \quad \text{with } |x_i|^2 = \Re(x_i)^2 + \Im(x_i)^2 = x_i\bar{x}_i, \text{ with } \bar{x}_i = \Re(x_i) - i\Im(x_i).$$

[2] Note that if A is real and the eigenvalues of A are real, then the eigenvectors of A are real as well. Do you see that?

This means that, analogous to the real case, $\|\vec{x}\|_2^2 = \bar{\vec{x}}^T\vec{x}$. Instead of $\bar{\vec{x}}^T$ we typically write $\vec{x}^*$. You will see here that the complex case looks just like the real case, but each time we use a transpose in the real case, we use the transpose of the complex conjugate in the complex case.

Vectors $\vec{x}$ and $\vec{y}$ in complex space are orthogonal if $\vec{x}^*\vec{y} = 0$.

We call a matrix A *Hermitian* if $A^* = A$. This is the analogue to symmetry in the real case.

We call a matrix U *unitary* if $U^*U = I$. This is the analogue to orthogonality in the real case.

If a real matrix A has a complex eigenvalue $\lambda = a + ib$, then $\lambda = a - ib$ is also an eigenvalue. This follows immediately from $A\vec{x} = \lambda\vec{x}$. Take the complex conjugate of this expression and use the fact that A is real. We get $A\bar{\vec{x}} = \bar{\lambda}\bar{\vec{x}}$. Clearly we have found another eigenvalue/eigenvector pair.

Examples The matrix $A = \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix}$ is Hermitian. Note that it is not symmetric in the real sense: for Hermitian matrices, we also take the complex conjugate.

The matrix $U = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -i/\sqrt{2} & i/\sqrt{2} & 0 \\ 0 & 0 & i \end{bmatrix}$ is a unitary matrix. Check this for yourself.

10.6.4 Schur form - real A with complex eigenvalues

If a matrix A is real, but has (some) complex eigenvalues, there is another form of the Schur form that comes in very handy.

Schur form Version 2 - real matrix with (some) complex eigenvalues *If A is real with (some) complex eigenvalues than there is a real orthogonal U such that $A = U\tilde{S}U^{-1}$ with $\tilde{S}$ quasi upper triangular.*

What does this quasi upper triangular matrix look like? All real eigenvalues of A again appear along the diagonal. But for each pair of complex eigenvalues (remember that all complex eigenvalues come in complex conjugate pairs), there is a little 2×2 matrix on the diagonal whose eigenvalues are the complex conjugate pair.

Let's look at an example. We take

$$A = \begin{bmatrix} 67 & 177.6 & -63.2 \\ -20.4 & 95.88 & -87.16 \\ 22.8 & 67.84 & 12.12 \end{bmatrix}.$$

Its eigenvalues are the set $\lambda(A) = \{25, 75 + 100i, 75 - 100i\}$. We can find the Schur form with

$$U = \begin{bmatrix} -0.8 & 0.36 & 0.48 \\ 0.36 & 0.928 & -0.096 \\ 0.48 & -0.096 & 0.872 \end{bmatrix}.$$

Check that this is orthogonal. The corresponding matrix $\tilde{S}$ is

$$\tilde{S} = \begin{bmatrix} 25 & -90 & 5 \\ 0 & 147 & -104 \\ 0 & 146 & 3 \end{bmatrix}.$$

The little 2×2 matrix that corresponds to the complex conjugate pair of eigenvalues is $\begin{bmatrix} 147 & -104 \\ 146 & 3 \end{bmatrix}$. Its eigenvalues are $75 \pm 100i$, just as desired.

10.6.5 Schur form - real or complex A with real or complex eigenvalues

The most general form of the Schur decomposition is given here. Note that this does not necessarily lead to real U or real S.

Schur form Version 3 - real of complex A with real of complex eigenvalues *For real or complex A with real or complex eigenvalues, we can find a unitary U and upper triangular S, both in $C^{n \times n}$, such that $A = USU^*$.*

As example, we take $A = \begin{bmatrix} 3 & 8 \\ -2 & 3 \end{bmatrix}$, and find $U = \begin{bmatrix} 0.8944i & 0.4472 \\ -0.4472 & -0.8944i \end{bmatrix}$, and $S = \begin{bmatrix} 3 + 4i & -6 \\ 0 & 3 - 4i \end{bmatrix}$.

Note that here, we find the complex eigenvalues along the diagonal of S. In the previous Schur form, we got a little 2×2 matrix representing the eigenvalues along the diagonal, and both $\tilde{S}$ and U were real. Here, S and U are complex.

10.6.6 Optional: How to prove a Schur form exists?

This is an optional subsection and is not examinable. Several of you have asked how we prove that the Schur form indeed exists.The proof uses induction.

We assume we have an $n \times n$ matrix A. We assume that it is proven that the Schur form exists for $(n-1) \times (n-1)$ matrices. We will show that if this is so, it also exists for $n \times n$ matrices (this is the induction part). All we need then is show that a Schur form exists for $n = 1$.

Let $\vec{y}$ be a normalized eigenvector of A for the eigenvalue λ. Create a matrix U with $\vec{y}$ as its first column. Then

$$U^*AU = \left[\begin{array}{c|c} \lambda & \vec{b}^* \\ \hline 0 & C \end{array}\right],$$

where C is some $(n-1) \times (n-1)$ matrix, and $\vec{b}^*$ is some row vector with $n-1$ elements. By hypothesis, there exists a Schur form of C. Let it be equal to $C = VSV^*$, with V a unitary matrix. Then create

$$W = U \left[\begin{array}{c|c} 1 & \vec{0}^* \\ \hline 0 & V \end{array}\right].$$

Note that W is also unitary. We see then that

$$W^*AW = \left[\begin{array}{c|c} \lambda & \vec{b}^*V \\ \hline 0 & S \end{array}\right] = T,$$

with T triangular. Therefore we have shown that there is a W such that $A = WTW^*$, which is a Schur form. We have now completed the induction. Of course for $n = 1$, a Schur form always exists. See that?

10.7 When is a matrix diagonalizable?

With the Schur form, we can now find a few classes of matrices that are guaranteed to be diagonalizable.

10.7.1 Symmetric matrices

Let A be real and symmetric. First, we can prove that A always has real eigenvalues (so we can use the Schur Version 1). Take

$$A\vec{x} = \lambda\vec{x},$$

and take the complex conjugate of both sides. We get

$$A\bar{\vec{x}} = \bar{\lambda}\bar{\vec{x}}, \text{ or } \bar{\vec{x}}^T A^T = \bar{\lambda}\bar{\vec{x}}^T, \text{ or } \vec{x}^* A = \bar{\lambda}\vec{x}^*$$

We now take the inner product of both expressions with $\vec{x}$, which gives

$$\vec{x}^* A\vec{x} = \lambda\vec{x}^*\vec{x}, \quad \vec{x}^* A\vec{x} = \bar{\lambda}\vec{x}^*\vec{x}.$$

Subtraction gives $0 = (\lambda - \bar{\lambda})\vec{x}^*\vec{x}$ and so $\lambda = \bar{\lambda}$.

Schur Version 1 now gives us that $A = USU^T$. $A = A^T$ means that $USU^T = US^TU^T$ so that $S = S^T$. We conclude that S is diagonal. This means that the Schur form is also a diagonalization of A. The eigenvalues of A are on the diagonal of S and the eigenvectors of A are the column vectors of U. We have found that a symmetric matrix is always diagonalizable, but also that its eigenvectors can always be chosen to be orthogonal.

Question: why do I write "can always be chosen to be orthogonal" and not simply "are always orthogonal"? Perhaps subtle, but an important observation.

10.7.2 Normal matrices

A *normal matrix* in mathematics is not a matrix that does not look strange. It is a matrix A that satisfies $A^*A = AA^*$. Hermitian matrices are certainly normal, but not all normal matrices are also Hermitian. An example is $A \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$. Check this for yourself.

From Schur Version 3, we know that $A = USU^*$. Using $AA^* = A^*A$, we find that $S^*S = SS^*$. Again, this shows that S is diagonal. In the book by Strang you can find an elegant proof, but try to find it first for yourself.

With this result, we can again conclude that A is always diagonalizable and has, again, orthogonal eigenvectors.

10.7.3 Distinct eigenvalues

The final class of diagonalizable matrices that we will discuss are the matrices with distinct eigenvalues. That is, the matrices for which none of the eigenvalues is repeated. It makes complete sense that then we have a full set of eigenvectors: each of the $A-\lambda_i I$ is a different matrix, and hence will have a different nullspace. Can we prove this more rigorously?

We take A to be a $n \times n$ matrix. As is given, it has n distinct eigenvalues $\lambda_i, i = 1, \ldots, n$. For each eigenvalue, we can obviously find an eigenvector as the nullspace of $A - \lambda_i I$ has rank at least 1. If the eigenvectors $\vec{y}_i$ do not span all of $\mathbb{R}^n$, at least one of them must be dependent on the others. We will show that this is impossible.

We will use induction. We will show that if the first $m-1$ eigenvectors are independent, then so are the first m. And we do that by contradiction.

If the first $m-1$ eigenvectors are independent, but the first m are not, it must be true that the mth eigenvector $\vec{y}_m$ is a linear combination of the first $m-1$. That is, we must have

$$\vec{y}_m = \sum_{j=1}^{m-1} \alpha_j \vec{y}_j$$

Since $A\vec{y}_m = \lambda_m \vec{y}_m$, we have

$$A \sum_{j=1}^{m-1} \alpha_j \vec{y}_j = \lambda_m \sum_{j=1}^{m-1} \alpha_j \vec{y}_j,$$

so that

$$\sum_{j=1}^{m-1} \alpha_j A\vec{y}_j = \sum_{j=1}^{m-1} \alpha_j \lambda_j \vec{y}_j = \sum_{j=1}^{m-1} \alpha_j \lambda_m \vec{y}_j.$$

But then

$$\sum_{j=1}^{m-1} \alpha_j (\lambda_m - \lambda_j) \vec{y}_j = \vec{0}.$$

Since the first $m-1$ eigenvectors were assumed independent, the α_j's are not all zero (do you see why not?) and the eigenvalues are distinct, this is impossible. We conclude that if the first $m-1$ eigenvectors are independent, so are the first m. If we start with $m = 2$, we can make our way all the way up to n and we have shown that A has a complete set of independent eigenvectors and hence is diagonalizable.

10.8 How to compute eigenvalues for large n?

In all the examples so far, we have used the characteristic polynomial to find the eigenvalues. For large n, this is no longer do-able. Instead, several other methods have been designed to quickly compute reasonable approximation to (some of) the eigenvalues. We will discuss two: the power method and the QR iteration.

10.8.1 The Power method

Suppose that you are interested in finding the largest eigenvalue (in absolute value) of a matrix A. The Power method will then come in handy.

We assume here that the matrix A is diagonalizable, that it is real and that all eigenvalues are real too. We also order the eigenvalues so that

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|.$$

Since A is diagonalizable, we now we have a full set of n independent eigenvectors $\vec{y}_i, i = 1, \ldots, n$. This means that we can express any vector $\vec{x}$ as a linear combination of the eigenvalues. Let's choose an arbitrary $\vec{x}^{(0)}$. We have

$$\vec{x}^{(0)} = \sum_{i=1}^{n} \alpha_i \vec{y}_i.$$

But then

$$A\vec{x}^{(0)} = \sum_{i=1}^{n} \alpha_i A\vec{y}_i = \sum_{i=1}^{n} \alpha_i \lambda_i \vec{y}_i,$$

and

$$A^k\vec{x}^{(0)} = \sum_{i=1}^{n} \alpha_i A^k\vec{y}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^k \vec{y}_i,$$

for any power k. We can write this also as

$$\begin{aligned} A^k\vec{x}^{(0)} &= \lambda_1^k \sum_{i=1}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \vec{y}_i \\ &= \lambda_1^k \left(\alpha_1\vec{y}_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \vec{y}_i\right). \end{aligned}$$

In this last expression, the last term will go to zero as $k \to \infty$, because $|\lambda_1| > |\lambda_i|$ for all $i = 2$ to n.

What does this mean? If we keep multiplying some vector $\vec{x}^{(0)}$ with the matrix A, we will get a vector that points more and more in the direction of the first eigenvector $\vec{y}_1$ as k grows! We have found a way to compute the first eigenvector.

We still need to think about how we could compute an approximation to the eigenvalue λ_1. How about this? We have $A\vec{y}_1 = \lambda_1\vec{y}_1$, so $\vec{y}_1^T A\vec{y}_1 = \lambda_i\vec{y}_i^T\vec{y}_i$. then

$$\lambda_i = \frac{\vec{y}_1^T A\vec{y}_1}{\vec{y}_i^T\vec{y}_i}.$$

We use this relation in the iteration, not with $\vec{y}_1$, which we do not know, but the approximation to $\vec{y}_1$, which will then give us an approximation to the eigenvalue λ_1.

