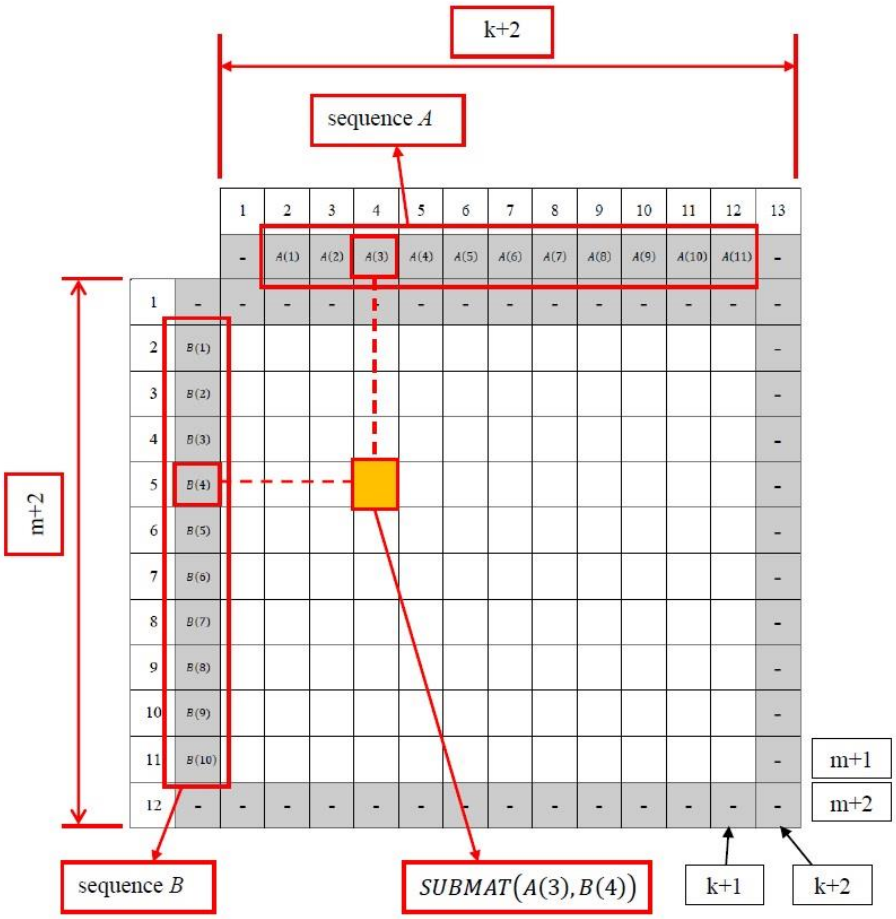


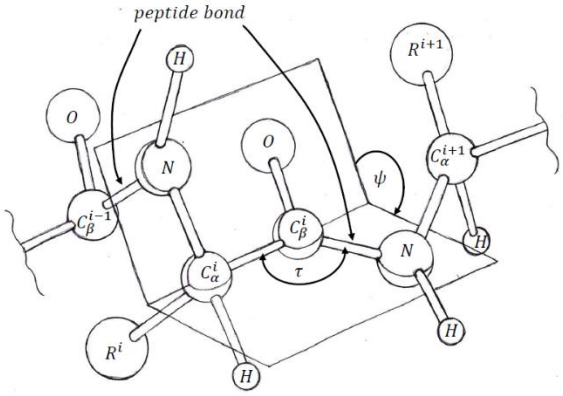
NEEDLEMAN-WUNSCH ALGORITHM

Global alignment between two peptides

Paolo Maccallini

2016

Index



1 Alignment of a pair of sequences

1.1 What is an alignment?

In an alignment of two proteins (*pair-wise alignment*), the two peptides are written one under the other, with the purpose of collecting identical or similar pair of amino acids in the same columns. The introduction of gaps is allowed in order to pursue the construction of as many identical or similar pairs as possible. When we have two identical amino acids in the same column, we have a *match* (evidenced by a shade of gray in Table 1); otherwise we have a *mismatch* (when different amino acids are in the same column) or a gap, when one of the two sequences has a deletion. For both matches and mismatches we can use a substitution matrix in order to read a score; we have to decide a negative score to add to the total score of the alignment, for each gap (Mount, 2001). Consider for instance the alignment in Table 1, where sequence A is a peptide from an enzyme of *Pseudomonas putida* and sequence B is a peptide from the same enzyme from *Homo sapiens*. In this alignment, a gap has been introduced at the third position, and if we use BLOSUM62↕ scoring matrix, and a gap penalty of -12, we have the total score of 37.
As it is easy to understand, the problem in building a pair wise alignment arises when we have to decide how many gaps to introduce, and where to introduce them. Another problem is how to calculate the value for a gap penalty. This problem is the subject for computational models that have been developed from the seventies, with the *dot matrix analysis* (Gibbs, et al., 1970)↕ and the *dynamic programming algorithm* (Needleman, et al., 1970)↕.

Table 1. Example of scoring a sequence with the presence of gap and a relative gap penalty.

Seq_A	N	M	P	G	Q	M	S	M	P	V	P	Y	T	G	I	I	N	N
Seq_B	N	F	-	S	E	A	I	L	P	V	P	V	N	G	V	I	N	N
score	6	0	-12	0	2	-1	-2	2	7	4	7	-1	0	6	3	4	6	6

1.2 Why to calculate alignments?

Sequences alignment has as a purpose that of highlighting functional, structural, and evolutionary relationships between two or more sequences of peptides, as proteins that are very much *similar* in their peptide chains, are likely to have the same function, common ancestor, and a similar three dimensional structure (Mount, 2001). So, applications of proteins alignment methods are many: from the study of function of enzymes, to the prediction of 3-D structures for new proteins, to phylogenetic trees reconstructions, to prediction of autoimmunity risks for immunizations or infections.
It may be useful to remember some definitions for possible relationships between proteins (and genes as well). We define *homologous* two sequences that are from different organisms, but share similarity and are thus thought to have a common ancestor. If, in particular, these two sequences didn't derive from a gene duplication, they are called *orthologs*. The two copies of proteins derived from a gene duplication, are sequences called *paralogs* (Mount, 2001).

1.3 How many alignments?

Let us consider two amino acids sequences: peptide $A = \{a_1, a_2, \dots, a_k\}$ and peptide $B = \{b_1, b_2, \dots, b_m\}$, where *k* and *m* are two integers in general not equal. It is not so easy to calculate the number $c(k, m)$ of possible alignments between these two peptides. A feasible avenue is that of evaluate a lower bound for $\eta(k, m)$ by considering the number of groups of alignments which in which the same pairs are present, with different gaps (Ewens, et al., 2005). In order to clarify the idea, let us consider sequence $A = \{a_1, a_2, a_3, a_4, a_5\}$ and sequence $B = \{b_1, b_2, b_3, b_4, b_5, b_6\}$. For

instance, the group of alignments between *A* and *B* where only the three pairs (a_1, b_1), (a_2, b_3), (a_3, b_4) are present, collects all the following seven alignments:

Eq. 1

$$\begin{array}{l} a_1-a_2a_3a_4a_5 \\ b_1b_2b_3b_4b_5b_6 \end{array} \qquad \begin{array}{l} a_1-a_2a_3--a_4a_5 \\ b_1b_2b_3b_4b_5b_6-- \end{array} \qquad \begin{array}{l} a_1-a_2a_3-a_4a_5 \\ b_1b_2b_3b_4b_5-b_6 \end{array}$$

$$\begin{array}{l} a_1-a_2a_3a_4-a_5 \\ b_1b_2b_3b_4-b_5b_6 \end{array} \qquad \begin{array}{l} a_1-a_2a_3a_4a_5-- \\ b_1b_2b_3b_4--b_5b_6 \end{array} \qquad \begin{array}{l} a_1-a_2a_3a_4a_5-- \\ b_1b_2b_3b_4--b_5b_6 \end{array}$$

Thus, if we define $\omega(k,m)$ the total number of possible groups, we have $\eta(k,m) \geq \omega(k,m)$. But how many are these groups? If we consider groups where the number of pairs without gaps are *p*, then we have to choose *p* amino acids from each sequence, and we can select them in $\binom{k}{p}$ ways from sequence *A* and in $\binom{m}{p}$ ways from sequence *B*, where $\binom{k}{p}$ is the binomial coefficient which indicates the number of combinations of *k* elements taken *p* at a time (Ghizzetti, et al., 1996). Thus, the total number of groups of alignment where only *p* pairs without gaps are present, is $\binom{m}{p} \cdot \binom{k}{p}$. But *p* goes from 0 to $min\{k,m\}$, thus we have found that:

Eq. 2 $$\omega(k,m) = \sum_{p=0}^{min\{k,m\}} \binom{m}{p}\binom{k}{p}$$

We will now prove that

Eq. 3 $$\omega(k,m) = \binom{k+m}{k} = \binom{k+m}{m}$$

As a first step, we will assume that

Eq. 4 $$\binom{k+m}{k} = \binom{k}{0}\binom{m}{k} + \binom{k}{1}\binom{m}{k-1} + \binom{k}{2}\binom{m}{k-2} + \cdots + \binom{k}{k}\binom{m}{0}$$

To justify Eq. 4 (which will not be proven here) consider the meaning of its first member: it is the number of combinations of *k* elements chosen from *k*+*m* elements. If we imagine to take those *k* elements from two groups, one of *k* elements and one of *m*, we have the second member of the equation. The second member of Eq. 4 gives

$$\sum_{p=0}^{k} \binom{m}{p}\binom{k}{k-p} = \binom{k}{0}\binom{m}{k} + \binom{k}{1}\binom{m}{k-1} + \cdots + \binom{k}{p}\binom{m}{k-p} + \cdots + \binom{k}{k-1}\binom{m}{1} + \binom{k}{k}\binom{m}{0} =$$

$$= \frac{m!}{k!\,(m-k)!} + \frac{k!}{(k-1)!}\frac{m!}{(k-1)!\,(m-k+1)!} + \cdots + \frac{k!}{p!\,(k-p)!}\frac{m!}{(k-p)!\,(m-k+p)!} + \cdots$$

$$\ldots + \frac{k!}{(k-1)!}\frac{m!}{(m-1)!} + 1 =$$

$$= \binom{k}{k}\binom{m}{k} + \binom{k}{k-1}\binom{m}{k-1} + \cdots + \binom{k}{k-p}\binom{m}{k-p} + \cdots + \binom{k}{1}\binom{m}{1} + \binom{k}{0}\binom{m}{0}$$

On the other hand, if $min\{k,m\} = k$, the second member of Eq. 2 is

$$\sum_{p=0}^{k}\binom{m}{p}\binom{k}{p} = \binom{m}{0}\binom{k}{0} + \binom{m}{1}\binom{k}{1} + \cdots + \binom{m}{k-p}\binom{k}{k-p} + \cdots$$

$$\ldots + \binom{m}{k-1}\binom{k}{k-1} + \binom{m}{k}\binom{k}{k} =$$

$$= 1 + \frac{m!}{(m-1)!}\frac{k!}{(k-1)!} + \cdots + \frac{m!}{(m-k+p)!\,(k-p)!}\frac{k!}{(k-p)!\,p!} + \cdots$$

$$\ldots + \frac{m!}{(m-k+1)!\,(k-1)!}\frac{k!}{(k-1)!} + \frac{m!}{(m-k)!\,k!}$$

Thus, the addend *p-th* of $\sum_{p=0}^{k}\binom{m}{p}\binom{k}{k-p}$ is equal to the addend (*k-p)-th* of $\sum_{p=0}^{k}\binom{m}{p}\binom{k}{p}$ and we have

$$\binom{k+m}{k} = \sum_{p=0}^{k}\binom{m}{p}\binom{k}{p}, with\ k < m$$

In the same way, we can prove that

$$\binom{k+m}{m} = \sum_{p=0}^{m}\binom{m}{p}\binom{k}{p}, with\ m < k$$

So, we have proven Eq. 3. If we now apply Stirling's approximation to Eq. 3, we have

Eq. 5 $$\omega(k,m) = \binom{k+m}{k} \sim \frac{1}{\sqrt{2\pi}}\frac{\sqrt{k+m}}{\sqrt{km}}\left(\frac{k+m}{k}\right)^{k}\left(\frac{k+m}{m}\right)^{m}$$

Thus, we conclude that the number of possible alignments between a protein of *k* amino acids and a protein of *m* amino acids is

Eq. 6 $$\eta(k,m) \geq \binom{k+m}{k} \sim \frac{1}{\sqrt{2\pi}}\frac{\sqrt{k+m}}{\sqrt{km}}\left(\frac{k+m}{k}\right)^{k}\left(\frac{k+m}{m}\right)^{m}$$

So, if we consider the number of possible alignments between *P. putida* enolase (*k*=429 *aa*) and human alpha enolase (*m*=434 *aa*), we have

$$\eta(429,434) \geq \binom{863}{429} \sim \frac{1}{\sqrt{2\pi}}\frac{\sqrt{863}}{\sqrt{186.186}}\left(\frac{863}{429}\right)^{429}\left(\frac{863}{434}\right)^{434} = 1.64 \cdot 10^{258}$$

If we have two proteins of the same length (let's say *n* amino acids), thus Eq. 5 becomes

Eq. 7 $$\eta(n,n) \geq \binom{2n}{n} \sim \frac{2^{2n}}{\sqrt{\pi n}}$$

As you can see the number of possible alignments is very high, and this induced the first mathematicians who faced the problem of protein alignment in the seventies, to search for a way to find the best alignment without having to calculate the score of every possible alignment.

1.4 Dynamic programming algorithm

The dynamic programming method was first developed for global alignment (Needleman, et al., 1970)↕ and then further developed for local alignment (Smith, et al., 1981)↕. Mathematical refinement and discussion was then added to proof the validity of this algorithm (Smith, et al., 1981)↕. Dynamic programming generates every possible alignment between two proteins, by introducing gaps in every possible number and position. For each of these alignments, it then calculates the global score, which is obtained by adding log odds score for each pair of amino acids and by subtracting penalties for gaps. Log odds score are calculated from a selected substitution matrix, such as Dayhoff PAM250 or BLOSUM62, while the value for gap penalties has to be calculated according to evaluations that we will see further ahead. The total score provides the ratio between the probability that the alignment is due to an evolutionary and/or functional relationship and the odds that it is due by chance, given the observed frequencies for amino acids. As mentioned, the score provided by alignment algorithms such as dynamic programming, are important for making functional, structural, and evolutionary predictions (Mount, 2001).

Table 2. Alignment generated from the column $(-, b_1)$, after three steps of the algorithm.

$-$ b_1								
$- -$ $b_1 b_2$			$-a_1$ $b_1 -$			$-a_1$ $b_1 b_2$		
$- - -$ $b_1 b_2 b_3$	$- - a_1$ $b_1 b_2 -$	$- - a_1$ $b_1 b_2 b_3$	$-a_1 -$ $b_1 - b_2$	$-a_1 a_2$ $b_1 - -$	$-a_1 a_2$ $b_1 - b_2$	$-a_1 -$ $b_1 b_2 b_3$	$-a_1 a_2$ $b_1 b_2 -$	$-a_1 a_2$ $b_1 b_2 b_3$

Table 3. Alignment generated from the column $(a_1, -)$, after three steps of the algorithm.

a_1 $-$								
$a_1 -$ $-b_1$			$a_1 a_2$ $- -$			$a_1 a_2$ $-b_1$		
$a_1 - -$ $-b_1 b_2$	$a_1 - a_2$ $-b_1 -$	$a_1 - a_1$ $-b_1 b_2$	$a_1 a_2 -$ $- - b_1$	$a_1 a_2 a_2$ $- - -$	$a_1 a_2 a_2$ $- - b_1$	$a_1 a_2 -$ $-b_1 b_2$	$a_1 a_2 a_2$ $-b_1 -$	$a_1 a_2 a_2$ $-b_1 b_2$

Table 4. Alignment generated from the column (a_1, b_1), after three steps of the algorithm.

a_1 b_1								
$a_1 -$ $b_1 b_2$			$a_1 a_2$ $b_1 -$			$a_1 a_2$ $b_1 b_2$		
$a_1 - -$ $b_1 b_2 b_3$	$a_1 - a_2$ $b_1 b_2 -$	$a_1 - a_2$ $b_1 b_2 b_3$	$a_1 a_2 -$ $b_1 - b_2$	$a_1 a_2 a_3$ $b_1 - -$	$a_1 a_2 a_3$ $b_1 - b_2$	$a_1 a_2 -$ $b_1 b_2 b_3$	$a_1 a_2 a_3$ $b_1 b_2 -$	$a_1 a_2 a_3$ $b_1 b_2 b_3$

1.4.1 'All the alignments' algorithm

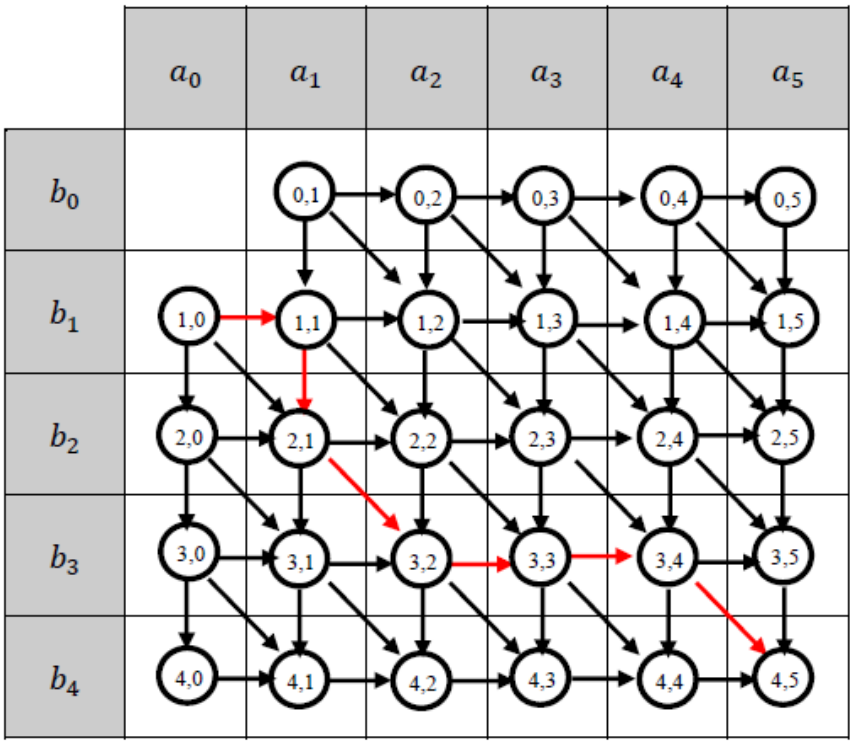
Let us consider two amino acids sequences: peptide $A = \{a_1, a_2, \dots, a_k\}$ and peptide $B = \{b_1, b_2, \dots, b_m\}$, where *k* and *m* are two integers, in general not equal. If we assume that amino acids are numerated from the N-terminal end, and we start the alignment from the N-terminus, then for the

first column we have three possible choices: we can align a gap for sequence *A* with amino acid b_1 for the other one $(-, b_1)$; we can introduce amino acid a_1 for sequence *A* and a gap for sequence *B* $(a_1, -)$; we can as the last choice align a_1 with $b_1 (a_1, b_1)$. The alignment of two gaps doesn't have sense, and it is thus not contemplated. In each of these alignments, we have three choices for the second column. If we consider first alignment, we can add the column $(-, b_2)$, the column $(a_1, -)$ or the column (a_1, b_2). And so forth. This alignment building goes on until all amino acids from the two sequences have been used. In Table 2 you can see the nine alignments generated for the first column $(-, b_1)$, after three steps of the algorithm. In Table 3 you have the first three steps of the alignment building for the first column $(a_1, -)$. In Table 4 you have the development when the first column is (a_1, b_1). As you can see, if we have n_i alignments in step i^{th}, the algorithm will generate $3n_i$ alignments in step $i + 1^{th}$. The two longest alignments have $k + m$ elements, and are indicated in Table 5. They have the same score, and have to be considered equivalent.

Table 5. The two longest alignment generated by the algorithm. They both have k+m columns.

Sequence *A*	$- - \quad - a_1 a_2 \quad a_k$
Sequence *B*	$b_1 b_1 \cdots b_m - - \cdots -$
Sequence *A*	$a_1 a_2 \quad a_k - - \quad -$
Sequence *B*	$- - \cdots - b_1 \; b_1 \cdots b_m$

Figure 1. Arrows indicate all the possible paths from the N-terminal of both peptides, to the C-terminals. In red one of those paths.



Another way to visualize this algorithm is through the matrix in Figure 1 where all possible alignments are reported for a sequence *A* of *5* amino acids and a sequence *B* of *4* amino acids. It is how to read that matrix:

- each step of the algorithm is represented by an arrow;
- each diagonal arrow represents a step in which a column with amino acids in both sequences are added;
- each vertical arrow represents a step in which a gap is added for sequence *A*;

- each horizontal arrow represents a step in which a gap is added for sequence *B*;
- each element of the matrix (*node*) represent a column of the alignment;
- element *i,j* represent a column where b_i and a_j are present if you arrive to this node through a diagonal arrow; if you arrive through a vertical arrow there is a gap instead of a_j, whereas if you arrive through a horizontal one, a gap instead of b_i will be present.

Red arrows in Figure 1 represent one of the possible alignment. This alignment is the following one:

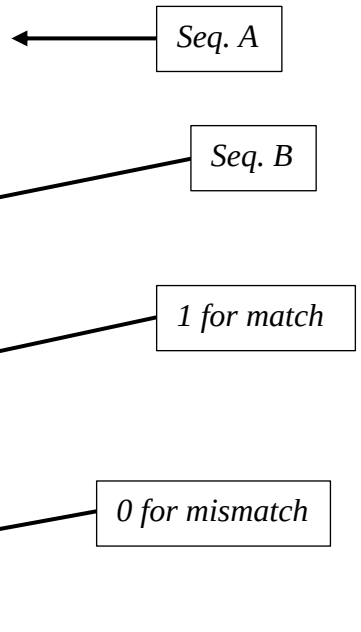
$-$	a_1	$-$	a_2	a_3	a_4	a_5
b_1	$-$	b_2	b_3	$-$	$-$	b_4

1.4.2 Needleman-Wunsch algorithm

Let us consider two amino acids sequences: peptide $A = \{a_1, a_2, \dots, a_k\}$ and peptide $B = \{b_1, b_2, \dots, b_m\}$, where *k* and *m* are two integers in general not equal. We display all the possible pairs $b_i a_j$ in a matrix with *m* rows and *k* columns. This array was called *MAT* by Needleman and Wunsch, and in Table 6 we present this array for the example from the original work by Needleman and Wunsch (Needleman, et al., 1970)↕. The value for the element $MAT(i,j)$ is the element (i,j) of the simplest substitution matrix, the identity matrix. In other words, we have $MAT(i,j) = 1$ if $b_i = a_j$ and $MAT(i,j) = 1$ if $b_i \neq a_j$.

Table 6. Array MAT for the example of the original paper of 1970, by Needleman and Wunsch. We have a value of one for a match and a value zero for a mismatch. Zeros have not been reported. In orange you can see one possible path, which represent a possible alignment between peptides A and B.

	A	B	C	N	J	R	O	C	L	C	R	P	M
A	1												
J					1								
C			1					1		1			
J					1								
N				1									
R						1					1		
C			1					1		1			
K													
C			1					1		1			
R						1					1		
B		1											
P												1	



Any possible alignment is a series of elements of array *MAT*, which we will refer to as $A/B = \{MAT(a,b), \dots, MAT(i,j), \dots, MAT(y,z)\}$ (being faithful to the notation of the original paper), where the following requirements have to be fulfilled:

1) $a \geq 1$ AND $b \geq 1$;
2) $a = 1$ OR $b = 1$;
3) $y \leq m$ AND $x \leq k$;

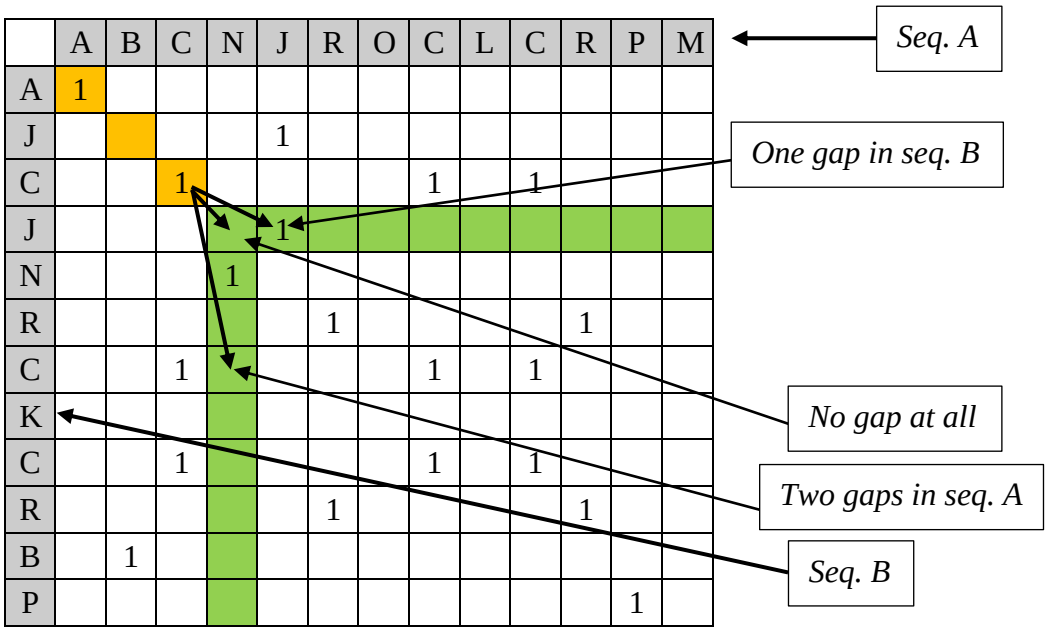
4) if $MAT(h,n) \in A/B$ and $MAT(h,n) \neq MAT(i,j)$ then $h < i$ and $n < j$ or $h > i$ and $n > j$;
5) both *i* and *j* must increase in value, with one of them that increases by one, and the other that increases by one or more.

