# **Eigenvalues and eigenvectors.**

In this chapter we are dealing only with *square* matrices. We have seen that when a matrix ***A*** acts on a vector ***x*** , that matrix behaves like a *function*, or, in the language of linear algebra, as an *operator*:

***Ax*** *=* ***f***(***x***)

***x ⇒ A*** acts on ***x ⇒ Ax***

Vector ***x*** comes *in*, and vector ***Ax***comes *out*. We are particularly interested in a special class of vectors ***x***:

those vectors ***x*** that come *in*, such that the vectors ***Ax*** that come *out*, come out in the same direction as ***x*** !

This means that ***Ax*** will be *parallel* to ***x***, although it might be of different *length* (a *multiple* or a *fraction* of***x***). That is very unusual, because in most cases ***Ax*** will come out in a different direction from ***x***. If ***Ax*** is *parallel* to ***x*** we say that ***x*** is an ***eigenvector*** of ***A***, and the number by which we multiply ***x*** to obtain ***Ax*** is its corresponding ***eigenvalue***. We can express this concept with the matrix equation:

eigenvector

eigenvalue

Of course ***x*** is not a single vector, but a basis vector for a vector space, because any multiple of ***x*** will still fulfill the same equation (***Ax*** will come out in the same direction as ***x***). can be any number (positive or negative) and even 0! But if is 0we have:

We have already encountered this case: what are the *eigenvectors* corresponding to = 0?They are the vectors in the *null space* of ***A***! By now we know that if there is something in the *null space*, it means the matrix ***A*** will bring some vectors to 0, and once you go to 0 there is no way to come back! Therefore:

if any eigenvalue of ***A*** is 0, then ***A*** is non-invertible = **SINGULAR**

But we are interested in all possible *eigenvalues* of ***A***, not just those with = 0, and unfortunately we do not have an easy way to solve the system of equations because we have 2 unknown (the *eigenpair* ***x*** and ), and they are multiplied together, so we can't use Gaussian elimination.

For some simple matrices, it is very easy to find some or all the possible *eigenpairs*, but special algorithms are required for large matrices. We will not discuss the algorithms that can be used for this purpose, but we will focus instead on the meaning of *eigenvectors* and *eigenvalues* and on some of their applications.

**Example 1:** consider a type of matrix we know very well: a projection matrix ***P***. We ask the question: what are the *eigenvectors* and *eigenvalues* of a projection matrix? As an example we can consider a projection matrix ***P*** that projects any vector ***b*** in **R*3*** onto a *plane* subspace of **R*3***.



In the example shown on the side, is ***b*** an *eigenvector* of projection matrix ***P*** ? Of course NO, because its projection ***p*** is not parallel to ***b***. However, any vector in the *plane* is an *eigenvector*, because the projection does not change its direction! What is the *eigenvalue* of any of these vectors? Of course it's 1, because the projection of any vector in the *plane* onto that same plane does not change its length!

So we have a full *plane* of *eigenvectors*! Any two linearly independent vectors (for example ***s1*** and ***s2***) in this plane will be *basis vectors* for the *eigenvectors space* and their *eigenvalues* will be λ1= λ2= 1.

Are there any other eigenvectors? In fact, there is one other vector that is an *eigenvector* of ***P***: it is the vector perpendicular to the plane (for example ***s3***), and it has *eigenvalue* λ3= 0 (that is, the projection of ***s3*** on the plane is just the origin).

Thus, we can say that the projection matrix ***P*** has three *eigenvectors* (as a basis):

λ3 ***=*** 0

***Ps1 = s1***

***Ps2 = s2***

λ1 ***=*** 1

λ2 ***=*** 1

***s1*** and ***s2*** are in the *column space* of ***P***

***Ps3 = 0***

***s3*** is in the *null space* of ***P***

When we studied the 4 fundamental spaces of a matrix we learned that the column space **C(*P*)** of ***P*** is perpendicular to the *left null space* **LN(*P*)** of ***P***. Now we learn that the *null space* **N(*P*)** of a projection matrix ***P*** is also perpendicular to **C(*P*)**. Thus, in a projection matrix **N(*P*)** = **LN(*P*)**. Since we know that **N(*P*)** is also perpendicular to the *row space* **R(*P*)**, it follows that in a projection matrix **R(*P*)** = **C(*P*)**, as we expected from the fact that ***P*** is symmetric.