Because λ_1^k may grow fast, the vector $A^k\vec{x}^{(0)}$ can grow very fast as well. We probably want to normalize after each application of A to keep things nice and tidy.

We now have all ingredients to formulate an iterative procedure. This is the Power method:

Algorithm 4 Power method to find λ_1 and $\vec{y}_1$

k=0
Choose $\vec{x}^{(0)}$
while not converged **do**
  Compute $\vec{x}^{(k+1)} = A\vec{x}^{(k)}$
  Normalize $\vec{x}^{(k+1)} = \vec{x}^{(k+1)}/\|\vec{x}^{(k+1)}\|_2$
  Compute $\lambda^{(k+1)} = (\vec{x}^{(k+1)})^T A\vec{x}^{(k+1)}$
  k = k+1
end while

In this algorithm, $\lim_{k\to\infty}\vec{x}^{(k)} \to \vec{y}_1$ and $\lim_{k\to\infty}\lambda^{(k)} \to \lambda_1$.

A couple of notes. To find $\lambda^{(k+1)}$, we do not have to divide by the inner product of $\vec{x}^{(k+1)}$ with itself as we normalized this vector, so the inner product is 1. Also to find $\lambda^{(k+1)}$, we compute $A\vec{x}^{(k+1)}$, which is the matrix-vector multiplication we need at the next iteration step. Of course, we will not compute it again there, but store it for use in the next iteration step.

Convergence criteria What should we use for convergence criteria? Often we iterate until

$$\left|\frac{\lambda^{(k+1)} - \lambda^{(k)}}{\lambda^{(k)}}\right| < \epsilon,$$

where ϵ is some small tolerance. What other convergence criteria can you think of?

How fast will the Power method converge? The convergence of the algorithm depends on the ratio of the first two eigenvalues. If the eigenvalues are well separated (large ratio) we will have a fast convergence. If the gap between the eigenvalues is small, the convergence is expected to be slow.

Let's look at an example. We take

$$A = \begin{bmatrix} -261 & 209 & -49 \\ -530 & 422 & -98 \\ -800 & 631 & -144 \end{bmatrix},$$

with eigenvalues equal to $\lambda_1 = 10, \lambda_2 = 4, \lambda_3 = 3$. As initial guess, we take $\vec{x}^{(0)} = [1, 0, 0]^T$. We find

k	$\lambda^{(k)}$
1	13.0606
2	10.7191
3	10.2073
4	10.0633
5	10.0198
6	10.0063
7	10.0020
8	10.0007
9	10.0002

The ratio between the two largest eigenvalues is 0.4. Look at the computational results. You can see that the error between the approximation $\lambda^{(k)}$ and the exact eigenvalue, which is 10, is (approximately) attenuated by 0.4 in each iteration step. As expected.

There are a few remaining questions that are quite interesting:

- What happens if we choose $\vec{x}^{(0)}$ by chance such that it does not have any component of $\vec{y}_1$ in it, in other words, $\alpha_1 = 0$?
- What happens if $\lambda_1 = \lambda_2$?
- What could we do if we are not interested in λ_1, the largest eigenvalues, but rather in λ_n, the smallest eigenvalue?
- What could we do if we are interested in finding an eigenvalue, not the largest nor the smallest, but one close to a given value μ?

The first two questions are for you to ponder over. We will address the last two questions in the next subsections.

10.8.2 The inverse Power method: Finding the smallest eigenvalue

The power method finds the largest eigenvalue (in absolute value) of a matrix A. Suppose that we are instead interested in the smallest eigenvalue (in absolute value) and we know that this smallest value is not zero. We can also use the Power method, but not on A but on A^{-1}. To understand this, we first make a couple of observations.

Observation: If A is singular it has a zero eigenvalue If a matrix A is singular, than the smallest eigenvalue (in absolute value) of A is zero. How do we know? If A is singular, the nullspace of A has dimension larger than zero. This means that there is a nonzero $\vec{y}$ such that $A\vec{y} = \vec{0}$. But this means that $\vec{y}$ is an eigenvector corresponding to $\lambda = 0$.

In the remainder of this subsection, we assume that A is nonsingular and that we are interested in the smallest eigenvalue of A, which will not be zero.

Observation: the largest eigenvalue of A^{-1} is the reciprocal of the smallest eigenvalue of A The second observation is that if λ is an eigenvalue of A then $1/\lambda$ is an eigenvalue of A^{-1}. This is not so hard to see. If λ is an eigenvalue of A and $\vec{y}$ its corresponding eigenvector, then we have $A\vec{y} = \lambda\vec{y}$, and so $\vec{y} = \lambda A^{-1}\vec{y}$, or $\frac{1}{\lambda}\vec{y} = A^{-1}\vec{y}$. This shows that $\vec{y}$ is again an eigenvector of A^{-1} with eigenvalue $1/\lambda$.

This last result is what inspires the inverse Power method. If λ_n is the smallest eigenvalue (in absolute value) of A, then $1/\lambda_n$ is the largest eigenvalue (in absolute value) of A^{-1}. To find λ_n, we can therefore apply the Power method to A^{-1}. In the algorithm, we would naively compute $\vec{x}^{(k+1)} = A^{-1}\vec{x}^{(k)}$. Since we are quite allergic to inverse, instead we solve $A\vec{x}^{(k+1)} = \vec{x}^{(k)}$. We now get the algorithm

Algorithm 5 Inverse power method to find λ_n and $\vec{y}_n$

k=0
Choose $\vec{x}^{(0)}$
while not converged **do**
  Solve $A\vec{x}^{(k+1)} = \vec{x}^{(k)}$
  Normalize $\vec{x}^{(k+1)} = \vec{x}^{(k+1)}/\|\vec{x}^{(k+1)}\|_2$
  Compute $\lambda^{(k+1)} = (\vec{x}^{(k+1)})^T A\vec{x}^{(k+1)} = (\vec{x}^{(k+1)})^T\vec{x}^{(k)}$
  k = k+1
end while

Question for you: Why do we not compute $\lambda^{(k+1)}$ as $(\vec{x}^{(k+1)})^T A^{-1}\vec{x}^{(k+1)}$ and then take the reciprocal to find λ_n?

10.8.3 The shifted inverse Power method: Finding an eigenvalue near a value μ

We can find intermediate eigenvalues using the shifted inverse Power method. Suppose that we are interested in an eigenvalue λ_k, which we know to be near the estimate μ. We can do this by applying the inverse Power method to $(A-\mu I)^{-1}$. To see this, we first need to make another observation about eigenvalues.

Observation: If a matrix A has an eigenvalue λ, then $A-\mu I$ has an eigenvalue $\lambda-\mu$. This is not hard to see: $A\vec{y}=\lambda\vec{y}$ gives $(A-\mu I)\vec{y}=(\lambda-\mu)\vec{y}$, so $\vec{y}$ is the eigenvector of the shifted matrix $A-\mu I$ with corresponding eigenvalue $\lambda-\mu$.

If λ_k is close to μ, then the shifted matrix $A-\mu I$ will have a small eigenvalue $\lambda_k-\mu$. If μ is a close approximation, then likely $\lambda_k-\mu$ will be the smallest eigenvalue of $A-\mu I$, so $\frac{1}{\lambda_k-\mu}$ will be the largest eigenvalue of $(A-\mu I)^{-1}$ and we can apply the inverse Power method. Note that again, the eigenvector stays the same throughout.

10.8.4 The QR iteration

In all of the following, we assume that the $n\times n$ matrix A is real, diagonalizable and also non-singular. We again order the eigenvalues of A such that

$$|\lambda_1|>|\lambda_2|\geq\lambda_3\geq\ldots\geq|\lambda_n|.$$

The corresponding eigenvectors are $\vec{y}_1,\vec{y}_2,\ldots,\vec{y}_n$. The eigenvalues of A can either be real or complex. Complex eigenvalues always come in conjugate pairs (remember why?).

In the previous section we learned how to compute a particular eigenvalue of A using the Power method (for the largest e-value in absolute value), the inverse power method (for the smallest eigenvalue in absolute value) and the shifted inverse power method (for an eigenvalue closest to a given value μ). In this section we look at a method that can be used to find *all* eigenvalues at the same time. It is based on the Schur decomposition and similarity. If you are not fully comfortable with either, please review the above sections that describe them.

We know that for a real matrix A with real eigenvalues, we can find a real Schur decomposition $A=VSV^T$, where S is a real upper triangular matrix and V a real and orthogonal matrix. The eigenvalues of A and S are the same (the matrices are similar) and so the eigenvalues of A can be read off from the diagonal of S.

We also discussed in the lectures that if some of the eigenvalues of the real matrix A are complex then S is not a real upper triangular, but a real quasi-triangular matrix $\tilde{S}$: each real eigenvalue of A will show up as a diagonal element of $\tilde{S}$, and each complex conjugate pair of eigenvalues will give rise to a real 2×2 matrix on the diagonal of $\tilde{S}$.

So, if we could somehow find these Schur forms, we would have a way to find all e-values of A. The algorithm we look at here that can do this is a little nugget called the QR iteration. It nudges A to S, or $\tilde{S}$ if A has complex eigenvalues, using a sequence of QR factorizations. How does it work?

Since A is a real and non-singular matrix (we assumed it was non-singular), we can compute the QR factorization of A with Q a real $n \times n$ orthogonal matrix and R a real nonsingular $n \times n$ triangular matrix. We do this with the Gram-Schmidt orthogonalization process.

Now, we flip the order of multiplication around and create a new matrix

$$A^{(1)} = RQ.$$

Since $A = QR$, we know that $R = Q^T A$, and so we can write $A^{(1)} = Q^T AQ$. What does this last expression mean? It means that $A^{(1)}$ is similar to A, and so they have the same eigenvalues. Do you see why? We have $QA^{(1)}Q^T = A$, or $QA^{(1)}Q^{-1} = A$ (because Q is orthogonal) and so we have a similarity transform.

Now we repeat the process: We compute the QR decomposition of $A^{(1)}$, which gives

$$A^{(1)} = Q^{(1)}R^{(1)}.$$

Then, we flip the $Q^{(1)}$ and $R^{(1)}$ matrices again and get

$$A^{(2)} = R^{(1)}Q^{(1)}.$$

Then, $A^{(2)} = (Q^{(1)})^T A^{(1)} Q^{(1)} = (Q^{(1)})^T Q^T AQQ^{(1)}$. Now we observe that $(Q^{(1)})^T Q^T = (QQ^{(1)})^T$ and that the matrix $QQ^{(1)}$ is again an orthogonal matrix (if you don't directly see that the product of two orthogonal matrices is again orthogonal, spend a bit of time to convince yourself).

So, we see that $A^{(2)}$ is similar to $A^{(1)}$ and also similar to A.

Continuing this process we get

$$A^{(k)} = (Q^{(k-1)})^T (Q^{(k-2)})^T \dots (Q^{(1)})^T Q^T AQQ^{(1)} \dots Q^{(k-2)}Q^{(k-1)},$$

or in other words

$$A^{(k)} = \tilde{Q}^T A\tilde{Q}, \text{ with } \tilde{Q} = QQ^{(1)} \dots Q^{(k-2)}Q^{(k-1)},$$

and $\tilde{Q}$ an orthogonal matrix. Again we conclude that A and $A^{(k)}$ are similar.

The magic of this iteration is now that this $\tilde{Q}$ will converge to the V of the real Schur form, and $A^{(k)}$ will convergence to the triangular S if the eigenvalues of A are real, or the quasi-triangular $\tilde{S}$ if A has (some) complex eigenvalues. Pretty amazing, don't you think? But really true. Proving that this is the case is not at all easy, and beyond the level of this class.

The simplest way to write this QR iteration is given in the algorithm below.

Algorithm 6 QR iteration to find the Schur form of a real matrix

k=0
(in this algorithm we overwrite the original A with subsequent iterates)
while (not converged) **do**
  Find the QR factorization of the current A: $A = QR$
  Swap Q and R to find the next iterate: $A = RQ$
  k = k+1
  Check for convergence
end while

So here, we overwrite A each time: At the end of each step k, A contains $A^{(k)}$.

As example, we apply this iteration to the 3×3 matrix A given by

$$A = \begin{bmatrix} -261 & 209 & -49 \\ -530 & 422 & -98 \\ -800 & 631 & -144 \end{bmatrix},$$

with eigenvalues $10, 3$ and 4.