Requirement number 2 means that the first element of the alignment must belong to row one OR to column one (OR as a logic operator!). Such an alignment is called a *necessary alignment*. Any alignment which doesn't fulfill requirement number 3, represent a permutation of two amino acids in one or both peptides. When an index increases of two, we have a gap: if it is *i*, then the gap is in sequence *A*; if it is in *j*, the gap is in sequence *B*. In Table 2 you have a possible alignment in yellow. This alignment is the following one:

Seq A	A	B	C	-	N	J	-	R	O	-	L	-	C	R	P	M
Seq B	A	J	C	J	N	R	C	K	C	C	R	B	P	-	-	-
Score	1	0	1	-p	1	0	-p	0	0	-p	0	-p	0	-p	-p	p

In Table 7 you can see in green all the possible steps that can be moved from element *C-C*. If we move on the principal diagonal, to element *J-N*, we have a column with no gap. If we move to one of the element of the green row, we will add one or more gaps to the sequence *B*; one gap if we move to *J-J*, two if we move to *J-R*, and so on. If we move to one of the elements of the green columns, we will add one or more gaps to sequence *A*; for instance, we will add Two gaps if we move to *C-N*.

Table 7. Array MAT for the example of the original paper of 1970, by Needleman and Wunsch. In orange the first three columns of one of the possible alignments. In green you have all the possible steps that can be moved from element C-C. If we move from C-C to J-N we will have no gap; if we move to element J-J we will have a gap in sequence B. If we move to element C-N, we will have two gaps in sequence A. And so on.



An alignment can thus be seen as a pathway through the elements of array *MAT*. The diagonal pathway corresponds to an alignment without gaps. This kind of alignment is possible only if *k=m*. Otherwise we will always have gaps, even if only at the end of the alignment, or at the beginning. To give a score to an alignment, we follow those rules:

1) we add 1 for each match;

2) we add 0 for each mismatch;
3) we subtract a penalty for each gap.

Table 8. Building of the last two rows of the trace back matrix. In yellow we have the element on which we are operating, in green those elements between which we have to calculate the maximum value.

	A	B	C	N	J	R	O	C	L	C	R	P	M
A	1												
J					1								
C			1					1		1			
J					1								
N				1									
R						1					1		
C			1					1		1			
K													
C			1					1		1			
R						1					1		
B		1									1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 9. Building of the fourth last row of the trace back matrix. In yellow we have the element on which we are operating, in green those elements between which we have to calculate the maximum value. The last three rows have been already filled with their values.

	A	B	C	N	J	R	O	C	L	C	R	P	M
A	1												
J					1								
C			1					1		1			
J					1								
N				1									
R						1					1		
C			1					1		1			
K													
C			1					1	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
B	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Penalties have to be defined. They could be a function of the number of gaps, for instance. Score matrices other than the identity matrix can be used. The alignment given by this algorithm as the answer to the query, is the one with the highest score. Let's see now how to define this (or those) alignment, the so-called *maximum-match pathway*, in case of a gap penalties of zero. The idea is to build a second matrix from the array *MAT* according to the following procedure:

1) For each cell $MAT(m-1, j)$, we define the maximum value from those of elements which can be reached from that cell, i.e. $MAT(m, l)$ where l goes from $j+1$ to k. Then we add this

value to the value in cell $MAT(m-1,j)$ and put this in the position $(m-1,j)$ of a new array, or write over matrix MAT. The new array is sometime *called trace back matrix.* In Table 4 we are evaluating the value to assign to element $MAT(11{,}10)$ (in yellow) while in green you can see the elements between which we have to calculate the maximum value to add to the value of $MAT(11{,}10)$, that in this case is zero.

2) For each cell $MAT(m-2,j)$, we operate as above, but in this case, we have to calculate the maximum value between elements $MAT(g,l)$, where g goes from $m-1$ to m and l goes from $j+1$ to k.
3) We repeat this algorithm for the remaining $m-2$ rows of array MAT. In Table 9and Table 10 you can see two further steps towards the building of trace back matrix.
4) Starting from the cell with the highest value in row m, we move to the cell with the highest value, which can be reached according to the rules discussed above. In this way, we obtain one or more alignment with the highest score. In Table 11 the complete trace back matrix with the two maximum match pathways.

Table 10. Building of the fifth last row of the trace back matrix. In yellow we have the element on which we are operating, in green those elements between which we have to calculate the maximum value. The last four rows have been already filled with their values.

	A	B	C	N	J	R	O	C	L	C	R	P	M
A	1												
J					1								
C			1					1		1			
J					1								
N				1									
R						1					1		
C			1					1		1			
K									3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
B	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

If we indicate with $S(i,j)$ the value of trace back matrix at position *i-j* and with $s(i,j)$ the score for the pair of amino acids *i,j*, then trace back matrix building algorithm described above can be put in a formal expression as follows:

Eq. 8 $$S(i,j) = s(i,j) + max\{S(i+1,j+1);\ \alpha(i,j);\ \beta(i,j)\}$$

where

Eq. 9 $$\begin{cases} \alpha(i,j) = max\{S(i+1,l) :\ l = j+2, j+3, \dots, m\} \\ \beta(i,j) = max\{S(g,j+1) :\ g = i+2, i+3, \dots, k\} \end{cases}$$

If we introduce gap penalties, the structure of the algorithm is the same as in Eq. 8, but $\alpha(i,j)$ and $\beta(i,j)$ have to be written in order to take in count also gap penalties, as follow:

Eq. 10

$$\begin{cases} \alpha(i,j) = max\{S(i+1,l) - w(l-j-1) : \ l = j+2, j+3, \dots, m\} \\ \beta(i,j) = max\{S(g,j+1) - w(g-i-1) : \ g = i+2, i+3, \dots, k\} \end{cases}$$

where *w* is a positive number which is a function of the length of the deletion. In Eq. 8, Eq. 9 and Eq. 10 we have used a notation introduced by Smith and Waterman (Smith, et al., 1981)↥.

Table 11. The complete trace back matrix from the example by Needleman and Wunsch in their original paper. In orange the maximum match pathway.

	A	B	C	N	J	R	O	C	L	C	R	P	M
A	8	7	6	6	5	4	4	3	3	2	1	0	0
J	7	7	6	6	6	4	4	3	3	2	1	0	0
C	6	6	7	6	5	4	4	4	3	3	1	0	0
J	6	6	6	5	6	4	4	3	3	2	1	0	0
N	5	5	5	6	5	4	4	3	3	2	1	0	0
R	4	4	4	3	4	5	4	3	3	2	2	0	0
C	3	3	4	3	3	3	3	4	3	3	1	0	0
K	3	3	3	3	3	3	3	3	3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
B	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 12. Results for the alignment between human beta hemoglobin and whale myoglobin from Needleman and Wunsch original work. In red I have highlighted a variable settings in which a gap penalty of 1.03 has been set and a score of 0.67 and of 0.33 for type two and type one pairs, respectively.

β-Hemoglobin–myoglobin maximum matches

Variable set	Match values for pair types 2	1	Penalty	Maximum-match value sum Real	Random†	s	Real X	Minimum deletions Real	Random†
1	0	0	0	63·00	55·60	1·80	4·11	35	36·2
2	0	0	1·00	38·00	27·80	2·09	4·88	4	5·5
3	0·67	0·33	0	97·00	91·47	1·55	3·57	18	24·3
4	0·67	0·33	1·03	89·63	80·25	1·11	8·46	1	3·6
5	0·25	0·05	0	71·55	64·78	1·59	4·27	46	45·0
6	0·25	0·05	1·05	51·95	40·54	1·46	7·80	3	7·5
7	0·25	0·05	25	47·30	33·80	1·52	8·87	0	0

Needleman and Wunsch wrote a code in FORTRAN for the CDC3400 computer, which used this algorithm on two pairs of homologous sequences: the first pair was *whale myoglobin* and *human β-hemoglobin*, the second was *bovine pancreatic ribonuclease* and *hen's egg lysozyme*. The operated with different settings, changing the penalty value for gaps from 0 to 25, and using different score matrices. In particular, although the score for a match was always one, and the score for a mismatch was always 0, the scores for pairs with two identical bases (*type two*) and for those with only one identical base (*type one*) were changed from one execution to the other. For example, in their fourth

variable setting the used a penalty gap of 1.03, a score of 0.67 for type 2 pairs and a score of 0.33 for type 1 pairs (Table 2).
If we start from the first row instead of the last one, we can write the algorithm in Eq. 8, Eq. 9, Eq. 10 as follows:

Eq. 11 $$S(i,j) = s(i,j) + max\{S(i-1,j-1);\ \alpha(i,j);\ \beta(i,j)\}$$

where

Eq. 12 $$\begin{cases}\alpha(i,j) = max\{S(i-1,l) - w(j-1-l):\ l = j-2, j-3, \dots, 1\} \\ \beta(i,j) = max\{S(g,j-1) - w(i-1-g):\ g = i-2, i-3, \dots, 1\}\end{cases}$$

with $S(0,0) = 0$, $S(0,j) = -w(j)$ and $S(i,0) = -w(i)$.

1.4.3 Needleman-Wunsch: a personal view

I will now propose a personal and an easier to implement view of the Needleman-Wunsch algorithm. We will use $MAT(i,j)$ for the generic element of *MAT* matrix, and $TBM(i,j)$ for the generic element of Trace Back Matrix. Keeping in mind Figure 1 and nomenclature introduced in paragraph 0, we have that TBM has *k+1* rows and *m+1* columns, as MAT does. We define the first row and the first column of TBM as follows:

Eq. 13 $$\begin{cases}TBM(0,0) = 0 \\ TBM(0,j) = -w(j) \\ TBM(i,0) = -w(i)\end{cases}$$

For the other $k \times m$ elements of MAT we have

Eq. 14 $$\begin{cases}TBM(i+1,j+1) = max\{TBM(i,j) + MAT(i+1,j+1);\ \alpha(i+1,j+1); \beta(i+1,j+1)\} \\ \quad \alpha(i+1,j+1) = max\{TBM(i+1;l) - w(j+1-l): l = 0,1,\dots,j\} \\ \quad \beta(i+1,j+1) = max\{TBM(g;j+1) - w(i+1-g): g = 0,1,\dots,i\}\end{cases}$$

It is also necessary to define first row and first column of MAT as follows:

Eq. 15 $$\begin{cases}MAT(0,j) = 0, \quad j = 0,1,2,\dots,k \\ MAT(i,0) = 0, \quad i = 1,2,\dots,m\end{cases}$$

1.4.4 Global alignment: an example

In paragraph 1.4.2, we have calculated the best alignments between two sequences, if the score matrix is 1 for match and -1 for mismatch, and if we have no penalty gap (Table 11). This example has been taken from the original paper by Needleman and Wunsch (Needleman, et al., 1970)[↑]. Let's now consider the introduction of a penalty gap of 2 and two shorter sequences, example from (Ewens, et al., 2005). Consider the two sequences in Table 13, where their matrix *MAT* for a scoring matrix given by identity matrix is also reported. The adaptation of the algorithm in *Eq. 13*, *Eq. 14* and *Eq. 15* gives

Eq. 16 $$\begin{cases}TBM(i+1,j+1) = max\{TBM(i,j) + MAT(i+1,j+1);\ \alpha(i+1,j+1); \beta(i+1,j+1)\} \\ \quad \alpha(i+1,j+1) = max\{TBM(i+1;l) - 2(j+1-l): l = 0,1,\dots,j\} \\ \quad \beta(i+1,j+1) = max\{TBM(g;j+1) - 2(i+1-g): g = 0,1,\dots,i\}\end{cases}$$

For the first row we have $TBM(0,\mathrm{j}) = -2 \cdot j$; for the first column we have $TBM(i,0) = -2 \cdot i$. For the second row ($i=1$) we have:

$\alpha(1,1) = \max\{\mathrm{TBM}(1,0) - 2\} = -4$
$\beta(1,1) = \max\{\mathrm{TBM}(0,1) - 2\} = -4$
$TBM(1,1) = \max\{TBM(0,0) + MAT(1,1), \alpha(1,1), \beta(1,1)\} = \max\{-1,-4,-4\} = -1$
from $TBM(0,0)$

$\alpha(1,2) = \max\{\mathrm{TBM}(1,0) - 4, \mathrm{TBM}(1,1) - 2\} = \max\{-6,-3\} = -3$
$\beta(1,2) = \max\{\mathrm{TBM}(0,2) - 2\} = -6$
$TBM(1,2) = \max\{TBM(0,1) + MAT(1,2), \alpha(1,2), \beta(1,2)\} = \max\{-3,-3,-6\} = -3$
from $TBM(0,1)$ and from $TBM(1,1)$

$\alpha(1,3) = \max\{\mathrm{TBM}(1,0) - 6, \mathrm{TBM}(1,1) - 4, \mathrm{TBM}(1,2) - 2\} = \max\{-8,-5,-5\} = -5$
$\beta(1,3) = \max\{\mathrm{TBM}(0,3) - 2\} = -8$
$TBM(1,3) = \max\{TBM(0,2) + MAT(1,3), \alpha(1,3), \beta(1,3)\} = \max\{-5,-5,-8\} = -5$
from $TBM(0,2)$, $\mathrm{TBM}(1,1)$, $\mathrm{TBM}(1,2)$

$\alpha(1,4) = \max\{\mathrm{TBM}(1,0) - 8, \mathrm{TBM}(1,1) - 6, \mathrm{TBM}(1,2) - 4, \mathrm{TBM}(1,3) - 2\} = \max\{-10,-7,-7,-7\} = -7$
$\beta(1,4) = \max\{\mathrm{TBM}(0,4) - 2\} = -10$
$TBM(1,4) = \max\{TBM(0,3) + MAT(1,4), \alpha(1,3), \beta(1,3)\} = \max\{-7,-7,-10\} = -7$
from $\mathrm{TBM}(1,1)$, $\mathrm{TBM}(1,2)$, $\mathrm{TBM}(1,3)$, $TBM(0,3)$

For row $i=2$ we have what follows:

$\alpha(2,1) = \max\{\mathrm{TBM}(2,0) - 2\} = -6$
$\beta(2,1) = \max\{\mathrm{TBM}(0,1) - 4, \mathrm{TBM}(1,1) - 2\} = \max\{-6,-3\} = -3$
$TBM(2,1) = \max\{TBM(1,0) + MAT(2,1), \alpha(1,1), \beta(1,1)\} = \max\{-3,-6,-3\} = -3$
from $\mathrm{TBM}(1,1)$, $TBM(1,0)$

$\alpha(2,2) = \max\{\mathrm{TBM}(2,0) - 4, \mathrm{TBM}(2,1) - 2\} = \max\{-8,-5\} = -5$
$\beta(2,2) = \max\{\mathrm{TBM}(0,2) - 4, \mathrm{TBM}(1,2) - 2\} = \max\{-8,-5\} = -5$
$TBM(2,2) = \max\{TBM(1,1) + MAT(2,2), \alpha(1,1), \beta(1,1)\} = \max\{0,-5,-5\} = 0$
from $TBM(1,1)$

$\alpha(2,3) = \max\{\mathrm{TBM}(2,0) - 6, \mathrm{TBM}(2,1) - 4, \mathrm{TBM}(2,2) - 2\} = \max\{-10,-7,-2\} = -2$
$\beta(2,3) = \max\{\mathrm{TBM}(0,3) - 4, \mathrm{TBM}(1,3) - 2\} = \max\{-10,-7\} = -7$
$TBM(2,3) = \max\{TBM(1,2) + MAT(2,3), \alpha(2,3), \beta(2,3)\} = \max\{-4,-2,-7\} = -2$
from $\mathrm{TBM}(2,2)$

$\alpha(2,4) = \max\{\mathrm{TBM}(2,0) - 8, \mathrm{TBM}(2,1) - 6, \mathrm{TBM}(2,2) - 4, \mathrm{TBM}(2,3) - 2\} = \max\{-12,-9,-4,-4\} = -4$
$\beta(2,4) = \max\{\mathrm{TBM}(0,4) - 4, \mathrm{TBM}(1,4) - 2\} = \max\{-12,-9\} = -9$
$TBM(2,4) = \max\{TBM(1,3) + MAT(2,4), \alpha(1,3), \beta(1,3)\} = \max\{-6,-4,-9\} = -4$
from $\mathrm{TBM}(2,2)$, $\mathrm{TBM}(2,3)$

For row $i=3$ we have what follows:

$\alpha(3,1) = \max\{\mathrm{TBM}(3,0) - 2\} = -8$
$\beta(3,1) = \max\{\mathrm{TBM}(0,1) - 6, \mathrm{TBM}(1,1) - 4, \mathrm{TBM}(2,1) - 2\} = \max\{-8,-5,-5\} = -5$
$TBM(3,1) = \max\{TBM(2,0) + MAT(3,1), \alpha(1,1), \beta(1,1)\} = \max\{-5,-8,-5\} = -5$
from $\mathrm{TBM}(1,1)$, $\mathrm{TBM}(2,1)$, $TBM(2,0)$

$\alpha(3,2) = \max\{\mathrm{TBM}(3,0) - 4, \mathrm{TBM}(3,1) - 2\} = \max\{-10,-7\} = -7$
$\beta(3,2) = \max\{\mathrm{TBM}(0,2) - 6, \mathrm{TBM}(1,2) - 4, \mathrm{TBM}(2,2) - 2,\} = \max\{-10,-7,-2\} = -2$
$TBM(3,2) = \max\{TBM(2,1) + MAT(3,2), \alpha(1,1), \beta(1,1)\} = \max\{-2,-7,-2\} = -2$
from $\mathrm{TBM}(2,2)$, $\mathrm{TBM}(2,1)$

$\alpha(3,3) = \max\{\text{TBM}(3,0) - 6, \text{TBM}(3,1) - 4, \text{TBM}(3,2) - 2\} = \max\{-12, -9, -4\} = -4$
$\beta(3,3) = \max\{\text{TBM}(0,3) - 6, \text{TBM}(1,3) - 4, \text{TBM}(2,3) - 2\} = \max\{-12, -9, -4\} = -4$
$TBM(3,3) = \max\{TBM(2,2) + MAT(3,3), \alpha(3,3), \beta(3,3)\} = \max\{-1, -4, -4\} = -1$
from $\text{TBM}(2,2)$

$\alpha(3,4) = \max\{\text{TBM}(3,0) - 8, \text{TBM}(3,1) - 6, \text{TBM}(3,2) - 4, \text{TBM}(3,3) - 2\} = \max\{-14, -11, -6, -3\} = -3$
$\beta(3,4) = \max\{\text{TBM}(0,4) - 6, \text{TBM}(1,4) - 4, \text{TBM}(2,4) - 2\} = \max\{-14, -11, -6\} = -6$
$TBM(3,4) = \max\{TBM(2,3) + MAT(3,4), \alpha(3,4), \beta(3,4)\} = \max\{-3, -3, -6\} = -3$
from $\text{TBM}(3,3)$, $\text{TBM}(2,3)$

For row *i=4* we have what follows:

$\alpha(4,1) = \max\{\text{TBM}(4,0) - 2\} = -10$
$\beta(4,1) = \max\{\text{TBM}(0,1) - 8, \text{TBM}(1,1) - 6, \text{TBM}(2,1) - 4, \text{TBM}(3,1) - 2\} = \max\{-10, -7, -7, -7\} = -7$
$TBM(4,1) = \max\{TBM(3,0) + MAT(4,1), \alpha(4,1), \beta(4,1)\} = \max\{-7, -10, -7\} = -7$
from $\text{TBM}(1,1), \text{TBM}(2,1), \text{TBM}(3,1), TBM(3,0)$

$\alpha(4,2) = \max\{\text{TBM}(4,0) - 4, \text{TBM}(4,1) - 2\} = \max\{-12, -9\} = -9$
$\beta(4,2) = \max\{\text{TBM}(0,2) - 8, \text{TBM}(1,2) - 6, \text{TBM}(2,2) - 4, \text{TBM}(3,2) - 2\} = \max\{-14, -11, -6, -3\} = -3$
$TBM(4,2) = \max\{TBM(3,2) + MAT(4,3), \alpha(4,2), \beta(4,2)\} = \max\{-1, -9, -3\} = -1$
from $\text{TBM}(3,2)$

$\alpha(4,3) = \max\{\text{TBM}(4,0) - 6, \text{TBM}(4,1) - 4, \text{TBM}(4,2) - 2\} = \max\{-14, -11, -6\} = -6$
$\beta(4,3) = \max\{\text{TBM}(0,3) - 8, \text{TBM}(1,3) - 6, \text{TBM}(2,3) - 4, , \text{TBM}(3,3) - 2\} = \max\{-12, -9, -4\} = -4$
$TBM(3,3) = \max\{TBM(2,2) + MAT(3,3), \alpha(4,3), \beta(4,3)\} = \max\{-1, -4, -4\} = -1$
from $\text{TBM}(2,2)$

$\alpha(4,4) = \max\{\text{TBM}(4,0) - 8, \text{TBM}(4,1) - 6, \text{TBM}(4,2) - 4, \text{TBM}(4,3) - 2\} = \max\{-16, -13, -8, -3\} = -3$
$\beta(4,4) = \max\{\text{TBM}(0,4) - 8, \text{TBM}(1,4) - 6, \text{TBM}(2,4) - 4, \text{TBM}(3,4) - 2\} = \max\{-16, -13, -8, -5\} = -5$
$TBM(4,4) = \max\{TBM(3,3) + MAT(4,4), \alpha(4,4), \beta(4,4)\} = \max\{0, -3, -5\} = 0$
from $\text{TBM}(3,3)$

For row *i=5* we have what follows:

$\alpha(5,1) = \max\{\text{TBM}(5,0) - 2\} = -12$
$\beta(5,1) = \max\{\text{TBM}(0,1) - 10, \text{TBM}(1,1) - 8, \text{TBM}(2,1) - 6, \text{TBM}(3,1) - 4, \text{TBM}(4,1) - 2\} = \max\{-12, -9, -9, -9, -9\} = -9$
$TBM(5,1) = \max\{TBM(4,0) + MAT(5,1), \alpha(5,1), \beta(5,1)\} = \max\{-7, -12, -9\} = -7$
from $\text{TBM}(4,0)$

$\alpha(5,2) = \max\{\text{TBM}(5,0) - 4, \text{TBM}(5,1) - 2\} = \max\{-14, -9\} = -9$
$\beta(5,2) = \max\{\text{TBM}(0,2) - 10, \text{TBM}(1,2) - 8, \text{TBM}(2,2) - 6, \text{TBM}(3,2) - 4, \text{TBM}(4,2) - 2\} = \max\{-14, -11, -6, -6, -6\} = -6$
$TBM(5,2) = \max\{TBM(4,2) + MAT(5,3), \alpha(5,2), \beta(5,2)\} = \max\{-8, -9, -6\} = -6$
from $\text{TBM}(2,2), \text{TBM}(3,2), \text{TBM}(4,2)$

$\alpha(5,3) = \max\{\text{TBM}(5,0) - 6, \text{TBM}(5,1) - 4, \text{TBM}(5,2) - 2\} = \max\{-16, -11, -8\} = -8$
$\beta(5,3) = \max\{\text{TBM}(0,3) - 10, \text{TBM}(1,3) - 8, \text{TBM}(2,3) - 6, \text{TBM}(3,3) - 4, \text{TBM}(4,3) - 2\} = \max\{-16, -13, -8, -5, -3\} = -3$
$TBM(5,3) = \max\{TBM(4,2) + MAT(5,3), \alpha(5,3), \beta(5,3)\} = \max\{-5, -8, -3\} = -3$
from $\text{TBM}(4,3)$

$\alpha(5,4) = \max\{\text{TBM}(5,0) - 8, \text{TBM}(5,1) - 6, \text{TBM}(5,2) - 4, \text{TBM}(5,3) - 2\} = \max\{-18, -13, -10, -5\} = -5$
$\beta(5,4) = \max\{\text{TBM}(0,4) - 10, \text{TBM}(1,4) - 8, \text{TBM}(2,4) - 6, \text{TBM}(3,4) - 4, \text{TBM}(4,4) - 2\} = \max\{-18, -15, -10, -7, -2\} = -2$
$TBM(5,4) = \max\{TBM(4,3) + MAT(5,4), \alpha(5,4), \beta(5,4)\} = \max\{-2, -5, -2\} = -2$
from $\text{TBM}(4,3), \text{TBM}(4,4)$

For row *i=6* we have what follows:

$\alpha(6,1) = \max\{\text{TBM}(6,0) - 2\} = -14$
$\beta(6,1) = \max\{\text{TBM}(0,1) - 12, \text{TBM}(1,1) - 10, \text{TBM}(2,1) - 8, \text{TBM}(3,1) - 6, \text{TBM}(4,1) - 4, \text{TBM}(5,1) - 2\} =$

$= \max\{-14, -11, -11, -11, -11, -9\} = -9$
$TBM(6,1) = \max\{TBM(5,0) + MAT(6,1), \alpha(6,1), \beta(6,1)\} = \max\{-11, -14, -9\} = -9$
from $\mathrm{TBM}(5,1)$

$\alpha(6,2) = \max\{\mathrm{TBM}(6,0) - 4, \mathrm{TBM}(6,1) - 2\} = \max\{-16, -11\} = -11$
$\beta(6,2) = \max\{\mathrm{TBM}(0,2) - 12, \mathrm{TBM}(1,2) - 10, \mathrm{TBM}(2,2) - 8, \mathrm{TBM}(3,2) - 6, \mathrm{TBM}(4,2) - 4, \mathrm{TBM}(5,2) - 2\} =$
$= \max\{-16, -13, -8, -8, -8, -8\} = -8$
$TBM(6,2) = \max\{TBM(5,1) + MAT(6,2), \alpha(6,2), \beta(6,2)\} = \max\{-8, -11, -8\} = -8$
from $\mathrm{TBM}(2,2), \mathrm{TBM}(3,2), \mathrm{TBM}(4,2), \mathrm{TBM}(5,2), TBM(5,1)$

$\alpha(6,3) = \max\{\mathrm{TBM}(6,0) - 6, \mathrm{TBM}(6,1) - 4, \mathrm{TBM}(6,2) - 2\} = \max\{-18, -13, -10\} = -10$
$\beta(6,3) = \max\{\mathrm{TBM}(0,3) - 12, \mathrm{TBM}(1,3) - 10, \mathrm{TBM}(2,3) - 8, \mathrm{TBM}(3,3) - 6, \mathrm{TBM}(4,3) - 4, \mathrm{TBM}(5,3) - 2\} =$
$= \max\{-18, -15, -10, -7, -5, -5\} = -5$
$TBM(6,3) = \max\{TBM(5,2) + MAT(6,3), \alpha(6,3), \beta(6,3)\} = \max\{-5, -10, -5\} = -5$
from $TBM(5,2), \mathrm{TBM}(4,3), \mathrm{TBM}(5,3)$

$\alpha(6,4) = \max\{\mathrm{TBM}(6,0) - 8, \mathrm{TBM}(6,1) - 6, \mathrm{TBM}(6,2) - 4, \mathrm{TBM}(6,3) - 2\} = \max\{-20, -15, -12, -7\} = -7$
$\beta(6,4) = \max\{\mathrm{TBM}(0,4) - 12, \mathrm{TBM}(1,4) - 10, \mathrm{TBM}(2,4) - 8, \mathrm{TBM}(3,4) - 6, \mathrm{TBM}(4,4) - 4, \mathrm{TBM}(5,4) - 2\} =$
$= \max\{-20, -17, -12, -11, -6, -4\} = -4$
$TBM(6,4) = \max\{TBM(5,3) + MAT(6,4), \alpha(6,4), \beta(6,4)\} = \max\{-2, -7, -4\} = -2$
from $\mathrm{TBM}(5,3)$

Table 13. Matrix MAT (on the left) for a scoring matrix given by identity matrix. Trace back matrix (TBM, on the right) for a score matrix given by 1 for match and -1 for mismatch, and a gap penalty of 2.

