Chapter IV

Eigenvalues and Eigenvectors

IV.1. Eigenvalues and Eigenvectors

Prerequisites and Learning Goals

After completing this section, you should be able to

- Write down the definition of eigenvalues and eigenvectors and compute them using the standard procedure involving finding the roots of the characteristic polynomial. You should be able to perform relevant calculations by hand or using specific MATLAB/Octave commands such as `poly`, `roots`, and `eig`.
- Define algebraic and geometric multiplicities of eigenvalues and eigenvectors; discuss when it is possible to find a set of eigenvectors that form a basis.
- Determine when a matrix is diagonalizable and use eigenvalues and eigenvectors to perform matrix diagonalization.
- Recognize the form of the Jordan Canonical Form for non-diagonalizable matrices.
- Explain the relationship between eigenvalues and the determinant and trace of a matrix.
- Use eigenvalues to compute powers of a diagonalizable matrix.

IV.1.1. Definition

Let A be an $n \times n$ matrix. A number λ and non-zero vector $\mathbf{v}$ are an eigenvalue eigenvector pair for A if

$$A\mathbf{v} = \lambda\mathbf{v}$$

Although $\mathbf{v}$ is required to be nonzero, $\lambda = 0$ is possible. If $\mathbf{v}$ is an eigenvector, so is $s\mathbf{v}$ for any number $s \neq 0$.

Rewrite the eigenvalue equation as

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

Then we see that $\mathbf{v}$ is a non-zero vector in the nullspace $N(\lambda I - A)$. Such a vector only exists if $\lambda I - A$ is a singular matrix, or equivalently if

$$\det(\lambda I - A) = 0$$

IV.1.2. Standard procedure

This leads to the standard textbook method of finding eigenvalues. The function of λ defined by $p(\lambda) = \det(\lambda I - A)$ is a polynomial of degree n, called the characteristic polynomial, whose zeros are the eigenvalues. So the standard procedure is:

- Compute the characteristic polynomial $p(\lambda)$
- Find all the zeros (roots) of $p(\lambda)$. This is equivalent to completely factoring $p(\lambda)$ as

  $$p(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n)$$

  Such a factorization always exists if we allow the possibility that the zeros $\lambda_1, \lambda_2, \ldots$ are complex numbers. But it may be hard to find. In this factorization there may be repetitions in the λ_i's. The number of times a λ_i is repeated is called its *algebraic multiplicity*.
- For each distinct λ_i find $N(\lambda I - A)$, that is, all the solutions to

  $$(\lambda_i I - A)\mathbf{v} = \mathbf{0}$$

  The non-zero solutions are the eigenvectors for λ_i.

IV.1.3. Example 1

This is the typical case where all the eigenvalues are distinct. Let

$$A = \begin{bmatrix} 3 & -6 & -7 \\ 1 & 8 & 5 \\ -1 & -2 & 1 \end{bmatrix}$$

Then, expanding the determinant, we find

$$\det(\lambda I - A) = \lambda^3 - 12\lambda^2 + 44\lambda - 48$$

This can be factored as

$$\lambda^3 - 12\lambda^2 + 44\lambda - 48 = (\lambda - 2)(\lambda - 4)(\lambda - 6)$$

So the eigenvalues are 2, 4 and 6.

These steps can be done with MATLAB/Octave using `poly` and `root`. If `A` is a square matrix, the command `poly(A)` computes the characteristic polynomial, or rather, its coefficients.

```
> A=[3 -6 -7; 1 8 5; -1 -2 1];
> p=poly(A)

p =

    1.0000  -12.0000   44.0000  -48.0000
```

Recall that the coefficient of the highest power comes first. The function `roots` takes as input a vector representing the coefficients of a polynomial and returns the roots.

```
>roots(p)

ans =

   6.0000
   4.0000
   2.0000
```

To find the eigenvector(s) for $\lambda_1 = 2$ we must solve the homogeneous equation $(2I - A)\mathbf{v} = \mathbf{0}$. Recall that `eye(n)` is the $n \times n$ identity matrix I

```
>rref(2*eye(3) - A)

ans =

   1   0  -1
   0   1   1
   0   0   0
```

From this we can read off the solution

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Similarly we find for $\lambda_2 = 4$ and $\lambda_3 = 6$ that the corresponding eigenvectors are

$$\mathbf{v}_2 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \quad \mathbf{v}_3 = \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix}$$

The three eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ are linearly independent and form a basis for $\mathbb{R}^3$.

The MATLAB/Octave command for finding eigenvalues and eigenvectors is `eig`. The command `eig(A)` lists the eigenvalues

```
>eig(A)

ans =

   4.0000
   2.0000
   6.0000
```

while the variant `[V,D] = eig(A)` returns a matrix `V` whose columns are eigenvectors and a diagonal matrix `D` whose diagonal entries are the eigenvalues.

```
>[V,D] = eig(A)

V =

   5.7735e-01   5.7735e-01  -8.9443e-01
   5.7735e-01  -5.7735e-01   4.4721e-01
  -5.7735e-01   5.7735e-01   2.2043e-16

D =

   4.00000   0.00000   0.00000
   0.00000   2.00000   0.00000
   0.00000   0.00000   6.00000
```

Notice that the eigenvectors have been normalized to have length one. Also, since they have been computed numerically, they are not exactly correct. The entry `2.2043e-16` (i.e., 2.2043×10^{-16}) should actually be zero.

IV.1.4. Example 2

This example has a repeated eigenvalue.

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

The characteristic polynomial is

$$\det(\lambda I - A) = \lambda^3 - 4\lambda^2 + 5\lambda - 2 = (\lambda - 1)^2(\lambda - 2)$$

In this example the eigenvalues are 1 and 2, but the eigenvalue 1 has algebraic multiplicity 2.

Let's find the eigenvector(s) for $\lambda_1 = 1$ we have

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

From this it is easy to see that there are two linearly independent eigenvectors for this eigenvalue:

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{w}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

In this case we say that the geometric multiplicity is 2. In general, the geometric multiplicity is the number of independent eigenvectors, or equivalently the dimension of $N(\lambda I - A)$

The eigenvalue $\lambda_2 = 2$ has eigenvector

$$\mathbf{v}_2 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

So, although this example has repeated eigenvalues, there still is a basis of eigenvectors.

IV.1.5. Example 3

Here is an example where the geometric multiplicity is less than the algebraic multiplicity. If

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

then the characteristic polynomial is

$$\det(\lambda I - A) = (\lambda - 2)^3$$

so there is one eigenvalue $\lambda_1 = 2$ with algebraic multiplicity 3.

To find the eigenvectors we compute

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

From this we see that there is only one independent solution

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Thus the geometric multiplicity $\dim(N(2I - A))$ is 1. What does MATLAB/Octave do in this situation?

```
>A=[2 1 0; 0 2 1; 0 0 2];
>[V D] = eig(A)

V =
```

```
   1.00000  -1.00000   1.00000
   0.00000   0.00000  -0.00000
   0.00000   0.00000   0.00000

D =

   2   0   0
   0   2   0
   0   0   2
```

It simply returned the same eigenvector three times.

In this example, there does not exist a basis of eigenvectors.

IV.1.6. Example 4

Finally, here is an example where the eigenvalues are complex, even though the matrix has real entries. Let

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

Then

$$\det(\lambda I - A) = \lambda^2 + 1$$

which has no real roots. However

$$\lambda^2 + 1 = (\lambda + i)(\lambda - i)$$

so the eigenvalues are $\lambda_1 = i$ and $\lambda_2 = -i$. The eigenvectors are found with the same procedure as before, except that now we must use complex arithmetic. So for $\lambda_1 = i$ we compute

$$iI - A = \begin{bmatrix} i & 1 \\ -1 & i \end{bmatrix}$$

There is trick for computing the null space of a singular 2×2 matrix. Since the two rows must be multiples of each other (in this case the second row is i times the first row) we simply need to find a vector $\begin{bmatrix} a \\ b \end{bmatrix}$ with $ia + b = 0$. This is easily achieved by flipping the entries in the first row and changing the sign of one of them. Thus

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ -i \end{bmatrix}$$

If a matrix has real entries, then the eigenvalues and eigenvectors occur in conjugate pairs. This can be seen directly from the eigenvalue equation $A\mathbf{v} = \lambda\mathbf{v}$. Taking complex conjugates (and using

that the conjugate of a product is the product of the conjugates) we obtain $\bar{A}\bar{\mathbf{v}} = \bar{\lambda}\bar{\mathbf{v}}$ But if A is real then $\bar{A} = A$ so $A\bar{\mathbf{v}} = \bar{\lambda}\bar{\mathbf{v}}$, which shows that $\bar{\lambda}$ and $\bar{\mathbf{v}}$ are also an eigenvalue eigenvector pair.

From this discussion it follows that $\mathbf{v}_2$ is the complex conjugate of $\mathbf{v}_1$

$$\mathbf{v}_2 = \begin{bmatrix} 1 \\ i \end{bmatrix}$$

IV.1.7. A basis of eigenvectors

In three of the four examples above the matrix A had a basis of eigenvectors. If all the eigenvalues are distinct, as in the first example, then the corresponding eigenvectors are always independent and therefore form a basis.

To see why this is true, suppose A has eigenvalues $\lambda_1, \ldots, \lambda_n$ that are all distinct, that is, $\lambda_i \neq \lambda_j$ for $i \neq j$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the corresponding eigenvectors.

Now, starting with the first two eigenvectors, suppose a linear combination of them equals zero:

$$c_1\mathbf{v}_1 + c_2\mathbf{v}_2 = \mathbf{0}$$

Multiplying by A and using the fact that these are eigenvectors, we obtain

$$c_1A\mathbf{v}_1 + c_2A\mathbf{v}_2 = c_1\lambda_1\mathbf{v}_1 + c_2\lambda_2\mathbf{v}_2 = \mathbf{0}$$

On the other hand, multiplying the original equation by λ_2 we obtain

$$c_1\lambda_2\mathbf{v}_1 + c_2\lambda_2\mathbf{v}_2 = \mathbf{0}.$$

Subtracting the equations gives

$$c_1(\lambda_2 - \lambda_1)\mathbf{v}_1 = \mathbf{0}$$

Since $(\lambda_2 - \lambda_1) \neq 0$ and, being an eigenvector, $\mathbf{v}_1 \neq \mathbf{0}$ it must be that $c_1 = 0$. Now returning to the original equation we find $c_2\mathbf{v}_2 = \mathbf{0}$ which implies that $c_2 = 0$ too. Thus $\mathbf{v}_1$ and $\mathbf{v}_2$ are linearly independent.

Now let's consider three eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. Suppose

$$c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + c_3\mathbf{v}_3 = \mathbf{0}$$

As before, we multiply by A to get one equation, then multiply by λ_3 to get another equation. Subtracting the resulting equations gives

$$c_1(\lambda_1 - \lambda_3)\mathbf{v}_1 + c_2(\lambda_2 - \lambda_3)\mathbf{v}_2 = \mathbf{0}$$

But we already know that $\mathbf{v}_1$ and $\mathbf{v}_2$ are independent. Therefore $c_1(\lambda_1 - \lambda_3) = c_2(\lambda_2 - \lambda_3) = 0$. Since $\lambda_1 - \lambda_3 \neq 0$ and $\lambda_2 - \lambda_3 \neq 0$ this implies $c_1 = c_2 = 0$ too. Therefore $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ are independent.

Repeating this argument, we eventually find that all the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are independent.

In example 2 above, we saw that it might be possible to have a basis of eigenvectors even when there are repeated eigenvalues. For some classes of matrices (for example symmetric matrices ($A^T = A$) or orthogonal matrices) a basis of eigenvectors always exists, whether or not there are repeated eigenvalues. Will will consider this in more detail later in the course.

IV.1.8. When there are not enough eigenvectors

Let's try to understand a little better the exceptional situation where there are not enough eigenvectors to form a basis. Consider

$$A_\epsilon = \begin{bmatrix} 1 & 1 \\ 0 & 1+\epsilon \end{bmatrix}$$

when $\epsilon = 0$ this matrix has a single eigenvalues $\lambda = 1$ and only one eigenvector $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. What happens when we change ϵ slightly? Then the eigenvalues change to 1 and $1+\epsilon$, and being distinct, they must have independent eigenvectors. A short calculation reveals that they are

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{v}_2 = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$$

These two eigenvectors are almost, but not quite, dependent. When ϵ becomes zero they collapse and point in the same direction.

In general, if you start with a matrix with repeated eigenvalues and too few eigenvectors, and change the entries of the matrix a little, some of the eigenvectors (the ones corresponding to the eigenvalues whose algebraic multiplicity is higher than the geometric multiplicity) will split into several eigenvectors that are almost parallel.

IV.1.9. Diagonalization

Suppose A is an $n \times n$ matrix with eigenvalues $\lambda_1, \cdots, \lambda_n$ and a basis of eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Form the matrix with eigenvectors as columns

$$S = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}$$

Then

$$\begin{aligned} AS &= \begin{bmatrix} A\mathbf{v}_1 & A\mathbf{v}_2 & \cdots & A\mathbf{v}_n \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \cdots & \lambda_n\mathbf{v}_n \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix} \\ &= SD \end{aligned}$$

where D is the diagonal matrix with the eigenvalues on the diagonal. Since the columns of S are independent, the inverse exists and we can write

$$A = SDS^{-1} \quad S^{-1}AS = D$$

This is called diagonalization.

Notice that the matrix S is exactly the one returns by the MATLAB/Octave call `[S D] = eig(A)`.

```
>A=[1 2 3; 4 5 6; 7 8 9];
>[S D] = eig(A);
>S*D*S^(-1)

ans =

   1.0000   2.0000   3.0000
   4.0000   5.0000   6.0000
   7.0000   8.0000   9.0000
```

IV.1.10. Jordan canonical form

If A is a matrix that cannot be diagonalized, there still exits a similar factorization called the Jordan Canonical Form. It turns out that any matrix A can be written as

$$A = SBS^{-1}$$

where B is a block diagonal matrix. The matrix B has the form

$$B = \begin{bmatrix} B_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & B_2 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & B_3 & \cdots & \mathbf{0} \\ \vdots & & & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & B_k \end{bmatrix}$$

Where each submatrix B_i (called a Jordan block) has a single eigenvalue on the diagonal and 1's on the superdiagonal.

$$B_i = \begin{bmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 \\ 0 & 0 & \lambda_i & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_i \end{bmatrix}$$

If all the blocks are of size 1×1 then there are no 1's and the matrix is diagonalizable.

IV.1.11. Eigenvalues, determinant and trace

Recall that the determinant satisfies $\det(AB) = \det(A)\det(B)$ and $\det(S^{-1}) = 1/\det(S)$. Also, the determinant of a diagonal matrix (or more generally of an upper triangular matrix) is the product of the diagonal entries. Thus if A is diagonalizable then

$$\det(A) = \det(SDS^{-1}) = \det(S)\det(D)\det(S^{-1}) = \det(S)\det(D)/\det(S) = \det(D) = \lambda_1\lambda_2\cdots\lambda_n$$

Thus the determinant of a matrix is the product of the eigenvalues. This is true for non-diagonalizable matrices as well, as can be seen from the Jordan Canonical Form. Notice that the number of times a particular λ_i appears in this product is the algebraic multiplicity of that eigenvalues.

The trace of a matrix is the sum of the diagonal entries. If $A = [a_{i,j}]$ then $\operatorname{tr}(A) = \sum_i a_{i,i}$. Even though it is not true that $AB = BA$ in general, the trace is not sensitive to the change in order:

$$\operatorname{tr}(AB) = \sum_{i,j} a_{i,j}b_{j,i} = \sum_{i,j} b_{j,i}a_{i,j} = \operatorname{tr}(BA)$$

Thus (taking $A = SD$ and $B = S^{-1}$)

$$\operatorname{tr}(A) = \operatorname{tr}(SDS^{-1}) = \operatorname{tr}(S^{-1}SD) = \operatorname{tr}(D) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$$

Thus the trace of a diagonalizable matrix is the sum of the eigenvalues. Again, this is true for non-diagonalizable matrices as well, and can be seen from the Jordan Canonical Form.

IV.1.12. Powers of a diagonalizable matrix

If A is diagonalizable then its powers A^k are easy to compute.

$$A^k = SDS^{-1}SDS^{-1}SDS^{-1}\cdots SDS^{-1} = SD^kS^{-1}$$

because all of the factors $S^{-1}S$ cancel. Since powers of the diagonal matrix D are given by

$$D^k = \begin{bmatrix} \lambda_1^k & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2^k & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3^k & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n^k \end{bmatrix}$$

this formula provides an effective way to understand and compute A^k for large k.

IV.2. Hermitian matrices and real symmetric matrices

Prerequisites and Learning Goals

After completing this section, you should be able to

- Define and compare real symmetric and Hermitian matrices.
- Show that the eigenvalues of Hermitian matrices are real and that eigenvectors corresponding to distinct eigenvalues are orthogonal.
- Explain why a Hermitian matrix can be diagonalized with a unitary matrix.
- Calculate the effective resistance between any two points in a resistor network using the eigenvalues and eigenfunctions of the Laplacian matrix.

IV.2.1. Hermitian and real symmetric matrices

An $n \times n$ matrix A is called *Hermitian* if $A^* = A$. (Recall that $A^* = \overline{A}^T$.) If $A = [a_{i,j}]$ then this is the same as saying $a_{j,i} = \overline{a_{i,j}}$. An equivalent condition is

$$\langle \mathbf{v}, A\mathbf{w} \rangle = \langle A\mathbf{v}, \mathbf{w} \rangle$$

for all vectors $\mathbf{v}$ and $\mathbf{w}$. Here are some examples of Hermitian matrices:

$$\begin{bmatrix} 2 & i \\ -i & 1 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1+i & 1 \\ 1-i & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

An $n \times n$ matrix A is called *real symmetric* if it is Hermitian and all its entries are real. In other words, A is real symmetric if has real entries and $A^T = A$.