We run the QR iteration for a couple of times. After 1 iteration we get

$$A^{(1)} = \begin{bmatrix} 13.1 & -9.11 & -1281 \\ 0.039 & 3.94 & -2.93 \\ 0.0236 & -0.0168 & 0.0045 \end{bmatrix}.$$

After 2 iterations, we get

$$A^{(2)} = \begin{bmatrix} 10.72 & -9.01 & 1281 \\ 0.013 & 3.96 & -0.85 \\ -0.004 & 0.00023 & 2.31 \end{bmatrix}.$$

The convergence here is quite fast. We see the lower triangular elements go to 0 quite quickly and the eigenvalues start to appear along the diagonal of $A^{(k)}$. For $k = 10$, we have

$$A^{(10)} = \begin{bmatrix} 10.0 & -14.4 & 1281 \\ 0.00001 & 3.998 & -3.74 \\ -0.0000002 & -0.0006 & 3.0 \end{bmatrix}.$$

As another example I ran the QR iteration in MATLAB also for the matrix

$$A = \begin{bmatrix} 67 & 177.6 & -63.2 \\ -20.4 & 95.88 & -87.16 \\ 22.8 & 67.84 & 12.12 \end{bmatrix}.$$

This matrix has eigenvalues $25, 75 + 100i$ and $75 - 100i$. So, now the QR iteration should lead to the quasi upper triangular matrix $\tilde{S}$.

Now, we find for $k = 2$ that

$$A^{(2)} = \begin{bmatrix} 114 & 60 & 66 \\ -193 & 33 & -50 \\ 4.4 & 2.9 & 28 \end{bmatrix},$$

and for $k = 7$, we have

$$A^{(7)} = \begin{bmatrix} 69 & 207 & 55 \\ -49 & 81 & 50 \\ -0.0007 & -0.0011 & 25 \end{bmatrix}.$$

The elements at positions $(3, 1)$ and $(3, 2)$ are going to zero, whereas the element at $(3, 3)$ converges to the real eigenvalue 25. When we compute the eigenvalues of the 2×2 matrix in the top left corner, we find that they are equal to $75.0000 + 99.9995i$ and $75.00099.9995i$.

Note that in the above I have not been totally consistent with my significant digits.

10.8.5 Optional: Where does this QR iteration come from?

This subsection is optional and non-examinable.

The QR iteration sounds a bit too good to be true. Where does it actually come from? It is in fact derived from what we call *simultaneous iteration*, which is nothing but the Power method applied to a whole bunch of vectors at the same time.

We start with an orthogonal matrix $V^{(0)}$, with columns $\vec{v}_j, j = 1, \ldots, n$. We assume that A is symmetric and has n orthogonal eigenvectors.

If we apply A to this matrix k times (as in the Power method), we get

$$V^{(k)} = A^k V^{(0)} = [A^k \vec{v}_1, A^k \vec{v}_2, \ldots, A^k \vec{v}_n].$$

We reasoned in the Power method that all of these columns will then ultimately converge to point in the same direction $\vec{y}_1$, where $\vec{y}_1$ is the eigenvector corresponding to the largest

eigenvalue (in absolute value) λ_1 of A. So, applying the power method to a whole matrix, so many vectors at the same time, does not seem to make much sense. But what if after each application of A we compute the QR decomposition of the resulting matrix and continue to the next step not with the matrix itself but with Q? We know then that the first column of the matrix is always kept (only normalized) and so will as before converge to $\vec{y}_1$. But the second column is not allowed to keep any component of $\vec{y}_1$: we explicitly remove it when performing Gram-Schmidt. The second column will then converge to $\vec{y}_2$, we expect. With similar arguments, the third column vector will converge to $\vec{y}_3$ and so on. See?

The algorithm is given below.

Algorithm 7 Simultaneous iteration

k=0
Choose $V^{(0)}$
while not converged **do**
  Compute $V^{(k+1)} = AV^{(k)}$
  Compute QR decomposition $V^{(k+1)} = Q^{(k+1)}R^{(k+1)}$
  Set $V^{(k+1)} = Q^{(k+1)}$
  k = k+1
end while

Now, what is not so hard to show now is that with $V^{(0)} = I$, the matrix $(Q^{(k)})^T AQ^{(k)}$ is the same matrix as the $A^{(k)}$ matrix in the QR iteration. It is clear then that this matrix $A^{(k)}$ will converge to the Schur form (which for symmetric matrices is diagonal).

It is much harder to give intuition for this iteration for the non-symmetric case. But it still works.

If you have any questions about this subsection, send them to me on Piazza.

10.9 Cayley-Hamilton

The last topic for this chapter is Cayley-Hamilton. It says that each matrix satisfies its own characteristic equation.

It's easiest to see what is meant with an example. Let's take

$$A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}.$$

This triangular matrix has the characteristic polynomial $p(\lambda) = (\lambda - 2)(\lambda - 1)$. The characteristic equation is then $(\lambda - 2)(\lambda - 1) = 0$ (it is the equation we use to find the

eigenvalues). It turns out that now also $(A-2I)(A-I)=0$! In other words, A satisfies its own characteristic equation. In more general terms, if $det(A-\lambda I)=0$ gives the characteristic equation

$$\lambda^n+\alpha_1\lambda^{n-1}+\ldots+\alpha_{n-1}\lambda+\alpha_n=0,$$

then

$$A^n+\alpha_1A^{n-1}+\ldots+\alpha_{n-1}A+\alpha_nI=0 \tag{10.5}$$

also.

It is quite easy to prove this if A is diagonalizable. Then we know that $A^k=Y\Lambda^kY^{-1}$, and substitution of this expression in 10.5 immediately shows that the equation is indeed valid:

$$\begin{aligned}&Y\Lambda^nY^{-1}+\alpha_1Y\Lambda^{n-1}Y^{-1}+\ldots+\alpha_{n-1}Y\Lambda Y^{-1}+\alpha_nI=\\&Y\left(\Lambda^n+\alpha_1\Lambda^{n-1}+\ldots+\alpha_{n-1}\Lambda+\alpha_nI\right)Y^{-1}=0.\end{aligned}$$

Using Cayley-Hamilton, *any* power $A^p, p\geq n$ can be expressed as a polynomial of degree $n-1$ in A. The characteristic equation itself gives

$$A^n=-\alpha_1A^{n-1}-\ldots-\alpha_nI,$$

which in turn gives

$$A^{n+1}=-\alpha_1A^n-\alpha_2A^{n-1}-\ldots-\alpha_nA,$$

or

$$A^{n+1}=-\alpha_1\left(-\alpha_1A^{n-1}-\ldots-\alpha_nI\right)-\alpha_2A^{n-1}-\ldots-\alpha_nA,$$

which is a polynomial of degree $n-1$ in A. And we can continue this process for higher powers of A. Of course we can do exactly the same thing for the eigenvalues.

Cayley-Hamilton is not used all that often, but it is quite fun as it allows us to compute all sorts of strange functions of matrices (and eigenvalues) when we realize that any function can be expressed in terms of a Taylor series, and the higher powers in the Taylor series can all be expressed in powers between 1 and $n-1$ as we showed above. For example, to compute $\sqrt{A}$ for

$$A=\begin{bmatrix}1&2\\0&4\end{bmatrix},$$

we postulate that we can write (since $n=2$)

$$\sqrt{A}=\alpha_1A+\alpha_2.$$

But, we then should also have a similar equation for the eigenvalues λ, that is,

$$\sqrt{\lambda}=\alpha_1\lambda+\alpha_2.$$

Plugging in the eigenvalues $\lambda_1 = 4$ and $\lambda_2 = 1$, we get the system of equations $4\alpha_1 + \alpha_2 = 2$ and $\alpha_1 + \alpha_2 = 1$. We can solve this system to give $\alpha_1 = 1/3, \alpha_2 = 2/3$. So for this particular matrix A, we have that $\sqrt{A} = 1/3A + 2/3I$.

As another example, we can use Cayley-Hamilton to compute the inverse of a matrix. Take

$$A = \begin{bmatrix} 1 & -2 & 4 \\ 0 & -1 & 2 \\ 2 & 0 & 3 \end{bmatrix}.$$

The characteristic equation is given by

$$\lambda^3 - 3\lambda^2 - 9\lambda + 3 = 0,$$

so we know that the matrix A satisfies

$$A^3 - 3A^2 - 9A + 3I = 0.$$

In other words,

$$I = -\frac{1}{3}(A^2 - 3A - 9)A,$$

which gives

$$A^{-1} = -\frac{1}{3}(A^2 - 3A - 9).$$

(note that this inverse also satisfies $AA^{-1} = I$ as you can easily check).

10.10 Optional: General Convergence Theorem

Now that we know about eigenvalues and eigenvectors, we can come back to the general convergence theorem, first mentioned in Chapter 7. The General Convergence Theorem is also explained in the Jolt on Stationary Iterative methods that you can find on the Mathematics channel of the Jolts pages.

The General Convergence Theorem states that an iterative method $M\vec{x}^{(k+1)} = N\vec{x}^{(k)} + \vec{b}$ converges from any initial guess $\vec{x}^{(0)}$ if and only if $\rho(G) < 1$, with $G = M^{-1}N$. How can we show this? We do this in two parts (and that is typically done for if and only if theorems like this). The proof is not straightforward and uses induced matrix norms and Schur's theorem.

We first start with the "if" side of the theory: the iterative method converges from any initial guess if $\rho(G) < 1$. This is a sufficiency condition meaning that if we satisfy the condition we will converge from any initial guess and if we don't satisfy the condition,

we just don't know meaning that we could converge for some initial guesses and not for others.

To prove this, we use the sufficient convergence criterion derived in the previous section: we converge from any initial guess if we can find a norm for which $\|G\|< 1$. So, assuming that $\rho(G) < 1$, we will construct a particular norm that gives $\|G\|< 1$ and we are done. But, finding such a norm is a little tricky. You will see from the proof that a specific norm is explicitly constructed. The norm we'll use is a so-called induced 1-norm $\|G\|_S \equiv \|SGS^{-1}\|_1$. Here we go.

First, observe that because of Schur's theorem[3], we can write $G = P^{-1}(D + U)P$, where P is some nonsingular matrix, U is strictly upper triangular and D is a diagonal matrix with the eigenvalues of G along its diagonal. We see immediately that $\|D\|_1 = \rho(G)$.

We rewrite the expression for G as $PGP^{-1} = D + U$. Now, we construct a diagonal matrix T with diagonal elements $t_{ii} = \frac{1}{t^i}$, $i = 1, \ldots, n$, for an arbitrary $t \neq 0$. Then, $TPGP^{-1}T^{-1} = D + TUT^{-1}$. The matrix $\hat{U} = TUT^{-1}$ has elements given by

$$\hat{u}_i j = \begin{cases} u_{ij}t^{j-i}, & j > i \\ 0, & j \leq i \end{cases}.$$

Do you see know why we chose T the way we did? Now we can make each element of $\hat{U}$ as small we want as t can be any nonzero number. We can certainly choose t such that $\|\hat{U}\|_1 \leq \epsilon$ for any small $\epsilon > 0$. But, that means that $\|G\|_S = \|SGS^{-1}\|_1 \leq \|D\|_1 + \|\hat{U}\|_1 \leq \rho(G) + \epsilon$. So, we see that $\|G\|_S \leq \rho(G) + \epsilon$, where $\epsilon > 0$. And now we are done with this part of the proof: If $\rho(G) < 1$, then $\|G\|_S < 1$ and we have found a norm that shows us convergence.

To prove the "only if" side of the theory, we should show that if $\rho(G) \geq 1$ there is an initial guess for which we are not converging. In other words, we are simply looking for a counter example. Many necessity proofs are of this form.

How to find this counter example? We should probably be looking a bit at eigenvectors and eigenvalues of G as the spectral radius is simply the largest eigenvalue in absolute value, so there is an immediate connection to explore. So, let's start by assuming $\rho(G) \geq 1$. Then, there is an eigenvalue/eigenvector pair that satisfies $G\vec{z} = \lambda\vec{z}$ with $|\lambda| = \rho(G) \geq 1$. Now we need to find an initial guess $\vec{x}^0$ that destroys our possibilities for convergence. Let's assume that $\vec{x}^0$ is chosen such that $\vec{e}^{(0)} = \vec{z}$ (note that we do not know what this corresponding $\vec{x}(0)$ is as we do not know $\vec{x}$, but this does not matter for the argument). Then, we have $\vec{e}^{(k)} = G^k\vec{e}^{(0)} = \lambda^k\vec{e}^{(0)}$, so that $\|\vec{e}^{(k)}\| \geq \vec{z}\|$ for all k. Clearly this means we cannot converge.