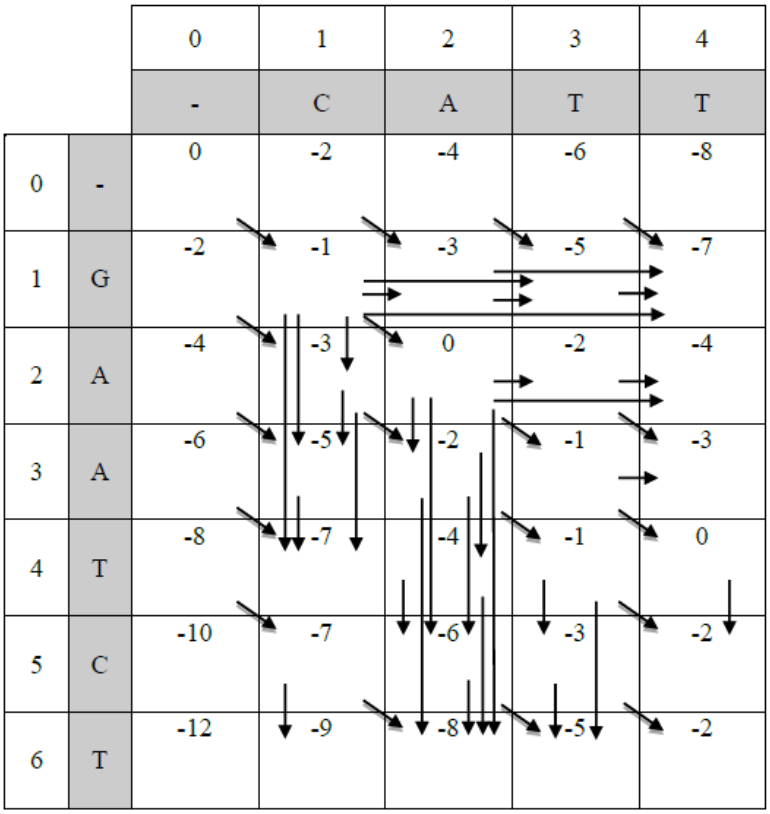
		0	1	2	3	4
		-	C	A	T	T
0	-	0	0	0	0	0
1	G	0	-1	-1	-1	-1
2	A	0	-1	1	-1	-1
3	A	0	-1	1	-1	-1
4	T	0	-1	-1	1	1
5	C	0	1	-1	-1	-1
6	T	0	-1	-1	1	1

		0	1	2	3	4
		-	C	A	T	T
0	-	0	-2	-4	-6	-8
1	G	-2	-1	-3	-5	-7
2	A	-4	-3	0	-2	-4
3	A	-6	-5	-2	-1	-3
4	T	-8	-7	-4	-1	0
5	C	-10	-7	-6	-3	-2
6	T	-12	-9	-8	-5	-2

We have thus built the trace back matrix in Table 13 (on the right). In order to identify the best alignment (or the best ones), we have to keep memory of where each element of TBM comes from, as in *Figure 2*. Each alignment which ends in element $TBM(6,4)$ has a score of -2 (which is the best score in this case). Each elements of the last column and of the last row of *TBM* represent the score of one or more best alignments, but only the element $TBM(6,4)$ gives the real score, as for each other element of the last column we have to add the penalties due to the gaps needed to complete the alignment; the same is true for each other element of the last row. If you consider, for instance, the alignment in yellow in *Figure 3*, you have to add to the score -3, the penalties for the further three gaps you have to add in order to align the last 3 amino acids of sequence B. The alignment and its score is:

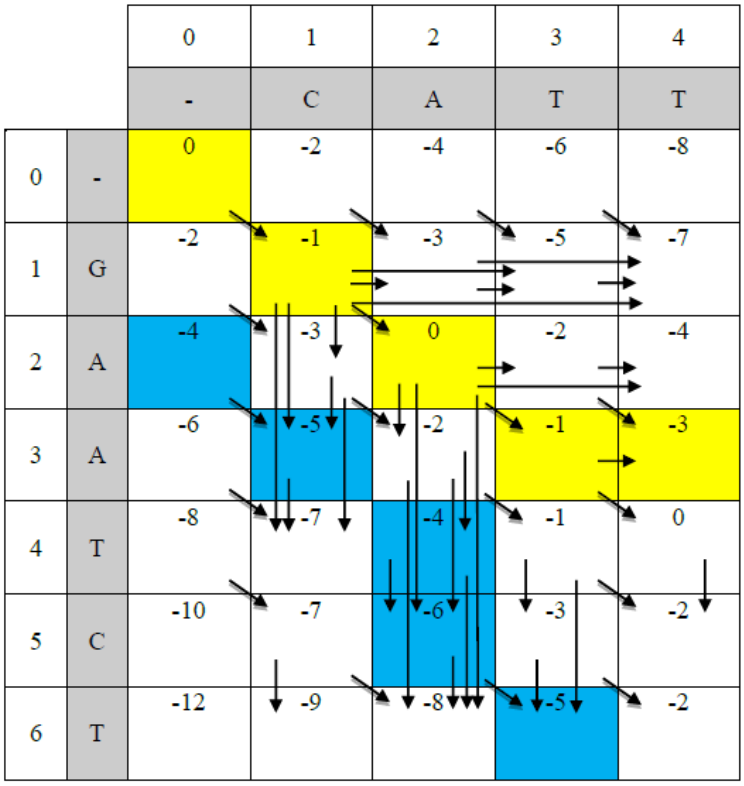
Seq A	G	A	T	T	-	-	-
Seq B	C	A	A	-	T	C	T
Score	-1	1	-1	-2	-2	-2	-2	-9

Figure 2. Trace back matrix with registration of where each element comes from.



		0	1	2	3	4
		-	C	A	T	T
0	-	0	-2	-4	-6	-8
1	G	-2	-1	-3	-5	-7
2	A	-4	-3	0	-2	-4
3	A	-6	-5	-2	-1	-3
4	T	-8	-7	-4	-1	0
5	C	-10	-7	-6	-3	-2
6	T	-12	-9	-8	-5	-2

Figure 3. Trace back matrix with registration of where each element comes from, and with two best alignments highlighted.



		0	1	2	3	4
		-	C	A	T	T
0	-	0	-2	-4	-6	-8
1	G	-2	-1	-3	-5	-7
2	A	-4	-3	0	-2	-4
3	A	-6	-5	-2	-1	-3
4	T	-8	-7	-4	-1	0
5	C	-10	-7	-6	-3	-2
6	T	-12	-9	-8	-5	-2

If you consider alignment in blue, on the other hand, you have to add to the score -5 a penalty of -2 due to a further gap you have to align with the last amino acid of sequence A. The complete alignment and its score is

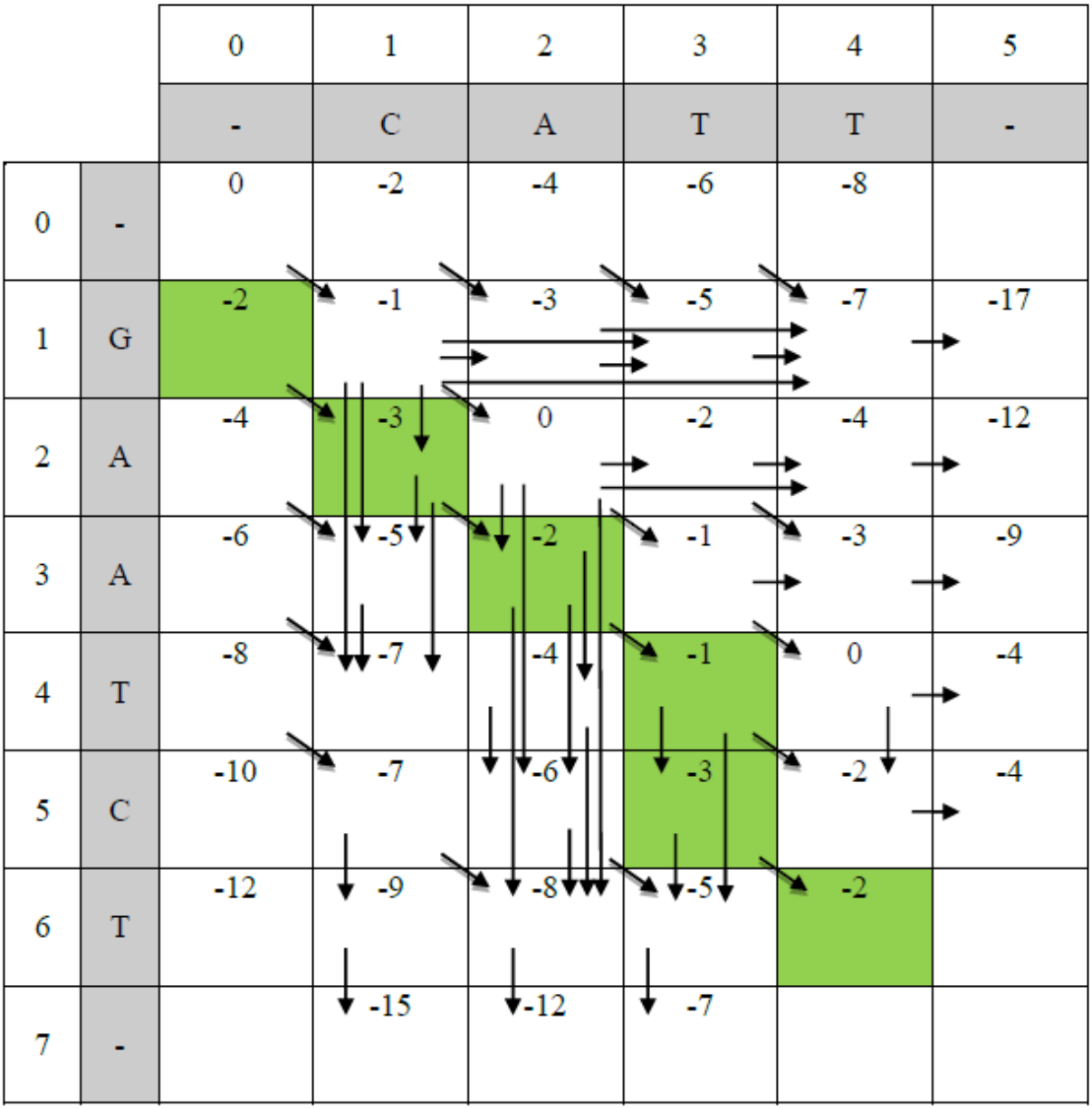
Seq A	-	C	A	-	T	T	
Seq B	A	A	T	C	T	-	
Score	-2	-1	-1	-2	1	-2	-7

Thus, the complete score matrix is the one in *Figure 4* where you can see alignments with the best score are those who end in element $TBM(6,4)$, with a score of -2, and they are three. The first one is highlighted in

Figure 4. Trace back matrix with registration of where each element comes from. Further elements have been added in order to take into account penalties due to gaps at the end of the alignments.

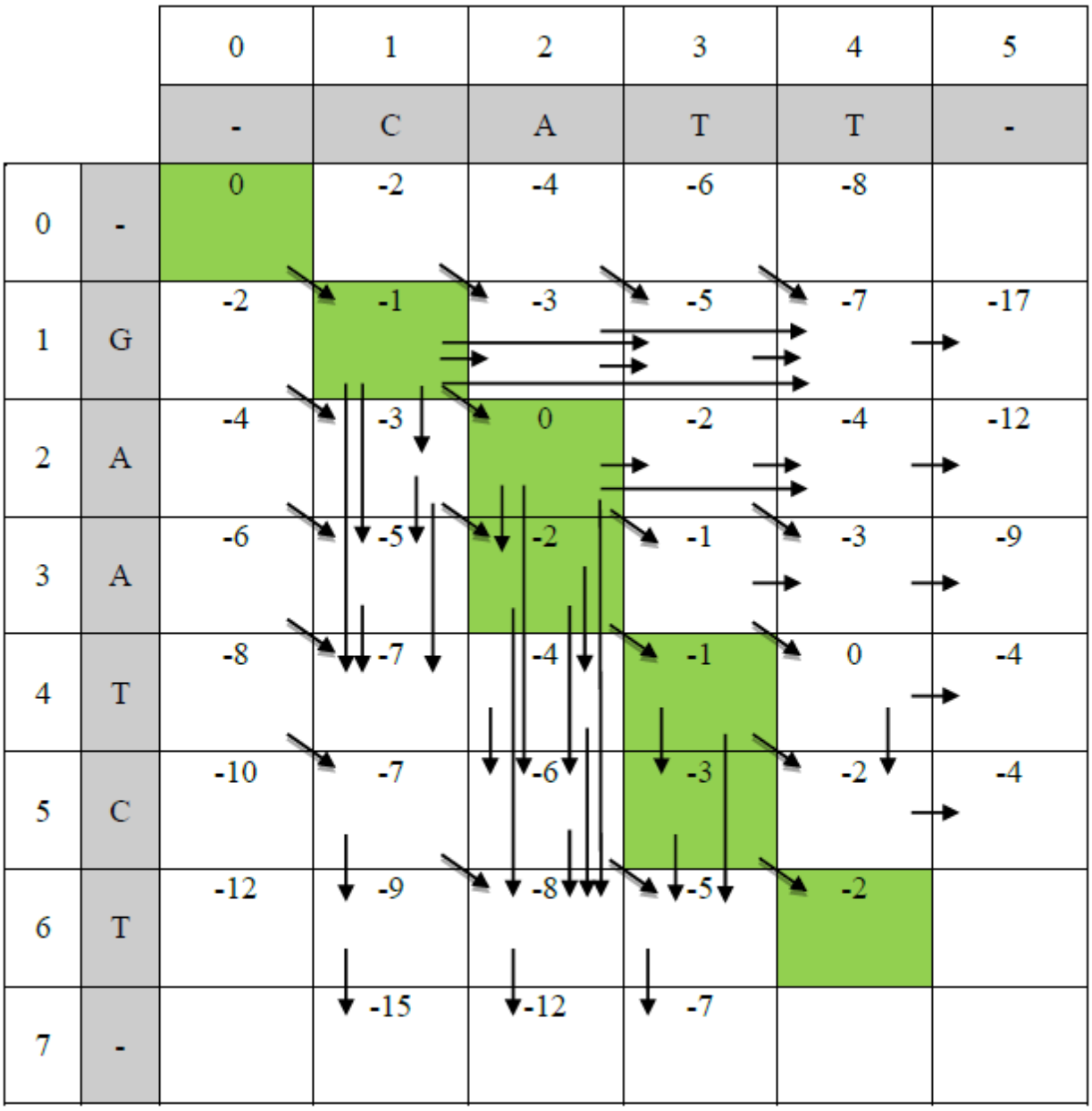
		0	1	2	3	4	5
		-	C	A	T	T	-
0	-	0	-2	-4	-6	-8	
1	G	-2	-1	-3	-5	-7	-17
2	A	-4	-3	0	-2	-4	-12
3	A	-6	-5	-2	-1	-3	-9
4	T	-8	-7	-4	-1	0	-4
5	C	-10	-7	-6	-3	-2	-4
6	T	-12	-9	-8	-5	-2	
7	-		-15	-12	-7		

Figure 5. One of the three alignments with the best score.



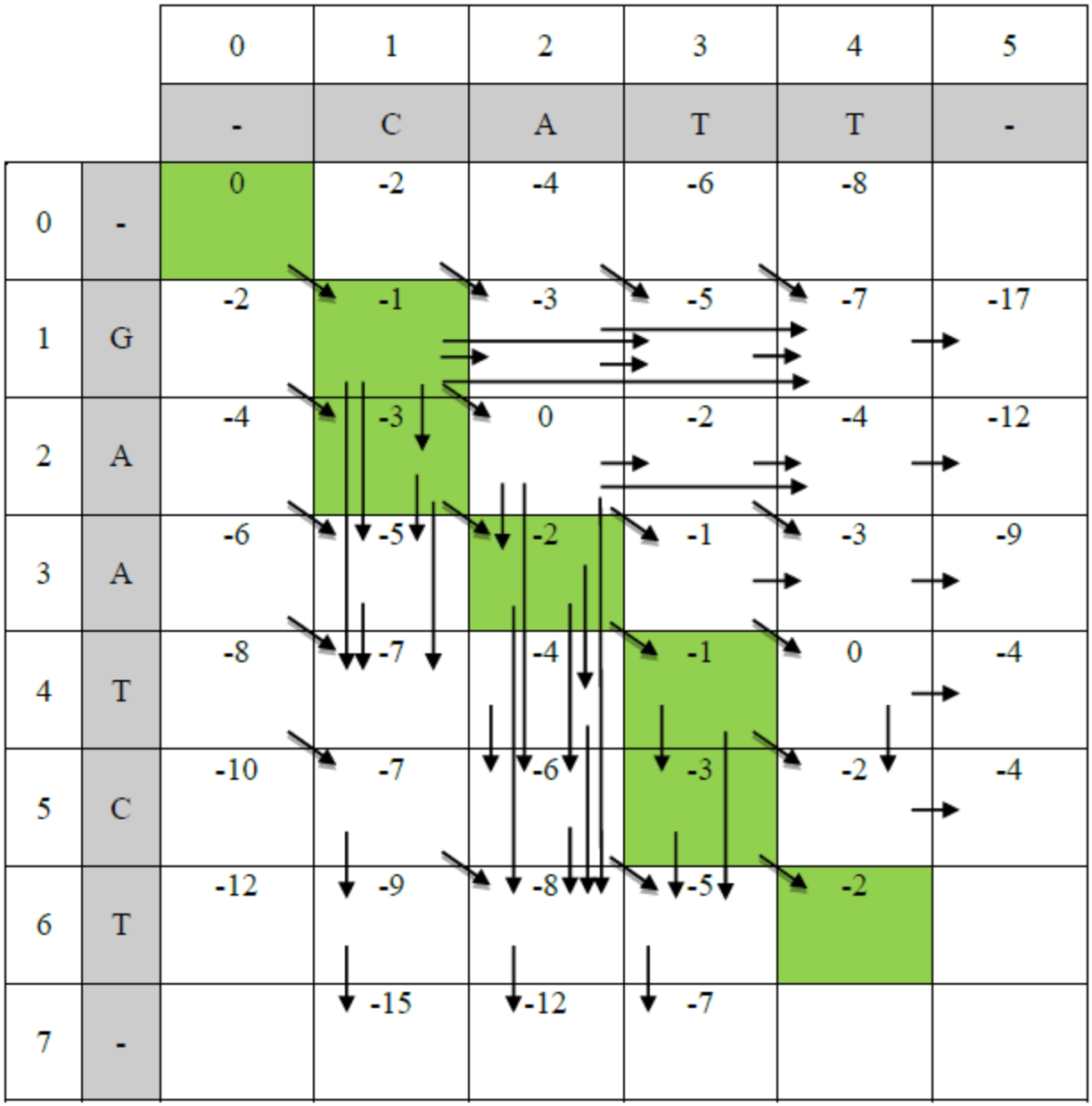
Seq A	-	C	A	T	_	T	
Seq B	G	A	A	T	C	T	
Score	-2	-1	1	1	-2	1	= -2

Figure 6. One of the three alignments with the best score.



Seq A	C	A	-	T	-	T	
Seq B	G	A	A	T	C	T	
Score	-1	1	-2	1	-2	1	-2

Figure 7. One of the three alignments with the best score.



Seq A	C	-	A	T	-	T	
Seq B	G	A	A	T	C	T	
Score	-1	-2	1	1	-2	1	-2

According to paragraph 1.3 the number of possible alignments for this example is

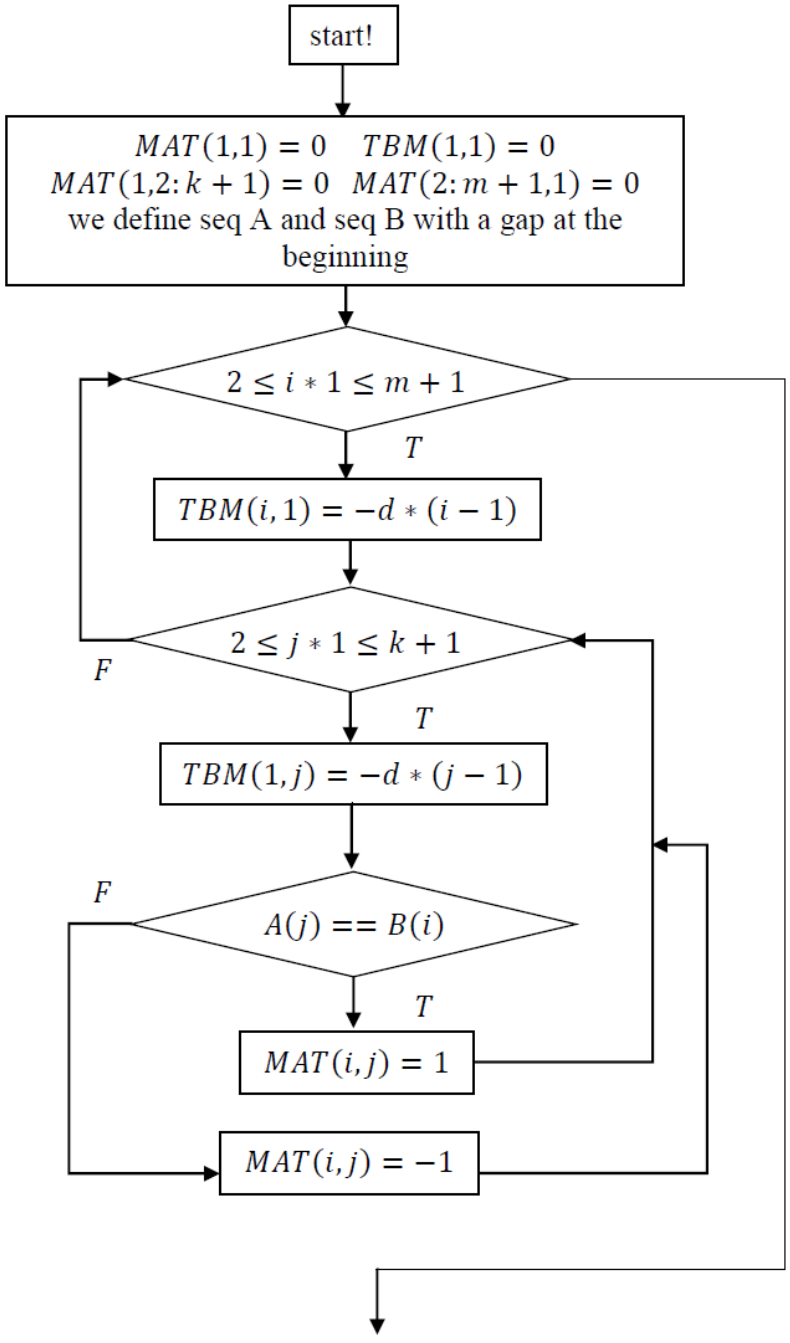
Eq. 17 $$\eta(6,4) \geq \binom{10}{4} = \frac{14!}{6!4!} = 5.045.040$$

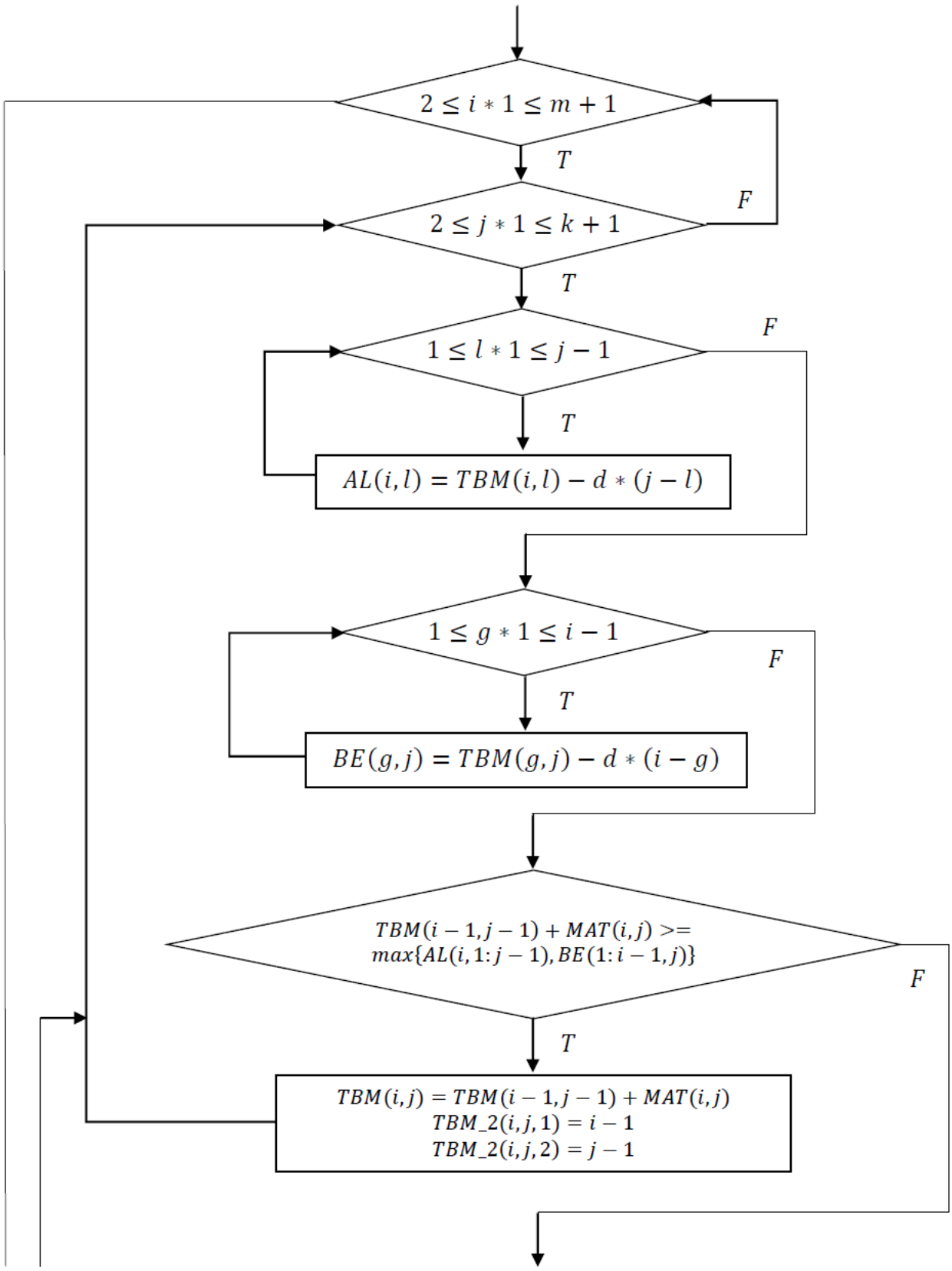
Thus, it is very interesting the possibility to determine the best alignment with only $6 \cdot 4 = 24$ steps. This example gives a sense of the worth of dynamic programming.