IV.2.2. Eigenvalues of Hermitian matrices are real

A general $n \times n$ matrix may have complex eigenvalues, even if all its entries are real. However, the eigenvalues of a Hermitian (possibly real symmetric) matrix are all real. To see this, let's start with an eigenvalue eigenvector pair λ, $\mathbf{v}$ for a Hermitian matrix A. For the moment, we allow the possibility that λ and $\mathbf{v}$ are complex. Since A is Hermitian, we have

$$\langle \mathbf{v}, A\mathbf{v} \rangle = \langle A\mathbf{v}, \mathbf{v} \rangle$$

and since $A\mathbf{v} = \lambda\mathbf{v}$ this implies

$$\langle \mathbf{v}, \lambda\mathbf{v} \rangle = \langle \lambda\mathbf{v}, \mathbf{v} \rangle$$

Here we are using the inner product for complex vectors given by $\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^* \mathbf{w} = \overline{\mathbf{v}}^T \mathbf{w}$. Thus $\langle \lambda \mathbf{v}, \mathbf{v} \rangle = \overline{\lambda} \langle \mathbf{v}, \mathbf{v} \rangle$, so that

$$\lambda \langle \mathbf{v}, \mathbf{v} \rangle = \overline{\lambda} \langle \mathbf{v}, \mathbf{v} \rangle.$$

Since $\mathbf{v}$ is an eigenvector, it cannot be zero. So $\langle \mathbf{v}, \mathbf{v} \rangle = \|\mathbf{v}\|^2 \neq 0$. Therefore we may divide by $\langle \mathbf{v}, \mathbf{v} \rangle$ to conclude

$$\lambda = \overline{\lambda}.$$

This shows that λ is real.

IV.2.3. Eigenvectors of Hermitian matrices corresponding to distinct eigenvalues are orthogonal

Suppose $A\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$ and $A\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$ with $\lambda_1 \neq \lambda_2$. Since A is Hermitian,

$$\langle A\mathbf{v}_1, \mathbf{v}_2 \rangle = \langle \mathbf{v}_1, A\mathbf{v}_2 \rangle.$$

Thus we find that

$$\lambda_1 \langle \mathbf{v}_1, \mathbf{v}_2 \rangle = \lambda_2 \langle \mathbf{v}_1, \mathbf{v}_2 \rangle$$

Here λ_1 should appear as $\overline{\lambda}_1$, but we already know that eigenvalues are real so $\overline{\lambda}_1 = \lambda_1$. This can be written

$$(\lambda_1 - \lambda_2) \langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$$

and since $\lambda_1 - \lambda_2 \neq 0$ this implies

$$\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$$

This calculation shows that if a Hermitian matrix A has distinct eigenvalues then the eigenvectors are all orthogonal, and by rescaling them, we can obtain an orthonormal basis of eigenvectors. If we put these orthonormal eigenvectors in the columns of a matrix U then U is unitary and $U^* A U = D$ where D is the diagonal matrix with the eigenvalues of A as diagonal entries.

In fact, even if A has repeated eigenvalues, it is still true that an orthonormal basis of eigenvectors exists, as we will show below.

If A is real symmetric, then the eigenvectors can be chosen to be real. This implies that the matrix that diagonalizes A can be chosen to be an orthogonal matrix. One way to see this is to notice that if once we know that λ is real then all the calculations involved in computing the nullspace of $\lambda I - A$ only involve real numbers. Another way to see this is to start with any eigenvector $\mathbf{v}$ (possibly complex) for A. We can write in terms of its real and imaginary parts as $\mathbf{v} = \mathbf{x} + i\mathbf{y}$ where $\mathbf{x}$ and $\mathbf{y}$ are real vectors. Then $A\mathbf{v} = \lambda \mathbf{v}$ implies $A\mathbf{x} + iA\mathbf{y} = \lambda \mathbf{x} + i\lambda \mathbf{y}$. Now we equate real and imaginary parts. Using that A and λ are real, this yields $A\mathbf{x} = \lambda \mathbf{x}$ and $A\mathbf{y} = \lambda \mathbf{y}$. Thus $\mathbf{x}$ and $\mathbf{y}$ are both eigenvectors with real entries. This might seem too good to be true - now we have two eigenvectors instead of one. But in most cases, $\mathbf{x}$ and $\mathbf{y}$ will be multiplies of each other, so that they are the same up to scaling.

If A is any square matrix with real entries, then $A + A^T$ is real symmetric. (The matrix $A^T A$ is also real symmetric, and has the additional property that all the eigenvalues are positive.) We can use this to produce random symmetric matrices in MATLAB/Octave like this:

```
>A = rand(4,4);
>A = A+A'

A =

   0.043236   1.240654   0.658890   0.437168
   1.240654   1.060615   0.608234   0.911889
   0.658890   0.608234   1.081767   0.706712
   0.437168   0.911889   0.706712   1.045293
```

Let's check the eigenvalues and vectors or A

```
>[V, D] = eig(A)

V =

  -0.81345   0.33753   0.23973   0.40854
   0.54456   0.19491   0.55585   0.59707
   0.15526   0.35913  -0.78824   0.47497
  -0.13285  -0.84800  -0.11064   0.50100

D =

  -0.84168   0.00000   0.00000   0.00000
   0.00000   0.36240   0.00000   0.00000
   0.00000   0.00000   0.55166   0.00000
   0.00000   0.00000   0.00000   3.15854
```

The eigenvalues are real, as expected. Also, the eigenvectors contained in the columns of the matrix V have been normalized. Thus V is orthogonal:

```
>V'*V

ans =

   1.0000e+00   6.5066e-17  -1.4700e-16   1.4366e-16
   9.1791e-17   1.0000e+00  -1.0432e-16   2.2036e-17
  -1.4700e-16  -1.0012e-16   1.0000e+00  -1.2617e-16
   1.3204e-16   2.2036e-17  -1.0330e-16   1.0000e+00
```

(at least, up to numerical error.)

IV.2.4. Every Hermitian matrix can be diagonalized by a unitary matrix

Now we will show that *every* Hermitian matrix A (even if it has repeated eigenvalues) can be diagonalized by a unitary matrix. Equivalently, every Hermitian matrix has an orthonormal basis. If A is real symmetric, then the basis of eigenvectors can be chosen to be real. Therefore this will show that every real symmetric matrix can be diagonalized by an orthogonal matrix.

The argument we present works whether or not A has repeated eigenvalues and also gives a new proof of the fact that the eigenvalues are real.

To begin we show that if A is any $n \times n$ matrix (not neccesarily Hermitian), then there exists a unitary matrix U such that U^*AU is upper triangular. To see this start with any eigenvector $\mathbf{v}$ of A with eigenvalue λ. (Every matrix has at least one eigenvalue.) Normalize $\mathbf{v}$ so that $\|\mathbf{v}\| = 1$. Now choose and orthonormal basis $\mathbf{q}_2, \ldots, \mathbf{q}_n$ for the orthogonal complement of the subspace spanned by $\mathbf{v}$, so that $\mathbf{v}, \mathbf{q}_2, \ldots, \mathbf{q}_n$ form an orthonormal basis for all of $\mathbb{C}^n$. Form the matrix U_1 with these vectors as columns. Then, using $*$ to denotes a number that is not neccesarily 0, we have

$$
\begin{aligned}
U_1^* A U_1 &= \left[\begin{array}{c} \overline{\mathbf{v}}^T \\ \hline \overline{\mathbf{q}}_2^T \\ \hline \vdots \\ \hline \overline{\mathbf{q}}_n^T \end{array}\right] A \left[\begin{array}{c|c|c|c} \mathbf{v} & \mathbf{q}_2 & \ldots & \mathbf{q}_n \end{array}\right] \\
&= \left[\begin{array}{c} \overline{\mathbf{v}}^T \\ \hline \overline{\mathbf{q}}_2^T \\ \hline \vdots \\ \hline \overline{\mathbf{q}}_n^T \end{array}\right] \left[\begin{array}{c|c|c|c} \lambda\mathbf{v} & A\mathbf{q}_2 & \ldots & A\mathbf{q}_n \end{array}\right] \\
&= \begin{bmatrix} \lambda\|\mathbf{v}\|^2 & * & \ldots & * \\ \lambda\langle \mathbf{q}_2, \mathbf{v}\rangle & * & \ldots & * \\ \vdots & \vdots & \vdots & \vdots \\ \lambda\langle \mathbf{q}_n, \mathbf{v}\rangle & * & \ldots & * \end{bmatrix} \\
&= \begin{bmatrix} \lambda & * & \ldots & * \\ 0 & * & \ldots & * \\ \vdots & \vdots & \vdots & \vdots \\ 0 & * & \ldots & * \end{bmatrix} \\
&= \begin{bmatrix} \lambda & * & \ldots & * \\ 0 & & & \\ \vdots & & A_2 & \\ 0 & & & \end{bmatrix}.
\end{aligned}
$$

Here A_2 is an $(n-1) \times (n-1)$ matrix.

Repeating the same procedure with A_2 we can construct an $(n-1)\times(n-1)$ unitary matrix $\tilde{U}_2$ with

$$\tilde{U}_2^* A_2 \tilde{U}_2 = \begin{bmatrix} \lambda_2 & * & \dots & * \\ 0 & & & \\ \vdots & & A_3 & \\ 0 & & & \end{bmatrix}.$$

Now we use the$(n-1)\times(n-1)$ unitary matrix $\tilde{U}_2$ to construct an $n\times n$ unitary matrix

$$U_2 = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \tilde{U}_2 & \\ 0 & & & \end{bmatrix}$$

Then it is not hard to see that

$$U_2^* U_1^* A U_1 U_2 = \begin{bmatrix} \lambda & * & * & * & * \\ 0 & \lambda_2 & * & \dots & * \\ 0 & 0 & & & \\ \vdots & \vdots & & A_3 & \\ 0 & 0 & & & \end{bmatrix}.$$

Continuing in this way, we find unitary matrices $U_3, U_4, \dots, U_{n-1}$ so that

$$U_{n-1}^* \cdots U_2^* U_1^* A U_1 U_2 \cdots U_{n-1} = \begin{bmatrix} \lambda & * & * & * & * \\ 0 & \lambda_2 & * & \dots & * \\ 0 & 0 & & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & & \lambda_n \end{bmatrix}.$$

Define $U = U_1 U_2 \cdots U_{n-1}$. Then U is unitary, since the product of unitary matrices is unitary, and $U^* = U_{n-1}^* \cdots U_2^* U_1^*$. Thus the equation above says that $U^* A U$ is upper triangular, that is,

$$U^* A U = \begin{bmatrix} \lambda & * & * & * & * \\ 0 & \lambda_2 & * & \dots & * \\ 0 & 0 & & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & & \lambda_n \end{bmatrix}$$

Notice that if we take the adjoint of this equation, we get

$$U^*A^*U = \begin{bmatrix} \overline{\lambda} & 0 & 0 & 0 & 0 \\ * & \overline{\lambda}_2 & 0 & \dots & 0 \\ * & * & & & \\ \vdots & \vdots & & \ddots & \\ * & * & & & \overline{\lambda}_n \end{bmatrix}$$

Now lets return to the case where A is Hermitian. Then $A^* = A$ so that the matrices appearing in the previous two equations are equal. Thus

$$\begin{bmatrix} \lambda & * & * & * & * \\ 0 & \lambda_2 & * & \dots & * \\ 0 & 0 & & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & & \lambda_n \end{bmatrix} = \begin{bmatrix} \overline{\lambda} & 0 & 0 & 0 & 0 \\ * & \overline{\lambda}_2 & 0 & \dots & 0 \\ * & * & & & \\ \vdots & \vdots & & \ddots & \\ * & * & & & \overline{\lambda}_n \end{bmatrix}$$

This implies that all the entries denoted $*$ must actually be zero. This also shows that $\lambda_i = \overline{\lambda}_i$ for every i. In other words, Hermitian matrices can be diagonalized by a unitary matrix, and all the eigenvalues are real.

IV.2.5. Powers and other functions of Hermitian matrices

The diagonalization formula for a Hermitian matrix A with eigenvalues $\lambda_1, \dots, \lambda_n$ and orthonormal basis of eigenvectors $\mathbf{v}_1, \dots \mathbf{v}_n$ can be written

$$A = UDU^* = \left[\begin{array}{c|c|c|c} & & & \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \\ & & & \end{array}\right] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \left[\begin{array}{c} \overline{\mathbf{v}}_1^T \\ \hline \overline{\mathbf{v}}_2^T \\ \hline \vdots \\ \hline \overline{\mathbf{v}}_n^T \end{array}\right]$$

This means that for any vector $\mathbf{w}$

$$A\mathbf{w} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \overline{\mathbf{v}}_i^T \mathbf{w}.$$

Since this is true for any vector $\mathbf{w}$ it must be that

$$A = \sum_{i=1}^{n} \lambda_i P_i$$

where $P_i = \mathbf{v}_i \overline{\mathbf{v}}_i^T$. The matrices P_i are the orthogonal projections onto the one dimensional subspaces spanned by the eigenvectors. We have $P_i P_i = P_i$ and $P_i P_j = 0$ for $i \neq j$.

Recall that to compute powers A^k we simply need to replace λ_i with λ_i^k in the diagonalization formula. Thus

$$A^k = \sum_{i=1}^{n} \lambda_i^k P_i.$$

This formula is valid for negative powers k provided none of the λ_i are zero. In particular, when $k = -1$ we obtain a formula for the inverse.

In fact we can define $f(A)$ for any function f to be the matrix

$$f(A) = \sum_{i=1}^{n} f(\lambda_i) P_i.$$

IV.2.6. Effective resistance revisited

In this section we will use its eigenvalues and eigenvectors of the Laplacian matrix to express the effective resistance between any two nodes of a resistor network. The Laplacian L is a real symmetric matrix.

The basic equation for a resistor network is

$$L\mathbf{v} = \mathbf{J}$$

where the entries of $\mathbf{v} = [v_1, v_2, \ldots, v_n]^T$ are the voltages the nodes, and the entries of $\mathbf{J} = [J_1, J_2, \ldots, J_n]$ are the currents that are flowing in or out of each node. If we connect a b volt battery to nodes i and j we are fixing the voltage difference between these nodes to be the battery voltage so that

$$v_i - v_j = b.$$

In this situation the there are no currents flowing in or out of the network except at nodes i and j, so except for these two nodes $\mathbf{J}$ has all entries equal to zero. Since the current flowing in must equal the current flowing out we have $J_i = c$, $J_j = -c$ for some value of c. This can be written

$$\mathbf{J} = c(\mathbf{e}_i - \mathbf{e}_j)$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ are the standard basis vectors. The effective resistance between the ith and jth nodes is

$$R = b/c.$$

This is the quantity that we wish to compute.

Let λ_i and $\mathbf{u}_i$, $i - 1 \ldots, n$ be the eigenvalues and eigenvectors for L. Since L is real symmetric, we know that the eigenvectors λ_i are real and the eigenvectors $\mathbf{u}_i$ can be chosen to be orthonormal. However for the Laplacian we know more than this. We will assume that all the nodes in the network are connected. Then L has a one dimensional nullspace spanned by the vector with constant entries. Thus $\lambda_1 = 0$ and $\mathbf{u}_1 = (1/\sqrt{n})[1, 1, \ldots, 1]^T$. Furthermore all the eigenvalues of L are non-negative

(see homework problem). Since only one of them is zero, all the other eigenvalues $\lambda_2, \ldots, \lambda_n$ must be strictly positive.

Write $L = \sum_{i=1}^n \lambda_i P_i$ where $P_i = \mathbf{u}_i \mathbf{u}_i^T$ is the projection onto the subspace spanned by the eigenvector $\mathbf{u}_i$. Since $\lambda_1 = 0$, the matrix L is not invertible. However we can define the partial inverse $G = \sum_{i=2}^n \lambda_i^{-1} P_i$. Then

$$GL = \left(\sum_{i=2}^n \lambda_i^{-1} P_i \right) \left(\sum_{j=1}^n \lambda_j P_j \right) = \sum_{i=2}^n P_i = P$$

where P is the projection onto the orthogonal complement of the nullpace.

Since $\mathbf{v}$ and $P\mathbf{v}$ differ by a multiple of the vector $\mathbf{u}_1$ with constant entries, we have

$$\begin{aligned} b = v_i - v_j &= (P\mathbf{v})_i - (P\mathbf{v})_j \\ &= \langle \mathbf{e}_i - \mathbf{e}_j, P\mathbf{v} \rangle = \langle \mathbf{e}_i - \mathbf{e}_j, GL\mathbf{v} \rangle \\ &= \langle \mathbf{e}_i - \mathbf{e}_j, G\mathbf{J} \rangle = c \langle \mathbf{e}_i - \mathbf{e}_j, G(\mathbf{e}_i - \mathbf{e}_j) \rangle \\ &= c \sum_{k=2}^n \langle \mathbf{e}_i - \mathbf{e}_j, \lambda_k^{-1} P_k (\mathbf{e}_i - \mathbf{e}_j) \rangle = c \sum_{k=2}^n \lambda_k^{-1} \left| \langle \mathbf{e}_i - \mathbf{e}_j, \mathbf{u}_k \rangle \right|^2 . \end{aligned}$$

Therefore the effective resistance is

$$R = b/c = \sum_{k=2}^n \lambda_k^{-1} \left| \langle \mathbf{e}_i - \mathbf{e}_j, \mathbf{u}_k \rangle \right|^2 = \sum_{k=2}^n \lambda_k^{-1} \left| (\mathbf{u}_k) i - (\mathbf{u}_k)_j \right|^2$$

Let's redo the calculation of the effective resistance in section II.3.10 using this formula. We first define L and compute the eigenvalues and eigenvectors.

```
> L=[2 -1 0 -1;-1 3 -1 -1;0 -1 2 -1;-1 -1 -1 3];
> [U,D]=eig(L);
```

Somewhat confusingly $(\mathbf{u}_k)_i = U(i,k)$, and $\lambda_k = D(k,k)$. Thus the effective resistance is obtained by summing the terms `D(k,k)^(-1)*(U(1,k)-U(2,k))^2`.

```
> R=0;
> for k=[2:4] R=R+D(k,k)^(-1)*(U(1,k)-U(2,k))^2 end
R =  0.25000
R =  0.25000
R =  0.62500
```

This gives the same answer $5/8 = 0.62500$ as before.

