A

Appendix

A.1 Operations with sparse matrices

appendix:sparse

In Chapter 2 we have seen that a sparse graph is represented by a sparse matrix. One of the most common operations with matrices is the matrix-vector multiplication, and this is where the sparse representation of the matrix allows a dramatic cut in the complexity.

There are several representations of sparse matrices that are commonly used in the literature. Here we consider the two most common, we show the relative advantages of each other, and how to switch from one representation to the other.

A common way to represent a sparse matrix is to assign three vectors, whose length is equal to the number of non zero elements of the matrix. Such vectors store the values of the nonzero elements of the matrix, and their row and column indices. This representation was introduced in chapter 2.

With this representation it is very simple to perform the matrix vector product. Let $A = (\mathbf{s}, \mathbf{i}, \mathbf{j})$ be a sparse $N \times N$ matrix , and let $x \in \mathbb{R}^N$ be a column vector. Let M denote the number of non zero elements of matrix A, i.e. M is the length of vectors $\mathbf{s}, \mathbf{i}, \mathbf{j}$.

The components of the product

$$y = Ax$$

are computed by the algorithm

```
for $k = 1 : M$
    $y(i(k)) = y(i(k)) + s(k) * x(i(k));$        (A.1)
end
```

eqA1:multIJ

Here we use a Matlab notation, so that most of the algorithms presented can be immediately implemented. However, such algorithms are not supposed to be implemented in Matlab the way they are written, since Matlab is very inefficient when used with scalar operations. A C version of the algorithms, which is much more efficient, can be downloaded from the web @@@Inserire il riferimento@@@.

The number of floating point operations (multiplications) required by the above algorithm is M. Note also that the algorithm can be written in a form which is more compact than the usual matrix-vector multiplication, which requires two nested cycles. Note also that before starting the cycle, one has to be sure that the vector y is zero. This can be done by explicitly setting all components of y to zero, before starting cycle A.1

When using the SLAP row format, the matrix information is stored in two three vectors, $\mathbf{s}, \mathbf{r}, \mathbf{j}$, with $\mathbf{s}$ a vector with M components storing the non zero values of the matrix, $\mathbf{j}$ a vector with M components storing the column index of each non zero component of the matrix, ordered by index, and $\mathbf{r}$ is a vector of dimension $N+1$, such that $r_i, i = 1, \ldots, N$ is the index of the first non zero element of the i=th row, and r_{N+1} is equal to $M+1$. In this way all non zero elements of the i-th row of the matrix will be stored in positions $k = r_i, \ldots, r_{i+1} - 1$.

Note that this representation is equivalent to assigning an adjacency list to each node. For this reason it is sometimes referred as adjacency-list representation **??**.

If a matrix $A = [\mathbf{s}, \mathbf{r}, \mathbf{j}]$ is stored in a SLAP row format, and $x \in \mathbb{R}^N$ is a column vector, then the matrix-vector product $y = Ax$ is obtained by the algorithm

```
for $i = 1 : N$
    for $k = r(i_1) : r(i_1 + 1) - 1$
        $y(k) = y(k) + s(k) * x(i_1);$          (A.2)
    end
end
```

eqA1:multRL

Sometimes it is useful to switch from one sparse matrix representation to the other. Here we show how to switch from the ij-representation of the matrix to the SLAP-row format, and vice versa. It is trivial to switch from the compressed row format to the ij-format, in fact it is enough to define a vector $\mathbf{i}$ as

```
for $j = 1 : N$
    for $k = r(j) : r(j + 1) - 1$
        $i(k) = j;$          (A.3)
    end
end
```

eqA1:computei

The construction of the SLAP representation from the ij representation is slightly more complicated, mainly because i and j vectors are stored without any specific order. One possibility to switch from ij- to SLAP- format is to sort vector i, and then create a vector r by scanning the sorted vector i. However, there ia a simpler and faster way to do it, which is shown in Algorithm illustrated in Table A.1. The method is faster that standard sorting, first because we know *a priori* what the possible values of i are, and because the j values will not be sorted. Given vector $\mathbf{i} \in \mathbb{N}^M$, the algorithm produces a vector $\mathbf{r} \in \mathbb{N}^{N+1}$ and a vector $\mathbf{L}$ such that the vector $\mathbf{j}$ in the SLAP format will be given by $j_{\mathrm{SLAP}}(k) = j(L(k)),\ k = 1, \ldots, M$.