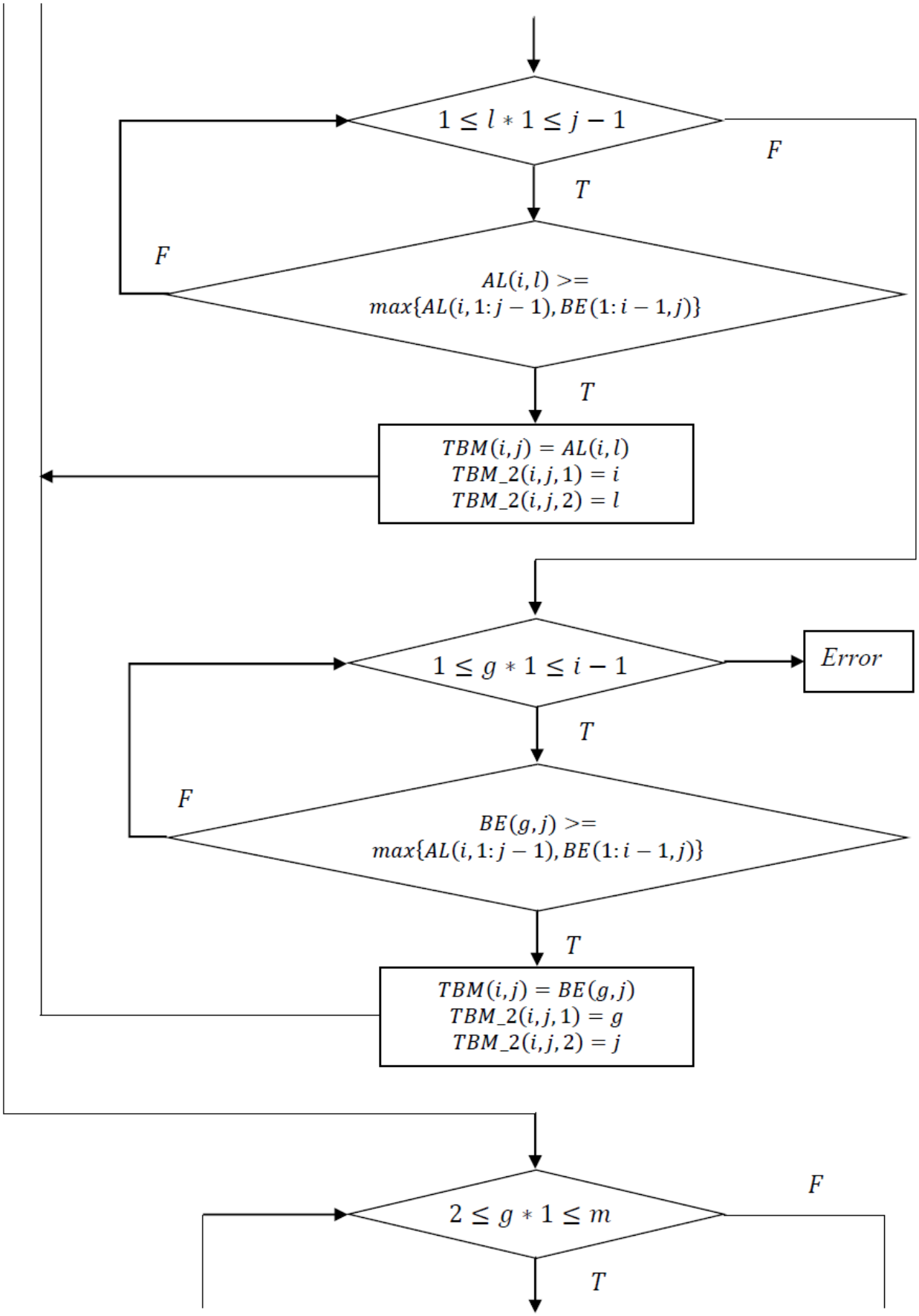
1.4.5 Code in Octave for Needleman-Wunsch algorithm

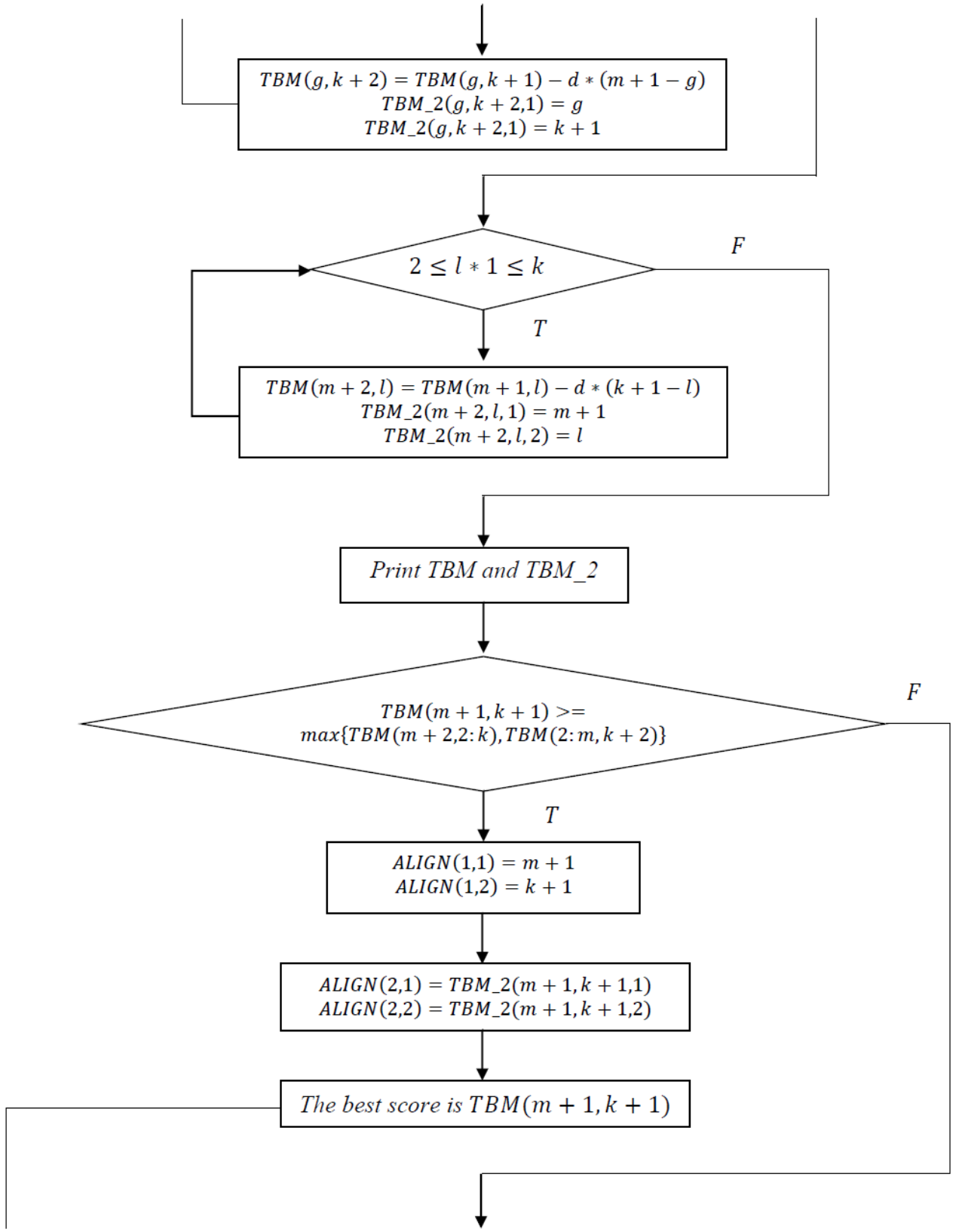
The following code in Octave implements Neeelman-Wunsch algorithm as described in paragraph 1.4.3, with a score system that display 1 for a match and -1 for a mismatch. We also use a liner gap model, with a penalty of -2 for each gap, included gaps at the end of the alignment. The program runs on the same example solved by hand in paragraph 1.4.4.
As we are interested not only in the best score between two peptides, but also in the alignment which leads to the best score, this program introduce a second trace back matrix (called TBM_2) which keeps record of where each best score pair comes from. The flow diagram for the code is in *Figure 8*. It contains some mistakes regard to the search for best alignments which have end gaps. Those mistakes have been emended in the code.

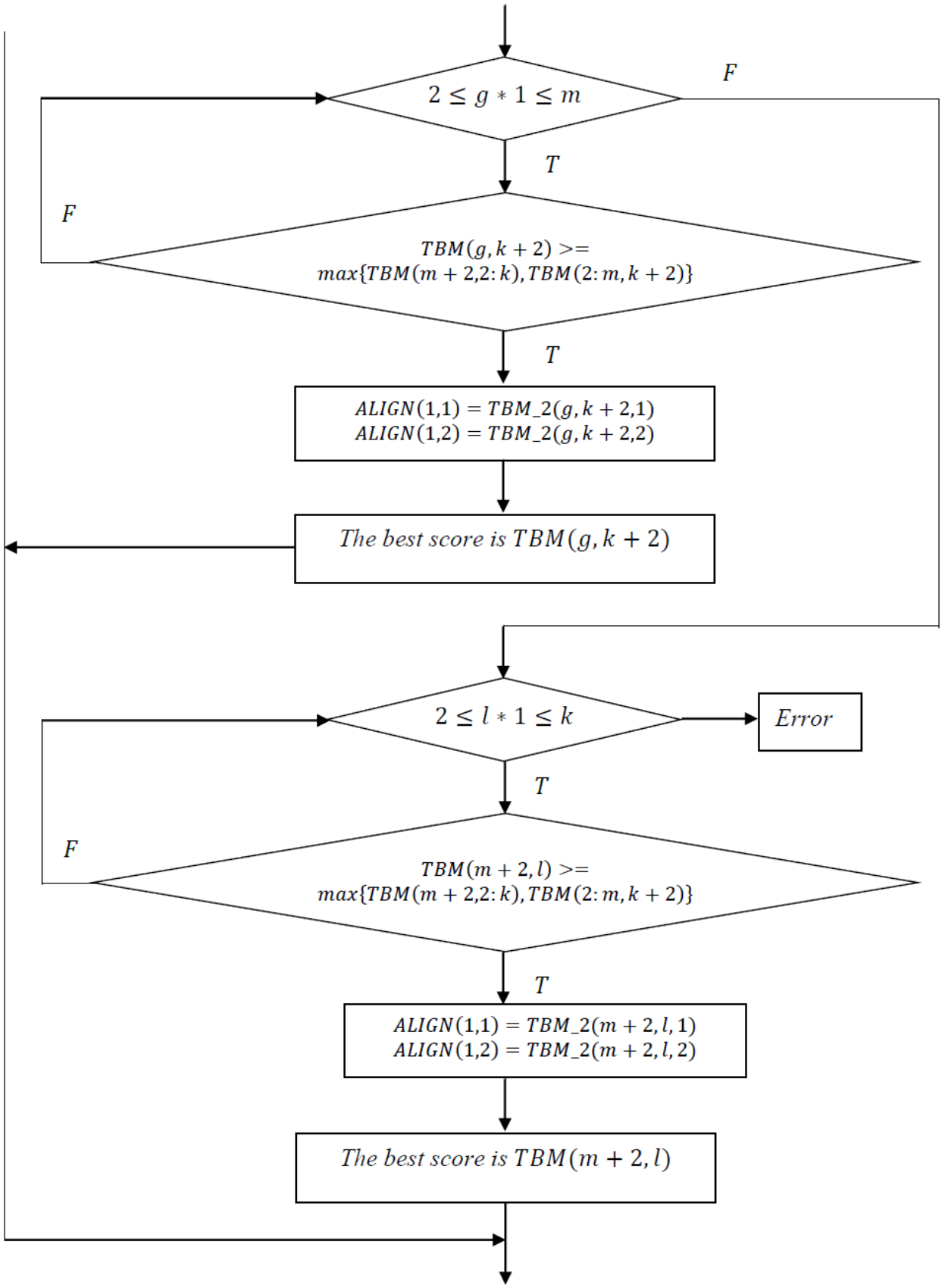
Figure 8. Flow diagram for the code which runs Needleman-Wunsch algorithm with a linear gap penalty and a simple substitution matrix.

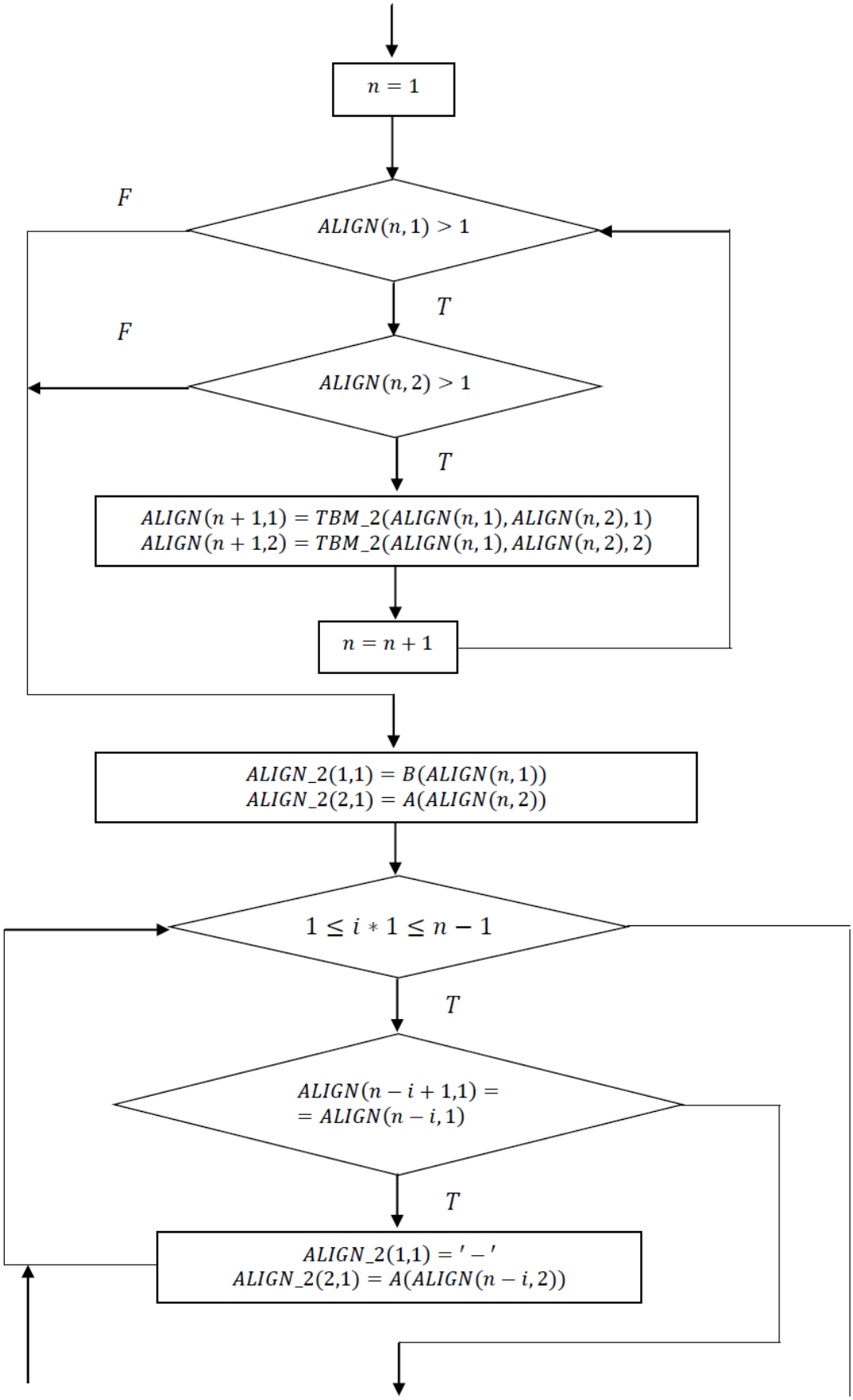


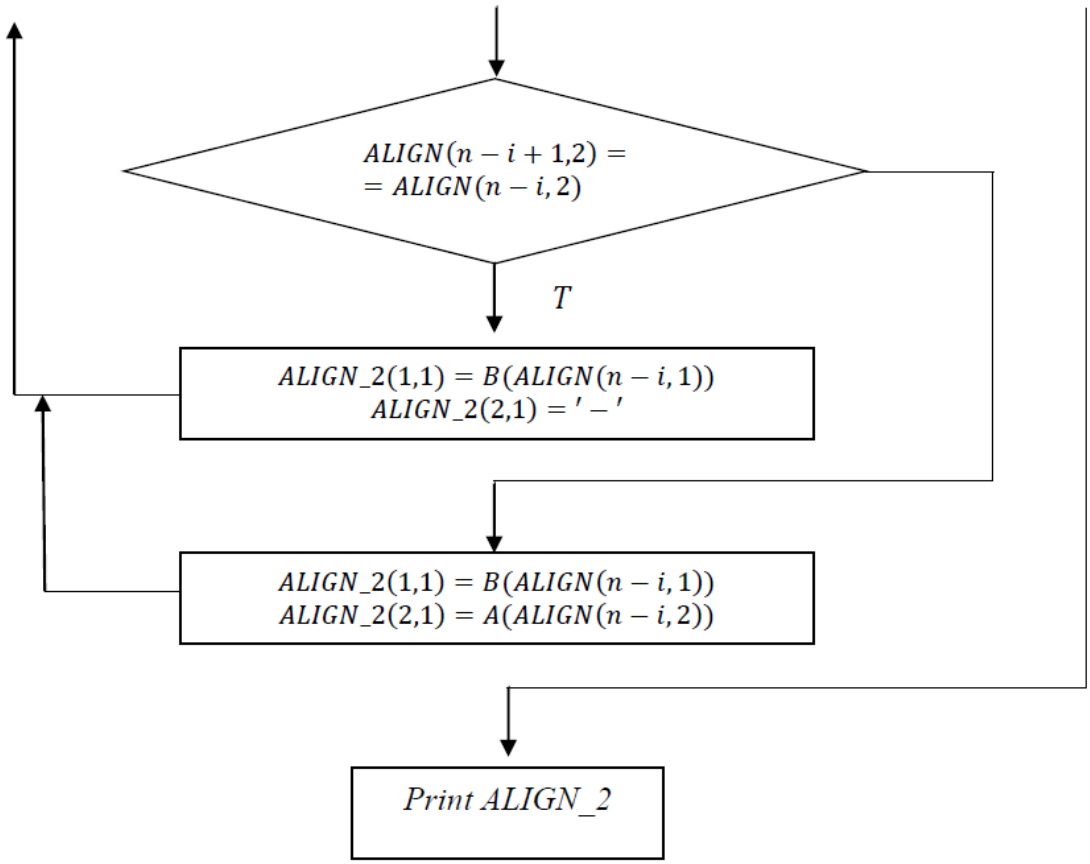

$2 \le i * 1 \le m + 1$
T
F
$2 \le j * 1 \le k + 1$
T
F
$1 \le l * 1 \le j - 1$
T
$AL(i,l) = TBM(i,l) - d * (j - l)$
$1 \le g * 1 \le i - 1$
F
T
$BE(g,j) = TBM(g,j) - d * (i - g)$
$TBM(i-1,j-1) + MAT(i,j) >=$
$max\{AL(i,1{:}j-1), BE(1{:}i-1,j)\}$
F
T
$TBM(i,j) = TBM(i-1,j-1) + MAT(i,j)$
$TBM\_2(i,j,1) = i - 1$
$TBM\_2(i,j,2) = j - 1$


$1 \le l * 1 \le j - 1$
F
T
F
$AL(i,l) >= max\{AL(i,1{:}j-1), BE(1{:}i-1,j)\}$
T
$TBM(i,j) = AL(i,l)$
$TBM\_2(i,j,1) = i$
$TBM\_2(i,j,2) = l$
$1 \le g * 1 \le i - 1$
Error
T
F
$BE(g,j) >= max\{AL(i,1{:}j-1), BE(1{:}i-1,j)\}$
T
$TBM(i,j) = BE(g,j)$
$TBM\_2(i,j,1) = g$
$TBM\_2(i,j,2) = j$
$2 \le g * 1 \le m$
F
T


$TBM(g,k+2) = TBM(g,k+1) - d * (m+1-g)$
$TBM\_2(g,k+2,1) = g$
$TBM\_2(g,k+2,1) = k+1$
$2 \le l * 1 \le k$
F
T
$TBM(m+2,l) = TBM(m+1,l) - d * (k+1-l)$
$TBM\_2(m+2,l,1) = m+1$
$TBM\_2(m+2,l,2) = l$
Print TBM and TBM_2
$TBM(m+1,k+1) >=$
$max\{TBM(m+2,2:k), TBM(2:m,k+2)\}$
F
T
$ALIGN(1,1) = m+1$
$ALIGN(1,2) = k+1$
$ALIGN(2,1) = TBM\_2(m+1,k+1,1)$
$ALIGN(2,2) = TBM\_2(m+1,k+1,2)$
The best score is $TBM(m+1,k+1)$


$2 \le g * 1 \le m$
F
T
F
$TBM(g, k+2) >= max\{TBM(m+2,2:k), TBM(2:m, k+2)\}$
T
$ALIGN(1,1) = TBM\_2(g, k+2,1)$
$ALIGN(1,2) = TBM\_2(g, k+2,2)$
The best score is $TBM(g, k+2)$
$2 \le l * 1 \le k$
Error
T
F
$TBM(m+2, l) >= max\{TBM(m+2,2:k), TBM(2:m, k+2)\}$
T
$ALIGN(1,1) = TBM\_2(m+2, l, 1)$
$ALIGN(1,2) = TBM\_2(m+2, l, 2)$
The best score is $TBM(m+2, l)$


$n = 1$
F
$ALIGN(n,1) > 1$
T
F
$ALIGN(n,2) > 1$
T
$ALIGN(n+1,1) = TBM\_2(ALIGN(n,1), ALIGN(n,2), 1)$
$ALIGN(n+1,2) = TBM\_2(ALIGN(n,1), ALIGN(n,2), 2)$
$n = n + 1$
$ALIGN\_2(1,1) = B(ALIGN(n,1))$
$ALIGN\_2(2,1) = A(ALIGN(n,2))$
$1 \le i * 1 \le n - 1$
T
$ALIGN(n-i+1,1) =$
$= ALIGN(n-i,1)$
T
$ALIGN\_2(1,1) = ' - '$
$ALIGN\_2(2,1) = A(ALIGN(n-i,2))$



