Topic 3

Iterative methods for $\mathbf{Ax} = \mathbf{b}$

3.1 Introduction

Earlier in the course, we saw how to reduce the linear system $\mathbf{Ax} = \mathbf{b}$ to echelon form using elementary row operations. Solution methods that rely on this strategy (e.g. **LU** factorization) are robust and efficient, and are fundamental tools for solving the systems of linear equations that arise in practice. These are known as direct methods, since the solution **x** is obtained following a single pass through the relevant algorithm. We are now going to look at some alternative approaches that fall into the category of iterative methods. These techniques can only be applied to square linear systems (*n* equations in *n* unknowns), but this is of course a common and important case.

Iterative methods for $\mathbf{Ax} = \mathbf{b}$ begin with an approximation to the solution, $\mathbf{x}_0$, then seek to provide a series of improved approximations $\mathbf{x}_1$, $\mathbf{x}_2$, … that converge to the exact solution. For the engineer, this approach is appealing because it can be stopped as soon as the approximations $\mathbf{x}_i$ have converged to an acceptable precision, which might be something as crude as 10^{-3}. With a direct method, bailing out early is not an option; the process of elimination and back-substitution has to be carried right through to completion, or else abandoned altogether. By far the main attraction of iterative methods, however, is that for certain problems (particularly those where the matrix **A** is large and sparse) they are much faster than direct methods. On the other hand, iterative methods can be unreliable; for some problems they may exhibit very slow convergence, or they may not converge at all.

3.2 Jacobi method ('simultaneous displacements')

The Jacobi method is the simplest iterative method for solving a (square) linear system **Ax** = **b**. Before developing a general formulation of the algorithm, it is instructive to explain the basic workings of the method with reference to a small example such as

$$\begin{bmatrix} 4 & 2 & 3 \\ 3 & -5 & 2 \\ -2 & 3 & 8 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 8 \\ -14 \\ 27 \end{bmatrix} \qquad (3.1)$$

Noting that there are no zeros on the leading diagonal, we can solve the first row for x, the second for y and the third for z:

$$\begin{aligned} x &= (8 - 2y - 3z)/4 \\ y &= (-14 - 3x - 2z)/(-5) \\ z &= (27 + 2x - 3y)/8 \end{aligned} \qquad (3.2)$$

This rearrangement is the key to the method, since it puts the equations in a form that is amenable to iteration. At each stage of the process, new values of x, y and z will be obtained by substituting the old values into the expressions on the right-hand side of eqn (3.2). In other words, we will apply the simple iterative scheme

$$
\begin{aligned}
x_{n+1} &= (8 - 2y_n - 3z_n)/4 \\
y_{n+1} &= (-14 - 3x_n - 2z_n)/(-5) \\
z_{n+1} &= (27 + 2x_n - 3y_n)/8
\end{aligned}
\qquad (3.3)
$$

In the absence of better information, we will begin with $x_0 = y_0 = z_0 = 0$ as our initial approximation. This makes the first iteration easy:

$$
\begin{aligned}
x_1 &= (8 - 2\times 0 - 3\times 0)/4 &&= 2 \\
y_1 &= (-14 - 3\times 0 - 2\times 0)/(-5) &&= 2.8 \\
z_1 &= (27 + 2\times 0 - 3\times 0)/8 &&= 3.375
\end{aligned}
$$

The second iteration gives

$$
\begin{aligned}
x_2 &= (8 - 2\times 2.8 - 3\times 3.375)/4 &&= -1.931 \\
y_2 &= (-14 - 3\times 2 - 2\times 3.375)/(-5) &&= 5.350 \\
z_2 &= (27 + 2\times 2 - 3\times 2.8)/8 &&= 2.825
\end{aligned}
$$

The third iteration gives

$$
\begin{aligned}
x_2 &= (8 - 2\times 5.350 - 3\times 2.825)/4 &&= -2.794 \\
y_2 &= (-14 - 3\times(-1.931) - 2\times 2.825)/(-5) &&= 2.771 \\
z_2 &= (27 + 2\times(-1.931) - 3\times 5.350)/8 &&= 0.886
\end{aligned}
$$

Although the early iterations don't look promising in terms of convergence, things do eventually settle down:

Iter.	*x*	*y*	*z*
1	2.000	2.800	3.375
2	-1.931	5.350	2.825
3	-2.794	2.771	0.886
4	-0.050	1.478	1.637
5	0.033	3.425	2.808
6	-1.819	3.943	2.099
7	-1.546	2.548	1.442
8	-0.355	2.449	2.033
9	-0.749	3.400	2.368
10	-1.476	3.297	1.913
11	-1.083	2.680	1.770
12	-0.667	2.858	2.099
13	-1.003	3.240	2.137
14	-1.222	3.053	1.909
15	-0.958	2.830	1.925
16	-0.859	2.995	2.074
17	-1.053	3.114	2.037
18	-1.085	2.983	1.944
19	-0.949	2.926	1.985
20	-0.952	3.024	2.040
21	-1.042	3.045	2.003
22	-1.025	2.976	1.973
23	-0.967	2.974	2.003
24	-0.989	3.021	2.018
25	-1.024	3.014	1.995

Iter.	*x*	*y*	*z*
26	-1.003	2.984	1.989
27	-0.984	2.994	2.005
28	-1.001	3.012	2.006
29	-1.011	3.002	1.995
30	-0.997	2.992	1.997
31	-0.993	3.000	2.004
32	-1.003	3.006	2.002
33	-1.004	2.999	1.997
34	-0.997	2.996	1.999
35	-0.998	3.001	2.002
36	-1.002	3.002	2.000
37	-1.001	2.999	1.999
38	-0.998	2.999	2.000
39	-1.000	3.001	2.001
40	-1.001	3.001	2.000
41	-1.000	2.999	1.999
42	-0.999	3.000	2.000
43	-1.000	3.001	2.000
44	-1.001	3.000	2.000
45	-1.000	3.000	2.000
46	-1.000	3.000	2.000
47	-1.000	3.000	2.000
48	-1.000	3.000	2.000
49	-1.000	3.000	2.000
50	-1.000	3.000	2.000

It can be verified by substitution in eqn (3.1) that the solution is indeed

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix}$$

3.3 Gauss-Seidel method ('successive displacements')

The Gauss-Seidel method is a variant of the Jacobi method that usually improves the rate of convergence. In the previous section, the procedure for conducting a Jacobi iteration was

$$\begin{aligned} x_{n+1} &= (8 - 2y_n - 3z_n)/4 \\ y_{n+1} &= (-14 - 3x_n - 2z_n)/(-5) \\ z_{n+1} &= (27 + 2x_n - 3y_n)/8 \end{aligned} \qquad \text{(3.3 bis)}$$

Note that in the second line, when working out y_{n+1}, we used the old value of x (i.e. x_n) even though a new and presumably more accurate value (i.e. x_{n+1}) had just been worked out in the first line. Similarly, in the third line, when working out z_{n+1}, we used the old values of x and y (i.e. x_n and y_n) even though updated values (i.e. x_{n+1} and y_{n+1}) had just been worked out in previous lines. The Gauss-Seidel method implements the strategy of always using the latest available value of a particular variable. For our small example, therefore, it gives the iterative scheme

$$
\begin{aligned}
x_{n+1} &= (8 - 2y_n - 3z_n)/4 \\
y_{n+1} &= (-14 - 3x_{n+1} - 2z_n)/(-5) \qquad (3.4) \\
z_{n+1} &= (27 + 2x_{n+1} - 3y_{n+1})/8
\end{aligned}
$$

These expressions should be compared very carefully with their Jacobi counterparts in eqn (3.3), where only 'old' variable values (subscript = n) appear on the right-hand sides. In the Gauss-Seidel method, we use 'new' variable values (subscript = $n + 1$) wherever possible.

To clarify the operation of the Gauss-Seidel method, we will go through the first few iterations of the example, again starting from $x_0 = y_0 = z_0 = 0$ as the initial approximation. The first iteration gives

$$
\begin{aligned}
x_1 &= (8 - 2\times 0 - 3\times 0)/4 &&= 2 \\
y_1 &= (-14 - 3\times 2 - 2\times 0)/(-5) &&= 4 \\
z_1 &= (27 + 2\times 2 - 3\times 4)/8 &&= 2.375
\end{aligned}
$$

which is already different from before. The second iteration gives

$$
\begin{aligned}
x_2 &= (8 - 2\times 4 - 3\times 2.375)/4 &&= -1.781 \\
y_2 &= (-14 - 3\times(-1.781) - 2\times 2.375)/(-5) &&= 2.681 \\
z_2 &= (27 + 2\times(-1.781) - 3\times 2.681)/8 &&= 1.924
\end{aligned}
$$

The third iteration gives

$$\begin{aligned} x_3 &= \left(8 - 2\times 2.681 - 3\times 1.924\right)/4 &&= -0.784 \\ y_3 &= \left(-14 - 3\times(-0.784) - 2\times 1.924\right)/(-5) &&= 3.099 \\ z_3 &= \left(27 + 2\times(-0.784) - 3\times 3.099\right)/8 &&= 2.017 \end{aligned}$$

Bearing in mind that the exact solution is $x = -1$, $y = 3$, $z = 2$, things are looking a great deal better than they were at the comparable stage of the Jacobi iterations, where we had $x = -2.794$, $y = 2.771$ and $z = 0.886$. If we carry on with the Gauss-Seidel iterations, we obtain:

Iter.	*x*	*y*	*z*
1	2.000	4.000	2.375
2	-1.781	2.681	1.924
3	-0.784	3.099	2.017
4	-1.062	2.969	1.996
5	-0.982	3.009	2.001
6	-1.006	2.997	2.000
7	-0.998	3.001	2.000
8	-1.001	3.000	2.000
9	-1.000	3.000	2.000
10	-1.000	3.000	2.000

Convergence to three decimal places in all components is achieved after just 9 iterations, whereas the Jacobi method took 45 iterations to achieve the same level of accuracy. It would be wrong, however, to assume that Gauss-Seidel is always superior to Jacobi; occasionally it is worse!