The algorithm works as follows: First the number of elements in each row is computed and stored in $r(i+1), i = 1, \ldots, N$. From this information, the vector r is computed as $r(1) = 1$, $r(i+1) = r(i+1) + r(i), / > i = 1, \ldots, N$. Finally, vector i is scanned again. Once the value $\bar{i}$ appears, its index is stored in vector $\mathbf{L}$.

```
% Set $r = 0$ and $p = 0$
for $k = 1 : N$
    $r(k+1) = 0;$
    $p(k) = 0;$
end
% Count the number of elements for each row
for $k = 1 : M$
    $r(i(k)+1) = r(i(k)+1) + 1;$
end
% Compute vector r
$r(1) = 1;$
for $ii = 1 : N$
    $r(ii+1) = r(ii) + r(ii+1);$
end
% Compute vector L
for $k = 1 : M$
    $ii = i(k);$
    $L(r(ii) + p(ii)) = k;$
    $p(ii) = p(ii) + 1;$
end
```

Table A.1. Conversion from ij- to SLAP row format. Vectors r and L are created such that the new vector $j_{\mathrm{SLAP}}(k) = j(L(k)), k = 1, \ldots, M$. tabA1:ij2rl

The procedure can be easily followed by looking at figure A.1.

A.2 Eigenvalues and Eigenvectors of a matrix

appendix:eigenvalues

Let $A \in \mathbb{R}^{n \times n}$ be a real square matrix. Suppose we want to find a vector whose direction remains unchanged under the action of the matrix, i.e. we look for a non trivial vector $\mathbf{u} \in \mathbb{C}^n$ proportional to $A\mathbf{u}$. Such relation means that we seek a vector such that

$$A\mathbf{u} = \lambda \mathbf{u} \tag{A.4}$$

eqA2.eig1

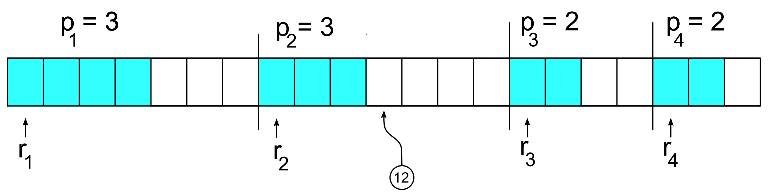

Fig. A.1. Creation of the L vector. Suppose that the first 11 entries of vector **i** contain four times $i = 1$, three times $i = 2$, two times $i = 3$, and two times $i = 4$, and suppose that during the generation of vector **r**, we counted seven times $i = 1$, seven times $i = 2$, four times $i = 3$ and three times $i = 4$. Then before the insertion of the 12-th value, we would have the situation shown in the figure. Suppose that the 12-th value of i is 2. Then we insert such value as indicated in the table, which means we set $L(r(2) + p(2))$ to 12, and increase $p(2)$ by one. figA1:ij2rl

for some values of λ. If λ and $\mathbf{u}$ exist, they are called, respectively, *eigenvalue* and *right eigenvector* (or simply *eigenvector*) of matrix A. Note that system A.4 may be written in the form

$$(\lambda I - A)\mathbf{u} = 0 \tag{A.5}$$

eqA2.eig2

where I denotes the $(n \times n)$ identity matrix. Because the system is homogeneous, if $\det(\lambda I - A) \neq 0$ it admits the only solution $\mathbf{u} = 0$. If we look for a non trivial solution, than we have to impose