The code in Octave which implement this flow diagram is the following one, while the execution of the program will be discussed in the paragraph which follows.

```
% file name = NeW_1.m
% date of creation = 28/03/2016
clear
% We define length of sequence A and of sequence B, respectively
k=4;
m=6;
% We define the penalty for gap
d=2;
% We define amino acids for sequence A
A(1:k+1)= ['-' 'c' 'a' 't' 't'];
% We define amino acids for sequence B
B(1:m+1)= ['-' 'g' 'a' 'a' 't' 'c' 't'];
% We define elements of MAT, TBM and TBM_2 which are zero
MAT(1,1)=0;
MAT(1,2:k+1)=0;
MAT(2:m+1,1)=0;
TBM(1,1)=0;
% We define elements of TBM and TBM_2 which have no role in calculation
% with a formal value of 100
TBM(1,k+2)=100;
TBM(m+2,1)=100;
TBM(m+1,k+2)=100;
TBM(m+2,k+2)=100;
TBM(m+2,k+1)=100;
TBM_2(1,1:k+2,1:2)=100;
```

```
TBM_2(1:m+2,1,1:2)=100;
TBM_2(m+1,k+2,1:2)=100;
TBM_2(m+2,k+2,1:2)=100;
TBM_2(m+2,k+1,1:2)=100;
% We define other elements of the first row and the first column of TBM
for i=2:1:m+1
 TBM(i,1)=-1*d*(i-1);
 for j=2:1:k+1
  TBM(1,j)=-1*d*(j-1);
  if ( A(j)==B(i) )
   MAT(i,j)=1;
  else
   MAT(i,j)=-1;
  endif
 endfor
endfor
% We print array MAT on the screen
disp(MAT)
% We define elements for TBM and TBM_2
for i=2:1:m+1
 for j=2:1:k+1
  for l=1:1:j-1
   AL(i,l)=TBM(i,l)-d*(j-l);
  endfor
  for g=1:1:i-1
   BE(g,j)=TBM(g,j)-d*(i-g);
  endfor
  if ( TBM(i-1,j-1)+MAT(i,j) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
   TBM(i,j)=TBM(i-1,j-1)+MAT(i,j);
   TBM_2(i,j,1)=i-1;
   TBM_2(i,j,2)=j-1;
  else
   for l=1:1:j-1
    if ( AL(i,l) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
     TBM(i,j)=AL(i,l);
     TBM_2(i,j,1)=i;
     TBM_2(i,j,2)=l;
    else
     for g=1:1:i-1
      if ( BE(g,j) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
       TBM(i,j)=BE(g,j);
       TBM_2(i,j,1)=g;
       TBM_2(i,j,2)=j;
      endif
     endfor
    endif
   endfor
  endif
 endfor
endfor
% We define elements for the last column of TBM and TBM_2
for g=2:1:m
 TBM(g,k+2)=TBM(g,k+1) - (d*(m+1-g));
 TBM_2(g,k+2,1)=g;
 TBM_2(g,k+2,2)=k+1;
endfor
% We define elements for the last row of TBM and TBM_2
for l=2:1:k
 TBM(m+2,l)=TBM(m+1,l) - (d*(k+1-l));
 TBM_2(m+2,l,1)=m+1;
```

```
 TBM_2(m+2,l,2)=l;
endfor
% We print arrays TBM and TBM_2
disp(TBM)
disp(TBM_2)
% We put into array ALIGN the sequence of elements of A and B
% which give the best alignment
% Here we start with the first row of ALIGN
%
%
control=0;
if ( TBM(m+1,k+1) >= max([max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
  ALIGN_a(1:2,1:1001)=0;
  ALIGN_a(1,1)= k+1;
  ALIGN_a(2,1)= m+1;
  best_score= TBM(m+1,k+1);
  control=1;
endif
if (control==1)
 % Now we define the following rows of ALIGN
 for n=1:1:1000
  if ( ALIGN_a(1,n)>1 )
   if ( ALIGN_a(2,n)>1 )
    ALIGN_a(1,n+1)=TBM_2(ALIGN_a(2,n),ALIGN_a(1,n),2);
    ALIGN_a(2,n+1)=TBM_2(ALIGN_a(2,n),ALIGN_a(1,n),1);
   endif
  endif
 endfor
 % We search for the numebr of elements of ALIGN
 n=1;
 while (ALIGN_a(1,n)>1)
  n=n+1;
 endwhile
 n=n-1;
 % We calculate the array ALIGN_a_2 which contains the alignment
 ALIGN_a_2(1,1)=A(ALIGN_a(1,n));
 ALIGN_a_2(2,1)=B(ALIGN_a(2,n));
 for i=1:1:n-1
  if ( ALIGN_a(1,n-i)==ALIGN_a(1,n-i+1) )
   ALIGN_a_2(1,i+1)='-';
   ALIGN_a_2(2,i+1)=B(ALIGN_a(2,n-i));
  elseif ( ALIGN_a(2,n-i)==ALIGN_a(2,n-i+1) )
   ALIGN_a_2(1,i+1)=A(ALIGN_a(1,n-i));
   ALIGN_a_2(2,i+1)='-';
  else
   ALIGN_a_2(1,i+1)=A(ALIGN_a(1,n-1));
   ALIGN_a_2(2,i+1)=B(ALIGN_a(2,n-i));
  endif
 endfor
 disp(" ")
 disp("One of the possible alignments with the best score is:")
 disp(" ")
 disp(["seq. A: " ALIGN_a_2(1,1:n)])
 disp(["seq. B: " ALIGN_a_2(2,1:n)])
 disp(" ")
 disp("The path through TBM, from the end to the beginning, is:")
 disp(" ")
 disp(ALIGN_a(1:2,1:n))
 disp(" ")
 % We print on the screen the best score
```

```
 disp(" ")
 disp('The best score for alignments is:')
 disp(" ")
 disp(best_score)
 disp(" ")
endif
% We searh for another best alignment
%
%
control=0;
for g=2:1:m
 if ( TBM(g,k+2) >= max([TBM(m+1,k+1) max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
  ALIGN_b(1:1000,1:2)=0;
  ALIGN_b(1,1)= TBM_2(g,k+2,1);
  ALIGN_b(1,2)= TBM_2(g,k+2,2);
  best_score= TBM(g,k+2);
  control=1
 endif
endfor
if (control==1)
 % Now we define the following rows of ALIGN
 for n=1:1:1000
  if ( ALIGN_b(1,n)>1 )
   if ( ALIGN_b(2,n)>1 )
    ALIGN_b(1,n+1)=TBM_2(ALIGN_b(2,n),ALIGN_b(1,n),2);
    ALIGN_b(2,n+1)=TBM_2(ALIGN_b(2,n),ALIGN_b(1,n),1);
   endif
  endif
 endfor
 % We search for the numebr of elements of ALIGN
 n=1;
 while (ALIGN_b(1,n)>1)
  n=n+1;
 endwhile
 n=n-1;
 % We calculate the array ALIGN_a_2 which contains the alignment
 ALIGN_b_2(1,1)=A(ALIGN_b(1,n));
 ALIGN_b_2(2,1)=B(ALIGN_b(2,n));
 for i=1:1:n-2
  if ( ALIGN_b(1,n-i)==ALIGN_b(1,n-i+1) )
   ALIGN_b_2(1,i+1)='-';
   ALIGN_b_2(2,i+1)=B(ALIGN_b(2,n-i));
  elseif ( ALIGN_b(2,n-i)==ALIGN_b(2,n-i+1) )
   ALIGN_b_2(1,i+1)=A(ALIGN_b(1,n-i));
   ALIGN_b_2(2,i+1)='-';
  else
   ALIGN_b_2(1,i+1)=A(ALIGN_b(1,n-1));
   ALIGN_b_2(2,i+1)=B(ALIGN_b(2,n-i));
  endif
 endfor
 for i=1:1:m-ALIGN_b(2,1)+1
  ALIGN_b_2(1,n+i-1)="-";
  ALIGN_b_2(2,n+i-1)=B(m+1-(m-ALIGN_b(2,1)+1)+i);
 endfor
 disp(" ")
 disp("One of the possible alignments with the best score is:")
 disp(" ")
 disp(["seq. A: " ALIGN_b_2(1,1:n)])
 disp(["seq. B: " ALIGN_b_2(2,1:n)])
 disp(" ")
```

```
 disp("The path through TBM, from the end to the beginning, is:")
 disp(" ")
 disp(ALIGN_b(1:2,1:n))
 disp(" ")
 % We print on the screen the best score
 disp(" ")
 disp('The best score for alignments is:')
 disp(" ")
 disp(best_score)
 disp(" ")
endif
% We searh for another best alignment
%
%
control=0;
for l=2:1:k
 if ( TBM(m+2,l) >= max([TBM(m+1,k+1) max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
  ALIGN_c(1:2,1:1001)=0;
  ALIGN_c(1,1)= TBM_2(m+2,l,2);
  ALIGN_c(2,1)= TBM_2(m+2,l,1);
  best_score= TBM(m+2,l);
  control=1;
 endif
endfor
if (control==1)
 % Now we define the following rows of ALIGN
 for n=1:1:1000
  if ( ALIGN_c(1,n)>1 )
   if ( ALIGN_c(2,n)>1 )
    ALIGN_c(1,n+1)=TBM_2(ALIGN_c(2,n),ALIGN_c(1,n),2);
    ALIGN_c(2,n+1)=TBM_2(ALIGN_c(2,n),ALIGN_c(1,n),1);
   endif
  endif
 endfor
 % We search for the numebr of elements of ALIGN
 n=1;
 while (ALIGN_c(1,n)>1)
  n=n+1;
 endwhile
 n=n-1;
 % We calculate the array ALIGN_a_2 which contains the alignment
 ALIGN_c_2(1,1)=A(ALIGN_c(1,n));
 ALIGN_c_2(2,1)=B(ALIGN_c(2,n));
 for i=1:1:n-2
  if ( ALIGN_c(1,n-i)==ALIGN_c(1,n-i+1) )
   ALIGN_c_2(1,i+1)='-';
   ALIGN_c_2(2,i+1)=B(ALIGN_c(2,n-i));
  elseif ( ALIGN_c(2,n-i)==ALIGN_c(2,n-i+1) )
   ALIGN_c_2(1,i+1)=A(ALIGN_c(1,n-i));
   ALIGN_c_2(2,i+1)='-';
  else
   ALIGN_c_2(1,i+1)=A(ALIGN_c(1,n-1));
   ALIGN_c_2(2,i+1)=B(ALIGN_c(2,n-i));
  endif
 endfor
 for i=1:1:k-ALIGN_c(1,1)+1
  ALIGN_c_2(1,n+i-1)=A(k+1-(k-ALIGN_c(1,1)+1)+i);
  ALIGN_c_2(2,n+i-1)="-";
 endfor
 disp(" ")
```

```
 disp("One of the possible alignments with the best score is:")
 disp(" ")
 disp(["seq. A: " ALIGN_c_2(1,1:n)])
 disp(["seq. B: " ALIGN_c_2(2,1:n)])
 disp(" ")
 disp("The path through TBM, from the end to the beginning, is:")
 disp(" ")
 disp(ALIGN_c(1:2,1:n))
 disp(" ")
 % We print on the screen the best score
 disp(" ")
 disp('The best score for alignments is:')
 disp(" ")
 disp(best_score)
 disp(" ")
endif
```

Figure 9. TBM_2 given by the code in Octave for the example in paragraph 1.4.4.

		1	2	3	4	5	6
		-	C	A	T	T	-
1	-	-	-	-	-	-	-
2	G	-	(1,1)	(1,2)	(1,3)	(1,4)	(2,5)
3	A	-	(2,1)	(2,2)	(3,3)	(3,4)	(3,5)
4	A	-	(3,1)	(3,2)	(3,3)	(3,4)	(4,5)
5	T	-	(4,1)	(4,3)	(4,3)	(4,4)	(5,5)
6	C	-	(5,1)	(5,3)	(5,4)	(5,4)	(6,5)
7	T	-	(6,2)	(6,2)	(6,3)	(6,4)	-
8	-	-	(7,2)	(7,3)	(7,4)	-	-

1.4.6 Execution of the code in Octave for Needleman-Wunsch algorithm

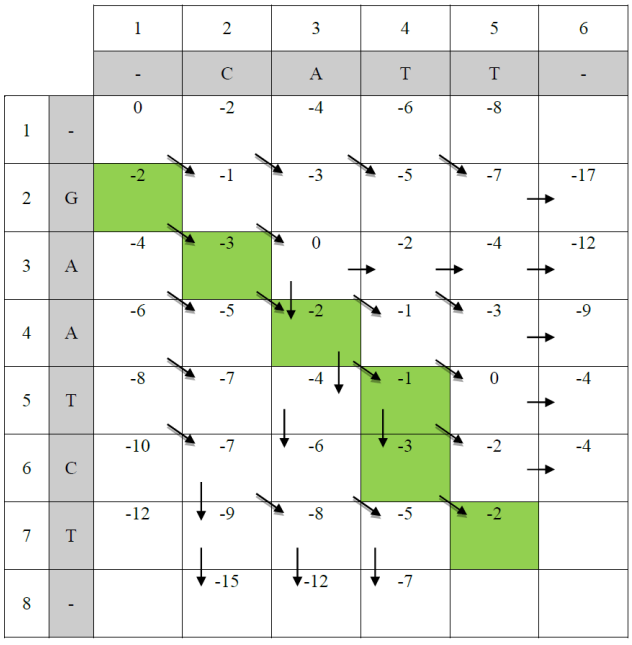
When the code in the previous paragraph runs on sequence A = CATT and sequence B = GAATCT, it gives as matrix MAT and TBM the ones calculated by hand and reported in Table 13, correctly. More exactly, the TBM given by the program is the extended version in *Figure 4*. But the program

gives also another matrix, the TBM_2, whose element *i,j* gives one of the pairs from which pair *i,j* comes from. It gives only one of the possible pairs with the highest scores for each element of TBM. At the end we will have from one to three of the alignment with the highest score. In **Errore. L 'origine riferimento non è stata trovata.** is the TBM_2 given by the program as output.

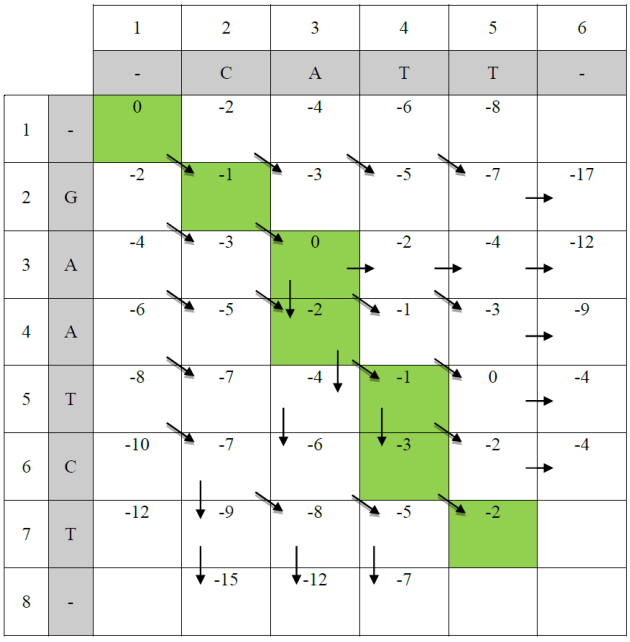
Figure 10. TBM with data from TBM_2.

		1	2	3	4	5	6
		-	C	A	T	T	-
1	-	0	-2	-4	-6	-8	-
2	G	-2	-1	-3	-5	-7	-17
3	A	-4	-3	0	-2	-4	-12
4	A	-6	-5	-2	-1	-3	-9
5	T	-8	-7	-4	-1	0	-4
6	C	-10	-7	-6	-3	-2	-4
7	T	-12	-9	-8	-5	-2	-
8	-	-	-15	-12	-7	-	-

Figure 11. The two best alignments deduced from TBM_2.



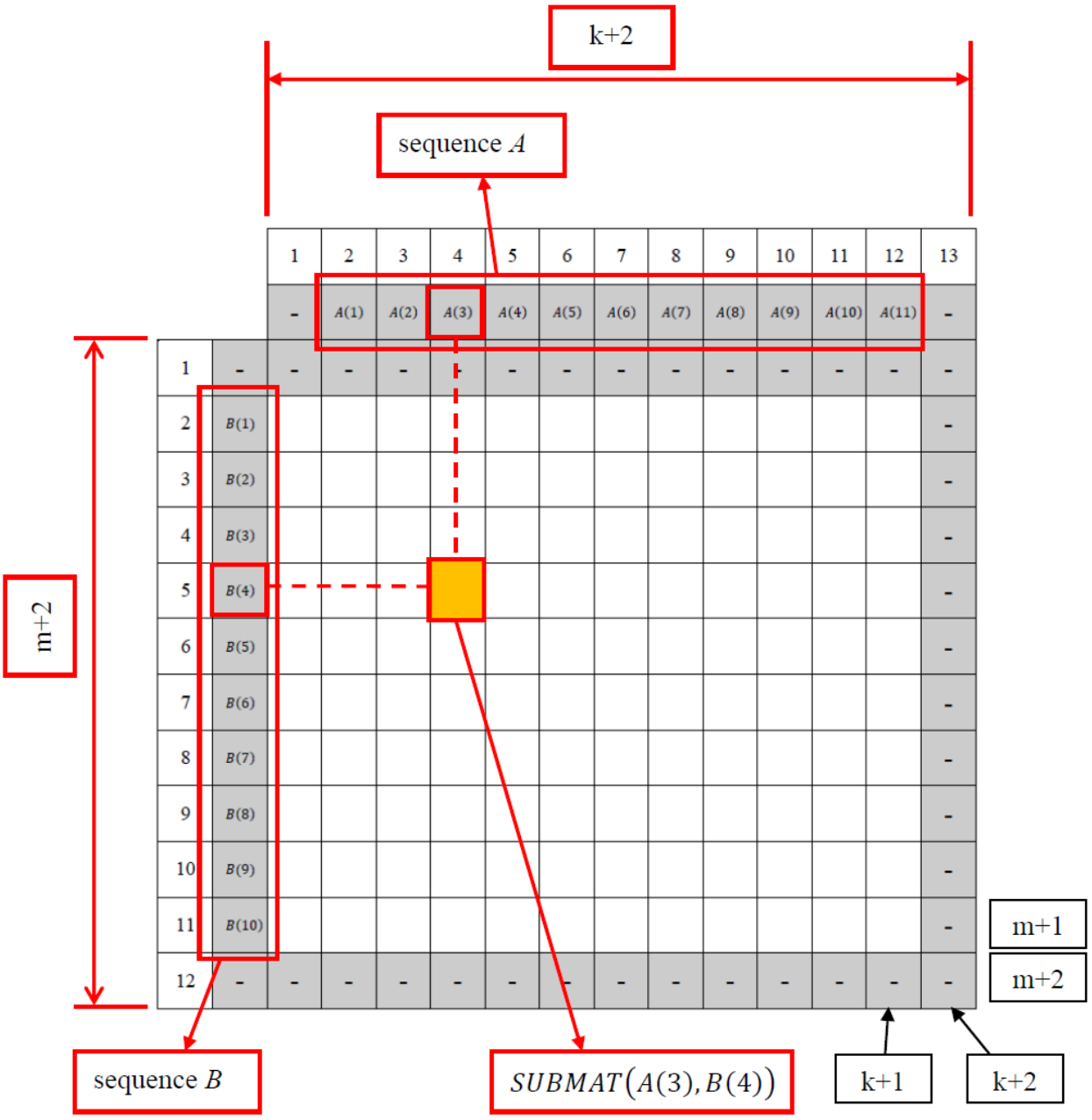
Seq A	-	C	A	T	_	T	
Seq B	G	A	A	T	C	T	
Score	-2	-1	1	1	-2	1	= -2



Seq A	C	A	-	T	-	T	
Seq B	G	A	A	T	C	T	
Score	-1	1	-2	1	-2	1	=-2

Consider the element (5,3) of TBM_2, for instance. It gives the value (4,3). This means that element (5,3) of TBM comes from elements (4,3), among other possible elements. Remember, the algorithm chooses only one possible element from TBM, among all the elements which give the best score. If we report the data from TBM_2 on TBM (as we did in paragraph 1.4.4), we obtain *Figure 10*. As you can see from the comparison with *Figure 4*, our program gives only a sub set of the alignments calculated manually. In particular, as the best score (-2) is only achieved by element (7,5), TBM_2 in *Figure 10*, leads us to define only two of the three best alignments, which are reported in *Figure 11*. As mentioned, TBM_2 gives only a subset of the best score alignments. Moreover the program choose only one of these best score alignments. In this case the alignment is the one on the left in *Figure 11*.

Figure 12. The generic MAT matrix for the complete code for Neddleman-Wunsch algorithm.

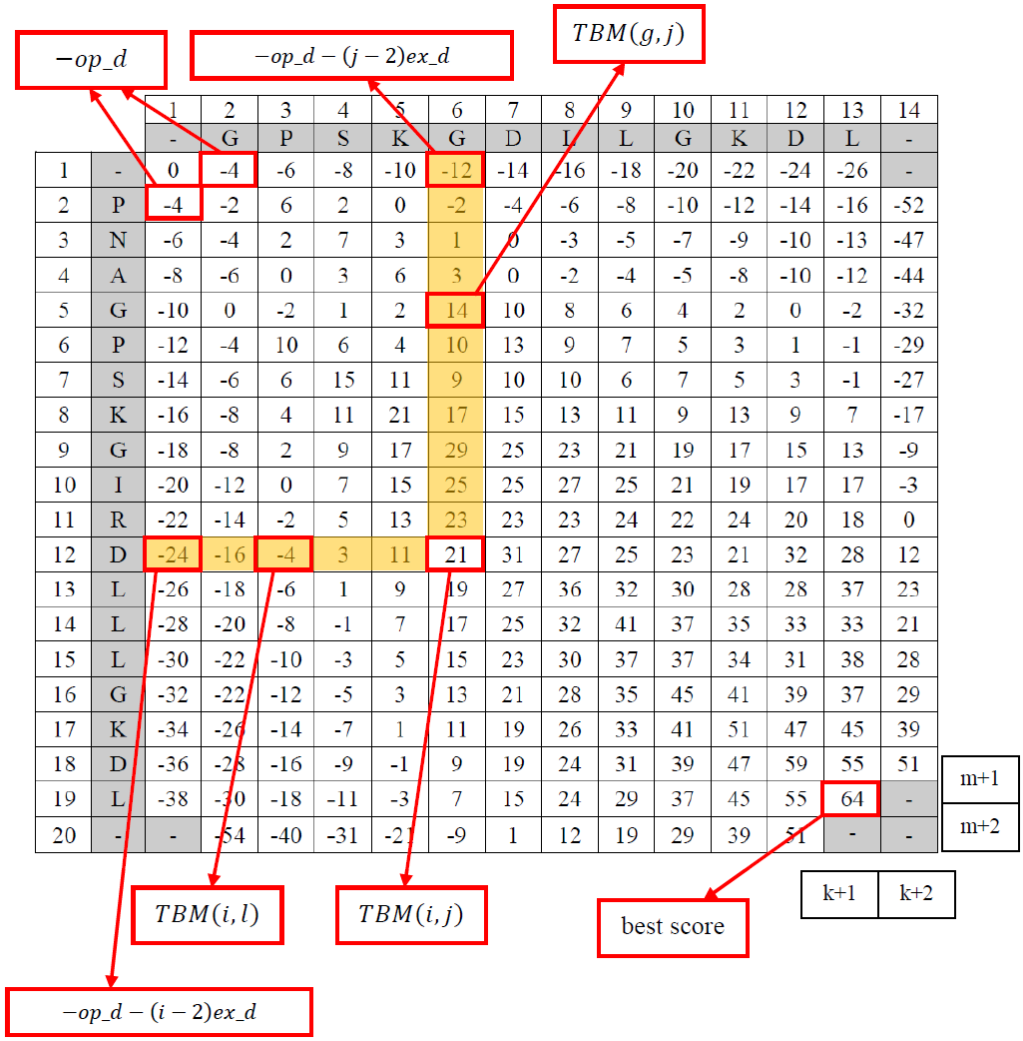


1.5 The complete software for global alignments

We will now present a software developed from the one introduced in paragraph 1.4.5, which presents the same functions and output of a professional global alignment software based on Needelman-

Wunsch algorithm. You can find two versions of this tool in the *Swiss Institute of Bioinformatics* (SIB) web site (↥) and in the *European Molecular Biology Laboratory* web site (↥). We will also provide in this section a series of application of our software, along with a comparison with the results given by one of those professional softwares on the same alignment.

Figure 13. An example of TBM matrix for the complete code for Neddleman-Wunsch algorithm.