Although the formula in this section can be used for calculations, the real advantage lies in the connection it exposes between the effective resistance and the eigenvalues of L. In many applications, one is interested in networks with low effective resistance between any two points. Such networks are highly connected - a desirable property when designing, say, communications networks. The formula in this section shows that the effective resistance will be low if the non-zero eigenvalues $\lambda_2, \lambda_3, \ldots$ are large.

IV.3. Power Method for Computing Eigenvalues

Prerequisites and Learning Goals

After completing this section, you should be able to

- Explain how to implement the power method in order to compute certain eigenvalues/eigenvectors, write out the steps of the algorithm and discuss its output; implement the algorithm in MATLAB/Octave and use it to compute the largest and smallest eigenvalues and the eigenvalue closest to a given number (and corresponding eigenvectors) of a Hermitian matrix; discuss what it means when the algorithm does not convergence.

IV.3.1. The power method

The power method is a very simple method for finding a single eigenvalue–eigenvector pair.

Suppose A is an $n \times n$ matrix. We assume that A is real symmetric, so that all the eigenvalues are real. Now let $\mathbf{x}_0$ be any vector of length n. Perform the following steps:

- Multiply by A
- Normalize to unit length.

repeatedly. This generates a series of vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$. It turns out that these vectors converge to the eigenvector corresponding to the eigenvalue whose absolute value is the largest.

To verify this claim, let's first find a formula for $\mathbf{x}_k$. At each stage of this process, we are multiplying by A and then by some number. Thus $\mathbf{x}_k$ must be a multiple of $A^k\mathbf{x}_0$. Since the resulting vector has unit length, that number must be $1/\|A^k\mathbf{x}_0\|$. Thus

$$\mathbf{x}_k = \frac{A^k\mathbf{x}_0}{\|A^k\mathbf{x}_0\|}$$

We know that A has a basis of eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. Order them so that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. (We are assuming here that $|\lambda_1| \neq |\lambda_2|$, otherwise the power method runs into difficulty.) We may expand our initial vector $\mathbf{x}_0$ in this basis

$$\mathbf{x}_0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots c_n\mathbf{v}_n$$

We need that $c_1 \neq 0$ for this method to work, but if $\mathbf{x}_0$ is chosen at random, this is almost always true.

Since $A^k\mathbf{v}_i = \lambda_i^k\mathbf{v}_i$ we have

$$\begin{aligned} A^k\mathbf{x}_0 &= c_1\lambda_1^k\mathbf{v}_1 + c_2\lambda_2^k\mathbf{v}_2 + \cdots c_n\lambda_n^k\mathbf{v}_n \\ &= \lambda_1^k\Big(c_1\mathbf{v}_1 + c_2(\lambda_2/\lambda_1)^k\mathbf{v}_2 + \cdots c_n(\lambda_n/\lambda_1)^k\mathbf{v}_n\Big) \\ &= \lambda_1^k(c_1\mathbf{v}_1 + \epsilon_k) \end{aligned}$$

where $\epsilon_k \to 0$ as $k \to \infty$. This is because $|(\lambda_i/\lambda_1)| < 1$ for every $i > 1$ so the powers tend to zero. Thus

$$\|A^k \mathbf{x}_0\| = |\lambda_1|^k \|c_1 \mathbf{v}_1 + \epsilon_k\|$$

so that

$$\begin{aligned} \mathbf{x}_k &= \frac{A^k \mathbf{x}_0}{\|A^k \mathbf{x}_0\|} \\ &= \left(\frac{\lambda_1}{|\lambda_1|}\right)^k \frac{c_1 \mathbf{v}_1 + \epsilon_k}{\|c_1 \mathbf{v}_1 + \epsilon_k\|} \\ &\to (\pm)^k \left(\pm \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}\right) \end{aligned}$$

We have shown that $\mathbf{x}_k$ converges, except for a possible sign flip at each stage, to a normalized eigenvector corresponding to λ_1. The sign flip is present exactly when $\lambda_1 < 0$. Knowing $\mathbf{v}_1$ (or a multiple of it) we can find λ_1 with

$$\lambda_1 = \frac{\langle \mathbf{v}_1, A\mathbf{v}_1 \rangle}{\|\mathbf{v}_1\|^2}$$

This gives a method for finding the largest eigenvalue (in absolute value) and the corresponding eigenvector. Let's try it out.

```
>A = [4 1 3;1 3 2; 3 2 5];
>x=rand(3,1);
>for k = [1:10]
>y=A*x;
>x=y/norm(y)
>end

x =

   0.63023
   0.38681
   0.67319

x =

   0.58690
   0.37366
   0.71828

x =

   0.57923
```

```
   0.37353
   0.72455

x =

   0.57776
   0.37403
   0.72546

x =

   0.57745
   0.37425
   0.72559

x =

   0.57738
   0.37433
   0.72561

x =

   0.57736
   0.37435
   0.72562

x =

   0.57735
   0.37436
   0.72562

x =

   0.57735
   0.37436
   0.72562

x =

   0.57735
   0.37436
   0.72562
```

This gives the eigenvector. We compute the eigenvalue with

```
>x'*A*x/norm(x)^2

ans =  8.4188
```

Let's check:

```
>[V D] = eig(A)

V =

   0.577350   0.577350   0.577350
   0.441225  -0.815583   0.374359
  -0.687013  -0.038605   0.725619

D =

   1.19440   0.00000   0.00000
   0.00000   2.38677   0.00000
   0.00000   0.00000   8.41883
```

As expected, we have computed the largest eigenvalue and eigenvector. Of course, a serious program that uses this method would not just iterate a fixed number (above it was 10) times, but check for convergence, perhaps by checking whether $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|$ was less than some small number, and stopping when this was achieved.

So far, the power method only computes the eigenvalue with the largest absolute value, and the corresponding eigenvector. What good is that? Well, it turns out that with an additional twist we can compute the eigenvalue closest to any number s. The key observation is that the eigenvalues of $(A - sI)^{-1}$ are exactly $(\lambda_i - s)^{-1}$ (unless, of course, $A - sI$ is not invertible. But then s is an eigenvalue itself and we can stop looking.) Moreover, the eigenvectors of A and $(A - sI)^{-1}$ are the same.

Let's see why this is true. If

$$A\mathbf{v} = \lambda\mathbf{v}$$

then

$$(A - sI)\mathbf{v} = (\lambda - s)\mathbf{v}.$$

Now if we multiply both sides by $(A - sI)^{-1}$ and divide by $\lambda - s$ we get

$$(\lambda - s)^{-1}\mathbf{v} = (A - sI)^{-1}\mathbf{v}.$$

These steps can be run backwards to show that if $(\lambda - s)^{-1}$ is an eigenvalue of $(A - sI)^{-1}$ with eigenvector $\mathbf{v}$, then λ is an eigenvalue of A with the same eigenvector.

Now start with an arbitrary vector $\mathbf{x}_0$ and define

$$\mathbf{x}_{k+1} = \frac{(A - sI)^{-1}\mathbf{x}_k}{\|(A - sI)^{-1}\mathbf{x}_k\|}.$$

Then $\mathbf{x}_k$ will converge to the eigenvector $\mathbf{v}_i$ of $(A - sI)^{-1}$ for which $|\lambda_i - s|^{-1}$ is the largest. But, since the eigenvectors of A and $A - sI$ are the same, $\mathbf{v}_i$ is also an eigenvector of A. And since $|\lambda_i - s|^{-1}$ is largest when λ_i is closest to s, we have computed the eigenvector $\mathbf{v}_i$ of A for which λ_i is closest to s. We can now compute λ_i by comparing $A\mathbf{v}_i$ with $\mathbf{v}_i$

Here is a crucial point: when computing $(A - sI)^{-1}\mathbf{x}_k$ in this procedure, we should not actually compute the inverse. We don't need to know the whole matrix $(A - sI)^{-1}$, but just the vector $(A - sI)^{-1}\mathbf{x}_k$. This vector is the solution $\mathbf{y}$ of the linear equation $(A - sI)\mathbf{y} = x_k$. In MATLAB/Octave we would therefore use something like `(A - s*eye(n))\Xk`.

Let's try to compute the eigenvalue of the matrix `A` above closest to 3.

```
>A = [4 1 3;1 3 2; 3 2 5];
>x=rand(3,1);
>for k = [1:10]
>y=(A-3*eye(3))\x;
>x=y/norm(y)
>end

x =

   0.649008
  -0.756516
   0.080449

x =

  -0.564508
   0.824051
   0.047657

x =

   0.577502
  -0.815593
  -0.036045
```

```
x =

  -0.576895
   0.815917
   0.038374

x =

   0.577253
  -0.815659
  -0.038454

x =

  -0.577311
   0.815613
   0.038562

x =

   0.577338
  -0.815593
  -0.038590

x =

  -0.577346
   0.815587
   0.038600

x =

   0.577349
  -0.815585
  -0.038603

x =

  -0.577350
   0.815584
   0.038604
```

This gives the eigenvector. Now we can find the eigenvalue

```
> lambda = x'*A*x/norm(x)^2

lambda = 2.3868
```

Comparing with the results of `eig` above, we see that we have computed the middle eigenvalue and eigenvector.

IV.4. Recursion Relations

Prerequisites and Learning Goals

After completing this section, you should be able to

- Derive a matrix equation from a recursion relation and use it to find the n-th term in the sequence given some initial conditions. You should be able to perform these computations by hand or with MATLAB/Octave.
- Discuss how the relative magnitude of the eigenvalues of the matrix obtained from a recursion relation determines the behaviour of the high power terms of the recursion.
- Determine initial values for which the solution of a recursion relation will become large or small (depending on the eigenvalues of the associated matrix).

IV.4.1. Fibonacci numbers

Consider the sequence of numbers given by a multiple of powers of the golden ratio

$$\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \quad n = 1, 2, 3 \ldots .$$

When n is large, these numbers are *almost* integers:

```
>format long;
>((1+sqrt(5))/2)^30/sqrt(5)

ans =  832040.000000241

>((1+sqrt(5))/2)^31/sqrt(5)

ans =  1346268.99999985

>((1+sqrt(5))/2)^32/sqrt(5)

ans =  2178309.00000009
```

Why? To answer this question we introduce the Fibonacci sequence:

$$0 \quad 1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21 \quad \ldots$$

where each number in the sequence is obtained by adding the previous two. If you go far enough along in this sequence you will encounter

$$\ldots \quad 832040 \quad 1346269 \quad 2178309 \quad \ldots$$

and you can check (without using MATLAB/Octave, I hope) that the third number is the sum of the previous two.

But why should powers of the golden ratio be very nearly, but not quite, equal to Fibonacci numbers? The reason is that the Fibonacci sequence is defined by a recursion relation. For the Fibonacci sequence $F_0, F_1, F_2, \ldots$ the recursion relation is

$$F_{n+1} = F_n + F_{n-1}$$

This equation, together with the identity $F_n = F_n$ can be written in matrix form as

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

Thus, taking $n = 1$, we find

$$\begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

Similarly

$$\begin{bmatrix} F_3 \\ F_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

and continuing like this we find

$$\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

Finally, since $F_0 = 0$ and $F_1 = 1$ we can write

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

We can diagonalize the matrix to get a formula for the Fibonacci numbers. The eigenvalues and eigenvectors of $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ are

$$\lambda_1 = \frac{1+\sqrt{5}}{2} \quad \mathbf{v}_1 = \begin{bmatrix} \frac{1+\sqrt{5}}{2} \\ 1 \end{bmatrix}$$

and

$$\lambda_2 = \frac{1-\sqrt{5}}{2} \quad \mathbf{v}_2 = \begin{bmatrix} \frac{1-\sqrt{5}}{2} \\ 1 \end{bmatrix}$$

This implies

$$\begin{bmatrix}\lambda_1 & \lambda_2\\ 1 & 1\end{bmatrix}\begin{bmatrix}\lambda_1^n & 0\\ 0 & \lambda_2^n\end{bmatrix}\begin{bmatrix}\lambda_1 & \lambda_2\\ 1 & 1\end{bmatrix}^{-1}=\frac{1}{\sqrt{5}}\begin{bmatrix}\lambda_1 & \lambda_2\\ 1 & 1\end{bmatrix}\begin{bmatrix}\lambda_1^n & 0\\ 0 & \lambda_2^n\end{bmatrix}\begin{bmatrix}1 & -\lambda_2\\ -1 & \lambda_1\end{bmatrix}$$

so that

$$\begin{bmatrix}F_{n+1}\\ F_n\end{bmatrix}=\frac{1}{\sqrt{5}}\begin{bmatrix}\lambda_1 & \lambda_2\\ 1 & 1\end{bmatrix}\begin{bmatrix}\lambda_1^n & 0\\ 0 & \lambda_2^n\end{bmatrix}\begin{bmatrix}1 & -\lambda_2\\ -1 & \lambda_1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix}=\frac{1}{\sqrt{5}}\begin{bmatrix}\lambda_1^{n+1}-\lambda_2^{n+1}\\ \lambda_1^n-\lambda_2^n\end{bmatrix}$$

In particular

$$F_n=\frac{1}{\sqrt{5}}\left(\lambda_1^n-\lambda_2^n\right)$$

Since $\lambda_2\sim -0.6180339880$ is smaller than 1 in absolute value, the powers λ_2^n become small very quickly as n becomes large. This explains why

$$F_n\sim\frac{1}{\sqrt{5}}\lambda_1^n$$

for large n.

If we want to use MATLAB/Octave to compute Fibonacci numbers, we don't need to bother diagonalizing the matrix.

```
>[1 1;1 0]^30*[1;0]

ans =

   1346269
    832040
```

produces the same Fibonacci numbers as above.

IV.4.2. Other recursion relations

The idea that was used to solve for the Fibonacci numbers can be used to solve other recursion relations. For example the three-step recursion

$$x_{n+1}=ax_n+bx_{n-1}+cx_{n-2}$$

can be written as a matrix equation

$$\begin{bmatrix}x_{n+1}\\ x_n\\ x_{n-1}\end{bmatrix}=\begin{bmatrix}a & b & c\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_n\\ x_{n-1}\\ x_{n-2}\end{bmatrix}$$

so given three initial values x_0, x_1 and x_2 we can find the rest by computing powers of a 3×3 matrix.

In the next section we will solve a recurrence relation that arises in Quantum Mechanics.

IV.5. The Tight Binding Model for an electron in one dimension

Prerequisites and Learning Goals

After completing this section, you should be able to

- describe a bound state with energy E for the discrete Schrodinger equation for a single electron moving in a one dimensional semi-infinite crystal.
- describe a scattering state with energy E.
- compute the energies for which a bound state exists and identify the conduction band, for a potential that has only one non-zero value.
- compute the conduction bands for a one dimensional crystal.

IV.5.1. Description of the model

Previously we studied how to approximate differential equations by matrix equations. If we apply this discretization procedure to the Schrödinger equation for an electron moving in a solid we obtain the tight binding model.

We will consider a single electron moving in a one dimensional semi-infinite crystal. The electron is constrained to live at discrete lattice points, numbered $0, 1, 2, 3, \ldots$. These can be thought of as the positions of the atoms. For each lattice point n there is a potential V_n that describes how much the atom at that lattice point attracts or repels the electron. Positive V_n's indicate repulsion, whereas negative V_n's indicate attraction. Typical situations studied in physics are where the V_n's repeat the same pattern periodically (a crystal), or where they are chosen at random (disordered media). The random case, where the potentials are chosen independently for each site, is known as the Anderson model.

The wave function for the electron is a sequence of complex numbers $\Psi = \{\psi_0, \psi_1, \psi_2, \ldots\}$. The sequence Ψ is called a *bound state* with energy E if satisfies the following three conditions:

(1) The discrete version of the time independent Schrödinger equation

$$-\psi_{n+1} - \psi_{n-1} + V_n\psi_n = E\psi_n,$$

(2) the boundary condition

$$\psi_0 = 0,$$

(3) and the normalization condition

$$N^2 = \sum_{n=0}^{\infty} |\psi_n|^2 < \infty.$$

This conditions are trivially satisfies if $\Psi = \{0, 0, 0, \ldots\}$ so we specifically exclude this case. (In fact Ψ is actually the eigenvector of an infinite matrix so this is just the condition that eigenvectors must be non-zero.)

Given an energy E, it is always possible to find a wave function Ψ to satisfy conditions (1) and (2). However for most energies E, none of these Ψ's will be getting small for large n, so the normalization condition (3) will not be satisfied. There are only a discrete set of energies E for which a bound state satisfying all three conditions is satisfied. In other words, the energy E is quantized.

If E is one of the allowed energy values and Ψ is the corresponding bound state, then the numbers $|\psi_n|^2/N^2$ are interpreted as the probabilities of finding an electron with energy E at the nth site. These numbers add up to 1, consistent with the interpretation as probabilities.

Notice that if Ψ is a bound state with energy E, then so is any non-zero multiple $a\Psi = \{a\psi_0, a\psi_1, a\psi_2, \ldots\}$. Replacing Ψ with $a\Psi$ has no effect on the probabilities because N changes to aN, so the a's cancel in $|\psi_n|^2/N^2$.