$$p(\lambda) = 0 \tag{A.6}$$

eqA2.characteristic

where $p(\lambda) \equiv \det(\lambda I - A)$ is called *characteristic polynomial* of matrix A. Therefore the eigenvalues of A are the roots of the polynomial p. Observe that p is a polynomial of degree exactly n. The fundamental theorem of algebra states that $p(\lambda)$ has n roots (in general complex), if each root is counted with the proper multiplicity. Once the eigenvalues $\lambda_1, \ldots, \lambda_n$ have been found, the eigenvectors can be computed by solving the linear homogeneous systems

$$(\lambda_\alpha I - A)\,\mathbf{u}^\alpha = 0, \quad \alpha = 1, \ldots, n \tag{A.7}$$

eqA2.eig3

The set of eigenvalues $\lambda(A) = \{\lambda_1, \ldots, \lambda_n\}$ is called the *spectrum* of A. The largest eigenvalue (in absolute value) is called the *spectral radius* of the matrix: $\rho(A) = max_{1\leq\alpha\leq n}|\lambda_\alpha|$.

Similarly, it is possible to define *left eigenvectors* $\mathbf{v}_i \in \mathbb{C}^n$, such that

$$\mathbf{v}^{\alpha\dagger} A = \lambda_\alpha \mathbf{v}_{\alpha\dagger}. \tag{A.8}$$

eqA2.lefteig

Here the symbol $\dagger$ denotes the adjoint of a vector: given a column vector $\mathbf{x} \in \mathbb{C}^n$, its adjoint $\mathbf{y} = \mathbf{x}^\dagger$ is a row vector such that $y_i = \bar{x}_i, i = 1, \ldots, n$, where $\forall x \in \mathbb{C}$, $\bar{x}$ denotes the complex conjugate of x.

First let us show the following orthogonality property.

Proposition A.1. *Left and right eigenvectors corresponding to different eigenvalues are orthogonal, i.e.* $\mathbf{v}^{\alpha\dagger}u^{\beta} = 0$ *if* $\lambda_\alpha \neq \lambda_\beta$.

Real symmetric matrices (and, more generally, Hermitian matrices) have special properties concerning eigenvalues. First, let us observe that the eigenvalues of real symmetric matrices (or in general of complex Hermitian matrices) are real. In fact, taking the adjoint of the relation

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

one as

$$\mathbf{u}^\dagger A^\dagger = \overline{\lambda}\mathbf{u}^\dagger \tag{A.9}$$

eqA2.transpose

left multiplying the first relation by $\mathbf{u}^\dagger$, right multiplying the second relation by $\mathbf{u}$ and subtracting each other, one obtains

$$\mathbf{u}^\dagger(A - A^\dagger)\mathbf{u} = (\lambda - \overline{\lambda})\mathbf{u}^\dagger\mathbf{u}$$

Let $\|\mathbf{u}\| \equiv (\sum_{i=1}^{N} |u_i|^2)^{1/2}$ denote the *Euclidean norm* of a vector $\mathbf{u}$. Since $\mathbf{u}^\dagger\mathbf{u} = \|\mathbf{u}\|^2 > 0$, if $A = A^\dagger$ it follows $\lambda = \overline{\lambda}$, i.e. $\lambda \in \mathbb{R}$.

Then let us observe that if a matrix is symmetric (or Hermitian), then left and right eigenvectors coincide. This is a consequence of Eq.(A.9), since $\lambda = \overline{\lambda}$.

Because the eigenvalues of a real symmetric matrix are real [8], it is possible to choose real eigenvectors (since the eigenvectors are computed from a real linear homogeneous system).