[3] Schur's theorem is often incredibly useful in proofs! It's one of my favorite theorems

So, now we established the General Convergence Theorem. Mind that that theorem has the requirement that convergence must be obtained *from all initial guesses*. It is certainly possible to have convergence from some initial guesses even if $\rho(G) \geq 1$.

10.11 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled ⋆ are more challenging.*

Exercise 10.1

Consider the symmetric matrix

$$A = \begin{bmatrix} 9 & -7 \\ -7 & 17 \end{bmatrix}$$

(a) Compute the eigenvalues and eigenvectors of A. Do this by hand.

(b) Show that the eigenvectors of A are orthogonal.

Exercise 10.2

We consider the matrix

$$A = \begin{bmatrix} -2 & -1 & 0 \\ -1 & -1 & -1 \\ 0 & -1 & -2 \end{bmatrix}.$$

(a) Show that the eigenvalues of A are $\lambda_1 = -3, \lambda_2 = -2, \lambda_3 = 0$ with corresponding normalized eigenvectors $\vec{y}_1 = \frac{1}{\sqrt{3}} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \vec{y}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, \vec{y}_3 = \frac{1}{\sqrt{6}} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$. Is A diagonalizable? Motivate your answer.

(b) Does the limit $\lim_{k\to\infty} A^k$ exist (in other words, is it finite)? Motivate your answer.

(c) Obtain the solution, as a function of time t, to the differential equation $\frac{d\vec{x}}{dt} = A\vec{x}$ with initial condition $\vec{x}(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, and discuss its behavior as $t \to \infty$.

Exercise 10.3

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false, it is sufficient to give one counter example. To show a statement is true, provide a general proof.

(a) If a symmetric matrix has repeated eigenvalues, it must be singular.

*(b) Any $n \times n$ real skew-symmetric matrix A, with n is odd, is singular. (A *skew-symmetric* matrix A satisfies $A^T = -A$.)

Exercise 10.4

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false, it is sufficient to give one counter example. To show a statement is itrue, provide a general proof.

(a) If the $n \times n$ matrix B is formed from the $n \times n$ matrix A by swapping two rows of A, then B and A have the same eigenvalues.

(b) Any invertible $n \times n$ matrix A can be diagonalized.

(c) A singular matrix must have repeated eigenvalues.

*(d) If the $n \times n$ matrices A and B are diagonalizable, so is the matrix AB.

*(e) Let A be an $n \times n$ matrix.

(i) The eigenvalues of A and A^T are the same.

(ii) The eigenvalues and eigenvectors of $A^T A$ and AA^T are the same.

Exercise 10.5

For this problem we assume that eigenvalues and eigenvectors are all real valued.

(a) Let A be an $n \times n$ symmetric matrix. Let $\vec{q}_i$ and $\vec{q}_j$ be the eigenvectors of A corresponding to the eigenvalues λ_i and λ_j respectively. Show that if $\lambda_i \neq \lambda_j$, then $\vec{q}_i$ and $\vec{q}_j$ are orthogonal.

(b) Let A be an $n \times n$ matrix. We say that A is **positive definite** if for any non-zero vector $\vec{x}$, the following inequality holds

$$\vec{x}^T A \vec{x} > 0.$$

Show that the eigenvalues of a positive definite matrix A are all positive.

**(c) Let A be an $n \times n$ matrix. Show that

$$\mathrm{tr}(A) = \sum_{i=1}^{n} \lambda_i,$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A (λ_i's do not have to be all different) and $\mathrm{tr}(A)$ is the trace of the matrix A.

[Hint 1: One way to prove this is to use the fact that any square matrix with real eigenvalues can be decomposed in the following way (called Schur decomposition)

$$A = QRQ^T,$$

where R is an upper triangular matrix and Q is an orthogonal matrix.]

[Hint 2: The following property of trace might be useful: given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times m}$, the trace of their product, $\mathrm{tr}(AB)$, is *invariant under cyclic permutations*, i.e. $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.
Note that this implies $\mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CAB)$ for any matrices A, B, C with appropriately chosen dimension.]

Exercise 10.6

A rabbit population r and a wolf population w are related according to the following system of differential equations:

$$\begin{aligned} \frac{dr}{dt} &= 5r - 2w \\ \frac{dw}{dt} &= r + 2w \end{aligned}$$

Here t denotes time.

(a) If initially (at $t = 0$) there were 100 rabbits and 50 wolves, find the numbers of rabbits and wolves as a function of time t.

(b) Design a matrix A for the above system such that the populations converge to a finite non-zero limit as t goes to infinity.

Exercise 10.7

(a) If $P^2 = P$, show that

$$e^P \approx I + 1.718P$$

(b) Convert the equation below to a matrix equation and then by using the exponential matrix find the solution in terms of $y(0)$ and $y'(0)$:

$$y'' = 0.$$

(c) Show that $e^{A+B} = e^A e^B$ is not generally true for matrices.

Exercise 10.8

Heat Equation. We look at the matrix

$$A = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

(a) Find the eigenvalues of A and the corresponding eigenvectors. A is symmetric, so the eigenvectors should be orthogonal. Check that this is the case.

(b) Give the algorithm of the power method that can be used to find the largest eigenvalue (in absolute value) of A.

(c) Execute this algorithm, either by hand or with MATLAB. As initial vector take one of the unit vectors, produce a figure showing the computed approximations as a function of iteration step. Repeat with an initial vector equal to an eigenvector corresponding to either of the two smallest eigenvalues. Observe that the algorithm still converges. but slower than before (round-off error accumulation helps in this case as we discussed in class).

Note: The matrix is well-conditioned and the round-off error will not accumulate particularly fast. Make sure you force the iteration to run for a while (we have found at least 60 iterations works) because you are only testing the relative error in successive iterates of the eigenvalues.

(d) Instead of looking at this 3×3 matrix, take 10×10 (or larger if you like) matrix with the same tridiagonal structure (-2 on the main diagonal and 1 on the subdiagonals). Find all eigenvalues of this larger matrix using the QR iterations. Check your answers against the eigenvalues computed by matlab using the `eig` command. Again, motivate your convergence criterion.

(e) Since the heat equation discretization was an example of numerically solving a differential equation, the plotting of solutions was rather important. Since we just computed eigenvectors, we might be interested to see what the eigenvectors look like when plotted. Consider 102×102 discretization for the heat equation so that the matrix A is

100×100. Use the `eig` command in matlab to obtain the eigenvectors $\vec{v}^{(1)}, \ldots, \vec{v}^{(100)}$ for $-A$ (this is $-A$ so there are positive 2s on the diagonal). For $j = 1, \ldots, 5$, plot $\sqrt{101}\vec{v}^{(j)}$ as we have done in previous homeworks (remember the boundary values).

Now verify for yourself that the functions $u_j(t) = \sqrt{2}\sin(j\pi t)$ for $j = 1, 2, 3, \ldots$ satisfy the differential equation,

$$-\frac{d^2u_j}{dt^2} = \lambda u_j, \quad \lambda = (j\pi)^2, \quad u_j(0) = u_j(1) = 0, \quad \int_0^1 u_j(t)^2 dt = 1.$$

How do the eigenvectors $\vec{v}^{(j)}$ of $-A$ compare to the "eigenfunctions" $u_j(t)$ of the differential equation? What significance does the factor $\sqrt{101}$ have when plotting $\sqrt{101}\vec{v}^{(j)}$ (think about a Riemann sum approximation of the above integral)?

Exercise 10.9

Consider the matrix

$$A = \begin{bmatrix} -2 & -1 & 0 \\ \alpha & -2 & \alpha \\ 0 & -1 & -2 \end{bmatrix}$$

(a) For which values(s) of α is A singular, and what is the rank of A

(b) For $\alpha = -1$, prove, without computing the eigenvalues and eigenvectors of A, that A is diagonalizable and that the eigenvectors of A are orthogonal.

(c) For $\alpha = -1$, give the orthogonal matrix Q and diagonal matrix Λ for which $A = Q\Lambda Q^{-1}$.

(d) Prove that for any α, the general solution $\frac{d\vec{u}}{dt} = A\vec{u}$ with $\vec{u}(0) = \vec{u}_0$ is given by $\vec{u} = e^{At}\vec{u}_0$, with the exponential matrix defined as $e^{At} = I + tA + \frac{1}{2}(tA)^2 + \frac{1}{3!}(tA)^3 + \ldots$.

(e) For $\alpha = -1$, what is the behavior of the solution to $\frac{d\vec{u}}{dt} = A\vec{u}$ with $\vec{u}(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ as $t \to \infty$?

Exercise 10.10

In this problem we consider a small mass-spring system, consisting of 2 masses that are connected to each other and to fixed walls by three identical springs as shown in figure ??.

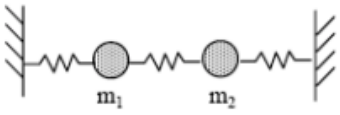


Figure 10.1: Mass-Spring System.

At time $t = 0$, we give the first mass m_1 a small displacement. As a result the mass-spring system will start to move in an oscillatory fashion. If the masses are both equal to 1, the system of equation that describes the displacement of the masses is given by

$$\frac{d^2\vec{u}}{dt^2} = A\vec{u},$$

where

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},$$

with u_1 and u_2 are the displacements of m_1 and m_2, respectively, and

$$A = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}.$$

We take the initial conditions (we need two of them for a second order differential equation) to be

$$\vec{u}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \frac{d\vec{u}}{dt}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

1. Find the matrices Y and Λ such that $A = Y\Lambda Y^{-1}$, with Y orthogonal, and Λ diagonal.

2. Using this decomposition of A, show that we can transform the system of differential equations to

$$\frac{d^2\vec{z}}{dt^2} = \Lambda\vec{z}, \quad \vec{z} = Y^T\vec{u}.$$

   The system for $\vec{z}$ is now "decoupled" so you can solve each component individually since they do not depend on each other. Solve the system of equations for $\vec{z}$ and from it find the displacements u_1 as a function of time t.

3. If the masses are not equal, the system of differential equations describing the motion of the mass-spring system is instead given by

$$M\frac{d^2\vec{u}}{dt^2} = A\vec{u}, \quad M = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}. \tag{10.6}$$

We guess solution will be of the form $\vec{u}(t) = e^{i\omega t}\vec{v}$, where ω and $\vec{v}$ are quantities determined by M and A. Plug in $\vec{u}(t) = e^{i\omega t}\vec{v}$ and show that finding ω and $\vec{v}$ corresponds to the so-called "generalized eigenvalue problem"

$$A\vec{v} = \lambda M\vec{v},$$

with $\lambda = (i\omega)^2$.

Now set $m_1 = 1$ and $m_2 = 2$. The characteristic polynomial for the generalized eigenvalue problem is given by $\det(A - \lambda M) = 0$. Solve the characteristic polynomial to find the two generalized eigenvalues λ_1, λ_2 (or if you wish, the two values of ω since $\lambda = (i\omega)^2$) and the two generalized eigenvectors $\vec{v}_1, \vec{v}_2$. Then give the solution to the differential equation in (10.6).

Exercise 10.11

Heat equation. Let's consider

$$\frac{d^2T}{dx^2} = 0, \text{ for } 0 \leq x \leq 1, \text{ with } T(0) = 1, T(1) = 2.$$

We discretize the equation in the points $x_i = ih$, with $i = 0, 1, \ldots, N$ and $N = \frac{1}{h}$.

(a) Take N = 4. Give the discrete system of equations for the given boundary conditions and the standard second order discretization in the form $A\vec{T} = \vec{b}$, with $\vec{T} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$.

(b) Show that the discretization used in (a) is indeed second order.

*(c) One of the eigenvalues of the matrix A found in (a) is close to -0.5. Give the algorithm of the shifted inverse power method that can be used to find an approximation to this eigenvalue, with an appropriate convergence criterion. You do not have to execute the algorithm.

*(d) What is the convergence rate of the algorithm given in part (d)? Motivate your answer clearly.

* Exercise 10.12

(a) Show that for an $n \times n$ matrix A, the determinant of A is equal to the product of its eigenvalues. That is, show that $\det(A) = \prod_{i=1}^{n} \lambda_i$

(b) Let A be a matrix where for each column, the sum of the elements is equal to a constant, say, μ. Show that μ must be an eigenvalue of A

Exercise 10.13

We will consider the ordinary differential equations $\frac{d\vec{x}}{dt} = A\vec{x}$, with $A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}$. Here, $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ is the vector of unknown functions. The initial condition is given by $\vec{x}(t=0) = \vec{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

(a) Show that A is diagonalizable, and that its eigenvalue are equal to -9 and -27. Give the matrices Y and Λ for which $A = Y\Lambda Y^{-1}$.

(b) Find the solution to the differential equation at time $t = 1$.