Intuitively, the Gauss-Seidel method seems more natural than the Jacobi method. If the solution is converging and updated information is available for some of the variables, surely it makes sense to use that information! From a programming point of view, the Gauss-Seidel method is definitely more convenient, since the old value of a variable can be overwritten as soon as a new value becomes available. With the Jacobi method, the values of all variables from the previous iteration need to be retained throughout the current iteration, which means that twice as much storage is needed. On the other hand, the Jacobi method is perfectly suited to parallel computation, whereas the Gauss-Seidel method is not. Because the Jacobi method updates or 'displaces' all of the variables at the same time (at the end of each iteration) it is often called the method of simultaneous displacements. The Gauss-Seidel method updates the variables one by one (during each iteration) so its corresponding name is the method of successive displacements.

3.4 Compact formulations

It is useful to be able to represent the Jacobi and Gauss-Seidel methods in terms of matrices and vectors. In fact, it turns out that both iterative schemes can be written (at least analytically – this is not how they are programmed on a computer!) in the compact form

$$\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q} \qquad (3.5)$$

where **P** is a constant matrix and **q** is a constant vector. Our task is to derive expressions for **P** and **q** in terms of the input data: the coefficient matrix **A** and the right-hand side vector **b**.

To help establish these general formulae, we will organize our thoughts by looking at a 3×3 system **Ax** = **b**:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Solving the first row for *x*, the second for *y* and the third for *z* gives the basic template

$$x = \left(b_1 - A_{12}y - A_{13}z\right)/A_{11}$$
$$y = \left(b_2 - A_{21}x - A_{23}z\right)/A_{22}$$
$$z = \left(b_3 - A_{31}x - A_{32}y\right)/A_{33}$$

Each iteration of the Jacobi method, cf. eqn (3.3), takes the form

$$x_{n+1} = \left(b_1 - A_{12}y_n - A_{13}z_n\right)/A_{11}$$
$$y_{n+1} = \left(b_2 - A_{21}x_n - A_{23}z_n\right)/A_{22}$$
$$z_{n+1} = \left(b_3 - A_{31}x_n - A_{32}y_n\right)/A_{33}$$

This can be rearranged as

$$A_{11}x_{n+1} = b_1 - A_{12}y_n - A_{13}z_n$$
$$A_{22}y_{n+1} = b_2 - A_{21}x_n - A_{23}z_n$$
$$A_{33}z_{n+1} = b_3 - A_{31}x_n - A_{32}y_n$$

or in matrix from

$$\begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \end{bmatrix}\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = -\begin{bmatrix} 0 & A_{12} & A_{13} \\ A_{21} & 0 & A_{23} \\ A_{31} & A_{32} & 0 \end{bmatrix}\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Thus, in general, the Jacobi method involves iterations of the form

$$\mathbf{D}\mathbf{x}_{n+1} = -\left(\mathbf{L} + \mathbf{U}\right)\mathbf{x}_n + \mathbf{b} \qquad (3.6)$$

where **D** is a 3×3 matrix containing the diagonal part of **A**, and **L** and **U** are 3×3 matrices containing the (strictly) lower and (strictly) upper triangular parts of **A**. (Note that the **L** and **U** matrices here have nothing to do

with those in the factorization $\mathbf{A} = \mathbf{LU}$.) Premultiplying by $\mathbf{D}^{-1}$ gives

$$\mathbf{x}_{n+1} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\mathbf{x}_n + \mathbf{D}^{-1}\mathbf{b} \qquad (3.7)$$

which is in the standard form of eqn (3.5).

Each iteration of the Gauss-Seidel method, cf. eqn (3.4), takes the form

$$\begin{aligned} x_{n+1} &= (b_1 - A_{12}y_n - A_{13}z_n)/A_{11} \\ y_{n+1} &= (b_2 - A_{21}x_{n+1} - A_{23}z_n)/A_{22} \\ z_{n+1} &= (b_3 - A_{31}x_{n+1} - A_{32}y_{n+1})/A_{33} \end{aligned}$$

This can be rearranged as

$$\begin{aligned} A_{11}x_{n+1} &= b_1 - A_{12}y_n - A_{13}z_n \\ A_{22}y_{n+1} &= b_2 - A_{21}x_{n+1} - A_{23}z_n \\ A_{33}z_{n+1} &= b_3 - A_{31}x_{n+1} - A_{32}y_{n+1} \end{aligned}$$

or in matrix form

$$\begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = -\begin{bmatrix} 0 & 0 & 0 \\ A_{21} & 0 & 0 \\ A_{31} & A_{32} & 0 \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} - \begin{bmatrix} 0 & A_{12} & A_{13} \\ 0 & 0 & A_{23} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Thus, in general, the Gauss-Seidel method involves iterations of the form

$$\mathbf{D}\mathbf{x}_{n+1} = -\mathbf{L}\mathbf{x}_{n+1} - \mathbf{U}\mathbf{x}_n + \mathbf{b} \qquad (3.8)$$

where **D**, **L** and **U** are as defined above. Premultiplying by $\mathbf{D}^{-1}$ gives

$$\mathbf{x}_{n+1} = -\mathbf{D}^{-1}\mathbf{L}\mathbf{x}_{n+1} - \mathbf{D}^{-1}\mathbf{U}\mathbf{x}_n + \mathbf{D}^{-1}\mathbf{b}$$

At first sight it looks strange to have $\mathbf{x}_{n+1}$ appearing on the right-hand side, until we realize that the matrix $\mathbf{D}^{-1}\mathbf{L}$ is (strictly) lower triangular. Thus, only part of the vector $\mathbf{x}_{n+1}$ is needed to process any given row, and that part is always available (from the results of previous rows). To represent the Gauss-Seidel method more compactly, we need to go back to eqn (3.8) and bring all terms related to $\mathbf{x}_{n+1}$ to the left-hand side,

$$(\mathbf{D} + \mathbf{L})\mathbf{x}_{n+1} = -\mathbf{U}\mathbf{x}_n + \mathbf{b} \qquad (3.9)$$

This expression should be compared with its Jacobi counterpart in eqn (3.6). Premultiplying by $(\mathbf{D} + \mathbf{L})^{-1}$ gives

$$\mathbf{x}_{n+1} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}\mathbf{x}_n + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b} \qquad (3.10)$$

which is in the standard form of eqn (3.5).

We can summarize the key equations (3.6, 3.7, 3.9, 3.10) as follows:

Jacobi	Gauss-Seidel
$\mathbf{D}\mathbf{x}_{n+1} = -(\mathbf{L}+\mathbf{U})\mathbf{x}_n + \mathbf{b}$	$(\mathbf{D}+\mathbf{L})\mathbf{x}_{n+1} = -\mathbf{U}\mathbf{x}_n + \mathbf{b}$
$\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q}$ where $\mathbf{P} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})$ $\mathbf{q} = \mathbf{D}^{-1}\mathbf{b}$	$\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q}$ where $\mathbf{P} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$ $\mathbf{q} = (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}$

In fact, both methods can be derived more quickly (though without such obvious connections to the basic algorithms as developed in Sections 3.1 and 3.2) by starting directly from the additive decomposition

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$$

The system $\mathbf{A}\mathbf{x} = \mathbf{b}$ becomes

$$(\mathbf{D} + \mathbf{L} + \mathbf{U})\mathbf{x} = \mathbf{b}$$

and it is then simply a question of how much we ‘leave’ on the left-hand side when setting up the iterative scheme.

If we leave just **Dx** we get

$$\mathbf{Dx} = -(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{b}$$

$$\Downarrow$$

$$\mathbf{Dx}_{n+1} = -(\mathbf{L} + \mathbf{U})\mathbf{x}_n + \mathbf{b}$$

which is the Jacobi method.

If we leave (**D** + **L**)**x** we get

$$(\mathbf{D} + \mathbf{L})\mathbf{x} = -\mathbf{U}\mathbf{x} + \mathbf{b}$$

$$\Downarrow$$

$$(\mathbf{D} + \mathbf{L})\mathbf{x}_{n+1} = -\mathbf{U}\mathbf{x}_n + \mathbf{b}$$

which is the Gauss-Seidel method.