Because of the orthogonality property of the eigenvectors, it appears natural to normalize them in such a way that their Eucledian norm is one, i.e. we impose that

$$\mathbf{u}^{\alpha\top}\mathbf{u}^{\alpha} = 1, \alpha = 1, \dots, n$$

Because of the orthogonality condition, this implies that if the eigenvalues are all distinct, i.e. if $\lambda_\alpha \neq \lambda_\beta \forall \alpha \neq \beta, \alpha, \beta = 1, \dots, n$

$$\mathbf{u}^{\alpha\top}\mathbf{u}^{\beta} = \delta_{\alpha\beta}, \alpha, \beta = 1, \dots, n$$

In a matrix form this can be written as $U^\top U = I$, where we denote by $U = (\mathbf{u}^1, \dots, \mathbf{u}^n)$ a square matrix formed by the eigenvectors of matrix A.

This relation tells us that the eigenvectors are independent (since $\det(U) \neq 0$), and, furthermore, that the transpose of matrix U is equal to its inverse, i.e. that $U^{-1} = U^\top$.

If the matrix is not symmetric, then it is not guaranteed that the eigenvalues are real. However, it may be shown that, if the eigenvalues are all distinct, i.e. if

$$\lambda^\alpha \neq \lambda^\beta, \quad \alpha \neq \beta, \quad \alpha, \beta = 1, \dots, n$$

we have that the n eigenvectors corresponding to the n eigenvalues are independent. This means that the set $\{\mathbf{u}^\alpha, \alpha = 1 \dots, n\}$ forms a basis of $\mathbb{C}^n$ (see, for example, [8], chapter VIII).

We now prove that if a matrix has all distinct eigenvalues then the eigenvectors form a basis of $\mathbb{C}^n$. This is in fact a corollary of the following

Theorem A.1. *Let $\lambda_1, \lambda_2, \ldots, \lambda_k$ denote $k \leq n$ distinct eigenvalues of matrix $A \in \mathbb{R}^{n\times n}$, and let us denote by $\mathbf{u}^1, \ldots, \mathbf{u}^k$ the corresponding eigenvectors. If $\lambda_i \neq \lambda_j$, $\forall i \neq j$, $i, j = 1, \ldots, k$, then the k eigenvectors $u^1, \ldots, u^k$ are linearly independent.*

For a proof of the theorem see, for example, [8]. An immediate consequence of the theorem is that if $k = n$ then the eigenvectors form a basis. Observe that the relation

$$A\mathbf{u}^\alpha = \lambda^\alpha \mathbf{u}^\alpha, \quad \alpha = 1, \ldots, n$$

can be written, using matrix notation, as

$$AU = U\Lambda$$

where $U = (\mathbf{u}^1, \ldots, \mathbf{u}^n)$ is the matrix whose columns are the eigenvectors, while $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix containing the eigenvalues. If the eigenvectors form a basis of $\mathbb{R}^n$, then $\det(U) \neq 0$, and one can write

$$\Lambda = U^{-1}AU.$$

In this case we say that the matrix is diagonalizable.

If the eigenvalues are not all distinct, then it may or may not still be possible to find a basis of eigenvectors, depending on the structure of the matrix itself.

Suppose that the distinct eigenvalues are $\lambda_1, \ldots, \lambda_m$, where the eigenvalue λ_i has multiplicity m_i, so that $\sum_{i=1}^m m_i = n$. Then, since the eigenvectors corresponding to different eigenvalues are linearly independent, and since there could be no more that m_i independent eigenvectors corresponding to a given eigenvalue λ_i, a necessary and sufficient condition for diagonalizability is that for each eigenvalue λ_i there are exactly m_i linearly independent eigenvectors. If this condition is satisfied, than the matrix is still diagonalizable. If this is not the case, then the matrix cannot be reduced to a diagonal matrix by a similarity transformation. It can be shown, however, that in this case the matrix can be reduced in the so called canonical Jordan form (see, for example, [8], chapter 10).

If the matrix is real symmetric or Hermitian, it is always possible to find a basis of eigenvectors [8].