(c) Show that the solution to the differential equation converges as $t \to \infty$. What is this converged solution, and what is the associated convergence rate?

* Exercise 10.14

Suppose that the $n \times n$ matrix A is symmetric, has only positive eigenvalues, and that its two largest eigenvalues are identical. Can the Power method still be used to compute the value of these two largest eigenvalues, and if so, what would the convergence rate be? Can the Power method be used to find the corresponding eigenvectors?

Exercise 10.15

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false it is sufficient to give one counter example. To show a statement is true, provide general proof.

(a) If the $n \times n$ matrices A and B are similar, in other words, if there is a $n \times n$ invertible matrix T such that $A = TBT^{-1}$, then A and B have the same eigenvalues.

(b) If Q is an orthogonal matrix, then $Q + \frac{1}{2}I$ is invertible.

(c) If the $n \times n$ matrices A and B have exactly the same eigenvalues and eigenvectors, then $A = B$.

*(d) The eigenvalues of a symmetric and real $n \times n$ matrix A are always real.

Exercise 10.16

Consider the matrix

$$A = \begin{bmatrix} -1 & 1 & 0 \\ \alpha & -2 & 1 \\ 0 & 1 & -1 \end{bmatrix}.$$

(a) For which value of α is pivoting required when computing the LU decomposition of A? For this value, give the permutation matrix P and the final L and U in the equation $PA = LU$.

(b) Take $\alpha = 0$. Use Gram-Schmidt orthogonalization to find an orthonormal basis for the column space of A.

For part (c), (d), and (e), take $\alpha = 1$.

(c) Explain clearly, why A is diagonalizable and give its canonical transform (in other words, give the matrix Y and diagonal matrix Λ for which $A = Y\Lambda Y^{-1}$). Show that the eigenvectors are orthogonal. Normalize them so they have unit length and $Y^{-1} = Y^T$.

(d) Which of the eigenvectors found in part (c) form a basis for the nullspace of A, and which a basis for the row space of A? Explain your answers clearly.

*(e) Use the Cayley-Hamilton theorem to find e^A. Compare the result to e^A obtained with the formula for the matrix exponential we derived in class.

Exercise 10.17

(a) Consider the matrix $A = \begin{bmatrix} 3 & \alpha \\ \alpha & 2 \end{bmatrix}$. We are interested in solving $A\vec{x} = \vec{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. We do this with the Gauss-Seidel iteration. Let $\rho(C)$ be the spectral radius of the amplification matrix C. The spectral radius is defined by the largest eigenvalue (in absolute value) of C. Here, $C = M^{-1}N$, with $A = M - N$. From the General Convergence Theorem, we know that the Gauss-Seidel iteration will converge if $\rho(C) < 1$. Find the values of α for which this is so.

(b) We are now interested in using a new stationary method for solving $A\vec{x} = vb$ for some $n \times n$ matrix A. As suggested by one of the students in class, an interesting stationary

method could be formulated with splitting $A = M - N$, where $M = I$ and $N = I - A$. This splitting leads to the iterative scheme

$$\vec{x}^{(k+1)} = (I - A)\vec{x}^{(k)} + \vec{b}$$

Why is this iterative method attractive from a computational point of view, provided of course that the scheme converges? What are the restrictions on the eigenvalues of A for the scheme to converge? Motivate your answers clearly.

Chapter 11

Case study: the PageRank algorithm

In this chapter, we'll have a closer look at the PageRank algorithm. The original PageRank algorithm was developed by Larry Page, Sergey Brin and their Stanford advisors Rajeev Motwani and Terry Winograd, and discussed in the Stanford Technical Report titled "The PageRank Citation Ranking: Bringing Order to the Web", dated 1999. Sergey and Larry met when they started their graduate studies in the CS department at Stanford in 1995. I was in the CS department at that time too, but did not interact with them, which of course I've always regretted! Soon after entering Stanford, Larry and Sergey started working on search engines and a method to rank webpages, that is, to find the relative importance of webpages. The work resulted in Google in 1998.

11.1 Web searching

When you google for a phrase, the search results are presented in a SERP, a Search Engine Results Page. You all have experience with this. Clearly Google (and other search companies) searches both for exact and related matches. The resulting web pages are ranked in some order. The speed with which the results are obtained is incredible. Of course, if you search for common phrases, Google could store results and simply return to you what was already computed. But also obscure phrases are returned very quickly. This is quite surprising considering that apparently, Google searches through billions of webpages. There are some neat mathematical tricks that speed up the search process. If you are interested, check out this talk titled How does Google Google?.

How does Google determine the order in which the results are given to you in the SERP?

The ranking of the results depends on over 200 different factors. They include the number of times a page is visited, the age of the page, recent edits, and also the so-called *page rank*. Google's secret sauce is the weighting of all these different factors.

11.2 The original page ranking

Perhaps the most intuitive way to rank a webpage would be to count the number of clicks on the page and compare to the visitation of other pages. However, this favors older pages always and would make it difficult for a new page to penetrate the higher ranks of the search results. Instead, the original page ranking is based on the number of *incoming links* to a page.

A webpage typically has hyperlink that you can click on to go elsewhere on the web. Such hyperlinks are *outgoing links*. If other pages have hyperlinks to a webpage, those hyperlinks are called the *incoming links*. It is probably easiest to think of the internet of pages as a graph: each webpage is a node, and arrows in the graph indicate incoming and outgoing links. Figure 11.1 shows an example of a very tiny internet with only six pages. Note that pages 1 is not linked to by any of the pages, so it has no incoming links. We call this a lonely page, or because we work with graphs, a lonely node. Page 6 is a pretty selfish node as it does not link out. Pages 4 and 5 form a mutual admiration society as they only link to each other.

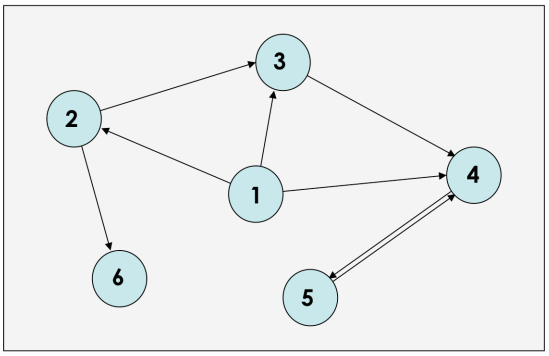


Figure 11.1: An internet graph with just six webpages. Note the lonely webpage at node 1, the selfish page at node 6, and the mutual admiration society of nodes 4 and 5

Now, the reasoning goes as follows: a page will be more important if it has more incoming links, but the incoming links are weighted by the importance of the webpage where they originate. The weighting makes a lot of sense. Otherwise it would be simple to cheat the

system: we can just make a deal that we all link to each other, increasing the importance of our pages. In other words, if your page is linked to from CNN.com that has a lot more weight than if it is linked to from margot.stanford.edu. There is one other thing to keep in mind. If a page gets linked to from an important page like CNN.com, the link is less impressive if CNN.com links to a billion different webpages and the page is merely one of many.

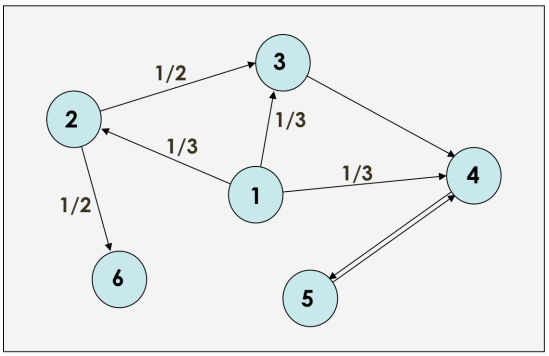
Can we define a formula for the importance x_i of a webpage i that takes all of this into account? How about we set

$$x_i = \sum_{j \in B_i} \frac{1}{N_j} x_j,$$

where B_i is the set of all webpages that link to page i and N_j is the number of outlinks from page j. In other words, if page j has 10 outgoing links, then the importance of page j is equally distributed over the pages it links to. Sometimes, we refer to this as *rank propagation*.

For our wee internet, this gives the following set of equations, shown in Figure **??**.

$$x_1 = 0$$
$$x_2 = \frac{1}{3} x_1$$
$$x_3 = \frac{1}{3} x_1 + \frac{1}{2} x_2$$
$$x_4 = \frac{1}{3} x_1 + x_3 + x_5$$
$$x_5 = x_4$$
$$x_6 = \frac{1}{2} x_2$$



A very small internet "graph"

Figure 11.2: The system of equations corresponding to the small internet with rank propagation. When a propagation factor is not given, it is 1.

We see, for example, that $x_3 = \frac{1}{3}x_1 + \frac{1}{2}x_2$, indicating that whatever the importance of page 1, one third of this importance is allocated to page 3, and page 3 also receives one half of the importance of page 2.

We end up with a coupled system of equations of the form

$$\vec{x} = P\vec{x}, \tag{11.1}$$

where P is called the *propagation matrix*. For this simple example, P is given by

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{11.2}$$

Do you recognize the form of equation 11.1? It says $P\vec{x} = \vec{x} = 1\ \vec{x}$, which indicates that the desired $\vec{x}$ is the eigenvector of P corresponding to the eigenvalue $\lambda = 1$. It turns out that $\lambda = 1$ is the spectral radius of P, and so we could use the Power method to find $\vec{x}$. We will look at this in the exercises.

A lonely node leads to a zero row in the propagation matrix P. See that? A selfish node leads to a zero column. Can you also understand how mutual admiration societies are found? Note also that all nonzero columns add up to one.

The matrix P is also referred to as the *probability matrix*, because the element p_{ij} of the matrix can also be interpreted as the probability that a random crawler will go to page i from page j.

Figure 11.3 shows an example of a large page rank matrix. This is the one that corresponds to a 2014 crawl of the Stanford internet domain. Rather than writing down all the actual numbers in the matrix, we create a spy plot: only nonzeroes are shown and each of them is represented by a single dot. We can clearly see clusters of webpages that are strongly interconnected and also find sparsely connected subsets of pages.

We can also look at the graph corresponding to the Stanford domain. This is shown in Figure 11.4.

11.3 Fixing up the probability matrix

An internet graph is typically a fixer upper. This is easy to see from the small example. Since x_1 is a lonely node, its importance would just be 0. This leads to the solution $x_1 = x_2 = x_3 = x_6 = 0$, and $x_4 = x_5$, which is not very useful.

There are various ways we can fix up the graph, or if you like, fix the matrix. We could pretend that selfish nodes link out, create artificial links to lonely nodes and break up mutual admiration societies. One way to do this is to add artificial links to simply all webpages from every selfish node or mutual admiration society. Clearly then, lonely nodes are no longer lonely either. Since the weights (or probabilities) of these new artificial links

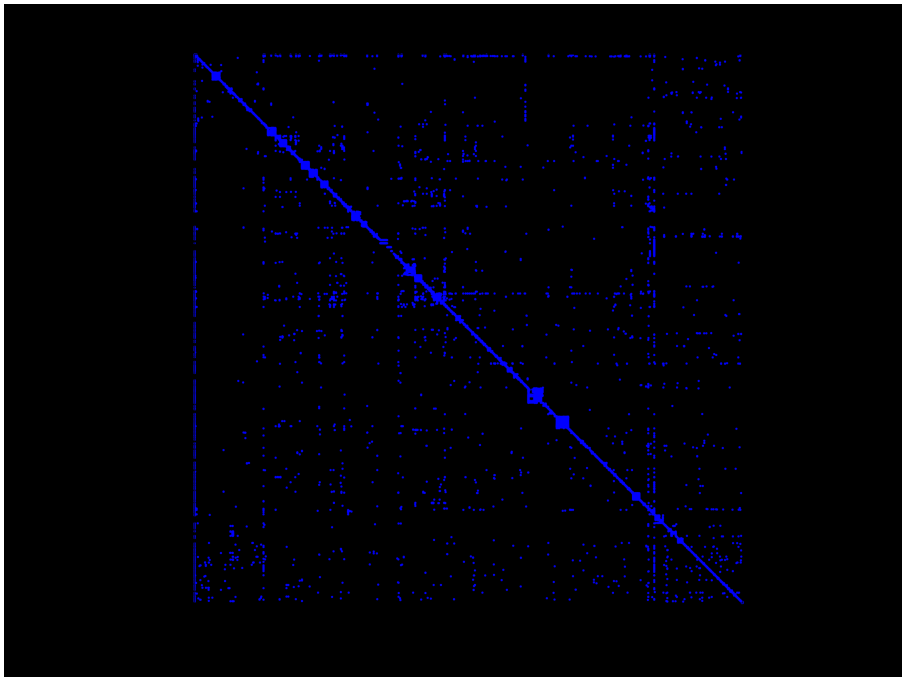


Figure 11.3: Spyplot of the stanford.edu domain. Each dot represents a link from the page in the column to the page in the row.

will be very small, this will not have a significant effect on ranking: it just ensures that we can actually find a reasonable solution.