IV.5.2. Recursion relation

The discrete Schrödinger equation (1) together with the initial condition (2) is a recursion relation that can be solved using the method we saw in the previous section. We have

$$\begin{bmatrix} \psi_{n+1} \\ \psi_n \end{bmatrix} = \begin{bmatrix} V_n - E & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \psi_n \\ \psi_{n-1} \end{bmatrix}$$

so if we set

$$\mathbf{x}_n = \begin{bmatrix} \psi_{n+1} \\ \psi_n \end{bmatrix}$$

and

$$A(z) = \begin{bmatrix} z & -1 \\ 1 & 0 \end{bmatrix}$$

then this implies

$$\mathbf{x}_n = A(V_n - E)A(V_{n-1} - E) \cdots A(V_1 - E)\mathbf{x}_0.$$

Condition (2) says that

$$\mathbf{x}_0 = \begin{bmatrix} \psi_1 \\ 0 \end{bmatrix},$$

since $\psi_0 = 0$.

In fact, we may assume $\psi_1 = 1$, since replacing Ψ with $a\Psi$ where $a = 1/\psi_1$ results in a bound state where this is true. Dividing by ψ_1 is possible unless $\psi_1 = 0$. But if $\psi_1 = 0$ then $\mathbf{x}_0 = \mathbf{0}$ and the recursion implies that every $\psi_k = 0$. This is not an acceptable bound state. Thus we may assume

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

So far we are able to compute $\mathbf{x}_n$ (and thus ψ_n) satisfying conditions (1) and (2) for any values of $V_1, V_2, \ldots$. Condition (3) is a statement about the large n behaviour of ψ_n. This can be very difficult to determine, unless we know more about the values V_n.

IV.5.3. A potential with most values zero

We will consider the very simplest situation where $V_1 = -a$ and all the other V_n's are equal to zero. Let us try to determine for what energies E a bound state exists.

In this situation

$$\mathbf{x}_n = A(-E)^{n-1}A(-a-E)\mathbf{x}_0 = A(-E)^{n-1}\mathbf{x}_1$$

where

$$\mathbf{x}_1 = A(-a-E)\mathbf{x}_0 = \begin{bmatrix} -a-E & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -(a+E) \\ 0 \end{bmatrix}$$

The large n behavior of $\mathbf{x}_n$ can be computed using the eigenvalues and eigenvectors of $A(-E)$. Suppose they are $\lambda_1, \mathbf{v}_1$ and $\lambda_2, \mathbf{v}_2$. Then we expand

$$\mathbf{x}_1 = a_1\mathbf{v}_1 + a_2\mathbf{v}_2$$

and conclude that

$$\mathbf{x}_n = A(-E)^{n-1}(a_1\mathbf{v}_1 + a_2\mathbf{v}_2) = a_1\lambda_1^{n-1}\mathbf{v}_1 + a_2\lambda_2^{n-1}\mathbf{v}_2$$

Keep in mind that all the quantities in this equation depend on E. Our goal is to choose E so that the $\mathbf{x}_n$ become small for large n.

Before computing the eigenvalues of $A(-E)$, let's note that $\det(A(-E)) = 1$. This implies that $\lambda_1\lambda_2 = 1$

Suppose the eigenvalues are complex. Then, since $A(-E)$ has real entries, they must be complex conjugates. Thus $\lambda_2 = \overline{\lambda}_1$ and $1 = \lambda_1\lambda_2 = \lambda_1\overline{\lambda}_1 = |\lambda_1|^2$. This means that λ_1 and λ_2 lie on the unit circle in the complex plane. In other words, $\lambda_1 = e^{i\theta}$ and $\lambda_2 = e^{-i\theta}$ for some θ. This implies that $\lambda_1^{n-1} = e^{i(n-1)\theta}$ is also on the unit circle, and is not getting small. Similarly λ_2^{n-1} is not getting small. So complex eigenvalues will never lead to bound states. In fact, complex eigenvalues correspond to *scattering* states, and the energy values for which eigenvalues are complex are the energies at which the electron can move freely through the crystal.

Suppose the eigenvalues are real. If $|\lambda_1| > 1$ then $|\lambda_2| = 1/|\lambda_1| < 1$ and vice versa. So one of the products λ_1^{n-1}, λ_2^{n-1} will be growing large, and one will be getting small. So the only way that $\mathbf{x}_n$ can be getting small is if the coefficient a_1 or a_2 sitting in front of the growing product is zero.

Now let us actually compute the eigenvalues. They are

$$\lambda = \frac{-E \pm \sqrt{E^2-4}}{2}.$$

If $-2 < E < 2$ then the eigenvalues are complex, so there are no bound states. The interval $[-2, 2]$ is the conduction band, where the electrons can move through the crystal.

If $E = \pm 2$ then there is only one eigenvalue, namely 1. In this case there actually is only one eigenvector, so our analysis doesn't apply. However there are no bounds states in this case.

Now let us consider the case $E < -2$. Then the large eigenvalue is $\lambda_1 = (-E + \sqrt{E^2 - 4})/2$ and the corresponding eigenvector is $\mathbf{v}_1 = \begin{bmatrix} -1 \\ E + \lambda_1 \end{bmatrix}$. The small eigenvalue is $\lambda_2 = (-E - \sqrt{E^2 - 4})/2$ and the corresponding eigenvector is $\mathbf{v}_2 = \begin{bmatrix} -1 \\ E + \lambda_2 \end{bmatrix}$. We must now compute a_1 and determine when it is zero. We have $[\mathbf{v}_1 | \mathbf{v}_2] \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \mathbf{x}_1$. This is 2×2 matrix equation that we can easily solve for $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$. A short calculation gives

$$a_1 = (\lambda_1 - \lambda_2)^{-1}(-(a + E)(E + \lambda_2) + 1).$$

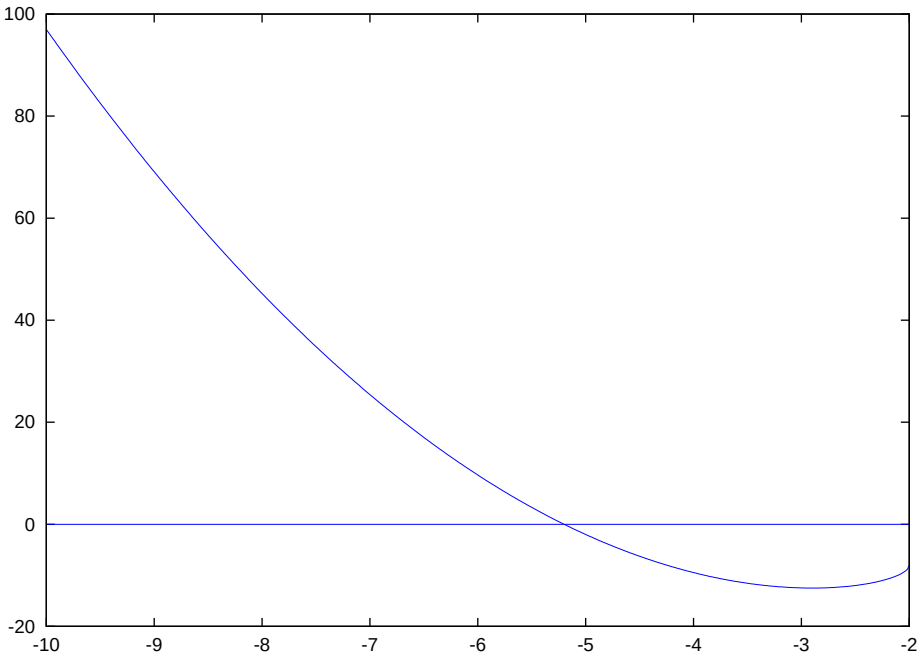
Thus we see that $a_1 = 0$ whenever

$$(a + E)(E - \sqrt{E^2 - 4}) - 2 = 0$$

Let's consider the case $a = 5$ and plot this function on the interval $[-10, -2]$. To see if it crosses the axis, we also plot the function zero.

```
>N=500;
>E=linspace(-10,-2,N);
>ONE=ones(1,N);
>plot(E,(5*ONE+E).*(E-sqrt(E.^2 - 4*ONE)) - 2*ONE)
>hold on
>plot(E,zeros(1,N))
```

Here is the result



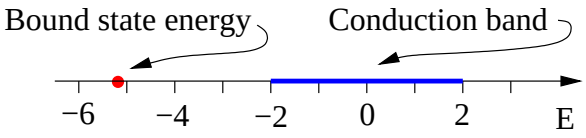
We can see that there is a single bound state in this interval, just below -5. In fact, the solution is $E = -5.2$.

The case $E > 2$ is similar. This time we end up with

$$(a + E)(E + \sqrt{E^2 - 4}) - 2 = 0$$

When $a = 5$ this never has a solution for $E > 2$. In fact the right side of this equation is bigger than $(5 + 2)(2 + 0) - 2 = 12$ and so can never equal zero.

In conclusion, if $V_1 = -5$, and all the other V_n's are zero, then there is exactly one bound state with energy $E = -5.2$. Here is a diagram of the energy spectrum for this potential.



For the bound state energy of $E = -5.2$, the corresponding wave function Ψ, and thus the probability that the electron is located at the nth lattice point can now also be computed. The evaluation of the infinite sum that gives the normalization constant N^2 can be done using a geometric series.

IV.5.4. Conduction bands for a crystal

The atoms in a crystal are arranged in a periodic array. We can model a one dimensional crystal in the tight binding model by considering potential values that repeat a fixed pattern. Let's focus on the case where the pattern is $1, 2, 3, 4$ so that the potential values are

$$V_1, V_2, V_3, \ldots = 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, \ldots$$

In this case, if we start with the formula

$$\mathbf{x}_n = A(V_n - E)A(V_{n-1} - E)\cdots A(V_1 - E)\begin{bmatrix}1\\0\end{bmatrix}$$

we can group the matrices into groups of four. The product

$$T(E) = A(V_4 - E)A(V_3 - E)A(V_2 - E)A(V_1 - E)$$

is repeated so that

$$\mathbf{x}_{4n} = T(E)^n \begin{bmatrix}1\\0\end{bmatrix}$$

Notice that the matrix $T(E)$ has determinant 1 since it is a product of matrices with determinant 1. So, as above, the eigenvalues λ_1 and λ_2 are either real with $\lambda_2 = 1/\lambda_1$, or complex conjugates on the unit circle. As before, the conduction bands are the energies E for which the eigenvalues of $T(E)$ are complex conjugates. It turns out that this happens exactly when

$$|\mathrm{tr}(T(E))| \leq 2$$

To see this, start with the characteristic polynomial for $T(E)$

$$\det(\lambda I - T(E)) = \lambda^2 - \mathrm{tr}(T(E))\lambda + \det(T(E))$$

(see homework problem). Since $\det(T(E)) = 1$ the eigenvalues are given by

$$\lambda = \frac{\mathrm{tr}(T(E)) \pm \sqrt{\mathrm{tr}(T(E))^2 - 4}}{2}.$$

When $|\mathrm{tr}(T(E))| \leq 2$ the quantity under the square root sign is negative, and so the eigenvalues have a non-zero imaginary part.

Let's use MATLAB/Octave to plot the values of $\mathrm{tr}(T(E))$ as a function of E. For convenience we first define a function that computes the matrices $A(z)$. To to this we type the following lines into a file called `A.m` in our working directory.

```
function A=A(Z)
  A=[Z -1; 1 0];
end
```

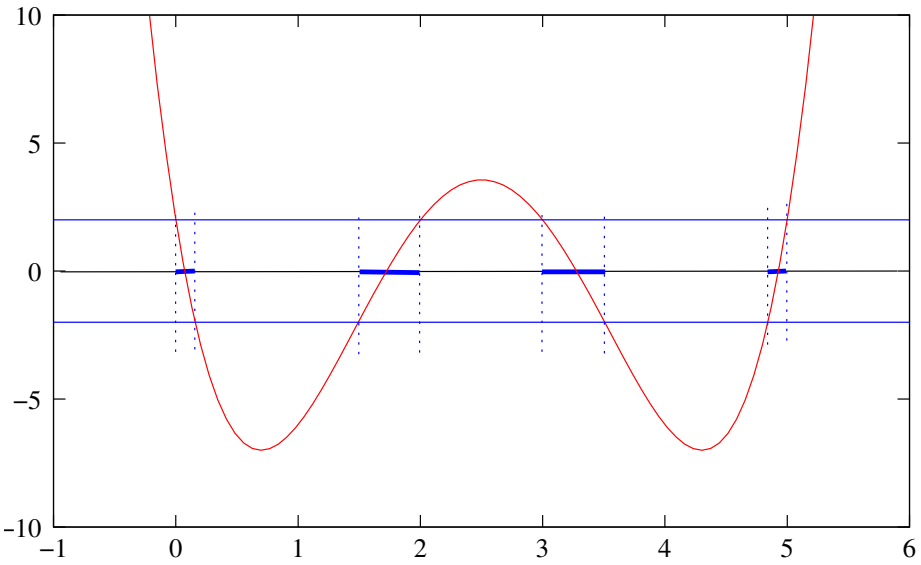
Next we start with a range of E values and define another vector T that contains the corresponding values of $\mathrm{tr}(T(E))$.

```
N=100;
E=linspace(-1,6,N);
T=[];
for e = E
  T=[T trace(A(4-e)*A(3-e)*A(2-e)*A(1-e))];
end
```

Finally, we plot `T` against `E`. At the same time, we plot the constant functions $E = 2$ and $E = -2$.

```
plot(E,T)
hold on
plot(E,2*ones(1,N));
plot(E,-2*ones(1,N));
axis([-1,6,-10,10])
```

On the resulting picture the energies where $T(E)$ lies between -2 and 2 have been highlighted.



We see that there are four conduction bands for this crystal.

IV.6. Markov Chains

Prerequisites and Learning Goals

After completing this section, you should be able to

- Write down the definition of a stochastic matrix, list its properties and explain why they are true; recognize when a matrix is stochastic.

- Write down the stochastic matrix and any relevant state vectors for a random walk, use them to describe the long-time behaviour of the random walk, explain why the probabilities in a random walk approach limiting values; use stochastic matrices to solve practical Markov chain problem. You should be able to perform these computations by hand or with MATLAB/Octave.

- For an internet specified by a directed graph, write down the stochastic matrix associated with the Google Page rank algorithm for a given damping factor α (and related algorithms), and compute the ranking of the sites (by hand or with MATLAB/Octave).

- Use the Metropolis algorithm to produce a stochastic matrix with a predetermined limiting probability distribution; use the algorithm to perform a random walk that converges with high probability to the maximum value of a given positive function.

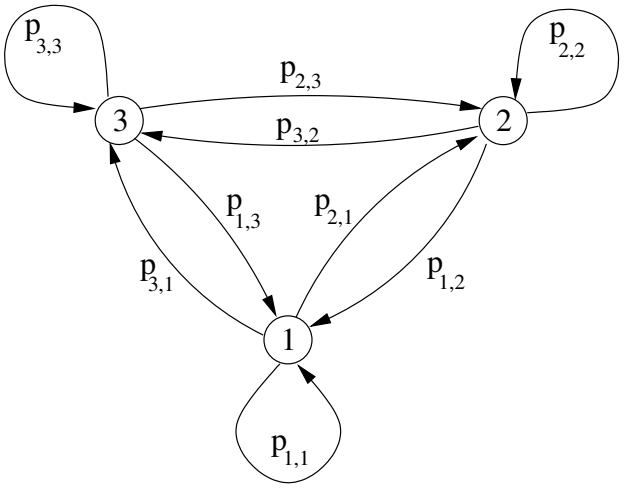
IV.6.1. Random walk

In the diagram below there are three sites labelled 1, 2 and 3. Think of a walker moving from site to site. At each step the walker either stays at the same site, or moves to one of the other sites according to a set of fixed probabilities. The probability of moving to the ith site from the jth site is denoted $p_{i,j}$. These numbers satisfy

$$0 \leq p_{i,j} \leq 1$$

because they are probabilities (0 means "no chance" and 1 means "for sure"). On the diagram they label the arrows indicating the relevant transitions. Since the walker has to go somewhere at each step the sum of all the probabilities leaving a given site must be one. Thus for every j,

$$\sum_i p_{i,j} = 1$$



Let $x_{n,i}$ be the probability that the walker is at site i after n steps. We collect these probabilities into a sequence of vectors called state vectors. Each state vector contains the probabilities for the nth step in the walk.

$$\mathbf{x}_n = \begin{bmatrix} x_{n,1} \\ x_{n,2} \\ \vdots \\ x_{n,k} \end{bmatrix}$$

The probability that the walker is at site i after $n+1$ steps can be calculated from probabilities for the previous step. It is the sum over all sites of the probability that the walker was at that site, times the probability of moving from that site to the ith site. Thus

$$x_{n+1,i} = \sum_j p_{i,j} x_{n,j}$$

This can be written in matrix form as

$$\mathbf{x}_n = P\mathbf{x}_{n-1}$$

where $P = [p_{i,j}]$. Using this relation repeatedly we find

$$\mathbf{x}_n = P^n \mathbf{x}_0$$

where $\mathbf{x}_0$ contains the probabilities at the beginning of the walk.

The matrix P has two properties:

1. All entries of P are non-negative.

2. Each column of P sum to 1.

A matrix with these properties is called a *stochastic* matrix.