Both iterative schemes are special cases of the more general

$$(\mathbf{D} + \omega\mathbf{L})\mathbf{x}_{n+1} = -\big((1-\omega)\mathbf{L} + \mathbf{U}\big)\mathbf{x}_n + \mathbf{b} \qquad (3.11)$$

With $\omega = 0$ we recover the Jacobi method. With $\omega = 1$ we recover the Gauss-Seidel method. We could, if we felt like it, take $\omega = 0.5$ (say) to obtain a method that lies 'between' Jacobi and Gauss-Seidel. We could even take $\omega > 1$, resulting in a method that goes 'beyond' Gauss-Seidel. This latter strategy takes us into the realm of over-relaxation, and for certain problems it turns out to be highly effective. It is possible to recast eqn (3.11) in such a way that the matrices on the left- and right-hand sides are lower and upper triangular respectively. This allows over-relaxation to be implemented in a manner similar to the Gauss-Seidel method, with new variable values overwriting old ones as soon as they become available. This type of iterative method is known as successive over-relaxation (SOR).

Example

Derive the Jacobi 'iteration matrix' **P** and 'iteration vector' **q** for the example used in Section 3.2.

Solution

From eqn (3.1) we have **Ax** = **b** with

$$\mathbf{A} = \begin{bmatrix} 4 & 2 & 3 \\ 3 & -5 & 2 \\ -2 & 3 & 8 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 8 \\ -14 \\ 27 \end{bmatrix}$$

We first split **A** into

$$\mathbf{D} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & 8 \end{bmatrix} \quad \text{and} \quad \mathbf{L} + \mathbf{U} = \begin{bmatrix} 0 & 2 & 3 \\ 3 & 0 & 2 \\ -2 & 3 & 0 \end{bmatrix}$$

Noting that $\mathbf{D}^{-1}$ is trivial to calculate from **D** (we just take the reciprocal of each term on the diagonal) we can easily work out

$$\mathbf{P} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$$
$$= -\begin{bmatrix} 0.25 & 0 & 0 \\ 0 & -0.2 & 0 \\ 0 & 0 & 0.125 \end{bmatrix}\begin{bmatrix} 0 & 2 & 3 \\ 3 & 0 & 2 \\ -2 & 3 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & -0.75 \\ 0.6 & 0 & 0.4 \\ 0.25 & -0.375 & 0 \end{bmatrix}$$

and

$$\mathbf{q} = \mathbf{D}^{-1}\mathbf{b}$$
$$= \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & -0.2 & 0 \\ 0 & 0 & 0.125 \end{bmatrix}\begin{bmatrix} 8 \\ -14 \\ 27 \end{bmatrix} = \begin{bmatrix} 2 \\ 2.8 \\ 3.375 \end{bmatrix}$$

The Jacobi iterations are given by $\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q}$, i.e.

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & -0.75 \\ 0.6 & 0 & 0.4 \\ 0.25 & -0.375 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} + \begin{bmatrix} 2 \\ 2.8 \\ 3.375 \end{bmatrix}$$

As expected, this is identical to the iterative scheme derived 'directly' in Section 3.2 (by making x the subject of the first equation, etc.):

$$\begin{aligned} x_{n+1} &= (8 - 2y_n - 3z_n)/4 \\ y_{n+1} &= (-14 - 3x_n - 2z_n)/(-5) \qquad \text{(3.3 bis)} \\ z_{n+1} &= (27 + 2x_n - 3y_n)/8 \end{aligned}$$

The first few Jacobi iterations using the compact form $\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q}$ look like this:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & -0.75 \\ 0.6 & 0 & 0.4 \\ 0.25 & -0.375 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2.8 \\ 3.375 \end{bmatrix} = \begin{bmatrix} 2 \\ 2.8 \\ 3.375 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & -0.75 \\ 0.6 & 0 & 0.4 \\ 0.25 & -0.375 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2.8 \\ 3.375 \end{bmatrix} + \begin{bmatrix} 2 \\ 2.8 \\ 3.375 \end{bmatrix} = \begin{bmatrix} -1.913 \\ 5.35 \\ 2.825 \end{bmatrix}$$

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & -0.75 \\ 0.6 & 0 & 0.4 \\ 0.25 & -0.375 & 0 \end{bmatrix} \begin{bmatrix} -1.913 \\ 5.35 \\ 2.825 \end{bmatrix} + \begin{bmatrix} 2 \\ 2.8 \\ 3.375 \end{bmatrix} = \begin{bmatrix} -2.794 \\ 2.771 \\ 0.886 \end{bmatrix}$$

The results are, of course, the same as those obtained previously.

3.5 Convergence criteria

Suppose we have a (square) linear system **Ax** = **b** with the exact solution **x** = **s**. If we apply an iterative scheme of the standard form

$$\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q} \qquad (3.12)$$

and the iterations converge, we will – by definition – eventually reach a steady state or 'fixed point' in which each iteration simply regenerates the exact solution:

$$\mathbf{s} = \mathbf{P}\mathbf{s} + \mathbf{q} \qquad (3.13)$$

If we subtract (3.13) from (3.12) the **q**'s disappear and we obtain

$$\mathbf{x}_{n+1} - \mathbf{s} = \mathbf{P}\left(\mathbf{x}_n - \mathbf{s}\right)$$

or equivalently

$$\mathbf{e}_{n+1} = \mathbf{P}\mathbf{e}_n \qquad (3.14)$$

where we have introduced the notation $\mathbf{e}_n$ to denote the error in the nth iterate $\mathbf{x}_n$ with respect to the exact solution:

$$\mathbf{e}_n = \mathbf{x}_n - \mathbf{s}$$

We can now proceed to analyze how the solution evolves from one iteration to the next. For convergence, we require the magnitude of the error vector **e** (as measured by some vector norm) to approach zero as the iterations proceed. Taking vector norms in eqn (3.14) gives

$$\|\mathbf{e}_{n+1}\| = \|\mathbf{P}\mathbf{e}_n\|$$

and applying the 'compatibility inequality' (see Section 2.2) gives

$$\|\mathbf{e}_{n+1}\| \leq \|\mathbf{P}\|\|\mathbf{e}_n\| \qquad (3.15)$$

This is a very important result. For if we can establish that the 'iteration matrix' **P** has the property

$$\|\mathbf{P}\| < 1$$

using <u>any</u> convenient matrix norm, then we can be certain (in advance!) that the iterative scheme will converge. The reason is that if $\|\mathbf{P}\|$ is strictly less than 1, then eqn (3.15) implies

$$\|\mathbf{e}_{n+1}\| < \|\mathbf{e}_n\|$$

for all *n*. (Note: here we tacitly presume the use of a vector norm that is compatible with the matrix norm used to establish $\|\mathbf{P}\| < 1$). Even if our initial approximation $\mathbf{x}_0$ has a large error (as measured by $\|\mathbf{e}_0\| = \|\mathbf{x}_0 - \mathbf{s}\|$), the first iteration is certain to reduce that error ($\|\mathbf{e}_1\| < \|\mathbf{e}_0\|$), the second is certain to reduce it again ($\|\mathbf{e}_2\| < \|\mathbf{e}_1\|$), and so on.

It follows that

$$\|\mathbf{e}_n\| \to 0 \quad \text{as} \quad n \to \infty$$

which in turn implies that

$$\mathbf{x}_n \to \mathbf{s} \quad \text{as} \quad n \to \infty$$

If possible, we would like to find a matrix norm $\|\mathbf{P}\|$ that is not just less than 1, but a good deal less than 1. If $\|\mathbf{P}\| = 0.1$, for example, eqn (3.15) shows that each iteration will reduce $\|\mathbf{e}\|$ by a factor of at least 10. We know in advance that the iterations will converge very quickly indeed.

If we choose a particular matrix norm, say the ∞-norm, and find that $\|\mathbf{P}\|_\infty$ is greater than 1, this does not necessarily indicate that the iterative scheme will fail to converge. For a start, there may be some other matrix norm (such as the 1-norm or Frobenius norm) that is strictly less than 1, in which case convergence is still guaranteed. In any case, however, the condition $\|\mathbf{P}\| < 1$ is only a sufficient condition for convergence, not a necessary one. That honour belongs to the spectral radius of **P** (the supremum among the absolute values of the eigenvalues):

The iterative scheme $\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q}$ is convergent if and only if every eigenvalue of $\mathbf{P}$ satisfies $|\lambda| < 1$, that is, the spectral radius $\rho(\mathbf{P}) < 1$.

The idea of over-relaxation (see Section 3.4) is to choose the factor ω in such a way that the spectral radius $\rho(\mathbf{P})$ is made as small as possible, thus giving the optimum rate of convergence. A similar idea is explored in Q6 of the tutorial sheet, in the context of a symmetric matrix (such that the spectral radius $\rho(\mathbf{P})$ and the matrix 2-norm $\|\mathbf{P}\|_2$ are the same).

Of course, matrix norms such as the 1-norm and ∞-norm are much easier to calculate than eigenvalues, so in practice we try to establish convergence using $\|\mathbf{P}\|_\infty$ and $\|\mathbf{P}\|_1$ (and maybe $\|\mathbf{P}\|_{\text{Fro}}$ if those fail).

Sometimes, a guarantee of convergence can be established by direct inspection of the coefficient matrix **A** (i.e. without needing to compute the iteration matrix **P**). In particular, if **A** has the following property, then the Jacobi and Gauss-Seidel schemes are both certain to converge.