		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	G	P	S	K	G	D	L	L	G	K	D	L	-
1	-	0	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-26	-
2	P	-4	-2	6	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-52
3	N	-6	-4	2	7	3	1	0	-3	-5	-7	-9	-10	-13	-47
4	A	-8	-6	0	3	6	3	0	-2	-4	-5	-8	-10	-12	-44
5	G	-10	0	-2	1	2	14	10	8	6	4	2	0	-2	-32
6	P	-12	-4	10	6	4	10	13	9	7	5	3	1	-1	-29
7	S	-14	-6	6	15	11	9	10	10	6	7	5	3	-1	-27
8	K	-16	-8	4	11	21	17	15	13	11	9	13	9	7	-17
9	G	-18	-8	2	9	17	29	25	23	21	19	17	15	13	-9
10	I	-20	-12	0	7	15	25	25	27	25	21	19	17	17	-3
11	R	-22	-14	-2	5	13	23	23	23	24	22	24	20	18	0
12	D	-24	-16	-4	3	11	21	31	27	25	23	21	32	28	12
13	L	-26	-18	-6	1	9	19	27	36	32	30	28	28	37	23
14	L	-28	-20	-8	-1	7	17	25	32	41	37	35	33	33	21
15	L	-30	-22	-10	-3	5	15	23	30	37	37	34	31	38	28
16	G	-32	-22	-12	-5	3	13	21	28	35	45	41	39	37	29
17	K	-34	-26	-14	-7	1	11	19	26	33	41	51	47	45	39
18	D	-36	-28	-16	-9	-1	9	19	24	31	39	47	59	55	51
19	L	-38	-30	-18	-11	-3	7	15	24	29	37	45	55	64	-
20	-	-	-54	-40	-31	-21	-9	1	12	19	29	39	51	-	-

1.6 A more sophisticated gap model

We introduced previously a simple model for gap penalty, where penalty for n consecutive gaps is simply given by $-d \times n$ (see Eq. 16). This model for penalties is appealing for its simplicity, but is often not realistic from a biological point of view, where it is harder for a gap (deletion or insertion) to open, than it is for it to extend. This leads us to consider a heavier penalty for the first gap, and a smaller one for gaps that follows (Ewens, et al., 2005). Thus, if we indicate with *op_d* the penalty for the opening gap, and with *ex_d* the one to use for the following gaps, we have to correct Eq. 18 as follows:

Eq. 18

$$\begin{cases} TBM(i+1,j+1) = max\{TBM(i,j) + MAT(i+1,j+1);\ \alpha(i+1,j+1); \beta(i+1,j+1)\} \\ \alpha(i+1,j+1) = max\{TBM(i+1;l) - op\_d - ex\_d(j-l-1): l = 0,1,\dots,j\} \\ \beta(i+1,j+1) = max\{TBM(g;j+1) - op\_d - ex\_d(i-g-1): g = 0,1,\dots,i\} \end{cases}$$

To best understand the notation used in Eq. 18, please consider TBM matrix in *Figure 13*. Further details for the implementation of this gap penalty model and other features of the complete code for the Needleman-Wunsch algorithm are in the following paragraph.

1.7 Code in Octave for the global alignment with Needelman-Wunsch algorithm

As a reference for the notation used, consider the generic MAT matrix in in *Figure 12* and the example of TBM in *Figure 13*. This code calls other files which store the main substitution matrices.

```
% file name = NeW_5.m
% date of creation = 29/03/2016
% We delete all previous values for variables
clear all
% We ask the user to choose a substitution matrix
SUBMAT = 0;
disp(" ")
disp("Please, select the substitution matrix to use for score calculation.")
disp(" ")
s = menu ("Substitution matrix", "BLOSUM45", "BLOSUM50", "BLOSUM62", "BLOSUM80", "PAM250");
% We load the chosen substitution matrix matrix
switch s
 case (1)
 load BLOSUM45.mat;
 SUBMAT=BLOSUM45;
 disp(" ")
 disp("You have chosen the substitution matrix BLOSUM45:")
 disp(" ")
 disp(BLOSUM45)
 case (2)
 load BLOSUM50.mat;
 SUBMAT=BLOSUM50;
 disp(" ")
 disp("You have chosen the substitution matrix BLOSUM50:")
 disp(" ")
 disp(BLOSUM50)
 case (3)
 load BLOSUM62.mat;
 SUBMAT=BLOSUM62;
 disp(" ")
 disp("You have chosen the substitution matrix BLOSUM62:")
 disp(" ")
 disp(BLOSUM62)
 case (4)
 load BLOSUM80.mat;
 SUBMAT=BLOSUM80;
 disp(" ")
 disp("You have chosen the substitution matrix BLOSUM80:")
 disp(" ")
 disp(BLOSUM80)
 otherwise
 load PAM250.mat;
 SUBMAT=PAM250;
 disp(" ")
 disp("You have chosen the substitution matrix PAM250:")
 disp(" ")
 disp(PAM250)
endswitch
% We define an array for the 20 amino acids
```

```
aa(1:20)= ['A' 'R' 'N' 'D' 'C' 'Q' 'E' 'G' 'H' 'I' 'L' 'K' 'M' 'F' 'P' 'S' 'T' 'W' 'Y' 'V'];
disp("Amino acids are intended in the following sequence:")
disp(" ")
disp(aa(1:20))
disp(" ")
% We ask the user for amino acids of sequence A
disp(" ")
disp("I will ask you for amino acids of sequence A.")
disp("Use capitol letters!")
disp(" ")
seq_A = input("Insert sequence A= ","s");
disp(" ")
A = ["-" seq_A];
% We ask the user for amino acids of sequence B
disp(" ")
disp("I will now ask you for amino acids of sequence B.")
disp("Again, use capitol letters!")
disp(" ")
seq_B = input("Insert sequence B= ","s");
disp(" ")
B = ["-" seq_B];
% We define length of sequence A and of sequence B, respectively
k=length(A)-1;
m=length(B)-1;
% We ask the user for the opening gap penalty
disp(" ")
disp("I will ask you for opening gap penalty.")
disp("Use a positive integer (suggested:12)!")
disp(" ")
op_d = input("Insert open gap penalty= ");
disp(" ")
% We ask the user for the extending gap penalty
disp(" ")
disp("I will ask you for extended gap penalty.")
disp("Use a positive integer (suggested:2)!")
disp(" ")
ex_d = input("Insert extended gap penalty= ");
disp(" ")
% We define elements of MAT, TBM and TBM_2 which are zero
MAT(1,1)=0;
MAT(1,2:k+1)=0;
MAT(2:m+1,1)=0;
TBM(1,1)=0;
TBM_2 = 0;
% We define elements of TBM and TBM_2 which have no role in calculation
% with a formal value of 100
TBM(1,k+2)=100;
TBM(m+2,1)=100;
TBM(m+1,k+2)=100;
TBM(m+2,k+2)=100;
TBM(m+2,k+1)=100;
TBM_2(1,1:k+2,1:2)=0;
TBM_2(1:m+2,1,1:2)=0;
TBM_2(m+1,k+2,1:2)=100;
TBM_2(m+2,k+2,1:2)=100;
TBM_2(m+2,k+1,1:2)=100;
% We define other elements of the first row and the first column of TBM
for i=2:1:m+1
 TBM(i,1)=-op_d-ex_d*(i-2);
 for j=2:1:k+1
```

```
   TBM(1,j)=-op_d-ex_d*(j-2);
  endfor
endfor
% We define elements of MAT
for i=2:1:m+1
  for j=2:1:k+1
    for h=1:1:20
      if ( A(j)==aa(h) )
        index_A=h;
      endif
      if ( B(i)==aa(h) )
        index_B=h;
      endif
    endfor
    MAT(i,j)= SUBMAT(index_B, index_A);
  endfor
endfor
% We print array MAT on the screen
disp(" ")
disp('MAT matrix contains the score between each pair of amino acids:')
disp(MAT(2:m+1,2:k+1))
disp(" ")
% We define elements for TBM and TBM_2
for i=2:1:m+1
  for j=2:1:k+1
    for l=1:1:j-1
      AL(i,l)=TBM(i,l)-op_d-ex_d*(j-l-1);
    endfor
    for g=1:1:i-1
      BE(g,j)=TBM(g,j)-op_d-ex_d*(i-g-1);
    endfor
    if ( TBM(i-1,j-1)+MAT(i,j) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
      TBM(i,j)=TBM(i-1,j-1)+MAT(i,j);
      TBM_2(i,j,1)=i-1;
      TBM_2(i,j,2)=j-1;
    else
      for l=1:1:j-1
        if ( AL(i,l) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
          TBM(i,j)=AL(i,l);
          TBM_2(i,j,1)=i;
          TBM_2(i,j,2)=l;
        else
          for g=1:1:i-1
            if ( BE(g,j) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
              TBM(i,j)=BE(g,j);
              TBM_2(i,j,1)=g;
              TBM_2(i,j,2)=j;
            endif
          endfor
        endif
      endfor
    endif
  endfor
endfor
% We define elements for the last column of TBM and TBM_2
for g=2:1:m
  TBM(g,k+2)=TBM(g,k+1) -op_d-ex_d*(m+1-g-1);
  TBM_2(g,k+2,1)=g;
  TBM_2(g,k+2,2)=k+1;
endfor
```

```
% We define elements for the last row of TBM and TBM_2
for l=2:1:k
 TBM(m+2,l)=TBM(m+1,l) -op_d-ex_d*(k+1-l-1);
 TBM_2(m+2,l,1)=m+1;
 TBM_2(m+2,l,2)=l;
endfor
% We print arrays TBM and TBM_2
disp(" ")
disp('TBM matrix contains the score between each partial alignments:')
disp(TBM)
disp('TBM_2 matrix contains the record of where each partial alignment comes from:')
disp(TBM_2)
disp(" ")
% We put into array ALIGN the sequence of elements of A and B
% which give the best alignment
% Here we start with the first row of ALIGN
%
%
control=0;
if ( TBM(m+1,k+1) >= max([max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
  ALIGN_a(1:2,1:1001)=0;
  ALIGN_a(1,1)= k+1;
  ALIGN_a(2,1)= m+1;
  best_score= TBM(m+1,k+1);
  control=1;
endif
if (control==1)
 % Now we define the following rows of ALIGN_a
 for n=1:1:1000
  if ( ALIGN_a(1,n)>0 )
   if ( ALIGN_a(2,n)>0 )
    ALIGN_a(1,n+1)=TBM_2(ALIGN_a(2,n),ALIGN_a(1,n),2);
    ALIGN_a(2,n+1)=TBM_2(ALIGN_a(2,n),ALIGN_a(1,n),1);
   endif
  endif
 endfor
 % We search for the numebr of elements of ALIGN_a
 n=1;
 while (ALIGN_a(1,n)>0)
  n=n+1;
 endwhile
 n=n-1;
 % We reverse the order of columns of ALIGN_a
 ALIGN_a(1:2,1:n)=fliplr(ALIGN_a(1:2,1:n));
 % We re-write ALIGN_a in ALIGNex_a
 ALIGNex_a(:,1)=ALIGN_a(:,1);
 i=1;
 j=1;
 while (ALIGN_a(1,j)>0)
  if ( ALIGN_a(1,j+1)==ALIGN_a(1,j) )
   delta=(ALIGN_a(2,j+1)-ALIGN_a(2,j));
   for s=1:1:delta
    ALIGNex_a(1,i+s)=ALIGN_a(1,j);
    ALIGNex_a(2,i+s)=ALIGN_a(2,j)+s;
   endfor
   i=i+delta;
   j=j+1;
  elseif ( ALIGN_a(2,j+1)==ALIGN_a(2,j) )
   delta=(ALIGN_a(1,j+1)-ALIGN_a(1,j));
   for s=1:1:delta
```

```
    ALIGNex_a(1,i+s)=ALIGN_a(1,j)+s;
    ALIGNex_a(2,i+s)=ALIGN_a(2,j);
   endfor
   i=i+delta;
   j=j+1;
  else
   ALIGNex_a(1,i+1)=ALIGN_a(1,j+1);
   ALIGNex_a(2,i+1)=ALIGN_a(2,j+1);
   i=i+1;
   j=j+1;
  endif
endwhile
%
% We search for the numebr of elements of ALIGNex_c
i=1;
while (ALIGNex_a(1,i)>0)
 i=i+1;
endwhile
i=i-1;
%
% We put ALIGNex_a in ALIGN_a
ALIGN_a(:,1:i)=ALIGNex_a(:,1:i);
% We calculate the array ALIGN_a_2 which contains the alignment
ALIGN_a_2(1,1)=A(ALIGN_a(1,1));
ALIGN_a_2(2,1)=B(ALIGN_a(2,1));
j=2;
while (ALIGN_a(1,j)>0)
 if ( ALIGN_a(1,j)==ALIGN_a(1,j-1) )
  ALIGN_a_2(1,j)='-';
  ALIGN_a_2(2,j)=B(ALIGN_a(2,j));
 elseif ( ALIGN_a(2,j)==ALIGN_a(2,j-1) )
  ALIGN_a_2(1,j)=A(ALIGN_a(1,j));
  ALIGN_a_2(2,j)='-';
 else
  ALIGN_a_2(1,j)=A(ALIGN_a(1,j));
  ALIGN_a_2(2,j)=B(ALIGN_a(2,j));
 endif
 j=j+1;
endwhile
%
if (ALIGN_a(1,1)!=1)
 for h=1:1:ALIGN_a(1,1)
  pre_ALIGN_a_2(1,h)=A(h);
  pre_ALIGN_a_2(2,h)="-";
 endfor
 ALIGN_a_2=[pre_ALIGN_a_2(:,1:ALIGN_a(1,1)) ALIGN_a_2(:,2:j)];
endif
if (ALIGN_a(2,1)!=1)
 for h=1:1:ALIGN_a(2,1)
  pre_ALIGN_a_2(1,h)="-";
  pre_ALIGN_a_2(2,h)=B(h);
 endfor
 ALIGN_a_2=[pre_ALIGN_a_2 ALIGN_a_2(:,2:j-1)];
endif
%
disp(" ")
disp("One of the possible alignments with the best score is:")
disp(" ")
disp(["seq. A: " ALIGN_a_2(1,:)])
disp(["seq. B: " ALIGN_a_2(2,:)])
```

```
 disp(" ")
 disp("The path through TBM, from the end to the beginning, is:")
 disp(" ")
 disp(ALIGN_a(1:2,1:j))
 disp(" ")
 % We print on the screen the best score
 disp(" ")
 disp('The best score for alignments is:')
 disp(" ")
 disp(best_score)
 disp(" ")
endif
% We searh for another best alignment
%
%
control=0;
for g=2:1:m
 if ( TBM(g,k+2) >= max([TBM(m+1,k+1) max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
  ALIGN_b(1:2,1:1001)=0;
  ALIGN_b(1,1)= TBM_2(g,k+2,2);
  ALIGN_b(2,1)= TBM_2(g,k+2,1);
  best_score= TBM(g,k+2);
  control=1;
 endif
endfor
if (control==1)
 % Now we define the following rows of ALIGN
 for n=1:1:1000
  if ( ALIGN_b(1,n)>0 )
   if ( ALIGN_b(2,n)>0 )
    ALIGN_b(1,n+1)=TBM_2(ALIGN_b(2,n),ALIGN_b(1,n),2);
    ALIGN_b(2,n+1)=TBM_2(ALIGN_b(2,n),ALIGN_b(1,n),1);
   endif
  endif
 endfor
 % We search for the numebr of elements of ALIGN
 n=1;
 while (ALIGN_b(1,n)>0)
  n=n+1;
 endwhile
 n=n-1;
 % We reverse the ordero of columns of ALIGN_b
 ALIGN_b(1:2,1:n)=fliplr(ALIGN_b(1:2,1:n));
 % We re-write ALIGN_b in ALIGNex_b
 ALIGNex_b(:,1)=ALIGN_b(:,1);
 i=1;
 j=1;
 while (ALIGN_b(1,j)>0)
  if ( ALIGN_b(1,j+1)==ALIGN_b(1,j) )
   delta=(ALIGN_b(2,j+1)-ALIGN_b(2,j));
   for s=1:1:delta
    ALIGNex_b(1,i+s)=ALIGN_b(1,j);
    ALIGNex_b(2,i+s)=ALIGN_b(2,j)+s;
   endfor
   i=i+delta;
   j=j+1;
  elseif ( ALIGN_b(2,j+1)==ALIGN_b(2,j) )
   delta=(ALIGN_b(1,j+1)-ALIGN_b(1,j));
   for s=1:1:delta
    ALIGNex_b(1,i+s)=ALIGN_b(1,j)+s;
```

```
    ALIGNex_b(2,i+s)=ALIGN_b(2,j);
   endfor
   i=i+delta;
   j=j+1;
  else
   ALIGNex_b(1,i+1)=ALIGN_b(1,j+1);
   ALIGNex_b(2,i+1)=ALIGN_b(2,j+1);
   i=i+1;
   j=j+1;
  endif
endwhile
%
% We search for the numebr of elements of ALIGNex_b
i=1;
while (ALIGNex_b(1,i)>0)
 i=i+1;
endwhile
i=i-1;
%
% We put ALIGNex_b in ALIGN_b
ALIGN_b(:,1:i)=ALIGNex_b(:,1:i);
% We calculate the array ALIGN_b_2 which contains the alignment
ALIGN_b_2(1,1)=A(ALIGN_b(1,1));
ALIGN_b_2(2,1)=B(ALIGN_b(2,1));
j=2;
while (ALIGN_b(1,j)>0)
 if ( ALIGN_b(1,j)==ALIGN_b(1,j-1) )
  ALIGN_b_2(1,j)='-';
  ALIGN_b_2(2,j)=B(ALIGN_b(2,j));
 elseif ( ALIGN_b(2,j)==ALIGN_b(2,j-1) )
  ALIGN_b_2(1,j)=A(ALIGN_b(1,j));
  ALIGN_b_2(2,j)='-';
 else
  ALIGN_b_2(1,j)=A(ALIGN_b(1,j));
  ALIGN_b_2(2,j)=B(ALIGN_b(2,j));
 endif
 j=j+1;
endwhile
%
if (ALIGN_b(1,1)!=1)
 for h=1:1:ALIGN_b(1,1)
  pre_ALIGN_b_2(1,h)=A(h);
  pre_ALIGN_b_2(2,h)="-";
 endfor
 ALIGN_b_2=[pre_ALIGN_b_2(:,1:ALIGN_b(1,1)) ALIGN_b_2(:,2:j)];
endif
if (ALIGN_b(2,1)!=1)
 for h=1:1:ALIGN_b(2,1)
  pre_ALIGN_b_2(1,h)="-";
  pre_ALIGN_b_2(2,h)=B(h);
 endfor
 ALIGN_b_2=[pre_ALIGN_b_2 ALIGN_b_2(:,2:j-1)];
endif
%
for p=1:1:m+1-ALIGN_b(2,i)
 ALIGN_b_2(1,i+p)="-";
 ALIGN_b_2(2,i+p)=B(ALIGN_b(2,i)+p);
endfor
disp(" ")
disp("One of the possible alignments with the best score is:")
```

```
 disp(" ")
 disp(["seq. A: " ALIGN_b_2(1,:)])
 disp(["seq. B: " ALIGN_b_2(2,:)])
 disp(" ")
 disp("The path through TBM, from the end to the beginning, is:")
 disp(" ")
 disp(ALIGN_b(1:2,1:i))
 disp(" ")
 % We print on the screen the best score
 disp(" ")
 disp('The best score for alignments is:')
 disp(" ")
 disp(best_score)
 disp(" ")
endif
% We searh for another best alignment
%
%
control=0;
for l=2:1:k
 if ( TBM(m+2,l) >= max([TBM(m+1,k+1) max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
  ALIGN_c(1:2,1:1001)=0;
  ALIGN_c(1,1)= TBM_2(m+2,l,2);
  ALIGN_c(2,1)= TBM_2(m+2,l,1);
  best_score= TBM(m+2,l);
  control=1;
 endif
endfor
if (control==1)
 % Now we define the following rows of ALIGN
 for n=1:1:1000
  if ( ALIGN_c(1,n)>0 )
   if ( ALIGN_c(2,n)>0 )
    ALIGN_c(1,n+1)=TBM_2(ALIGN_c(2,n),ALIGN_c(1,n),2);
    ALIGN_c(2,n+1)=TBM_2(ALIGN_c(2,n),ALIGN_c(1,n),1);
   endif
  endif
 endfor
 % We search for the numebr of elements of ALIGN
 n=1;
 while (ALIGN_c(1,n)>0)
  n=n+1;
 endwhile
 n=n-1;
 % We reverse the ordero of columns of ALIGN_c
 ALIGN_c(1:2,1:n)=fliplr(ALIGN_c(1:2,1:n));
 % We re-write ALIGN_b in ALIGNex_c
 ALIGNex_c(:,1)=ALIGN_c(:,1);
 i=1;
 j=1;
 while (ALIGN_c(1,j)>0)
  if ( ALIGN_c(1,j+1)==ALIGN_c(1,j) )
   delta=(ALIGN_c(2,j+1)-ALIGN_c(2,j));
   for s=1:1:delta
    ALIGNex_c(1,i+s)=ALIGN_c(1,j);
    ALIGNex_c(2,i+s)=ALIGN_c(2,j)+s;
   endfor
   i=i+delta;
   j=j+1;
  elseif ( ALIGN_c(2,j+1)==ALIGN_c(2,j) )
```

```
   delta=(ALIGN_c(1,j+1)-ALIGN_c(1,j));
   for s=1:1:delta
    ALIGNex_c(1,i+s)=ALIGN_c(1,j)+s;
    ALIGNex_c(2,i+s)=ALIGN_c(2,j);
   endfor
   i=i+delta;
   j=j+1;
  else
   ALIGNex_c(1,i+1)=ALIGN_c(1,j+1);
   ALIGNex_c(2,i+1)=ALIGN_c(2,j+1);
   i=i+1;
   j=j+1;
  endif
endwhile
%
% We search for the numebr of elements of ALIGNex_c
i=1;
while (ALIGNex_c(1,i)>0)
 i=i+1;
endwhile
i=i-1;
%
% We put ALIGNex_c in ALIGN_c
ALIGN_c(:,1:i)=ALIGNex_c(:,1:i);
% We calculate the array ALIGN_c_2 which contains the alignment
ALIGN_c_2(1,1)=A(ALIGN_c(1,1));
ALIGN_c_2(2,1)=B(ALIGN_c(2,1));
j=2;
while (ALIGN_c(1,j)>0)
 if ( ALIGN_c(1,j)==ALIGN_c(1,j-1) )
  ALIGN_c_2(1,j)='-';
  ALIGN_c_2(2,j)=B(ALIGN_c(2,j));
 elseif ( ALIGN_c(2,j)==ALIGN_c(2,j-1) )
  ALIGN_c_2(1,j)=A(ALIGN_c(1,j));
  ALIGN_c_2(2,j)='-';
 else
  ALIGN_c_2(1,j)=A(ALIGN_c(1,j));
  ALIGN_c_2(2,j)=B(ALIGN_c(2,j));
 endif
 j=j+1;
endwhile
%
if (ALIGN_c(1,1)!=1)
 for h=1:1:ALIGN_c(1,1)
  pre_ALIGN_c_2(1,h)=A(h)
  pre_ALIGN_c_2(2,h)="-"
 endfor
 ALIGN_c_2=[pre_ALIGN_c_2(:,1:ALIGN_c(1,1)) ALIGN_c_2(:,2:j)]
endif
if (ALIGN_c(2,1)!=1)
 for h=1:1:ALIGN_c(2,1)
  pre_ALIGN_c_2(1,h)="-"
  pre_ALIGN_c_2(2,h)=B(h)
 endfor
 ALIGN_c_2=[pre_ALIGN_c_2 ALIGN_c_2(:,2:j-1)]
endif
%
for p=1:1:k+1-ALIGN_c(1,i)
 ALIGN_c_2(1,i+p)=A(ALIGN_c(1,i)+p);
 ALIGN_c_2(2,i+p)="-";
```

```
 endfor
 disp(" ")
 disp("One of the possible alignments with the best score is:")
 disp(" ")
 disp(["seq. A: " ALIGN_c_2(1,:)])
 disp(["seq. B: " ALIGN_c_2(2,:)])
 disp(" ")
 disp("The path through TBM, from the end to the beginning, is:")
 disp(" ")
 disp(ALIGN_c(1:2,1:i))
 disp(" ")
 % We print on the screen the best score
 disp(" ")
 disp('The best score for alignments is:')
 disp(" ")
 disp(best_score)
 disp(" ")
endif
```

1.7.1 A first application and comparison with professional software[1]

Consider the following two sequences:

Eq. 19
$$\begin{cases} seq.A = NGPIRDLLLGKD \\ seq.B = STIAPALISS \end{cases}$$

As a first application we will calculate the best score and a best score alignment between them using BLOSUM62 substitution matrix ↕. We will use the following settings:

Eq. 20
$$\begin{cases} open\ gap\ penalty = 2 \\ extended\ gap\ penalty = 2 \\ substitution\ matrix = BLOSUM62 \end{cases}$$

Table 14. MAT matrix, based on BLOSUM62 substitution matrix.

		1	2	3	4	5	6	7	8	9	10	11	12	13
		-	N	G	P	I	R	D	L	L	L	G	K	D
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	S	-	1	0	-1	-2	-1	0	-2	-2	-2	0	0	0
3	T	-	0	-2	-1	-1	-1	-1	-1	-1	-1	-2	-1	-1
4	I	-	-3	-4	-3	4	-3	-3	2	2	2	-4	-3	-3
5	A	-	-2	0	-1	-1	-1	-2	-1	-1	-1	0	-1	-2
6	P	-	-2	-2	7	-3	-2	-1	-3	-3	-3	-2	-1	-1
7	A	-	-2	0	-1	-1	-1	-2	-1	-1	-1	0	-1	-2
8	L	-	-3	-4	-3	2	-2	-4	4	4	4	-4	-2	-4
9	I	-	-3	-4	-3	4	-3	-3	2	2	2	-4	-3	-3
10	S	-	1	0	-1	-2	-1	0	-2	-2	-2	0	0	0
11	S	-	1	0	-1	-2	-1	0	-2	-2	-2	0	0	0

[1] I realized later that in this and in following examples (from 1 to 7), substitution matrices had not been scaled so to have logarithms to base 2 of odds ratio. I later corrected the code with the right scale for each substitution matrix and I repeated those tests.

The output for MAT is reported in *Table 14* while you can find TBM in *Table 15* and TBM_2 in *Table 16*. From TBM we identify that the best score is 3, and we have this for alignments which ends in 12,12 and in 11,13. Using data from TBM_2 we can identify the two best score alignments reported in *Figure 14* ant in *Table 17*.