The goal is to determine where the walker is likely to be located after many steps. In other words, we want to find the large n behaviour of $\mathbf{x}_n = P^n \mathbf{x}_0$. Let's look at an example. Suppose there are three sites, the transition probabilities are given by

$$P = \begin{bmatrix} 0.5 & 0.2 & 0.1 \\ 0.4 & 0.2 & 0.8 \\ 0.1 & 0.6 & 0.1 \end{bmatrix}$$

and the walker starts at site 1 so that initial state vector is

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Now let's use MATLAB/Octave to calculate the subsequent state vectors for $n = 1, 10, 100, 1000$.

```
>P=[.5 .2 .1; .4 .2 .8; .1 .6 .1];
>X0=[1; 0; 0];
>P*X0

ans =

   0.50000
   0.40000
   0.10000

>P^10*X0

ans =

   0.24007
   0.43961
   0.32032

>P^100*X0

ans =

   0.24000
   0.44000
   0.32000

P^1000*X0

ans =
```

```
   0.24000
   0.44000
   0.32000
```

The state vectors converge. Let's see what happens if the initial vector is different, say with equal probabilities of being at the second and third sites.

```
>X0 = [0; 0.5; 0.5];
>P^100*X0

ans =

   0.24000
   0.44000
   0.32000
```

The limit is the same. Of course, we know how to compute high powers of a matrix using the eigenvalues and eigenvectors. A little thought would lead us to suspect that P has an eigenvalue of 1 that is largest in absolute value, and that the corresponding eigenvector is the limiting vector, up to a multiple. Let's check

```
>eig(P)

ans =

   1.00000
   0.35826
  -0.55826

>P*[0.24000; 0.44000; 0.32000]

ans =

   0.24000
   0.44000
   0.32000
```

IV.6.2. Properties of stochastic matrices

The fact that the matrix P in the example above has an eigenvalue of 1 and an eigenvector that is a state vector is no accident. Any stochastic matrix P has the following properties:

(1) If $\mathbf{x}$ is a state vector, so is $P\mathbf{x}$.

(2) P has an eigenvalue $\lambda_1 = 1$.

(3) The corresponding eigenvector $\mathbf{v}_1$ has all non-negative entries.

(4) The other eigenvalues λ_i have $|\lambda_i| \leq 1$

If P or some power P^k has all positive entries (that is, no zero entries) then

(3') The eigenvector $\mathbf{v}_1$ has all positive entries.

(4') The other eigenvalues λ_i have $|\lambda_i| < 1$

(Since eigenvectors are only defined up to non-zero scalar multiples, strictly speaking, (3) and (3') should say that after possibly multiplying $\mathbf{v}_1$ by -1 the entries are non-negative or positive.) These properties explain the convergence properties of the state vectors of the random walk. Suppose (3') and (4') hold and we expand the initial vector $\mathbf{x}_0$ in a basis of eigenvectors. (Here we are assuming that P is diagonalizable, which is almost always true.) Then

$$\mathbf{x}_0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k$$

so that

$$\mathbf{x}_n = P^n\mathbf{x}_0 = c_1\lambda_1^n\mathbf{v}_1 + c_2\lambda_2^n\mathbf{v}_2 + \cdots + c_k\lambda_k^n\mathbf{v}_k$$

Since $\lambda_1 = 1$ and $|\lambda_i| < 1$ for $i \neq 1$ we find

$$\lim_{n\to\infty} \mathbf{x}_n = c_1\mathbf{v}_1$$

Since each $\mathbf{x}_n$ is a state vector, so is the limit $c_1\mathbf{v}_1$. This allows us to compute c_1 easily. It is the reciprocal of the sum of the entries of $\mathbf{v}_1$. In particular, if we chose $\mathbf{v}_1$ to be a state vector then $c_1 = 1$.

Now we will go through the properties above and explain why they are true

(1) *P preserves state vectors:*

Suppose $\mathbf{x}$ is a state vector, that is, $\mathbf{x}$ has non-negative entries which sum to 1. Then $P\mathbf{x}$ has non-negative entries too, since all the entries of P are non-negative. Also

$$\sum_i (P\mathbf{x})i = \sum_i \sum_j P_{i,j}x_j = \sum_j \sum_i P_{i,j}x_j = \sum_j x_j \sum_i P_{i,j} = \sum_j x_j = 1$$

Thus the entries of $P\mathbf{x}$ also sum to one, and $P\mathbf{x}$ is a state vector.

(2) *P has an eigenvalue $\lambda_1 = 1$*

The key point here is that P and P^T have the same eigenvalues. To see this recall that $\det(A) = \det(A^T)$. This implies that

$$\det(\lambda I - P) = \det((\lambda I - P)^T) = \det(\lambda I^T - P^T) = \det(\lambda I - P^T)$$

So P and P^T have the same characteristic polynomial. Since the eigenvalues are the zeros of the characteristic polynomial, they must be the same for P and P^T. (Notice that this *does not* say that the eigenvectors are the same.)

Since P has columns adding up to 1, P^T has rows that add up to 1. This fact can be expressed as the matrix equation

$$P \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

But this equation says that 1 is an eigenvalue for P^T. Therefore 1 is an eigenvalue for P as well.

(4) *Other eigenvalues of P have modulus ≤ 1*

To show that this is true, we use the 1-norm introduced at the beginning of the course. Recall that the norm $\|\cdot\|_1$ is defined by

$$\left\| \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \right\|_1 = |x_1| + |x_2| + \cdots + |x_n|.$$

Multiplication by P decreases the length of vectors if we use this norm to measure length. In other words

$$\|P\mathbf{x}\|_1 \leq \|\mathbf{x}\|_1$$

for any vector $\mathbf{x}$. This follows from the calculation (almost the same as one above, and again using the fact that columns of P are positive and sum to 1)

$$\begin{aligned}
\|P\mathbf{x}\|_1 &= \sum_i |(P\mathbf{x})_i| \\
&= \sum_i |(\sum_j P_{i,j}x_j)| \\
&\leq \sum_i \sum_j P_{i,j}|x_j| \\
&= \sum_j \sum_i P_{i,j}|x_j| \\
&= \sum_j |x_j| \sum_i P_{i,j} \\
&= \sum_j |x_j| \\
&= \|\mathbf{x}\|_1
\end{aligned}$$

Now suppose that λ is an eigenvalue, so that $P\mathbf{v} = \lambda\mathbf{v}$ for some non-zero $\mathbf{v}$. Then

$$\|\lambda\mathbf{v}\|_1 = \|P\mathbf{v}\|_1$$

Since $\|\lambda\mathbf{v}\|_1 = |\lambda|\|\mathbf{v}\|_1$ and $\|P\mathbf{v}\|_1 \leq \|\mathbf{v}\|_1$ this implies

$$|\lambda|\|\mathbf{v}\|_1 \leq \|\mathbf{v}\|_1$$

Finally, since $\mathbf{v}$ is not zero, $\|\mathbf{v}\|_1 > 0$. Therefore we can divide by $\|\mathbf{v}\|_1$ to obtain

$$|\lambda| \leq 1$$

(3) *The eigenvector* $\mathbf{v}_1$ *(or some multiple of it) has all non-negative entries*

We can give a partial proof of this, in the situation where the eigenvalues other than $\lambda_1 = 1$ obey the strict inequality $|\lambda_i| < 1$. In this case the power method implies that starting with any initial vector $\mathbf{x}_0$ the vectors $P^n\mathbf{x}_0$ converge to a multiple $c_1\mathbf{v}_1$ of $\mathbf{v}_1$. If we choose the initial vector $\mathbf{x}_0$ to have only positive entries, then every vector in the sequence $P^n\mathbf{x}_0$ has only non-negative entries. This implies that the limiting vector must have non-negative entries.

(3) *and* (4) *vs.* (3') *and* (4') *and* P^n *with all positive entries*

Saying that P^n has all positive entries means that there is a non-zero probability of moving between any two sites in n steps. The fact that in this case all the eigenvalues other than $\lambda_1 = 1$ obey the strict inequality $|\lambda_i| < 1$ follows from a famous theorem in linear algebra called the Perron–Frobenius theorem.

Although we won't be able to prove the Perron–Frobenius theorem, we can give some examples to show that if the condition that P^n has all positive entries for some n is violated, then (3') and (4') need not hold.

The first example is the matrix

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

This matrix represents a random walk with two sites that isn't very random. Starting at site one, the walker moves to site two with probability 1, and vice versa. The powers P^n of P are equal to I or P depending on whether n is even or odd. So $P^n \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ doesn't converge: it is equal to $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ for even n and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ for odd n. The eigenvalues of P are easily computed to be 1 and -1. They both have modulus one.

For the second example, consider a random walk where the sites can be divided into two sets A and B and the probability of going from any site in B to any site in A is zero. In this case the i, j entries P^n with the i site in A and the jth site in B are always zero. Also, applying P to any state vector drains probability from A to B without sending any back. This means that in the limit $P^n\mathbf{x}_0$ (that is, the eigenvector $\mathbf{v}_1$) will have zero entries for all sites in A. For example, consider a three site random walk (the very first example above) where there is no chance of ever going back to site 1. The matrix

$$P = \begin{bmatrix} 1/3 & 0 & 0 \\ 1/3 & 1/2 & 1/2 \\ 1/3 & 1/2 & 1/2 \end{bmatrix}$$

corresponds to such a walk. We can check

```
>P=[1/3 0 0;1/3 1/2 1/2; 1/3 1/2 1/2];
>[V D] = eig(P)

V =

   0.00000   0.00000   0.81650
   0.70711   0.70711  -0.40825
   0.70711  -0.70711  -0.40825

D =

   1.00000   0.00000   0.00000
   0.00000   0.00000   0.00000
   0.00000   0.00000   0.33333
```

The eigenvector corresponding to the eigenvalue 1 is $\begin{bmatrix} 0 \\ 1/2 \\ 1/2 \end{bmatrix}$ (after normalization to make it a state vector).

IV.6.3. Google PageRank

I'm going to refer you to the excellent article by David Austin:
`http://www.ams.org/featurecolumn/archive/pagerank.html`

Here are the main points:

The sites are web pages and connections are links between pages. The random walk is a web surfer clicking links at random. The rank of a page is the probability that the surfer will land on that page in the limit of infinitely many clicks.

We assume that the surfer is equally likely to click on any link on a given page. In other words, we assign to each outgoing link a transition probability of

$$1/(\text{total number of outgoing links for that page}).$$

This rule doesn't tell us what happens when the surfer lands on a page with no outgoing links (so-called dangling nodes). In this case, we assume that any page on the internet is chosen at random with equal probability.

The two rules above define a stochastic matrix

$$P = P_1 + P_2$$

where P_1 contains the probabilities for the outgoing links and P_2 contains the probabilities for the dangling nodes. The matrix P_1 is very sparse, since each web page has typically about 10 outgoing links. This translates to 10 non-zero entries (out of about 2 billion) for each column of P_1. The matrix P_2 has a non-zero column for each dangling node. The entries in each non-zero column are all the same, and equal to $1/N$ where N is the total number of sites.

If $\mathbf{x}$ is a state vector, then $P_1\mathbf{x}$ is easy to compute, because P_1 is so sparse, and $P_2\mathbf{x}$ is a vector with all entries equal to the total probability that the state vector $\mathbf{x}$ assigns to dangling nodes, divided by N, the total number of sites.

We could try to use the matrix P to define the rank of a page, by taking the eigenvector $\mathbf{v}_1$ corresponding to the eigenvalue 1 of P and defining the rank of a page to be the entry in $\mathbf{v}_1$ corresponding to that page. There are problems with this, though. Because P has so many zero entries, it is virtually guaranteed that P will have many eigenvalues with modulus 1 (or very close to 1) so we can't use the power method to compute $\mathbf{v}_1$. Moreover, there probably will also be many web pages assigned a rank of zero.

To avoid these problems we choose a number α between 0 and 1 called the damping factor. We modify the behaviour of the surfer so that with probability α the rules corresponding to the matrix P above are followed, and with probability $1-\alpha$ the surfer picks a page at random. This behaviour is described by the stochastic matrix

$$S = (1-\alpha)Q + \alpha P$$

where Q is the matrix where each entry is equal to $1/N$ (N is the total number of sites.) If $\mathbf{x}$ is a state vector, then $Q\mathbf{x}$ is a vector with each entry equal to $1/N$.

The value of α used in practice is about 0.85. The final ranking of a page can then be defined to be the entry in $\mathbf{v}_1$ for S corresponding to that page. The matrix S has all non-zero entries

IV.6.4. The Metropolis algorithm

So far we have concentrated on the situtation where a stochastic matrix P is given, and we are interested in finding invariant distribution (that is, the eigenvector with eigenvalue 1). Now we want to turn the situation around. Suppose we have a state vector π and we want to find a stochastic matrix that has this vector as its invariant distribution. The Metropolis algorithm, named after the mathematician and physicist Nicholas Metropolis, takes an arbitrary stochastic matrix P and modifies to produce a stochastic matrix Q that has π as its invariant distribution.

This is useful in a situation where we have an enormous number of sites and some function p that gives a non-negative value for each site. Suppose that there is one site where p is very much larger than for any of the other sites, and that our goal is to find that site. In other words, we have a discrete maximization problem. We assume that it is not difficult to compute p for any given site, but the number of sites is too huge to simply compute p for every site and see where it is the largest.

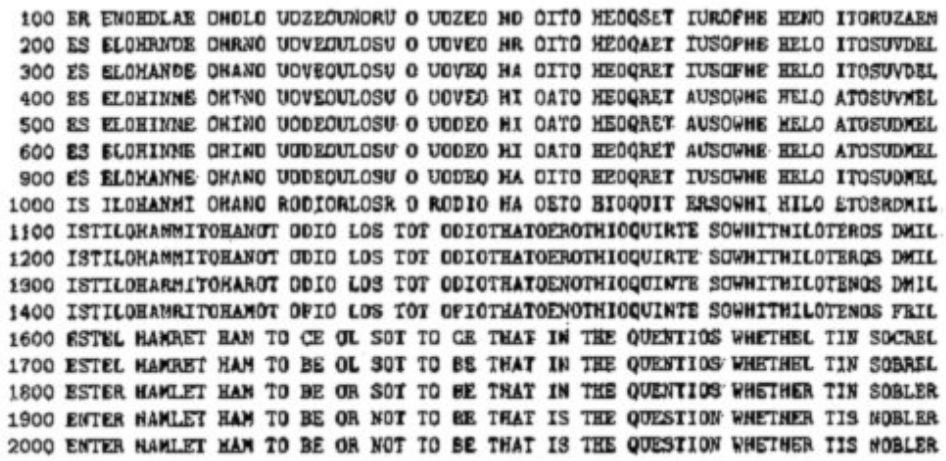
To solve this problem let's assume that the sites are labeled by integers $1, \ldots, N$. The vector $\mathbf{p} = [p_1, \ldots, p_N]^T$ has non-negative entries, and our goal is to find the largest one. We can form a state vector π (in principle) by normalizing $\mathbf{p}$, that is, $\pi = \mathbf{p}/\sum_i p_i$. Then the state vector π gives a very large probability to the site we want to find. Now we can use the Metropolis algorithm to define a random walk with π as its invariant distribution. If we step from site to site according to this random walk, chances are high that after a while we end up at the site where π is large.

In practice we don't want to compute the sum $\sum_i p_i$ in the denominator of the expression for π vector, since the number of terms is huge. An important feature of the Metropolis algorithm is that this computation is not required.

You can more learn about the Metropolis algorithm in the article *The Markov chain Monte Carlo revolution* by Persi Diaconis at

`http://www.ams.org/bull/2009-46-02/S0273-0979-08-01238-X/home.html`.

In this article Diaconis presents an example where the Metropolis algorithm is used to solve a substitution cipher, that is, a code where each letter in a message is replaced by another letter. The "sites" in this example are all permutations of a given string of letters and punctuation. The function p is defined by analyzing adjacent pairs of letters. Every adjacent pair of letters has a certain probablility of occuring in an English text. Knowing these probablities is it possible to construct a function p that is large on strings that are actually English. Here is the output of a random walk that is attempting to maximizde this function.



IV.6.5. Description of the algorithm

To begin, notice that a square matrix P with non-negative entries is stochastic if $P^T\mathbb{1} = \mathbb{1}$, where $\mathbb{1} = [1, 1, \ldots, 1]^T$ is the vector with entries all equal to 1. This is just another way of saying that the columns of P add up to 1.

Next, suppose we are given a state vector $\pi = [\pi_1, \pi_2, \ldots, \pi_n]^T$. Form the diagonal matrix Π that has these entries on the diagonal. Then, if P is a stochastic matrix, the condition

$$P\Pi = \Pi P^T$$

implies that π is the invartiant distribution for P. To see this, notice that $\Pi\mathbb{1} = \pi$ so applying the both sides of the equation to $\mathbb{1}$ yields $P\pi = \pi$.

This condition can be written as a collection of conditions on the components of P.

$$p_{i,j}\pi_j = \pi_i p_{j,i}$$

Notice that for diagonal entries $p_{i,i}$ this condition is always true. So we may make any changes we want to the diagonal entries without affecting this condition. For the off-diagonal entries ($i \neq j$) there is one equation for each pair $p_{i,j}$, $p_{j,i}$.

Here is how the Metropolis algorithm works. We start with a stochastic matrix P where these equations are not neccesarily true. For each off-diagonal pair $p_{i,j}, p_{j,i}$, of matrix entries, we make the equation above hold by by decreasing the value of the one of the entries, while leaving the other entry alone. It is easy to see that the adjusted value will still be non-negative.

Let Q denote the adjusted matrix. Then $Q\Pi = \Pi Q^T$ as required, but Q is not neccesarily a stochastic matrix. However we have the freedom to change the diagonal entries of Q. So set the diagonal entries of Q equal to

$$q_{i,i} = 1 - \sum_{j \neq i} q_{j,i}$$

Then the columns of Q add up to 1 as required. The only thing left to check is that the entries we just defined are non-negative. Since we have always made the adjustment so that $q_{i,j} \leq p_{i,j}$ we have

$$q_{i,i} \geq 1 - \sum_{j \neq i} p_{j,i} = p_{i,i} \geq 0$$

In practice we may not be presented with state vector π but with a positive vector $\mathbf{p} = [p_1, p_2, \ldots, p_n]^T$ whose maximal component we want to find. Although in principle it is easy to obtain π as $\pi = \mathbf{p}/(\sum_i p_i)$, in practice there may be so many terms in the sum that it is impossible to compute. However notice that the crucial equation $p_{i,j}\pi_j = \pi_i p_{j,i}$ is true for π_i and π_j if and only if it is true for p_i and p_j. Thus we may use the values of $\mathbf{p}$ instead.