These properties of the specific projection matrix we have analyzed are valid for all projection matrices that project orthogonally to a space, so we can conclude that no matter how large a projection matrix is, its *eigenvalues* are always 1 and 0. Since at least 1 *eigenvalue* is 0, projection matrices that project onto a subspace are always **SINGULAR**.

**Example 2:** consider the identity matrix:

If we switch rows 2 and 3 we obtain a new matrix ***P***:

If we multiply the new matrix by a vector ***x*** we find that the resulting vector ***Px*** is a *permutation* of the rows of ***x***.

For this reason ***P*** is called a *permutation* matrix (not to be confused with a *projection* matrix, which we typically also identify with the letter P) . Permutation matrices are commonly used in the software implementation of Gaussian elimination, because in many cases the operation can be faster and numerically more stable if the rows of a matrix are exchanged. However, in this case we are not interested in the application of permutation matrices, but in their *eigenvectors* and *eigenvalues*. These can be easily found for the matrix ***P*** above. For example, we can see that the following three products provide us with the desired information:

***P******s1*** *λ1* ***s1***

***P******s2*** *λ2*  ***s2***

***P******s3*** *= λ3*  ***s3***

Let's think about the meaning of what we just found. We have 3 basis vectors, the *eigenvectors* ***s1****,* ***s2****,* ***s3***, and we know that the operation carried out by the matrix ***P*** on each of the three *eigenvectors* is *simply* the multiplication of each *eigenvector* by its corresponding *eigenvalue*. We can represent this as the following matrix equation:

***P******S*** *=* ***S*** ***Λ***

where **Λ** is a diagonal matrix containing all the *eigenvalues*.

Now let's imagine we have a vector ***a*** in standard Cartesian basis

and we want to represent it as ***as*** using the *eigenvector* basis **S**:

Clearly, we can find some linear combination of the three *eigenvectors* ***s1****,* ***s2****,* ***s3***:

++

where *a1*, *a2*, and *a3* are just the scalar coefficients multiplying each of the basis vectors such that:

Now let's consider the product ; substituting for the linear combination

we obtain:

but we know that so we can further substitute:

Notice what we are doing in order to calculate (going for each element from left to right):

1. we find the coordinates *a1*, *a2*, *a3* of ***a*** in the eigenvector basis **S**. We know how to do this: it is a simple change of basis:

2. we multiply each coordinate by its corresponding *eigenvalue*:

3. we multiply the resulting product by each *eigenvector*:

The three steps are:

move ***a*** into ***S*** space

mutiply by the *eigenvalues*

move back into *standard* space

The final result is:

which means we are representing ***P*** as:

This is called the *spectral* or *eigen decomposition* of ***P***.

We can view the equality from another point of view: as a *similarity transformation*! In fact:

This means that ***Λ*** is *similar* to ***P*** in ***S*** space: we say the ***P*** is *diagonalized* in ***S*** space.

It's *very important* to remember that if we have found the *eigenvalues* and *eigenvectors* for a general square matrix ***A***, the following is always true:

and we can always write:

However, if we want to bring ***S*** on the other side (in other words, if we want to carry out an *eigen decomposition* or a *similarity transformation* of ***A*** as:

***S*** must be invertible, which means that

***S*** must have ***n*** independent *columns*

If this condition is not met, ***A*** is not *diagonalizable*, which means that the diagonal matrix ***Λ*** of *eigenvalues* is not *similar* to ***A*** in anyspace. It turns out (but we will not offer proof of it) that:

if the λ's of ***A*** are all different ⇒ ***S*** is invertible

if some of the λ's of ***A*** are the same ⇒ ***S*** may or may not be invertible

There are many cases in biochemistry (we will encounter some soon) where we want to solve problems of the type:

***uk*** = ***Au0***

but the meaning of the matrix product (the *coupling* between the elements in vector ***u0*** by a matrix to produce vector ***uk***)is not immediately obvious. In those cases, it is convenient to move the problem into a space, the *eigenvector space*, in which the *coupling* is removed.