A.3 Eigenvalue and eigenvector computation

appendix:powermethod

The problem of finding eigenvalues and eigenvectors of a matrix is a classical problem of numerical analysis, which is very important for the numerous applications in science. Several methods have been designed to solve this problem.

All methods are based on iterations, since the eigenvalue problem is equivalent to the problem of finding the root of a polynomial, and there no direct methods that can solve algebraic equations of degree greater than four.

We can distinguish between methods that are suited to compute one or few eigenvalues (and eigenvectors) and methods that will provide all eigenvalues and eigenvectors. Another distinction depends on whether the matrix is symmetric or not.

A.3.1 The power method

The most common methods for finding the largest eigenvalue and eigenvector of a matrix are the power method, and the Lanczos methods. Although the methods based on the latter approach are probably the most effective ones for the computation in the case of a sparse matrix, their description is beyond the scope of the the book, and we address the interested reader to the book by Golub and Van Loan, which is an excellend reference book for numerical linear algebra [Golub and Van Loan, second edition, chapters 7–9]. In the following we describe in detail the power method. An implementation of the method will be found at the web site of the book. Let $A \in \mathbb{R}^{N\times N}$ be a real square matrix. Let us denote by $\lambda_1, \ldots, \lambda_N$ its eigenvalues, in decreasing order of absolute value, and assume λ_1 is simple, i.e.

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_N|. \tag{A.10}$$

eqA1:condition

Let us assume, furthermore, that A admits a basis of eigenvectors, and let us denote them by $u_i : A\mathbf{u}^i = \lambda_i \mathbf{u}^i, i = 1\ldots, N$. This assumption is not strictly necessary, as we shall see by some examples later, but it will simplify the description of the method.

Let us generate a sequence of column vectors x_n as follows: x_0 is an arbitrary non zero vector, for example $x_0 = (1, \ldots, 1)^T$. Then we generate a sequence of vectors according to the rule

$$\mathbf{x}^{n+1} = A\mathbf{x}^n.$$

We claim that the vector $\mathbf{x}^n$ will tend to be aligned to the eigenvector u^1 and that $\mathbf{x}^{n+1}$ will be approximately proportional to $\mathbf{x}^n$, with proportionality constant equal to the first eigenvalue, i.e. that $\mathbf{x}^{n+1} \approx \lambda^1 \mathbf{x}^n$.

Let us express the vector $\mathbf{x}^0$ on the basis of the eigenvectors of A

$$\mathbf{x}^0 = \sum_{i=1}^{N} \alpha_i u^i.$$

Then we have:

$$\mathbf{x}^1 = A\mathbf{x}^0 = \sum_{i=1}^{N} \alpha_i A u^i = \sum_{i=1}^{N} \alpha_i \lambda_i u^i,$$

and, in general,

$$\mathbf{x}^n = A\mathbf{x}^{n-1} = \sum_{i=1}^{N} \alpha_i \lambda_i^n u^i = \lambda_1^n \left(\alpha_1 u^1 + \sum_{i=2}^{N} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^n u^i \right) \tag{A.11}$$

$\boxed{\texttt{eqA1:xn}}$

It is clear that, if $\alpha_1 \neq 0$, which we assume true, then the first component will be the dominant one, since $(\lambda_i/\lambda_1) \to 0$ as $n \to \infty$, and therefore the vector $\mathbf{x}^n$ will tend to be proportional to u^1. To make this assertion more formal, let us take the ratio between the generic component, j, of vectors $\mathbf{x}^{n+1}$ and $\mathbf{x}^n$

$$\frac{\mathbf{x}^{n+1,j}}{\mathbf{x}^{n,j}} = \lambda_1 \frac{\alpha_1 u^{1,j} + \sum_{i=2}^{N} \alpha_i (\lambda_i/\lambda_1)^{n+1} u^{i,j}}{\alpha_1 u^{1,j} + \sum_{i=2}^{N} \alpha_i (\lambda_i/\lambda_1)^{n} u^{i,j}} \tag{A.12}$$