Notice that if one of $p_{i,j}$ or $p_{j,i}$ is 0 then $q_{i,j} = q_{j,i} = 0$. So it is possible that this algorithm would yield $Q = I$. This is indeed a stochastic matrix where every state vector is an invariant distribution, but it is not very useful. To avoid examples like this, one could restrict the use of the Metropolis algorithm to matrices P where $p_{i,j}$ and $p_{j,i}$ are always either both zero or both nonzero.

IV.6.6. An example

In this example we will use the Metropolis algorithm to try to maximize a function $f(x, y)$ defined in the square $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$. We put down a uniform $(2N + 1) \times (2N + 1)$ grid and take the resulting lattice points to be our sites. The initial stochastic matrix P is the one that gives equal probabilities for travelling form a site to each neighbouring site. We use the Metropolis algorithm to modify this, obtaining a stochastic matrix Q whose invariant distribution is proportional to the sampled values of f. We then use Q to produce a random path that has a high probability of converging to the maximum of f.

To begin we write a MATLAB/Octave function `f.m` that defines a Gaussian function f with a peak at $(0, 0)$.

```
function f=f(x,y)

a = 10;
f = exp(-(a*x)^2-(a*y)^2);

end
```

Next we write a function `p.m` such that `p(i,j,k,l,N)` defines the matrix element for stochastic matrix P corresponding to the sites (i, j) and (k, l) when the grid size is $(2N + 1) \times (2N + 1)$.

```
function p = p(i,j,k,l,N)

% [i,j] and [k,l] are two points on a grid with
% -N <= i,j,k,l <= N. The probabilities are those for
% a uniform random walk, taking into account the edges

% Note: with our convention, p([i,j],[k,l],N) is the
% probability of going from [k,l] to [i,j]

% Note: if i,j,k,l are out of range, then p=0

if( ([k,l]==[N,N] | [k,l]==[N,-N] | [k,l]==[-N,N] | [k,l]==[-N,-N])
& abs(i-k)+abs(j-l)==1)
% starting point [k,l] is a corner:
p = 1/2;
elseif( (k==N | k==-N | l==N | l==-N) &  abs(i-k)+abs(j-l)==1)
% starting point [k,l] is on the edge:
p = 1/3;
elseif(abs(i-k)+abs(j-l)==1)
% starting point is inside
p = 1/4;
else
p = 0;
end

% handle out of range cases:
if(i<-N | i>N | j<-N | j>N | k<-N | k>N | l<-N | l>N )
p=0;
end

end
```

The next function `q.m` implements the Metropolis algorithm to give the matrix elements of Q.

```
function qq = q(i,j,k,l,N)

if(i~=k | j~=l)
if(p(i,j,k,l,N)==0)
qq = 0;
elseif(p(i,j,k,l,N)*f(k/N,l/N) > f(i/N,j/N)*p(k,l,i,j,N))
qq = p(i,j,k,l,N)*f(i/N,j/N)/f(k/N,l/N);
else
qq = p(i,j,k,l,N);
```

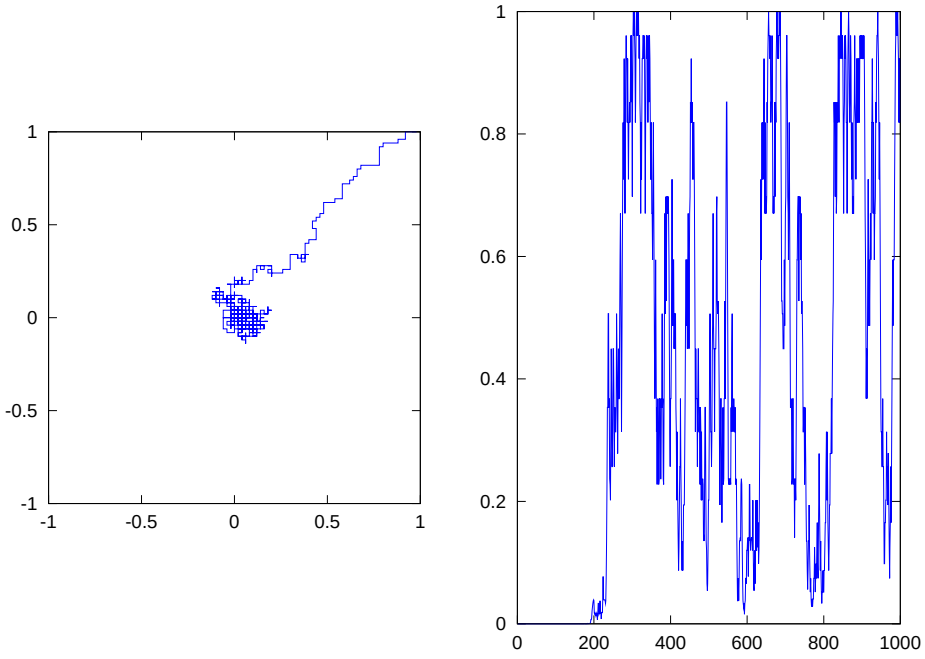
```
end
else
qq = 1 - q(k+1,l,k,l,N) - q(k-1,l,k,l,N) - q(k,l+1,k,l,N) - q(k,l-1,k,l,N);
end
```

Finally, here are the commands that compute a random walk defined by Q

```
% set the grid size
N=50;

% set the number of iterations
niter=1000

% pick starting grid point [i,j] at random
k=discrete_rnd(1,[-N:N],ones(1,2*N+1)/(2*N+1));
l=discrete_rnd(1,[-N:N],ones(1,2*N+1)/(2*N+1));

% or start in a corner (if these are uncommented)
k=N;
l=N;

% initialize: X and Y contain the path
% F contains f values along the path
X=zeros(1,niter);
Y=zeros(1,niter);
F=zeros(1,niter);

% the main loop
for count=1:niter
% pick the direction to go in the grid,
% according to the probabilites in the stochastic matrix q
probs = [q(k,l+1,k,l,N),q(k+1,l,k,l,N),q(k,l-1,k,l,N),q(k-1,l,k,l,N),q(k,l,k,l,N)];
dn = discrete_rnd(1,[1,2,3,4,5],probs);
switch dn
case 1
l=l+1;
case 2
k=k+1;
case 3
l=l-1;
case 4
k=k-1;
end
```

```
% calculate X, Y and F values for the new grid point
X(count)=k/N;
Y(count)=l/N;
F(count)=f(k/N,l/N);
end

% plot the path
subplot(1,2,1);
plot(X,Y);
axis([-1,1,-1,1]);
axis equal

%plot the values of F along the path
subplot(1,2,2);
plot(F);
```

The resulting plot looks like



IV.7. The Singular Value Decomposition

Prerequisites and Learning Goals

After completing this section, you should be able to

- explain what the singular value decomposition of a matrix is
- explain why a Hermitian matrix always has real eigenvalues and an orthonormal basis of eigenvectors.
- write down the relationship between the singular values of a matrix and its matrix norm

IV.7.1. The matrix norm and eigenvalues

Recall that the matrix norm $\|D\|$ of a diagonal matrix

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

is the largest absolute value of a diagonal entry, that is, the largest value of $|\lambda_i|$. Since, for a diagonal matrix the diagonal entries λ_i are also the eigenvalues of D it is natural to conjecture that for *any* matrix A the matrix norm $\|A\|$ is the largest absolute value of an eigenvalue of A.

Let's test this conjecture on a 2×2 example.

```
A=[1 2; 3 4]
A =

   1   2
   3   4

eig(A)
ans =

  -0.37228
   5.37228

norm(A)
ans =  5.4650
```

Since $5.37228 \neq 5.4650$ we see that the conjecture is false. But before giving up on this idea, let's try one more time.

```
B=[1 3; 3 4]
B =

   1   3
   3   4

eig(B)
ans =

  -0.85410
   5.85410

norm(B)
ans =  5.8541
```

This time the norm and the largest absolute value of an eigenvalue are both equal to 5.8541

The difference between these two examples is that B is a Hermitian matrix (in fact, a real symmetric matrix) while A is not. It turns out that for any Hermitian matrix (recall this means $A^* = A$) the matrix norm *is* equal to the largest eigenvalue in absolute value. This is related to the fact that every Hermitian matrix A can be diagonalized by a unitary U: $A = UDU^*$ with D diagonal with the eigenvalues on the diagonal and U unitary. The point is that multiplying A by a unitary matrix doesn't change the norm. Thus D has the same norm as UD which has the same norm as $UDU^* = A$. But the norm of D is the largest eigenvalue in absolute value.

The singular value is a decompostion similar to this that is valid for any matrix. For an arbitrary matrix it takes the form

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

where U and V are unitary and Σ is a diagonal matrix with positive entries on the diagonal. These positve numbers are called the singular values of A. As we will see it is the largest singluar value that is equal to the matrix norm. The matrix A does not have to be a square matrix. In this case, the unitary matrices U and V are not only different, but have different sizes. The matrix Σ has the same dimensions as A.

IV.7.2. The singular value decomposition

When A is Hermitian then we can write $A = UDU^*$ where U is unitary and D is diagonal. There is a similar factorization that is valid for *any* $n \times n$ matrix A. This factorization is called the singular value decomposition, and takes the form

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

where U is an $n \times n$ unitary matrix, V is an $n \times n$ unitary matrix and Σ is an $n \times m$ diagonal matrix with non-negative entries. (If $n \neq m$ the diagonal of Σ starts at the top left corner and runs into one of the sides of the matrix at the other end.) Here is an example.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{3} & -1/\sqrt{2} & -1/\sqrt{6} \\ -1/\sqrt{3} & -1/\sqrt{2} & 1/\sqrt{6} \\ 1/\sqrt{3} & 0 & 2\sqrt{6} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The positive diagonal entries of Σ are called the singular values of A.

Here is how to construct the singular value decomposition in the special case where A is an invertible square $(n \times n)$ matrix. In this case U Σ and V will all be $n \times n$ as well.

To begin, notice that A^*A is Hermitian, since $(A^*A)^* = A^*A^{**} = A^*A$), Moreover, the (real) eigenvalues are all positive. To see this, suppose that $A^*A\mathbf{v} = \lambda\mathbf{v}$. Then, taking the inner product of both sides with $\mathbf{v}$, we obtain $\langle \mathbf{v}, A^*A\mathbf{v}\rangle = \langle A^*\mathbf{v}, A\mathbf{v}\rangle = \lambda\langle \mathbf{v}, \mathbf{v}\rangle$. The eigenvector $\mathbf{v}$ is by definition not the zero vector, and because we have assumed that A is invertible $A\mathbf{v}$ is not zero either. Thus $\lambda = \|A\mathbf{v}\|^2/\|\mathbf{v}\|^2 > 0$.

Write the positive eigenvalues of A^*A as $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$ and let Σ be the matrix with $\sigma_1, \sigma_2, \ldots, \sigma_n$ on the diagonal. Notice that Σ is invertible, and Σ^2 has the eigenvalues of A^*A on the diagonal. Since A^*A is Hermitian, we can find a unitary matrix V such that $A^*A = V\Sigma^2V^*$. Define $U = AV\Sigma^{-1}$. Then U is unitary, since $U^*U = \Sigma^{-1}V^*A^*AV\Sigma^{-1} = \Sigma^{-1}\Sigma^2\Sigma^{-1} = I$.

With these definition

$$U\Sigma V^* = AV\Sigma^{-1}\Sigma V^* = AVV^* = A$$

so we have produced the singular value decomposition.

Notice that U is in fact the unitary matrix that diagonalizes AA^* since $U\Sigma^2U^* = AV\Sigma^{-1}\Sigma^2\Sigma^{-1}V^*A^* = AVV^*A^* = AA^*$. Moreover, this formula shows that the eigenvalues of AA^* are the same as those of A^*A, since they are the diagonal elements of Σ^2.

In MATLAB/Octave, the singular value decomposition is computed using `[U,S,V]=svd(A)`. Let's try this on the example above:

```
 [U S V] = svd([1 1;1 -1;0 1])
U =

   0.57735  -0.70711  -0.40825
  -0.57735  -0.70711   0.40825
   0.57735   0.00000   0.81650

S =

   1.73205   0.00000
```

```
   0.00000   1.41421
   0.00000   0.00000

V =

   0  -1
   1  -0
```

IV.7.3. The matrix norm and singular values

The matrix norm of a matrix is the value of the largest singular value. This follows from the fact that multiplying a matrix by a unitary matrix doesn't change its matrix norm. So, if $A = U\Sigma V^*$, then the matrix norm $\|A\|$ is the same as the matrix norm $\|\Sigma\|$. But the Σ is a diagonal matrix with the singular values of A along the diagonal. Thus the matrix norm of Σ is the largest singular value of A.

Actually, the Hilbert Schmidt norm is also related to the singular values. Recall that for a matrix with real entries $\|A\|_{HS} = \sqrt{\mathrm{tr}(A^T A)}$. For a matrix with complex entries, the definition is $\|A\|_{HS} = \sqrt{\mathrm{tr}(A^* A)}$. Now, if $A = U\Sigma V^*$ is the singular value decomposition of A, then $A^*A = V\Sigma^2 V^*$. Thus $\mathrm{tr}(A^*A) = \mathrm{tr}(V\Sigma^2 V^*) = \mathrm{tr}(V^*V\Sigma^2) = \mathrm{tr}(\Sigma^2)$. Here we used that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ for any two matrices A and B and that $V^*V = I$. Thus $\|A\|_{HS} = \sqrt{\sum_i \sigma_i^2}$, where σ_i are the singular values.

Notice that if we define the vector of singular values $\sigma = [\sigma_1, \ldots, \sigma_n]^T$ then $\|A\| = \|\sigma\|_\infty$ and $\|A\|_{HS} = \|\sigma\|_2$. By taking other vector norms for σ, like the 1-norm, or the p-norm for other values of p, we can define a whole new family of norms for matrices. These norms are called Schatten p-norms and are useful in some situations.

IV.7.4. The null space of a measured formula matrix

We return to the formula matrix of a chemical system considered in section II.2.9, and consider a situation where the formula matrix A contains experimentally measured, rather than exact, values. For example, the chemical system could be a rock sample, instead of elements the chemical constituents could be oxides $O_1, \ldots O_n$ and the species a collection of minerals $M_1, \ldots M_m$ found in the rock. The formula matrix

$$A = \begin{array}{c} \\ O_1 \\ O_2 \\ \vdots \\ O_n \end{array} \begin{array}{c} \begin{array}{cccc} M_1 & M_2 & \cdots & M_m \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix} \end{array}$$

contains measured values $a_{i,j}$. As before, we want to determine possible reactions by finding the null space $N(A)$. However, if $n \geq m$, a generic $n \times m$ matrix does not have a non-zero nullspace.

So, because of the errors in the measured values $a_{i,j}$, there will practically never be a non-zero null space $N(A)$, even if it is actually present in the chemical system. We can use the singular value decomposition of A to determine if the null space is present, within experimental error.

According to the singular value decomposition A can be written as a product

$$A = U\Sigma V^T$$

where U is an $n \times n$ orthogonal matrix, V is an $m \times m$ orthogonal matrix, and (for $n \geq m$)

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

The singular values σ_i are non-negative numbers arranged in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$. The dimension of $N(A)$ is equal to the number of σ_i that equal zero. In a situation, say, where $\sigma_1, \ldots \sigma_{m-1}$ are large but σ_m is very small, we can simply set $\sigma_m = 0$ in Σ. If $\hat{\Sigma}$ is the resulting matrix, we compute $\hat{A} = U\hat{\Sigma}V^T$. Then $N(\hat{A})$ will be one dimensional and we can find the corresponding reaction. In fact, since $\hat{\Sigma}[0, 0, \ldots, 1]^T = 0$ it is easy to see that $\hat{A}V[0, 0, \ldots, 1]^T = U\hat{\Sigma}V^TV[0, 0, \ldots, 1]^T = U\hat{\Sigma}[0, 0, \ldots, 1]^T = 0$. Therefore $V[0, 0, \ldots, 1]^T$, which is the last column of V, is a non zero vector in $N(\hat{A})$ from which we can read off the reaction. If k of the σ_i are small enought to be discarded, then the corresponding reactions can be read off from the last k columns of V. (To satisfy the non-degeneracy condition we may need to take linear combinations if we are using more than one column.)

How do we decide if a σ_i is small enough to throw away? We need to know the error Δa in our measurments. Form the matrix $\hat{\Sigma}_i$ where all the singluar values *except* σ_i are set to zero. The largest entry of $\hat{A}_i = U\hat{\Sigma}_iV^T$ is the largest change that will occur if we discard σ_i. So if the largest entry of $\hat{A}_i$ is less than Δa then σ_i can be discarded. The k, jth entry of $\hat{A}_i$ can be written as an inner product $\langle e_k, \hat{A}_i e_j \rangle$ where e_k and e_j are standard unit vectors. Since

$$\begin{aligned} |\langle e_k, \hat{A}_i e_j \rangle| &\leq \|e_k\| \|\hat{A}_i\| \|e_j\| \\ &= \|U\hat{\Sigma}_iV^T\| \\ &\leq \|U\| \|\hat{\Sigma}_i\| \|V^T\| \\ &= \|\hat{\Sigma}_i\| \\ &= \sigma_i \end{aligned}$$

a sufficient condition is $\sigma_i \leq \Delta a$.