For this purpose we will always follow a 3 step general strategy of working by *similarity transformation*:

1. represent ***u0*** in the *eigenvector* basis ***S***

2. follow each *eigenvector* component separately

3. go back in standard basis

Notice the conceptual similarity between these three steps and the alternative 3 step solution:

1. find the coordinates ***uS*** of ***u0*** in *eigenvector* space:

2. write ***u0*** as a linear combination of the *eigenvectors* of***A***:

3. multiply each *eigenvector* by its *eigenvalue*:

the solution is a linear combination of the '*pure*' solutions : these are the independent *normal modes* of the system. Notice something very important: the coefficients of the linear combination are unchanged between and **.**

**Some useful information about *eigenvalues* and *eigenvectors*.**

While we are not discussing the algorithms used to identify the *eigenvalues* and *eigenvectors* of a matrix, it is convenient to remember some of their important properties:

1. The *trace* (sum of the diagonal elements) of a matrix is the sum of the *eigenvalues*.

rng(1)

M = randi(10,4)

[S,D] = eig(M)

trace(M)

sum(diag(D))

2. The *product* of the *pivots* in the Gaussian elimination of a matrix is equal to the *product* of the *eigenvalues*.

[x,A] = gauss\_elim\_step\_by\_step(M,[1 2 3 4]')

prod(diag(A))

prod(diag(D))

3. The *determinant* of a matrix is the *product* of the *eigenvalues* (or equivalently the product of the *pivots* in Gaussian elimination). Therefore, if the *determinant* is 0, it means that at least 1 *eigenvalue* is 0 (there is something in the *nullspace*), and thus the matrix is not invertible (SINGULAR). The *pseudodeterminant* of a singular matrix is the *product* of the *eigenvalues* different from 0.

det(M)

We recall here some simple alternative formulas to find the *determinant* of a matrix. For example:

For larger matrices we can use the *cofactor* formula:

where the signs of the coefficients *a*, *b*, *c*, are determined by the sum of the values of their indices (+ for even, - for odd). For example, for the matrix above the sign matrix is:

However, typically all modern software uses Gaussian elimination or determine the *eigenvalues* to calculate the *determinant*.

4. Adding or subtracting a multiple ***nI*** of the identity matrix ***I*** to a matrix does not change the *eigenvectors* and adds or subtracts ***n*** to all the *eigenvalues*.

M2 = M + 0.00001\*eye(4)

[S2,D2] = eig(M2)

S

S2

diag(D)

diag(D2)

This property is very important, because it is used in the *conditioning* of non-invertible matrices, by adding or subtracting fractions of ***I*** until all the 0 *eigenvalues* disappear, and the matrix becomes invertible.

5. The *eigenvalues* of an upper or lower triangular matrix or of a diagonal matrix are the numbers on the diagonal.

M\_ut = triu(M)

[S\_ut,D\_ut] = eig(M\_ut)

M\_lt = tril(M)

[S\_lt,D\_lt] = eig(M\_lt)

M\_diag = diag(diag(M))

[S\_diag,D\_diag] = eig(M\_diag)

6. The *eigenvalues* of ***AT*** are the same as the *eigenvalues* of ***A***. The *eigenvectors* ***s***of ***AT***, such that ***ATs*** *=*λ***s***, are called the *left eigenvectors*of ***A* (**as opposed to the standard *right eigenvectors*)because they fulfill the matrix equation ***sTA*** *=*λ***sT***.