$\boxed{\texttt{eqA1:ratio1}}$

If $\alpha_1 u^{1,j} \neq 0$, the fraction above will tend to 1, and therefore

$$\lim_{n\to\infty} \frac{\mathbf{x}^{n+1,j}}{\mathbf{x}^{n,j}} = \lambda_1 .$$

In particular, the dominant term of the error will be proportional to $(\lambda_2/\lambda_1)^n$, therefore one has

$$\frac{\mathbf{x}^{n+1,j}}{\mathbf{x}^{n,j}} = \lambda_1 + O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^n \right)$$

Which component j should we use? Ideally, we would like to use the largest component, relative to the other terms appearing in the sums in the numerator and denominator of Eq. A.12, but we do not know which one it is. A better solution os to use a weighted average of all the components, with weights proportional to the components of vector $\mathbf{x}^n$:

$$\sigma_n \equiv \frac{\mathbf{x}^{n\top} A \mathbf{x}^n}{\mathbf{x}^{n\top} x^n}$$

This quantity

$$\sigma(A, \mathbf{x}) \equiv \frac{\mathbf{x}^\dagger A \mathbf{x}}{\mathbf{x}^\dagger \mathbf{x}}$$

is called the *Rayleigh quotient*, and, for matrix with real eigenvalues, has the property of being $\min_i \lambda_i \leq \sigma(A, \mathbf{x}) \leq \max_i \lambda_i$. Using the same argument as before, we can show that

$$\lim_{n\to\infty} \sigma_n = \lambda_1$$

Furthermore, if the matrix is symmetric, then the Rayleig quotient will provide a faster convergence to the eigenvalue. In fact, in this case the vectors u^i will form an orthogonal basis, and we can write

$$\mathbf{x}^{n\top} A \mathbf{x}^n = \sum_{i=1}^{N} \alpha_i \lambda_i^n \mathbf{u}^{i\top} \sum_{j=1}^{N} \alpha_j \lambda_j^{n+1} \mathbf{u}^j = \sum_{i=1}^{N} |\alpha_i|^2 \lambda_i^{2n+1} \|\mathbf{u}^i\|^2$$

and therefore

$$\sigma_n = \lambda_1 \frac{|\alpha_1|^2\|\mathbf{u}^1\|^2 + \sum_{i=2}^N |\alpha_i|^2\|\mathbf{u}^i\|^2(\lambda_i/\lambda_1)^{2n+1}}{|\alpha_1|^2\|\mathbf{u}^1\|^2 + \sum_{i=2}^N |\alpha_i|^2\|\mathbf{u}^i\|^2(\lambda_i/\lambda_1)^{2n}}$$

which means

$$\sigma_n = \lambda_1 + O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^{2n}\right)$$

Stopping criteria A practical stopping criterion is obtained by imposing a small relative residual, i.e.

$$\|A\mathbf{x}^n - \sigma_n\mathbf{x}^n\|_2 \le \varepsilon\|A\mathbf{x}^n\|_2 \tag{A.13}$$ eqA1:stop1

where ε is a preassigned tolerance. Then the ε represents a bound on the relative error of the eigenvalue. This property is due to the following theorem Then the following teorem holds [5]

Theorem A.2. *For any $\sigma \in \mathbb{C}$, and $\mathbf{x} \in \mathbb{C}^n$, let $\eta(\sigma,\mathbf{x}) = A\mathbf{x} - \sigma\mathbf{x}$. Then it follows*

$$\min_i |\lambda_i - \sigma| \le \frac{\|\eta\|}{\|\mathbf{x}\|}$$

If condition A.13 is satisfied, then one has

$$\min_i |\lambda_i - \sigma_n \le \frac{A\mathbf{x}^n - \sigma_n\mathbf{x}^n}{\mathbf{x}^n} \le \varepsilon\frac{A\mathbf{x}^n}{\mathbf{x}^n} \approx \varepsilon\lambda_1$$

which shows that ε is basically a bound on the relative error on the eigenvalue. **Practical considerations** The method just described is not practical because, if $|\lambda_1| > 1$, then the sequence of vectors $\mathbf{x}^n$ diverges, and the computer program will incur in an overflow. On the other hand, if $|\lambda_1| < 1$ then $\mathbf{x}^n \to 0$, and the computer will signal an underflow, or it will replace the components of $\mathbf{x}^n$ by machine zero.