Here is an example with made up data. Suppose that the measured formula matrix is

$$A = \begin{bmatrix} 33.2 & 6.3 & 1.0 & 22.2 & 31.7 \\ 16.7 & 59.4 & 3.7 & 8.7 & 0.0 \\ 2.9 & 9.0 & 5.0 & 35.9 & 26.1 \\ 10.0 & 5.3 & 74.8 & 36.8 & 31.3 \\ 37.1 & 20.1 & 15.3 & 3.8 & 16.4 \end{bmatrix}$$

and the error in the measurements is $\Delta a = 0.1$

We calculate the singular value decomposition in MATLAB/Octave as follows.

```
> A = [33.2, 6.3, 1.0, 22.2, 31.7; 16.7, 59.4, 3.7, 8.7, 0.0; 2.9, 9.0, 5.0,
35.9, 26.1;10.0, 5.3, 74.8, 36.8, 31.3; 37.1, 20.1, 15.3, 3.8, 16.4];

> [U S V]=svd(A)
U =

  -0.3607348   0.1958034   0.6938994   0.1309949  -0.5769536
  -0.2782364   0.7614613  -0.4810714  -0.2626182  -0.2058233
  -0.3285601  -0.0042285   0.3696774  -0.7213246   0.4848297
  -0.7515318  -0.5244669  -0.3700403   0.0623230  -0.1389982
  -0.3459814   0.3267328   0.1161153   0.6242425   0.6085894

S =

Diagonal Matrix

   1.0824e+02            0            0            0            0
            0   6.5227e+01            0            0            0
            0            0   4.4061e+01            0            0
            0            0            0   3.2641e+01            0
            0            0            0            0   7.0541e-03

V =

  -3.5041e-01   3.9986e-01   3.7864e-01   6.6340e-01   3.6587e-01
  -3.0206e-01   7.6983e-01  -4.6536e-01  -2.5699e-01  -1.8306e-01
  -5.9629e-01  -4.7893e-01  -5.7058e-01   2.9917e-01   3.5851e-04
  -4.7299e-01  -1.1098e-01   2.5679e-01  -6.3131e-01   5.4724e-01
  -4.5463e-01  -7.6054e-02   4.9857e-01  -7.6154e-02  -7.3018e-01
```

Since $\sigma_5 = 00.0070541 \leq 0.1 = \Delta a$ we can set it to zero. The reaction coefficients are then given

by the last column of V. Thus the reaction is

$$0.36587M_1 + 0.0003.5851M_3 + 0.54724M_4 = 0.18306M_2 + 0.73018M_5$$

In fact, we can get a nicer looking equation if we normalize the coefficients. Treating the third coefficient as zero, we normalize so that the next smallest is one.

```
> VN=V(:,5)/V(2,5)
VN =

  -1.9986309
   1.0000000
  -0.0019584
  -2.9893892
   3.9887438
```

This yields the equivalent reaction $2M_1 + 3M_4 = M_2 + 4M_5$.

Applications of the linear algebra to petrology can be found in [4]. T. Gordon's student Curtis Brett tested their results using the singular value decomposition in [5]. See also [6] for the use of the singular value decomposition in petrology.

References

[3] E. Froese, *An outline of phase equilibrium*, preprint

[4] C. J. N. Fletcher and H. J. Greenwood, *Metamorphism and Structure of Penfold Creek Area, near Quesnel Lake, British Columbia*, J. Petrology (1979) 20 (4): 743-794.

[5]C. Brett, *Algebraic Applications to Petrology*, report submitted to Terence M. Gordon in partial fulfillment of the requirements for the degree of Masters of Science, Earth and Oceanic Science department UBC, (2007)

[6] G. W. Fisher, *Matrix analysis of metamorphic mineral assemblages and reactions*, Contributions to Mineralogy and Petrology, (1989) Volume 102, Number 1, 69-77

IV.8. Principal Coordinates Analysis (PCA)

IV.8.1. Introduction

Imagine we are interested in comparing a selection of n different wines from different vineyards. A connoisseur is given samples from pairs of wines and compares them, giving a (somewhat subjective) comparison rating for how similar or dissimilar they are: 0 if the wines cannot be told apart up to 10 if they are totally different. This comparison rating is referred to as the *dissimilarity* between the two wines. At the end of the day the connoisseur has compared all the different wines and assigned ratings to describe the similarity and dissimilarity of each pair. This data is combined into a *dissimilarity matrix* T where T_{ij} is the rating given to the comparison between wines i and j.

Even for a moderate selection, trying to make sense out of this data can be challenging. We want to group the wines into similar-tasting families but, unless there is a very conspicuous characteristic that is different (such as half of the selection being red and the other half white), it may be difficult to identify significant patterns or relationships just by staring at the data.

The idea of Principal Coordinates Analysis (PCA) is firstly to represent the different objects under consideration (in this case the wines) graphically, as points $\mathbf{v}_1$, $\mathbf{v}_2$, ..., $\mathbf{v}_n$ in $\mathbb{R}^p$ for some suitable choice of dimension p. The distance $\|\mathbf{v}_i - \mathbf{v}_j\|$ between points in the plot is chosen to reflect as closely as possible the entry T_{ij} in the dissimilarity matrix T. Our eye is much more capable of spotting relationships and patterns in such a plot than it is spotting them in the raw data. However another problem arises in doing this: we can only visualize two or possibly three dimensions ($p = 2$ or 3) whereas the data would most naturally be graphed in a much higher dimensional, possibly even $n-1$-dimensional, space. The second key idea of PCA is to reduce the number of dimensions being considered to those in which the variation is greatest.

IV.8.2. Definitions and useful properties

A real square matrix A is called *positive definite* if

$$\langle \mathbf{x}, A\mathbf{x} \rangle > 0$$

for any $\mathbf{x} \neq \mathbf{0}$.

A real square matrix A is called *positive semi-definite* if

$$\langle \mathbf{x}, A\mathbf{x} \rangle \geq 0$$

for any $\mathbf{x} \neq \mathbf{0}$.

A real matrix of the form $A^T A$ for any real matrix A is always positive semi-definite because

$$\langle \mathbf{x}, A^T A\mathbf{x} \rangle = \langle A\mathbf{x}, A\mathbf{x} \rangle = \|A\mathbf{x}\|^2 \geq 0.$$

A positive semi-definite matrix has all non-negative eigenvalues. The proof is as follows. Let $\mathbf{v}$ be an eigenvector of A with eigenvalue λ so

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

Because A is positive semi-definite we have

$$\langle \mathbf{v}, A\mathbf{v} \rangle \geq 0.$$

Using the fact that $\mathbf{v}$ is an eigenvector we also have

$$\langle \mathbf{v}, A\mathbf{v} \rangle = \langle \mathbf{v}, \lambda\mathbf{v} \rangle = \lambda \langle \mathbf{v}, \mathbf{v} \rangle.$$

Combining these two results and remembering that $\langle \mathbf{v}, \mathbf{v} \rangle > 0$ for an eigenvector, we have our result:

$$\lambda \geq 0.$$

IV.8.3. Reconstructing points in $\mathbb{R}^p$

We begin with a simple example, taking our objects to be a set of n points $\mathbf{w}_1$, $\mathbf{w}_2$, ..., $\mathbf{w}_n$ in $\mathbb{R}^p$. We take the dissimilarity between objects i and j to be the usual distance between those points

$$T_{ij} = \|\mathbf{w}_i - \mathbf{w}_j\|.$$

Is it possible to reconstruct the relative location of the points from the data in T alone?

It is important to notice that any reconstruction that we find is not unique. We are free to rotate, reflect and translate the points and they still satisfy the only requirement that we make of them, namely that the distances between the points are as specified.

Reconstructing two points in $\mathbb{R}^p$

Consider two points, $\mathbf{w}_1$ and $\mathbf{w}_2$, that are a known distance, say 2, apart. A single line can always be placed between any two points and therefore we expect that the points can be represented in one dimension. If we only know the distance between the points, then a possible representation of them is $\mathbf{v}_1 = \begin{bmatrix} 0 \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} 2 \end{bmatrix}$.

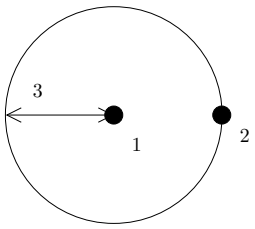
Reconstructing three points in $\mathbb{R}^p$

Now consider three points, $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ with dissimilarity matrix:

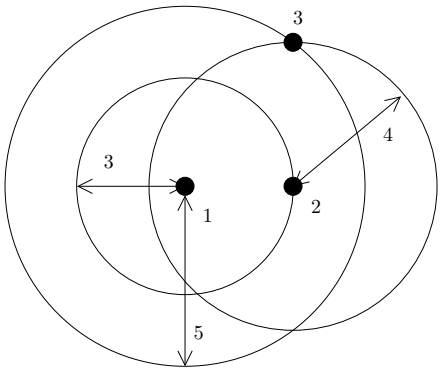
$$T = \begin{bmatrix} 0 & 3 & 5 \\ 3 & 0 & 4 \\ 5 & 4 & 0 \end{bmatrix}.$$

A plane can always be placed through any three points and therefore we expect that the points can be represented in two dimensions. We can find such a representation using trilateration. First we choose a point to represent $\mathbf{w}_1$:

Next we draw a circle of radius 3 centred at our first point and choose a second point on the circle to represent $\mathbf{w}_2$:



Finally we draw a circle of radius 5 centred at our first point and a circle of radius 4 centred at our second point and choose our third point to be at one of the intersections of these two circles to represent $\mathbf{w}_3$:



Thus we have been able to create a representation of the three points: $\mathbf{v}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{v}_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$ and $\mathbf{v}_3 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.

Arbitrary number of points in $\mathbb{R}^p$

The approach above works for a low dimensional space and with few points, but how can we handle many dimensions and/or many points?

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the representation of the points that we are trying to find. We assume that these points are in $\mathbb{R}^n$ (from the discussion above we can infer that the points can be chosen to be in an $n-1$ dimensional space, but choosing n makes the analysis below a little cleaner).

Let V be the matrix whose rows are the vectors for the points we are trying to find:

$$V = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \mathbf{v}_3^T \\ \vdots \\ \mathbf{v}_n^T \end{bmatrix}$$

We assume that the points are chosen such that the sum of each column of V is 0, in other words the centroid of the set of points is placed at the origin of $\mathbb{R}^n$. We are allowed to do this because of the translational freedom that we have in finding a solution.

All we know about the points is the distance that they are required to be apart. We have

$$\begin{aligned} (T_{ij})^2 &= \|\mathbf{v}_i - \mathbf{v}_j\|^2 = \langle \mathbf{v}_i - \mathbf{v}_j, \mathbf{v}_i - \mathbf{v}_j \rangle \\ &= \langle \mathbf{v}_i, \mathbf{v}_i \rangle + \langle \mathbf{v}_j, \mathbf{v}_j \rangle - 2\langle \mathbf{v}_i, \mathbf{v}_j \rangle. \end{aligned} \tag{IV.1}$$

It is easier to work with this equation in matrix form so we make the following definitions. Let B be the matrix whose entries are the inner products between the points

$$B_{ij} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \mathbf{v}_i^T \mathbf{v}_j,$$

or equivalently

$$B = VV^T.$$

Let Δ be the matrix containing the squares of the dissimilarities with entries

$$\Delta_{ij} = (T_{ij})^2.$$

Finally, let Q be the matrix with entries

$$Q_{ij} = \|\mathbf{v}_i\|^2$$

or equivalently

$$Q = \mathbf{q}\mathbf{e}^T,$$

where $\mathbf{q} = \begin{bmatrix} \|v_1\|^2 \\ \|v_2\|^2 \\ \vdots \\ \|v_n\|^2 \end{bmatrix}$ and $\mathbf{e} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$.

We can now rewrite (IV.1) as the matrix equation

$$\Delta = Q + Q^T - 2VV^T. \tag{IV.2}$$

We want to find the matrix V, and know the matrix Δ. So the next step is to eliminate Q from this equation. Notice that $\mathbf{q}\mathbf{e}^T\mathbf{e} = n\mathbf{q}$. This suggests that post-multiplying equation (IV.2) by

$$H = I - \frac{1}{n}\mathbf{e}\mathbf{e}^T$$

will eliminate Q:

$$QH = \mathbf{q}\mathbf{e}^T\left(I - (1/n)\mathbf{e}\mathbf{e}^T\right) = \mathbf{q}\mathbf{e}^T - \mathbf{q}\mathbf{e}^T = 0,$$

where 0 here represents the zero matrix. We also have that H is symmetric ($H = H^T$) so $HQ^T = H^TQ^T = (QH)^T = 0$. Therefore pre-multiplying by H eliminates Q^T. Applying these operations to (IV.2) we obtain the matrix equation

$$H\Delta H = -2HVV^TH = -2HVV^TH^T = -2HV(HV)^T.$$

We can simplify this by finding HV:

$$HV = \left(I - \frac{1}{n}\mathbf{e}\mathbf{e}^T\right)V = V - \frac{1}{n}\mathbf{e}\mathbf{e}^TV$$

but each entry of the row vector $\mathbf{e}^TV$ is the sum of each column of V which we assumed was zero. Therefore $\mathbf{e}^TV = \mathbf{0}^T$ and

$$HV = V.$$

We have now obtained the equation

$$VV^T = -\frac{1}{2}H\Delta H,$$

which is an equation that we are able to solve to find V. The matrix on the left is symmetric and positive semi-definite (using the properties we saw in §IV.8.2). The matrix on the right is symmetric (the dissimilarity matrix should be symmetric). In order for a solution to exist, it must also be positive semi-definite. In the example we are considering, we *know* that a solution of the problem exists because the dissimilarity matrix was constructed from a set of points in $\mathbb{R}^n$. Therefore in this case the matrix on the right *must* be positive semi-definite.

We can find a solution as follows. Because $-(1/2)H\Delta H$ is a real symmetric matrix it can be diagonalized as

$$-\frac{1}{2}H\Delta H = SDS^T,$$

where S is the matrix of normalized eigenvectors and D is the diagonal matrix of eigenvalues. Because it is positive semi-definite all eigenvalues are non-negative, and we can take the square root $D^{1/2}$ of D, where the entries on the leading diagonal are the square roots of the eigenvalues. We can now write

$$VV^T = (SD^{1/2})(SD^{1/2})^T. \tag{IV.3}$$

and a solution for V is

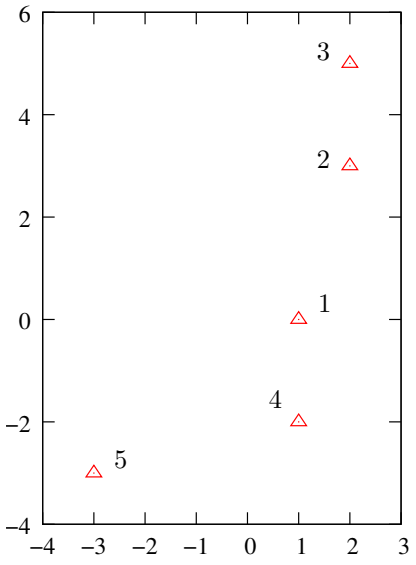
$$\boxed{V = SD^{1/2}}$$

Cautionary note: all we have actually shown is (IV.3). Our claim that $SD^{1/2}$ is a possible solution for V needs more justification. It is not immediately clear that just because this relation between V and $SD^{1/2}$ holds the rows of $SD^{1/2}$ must satisfy the same distance requirements as the rows of V. This result is in fact a particular property of matrices of the form AA^T. Assuming $AA^T = BB^T$, the fundamental idea is that the entries ij of AA^T and BB^T are the dot products of the rows i and j of A and B respectively. This means that corresponding rows of A and B must have the same length, and also the angle between rows i and j of A must equal the angle between the same rows of B for all i and j. Therefore the points represented by the rows of A and the points represented by the rows of B have the same magnitudes and relative angles and so we can find rotations and reflections mapping the points of A to the points of B.

Example

Consider the five points in $\mathbb{R}^2$ given by

$$\mathbf{w}_1 = \begin{bmatrix}1\\0\end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix}2\\3\end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix}2\\5\end{bmatrix}, \quad \mathbf{w}_4 = \begin{bmatrix}1\\-2\end{bmatrix}, \quad \mathbf{w}_5 = \begin{bmatrix}-3\\-3\end{bmatrix},$$



The dissimilarity matrix for these points is

$$T = \begin{bmatrix} 0 & \sqrt{10} & \sqrt{26} & 2 & 5 \\ \sqrt{10} & 0 & 2 & \sqrt{26} & \sqrt{61} \\ \sqrt{26} & 2 & 0 & 5\sqrt{2} & \sqrt{89} \\ 2 & \sqrt{26} & 5\sqrt{2} & 0 & \sqrt{17} \\ 5 & \sqrt{61} & \sqrt{89} & \sqrt{17} & 0 \end{bmatrix}$$

Knowing only T, can we find a representation of the points?

Using MATLAB/Octave we have

```
> T
T =

   0.00000   3.16228   5.09902   2.00000   5.00000
   3.16228   0.00000   2.00000   5.09902   7.81025
   5.09902   2.00000   0.00000   7.07107   9.43398
   2.00000   5.09902   7.07107   0.00000   4.12311
   5.00000   7.81025   9.43398   4.12311   0.00000

> H = eye(5)-1/5*ones(5,1)*ones(1,5);
> Delta = T.^2; % this finds the square of each element in T
> [S D] = eig(-0.5*H*Delta*H)
S =
  -0.0450409   0.9633529   0.2644287   0.2207222   0.7175132
   0.3690803   0.0063295   0.0398071  -0.6871823   0.3848777
   0.6031796   0.1253245  -0.3538343  -0.2243887  -0.3502305
  -0.2791402  -0.1936833   0.6580702  -0.5270235  -0.4611091
  -0.6480788   0.1367176  -0.6084717  -0.3885333   0.0419632

D =
   56.60551    0.00000    0.00000    0.00000    0.00000
    0.00000   -0.00000    0.00000    0.00000    0.00000
    0.00000    0.00000    5.79449    0.00000    0.00000
    0.00000    0.00000    0.00000    0.00000    0.00000
    0.00000    0.00000    0.00000    0.00000   -0.00000
```

Remember that MATLAB/Octave returns the eigenvectors normalized, so we do not need to normalize the columns of S.