M4 = randi(10,4,4)

[S,D] = eig(M4)

[D,sort\_ind] = sort(diag(D),'descend')

D = diag(D)

S = S(:,sort\_ind)

M4\*S

S\*D

[S2,D2] = eig(M4')

[D2,sort\_ind\_2] = sort(diag(D2),'descend')

D2 = diag(D2)

S2 = S2(:,sort\_ind\_2)

M4'\*S2

S2\*D2

S2'\*M4

D2\*S2'

Matlab function *eig* provides directly both *right* and *left eigenvector* of a matrix as its 3rd output***.***

[V,D,W] = eig(M4)

M4\*V

V\*D

W'\* M4

D\*W'

7. The *eigenvalues* of ***A-1*** are the reciprocal (1/λ) of the *eigenvalues* of ***A***.

eig(inv(M))

1./eig(M)

8. Since the sum of the eigenvalues is the trace of a matrix, and the trace of ***A***+***B***  is the sum of the traces of ***A*** and ***B***, one might expect that the eigenvalues of ***A***+***B*** are the sum of the eigenvalues of ***A*** and ***B***. However, this is normally not the case with some exceptions. If ***A*** and ***B*** have the same *eigenvectors*, then the *eigenvalues* of ***AB*** are the same as the *eigenvalues* of ***BA***.

S = rand(4)

D1 = diag(rand(4,1))

M1 = S\*D1\*inv(S)

D2 = diag(rand(4,1))

M2 = S\*D2\*inv(S)

eig(M1\*M2)

eig(M2\*M1)

D1+D2

eig(M1+M2)

More generally, if ***A*** and ***B*** *are complex* and *commute* (***AB = BA***) it means they are *hermitian*. A hermitian matrix is a *self-adjoint* matrix, which is the 'complex' equivalent of a symmetric matrix. When we take the transpose of a complex matrix the sign of the imaginary part changes, and properly we define it as the *conjugate transpose* or the *adjoint*: this transposition is represented alternatively as or as or . Thus, a *hermitian* or *self-adjoint* matrix is a complex matrix that remains unchanged upon *conjugate transposition*. In the case of two hermitian matrices that share the same eigenvectors the eigenvalues of (***A***+***B***) are the sum of the eigenvalues of ***A*** and the eigenvalues of ***B*** and the eigenvalues of ***AB*** are the product of the eigenvalues of the individual matrices. This is valid also for the 'real' equivalent of hermitian matrices, when ***A*** and ***B*** are both real, symmetric, and commute.

9. If the *eigenvector* matrix ***S*** of ***A*** is invertible, then any power (including negative and fractional powers = roots) of ***A*** can be easily calculated as:

Examples:

An important corollary of this property is the calculation of a *matrix exponential*, which is very important in biochemical applications. If the *eigenvector* matrix ***S*** of ***A*** is invertible, then:

We can understand exponentiation to a matrix by looking at the power series representation of :

similarly we get for our matrix exponential:

where ***I*** is the identity matrix. Expanding we obtain:

M1 = rand(4)

M1 = M1'\*M1

[S,D] = eig(M1)

M1^4

S\*D^4/S

notice the difference between the *matrix* exponential *expm* and the *elementwise* exponential of matrix elements *exp*:

exp(M1)

expm(M1)

S\*expm(D)/S

S\*diag(exp(diag(D)))/S

and likewise for any exponentiation of a number to a matrix:

(exp(1))^M1

5^M1

S\*diag(5.^diag(D))/S

This result allows us to make some predictions on the outcome of a matrix product that involves powers or exponentiation. For example, if we have the equation:

***uk+1*** = ***Auk***

and we wanto to know what happens if we apply the same operation over and over (let's say 100 times) to the result of the previous operation. This is the same as:

***u100*** = (***A*** ...(***A*** (***A*** (***A*** (***A*** (***Au0***))))) = ***A100u0***

since it follows that:

Likewise, for the case in which the matrix is the exponent, since we have:

10. If ***A*** is *symmetric* (***AT*** *=* **A**) all *eigenvalues* are *real* numbers and all the *eigenvectors* can be chosen as *orthogonal* .

M3 = rand(4)

M3 = M3\*M3'

M3 = M3'\*M3

[S,D] = eig(M3)

S'\*S

S\*S'

S\*D\*S'

In this case we can write ***S*** as ***Q***, the letter typically used to represent an *orthogonal* matrix: this is a square matrix with all columns as unit vectors perpendicular to each other such that ***QT*** *=* ***Q-1*,** and ***QTQ*** *=* ***QQT*** *=* ***I***.