Another approach, which we will discuss here, is to simply give all pages a small constant importance from the get go. Instead of 11.1, we could write

$$\vec{x} = \alpha P\vec{x} + \vec{v}, \quad \alpha > 0, \tag{11.3}$$

where α is some relaxation parameter that can be adjusted to speed up computations (a bit like the α, or ω factor we encountered in Successive Over Relaxation). Here, $\vec{v}$ is the vector containing a small constant importance that is given to every single page (we could also give this to selective pages, but the simplest solution is to simply give all pages the same). Often, the constant is set to $1/N$, where N is the total number of pages in the internet domain under investigation.

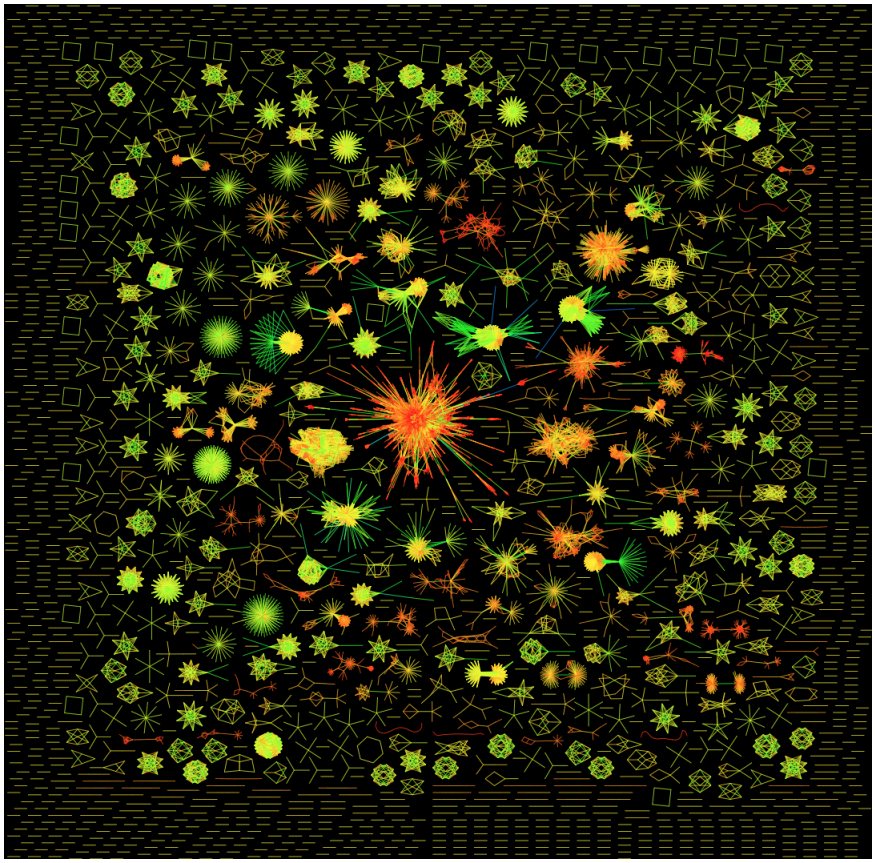


Figure 11.4: The graph of the Stanford domain stanford.edu. Note that the graph is not fully connected.

11.4 Solving for the page rank

How would we solve an equation like 11.3? It looks like a great candidate for an iterative method. We could in fact simply set

$$\vec{x}^{k+1} = \alpha P \vec{x}^k + \vec{v}, \quad k = 0, 1, 2, \cdots \tag{11.4}$$

Iteration 11.4 is a good old Jacobi iteration: it is Jacobi applied to the system $A\vec{x} = \vec{b}$, where $A = (I - \alpha P)$ and $\vec{b} = \vec{v}$. The matrix A must be convertible for the solution to be unique. We prove in the exercises in this chapter that it is.

The amplification matrix is $G = \alpha P$. We know that the spectral radius of P is 1. Hence the spectral radius of αP is α. Naturally, we must take $\alpha < 1$.

When we apply this to the example with $\alpha = 0.85$, which is a very commonly used value, we set the elements of $\vec{v}$ equal to $1/N = 1/6$ and select $\vec{x}^0 = \vec{v}$, then we obtain the following iterates:

$$\vec{x}^1 = \begin{bmatrix} .17 \\ .21 \\ .28 \\ .50 \\ .31 \\ .24 \end{bmatrix}, \quad \vec{x}^2 = \begin{bmatrix} .17 \\ .21 \\ .30 \\ .72 \\ .58 \\ .26 \end{bmatrix}, \quad ,\cdot, \quad \vec{x}^\infty = \begin{bmatrix} .17 \\ .21 \\ .30 \\ 2.2 \\ 2.0 \\ .27 \end{bmatrix}.$$

As expected, page 1 does not get anymore than what it initially receives. Page 6, with only page 2 linking to it, is ranked low as well, and pages 4 and 5 are ranked highest.

We know from the discussion of Jacobi that it is a relatively slow iteration. Why is Jacobi so popular here? Well, the big advantage of Jacobi is that it is trivially parallelizable: all elements of the new iterate $\vec{x}^{k+1}$ can be computed independently. It therefore performs extremely well on an ultra-large parallel computer, and search companies have built custom-made ultra-large parallel systems to solve equations like this.

It used to be possible to check the page rank of any of your webpages through the Google bar. Lately that has become harder. The page ranks that are returned by Google are normalized Page ranks on a log scale.

11.5 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled ⋆ are more challenging.*

Exercise 11.1

The unfixed page rank equation is $P\vec{x} = \vec{x}$. We reasoned that this indicates that $\vec{x}$ is the eigenvector of P corresponding to $\lambda = 1$. We also know that $\lambda = 1$ is the largest eigenvalue. For the P in the wee internet example, implement the Power iteration to find $\vec{x}$. Run it for various initial guesses. What do you observe? Explain the behavior. Hint: compute the eigenvalues of P.

* Exercise 11.2

We use the Jacobi iteration 11.4 to solve the page rank system $A\vec{x} = \vec{v}$, with $A = I - \alpha P$.

(a) The converged solution is $\vec{x}^{\infty}$. Analyze the distance between $\vec{x}^{\infty}$ and successive iterates of the Jacobi iteration, that is, find the relationship between $\|\vec{x}^{(k+1)} - \vec{x}^{\infty}\|_1$ and $\|\vec{x}^{(k)} - \vec{x}^{\infty}\|_1$. Note that we measure distance in terms of the 1-norm here. Your analysis must hold for any $n \times n$ page rank matrix P. Form your analysis, show that α must be chosen to be smaller than 1.
(Hint: first compute the 1-norm of P.)

(b) Now, show that the matrix $I - \alpha P$ is invertible for any $n \times n$ page rank matrix P.
(Hint: don't forget that we have $0 < \alpha < 1$.)

* Exercise 11.3

For this problem, please refer back to Exercise 11.2. The solutions will help you in coding your PageRank algorithm.

Recall the linear system for Page Rank:

$$(I - \alpha P)\vec{x} = \vec{v}$$

where $\vec{v}$ the initial rank we give to each page. In our problem, we set $\alpha = 0.85$ and the entries of $\vec{v}$ to be $\vec{v}_i = \frac{1}{N}$, with N the total number of pages in our network.

The PageRank calculation need not only be used for the internet! In fact, PageRank can be calculated for nodes on any connected graph representing any abstraction. For this problem, we will consider a graph of movies connected by links if they share at least one common actor. For this purpose, we provide a matlab date file `movies.mat` that can be downloaded from Coursework (this file will later be put in a github repository for non-Stanford readers). Place this file in a local directory accessible in matlab and type `clear all; load movies.mat`. If you look in the workspace, you should have a collection of new variables, defined as follows:

- `links` : rows are (movie, person) pairs (e.g., for `links(1,:)` equal to [779,20278] means that actor personName20278 was in movie movieName779) (James Jimmy Stewart in the movie Rope)
- `movieIMDbID` : the IMDb IDs of each movie
- `movieName` : the name of each movie
- `movieRating` : the average IMDb rating of people who have rated this movie online
- `movieVotes` : the number of people who have rated this movie on IMDb
- `movieYear` : the year in which this movie was released
- `personIMDBID` : the IMDB IDs of each actor/actress
- `personName` : the name of each actor/actress

None of these are the proper PageRank matrix P.

(a) Let C be the $m \times n$ matrix defined by $C_{ij} = 1$ if movie i contains actor/actress j. Let B be the $m \times m$ matrix where B_{ij} is the number of actors starring in both movie i and movie j.

  (i) Show how to calculate B from C.

  (ii) Show how to calculate the PageRank matrix P from B: (Hint: it may help to construct a small graph of movies and actors (need not be based on real data) and to actually construct these individual matrices). Remember that movie i and movie j sharing at least one actor counts as one link from movie i to movie j.

(b) Using matlab, construct the PageRank matrix P. Create the spy plot of P using the command `spy(P)`. Use sparse matrix commands to assist you; otherwise, matlab may be temperamental.

(c) Compute the PageRank vector $\vec{x}$ of the movies in this graph and normalize this quantity so $||\vec{x}||_1 = 1$. List the PageRank values and titles of the movies with the five highest PageRank values.

(d) Compute the PageRank vector $\vec{x}$ of the movies via a Jacobi iteration that you write yourself and normalize this quantity so $||\vec{x}||_1 = 1$ after each iteration. Decide on an intelligent measure for convergence (assume you do not know the actual PageRank vector $\vec{x}$ because your system is too large for a simple backslash calculation.) Explain your choice of convergence criterion. Next, plot this convergence measure over the steps in your iteration. How many iterations does it take your implementation to get to a tolerance of 10^{-4}.

(e) Plot IMDb movie rating vs. PageRank and IMDb movie votes vs. PageRank. Is PageRank a good predictor of movie rating or movie votes?

Chapter 12

The Singular Value Decomposition

Finally, we look at one other decomposition, the *Singular Value Decomposition*. This is just the wee dessert for the course. It is a very powerful decomposition, and you will find it useful in advanced courses.

12.1 Another look at $A\vec{x}$

When applying A to $\vec{x}$, we generally get a vector with a different direction and different length than $\vec{x}$. But we have now also seen that for some $\vec{x}$, the eigenvectors, $A\vec{x}$ is still oriented along the same line as $\vec{x}$.

We can look at $A\vec{x}$ in a slightly different way here and that gives us some additional insight. Let's take all vectors along the unit circle, that is, all $\vec{x}$ with $\|\vec{x}\|_2 = 1$. What does A do to all of these vectors? We don't expect the circle to be preserved when we let A loose on the vectors. But what does A create out of the circle? Any interesting shape, or does it perturb it in an unstructured way?

We will look at a 2×2 example with $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ a generic matrix. We set $\vec{z} = A\vec{x}$ and find that $\vec{z} = \begin{bmatrix} ax_1 + bx_2 \\ cx_1 + dx_2 \end{bmatrix}$. We assume A is nonsingular, so A^{-1} exists. Then, $\vec{x} = A^{-1}\vec{z}$, and we know that $\|A^{-1}\vec{z}\|_2 = 1$. Computing $A^{-1}\vec{z}$ we get $A^{-1}\vec{z} = \frac{1}{ad-bc}\begin{bmatrix} dz_1 - bz_2 \\ -cz_1 + az_2 \end{bmatrix}$.

Then $\|A^{-1}\vec{z}\|_2^2 = 1$ gives, after some algebra

$$(c^2+d^2)z_1^2 + (b^2+a^2)z_2^2 - 2(ac+bd)z_1z_2 - (ad-bc)^2 = 0.$$

What is interesting is that this is the equation for an ellipse. The generic ellipse equation is $\alpha x^2 + \beta xy + \gamma y^2 + \delta x + \epsilon y + \xi = 0$, with $\beta^2 - 4\alpha\gamma < 0$. Here, the constraint gives $-4(ad-bc)^2 < 0$, which is of course always the case if A is invertible.

So, we see that the unit circle is transformed into an ellipse centered at the origin.

12.2 Singular values and vectors in 2D

Figure 12.1 shows such an ellipse.