For this reason it is better to normalize the vector $\mathbf{x}^n$. A normalized version of the method reads

```
∘ Start from $\mathbf{x}^0 \in \mathbb{R}^N$ arbitrary, for example $\mathbf{x}^0 = (1,\dots,1)^\top$
∘ set $n = 0$
1.set $\mathbf{y}^n = \mathbf{x}^n/\|\mathbf{x}^n\|$
  $\mathbf{x}^{n+1} = A\mathbf{y}^n$
  $\sigma_n = \mathbf{y}^{n\top}\mathbf{x}^{n+1}$
  if $\|\mathbf{x}^{n+1} - \sigma_n\mathbf{y}^n\| > \varepsilon\|\mathbf{x}^{n+1}\|$
    n = n+1
    go to 1.
  endif
```

A careful implementation of the algorithm requires the storage for two vectors and one sparse matrix, and it requires one matrix-vector product and two norm computation at each time step.

The condition $\alpha_1 \neq 0$ will be satisfied in most cases, since it is extremely unlikely that an arbitrary vector $\mathbf{x}^0$ contains no component of the first eigenvector. If $\alpha_1 = 0$, and if $|\lambda_2| > |\lambda_3|$, using the procedure described above, one the Rayleigh quotient should converge to the second eigenvalue, e.i.

$$\sigma_n \to \lambda_2, \text{ and } \mathbf{y}^n \to u^2 \quad \text{as } n \to \infty.$$

However, in practice, because of round-off errors, there will always be a small component of u^1 in $\mathbf{x}^n$, and therefore the procedure will converge to the first eigenvalue and the corresponding eigenvector.

The computation of λ_2 and of the corresponding eigenvector can be obtained by subtracting, from $\mathbf{x}^n$, the contribution proportional to u^1. In this case one would start from

$$\mathbf{x}^0 = \sum_{i=2}^{N} \alpha_i u^i$$

and the application of the power method should produce a Rayleigh quotient σ_n converging to λ_2, if the assumption $|\lambda_2| < |\lambda_3|$ is satisfied. A straightforward application of the method, however will not work. Because of inaccuracy in the initial condition and of round-off errors in machine arithmetic, the contribution of the first eigenvector in $\mathbf{x}^n$ is not zero, and therefore it will evantually dominate over the other terms. This technique can nevertheless be applied, if one regularly subtracts the contribution of the first eigenvector from $\mathbf{x}^n$ (say every iteration or every few iterations). This approach, called deflation, is described in [5] @@@Verifica che e' cosi@@@. Applying deflation, the power method can be used for the computation of few largest eigenvalues (and corresponding eigenvectors). The power method, applied to the inverse of matrix A, can be used for the computation of the smallest eigenvalue (in absolute value), and for the improvement of the approximation of a given eigenvalue [5].

A.3.2 All eigenvalues

If several eigenvalues are required, then the power method is no longer the method of choice. The most popular methods for the computations of all eigenvalues and eigenvectors are based on the so called QR factorization of the matrix and its variants [**?**]. More modern methods, particularly suited for the computation of a few largest and smallest eigenvalues of a large sparse matrix, are based on the Krylov subspace generated by a vector. The interested reader can find an extensive treatment of numerical methods for large sparse matrices in the book by Yousuf Saad [9].

Matlab provides an effective tool for the computation of all eigenvalues of a matrix, even if the matrix is in sparse form [10].