A matrix (real or complex) that commutes with its transpose is called a ***normal*** matrix: thus, if ***AT*A *= AAT*** or more generally, if ***A\**A *= AA\****, where ***A\**** is the *conjugate transpose* or *Hermitian transpose* or *adjoint* of the complex matrix ***A*** obtained by taking the transpose and then the complex conjugate (i.e., negating the imaginary parts but not the real part), then ***A*** is normal.

The complex equivalent of an orthogonal matrix ***Q*** is a ***Unitary*** matrix ***U*** for which the following holds:

Clearly, orthogonal ***Q*** and unitary matrices ***U*** are normal. If ***A*** is a normal matrix, then it is always true that there exist a diagonal matrix ***Λ*** and a unitary matrix ***U*** such that:

This is called the ***spectral theorem***, which states that:

*any normal matrix is always diagonalizable by a unitary matrix.*

11. If ***A*** is *antisymmetric* (***AT*** *= -***A**), also called *skewsymmetric,* but not orthogonal, all *eigenvalues* are *imaginary* numbers, and the *eigenvectors* are not orthogonal.

M3 = rand(4)

M3 = triu(M3)-triu(M3)'+diag(diag(M3))

[S,D] = eig(M3)

S'\*S

S\*S'

12. If ***A*** is *orthogonal* (***Q***), then all the *eigenvectors* can be chosen as *orthogonal*.

Q1 = [cos(pi/2) sin(pi/2) 0;...

-sin(pi/2) cos(pi/2) 0;...

0 0 1]

or

Q1 = [1 0 0;...

0 cos(pi/2) sin(pi/2) ;...

0 -sin(pi/2) cos(pi/2)]

[S,D] = eig(Q1)

S'\*S

S\*S'

From the 3 properties above it follows that the *eigen decomposition*

of a *symmetric* or *orthogonal* matrix is always possible as:

In all other cases in which is invertible only the decomposition

is possible. If is not invertible, the following equality still holds:

13. If all the *eigenvalues* of a *symmetric* matrix ***A*** are >0, ***A*** is called *positive definite*. An important property of *positive definite* matrices is that the product:

In many applications this number or more often represents the ***energy*** of the system, which cannot be negative. As a consequence, if ***A*** and ***B*** are both *positive definite* then ***C = A+B*** is also *positive definite* because the sum of two positive energies is a positive energy.

If ***A*** is *positive definite* (***A*** is invertible) it is always true that ***A*** = ***RTR***, where ***R*** is a matrix with independent columns (***R*** is invertible). Conversely, if ***R*** is a matrix with independent columns, ***RTR*** is *positive definite*, and therefore, by definition, invertible.

14. If the *eigenvalues* of ***A*** are ≥ 0, ***A*** is *positive semidefinite*. In this case, at least 1 *eigenvalue* is always 0, and thus the matrix is *singular*. An important property of *positive semidefinite* matrices is that the product:

If a matrix is *positive semidefinite*, it can be factorized as ***RTR***, in which ***R*** has some dependent columns. Conversely, if ***R*** is a matrix with some dependent columns, ***RTR*** is *positive semidefinite*, and therefore, by definition, singular.

15. If the *eigenvalues* of ***A*** are both ≥ 0 and < 0, ***A*** is *indefinite*.

16. A linear combination of the eigenvectors of ***A*** is NOT an eigenvector of ***A*** corresponding to any eigenvalues of ***A.***

Suppose that are two *distinct* eigenvalues of a square matrix ***A*** and let be the corresponding eigenvectors. We assume that is an eigenvector of ***A*** corresponding to an eigenvalue . Thus we have:

(\*)

Since by definition , we calculate the left hand side of this equality as follows:

(\*\*)

From (\*) and (\*\*) we obtain:

Since eigenvectors corresponding to distinct eigenvalues are linearly independent, and are linearly independent, and thus the coefficients of the above linear combinations must be zero:

However, since , this implies:

which is a contradiction, because and were defined as distinct. Hence, cannot be an eigenvector of any eigenvalue of ***A***.