Table 15. TBM matrix.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	N	G	P	I	R	D	L	L	L	G	K	D	-
1	-	-	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-
2	S	-2	1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-39
3	T	-4	-1	-1	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-36
4	I	-6	-3	-3	-4	2	0	-2	-4	-6	-8	-10	-12	-14	-28
5	A	-8	-5	-3	-4	0	1	-1	-3	-5	-7	-8	-10	-12	-24
6	P	-10	-7	-5	4	2	0	0	-2	-4	-6	-8	-9	-11	-21
7	A	-12	-9	-7	2	3	1	-1	-1	-3	-5	-6	-8	-10	-18
8	L	-14	-11	-9	0	4	2	0	3	3	1	-1	-3	-5	-11
9	I	-16	-13	-11	-2	4	2	0	2	5	5	3	1	-1	-5
10	S	-18	-15	-13	-4	2	3	2	0	3	3	5	3	1	-1
11	S	-20	-17	-15	-6	0	1	3	1	1	1	3	5	3	-
12	-	-	-39	-35	-24	-16	-13	-9	-9	-7	-5	-1	3	-	-

Table 16. TBM_2 matrix.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	N	G	P	I	R	D	L	L	L	G	K	D	-
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	S	-	1,1	2,1	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10	2,11	2,12	2,13
3	T	-	2,2	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	3,10	2,11	2,12	3,13
4	I	-	3,2	3,3	3,3	3,4	4,5	4,6	4,7	4,8	4,9	4,10	4,11	4,12	4,13
5	A	-	4,2	4,2	4,3	4,5	4,5	5,6	4,7	4,8	4,9	4,10	5,11	5,12	5,13
6	P	-	5,2	5,3	5,3	6,4	6,5	5,6	6,7	6,8	6,9	6,10	5,11	5,12	6,13
7	A	-	6,2	6,2	6,4	6,4	6,5	7,6	6,7	6,8	6,9	6,10	7,11	7,12	7,13
8	L	-	7,2	7,3	7,4	7,4	8,5	8,6	7,7	7,8	7,9	8,10	8,11	8,12	8,13
9	I	-	8,2	8,3	8,4	8,4	9,5	9,6	8,7	8,8	8,9	9,10	9,11	9,12	9,13
10	S	-	9,1	9,2	9,4	9,5	9,5	9,6	10,7	9,9	9,9	9,10	9,11	9,12	10,13
11	S	-	10,1	10,2	10,4	10,5	10,5	10,6	11,7	10,9	10,9	10,10	10,11	10,12	-
12	-	-	11,2	11,3	11,4	11,5	11,6	11,7	11,8	11,9	11,10	11,11	11,12	-	-

Our software gives as output both the alignments in in *Figure 14* ant in *Table 17* for which it calculates correctly a score of 3, which is indeed the best score for sequences in **Errore. L'origine riferimento non è stata trovata.**, with settings in Eq. 20.

The software by *Swiss Institute of Bioinformatics* (SIB) (↥) gives as best alignments only the second reported in *Table 17*. Thus, in this case our software gives a correct output.

Figure 14. The two best alignments deduced from TBM_2.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	N	G	P	I	R	D	L	L	L	G	K	D	-
1	-	-	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-
2	S	-2	1	1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-39
3	T	-4	-1	-1	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-36
4	I	-6	-3	-3	-4	2	0	-2	-4	-6	-8	-10	-12	-14	-28
5	A	-8	-5	-3	-4	0	1	-1	-3	-5	-7	-8	-10	-12	-24
6	P	-10	-7	-5	4	2	0	0	-2	-4	-6	-8	-9	-11	-21
7	A	-12	-9	-7	2	3	1	-1	-1	-3	-5	-6	-8	-10	-18
8	L	-14	-11	-9	0	4	2	0	3	3	1	-1	-3	-5	-11
9	I	-16	-13	-11	-2	4	2	0	2	5	5	3	1	-1	-5
10	S	-18	-15	-13	-4	2	3	2	0	3	3	5	3	1	-1
11	S	-20	-17	-15	-6	0	1	3	1	1	1	3	5	3	-
12	-	-	-39	-35	-24	-16	-13	-9	-9	-7	-5	-1	3	-	-

Table 17. The two best alignments deduced from TBM_2. Opening gap is 2, extending gap is 2 and we used BLOSUM62 substitution matrix.

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	-	S	S	
Score	1	-2	-1	4	-1	-1	-1	4	2	-2	0	0	=3

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	S	S	-	
Score	1	-2	-1	4	-1	-1	-1	4	2	0	0	-2	=3

Table 18. TBM matrix.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	N	G	P	I	R	D	L	L	L	G	K	D	-
1	-	-	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-
2	S	-4	1	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-25
3	T	-5	-3	-1	-4	-5	-6	-7	-8	-9	-10	-12	-12	-13	-24
4	I	-6	-4	-5	-4	0	-4	-5	-5	-6	-7	-9	-10	-11	-21
5	A	-7	-5	-5	-6	-4	-1	-5	-6	-6	-7	-7	-10	-11	-20
6	P	-8	-6	-7	3	-1	-2	-2	-4	-5	-6	-7	-8	-9	-17
7	A	-9	-7	-6	-1	2	-2	-3	-3	-5	-6	-6	-8	-9	-16
8	L	-10	-8	-9	-2	1	0	-4	1	1	-1	-4	-5	-6	-12
9	I	-11	-9	-10	-3	2	-2	-3	-2	3	3	-1	-2	-3	-8
10	S	-12	-10	-9	-4	-2	1	-4	-1	1	3	-1	3	-2	-6
11	S	-13	-11	-10	-5	-3	-3	1	-3	-2	-2	1	3	-1	-
12	-	-	-25	-23	-17	-14	-13	-8	-11	-9	-8	-4	-1	-	-

SIB software gives as score the value -1, which is 3-4, where -4 is the extending gap penalty that SIB's software adds always to the first gap penalty.

1.7.2 Second application

We will use the same sequences in Eq. 19, with following settings:

Eq. 21
$$\begin{cases} open\ gap\ penalty = 4 \\ extended\ gap\ penalty = 1 \\ substitution\ matrix = BLOSUM62 \end{cases}$$

The output for MAT is the same as in *Table 14* while you can find TBM in *Table 18* and TBM_2 in *Table 19*. The two best alignments given by the program are in *Figure 15*. They are the same as in *Table 17*, but in this case with a score of -1.

Table 19. TBM_2 matrix.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	N	G	P	I	R	D	L	L	L	G	K	D	-
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	S	-	1,1	2,2	2,2	2,2	2,2	2,2	2,2	2,2	2,2	2,2	2,2	2,2	2,13
3	T	-	2,2	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	3,10	2,11	2,12	3,13
4	I	-	2,2	3,3	3,3	3,4	4,5	4,5	3,7	3,8	3,9	4,5	4,5	4,5	4,13
5	A	-	2,2	4,2	4,3	4,5	4,5	5,6	4,7	4,8	4,9	4,10	4,11	5,6	5,13
6	P	-	2,2	5,2	5,3	6,4	6,4	5,6	6,4	6,4	6,4	6,4	5,11	6,4	6,13
7	A	-	2,2	6,2	6,4	6,4	6,5	7,5	6,7	6,8	6,9	6,10	6,11	7,5	7,13
8	L	-	2,2	3,3	6,4	7,4	7,5	8,6	7,7	7,8	7,9	8,9	8,9	8,9	8,13
9	I	-	2,2	3,3	6,4	8,4	8,5	8,6	8,7	8,8	8,9	9,10	9,10	9,10	9,13
10	S	-	9,1	9,2	6,4	9,5	9,5	9,6	8,8	9,9	9,9	9,10	9,11	9,12	10,13
11	S	-	10,1	10,2	6,4	9,5	10,5	10,6	11,7	9,9	9,10	10,10	10,11	10,12	-
12	-	-	11,2	11,3	11,4	11,5	11,6	11,7	11,8	11,9	11,10	11,11	11,12	-	-

Figure 15. The two best alignments deduced from TBM_2.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	N	G	P	I	R	D	L	L	L	G	K	D	-
1	-	-	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-
2	S	-4	1	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-25
3	T	-5	-3	-1	-4	-5	-6	-7	-8	-9	-10	-12	-12	-13	-24
4	I	-6	-4	-5	-4	0	-4	-5	-5	-6	-7	-9	-10	-11	-21
5	A	-7	-5	-5	-6	-4	-1	-5	-6	-6	-7	-7	-10	-11	-20
6	P	-8	-6	-7	3	-1	-2	-2	-4	-5	-6	-7	-8	-9	-17
7	A	-9	-7	-6	-1	2	-2	-3	-3	-5	-6	-6	-8	-9	-16
8	L	-10	-8	-9	-2	1	0	-4	1	1	-1	-4	-5	-6	-12
9	I	-11	-9	-10	-3	2	-2	-3	-2	3	3	-1	-2	-3	-8
10	S	-12	-10	-9	-4	-2	1	-4	-1	1	3	-1	3	-2	-6
11	S	-13	-11	-10	-5	-3	-3	1	-3	-2	-2	1	3	-1	-
12	-	-	-25	-23	-17	-14	-13	-8	-11	-9	-8	-4	-1	-	-

Table 20. The two best alignments deduced from TBM_2. Opening gap is 4, extending gap is 1 and we used BLOSUM62 substitution matrix.

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	-	S	S	
Score	1	-4	-1	4	-1	-1	-1	4	2	-4	0	0	=-1

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	S	S	-	
Score	1	-4	-1	4	-1	-1	-1	4	2	0	0	-4	=-1

Software by SIB (↑) gives as best alignments only the second reported in *Table 20*, with a score of -3, which is -1-1-1. Thus, the comparison between the two software gives a positive result.

Table 21. The two best alignments deduced from TBM_2. Opening gap is 12, extending gap is 2 and we used BLOSUM62 substitution matrix.

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	-	S	S	
Score	1	-12	-1	4	-1	-1	-1	4	2	-12	0	0	=-17

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	S	S	-	
Score	1	-12	-1	4	-1	-1	-1	4	2	0	0	-12	=-17

Table 22. The best alignments given by SIB software.

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	T	I	A	P	A	L	I	-	-	S	S	
Score	1	-2	-3	-1	-2	-2	4	2	-12	-2	0	0	=-17

Table 23. MAT matrix, based on BLOSUM62 substitution matrix.

		1	2	3	4	5	6	7	8	9	10	11	12
		-	P	S	T	I	A	P	A	L	I	S	S
1	-	-	-	-	-	-	-	-	-	-	-	-	-
2	P	-	7	-1	-1	-3	-1	7	-1	-3	-3	-1	-1
3	N	-	-2	1	0	-3	-2	-2	-2	-3	-3	1	1
4	G	-	-2	0	-2	-4	0	-2	0	-4	-4	0	0
5	P	-	7	-1	-1	-3	-1	7	-1	-3	-3	-1	-1
6	I	-	-3	-2	-1	4	-1	-3	-1	2	4	-2	-2
7	R	-	-2	-1	-1	-3	-1	-2	-1	-2	-3	-1	-1
8	D	-	-1	0	-1	-3	-2	-1	-2	-4	-3	0	0
9	L	-	-3	-2	-1	2	-1	-3	-1	4	2	-2	-2
10	L	-	-3	-2	-1	2	-1	-3	-1	4	2	-2	-2
11	L	-	-3	-2	-1	2	-1	-3	-1	4	2	-2	-2
12	G	-	-2	0	-2	-4	0	-2	0	-4	-4	0	0
13	K	-	-1	0	-1	-3	-1	-1	-1	-2	-3	0	0
14	D	-	-1	0	-1	-3	-2	-1	-2	-4	-3	0	0
15	L	-	-3	-2	-1	2	-1	-3	-1	4	2	-2	-2

1.7.3 Third application

We will use the same sequences in Eq. 19, with following settings:

Eq. 22
$$\begin{cases} open\ gap\ penalty = 12 \\ extended\ gap\ penalty = 2 \\ substitution\ matrix = BLOSUM62 \end{cases}$$

The output of the program are the two alignments in *Table 21* while the best alignment given by SIB software (↑) is in *Table 22*. As you can see, although the two softwares give different best alignments, nevertheless the best score is the same. Thus, this comparison give a positive result too, as our software select only one best alignment for each element of TBM_2 and gives no more than three alignments as output, while SIB software seems to give always only a best alignment.

Table 24. TBM matrix.

		1	2	3	4	5	6	7	8	9	10	11	12	13
		-	P	S	T	I	A	P	A	L	I	S	S	-
1	-	0	-10	-12	-14	-16	-18	-20	-22	-24	-26	-28	-30	-
2	P	-10	7	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-55
3	N	-12	-3	8	-2	-4	-6	-8	-10	-12	-14	-16	-18	-50
4	G	-14	-5	-2	6	-4	-4	-8	-8	-12	-14	-14	-16	-46
5	P	-16	-7	-4	-3	3	-5	3	-7	-9	-11	-13	-15	-43
6	I	-18	-9	-6	-5	1	2	-7	2	-5	-5	-12	-14	-40
7	R	-20	-11	-8	-7	-8	0	0	-8	0	-8	-6	-13	-37
8	D	-22	-13	-10	-9	-10	-10	-1	-2	-10	-3	-8	-6	-28
9	L	-24	-15	-12	-11	-7	-11	-11	-2	2	-8	-5	-10	-30
10	L	-26	-17	-14	-13	-9	-8	-13	-12	2	4	-6	-7	-25
11	L	-28	-19	-16	-15	-11	-10	-11	-14	-8	4	2	-8	-24
12	G	-30	-21	-18	-18	-19	-11	-12	-11	-10	-6	4	2	-12
13	K	-32	-23	-20	-19	-21	-20	-12	-13	-12	-8	-6	4	-8
14	D	-34	-25	-22	-21	-22	-22	-21	-14	-14	-10	-8	-6	-16
15	L	-36	-27	-24	-23	-19	-23	-23	-22	-10	-12	-10	-8	-
16	-	-	-55	-50	-47	-41	-43	-41	-38	-24	-24	-20	-	-

1.7.4 Fourth application

We will use the following sequences

Eq. 23
$$\begin{cases} seq.A = PSTIAPALISS \\ seq.B = PNGPIRDLLLGKDL \end{cases}$$

with following settings:

Eq. 24
$$\begin{cases} open\ gap\ penalty = 10 \\ extended\ gap\ penalty = 2 \\ substitution\ matrix = BLOSUM62 \end{cases}$$

In *Figure 16* you find the two best score alignments. You have two paths for the same alignment, and this is due to how I built the TBM matrix, particularly the last column and the last row. The program

give correctly the same alignment for two times. The alignment is in *Table 25*. MAT is in *Table 23* while TBM is in *Table 24*.

Figure 16. The two best alignments deduced from TBM_2.

		1	2	3	4	5	6	7	8	9	10	11	12	13
		-	P	S	T	I	A	P	A	L	I	S	S	-
1	-	0	-10	-12	-14	-16	-18	-20	-22	-24	-26	-28	-30	-
2	P	-10	7	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-55
3	N	-12	-3	8	-2	-4	-6	-8	-10	-12	-14	-16	-18	-50
4	G	-14	-5	-2	6	-4	-4	-8	-8	-12	-14	-14	-16	-46
5	P	-16	-7	-4	-3	3	-5	3	-7	-9	-11	-13	-15	-43
6	I	-18	-9	-6	-5	1	2	-7	2	-5	-5	-12	-14	-40
7	R	-20	-11	-8	-7	-8	0	0	-8	0	-8	-6	-13	-37
8	D	-22	-13	-10	-9	-10	-10	-1	-2	-10	-3	-8	-6	-28
9	L	-24	-15	-12	-11	-7	-11	-11	-2	2	-8	-5	-10	-30
10	L	-26	-17	-14	-13	-9	-8	-13	-12	2	4	-6	-7	-25
11	L	-28	-19	-16	-15	-11	-10	-11	-14	-8	4	2	-8	-24
12	G	-30	-21	-18	-18	-19	-11	-12	-11	-10	-6	4	2	-12
13	K	-32	-23	-20	-19	-21	-20	-12	-13	-12	-8	-6	4	-8
14	D	-34	-25	-22	-21	-22	-22	-21	-14	-14	-10	-8	-6	-16
15	L	-36	-27	-24	-23	-19	-23	-23	-22	-10	-12	-10	-8	-
16	-	-	-55	-50	-47	-41	-43	-41	-38	-24	-24	-20	-	-

Table 25. The two best alignments deduced from TBM_2. Opening gap is 10, extending gap is 2 and we used BLOSUM62 substitution matrix.

Seq A	P	S	-	T	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	7	1	-10	-1	4	-1	-1	-1	4	2	0	0	-10	-2	=-8

This alignment is the same given by SIB's software, thus also in this case the correct execution of our program is verified. SIB's software gives a score of -12, which is -8-2-2, as SIB's software -as mentioned- adds the extending gap penalty also to the first a gap, even if we have only one gap.

1.7.4.1 Comparison with other substitution matrices

We will now run the program on the same sequences in Eq. 23 and settings in Eq. 24, but with the use other substitution matrices. The results with the use of BLOSUM80 are in *Table 26*, where the calculation given by SIB software is the second alignment. As you can see, although the two best alignments are different, they have the same score, thus the best score given by the two software is the same.

Table 26. Opening gap is 10, extending gap is 2 and we used BLOSUM80 substitution matrix. The second alignment is the one given by SIB's software.

Seq A	P	S	-	T	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	8	0	-10	-2	5	-2	-2	-2	4	1	-1	-1	-10	-2	-14

Seq A	P	S	T	-	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	8	0	-2	-10	5	-2	-2	-2	4	1	-1	-1	-10	-2	-14

If we run now our program with same sequences and settings, but with a scoring matrix given by BLOSUM50, we have the best alignment in *Table 27* with a score of -5. SIB software gives exactly the same alignment.

Table 27. Opening gap is 10, extending gap is 2 and we used BLOSUM50 substitution matrix. The same alignment is the one given by SIB's software.

Seq A	P	S	-	T	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	10	1	-10	-1	5	-2	-1	-2	5	2	0	0	-10	-2	-5

We will now use BLOSUM 45, with which both our program and SIB software give the alignment in *Table 28*.

Table 28. Opening gap is 10, extending gap is 2 and we used BLOSUM45 substitution matrix. The same alignment is the one given by SIB's software.

Seq A	P	S	-	T	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	9	1	-10	-1	5	-2	-1	-1	5	2	0	-1	-10	-2	-6

The last search is the one with PAM250 matrix and gives the alignment in *Table 29*. Both our program and SIB software give the same result with a score of -6, which is -4-2.

Table 29. Opening gap is 10, extending gap is 2 and we used PAM250 substitution matrix. The same alignment is the one given by SIB's software.

Seq A	P	S	T	I	A	P	A	L	I	S	S	-	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	6	1	0	-2	-1	0	0	6	2	-3	1	-10	-2	-2	-4

Figure 17. The best alignment deduced from TBM_2.

		1	2	3	4	5	6	7	8	9	10	11	12	13
		-	P	N	G	P	A	P	L	G	K	D	L	-
1	-	0	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44	-48	-
2	P	-8	10	2	-2	-6	-10	-14	-18	-22	-26	-30	-34	-90
3	N	-12	2	17	9	5	1	-3	-7	-11	-15	-19	-23	-75
4	G	-16	-2	9	25	17	13	9	5	1	-3	-7	-11	-59
5	P	-20	-6	5	17	35	27	23	19	15	11	7	3	-41
6	I	-24	-10	1	13	27	34	26	25	18	14	10	9	-31
7	R	-28	-14	-3	9	23	26	31	23	22	21	13	9	-27
8	D	-32	-18	-7	5	19	22	25	27	22	21	29	21	-11
9	L	-36	-22	-11	1	15	18	19	30	23	19	21	34	6
10	L	-40	-26	-15	-3	11	14	15	24	26	20	17	26	2
11	L	-44	-30	-19	-7	7	10	11	20	20	23	16	22	2
12	G	-48	-34	-23	-11	3	7	8	14	28	20	22	18	2
13	K	-52	-38	-27	-15	-1	2	6	10	20	34	26	22	10
14	D	-56	-42	-31	-19	-5	-2	1	6	16	26	42	34	26
15	L	-60	-46	-35	-23	-9	-6	-5	6	12	22	34	47	-
16	-	-	-90	-75	-59	-41	-34	-29	-14	-4	10	26	-	-

1.7.5 Fifth application

Consider the following sequences and settings:

Eq. 25
$$\begin{cases} seq.A = PNGPAPLGKDL \\ seq.B = PNGPIRDLLLGKDL \end{cases}$$

Eq. 26
$$\begin{cases} open\ gap\ penalty = 8 \\ extended\ gap\ penalty = 4 \\ substitution\ matrix = BLOSUM50 \end{cases}$$

Matrix TBM and the alignment deduced by TBM_2 is reported in *Figure 17* and the alignment is also reported in
Table 31. SIB's software gives the same alignment with a score of 44, which is 47-4.

1.7.6 Sixth application

Consider the following sequences and settings:

Eq. 27
$$\begin{cases} seq.A = GPSKGDLLGKDL \\ seq.B = PNAGPSKGIRDLLLGKDL \end{cases}$$

Eq. 28
$$\begin{cases} open\ gap\ penalty = 4 \\ extended\ gap\ penalty = 2 \\ substitution\ matrix = BLOSUM50 \end{cases}$$

In *Figure 18* you can find the best alignment traced on TBM, while the alignment with its score is in **Errore. L'origine riferimento non è stata trovata.**. SIB software doesn't seem to calculate this a lignment. I don't know why.

Figure 18. The best alignment deduced from TBM_2.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
		-	G	P	S	K	G	D	L	L	G	K	D	L	-
1	-	0	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-26	-
2	P	-4	-2	6	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-52
3	N	-6	-4	2	7	3	1	0	-3	-5	-7	-9	-10	-13	-47
4	A	-8	-6	0	3	6	3	0	-2	-4	-5	-8	-10	-12	-44
5	G	-10	0	-2	1	2	14	10	8	6	4	2	0	-2	-32
6	P	-12	-4	10	6	4	10	13	9	7	5	3	1	-1	-29
7	S	-14	-6	6	15	11	9	10	10	6	7	5	3	-1	-27
8	K	-16	-8	4	11	21	17	15	13	11	9	13	9	7	-17
9	G	-18	-8	2	9	17	29	25	23	21	19	17	15	13	-9
10	I	-20	-12	0	7	15	25	25	27	25	21	19	17	17	-3
11	R	-22	-14	-2	5	13	23	23	23	24	22	24	20	18	0
12	D	-24	-16	-4	3	11	21	31	27	25	23	21	32	28	12
13	L	-26	-18	-6	1	9	19	27	36	32	30	28	28	37	23
14	L	-28	-20	-8	-1	7	17	25	32	41	37	35	33	33	21
15	L	-30	-22	-10	-3	5	15	23	30	37	37	34	31	38	28
16	G	-32	-22	-12	-5	3	13	21	28	35	45	41	39	37	29
17	K	-34	-26	-14	-7	1	11	19	26	33	41	51	47	45	39
18	D	-36	-28	-16	-9	-1	9	19	24	31	39	47	59	55	51
19	L	-38	-30	-18	-11	-3	7	15	24	29	37	45	55	64	-
20	-	-	-54	-40	-31	-21	-9	1	12	19	29	39	51	-	-

Table 30. Opening gap is 4, extending gap is 2 and we used BLOSUM50 substitution matrix.

Seq A	-	-	-	G	P	S	K	G	-	-	D	-	L	L	G	K	D	L	
Seq B	P	N	A	G	P	S	K	G	I	R	D	L	L	L	G	K	D	L	
Score	-4	-2	-2	8	10	5	6	8	-4	-2	8	-4	5	5	8	6	8	5	=64

Table 31. Opening gap is 8, extending gap is 4 and we used BLOSUM50 substitution matrix. The same alignment is the one given by SIB's software.

Seq A	P	N	G	P	A	P	-	-	-	L	G	K	D	L	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	10	7	8	10	-1	-3	-8	-4	-4	5	8	6	8	5	=47

1.7.7 Seventh application

Consider the following sequences and settings:

Eq. 29
$$\begin{cases} seq.A = MSILKIHAREIFDSRGNPTVEVDLFTSKGLFRAAVPSGASTGIYEALELR \\ seq.B = MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV \end{cases}$$

Eq. 30
$$\begin{cases} open\ gap\ penalty = 12 \\ extended\ gap\ penalty = 2 \\ substitution\ matrix = BLOSUM50 \end{cases}$$

Our program gives the same best alignment as SIB software, with a score of 155. You can find this alignment in *Table 32*. The score given by SIB's software is 155-2-2-2=149.

Table 32. Opening gap is 12, extending gap is 4 and we used BLOSUM50 substitution matrix.

seq. A:	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEALELR
seq. B:	MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV---

1.7.8 Eighth application[2]

We run again our software with settings of example in 1.7.1, but this time substitution matrix is scaled in order to have its elements expressed as logarithms to base two of odds ratios. Our software gives these two best alignments:

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	-	S	S	
Score	0.5	-2	-0.5	2	-0.5	-0.5	-0.5	2	1	-2	0	0	=-0.5

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	S	S	-	
Score	0.5	-2	-0.5	2	-0.5	-0.5	-0.5	2	1	0	0	-2	=-0.5

SIBO's software gives the first one above, with a score of -1.

[2] From this application on, I use the software with the substitution matrices scaled in order to have elements represented by logarithms to base two of odds ratios.

1.7.9 Ninth application

We repeat example in 1.7.2 with matrix scaled to a base two logarithm. We obtain:

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	-	S	S	
Score	0.5	-4	-0.5	2	-0.5	-0.5	-0.5	2	1	-4	0	0	=-4.5

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	-	T	I	A	P	A	L	I	S	S	-	
Score	0.5	-4	-0.5	2	-0.5	-0.5	-0.5	2	1	0	0	-4	=-4.5

SIB's software gives the second alignment, with a score of -3.

1.7.10 Tenth application

We will run our software on example in 1.7.3 with the substitution matrix scaled to a logarithm to base two. We have:

Seq A	N	G	P	I	R	D	L	L	L	G	K	D	
Seq B	S	T	I	A	P	A	L	I	-	-	S	S	
Score	0.5	-1	-1.5	-0.5	-1	-1	2	1	-12	-2	0	0	=-15.5

SIBO's software gives the same alignment with a score of -19.