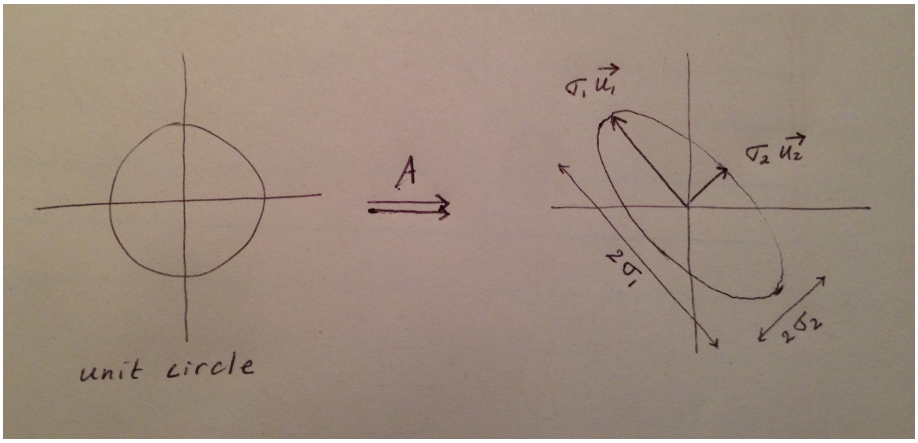


Figure 12.1: Another smashing illustration by MG

The major axis (the largest diameter) is $2\sigma_1$. The smallest diameter is $2\sigma_2$. So $\sigma_1 \geq \sigma_2 \geq 0$. The vectors $\vec{u}_1$ and $\vec{u}_2$ are vectors along the two axes and of unit length. In an ellipse, these two vectors are orthogonal to each other.

Now we know that the ellipse is formed by applying A to the unit circle. So, along the unit circle are two vectors, which we will denote by $\vec{v}_1$ and $\vec{v}_2$, such that

$$A\vec{v}_1 = \sigma_1\vec{u}_1,$$
$$A\vec{v}_2 = \sigma_2\vec{u}_2.$$

To create an ellipse from a unit circle, A applies some combination of stretch/squeeze and rotation to the unit circle. Thinking about it this way, and knowing that the vectors $\vec{v}_1$

and $\vec{v}_2$ produce orthogonal $\vec{u}_1$ and $\vec{u}_2$ it is not hard to convince yourself that $\vec{v}_1$ and $\vec{v}_2$ must themselves be orthogonal.

We can write

$$A\,[\vec{v}_1\ \vec{v}_2] = [\vec{u}_1\ \vec{u}_2] \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix},$$

or $AV = U\Sigma$, or

$$A = U\Sigma V^T. \tag{12.1}$$

This is a new decomposition! it is called the singular value decomposition, or SVD.

If the matrix A is singular, one of the singular values will be zero (if both are zero, A must be the null matrix), and the ellipse collapses into a line segment.

12.3 Singular values and vectors for general A

We looked at just a 2×2 case, but the same can be done for any matrix A and more generally for $m \times n$ matrices A. An $m \times n$ matrix A will take a unit circle in $\mathbb{R}^n$ and create a hyper ellipse out of it in $\mathbb{R}^m$. Each semi-axis of this hyper ellipse that is not collapsed (so has a nonzero length), corresponds to one nonzero singular value. We can draw the conclusion that the number of non collapsed semi-axis is exactly the rank of the matrix. In other words, the number of nonzero singular values gives the rank of the matrix. This is extremely handy and is typically used to determine singularity for large matrices. At the start of the book, we discussed the determinant as a singularity indicator: if the determinant is zero, the matrix is singular. In practice the singular value check is preferred as it tends to be less sensitive to round-off error and gives not just whether a matrix is singular or not, but also the rank of the matrix.

We always number the singular values from big to small. So, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. Again, we can write

$$A\vec{v}_i = \sigma_i \vec{u}_i,\ i = 1, \ldots, r,$$

which gives

$$A\,[\vec{v}_1\ \ldots\ \vec{v}_r] = [\vec{u}_1\ \ldots\ \vec{u}_r] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}.$$

We could regard this as a skinny version of the SVD. We can easily augment the matrices

with vectors $\vec{v}$ and $\vec{u}$ so that the matrices become square and orthogonal. We write

$$A\,[\vec{v}_1\ \ldots\ \vec{v}_r\ \vec{v}_{r+1}\ \ldots\ \vec{v}_n] = [\vec{u}_1\ \ldots\ \vec{u}_r\ \vec{u}_{r+1}\ \ldots\ \vec{u}_m] \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}.$$

This will give $AV = U\Sigma$, where V is an orthogonal $n \times n$ matrix, U an orthogonal $m \times m$ matrix and Σ an $m \times n$ matrix. The columns of U are also referred to as the left singular vectors, and the columns of V as the right singular vectors.

With $A = U\Sigma V^*$, we can interpret $A\vec{x}$ a little differently. We get

$$A\vec{x} = U\Sigma V^*\vec{x}.$$

When A acts on $\vec{x}$ it can be interpreted as first applying an orthogonal matrix V^*. This can be a rotation or reflection. In any case, it preserves the unit sphere. After this Σ is applied. This is the stretch/squeeze operator. Finally, another orthogonal operator is applied, which is again a rotation or reflection. This is a lot easier to see with an example. Let's take the matrix, in MATLAB notation, $A = [1, 3; -1, 1.25]$. In figure 12.2, we see the unit circle and the ellipse created by application of A to all unit vectors on the circle.

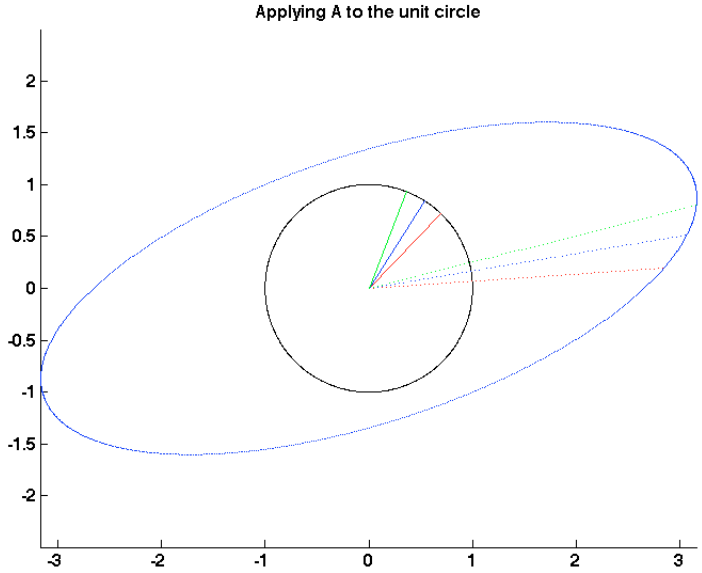
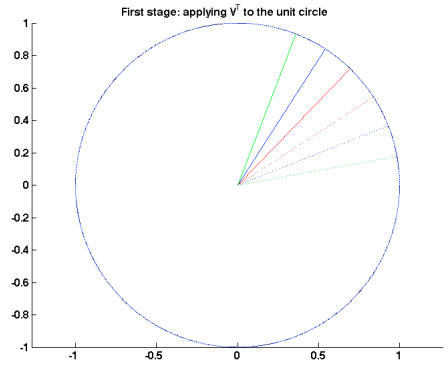
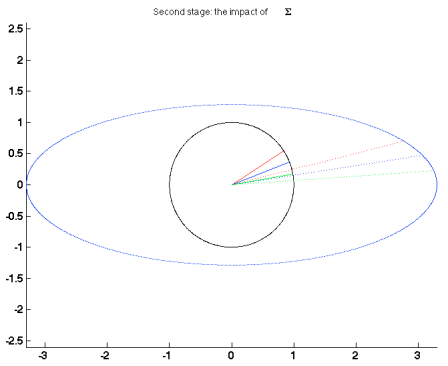
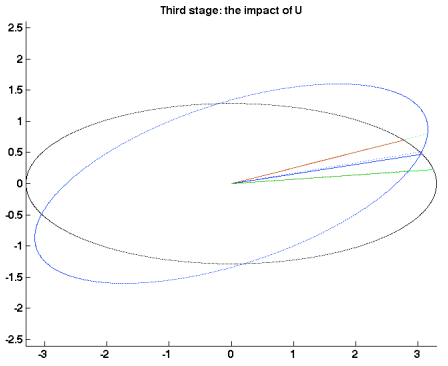


Figure 12.2: Applying $A = [1, 3; -1, 1.25]$ to the unit circle deforms it into an ellipse. Also shown are three vectors (solid lines) and their transformation after application of A (dotted lines). Judging from this picture, what are the singular values of A?

We now look at the three stages: first we apply V^T, then Σ and finally U. The results are shown in figures 12.3 through 12.3.







It turns out that the column space of A is spanned by the vectors $\vec{u}_1$ through $\vec{u}_r$. Can

you prove this? It is a nice exercise. Can you also see which vectors span the nullspace of A?

Another interesting observation is that A^*A is similar to $\Sigma^*\Sigma$. Check this. What does this tell you? Think about the eigenvalues of $\Sigma^*\Sigma$.

The SVD is extremely handy. In contrast to the diagonalization, the SVD is also applicable to matrices that are not square. If we could compute the singular values, we would immediately know the rank of a rectangular matrix, as it is simply the number of nonzero singular values.

So, to reiterate: eigenvalues and eigenvectors can only be computed for square matrices, whereas singular values and vectors can be computed for any sized matrix, and the number of nonzero singular values gives you the rank of the matrix.

12.4 SVD in image compression

The SVD formed the backbone for one of the first image compression schemes. To understand this, we should know that the expression $A = U\Sigma V*$ can also be written as

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^* + \ldots + \sigma_r \vec{u}_r \vec{v}_r^* = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_j^*,$$

that is as a linear combination of rank 1 matrices. A black and white image can be represented as a matrix: each element of the matrix corresponds to a pixel and contains a value that indicates the gray scale of the pixel (for example, 0 for pure white and 1 for pure black and everything in between). Color photographs are stored in three matrices, one for each of RGB. Suppose that we compute the SVD of this image matrix. We very often find that many of the nonzero singular values are small and only a few are significant. We can then truncate the above summation and only keep the terms with the large singular values. Perhaps only σ_1 and σ_2 are significant, then we could approximate the image matrix A by

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^* + \sigma_2 \vec{u}_2 \vec{v}_2^*.$$

The image now only requires the storage of four vectors and two scalars instead of an $m \times n$ matrix. A considerable savings.

The figures show a couple of examples of compression using the SVD. Current jpeg or other image compression methods do no longer use the SVD (instead they use the Discrete Cosine Transform, or wavelets). But the SVD is still used extensively for compression in other areas of engineering and the sciences, for example, in reduced order modeling.

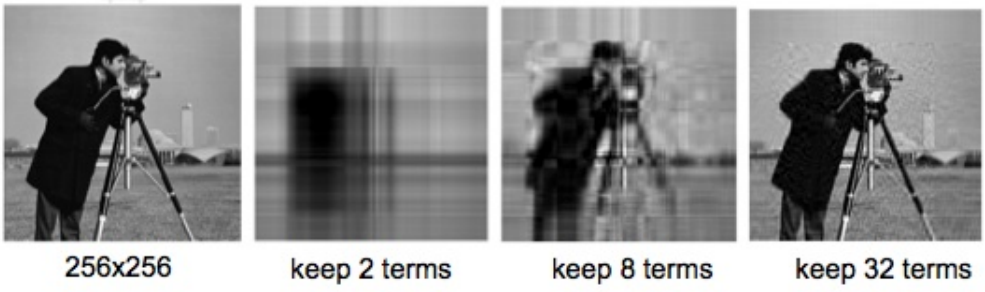


Figure 12.3: Example compression of black/white photograph

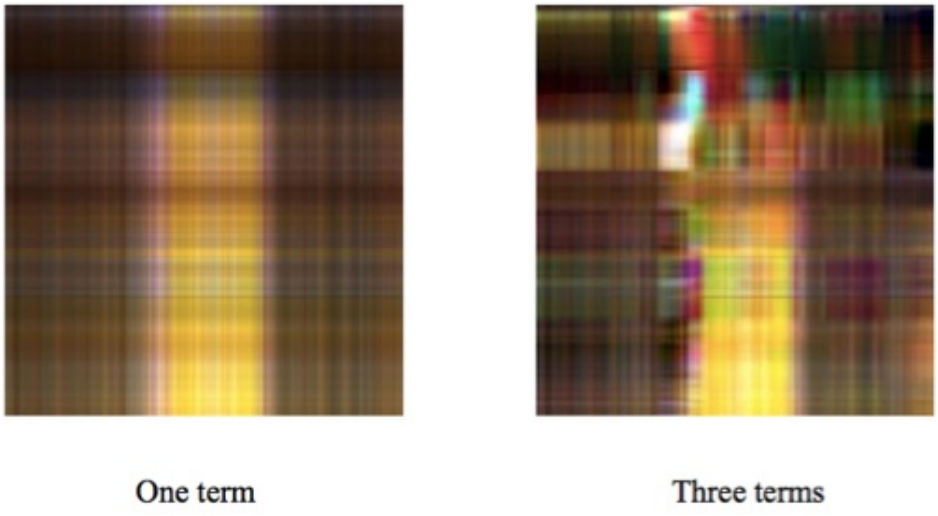


Figure 12.4: Very rough compression

12.5 Exercises

The exercises contain both theoretical questions, including proofs, and applications. Answers to the even-numbered exercises are provided at the end of the book. Try the exercises yourself first and discuss with others before looking at the solutions. *Problems labeled* $\star$ *are more challenging.*

* Exercise 12.1

The singular value decomposition of the $m \times n$ matrix A is given by $A = U\Sigma V^T$, where

Figure 12.5: For this color picture we definitely need more terms

the $m \times m$ matrix U and the $n \times n$ matrix V are orthogonal. The matrix Σ is an $m \times n$ matrix with the singular values σ_i on its diagonal. For simplicity, assume that $m \geq n$, so that A has n singular values.