A matrix **A** is diagonally dominated if, in each row, the absolute value of the entry on the diagonal is greater than the sum of the absolute values of the other entries. More compactly, **A** is diagonally dominated if

$$\left|A_{ii}\right| > \sum_{j,\, j\neq i} \left|A_{ij}\right| \quad \text{for all } i$$

In the case of the Jacobi scheme, it is not too hard to show that if **A** is diagonally dominated, then the 'iteration matrix' **P** has an ∞-norm that is strictly less than 1 (thus guaranteeing convergence). From the formula

$$\mathbf{P} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$$

it follows that

$$P_{ij} = \begin{cases} -A_{ij}/A_{ii} & \text{when } j \neq i \\ 0 & \text{when } j = i \end{cases} \qquad (3.16)$$

To understand where these expressions come from, it is instructive to write out $\mathbf{P} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})$ in full:

$$\begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix} = -\begin{bmatrix} 1/A_{11} & 0 & \cdots & 0 \\ 0 & 1/A_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/A_{nn} \end{bmatrix} \begin{bmatrix} 0 & A_{12} & \cdots & A_{1n} \\ A_{21} & 0 & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -A_{12}/A_{11} & \cdots & -A_{1n}/A_{11} \\ -A_{21}/A_{22} & 0 & \cdots & -A_{2n}/A_{22} \\ \vdots & \vdots & \ddots & \vdots \\ -A_{n1}/A_{nn} & -A_{n2}/A_{nn} & \cdots & 0 \end{bmatrix}$$

The proof itself is short:

$$\left|A_{ii}\right| > \sum_{j,\, j\neq i} \left|A_{ij}\right| \quad \forall i \qquad \text{definition of diagonal dominance}$$

$$\sum_{j,\, j\neq i} \left|\frac{A_{ij}}{A_{ii}}\right| < 1 \quad \forall i$$

$$\sum_{j} \left|P_{ij}\right| < 1 \quad \forall i \qquad \text{noting } P_{ii} = 0 \quad \forall i$$

$$\max_{i} \left(\sum_{j} \left|P_{ij}\right| \right) < 1$$

$$\left\|\mathbf{P}\right\|_{\infty} < 1$$

Example

Show that for each of the following matrices **A**, the system **Ax** = **b** can be solved by Jacobi iteration with guaranteed convergence.

$$\text{(a)}\ \begin{bmatrix} 5 & -1 & 3 \\ 2 & -8 & 1 \\ -2 & 0 & 4 \end{bmatrix} \qquad \text{(b)}\ \begin{bmatrix} -2 & 0 & 4 \\ 2 & -8 & 1 \\ 5 & -1 & 3 \end{bmatrix} \qquad \text{(c)}\ \begin{bmatrix} 4 & 2 & -2 \\ 0 & 4 & 2 \\ 1 & 0 & 4 \end{bmatrix}$$

Solution

(a) This matrix is diagonally dominated since $|5| > |-1| + |3|$, $|-8| > |2| + |1|$ and $|4| > |-2| + |0|$ are all true. From the above theorem we know that the Jacobi 'iteration matrix' **P** must have an ∞-norm that is strictly less than 1, but this is also easy to verify directly:

$$\mathbf{P} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$$

$$= -\begin{bmatrix} 0.2 & 0 & 0 \\ 0 & -0.125 & 0 \\ 0 & 0 & 0.25 \end{bmatrix}\begin{bmatrix} 0 & -1 & 3 \\ 2 & 0 & 1 \\ -2 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.2 & -0.6 \\ 0.25 & 0 & 0.125 \\ 0.5 & 0 & 0 \end{bmatrix}$$

Hence $\|\mathbf{P}\|_\infty = 0.8$, which as expected is < 1.

(b) This matrix is not diagonally dominated since $|-2| > |0| + |4|$ is false; the third row also violates. However, the first and third equations of **Ax** = **b** can be swapped to give a new system **A'x** = **b'** with

$$\mathbf{A}' = \begin{bmatrix} 5 & -1 & 3 \\ 2 & -8 & 1 \\ -2 & 0 & 4 \end{bmatrix}$$

which is diagonally dominated; in fact it is the matrix of part (a). The reordered system is suitable for Jacobi iteration.

(c) This matrix is not diagonally dominated since $|4| > |2| + |-2|$ is false (we need strict satisfaction of the inequality in each row). In this case, we cannot we achieve diagonal dominance by suitably reordering the rows. We therefore press on to compute the Jacobi 'iteration matrix'

$$\mathbf{P} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})$$
$$= -\begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.25 \end{bmatrix}\begin{bmatrix} 0 & 2 & -2 \\ 0 & 0 & 2 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & 0.5 \\ 0 & 0 & -0.5 \\ -0.25 & 0 & 0 \end{bmatrix}$$

and see what its norms look like. The row-sum norm $\|\mathbf{P}\|_\infty = 1$ (no great surprise – why?) and the column-sum norm $\|\mathbf{P}\|_1 = 1$. We are still looking for a matrix norm of **P** that is strictly less than 1. The Frobenius norm is our last chance, and happily we find

$$\|\mathbf{P}\|_{\text{Fro}} = \sqrt{(-0.5)^2 + (0.5)^2 + (-0.5)^2 + (-0.25)^2} = 0.901$$

3.6 Tips for hand calculations

When asked to solve a small system of equations by hand using the Jacobi method, we can either obtain the iterative scheme 'directly' by rearranging the equations, or we can form the matrix $\mathbf{P} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})$ and vector $\mathbf{q} = \mathbf{D}^{-1}\mathbf{b}$ to be used in $\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q}$. In terms of efficiency, there is nothing much to choose between the two approaches, and it is largely a matter of personal preference (or circumstance, e.g. we might already have worked out **P** to check its norms for convergence). When applying the Gauss-Seidel method, however, it is definitely easier to obtain the iterative scheme 'directly'; setting it up via $\mathbf{P} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$, $\mathbf{q} = (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}$ and $\mathbf{x}_{n+1} = \mathbf{P}\mathbf{x}_n + \mathbf{q}$ involves inverting the lower triangular matrix **D** + **L**. Actually this is not too hard to do by inspection, but it is certainly not as trivial as inverting **D**.

Example

Solve the following system of equations using (a) 3 iterations of the Jacobi method and (b) 3 iterations of the Gauss-Seidel method. Start with $x = y = z = 0$.

$$\begin{bmatrix} 2 & -1 & 0 \\ 1 & -3 & 1 \\ -1 & 1 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ -6 \end{bmatrix}$$

Solution

Rearranging the equations, we get the basic template

$$x = (2 + y)/2$$
$$y = (-2 - x - z)/(-3)$$
$$z = (-6 + x - y)/(-3)$$

(a) The iterative scheme for the Jacobi method is

$$x_{n+1} = (2 + y_n)/2$$
$$y_{n+1} = (-2 - x_n - z_n)/(-3)$$
$$z_{n+1} = (-6 + x_n - y_n)/(-3)$$

Jacobi iteration #1:

$$\begin{aligned} x_1 &= (2+0)/2 &&= 1 \\ y_1 &= (-2-0-0)/(-3) &&= 0.666 \\ z_1 &= (-6+0-0)/(-3) &&= 2 \end{aligned}$$

Jacobi iteration #2:

$$\begin{aligned} x_2 &= (2+0.666)/2 &&= 1.333 \\ y_2 &= (-2-1-2)/(-3) &&= 1.667 \\ z_2 &= (-6+1-0.666)/(-3) &&= 1.889 \end{aligned}$$

Jacobi iteration #3:

$$\begin{aligned} x_3 &= (2+1.667)/2 &&= 1.833 \\ y_3 &= (-2-1.333-1.889)/(-3) &&= 1.741 \\ z_3 &= (-6+1.333-1.667)/(-3) &&= 2.111 \end{aligned}$$

(b) The iterative scheme for the Gauss-Seidel method is

$$\begin{aligned} x_{n+1} &= (2+y_n)/2 \\ y_{n+1} &= (-2-x_{n+1}-z_n)/(-3) \\ z_{n+1} &= (-6+x_{n+1}-y_{n+1})/(-3) \end{aligned}$$

Gauss-Seidel iteration #1:

$$\begin{aligned} x_1 &= (2+0)/2 &&= 1 \\ y_1 &= (-2-1-0)/(-3) &&= 1 \\ z_1 &= (-6+1-1)/(-3) &&= 2 \end{aligned}$$

Gauss-Seidel iteration #2:

$$\begin{aligned} x_2 &= (2+1)/2 &&= 1.5 \\ y_2 &= (-2-1.5-2)/(-3) &&= 1.833 \\ z_2 &= (-6+1.5-1.833)/(-3) &&= 2.111 \end{aligned}$$

Gauss-Seidel iteration #3:

$$\begin{aligned} x_3 &= (2+1.833)/2 &&= 1.917 \\ y_3 &= (-2-1.917-2.111)/(-3) &&= 2.009 \\ z_3 &= (-6+1.917-2.009)/(-3) &&= 2.031 \end{aligned}$$

The Gauss-Seidel result is closer to the exact solution ($x = y = z = 2$).