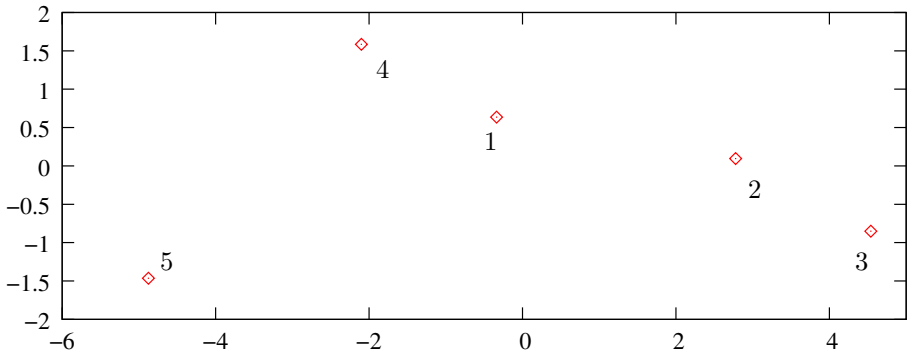
```
>  V = S*D^0.5
V =
 Columns 1 through 3:
```

```
  -0.33887 + 0.00000i   0.00000 + 0.00000i   0.63653 + 0.00000i
   2.77684 + 0.00000i   0.00000 + 0.00000i   0.09582 + 0.00000i
   4.53812 + 0.00000i   0.00000 + 0.00000i  -0.85174 + 0.00000i
  -2.10016 + 0.00000i  -0.00000 - 0.00000i   1.58409 + 0.00000i
  -4.87593 + 0.00000i   0.00000 + 0.00000i  -1.46470 + 0.00000i

 Columns 4 and 5:

   0.00000 + 0.00000i   0.00000 + 0.00000i
  -0.00000 + 0.00000i   0.00000 + 0.00000i
  -0.00000 + 0.00000i  -0.00000 - 0.00000i
  -0.00000 + 0.00000i  -0.00000 - 0.00000i
  -0.00000 + 0.00000i   0.00000 + 0.00000i
```

Each row of V is now one of the points we want to find in $\mathbb{R}^5$. Notice that only the first and third entries of each row are non-zero (this is a result of the second, fourth and fifth eigenvalues in D being zero). Therefore we can plot the result in $\mathbb{R}^2$ as follows:

```
> plot(V(:,1),V(:,3),'rs')
> axis([-6 5 -2 2])
```

and we obtain:



This plot is rotated, translated and reflected from the plot that we started off with, but the relative positions of the points are the same as those in the original.

IV.8.4. The dimension of the plot

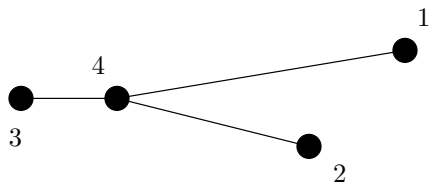
In the example above, if we had only been given the dissimilarity matrix T our best guess initially would have been that the 5 points came from a four-dimensional space. However we saw that we only needed two dimensions to plot the data: the components in the second, fourth and fifth coordinate directions were all zero because the corresponding eigenvalue was zero. Often most of the variation takes place in only a few directions (the directions of the largest eigenvalues) and we can look at the projection of the data points onto those coordinate directions (in the example above the first and third coordinates). Because the matrix S in the construction contains orthonormal vectors, the coordinate axes chosen in this way will be orthonormal.

There is a bit of an art (and controversy) to choosing the correct number of coordinates, but if the first two or three eigenvalues are substantially larger than the remainder it is reasonably safe just to use those for a comparison.

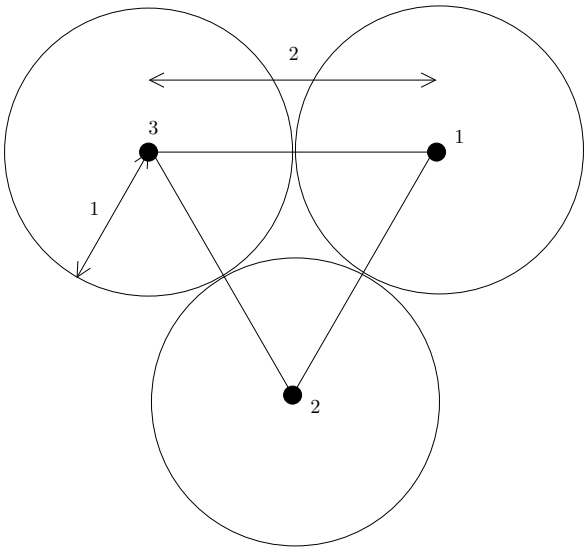
IV.8.5. Real examples

In most real-life applications, the dissimilarity matrix T does not produce a matrix $-H\Delta H/2$ that is positive semi-definite and so taking the square root of the matrix of eigenvalues D gives imaginary numbers. Consequently the decomposition $V = SD^{1/2}$ does not make sense.

An example is four cities connected by roads as follows:



We take the dissimilarity between two cities to be the driving time between them. If the journey time between cities along each road is exactly 1 hour (some roads are much better than others), then cities 1, 2 and 3 must all be an equal distance 2 apart in our representation. Therefore they are the vertices of an equilateral triangle. But now city 4 is only an hour apart from all three other cities and so must be a distance 1 apart from all the vertices of the equilateral triangle. But this is impossible (even with arbitrarily many dimensions):



One solution for this problem is to use only the coordinate directions with real eigenvalues. If these eigenvalues have significantly larger magnitudes than the negative eigenvalues, then this still produces a useful representation of the objects under consideration. This is the approach we will use in the examples below.

A difficult aspect of multi-dimensional scaling is determining how best to measure the dissimilarity between objects. This will depend on such things as what particular properties of the objects we are interested in and whether this data is quantitative (for example if we are comparing people we may be interested in their heights or their hair or eye colour). For our examples, we will assume that an appropriate choice of measurement has already been made.

Example 1

Take the set of objects to be the following towns and cities: Dawson Creek, Fort Nelson, Kamloops, Nanaimo, Penticton, Prince George, Prince Rupert, Trail, Vancouver and Victoria. We index them as

1	Dawson Creek
2	Fort Nelson
3	Kamloops
4	Nanaimo
5	Penticton
6	Prince George
7	Prince Rupert
8	Trail
9	Vancouver
10	Victoria

If our interest in the cities is their relative geographical location, then an appropriate measure of dissimilarity is the distance 'as the crow flies' between them. Then the entry T_{11} is the distance from Dawson Creek to itself (so 0 km), T_{12} and T_{21} are the distance between Dawson Creek and Fort Nelson, 374 km, and so on. The full dissimilarity matrix in km is:

$$T = \begin{bmatrix} 0 & 374 & 558 & 772 & 698 & 261 & 670 & 752 & 756 & 841 \\ 374 & 0 & 913 & 1077 & 1059 & 551 & 695 & 1124 & 1074 & 1159 \\ 558 & 913 & 0 & 306 & 151 & 384 & 782 & 261 & 260 & 330 \\ 772 & 1077 & 306 & 0 & 318 & 530 & 718 & 453 & 58 & 98 \\ 698 & 1059 & 151 & 318 & 0 & 535 & 912 & 140 & 260 & 295 \\ 261 & 551 & 384 & 530 & 535 & 0 & 504 & 629 & 524 & 610 \\ 670 & 695 & 782 & 718 & 912 & 504 & 0 & 1041 & 753 & 816 \\ 752 & 1124 & 261 & 453 & 140 & 629 & 1041 & 0 & 395 & 417 \\ 756 & 1074 & 260 & 58 & 260 & 524 & 753 & 395 & 0 & 86 \\ 841 & 1159 & 330 & 98 & 295 & 610 & 816 & 417 & 86 & 0 \end{bmatrix}$$

We can now find the PCA representation of the points using MATLAB/Octave (the m-file is on the web in `distance.m`):

```
> T = [0.   374.   558.   772.   698.   261.   670.   752.   756.    841.;
     374.     0.   913.  1077.  1059.   551.   695.  1124.  1074.   1159.;
     558.   913.     0.   306.   151.   384.   782.   261.   260.    330.;
     772.  1077.   306.     0.   318.   530.   718.   453.    58.     98.;
     698.  1059.   151.   318.     0.   535.   912.   140.   260.    295.;
     261.   551.   384.   530.   535.     0.   504.   629.   524.    610.;
     670.   695.   782.   718.   912.   504.     0.  1041.   753.    816.;
     752.  1124.   261.   453.   140.   629.  1041.     0.   395.    417.;
     756.  1074.   260.    58.   260.   524.   753.   395.     0.     86.;
     841.  1159.   330.    98.   295.   610.   816.   417.    86.      0.];

> Delta = T.^2;
> H = eye(size(T)) - 1./length(T)*ones(length(T),1)*ones(1,length(T));
> [S D] = eig(-0.5*H*Delta*H);
```

We would like to sort the eigenvalues such that the largest corresponds to the first coordinate, the next largest to the second, and so on. We can use the MATLAB/Octave command `sort` for this:

```
> [lambda,I] = sort(diag(D),'descend')
lambda =
   1.4615e+06
   4.4276e+05
   7.6808e+02
   2.4605e+02
```

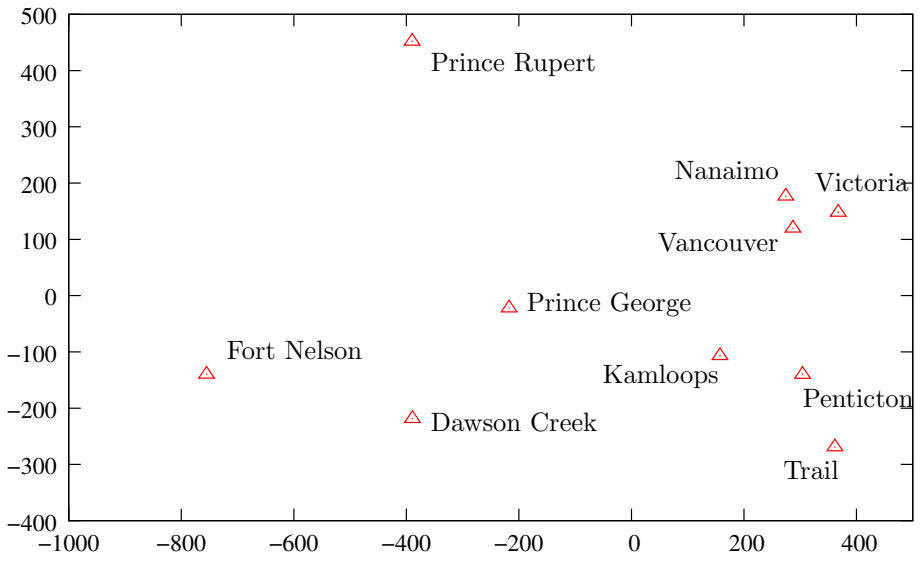
```
   1.5347e+02
   3.9772e+00
   1.4233e-11
  -2.9002e+02
  -4.5881e+02
  -1.1204e+03

I =
    1
    2
    4
    7
    8
   10
    9
    6
    5
    3
```

Here the vector `lambda` is the eigenvalues sorted from largest to smallest and `I` contains the original indices of the elements in the sorted vector. We see that the two largest eigenvalues have substantially greater magnitudes than the following ones. This is what we would expect: it should be approximately possible to represent these towns and cities as points on a plane, so only two dimensions are needed. We also notice that it is not possible to represent the points perfectly in any dimensional space because there are some negative eigenvalues. These result partly from the fact that the distances are measured along the (curved) surface of the Earth and partly from rounding the distances in T.

If we just take the first two coordinates and plot the points

```
> X = S(:,I(1))*sqrt(D(I(1),I(1)));
> Y = S(:,I(2))*sqrt(D(I(2),I(2)));
> plot(X,Y,'bo');
> axis equal
```

we obtain



Notice that although the orientation of the cities is both rotated and reflected, their relative positions are exactly what we would expect.

Example 2

Instead, we may be interested in how easily and quickly we can get between the different towns and cities. In this case, a more appropriate measure of dissimilarity is the *driving* distance between the different towns and cities, or better still the driving *time* between them. The dissimilarity matrix for driving times (in hours and minutes) is:

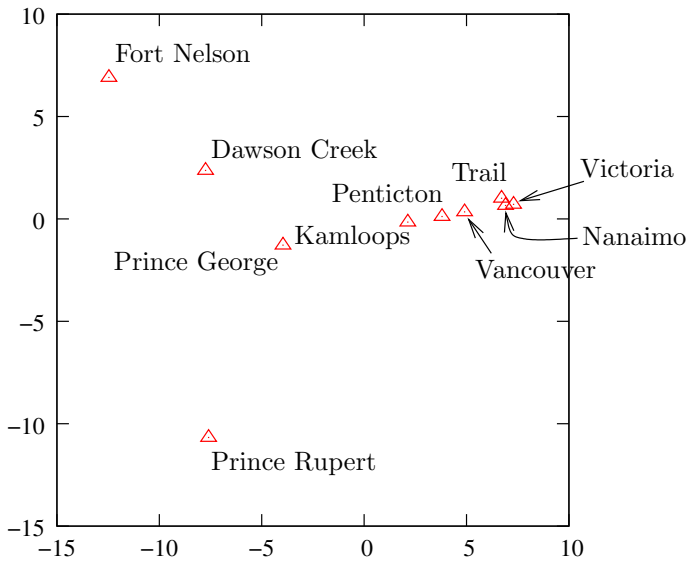
$$T = \begin{bmatrix} 0:00 & 5:27 & 10:06 & 15:02 & 12:14 & 4:27 & 12:38 & 15:28 & 12:47 & 15:20 \\ 5:27 & 0:00 & 15:42 & 20:39 & 17:50 & 10:03 & 18:14 & 20:55 & 18:23 & 20:56 \\ 10:06 & 15:42 & 0:00 & 5:49 & 2:53 & 5:39 & 13:49 & 5:43 & 3:33 & 6:04 \\ 15:02 & 20:39 & 5:49 & 0:00 & 8:45 & 10:35 & 18:46 & 9:24 & 2:27 & 1:31 \\ 12:14 & 17:50 & 2:53 & 8:45 & 0:00 & 7:47 & 15:57 & 3:29 & 4:25 & 6:56 \\ 4:27 & 10:03 & 5:39 & 10:35 & 7:47 & 0:00 & 8:11 & 11:12 & 8:20 & 10:52 \\ 12:38 & 18:14 & 13:49 & 18:46 & 15:57 & 8:11 & 0:00 & 19:23 & 16:31 & 19:03 \\ 15:28 & 20:55 & 5:43 & 9:24 & 3:29 & 11:12 & 19:23 & 0:00 & 7:09 & 9:41 \\ 12:47 & 18:23 & 3:33 & 2:27 & 4:25 & 8:20 & 16:31 & 7:09 & 0:00 & 3:03 \\ 15:20 & 20:56 & 6:04 & 1:31 & 6:56 & 10:52 & 19:03 & 9:41 & 3:03 & 0:00 \end{bmatrix}$$

If we try the same code as used for the geographical distance with the dissimilarity matrix containing the driving times between the cities we find

```
lambda =
   4.7754e+02
```

```
  1.7095e+02
  7.5823e+01
  1.0814e+01
  1.2364e+00
 -1.4217e-15
 -4.6973e-01
 -3.4915e+00
 -1.0043e+01
 -3.3626e+01
```

and using just the first two coordinates we obtain this plot:

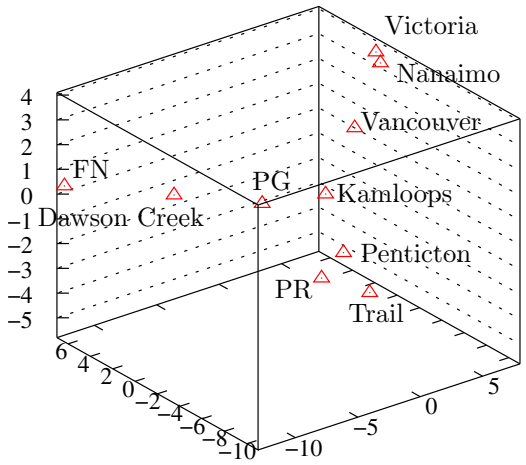


We notice that the structure is similar to that found for the distances, but there are a number of important differences. Roughly speaking major roads appear as straight lines on this plot: Vancouver to Kamloops is approximately Highways 1 and 5, then north from Kamloops to Prince George is Highway 97. Dawson Creek and Fort Nelson are found continuing along Highway 97 from Prince George while Prince Rupert is found by turning on to Highway 16. However there is a problem with what we have found: Trail is placed almost on top of Nanaimo and Victoria, when it is at the other side of the Province!

The problem is that we need to consider more principal coordinates to distinguish between Trail and Vancouver Island: the third eigenvalue is not much smaller than the second. If we add another principal coordinate

```
> X = S(:,I(1))*sqrt(D(I(1),I(1)));
> Y = S(:,I(2))*sqrt(D(I(2),I(2)));
> Z = S(:,I(3))*sqrt(D(I(3),I(3)));
> plot3(X,Y,Z,'bo')
> axis equal
```

then we see the following:



(The plot is much easier to understand if you plot it yourself and rotate it. The m-file is on the website in `hours.m`)

We can now identify Highways 3 and 97 to Trail branching off at Kamloops and the route by ferry to Vancouver Island (where a large part of the time is the ferry crossing which is similar for both Nanaimo and Victoria).