(a) Prove that r, the rank of A, is equal to the number of nonzero singular values.

(b) Prove that the column space of A is spanned by the first r columns of U, where r is the rank of A.

Exercise 12.2

Let A be an $m \times n$ matrix. Assume $n > m$ and the rank of A is m. In this case, $A\vec{x} = \vec{b}$ has an infinite number of solutions. We are interested in finding the solution to $A\vec{x} = \vec{b}$ that has minimum norm., We will use 2-norm here. The solution we after is called the "minimal 2-norm solution".
Show that the minimal 2-norm solution is given by $\vec{x} = V \begin{bmatrix} S^{-1}U^T\vec{b} \\ 0 \end{bmatrix}$, where S is the $m \times m$ matrix containing the first m columns of Σ.

* Exercise 12.3

Indicate whether the following statement is TRUE or FALSE and motivate your answer clearly. To show a statement is false it is sufficient to give one counter example. To show a statement is true, provide a general proof.

If an $n \times n$ matrix A is symmetric, its singular values are the same as its eigenvalues.

* Exercise 12.4

In this chapter, we discussed that the number of nonzero singular values of a matrix A gives the rank of A. This is not generally true for eigenvalues. That is, if $n \times n$ matrix A has p nonzero eigenvalues, we cannot conclude that the rank of A equals p, *unless* A belongs to a special class of matrices. Explain why this must be so, and identify this special class of matrices.

Chapter 13

Review: Test Your Knowledge

This review chapter gives a hotchpotch of exercises that jump throughout the book. Choose random selections to test your knowledge. The exercises are jumbled up on purpose and many combine material from different chapters.

Exercise 13.1

Power method

In one of the exercises, we looked at the Power method for a matrix that has $\lambda_1 = \lambda_2$. What would happen if $\lambda_1 = -\lambda_2$, in other words, what would happen if the eigenvalues are the same in absolute value, but have a different sign?

Exercise 13.2

Canonical transform and SVDs

We know that a symmetric matrix has a canonical transform of the type $A = Y\Lambda Y^{-1}$, with $Y^{-1} = Y^T$. We also know that a symmetric matrix has real eigenvalues, but the eigenvalues could of course be negative. Every matrix, so also a symmetric matrix, has an SVD of the form $A = U\Sigma V^T$. Here, the singular values that appear on the diagonal of Σ are real and always nonnegative. Can you find this singular value decomposition (so the matrices U, V and Σ for a symmetric matrix from its canonical transform?

Exercise 13.3

Symmetric matrices

(a) Use the Schur form to prove that the eigenvectors of a symmetric matrix can always be chosen to be orthogonal.

(b) Now show the same as in question (a) but only by using the relations $A = A^T$ and $A\vec{y}_i = \lambda_i \vec{y}_i$ that $\vec{y}_i^T \vec{y}_j = 0$ when $i \neq j$. You may assume that the eigenvalues are distinct in this case.

(c) The symmetric matrix A is given in MATLAB notation as

```
A = [3, 1, 0; 1, 3, 0; 0, 0, 2].
```

Compute the eigenvectors and eigenvalues of A. According to problems (a) and (b), we should be able to get an orthogonal set of eigenvectors. Check this.

(d) Suppose that A is a symmetric matrix with eigenvalues $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 2$. The corresponding eigenvectors are $\vec{y}_1, \vec{y}_2,$ and $\vec{y}_3$. Show that both the column and the row spaces of A are spanned by $\vec{y}_2$ and $\vec{y}_3$.

Exercise 13.4

True/false questions jumping around the material

Indicate whether the following statements are TRUE or FALSE and motivate your answers clearly. To show a statement is false it is sufficient to give one counter example. To prove a statement true, provide general proof.

(a) If A has the same nullspace as A^T, then A must be a square matrix.

(b) If the triangular matrix T is similar to a diagonal matrix, T must be diagonal itself.

(c) If all eigenvalues of a $n \times n$ matrix A are positive, then $A\vec{x} = \vec{b}$ always has a solution for any $\vec{b}$.

(d) If the $n \times n$ matrix A is symmetric, then the matrix R in the QR decomposition of A is diagonal.

(e) If A is symmetric and positive definite, and Q is an orthogonal matrix then QAQ^T is symmetric but not necessarily positive definite.

(f) The eigenvalues of A are the same as the eigenvalues of U. Here U is the matrix after Gaussian Elimination. You may assume that pivoting is not required.

* Exercise 13.5

Eigenvalues and Gaussian Elimination

Show that the product of the eigenvalues of a matrix A is equal to the product of the pivots in Gaussian elimination. You may assume that pivoting is not required.

Exercise 13.6

Positive-definite matrices

Positive-definite matrices occur on a regular basis in science and engineering. A matrix B is *positive definite* if for *all* $\vec{x} \neq \vec{0}$, we have that $\vec{x}^T B\vec{x} > 0$. Here, we assume that all matrices and vectors we work with are real.

(a) Show that positive definite matrices have only positive eigenvalues.

(b) Is it true that if a $m \times n$ matrix A has rank n, then the matrix $A^T A$ is symmetric and positive definite?

Exercise 13.7

Orthogonalization

(a) Suppose that we are performing Gram-schmidt on a matrix A that is 10×10. Suppose that the third column of A is linearly dependent on the other nine columns. Explain what happens when you get to this column in QR, and what you could do to resolve the conundrum.

(b) QR can be used to solve systems of equations $A\vec{x} = \vec{b}$, with A nonsingular. In the notes it is explained how. Try this method for a few nonsingular matrices in MATLAB. You really have two direct (non-iterative) methods to solve $A\vec{x} = \vec{b}$: LU and QR. LU is often well understood, but very few remember that QR is also possible and equally attractive in most cases. Hence, this little play to consolidate it in your mind. Can you think of matrices for which LU is more attractive than QR, and perhaps vice versa?

Exercise 13.8

Diagonalizability

In the notes, it is shown that symmetric matrices are diagonalizable using the Schur form. We can do the same for an Hermitian matrix A: $A = USU^*$ and $A = A^*$ give $USU^* = US^*U^*$ so that $S = S^*$ and therefore A is diagonalizable. Fill in the gaps, that is, explain how we transition from one statement to the next in this proof.

Exercise 13.9

Decompositions

Collect all decompositions of a matrix you have seen thus far in the book, and for each of them list/discuss:

- uses
- computational complexity
- special cases (eg. does it simplify if the matrix is symmetric? any other special cases we discussed?)

If uses for decompositions overlap (eg. both are used to solve some system) then discuss pros and cons as well.

Exercise 13.10

Complex matrices

(a) Find two complex Hermitian matrices that are 2×2 and 3×3. Are these matrices also symmetric? Repeat the same for unitary matrices.

(b) For orthogonal matrices Q, we found that $\|Q\vec{x}\|_2 = \|\vec{x}\|_2$. Convince yourself of this again. Does a similar relation hold for unitary matrices U? In other words, if U is unitary, is it true that $\|U\vec{x}\|_2 = \|\vec{x}\|_2$, where $\vec{x}$ is a complex vector?

Exercise 13.11

Systems of ODEs

(a) Find a 2×2 matrix A that has 2 distinct eigenvalues. Compute its canonical form.

(b) For the matrix you choose, compute the solution to $\frac{d\vec{x}}{dt} = A\vec{x}$, with initial condition $\vec{x}(0) = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$.

(c) Find a 2×2 matrix B that has a repeated eigenvalue but is still diagonalizable. Again, compute the solution to $\frac{d\vec{x}}{dt} = B\vec{x}$, with initial condition $\vec{x}(0) = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$. What is $\vec{x}$ at time $t = 10$?

(d) Find a 2×2 matrix C that has a repeated eigenvalue and is not diagonalizable. Again, compute the solution to $\frac{d\vec{x}}{dt} = C\vec{x}$, with initial condition $\vec{x}(0) = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$. What is $\vec{x}$ at time $t = 10$?

Exercise 13.12

Determinants

(a) Why is the determinant of a triangular matrix equal to the product of the diagonal elements? Do a few examples for 2×2 case, and then argue that it holds for any $m \times m$ case.

(b) Explain in your own words why it makes sense that matrices that are singular have a zero determinant.

*(c) Show that $\det(AB) = \det(A)\det(B)$? This is not as easy as it looks.

(c) Explain why the determinant is equal to the product of the pivots in LU decomposition (assume $A = LU$ so no row swapping was needed in the Gaussian elimination process). You may use the result of (c).

(d) A matrix A is singular if $|A| = 0$ but if the determinant is not zero, we cannot say anything about the matrix being close to singular or not. This is an important observation. Give an example of a matrix that is far from being singular, yet has a tiny determinant, and an example of a matrix that is very close to being singular, but has a large determinant.

Exercise 13.13

Stationary iterative methods

We know that both Jacobi and Gauss-Seidel converge for diagonally dominant matrices. Find a diagonally dominant 3×3 matrix (or larger if you can use MATLAB) and compare the convergence rates (here, the norms of the amplification matrices). Do you expect GS to converge faster than Jacobi? Does the answer make sense? Now, increase the diagonal dominance by making the diagonal heavier. Check again. Do you see a faster convergence overall? Is that expected?

* Exercise 13.14

Random questions

(a) Suppose that for the matrices C and D, it holds that $CD = -DC$. Is there a flaw in the following argument:
"Taking determinants gives $\det(CD) = \det(C)\det(D) = -\det(DC) = -\det(D)\det(C)$ *so we end up with* $\det(C)\det(D) = -\det(C)\det(D)$*, which means that eigher C or D must have a zero determinant. Therefore $CD = -DC$ is only possible if C or D is singular."*

(b) A matrix A is given by $A = \vec{u}\vec{v}^T$, where $\vec{u}$ and $\vec{v}$ are in $\mathbb{R}^n$. Show that this matrix always has either rank 0 or rank 1, and give examples for these two cases.

(c) We have seen in the least squares discussions that $P = A\left(A^TA\right)^{-1}A^T$ is the projection matrix that projects onto the column space of A provided A is full rank. What is the projection matrix that projects onto the *row* space of A?

(d) Is the following statement true or false? For any $n \times n$ matrix A, the eigenvalues of A and its transpose are always the same.

Exercise 13.15

Spaces and dimensions

(a) Prove that for any $m \times n$ matrix A, the dimensions of the row space and the column space of A are the same.

(b) The dimension of the nullspace of the $m \times n$ matrix A is $n - r$, where r is the rank of the matrix. Take $m > n$.

  (i) Create a matrix that is full rank (so has rank n). If $\vec{b}$ is in the column space of A, is the solution to $A\vec{x} = \vec{b}$ unique? Is every $\vec{b}$ in the column space of A? Explain.

  (ii) Create a matrix that is not full rank (so has rank $< n$). Repeat the questions in (i).

Exercise 13.16

QR decomposition

(a) Derive the full QR decomposition for the matrix

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

(b) We have an $m \times n$ matrix A. Is it true that if A is full rank, the skinny QR decomposition and the full QR decomposition are always the same?

(c) In the Gram-Schmidt process, we project vectors onto lines. What is the corresponding projection matrix?

Exercise 13.17

Symmetric matrices

Go through the notes and collect everything you know about symmetric matrices, then answer the following questions:

(a) How can you take advantage of symmetry in the LU decomposition?

(b) Can you take advantage of symmetry in the QR decomposition?

(c) Is there anything special you can do for least squares when you have a symmetric matrix?

(d) What do you know about the various spaces of a symmetric matrix (row, column, null)?

(e) Can you say anything about singularity of a symmetric matrix?

(f) Can you say anything about the determinant of a symmetric matrix?