Example (A1 2002 Q3)

(a) Define the l_p norm of a vector **x**. Determine the l_1, l_2 and l_∞ norms of the vector $\begin{bmatrix}2 & 0.2 & -5\end{bmatrix}^T$. Give an example of an application where the l_1 norm would be more applicable than the l_2 norm.

(b) Given that the l_∞ norm of a matrix is $\|\mathbf{A}\|_\infty = \max_i \left(\sum_j |A_{ij}| \right)$ (you do not need to prove this), what is the l_∞ norm of the matrix $\begin{bmatrix}17 & 2 \\ 10 & -16\end{bmatrix}$?

(c) Describe the Jacobi algorithm for solution of the system of equations **Ax** = **b**. State a condition on the norm of a matrix (not necessarily **A**), which must be satisfied for the method to converge.

(d) On the basis of the appropriate l_∞ matrix norm choose which of the equations below is more suitable for solution using Jacobi iteration. Obtain an approximate solution using three iterations starting from $\mathbf{x} = \begin{bmatrix}0.5 & 0.5\end{bmatrix}^T$, and comment on the solution.

$$\begin{bmatrix}3 & 5 \\ 3 & -4\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} = \begin{bmatrix}2 \\ 1\end{bmatrix} \quad \text{or} \quad \begin{bmatrix}3 & 2 \\ 3 & -4\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} = \begin{bmatrix}2 \\ 1\end{bmatrix}$$

Topic 4

Eigenvalue computations

4.1 Introduction

Let us first recall some terminology. If **A** is a square ($n \times n$) matrix, a nonzero vector **x** in $\mathbb{R}^n$ is an eigenvector of **A** if

$$\mathbf{Ax} = \lambda\mathbf{x} \qquad (4.1)$$

for some scalar λ. The scalar λ is an eigenvalue of **A**, and we say that **x** is an eigenvector corresponding to λ (or alternatively that **x** and λ are an eigenvector-eigenvalue pair). Geometrically, the $n \times n$ matrix **A** defines a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$. If the 'output' vector **Ax** points in the same direction as the 'input' vector **x**, then **x** is an eigenvector of the transformation. The eigenvalue λ is the factor (possibly negative) by which **x** is scaled to give **Ax**. Those of you who have ready access to Matlab might enjoy playing with the **eigshow** demonstration.

As you know from P1, eigenvalues can be found by hand provided the matrix **A** is small, e.g. $n = 3$. Eqn (4.1) is rearranged as $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$, and for non-trivial **x** we require the determinant of $\mathbf{A} - \lambda\mathbf{I}$ to be zero. This leads to the characteristic polynomial (an nth degree polynomial in λ, e.g. a cubic when $n = 3$), the roots of which are the eigenvalues. If **A** is large, however, this approach is no longer an option. Given that in practice we could be

dealing with huge matrices (n = 100,000 or more), even finding the coefficients of the characteristic polynomial is no longer on option! Instead we move to numerical methods that can find a small number of eigenvalues cheaply. Typically it is only the largest few eigenvalues, or else the smallest few, that are of interest in any given problem.

In this brief treatment, we will assume that all matrices

- are real and symmetric;
- have a dominant eigenvalue.

An eigenvalue of an $n \times n$ matrix **A** is called the dominant eigenvalue of **A** if its absolute value is strictly greater than the absolute values of all remaining $n - 1$ eigenvalues. The eigenvector corresponding to the dominant eigenvalue is called the dominant eigenvector.

4.2 Power method

The power method is a simple but effective way of finding the dominant eigenvalue and eigenvector of a matrix **A**. It starts with a (nonzero) estimate $\mathbf{x}_0$ for the dominant eigenvector, then iterates according to

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{A}\mathbf{x}_0\\ \mathbf{x}_2 &= \mathbf{A}\mathbf{x}_1\\ &\vdots\end{aligned}$$

Let's see how this works on a simple matrix such as

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

In order to evaluate the numerical results, we first find the exact ones:

$$\begin{aligned}&\det\begin{bmatrix} -1-\lambda & 2 \\ 2 & 2-\lambda \end{bmatrix} = 0\\ &(-1-\lambda)(2-\lambda) - 4 = 0\\ &\lambda^2 - \lambda - 6 = 0\\ &(\lambda+2)(\lambda-3) = 0\\ &\lambda = 3, -2\end{aligned}$$

Hence **A** has a dominant eigenvalue equal to 3 (since $|3| > |-2|$), and

$$\begin{bmatrix} -1-3 & 2 \\ 2 & 2-3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

gives the (non-normalized) dominant eigenvector as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

But the point of the example is to see how the power method gets on. We choose an arbitrary non-zero vector to get started, say

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

then launch into the iterations:

Iter.	**x**	**Ax**	Magn. ratio
1	$\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}1\\4\end{bmatrix}$	$\frac{4}{1}=4.000$
2	$\begin{bmatrix}1\\4\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}1\\4\end{bmatrix}=\begin{bmatrix}7\\10\end{bmatrix}$	$\frac{10}{4}=2.500$
3	$\begin{bmatrix}7\\10\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}7\\10\end{bmatrix}=\begin{bmatrix}13\\34\end{bmatrix}$	$\frac{34}{10}=3.400$
4	$\begin{bmatrix}13\\34\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}13\\34\end{bmatrix}=\begin{bmatrix}55\\94\end{bmatrix}$	$\frac{94}{34}=2.765$
5	$\begin{bmatrix}55\\94\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}55\\94\end{bmatrix}=\begin{bmatrix}133\\298\end{bmatrix}$	$\frac{298}{94}=3.170$
6	$\begin{bmatrix}133\\298\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}133\\298\end{bmatrix}=\begin{bmatrix}463\\862\end{bmatrix}$	$\frac{862}{298}=2.893$
7	$\begin{bmatrix}463\\862\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}463\\862\end{bmatrix}=\begin{bmatrix}1261\\2650\end{bmatrix}$	$\frac{2650}{862}=3.074$

The method clearly works. The magnitude of **x** is growing steadily (this can easily be fixed if required, see below) but its direction is converging to that of the dominant eigenvector (= $\begin{bmatrix}1 & 2\end{bmatrix}^T$). This can be seen if we look at the ratios of the components in the last few iterates:

$$\mathbf{x}_5 = \begin{bmatrix}133\\298\end{bmatrix} \qquad 298 / 133 = 2.241$$

$$\mathbf{x}_6 = \begin{bmatrix}463\\862\end{bmatrix} \qquad 862 / 463 = 1.862$$

$$\mathbf{x}_7 = \begin{bmatrix}1261\\2650\end{bmatrix} \qquad 2650 / 1261 = 2.102$$

Remember that eigenvectors are only unique up to a scalar multiple – it is the direction that counts. Even if we normalize the eigenvector, there are still two possibilities ($\hat{\mathbf{x}}$ and $-\hat{\mathbf{x}}$). Strictly speaking, we should not talk about 'the' dominant eigenvector, but 'a' dominant eigenvector, since there are infinitely many.

Note: we have not yet explained why **x** converges to the direction of the dominant eigenvector. This is discussed below.

As for the dominant eigenvalue, the final column of the table tracks the evolution of the 'magnification ratio', defined for convenience as the factor by which the largest (in this case the second) component of **x** is scaled

during an iteration[1]. As we work through the iterations, we see that the magnification ratio is approaching the dominant eigenvalue (= 3). This makes sense. We know that **A** scales its dominant eigenvector $\begin{bmatrix}1 & 2\end{bmatrix}^T$ by a factor of exactly 3, so as **x** approaches the direction $\begin{bmatrix}1 & 2\end{bmatrix}^T$ in the iterations of the power method, we expect **A** to scale it by a factor that gradually approaches 3. Note, however, that in any given iteration prior to convergence, the individual components of **x** are <u>not</u> scaled by exactly the same factor under the action of **A** (e.g. in the final row of the table, 2650 / 862 = 3.074, but 1261 / 463 = 2.724). If the components <u>were</u> all scaled by the same factor, this would imply that **x** was an eigenvector of **A**, and the iterations would have converged!

[1] Here the other component is of course magnified by a similar ratio, but in general it is advisable to monitor the magnification of the (absolutely) largest element. This ensures that we don't inadvertently monitor the noisy fluctuations of a component that is converging to zero.

The power method is fine for hand calculations on small matrices, but in practice the steady growth of the components of **x** cannot be allowed because it will eventually lead to floating-point overflow (or underflow, if the dominant eigenvalue happens to be less than 1). On my computer, I could make it to $\mathbf{x}_{645} = \begin{bmatrix} 3.3217 \times 10^{307} & 6.6434 \times 10^{307} \end{bmatrix}^T$ before overflow kicked in. Now 645 iterations may sound like plenty, and for this problem it is (note how the ratio of the ratio of the second component to the first component has reached 2 exactly), but in real problems the power method may be very slow to converge. We will find out why shortly, but for now what we need is a method for keeping the magnitude of **x** under control.