Thus, in general, given two eigenvectors and of ***A***with associated eigenvalues , , unless or and are linearly dependent:

17. The *generalized* *eigenvalue problem* is defined by the equivalence:

eigenvector

eigenvalue

In this case, a general solution for the *eigenpair* can be found using the strategy outlined in the practice section below, based on the solution for of the *characteristic polynomial*, followed by back-substitution to find the eigenvectors in the *null space* of :

However, an intuitive feeling for the solution of this problem can be gained if ***B*** is invertible, in which case the generalized eigenvalue problem can be recognized as a special case of the basic problem:

eigenvector

eigenvalue

rng(5)

A = randi([-10 10],4,4)

B = randi([-10 10],4,4)

A = A'\*A

B = B'\*B

[V,D,W] = eig(A,B)

[D,ind] = sort(diag(D),'descend')

D = diag(D)

V = V(:,ind)

W = W(:,ind)

A\*V

B\*V\*D

W'\*A

D\*W'\*B

where ***V*** and ***W*** are the *right* and *left eigenvectors*. In this case ***V*** and ***W*** are just the transpose of each other because ***A*** and ***B*** are symmetric.

**PRACTICE**

**1.** Using known properties of the spectral decomposition of matrices, find the eigenvalues of the following matrices , .

**2.** We know that for every *eigenvector* ***x***of ***A*** the following equality is true:

From which we derive:

Clearly vector ***x*** must be in the *null space* of the matrix and therefore this matrix is singular (non invertible); we have also learned that if a matrix is singular the product of its *eigenvalues* is 0, and therefore the value of the determinant is also 0:

This gives us a way of finding the *eigenvalues* of a simple matrix; for example consider the matrix:

The final polynomial in λ of degree ***n*** is called the *'characteristic polynomial'*: we can use the formula for quadratic roots to find the two values of λ:

A = [3 1;1 3]

syms x

charpoly(sym(A),x)

ch\_poly = charpoly(A)

D = roots(double(ch\_poly))

Now that we know the *eigenvalues* λ1 and λ2 we can substitute them back in the equation to find the *eigenvectors* (=*null space* of the equation). Doing the substitution one at a time we get:

for λ1 = 2

for λ2 = 4

S1 = null(A-D(1)\*eye(2)) , S2 = null(A-D(2)\*eye(2))

Thus, the two *eigenpairs* are: λ1 = 2, ***x1*** = and λ2 = 4, ***x2*** = . Using this example as guidance, find the *eigenvectors* of ***A***:

**IMPORTANT:** this simple algorithm to find the eigenvalues and eigenvectors of a matrix is based on the assumption that we have a method to find a basis for the *nullspace* of a singular matrix. This task is accomplished quite easily for small matrices by Gaussian elimination. For example, let’s find the basis for the *nullspace* **N**(***A***) of ***A***:

A = [1 1 2 3 1 1 1;2 2 8 10 3 2 1;3 3 10 13 1 2 3]

[m,n] = size(A);

r = rank(A);

rref(sym(A))

***A*** has rank 3, and its *row reduced echelon* matrix is:

**pivot columns**

**free columns**

We know there are 4 vectors in the *nullspace* of ***A***. We are going to set up 4 ***x*** vectors (these are the 4 *special solutions* of **N**(***A***)), each containing 3 *pivot variables* (x1 x3 x5) and 4 *free variables* (x2 x4 x6 x7), corresponding to the *pivot columns* and the *free columns* of the *row reduced echelon* matrix. The pivot columns correspond to the *independent* columns of ***A***, while the free columns correspond to the dependent columns of ***A***. Since each dependent column can be derived as a linear combination of the independent columns, in each special solution, in turn, a free variable is set to 1 (*special choice*), and the other 3 free variables are set to 0. Based on the choice of free variables, the values of the pivot variables are automatically determined by solving for those variables by Gaussian elimination. This corresponds to finding the linear combination of the independent columns of ***A*** that give the chosen dependent column and subtracting that combination from the dependent column. In practice, for each *special solution* vector, the values of the 3 pivot variables (the combination of independent columns) are read directly from the free column that provides the *special choice*, with the sign changed:

x1 x3 x5 for all nullspace vectors are read from the rref with the negative sign:

template for all special solutions = [x1 x2 x3 x4 x5 x6 x7]

template for s1 = [x1 1 x3 0 x5 0 0] ⇒ [-1 1 -0 0 -0 0 0]

template for s2 = [x1 0 x3 1 x5 0 0] ⇒ [-1 0 -1 1 -0 0 0]

template for s3 = [x1 0 x3 0 x5 1 0] ⇒ [-5/6 0 1/12 0 -1/3 1 0]

template for s4 = [x1 0 x3 0 x5 0 1] ⇒ [-5/3 0 1/6 0 1/3 0 1]

Alternatively, solve for **X** in:

,

,

Bringing all together:

independent columns dependent columns

x\_ind = [1:7];

[~,pivot\_ind] = rref(A);

free\_ind = setdiff(x\_ind,pivot\_ind);

npivots = length(pivot\_ind);

nfree = length(free\_ind);

Ap = -A(:,pivot\_ind);

Af = A(:,free\_ind);

X = Ap\Af;

I = eye(nfree);

NA = zeros(n,n-r);

for i = 1:nfree

NA(pivot\_ind,i) = X(:,i);

NA(free\_ind,i) = I(:,i);

end

NA

The special solutions can be easily orthogonalized by the *Gram-Schmidt* method. The first solution s1 is accepted. We determine the projection p2 of s2 onto s1 and we subtract from s2: the difference vector is orthogonal to s1. We determine the projection p31 and p32 of s3 onto s1 and s2 and we subtract from s3: the difference vector is orthogonal to s1 and s2. The basic idea is to subtract from every new vector its projections in the directions already defined. At the end we convert everything into unit vectors.

Gram-Schmidt

s1 = NA(:,1);s2 = NA(:,2);s1 = NA(:,3);s1 = NA(:,4);

s2 = s2-s1\*inv(s1'\*s1)\*s1'\*s2

s3 = s3-[s1 s2]\*inv([s1 s2]'\*[s1 s2])\*[s1 s2]'\*s3

s4 = s4-[s1 s2 s3]\*inv([s1 s2 s3]'\*[s1 s2 s3])\*[s1 s2 s3]'\*s4

s1 = s1/norm(s1); s2 = s2/norm(s2); s3 = s3/norm(s3); s4 = s4/norm(s4)

oNA = [s1 s2 s3 s4]

oNA'\*oNA

A\*oNA

**3.** Find the *eigenvalues*, the *eigenvectors*, and the inverse of ***A***:

Notice that the two matrices that sum up to A are both symmetric, real, and also commute. Recalling that if a matrix has rank *n* there can't be more than *n* eigenvalues different from 0, try to solve the problem without using MATLAB; however, if using MATLAB, it is convenient to convert the matrix A into a symbolic matrix. For example:

A1 = eye(4)\*5

A2 = -ones(4)

A = A1 + A2

A = sym(A)

**4**. True or false: if the columns of ***S*** (*eigenvectors* of ***A***) are linearly independent, then:

a) ***A*** is invertible

b) ***A*** is diagonalizable

c) ***S*** is invertible

d) ***S*** is diagonalizable

**5**. We know the *eigenvalues* of ***AT*** are the same as the *eigenvalue* of ***A***.

a. What are the *eigenvectors* ***y***of ***AT*** such that ***ATy*** *=*λ***y***? These are often called the ***left eigenvectors*** of ***A* (**as opposed to the standard *right eigenvectors*)because they fulfill the matrix equation ***yTA*** *=*λ***yT***.

b. What is the relationship between the left and the right *eigenvectors* if ***A*** is symmetric?

**6.** Which of the four subspaces of a projection matrix contains *eigenvectors* with λ = 1? Which subspace contains *eigenvectors* with λ = 0?Can you say if a projection matrix can be diagonalized?