The solution is simple. At each iteration, before computing **Ax**, we first normalize the vector **x** by its ∞-norm (or some other convenient vector norm). This is the power method with scaling. It makes sure that as the iterations proceed, no component of **x** acquires an absolute value greater than 1. When the method is applied to the above example with the same starting vector $\mathbf{x}_0$, we get:

Iter.	**x**	**Ax**	Magn. ratio
1	$\begin{bmatrix}1\\1\end{bmatrix}\rightarrow\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}1\\4\end{bmatrix}$	4.000
2	$\begin{bmatrix}1\\4\end{bmatrix}\rightarrow\begin{bmatrix}0.25\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0.25\\1\end{bmatrix}=\begin{bmatrix}1.75\\2.5\end{bmatrix}$	2.500
3	$\begin{bmatrix}1.75\\2.5\end{bmatrix}\rightarrow\begin{bmatrix}0.7\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0.7\\1\end{bmatrix}=\begin{bmatrix}1.3\\3.4\end{bmatrix}$	3.400
4	$\begin{bmatrix}1.3\\3.4\end{bmatrix}\rightarrow\begin{bmatrix}0.382\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0.382\\1\end{bmatrix}=\begin{bmatrix}1.618\\2.765\end{bmatrix}$	2.765
5	$\begin{bmatrix}1.618\\2.765\end{bmatrix}\rightarrow\begin{bmatrix}0.585\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0.585\\1\end{bmatrix}=\begin{bmatrix}1.415\\3.170\end{bmatrix}$	3.170
6	$\begin{bmatrix}1.415\\3.170\end{bmatrix}\rightarrow\begin{bmatrix}0.446\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0.446\\1\end{bmatrix}=\begin{bmatrix}1.554\\2.893\end{bmatrix}$	2.893
7	$\begin{bmatrix}1.554\\2.893\end{bmatrix}\rightarrow\begin{bmatrix}0.537\\1\end{bmatrix}$	$\begin{bmatrix}-1 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0.537\\1\end{bmatrix}=\begin{bmatrix}1.463\\3.074\end{bmatrix}$	3.074

This involves more work, but is somewhat 'nicer'. The components of **x** remain small, and at each iteration the magnification ratio can be read off directly from the largest component of **Ax** (note that the magnification ratios are identical to those that appeared in the unscaled power method – can you see why?). Also, normalizing at each iteration makes it more obvious that **x** is converging to a particular direction: that of the dominant eigenvector $\begin{bmatrix}1 & 2\end{bmatrix}^T \equiv \begin{bmatrix}0.5 & 1\end{bmatrix}^T$. Other scaling strategies could have been implemented, e.g. normalizing **x** by its 2-norm to obtain a conventional unit vector. But this would have involved extra work to find the 2-norm, and we are trying to keep each iteration as cheap as possible.

We now need to address two important questions: why does the power method work, and what controls its rate of convergence? Consider an $n \times n$ matrix **A** with a dominant eigenvalue λ_1, such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n| \qquad (4.2)$$

Note that the leftmost inequality is strict. Since **A** is real and symmetric, we know that we can always find n corresponding eigenvectors

$$\mathbf{v}_1,\ \mathbf{v}_2,\ \mathbf{v}_3,\ \ldots,\ \mathbf{v}_n$$

that are linearly independent (in fact, as you may recall from P1, we can find n corresponding eigenvectors that are mutually orthogonal).

This means that any vector $\mathbf{x}_0$ that is used to start the power method can be expressed as a linear combination of the eigenvectors of $\mathbf{A}$:

$$\mathbf{x}_0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_n\mathbf{v}_n \qquad (4.3)$$

The first iteration gives

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{A}\mathbf{x}_0 \\ &= c_1\mathbf{A}\mathbf{v}_1 + c_2\mathbf{A}\mathbf{v}_2 + \ldots + c_n\mathbf{A}\mathbf{v}_n \\ &= c_1\lambda_1\mathbf{v}_1 + c_2\lambda_2\mathbf{v}_2 + \ldots + c_n\lambda_n\mathbf{v}_n\end{aligned}$$

The second iteration gives

$$\begin{aligned}\mathbf{x}_2 &= \mathbf{A}\mathbf{x}_1 \\ &= c_1\lambda_1\mathbf{A}\mathbf{v}_1 + c_2\lambda_2\mathbf{A}\mathbf{v}_2 + \ldots + c_n\lambda_n\mathbf{A}\mathbf{v}_n \\ &= c_1\lambda_1^2\mathbf{v}_1 + c_2\lambda_2^2\mathbf{v}_2 + \ldots + c_n\lambda_n^2\mathbf{v}_n\end{aligned}$$

After the *k*th iteration we will have

$$\mathbf{x}_k = c_1\lambda_1^k\mathbf{v}_1 + c_2\lambda_2^k\mathbf{v}_2 + \ldots + c_n\lambda_n^k\mathbf{v}_n \qquad (4.4)$$

Since the eigenvalue λ_1 has been assumed dominant, see eqn (4.2), it is clear that as k increases, the first term on the right-hand side of eqn (4.4) will begin to dominate all of the remaining terms. As this happens, $\mathbf{x}$ will gradually approach the direction of the dominant eigenvector $\mathbf{v}_1$ (and in the limit, become parallel to it).

Taking another view, we can write eqn (4.4) like this:

$$\mathbf{x}_k = \lambda_1^k \left[c_1 \mathbf{v}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \mathbf{v}_2 + \ldots + c_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \mathbf{v}_n \right] \qquad (4.5)$$

Since λ_1 is dominant, the fractions λ_2/λ_1, …, λ_n/λ_1 all have absolute values that are strictly less than 1, hence $(\lambda_2/\lambda_1)^k$, …, $(\lambda_n/\lambda_1)^k$ will all tend to zero as *k* increases. This again explains why the power method gradually 'turns' the vector **x** until it becomes parallel to the dominant eigenvector $\mathbf{v}_1$.

Furthermore, eqn (4.5) reveals that the rate of convergence of the power method will depend on the ratio of the two largest eigenvalues, λ_2/λ_1. If $|\lambda_2/\lambda_1|$ is small, e.g. 0.1, the influence of the 'second most dominant' eigenvector $\mathbf{v}_2$ (and all subsequent eigenvectors) will quickly tail off as *k* increases; convergence will be very fast. If $|\lambda_2/\lambda_1|$ is only a little smaller than 1, e.g. 0.95, then convergence will be much slower. In practice this difficulty can be partially overcome by using the shifted power method, which temporarily offsets the whole spectrum of eigenvalues in order to make the ratio $|\lambda_2/\lambda_1|$ smaller.

A final important observation is that the power method can fail. If we are unlucky enough to choose a starting vector $\mathbf{x}_0$ that has no component in the direction of the dominant eigenvector $\mathbf{v}_1$ (i.e. $\mathbf{x}_o$ is perpendicular to $\mathbf{v}_1$) then the coefficient c_1 in the initial linear combination, eqn (4.3), will be zero. If $c_1 = 0$ in eqns (4.4) and (4.5), it is clear that the iterations of the power method will cause $\mathbf{x}$ to approach the direction of $\mathbf{v}_2$, the 'second most dominant' eigenvector, rather than $\mathbf{v}_1$. Also, the 'magnification ratios' will converge to λ_2 rather than λ_1. A good precaution in practice is to try two different starting vectors $\mathbf{x}_0$ that are not parallel.

4.3 Rayleigh method

There is a very effective way to improve the estimate of the dominant eigenvalue λ_1 at each iteration of the power method. So far we have effectively been making a 'direct' estimate of λ_1 by monitoring the factor by which the (absolutely) largest component of **x** is magnified under the action of **A**. As **x** turns towards the dominant eigenvector $\mathbf{v}_1$, this ratio approaches λ_1.

Another approach is to focus on the fundamental eigenvector/eigenvalue equation

$$\mathbf{Ax} = \lambda\mathbf{x} \qquad (4.1 \text{ bis})$$

We rearrange this to read

$$\mathbf{Ax} - \lambda\mathbf{x} = \mathbf{0} \qquad (4.6)$$

noting that the **0** on the right-hand side is the zero vector.

When the power method eventually converges, eqn (4.6) will be satisfied exactly (with $\mathbf{x} = \mathbf{v}_1$ and $\lambda = \lambda_1$). Prior to convergence, however, **x** is only an approximation of the dominant eigenvector, so the best we can hope for is to find an approximation of the dominant eigenvalue (μ, say) such that

$$\mathbf{Ax} - \mu\mathbf{x} \approx \mathbf{0}$$

To do this rigorously, we should endeavour to choose μ in such a way that some norm of the 'residual' vector

$\mathbf{Ax} - \mu\mathbf{x}$ is minimized. It is most natural to do this minimization with respect to the ordinary Euclidean norm (i.e. the 2-norm), in which case the optimum value of μ turns out to be the Rayleigh quotient.

Let **A** be a real symmetric matrix with a dominant eigenvalue λ_1, and let **x** be a given approximation of the dominant eigenvector $\mathbf{v}_1$. The Rayleigh quotient, defined as

$$r(\mathbf{x}) = \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$$

provides an approximation of λ_1 that is optimal in a least-squares sense, i.e. the 2-norm of the residual vector

$$\mathbf{Ax} - \mu\mathbf{x}$$

is minimized by taking $\mu = r(\mathbf{x})$.

This is an important result, and you are asked to prove it in Q7 of the tutorial sheet. A hint is provided to get you going, and here is another one: use the fact that

$$(\mathbf{Ax} - \mu\mathbf{x})^T = (\mathbf{x}^T\mathbf{A}^T - \mu\mathbf{x}^T)$$

then expand out the product with the other bracketed term.

If we apply the power method as before, but additionally evaluate the Rayleigh quotient at each iteration, we have the Rayleigh method for finding the dominant eigenvalue (and the corresponding eigenvector). We will not go into the details here, but it can be shown that the power method provides only linear convergence to the dominant eigenvalue, whereas the Rayleigh method provides quadratic convergence (so it always gives better performance). With a more sophisticated scheme known as the shifted inverse power method, it is possible to obtain cubic convergence. Note however that the use of the Rayleigh quotient does not improve the rate of convergence of $\mathbf{x}$ to the dominant eigenvector $\mathbf{v}_1$; this remains the same as in the standard power method.
It is worth pointing out that the Rayleigh quotient can also be expressed in terms of dot products:

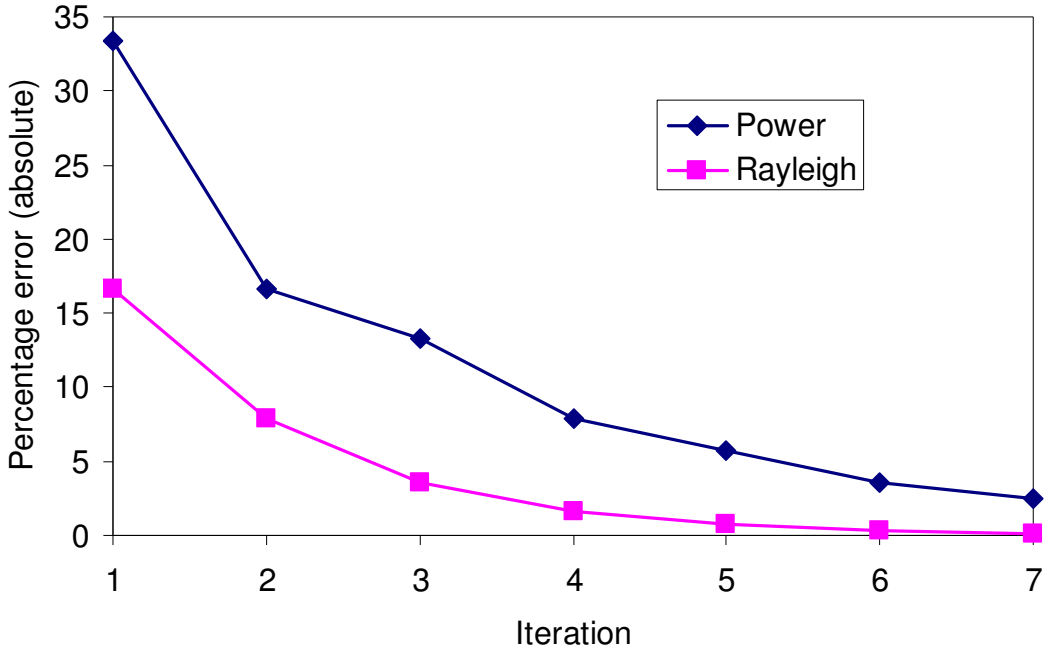
$$r(\mathbf{x}) = \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \frac{\mathbf{x} \bullet (\mathbf{A}\mathbf{x})}{\mathbf{x} \bullet \mathbf{x}}$$

The normalization by $\mathbf{x} \bullet \mathbf{x}$ ensures that we obtain a correct estimate of the eigenvalue even if $\mathbf{x}$ is not a unit vector.

To show the effectiveness of using the Rayleigh quotient, here is the first example (no scaling) again, with an extra column appended:

Iter.	$\mathbf{x}$	$\mathbf{Ax}$	Magn. ratio	$r(\mathbf{x}) = \dfrac{\mathbf{x} \bullet (\mathbf{Ax})}{\mathbf{x} \bullet \mathbf{x}}$
1	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$	$\dfrac{4}{1} = 4.000$	$\dfrac{5}{2} = 2.500$
2	$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	$\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 7 \\ 10 \end{bmatrix}$	$\dfrac{10}{4} = 2.500$	$\dfrac{47}{17} = 2.765$
3	$\begin{bmatrix} 7 \\ 10 \end{bmatrix}$	$\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 7 \\ 10 \end{bmatrix} = \begin{bmatrix} 13 \\ 34 \end{bmatrix}$	$\dfrac{34}{10} = 3.400$	$\dfrac{431}{149} = 2.893$
4	$\begin{bmatrix} 13 \\ 34 \end{bmatrix}$	$\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 13 \\ 34 \end{bmatrix} = \begin{bmatrix} 55 \\ 94 \end{bmatrix}$	$\dfrac{94}{34} = 2.765$	$\dfrac{3911}{1325} = 2.952$
5	$\begin{bmatrix} 55 \\ 94 \end{bmatrix}$	$\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 55 \\ 94 \end{bmatrix} = \begin{bmatrix} 133 \\ 298 \end{bmatrix}$	$\dfrac{298}{94} = 3.170$	$\dfrac{35327}{11861} = 2.978$
6	$\begin{bmatrix} 133 \\ 298 \end{bmatrix}$	$\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 133 \\ 298 \end{bmatrix} = \begin{bmatrix} 463 \\ 862 \end{bmatrix}$	$\dfrac{862}{298} = 2.893$	$\dfrac{318455}{106493} = 2.990$
7	$\begin{bmatrix} 463 \\ 862 \end{bmatrix}$	$\begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 463 \\ 862 \end{bmatrix} = \begin{bmatrix} 1261 \\ 2650 \end{bmatrix}$	$\dfrac{2650}{862} = 3.074$	$\dfrac{2868143}{957413} = 2.996$

As expected, the Rayleigh quotient provides a far superior rate of convergence to the dominant eigenvalue, $\lambda_1 = 3$.


35
30
25
20
15
10
5
0
Percentage error (absolute)
1
2
3
4
5
6
7
Iteration
Power
Rayleigh

4.4 Deflation

It is a theorem (stated, but not proven, in P1) that a real symmetric matrix **A** can always be factored as

$$\mathbf{A} = \mathbf{M}\mathbf{D}\mathbf{M}^T \qquad (4.7)$$

where

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

is a matrix containing the eigenvalues of **A** on its diagonal, and

$$\mathbf{M} = \left[\hat{\mathbf{v}}_1 \mid \hat{\mathbf{v}}_2 \mid \cdots \mid \hat{\mathbf{v}}_n \right]$$

is a matrix containing an orthonormal set of corresponding eigenvectors, arranged columnwise (note 'ortho' = mutually perpendicular, 'normal' = unit vectors). The matrix **M** is orthogonal, and has rank *n*.

We can express eqn (4.7) in an alternative form that will be more useful for the derivations to come:

$$\begin{aligned}
\mathbf{A} &= \mathbf{MDM}^T \\
&= \left[\hat{\mathbf{v}}_1 \mid \hat{\mathbf{v}}_2 \mid \cdots \mid \hat{\mathbf{v}}_n\right]\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}\left[\hat{\mathbf{v}}_1 \mid \hat{\mathbf{v}}_2 \mid \cdots \mid \hat{\mathbf{v}}_n\right]^T \\
&= \left[\lambda_1\hat{\mathbf{v}}_1 \mid \lambda_2\hat{\mathbf{v}}_2 \mid \cdots \mid \lambda_n\hat{\mathbf{v}}_n\right]\begin{bmatrix} \hat{\mathbf{v}}_1{}^T \\ \hline \hat{\mathbf{v}}_2{}^T \\ \hline \vdots \\ \hline \hat{\mathbf{v}}_n{}^T \end{bmatrix} \\
&= \lambda_1\hat{\mathbf{v}}_1\hat{\mathbf{v}}_1{}^T + \lambda_2\hat{\mathbf{v}}_2\hat{\mathbf{v}}_2{}^T + \ldots + \lambda_n\hat{\mathbf{v}}_n\hat{\mathbf{v}}_n{}^T
\end{aligned}$$

This is known as the spectral decomposition or eigendecomposition of **A**. It shows that a real symmetric $n \times n$ matrix can be expressed as the sum of n component matrices, each of which is formed by taking the outer product of a normalized eigenvector with itself, then applying a 'weighting factor' equal to the relevant eigenvalue. Note that all of the component matrices are themselves symmetric (and, incidentally, have rank 1).

We need to establish one more important result. Suppose we have a real symmetric $n \times n$ matrix **A** and form the spectral decomposition

$$\mathbf{A} = \lambda_1 \hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1^T + \lambda_2 \hat{\mathbf{v}}_2 \hat{\mathbf{v}}_2^T + \ldots + \lambda_n \hat{\mathbf{v}}_n \hat{\mathbf{v}}_n^T$$

If we take the first component matrix $\lambda_1 \hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1^T$ to the left-hand side, we obtain a new matrix

$$\begin{aligned} \mathbf{B} &= \mathbf{A} - \lambda_1 \hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1^T \\ &= 0\hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1^T + \lambda_2 \hat{\mathbf{v}}_2 \hat{\mathbf{v}}_2^T + \ldots + \lambda_n \hat{\mathbf{v}}_n \hat{\mathbf{v}}_n^T \end{aligned} \qquad (4.8)$$

From the second line, we expect the 'deflated' matrix **B** to have the same eigenvectors and eigenvalues as **A**, but with zero replacing λ_1 in the list of eigenvalues. That is, we expect $\hat{\mathbf{v}}_1$, $\hat{\mathbf{v}}_2$, …, $\hat{\mathbf{v}}_n$ to be eigenvectors of **B** corresponding to eigenvalues 0, λ_2, … λ_n respectively.

This is not too difficult to prove. If we postmultiply both sides of eqn (4.8) by $\hat{\mathbf{v}}_1$ we get

$$\mathbf{B}\hat{\mathbf{v}}_1 = \mathbf{A}\hat{\mathbf{v}}_1 - \lambda_1 \left(\hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1^T \right) \hat{\mathbf{v}}_1 = \lambda_1 \hat{\mathbf{v}}_1 - \lambda_1 \hat{\mathbf{v}}_1 \left(\hat{\mathbf{v}}_1^T \hat{\mathbf{v}}_1 \right) = \lambda_1 \hat{\mathbf{v}}_1 - \lambda_1 \hat{\mathbf{v}}_1 (1) = 0\hat{\mathbf{v}}_1$$

noting that $\hat{\mathbf{v}}_1^T \hat{\mathbf{v}}_1 = \hat{\mathbf{v}}_1 \bullet \hat{\mathbf{v}}_1 = 1$. This shows that $\hat{\mathbf{v}}_1$ is an eigenvector of **B** with eigenvalue 0 (whereas it was an eigenvector **A** with eigenvalue λ_1). If we postmultiply both sides of eqn (4.8) by $\hat{\mathbf{v}}_2$ we get

$$\mathbf{B}\hat{\mathbf{v}}_2 = \mathbf{A}\hat{\mathbf{v}}_2 - \lambda_1 \left(\hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1^T \right) \hat{\mathbf{v}}_2 = \lambda_2 \hat{\mathbf{v}}_2 - \lambda_1 \hat{\mathbf{v}}_1 \left(\hat{\mathbf{v}}_1^T \hat{\mathbf{v}}_2 \right) = \lambda_2 \hat{\mathbf{v}}_2 - \lambda_1 \hat{\mathbf{v}}_1 (0) = \lambda_2 \hat{\mathbf{v}}_2$$

where we have used the fact that $\hat{\mathbf{v}}_i^T \hat{\mathbf{v}}_j = \hat{\mathbf{v}}_i \bullet \hat{\mathbf{v}}_j = 0$ if $i \neq j$ (recall that the eigenvectors are mutually perpendicular). This shows that $\hat{\mathbf{v}}_2$ is an eigenvector of **B** with eigenvalue λ_2 (just as it was an eigenvector of **A** with eigenvalue λ_2). In the same way, it can be shown that all of the remaining eigenvectors of **A** ($\hat{\mathbf{v}}_3, \hat{\mathbf{v}}_4, \ldots, \hat{\mathbf{v}}_n$) are still eigenvectors of **B**, with unchanged eigenvalues ($\lambda_3, \lambda_4, \ldots, \lambda_n$).

Finally we are ready to outline the technique of deflation. Suppose we have a real symmetric $n \times n$ matrix **A** with eigenvalues

$$|\lambda_1| > |\lambda_2| > |\lambda_3| \geq |\lambda_4| \geq \ldots \geq |\lambda_n|$$

Note that the leftmost two inequalities are strict. Let us presume that the dominant eigenvalue λ_1 and dominant eigenvector $\mathbf{v}_1$ have been found using the power method or Rayleigh method. The dominant eigenvector $\mathbf{v}_1$ can be normalized to give a unit dominant eigenvector $\hat{\mathbf{v}}_1$ (this step is crucial!) and the 'deflated' matrix

$$\mathbf{B} = \mathbf{A} - \lambda_1 \hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1^T$$

can be formed. As we know from above, the eigenvalues of **B** are the same as those of **A**, but with 0 replacing λ_1:

$$|\lambda_2| > |\lambda_3| \geq |\lambda_4| \geq \ldots \geq |\lambda_n| \geq 0$$

The power method or Rayleigh method can now be applied to **B**. This will give the dominant eigenvalue and eigenvector of **B**, namely λ_2 and $\mathbf{v}_2$, which are the 'second most dominant' eigenvalue and eigenvector of the original matrix **A**. The deflation process can, in principle, be repeated by forming

$$\mathbf{C} = \mathbf{B} - \lambda_2 \hat{\mathbf{v}}_2 \hat{\mathbf{v}}_2^T$$

and applying the power method or Rayleigh method to **C**, and so on, until all *n* eigenvalues and eigenvectors of the original matrix **A** have been found. In practice, however, numerical roundoff errors build up, and the accuracy of the computed results starts to deteriorate. The deflation technique is only suitable for obtaining the largest few eigenvalues of **A** and their corresponding eigenvectors.

If the smallest few eigenvalues are required (e.g. the fundamental natural frequency and first few harmonics of a dynamic system) then it is much more efficient to use a technique known as the inverse power method. This method computes the (absolutely) smallest eigenvalue of **A** by finding the dominant eigenvalue of $\mathbf{A}^{-1}$, but does so without requiring the inverse to be formed explicitly. If all the eigenvalues of a matrix are required, the method of choice is the **QR** algorithm. These advanced methods are beyond the scope of the present course.

We will finish this section by demonstrating the application of deflation to our example problem involving the matrix

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

We assume that the power method or Rayleigh method has been applied to **A** and continued to convergence, such that its dominant eigenvalue and eigenvector are known exactly:

$$\lambda_1 = 3\,, \qquad \mathbf{v}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

We then normalize $\mathbf{v}_1$ to make it a unit vector (crucial step!)

$$\hat{\mathbf{v}}_1 = \frac{1}{\sqrt{5}} \times \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

and form the ‘deflated’ matrix

$$\begin{aligned}\mathbf{B} &= \mathbf{A} - \lambda_1 \hat{\mathbf{v}}_1 \hat{\mathbf{v}}_1{}^T \\ &= \begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix} - 3 \times \frac{1}{5} \times \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix} - 0.6 \times \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \\ &= \begin{bmatrix} -1.6 & 0.8 \\ 0.8 & -0.4 \end{bmatrix}\end{aligned}$$

The eigenvalues and eigenvectors of **B** are easy to find:

$$\begin{aligned}&\det \begin{bmatrix} -1.6 - \lambda & 0.8 \\ 0.8 & -0.4 - \lambda \end{bmatrix} = 0 \\ &(1.6 + \lambda)(0.4 + \lambda) - 0.64 = 0 \\ &\lambda^2 + 2\lambda = 0 \\ &\lambda(\lambda + 2) = 0 \\ &\lambda = 0,\ -2\end{aligned}$$

This is exactly as expected. We happen to know (from Section 4.2) that the eigenvalues of **A** are 3 and –2, so the dominant eigenvalue (i.e. 3) has been correctly ‘altered’ to zero in the list of eigenvalues of the deflated matrix **B**.

The dominant eigenvalue of **B** is $\lambda = -2$, and the corresponding eigenvector can be found in the usual way:

$$\begin{bmatrix} -1.6+2 & 0.8 \\ 0.8 & -0.4+2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.4 & 0.8 \\ 0.8 & 1.6 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which gives the (non-normalized) dominant eigenvector of **B** as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

We can therefore deduce that the 'second most dominant' eigenvalue and eigenvector of the original matrix **A** must be

$$\lambda_2 = -2, \qquad \mathbf{v}_2 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

This can readily be verified, since

$$\mathbf{A}\mathbf{v}_2 = \begin{bmatrix} -1 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \end{bmatrix} = -2 \times \begin{bmatrix} -2 \\ 1 \end{bmatrix} = \lambda_2 \mathbf{v}_2$$

In practice, of course, only the dominant eigenvalue and eigenvector of **A** would be known following the initial application of the power method or Rayleigh method to **A**. Also, the dominant eigenvalue and eigenvector of the deflated matrix **B** would need to be found using the power method or Rayleigh method, rather than analytically. Incidentally, if we had done this, we would have seen the 'magnification ratio' changing sign at each iteration. This simply indicates that the dominant eigenvalue, to which the method is converging, is negative (in this case –2).

Example (A1 1997 Q9)

(a) Prove, in the case of a symmetric matrix, that the power method converges to the dominant eigenvalue and its associated eigenvector.

(b) The power method is applied to a matrix $\mathbf{A}$ which has a dominant eigenvalue of λ. If the starting vector is denoted by $\mathbf{x}_0$ and the result of the nth iteration of the method by $\mathbf{x}_n$, show that $\mathbf{x}_{n+1} - \lambda\mathbf{x}_n$ is an estimate of the next most dominant eigenvector of $\mathbf{A}$.

(c) Use the power method to determine the dominant eigenvalue and normalized eigenvector for the matrix

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

Start from the trial vector $\mathbf{x}_0 = \begin{bmatrix} 1 & -1 & 0 \end{bmatrix}^T$ and iterate 5 times working to four significant figures in the vector.

(d) Estimate the next most dominant eigenvector.