1.7.11 Eleventh application

We re-calculate example in 1.7.4 with a scaled substitution matrix. We obtain:

Seq A	P	S	T	I	A	P	A	L	I	-	-	-	S	S	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	3.5	0.5	-1	-1.5	-0.5	-1	-1	2	1	-10	-2	-2	0	-1	=-13

Seq A	P	S	T	I	A	P	A	L	I	S	S	-	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	3.5	0.5	-1	-1.5	-0.5	-1	-1	2	1	-1	0	-10	-2	-2	=-13

SIB's software gives the second alignment below, with a score of -12.

Seq A	P	S	-	T	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	3.5	0.5	-10	-0.5	2	-0.5	-0.5	-0.5	2	1	0	0	-10	-2	=-15

As you can see, in this case our software gives an alignment with a score different from the one given by SIB's software. I can't explain this.

1.7.11.1 Comparison with other substitution matrices

Same sequences and settings as above, but with BLOSUM80 matrix. We have:

Seq A	P	S	T	I	A	P	A	L	I	-	-	-	S	S	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	4	0	-1	-2	-1	-1	-1	2	0,5	-10	-2	-2	-0,5	-1,5	=-15,5

Seq A	P	S	T	I	A	P	A	L	I	S	S	-	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	4	0	-1	-2	-1	-1	-1	2	0,5	-1,5	-0,5	-10	-2	-2	=-15,5

SIB's software gives instead:

Seq A	P	S	T	-	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	4	0	-1	-10	2,5	-1	-1	-0,5	2	0,5	-0,5	-0,5	-10	-2	=-17,5

In this case we obtain an incongruent result, too: our software find a better alignment. Using BLOSUM50 substitution matrix we have:

Seq A	P	S	T	I	A	P	A	L	I	-	-	-	S	S	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	5	0,5	-1	-1,5	-0,5	-1,5	-1	2,5	1	-10	-2	-2	0	-1,5	=-12

Seq A	P	S	T	I	A	P	A	L	I	S	S	-	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	5	0,5	-1	-1,5	-0,5	-1,5	-1	2,5	1	-1,5	0	-10	-2	-2	=-12

SIB's software gives instead:

Seq A	P	S	-	T	I	A	P	A	L	I	S	S	-	-	
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L	
Score	5	0,5	-10	-0,5	2,5	-1	-0,5	-1	2,5	1	0	0	-10	-2	=-13,5

Again, a difference between the two software. With PAM250 we have the following alignment with both the programs. The score is -10,667.

Seq A	P	S	T	I	A	P	A	L	I	S	S	-	-	-
Seq B	P	N	G	P	I	R	D	L	L	L	G	K	D	L

1.7.12 Twelfth application

The same settings as in 1.7.6 give the following alignment, with a score of 56.5. SIB's software gives the same alignment.

seq. A:	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEALELR
seq. B:	MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV---

In conclusion, our software has a closer behavior with SIB's software if we do not scale substitution matrices.

1.7.13 Comparison with SIB's software

From application one to seven, we have been able to verify that our software gives the same best score as SIB's software, if only we take into account that SIB's software doesn't scale the substitution matrix of choice to a matrix expressed in bits: it does maintain BLOSUM matrices as expressed as logarithms to base 2 multiplied by 2, and PAM matrices as logarithm to base 10 multiplied by 10. Moreover, SIB's software adds the extending gap penalty to each first gap, even when the first gap is the only gap. We present in the following paragraph a version of code in paragraph 1.7 which is equivalent to SIB's software (it doesn't scale matrices and adds extending gap penalty also to the first gap).

1.8 Code in Octave for global alignment equivalent to SIB's LALIGN

The following one is a version of code in paragraph 1.7 where the following changes have been operated:

- substitution matrices are not scaled to logarithm to the base two of odds ratios;
- extending gap penalty is added to the opening gap penalty, always;
- end gap penalties are included.

These settings have been implemented in order to obtain a software equivalent to the tool LALIGN offered by the *Swiss Institute of Bioinformatics* (SIB) web site (↥).

```
% file name = NeW_6.m
% date of creation = 03/05/2016
% this software is equivalent to LALIGN by Swiss Institute of Bioinformatics
% it doesn't scale substitution matrices to log to base two
% it adds exending gap penalty to opening gap penalty
% We delete all previous values for variables
clear all
% We ask the user to choose a substitution matrix
SUBMAT = 0;
disp(" ")
disp("Please, select the substitution matrix to use for score calculation.")
disp(" ")
s = menu ("Substitution matrix", "BLOSUM45", "BLOSUM50", "BLOSUM62", "BLOSUM80", "PAM250");
% We load the chosen substitution matrix matrix
switch s
  case (1)
  load BLOSUM45.mat;
  SUBMAT=BLOSUM45;
  disp(" ")
  disp("You have chosen the substitution matrix BLOSUM45:")
  disp(" ")
  disp(BLOSUM45)
  case (2)
  load BLOSUM50.mat;
  SUBMAT=BLOSUM50;
  disp(" ")
  disp("You have chosen the substitution matrix BLOSUM50:")
  disp(" ")
  disp(BLOSUM50)
  case (3)
  load BLOSUM62.mat;
  SUBMAT=BLOSUM62;
  disp(" ")
```

```
 disp("You have chosen the substitution matrix BLOSUM62:")
 disp(" ")
 disp(BLOSUM62)
 case (4)
 load BLOSUM80.mat;
 SUBMAT=BLOSUM80;
 disp(" ")
 disp("You have chosen the substitution matrix BLOSUM80:")
 disp(" ")
 disp(BLOSUM80)
 otherwise
 load PAM250.mat;
 SUBMAT=PAM250;
 disp(" ")
 disp("You have chosen the substitution matrix PAM250:")
 disp(" ")
 disp(PAM250)
endswitch
% We define an array for the 20 amino acids
aa(1:20)= ['A' 'R' 'N' 'D' 'C' 'Q' 'E' 'G' 'H' 'I' 'L' 'K' 'M' 'F' 'P' 'S' 'T' 'W' 'Y' 'V'];
disp("Amino acids are intended in the following sequence:")
disp(" ")
disp(aa(1:20))
disp(" ")
% We ask the user for amino acids of sequence A
disp(" ")
disp("I will ask you for amino acids of sequence A.")
disp("Use capitol letters!")
disp(" ")
seq_A = input("Insert sequence A= ","s");
disp(" ")
A = ["-" seq_A];
% We ask the user for amino acids of sequence B
disp(" ")
disp("I will now ask you for amino acids of sequence B.")
disp("Again, use capitol letters!")
disp(" ")
seq_B = input("Insert sequence B= ","s");
disp(" ")
B = ["-" seq_B];
% We define length of sequence A and of sequence B, respectively
k=length(A)-1;
m=length(B)-1;
% We ask the user for the opening gap penalty
disp(" ")
disp("I will ask you for opening gap penalty.")
disp("Use a positive integer (suggested:12)!")
disp(" ")
op_d = input("Insert open gap penalty= ");
disp(" ")
% We ask the user for the extending gap penalty
disp(" ")
disp("I will ask you for extended gap penalty.")
disp("Use a positive integer (suggested:2)!")
disp(" ")
ex_d = input("Insert extended gap penalty= ");
disp(" ")
% We define elements of MAT, TBM and TBM_2 which are zero
MAT(1,1)=0;
MAT(1,2:k+1)=0;
MAT(2:m+1,1)=0;
```

```
TBM(1,1)=0;
TBM_2 = 0;
% We define elements of TBM and TBM_2 which have no role in calculation
% with a formal value of 100
TBM(1,k+2)=100;
TBM(m+2,1)=100;
TBM(m+1,k+2)=100;
TBM(m+2,k+2)=100;
TBM(m+2,k+1)=100;
TBM_2(1,1:k+2,1:2)=0;
TBM_2(1:m+2,1,1:2)=0;
TBM_2(m+1,k+2,1:2)=100;
TBM_2(m+2,k+2,1:2)=100;
TBM_2(m+2,k+1,1:2)=100;
% We define other elements of the first row and the first column of TBM
for i=2:1:m+1
  TBM(i,1)=-op_d-ex_d*(i-1);
  for j=2:1:k+1
    TBM(1,j)=-op_d-ex_d*(j-1);
  endfor
endfor
% We define elements of MAT
for i=2:1:m+1
  for j=2:1:k+1
    for h=1:1:20
      if ( A(j)==aa(h) )
        index_A=h;
      endif
      if ( B(i)==aa(h) )
        index_B=h;
      endif
    endfor
    MAT(i,j)= SUBMAT(index_B, index_A);
  endfor
endfor
% We print array MAT on the screen
disp(" ")
disp('MAT matrix contains the score between each pair of amino acids:')
disp(MAT(2:m+1,2:k+1))
disp(" ")
% We define elements for TBM and TBM_2
for i=2:1:m+1
  for j=2:1:k+1
    for l=1:1:j-1
      AL(i,l)=TBM(i,l)-op_d-ex_d*(j-l);
    endfor
    for g=1:1:i-1
      BE(g,j)=TBM(g,j)-op_d-ex_d*(i-g);
    endfor
    if ( TBM(i-1,j-1)+MAT(i,j) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
      TBM(i,j)=TBM(i-1,j-1)+MAT(i,j);
      TBM_2(i,j,1)=i-1;
      TBM_2(i,j,2)=j-1;
    else
      for l=1:1:j-1
        if ( AL(i,l) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
          TBM(i,j)=AL(i,l);
          TBM_2(i,j,1)=i;
          TBM_2(i,j,2)=l;
        else
          for g=1:1:i-1
```

```
          if ( BE(g,j) >= max([AL(i,1:j-1) max(BE(1:i-1,j))]) )
            TBM(i,j)=BE(g,j);
            TBM_2(i,j,1)=g;
            TBM_2(i,j,2)=j;
          endif
        endfor
      endif
    endfor
  endif
 endfor
endfor
% We define elements for the last column of TBM and TBM_2
for g=2:1:m
 TBM(g,k+2)=TBM(g,k+1) -op_d-ex_d*(m+1-g);
 TBM_2(g,k+2,1)=g;
 TBM_2(g,k+2,2)=k+1;
endfor
% We define elements for the last row of TBM and TBM_2
for l=2:1:k
 TBM(m+2,l)=TBM(m+1,l) -op_d-ex_d*(k+1-l);
 TBM_2(m+2,l,1)=m+1;
 TBM_2(m+2,l,2)=l;
endfor
% We print arrays TBM and TBM_2
disp(" ")
disp('TBM matrix contains the score between each partial alignments:')
disp(TBM)
disp('TBM_2 matrix contains the record of where each partial alignment comes from:')
disp(TBM_2)
disp(" ")
% We put into array ALIGN the sequence of elements of A and B
% which give the best alignment
% Here we start with the first row of ALIGN
%
%
control=0;
if ( TBM(m+1,k+1) >= max([max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
  ALIGN_a(1:2,1:1001)=0;
  ALIGN_a(1,1)= k+1;
  ALIGN_a(2,1)= m+1;
  best_score= TBM(m+1,k+1);
  control=1;
endif
if (control==1)
 % Now we define the following rows of ALIGN_a
 for n=1:1:1000
  if ( ALIGN_a(1,n)>0 )
   if ( ALIGN_a(2,n)>0 )
     ALIGN_a(1,n+1)=TBM_2(ALIGN_a(2,n),ALIGN_a(1,n),2);
     ALIGN_a(2,n+1)=TBM_2(ALIGN_a(2,n),ALIGN_a(1,n),1);
   endif
  endif
 endfor
 % We search for the numebr of elements of ALIGN_a
 n=1;
 while (ALIGN_a(1,n)>0)
  n=n+1;
 endwhile
 n=n-1;
 % We reverse the order of columns of ALIGN_a
 ALIGN_a(1:2,1:n)=fliplr(ALIGN_a(1:2,1:n));
```

```
% We re-write ALIGN_a in ALIGNex_a
ALIGNex_a(:,1)=ALIGN_a(:,1);
i=1;
j=1;
while (ALIGN_a(1,j)>0)
  if ( ALIGN_a(1,j+1)==ALIGN_a(1,j) )
    delta=(ALIGN_a(2,j+1)-ALIGN_a(2,j));
    for s=1:1:delta
      ALIGNex_a(1,i+s)=ALIGN_a(1,j);
      ALIGNex_a(2,i+s)=ALIGN_a(2,j)+s;
    endfor
    i=i+delta;
    j=j+1;
  elseif ( ALIGN_a(2,j+1)==ALIGN_a(2,j) )
    delta=(ALIGN_a(1,j+1)-ALIGN_a(1,j));
    for s=1:1:delta
      ALIGNex_a(1,i+s)=ALIGN_a(1,j)+s;
      ALIGNex_a(2,i+s)=ALIGN_a(2,j);
    endfor
    i=i+delta;
    j=j+1;
  else
    ALIGNex_a(1,i+1)=ALIGN_a(1,j+1);
    ALIGNex_a(2,i+1)=ALIGN_a(2,j+1);
    i=i+1;
    j=j+1;
  endif
endwhile
%
% We search for the numebr of elements of ALIGNex_c
i=1;
while (ALIGNex_a(1,i)>0)
  i=i+1;
endwhile
i=i-1;
%
% We put ALIGNex_a in ALIGN_a
ALIGN_a(:,1:i)=ALIGNex_a(:,1:i);
% We calculate the array ALIGN_a_2 which contains the alignment
ALIGN_a_2(1,1)=A(ALIGN_a(1,1));
ALIGN_a_2(2,1)=B(ALIGN_a(2,1));
j=2;
while (ALIGN_a(1,j)>0)
  if ( ALIGN_a(1,j)==ALIGN_a(1,j-1) )
    ALIGN_a_2(1,j)='-';
    ALIGN_a_2(2,j)=B(ALIGN_a(2,j));
  elseif ( ALIGN_a(2,j)==ALIGN_a(2,j-1) )
    ALIGN_a_2(1,j)=A(ALIGN_a(1,j));
    ALIGN_a_2(2,j)='-';
  else
    ALIGN_a_2(1,j)=A(ALIGN_a(1,j));
    ALIGN_a_2(2,j)=B(ALIGN_a(2,j));
  endif
  j=j+1;
endwhile
%
if (ALIGN_a(1,1)!=1)
  for h=1:1:ALIGN_a(1,1)
    pre_ALIGN_a_2(1,h)=A(h);
    pre_ALIGN_a_2(2,h)="-";
  endfor
```

```
    ALIGN_a_2=[pre_ALIGN_a_2(:,1:ALIGN_a(1,1)) ALIGN_a_2(:,2:j)];
   endif
   if (ALIGN_a(2,1)!=1)
    for h=1:1:ALIGN_a(2,1)
     pre_ALIGN_a_2(1,h)="-";
     pre_ALIGN_a_2(2,h)=B(h);
    endfor
    ALIGN_a_2=[pre_ALIGN_a_2 ALIGN_a_2(:,2:j-1)];
   endif
   %
   disp(" ")
   disp("One of the possible alignments with the best score is:")
   disp(" ")
   disp(["seq. A: " ALIGN_a_2(1,:)])
   disp(["seq. B: " ALIGN_a_2(2,:)])
   disp(" ")
   disp("The path through TBM, from the end to the beginning, is:")
   disp(" ")
   disp(ALIGN_a(1:2,1:j))
   disp(" ")
   % We print on the screen the best score
   disp(" ")
   disp('The best score for alignments is:')
   disp(" ")
   disp(best_score)
   disp(" ")
  endif
  % We searh for another best alignment
  %
  %
  control=0;
  for g=2:1:m
   if ( TBM(g,k+2) >= max([TBM(m+1,k+1) max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
    ALIGN_b(1:2,1:1001)=0;
    ALIGN_b(1,1)= TBM_2(g,k+2,2);
    ALIGN_b(2,1)= TBM_2(g,k+2,1);
    best_score= TBM(g,k+2);
    control=1;
   endif
  endfor
  if (control==1)
   % Now we define the following rows of ALIGN
   for n=1:1:1000
    if ( ALIGN_b(1,n)>0 )
     if ( ALIGN_b(2,n)>0 )
      ALIGN_b(1,n+1)=TBM_2(ALIGN_b(2,n),ALIGN_b(1,n),2);
      ALIGN_b(2,n+1)=TBM_2(ALIGN_b(2,n),ALIGN_b(1,n),1);
     endif
    endif
   endfor
   % We search for the numebr of elements of ALIGN
   n=1;
   while (ALIGN_b(1,n)>0)
    n=n+1;
   endwhile
   n=n-1;
   % We reverse the ordero of columns of ALIGN_b
   ALIGN_b(1:2,1:n)=fliplr(ALIGN_b(1:2,1:n));
   % We re-write ALIGN_b in ALIGNex_b
   ALIGNex_b(:,1)=ALIGN_b(:,1);
   i=1;
```

```
j=1;
while (ALIGN_b(1,j)>0)
  if ( ALIGN_b(1,j+1)==ALIGN_b(1,j) )
    delta=(ALIGN_b(2,j+1)-ALIGN_b(2,j));
    for s=1:1:delta
      ALIGNex_b(1,i+s)=ALIGN_b(1,j);
      ALIGNex_b(2,i+s)=ALIGN_b(2,j)+s;
    endfor
    i=i+delta;
    j=j+1;
  elseif ( ALIGN_b(2,j+1)==ALIGN_b(2,j) )
    delta=(ALIGN_b(1,j+1)-ALIGN_b(1,j));
    for s=1:1:delta
      ALIGNex_b(1,i+s)=ALIGN_b(1,j)+s;
      ALIGNex_b(2,i+s)=ALIGN_b(2,j);
    endfor
    i=i+delta;
    j=j+1;
  else
    ALIGNex_b(1,i+1)=ALIGN_b(1,j+1);
    ALIGNex_b(2,i+1)=ALIGN_b(2,j+1);
    i=i+1;
    j=j+1;
  endif
endwhile
%
% We search for the numebr of elements of ALIGNex_b
i=1;
while (ALIGNex_b(1,i)>0)
  i=i+1;
endwhile
i=i-1;
%
% We put ALIGNex_b in ALIGN_b
ALIGN_b(:,1:i)=ALIGNex_b(:,1:i);
% We calculate the array ALIGN_b_2 which contains the alignment
ALIGN_b_2(1,1)=A(ALIGN_b(1,1));
ALIGN_b_2(2,1)=B(ALIGN_b(2,1));
j=2;
while (ALIGN_b(1,j)>0)
  if ( ALIGN_b(1,j)==ALIGN_b(1,j-1) )
    ALIGN_b_2(1,j)='-';
    ALIGN_b_2(2,j)=B(ALIGN_b(2,j));
  elseif ( ALIGN_b(2,j)==ALIGN_b(2,j-1) )
    ALIGN_b_2(1,j)=A(ALIGN_b(1,j));
    ALIGN_b_2(2,j)='-';
  else
    ALIGN_b_2(1,j)=A(ALIGN_b(1,j));
    ALIGN_b_2(2,j)=B(ALIGN_b(2,j));
  endif
  j=j+1;
endwhile
%
if (ALIGN_b(1,1)!=1)
  for h=1:1:ALIGN_b(1,1)
    pre_ALIGN_b_2(1,h)=A(h);
    pre_ALIGN_b_2(2,h)="-";
  endfor
  ALIGN_b_2=[pre_ALIGN_b_2(:,1:ALIGN_b(1,1)) ALIGN_b_2(:,2:j)];
endif
if (ALIGN_b(2,1)!=1)
```

```
    for h=1:1:ALIGN_b(2,1)
      pre_ALIGN_b_2(1,h)="-";
      pre_ALIGN_b_2(2,h)=B(h);
    endfor
    ALIGN_b_2=[pre_ALIGN_b_2 ALIGN_b_2(:,2:j-1)];
  endif
  %
  for p=1:1:m+1-ALIGN_b(2,i)
    ALIGN_b_2(1,i+p)="-";
    ALIGN_b_2(2,i+p)=B(ALIGN_b(2,i)+p);
  endfor
  disp(" ")
  disp("One of the possible alignments with the best score is:")
  disp(" ")
  disp(["seq. A: " ALIGN_b_2(1,:)])
  disp(["seq. B: " ALIGN_b_2(2,:)])
  disp(" ")
  disp("The path through TBM, from the end to the beginning, is:")
  disp(" ")
  disp(ALIGN_b(1:2,1:i))
  disp(" ")
  % We print on the screen the best score
  disp(" ")
  disp('The best score for alignments is:')
  disp(" ")
  disp(best_score)
  disp(" ")
endif
% We searh for another best alignment
%
%
control=0;
for l=2:1:k
  if ( TBM(m+2,l) >= max([TBM(m+1,k+1) max(TBM(m+2,2:k)) max(TBM(2:m,k+2))]) )
    ALIGN_c(1:2,1:1001)=0;
    ALIGN_c(1,1)= TBM_2(m+2,l,2);
    ALIGN_c(2,1)= TBM_2(m+2,l,1);
    best_score= TBM(m+2,l);
    control=1;
  endif
endfor
if (control==1)
  % Now we define the following rows of ALIGN
  for n=1:1:1000
    if ( ALIGN_c(1,n)>0 )
      if ( ALIGN_c(2,n)>0 )
        ALIGN_c(1,n+1)=TBM_2(ALIGN_c(2,n),ALIGN_c(1,n),2);
        ALIGN_c(2,n+1)=TBM_2(ALIGN_c(2,n),ALIGN_c(1,n),1);
      endif
    endif
  endfor
  % We search for the numebr of elements of ALIGN
  n=1;
  while (ALIGN_c(1,n)>0)
    n=n+1;
  endwhile
  n=n-1;
  % We reverse the ordero of columns of ALIGN_c
  ALIGN_c(1:2,1:n)=fliplr(ALIGN_c(1:2,1:n));
  % We re-write ALIGN_b in ALIGNex_c
  ALIGNex_c(:,1)=ALIGN_c(:,1);
```

```
i=1;
j=1;
while (ALIGN_c(1,j)>0)
  if ( ALIGN_c(1,j+1)==ALIGN_c(1,j) )
    delta=(ALIGN_c(2,j+1)-ALIGN_c(2,j));
    for s=1:1:delta
      ALIGNex_c(1,i+s)=ALIGN_c(1,j);
      ALIGNex_c(2,i+s)=ALIGN_c(2,j)+s;
    endfor
    i=i+delta;
    j=j+1;
  elseif ( ALIGN_c(2,j+1)==ALIGN_c(2,j) )
    delta=(ALIGN_c(1,j+1)-ALIGN_c(1,j));
    for s=1:1:delta
      ALIGNex_c(1,i+s)=ALIGN_c(1,j)+s;
      ALIGNex_c(2,i+s)=ALIGN_c(2,j);
    endfor
    i=i+delta;
    j=j+1;
  else
    ALIGNex_c(1,i+1)=ALIGN_c(1,j+1);
    ALIGNex_c(2,i+1)=ALIGN_c(2,j+1);
    i=i+1;
    j=j+1;
  endif
endwhile
%
% We search for the numebr of elements of ALIGNex_c
i=1;
while (ALIGNex_c(1,i)>0)
  i=i+1;
endwhile
i=i-1;
%
% We put ALIGNex_c in ALIGN_c
ALIGN_c(:,1:i)=ALIGNex_c(:,1:i);
% We calculate the array ALIGN_c_2 which contains the alignment
ALIGN_c_2(1,1)=A(ALIGN_c(1,1));
ALIGN_c_2(2,1)=B(ALIGN_c(2,1));
j=2;
while (ALIGN_c(1,j)>0)
  if ( ALIGN_c(1,j)==ALIGN_c(1,j-1) )
    ALIGN_c_2(1,j)='-';
    ALIGN_c_2(2,j)=B(ALIGN_c(2,j));
  elseif ( ALIGN_c(2,j)==ALIGN_c(2,j-1) )
    ALIGN_c_2(1,j)=A(ALIGN_c(1,j));
    ALIGN_c_2(2,j)='-';
  else
    ALIGN_c_2(1,j)=A(ALIGN_c(1,j));
    ALIGN_c_2(2,j)=B(ALIGN_c(2,j));
  endif
  j=j+1;
endwhile
%
if (ALIGN_c(1,1)!=1)
  for h=1:1:ALIGN_c(1,1)
    pre_ALIGN_c_2(1,h)=A(h)
    pre_ALIGN_c_2(2,h)="-"
  endfor
  ALIGN_c_2=[pre_ALIGN_c_2(:,1:ALIGN_c(1,1)) ALIGN_c_2(:,2:j)]
endif
```

```
 if (ALIGN_c(2,1)!=1)
  for h=1:1:ALIGN_c(2,1)
   pre_ALIGN_c_2(1,h)="-"
   pre_ALIGN_c_2(2,h)=B(h)
  endfor
  ALIGN_c_2=[pre_ALIGN_c_2 ALIGN_c_2(:,2:j-1)]
 endif
 %
 for p=1:1:k+1-ALIGN_c(1,i)
  ALIGN_c_2(1,i+p)=A(ALIGN_c(1,i)+p);
  ALIGN_c_2(2,i+p)="-";
 endfor
 disp(" ")
 disp("One of the possible alignments with the best score is:")
 disp(" ")
 disp(["seq. A: " ALIGN_c_2(1,:)])
 disp(["seq. B: " ALIGN_c_2(2,:)])
 disp(" ")
 disp("The path through TBM, from the end to the beginning, is:")
 disp(" ")
 disp(ALIGN_c(1:2,1:i))
 disp(" ")
 % We print on the screen the best score
 disp(" ")
 disp('The best score for alignments is:')
 disp(" ")
 disp(best_score)
 disp(" ")
endif
```

1.8.1 First group of comparisons of the new code with SIB's software

We will now test the new code in paragraph 1.8 again in comparison with SIB's software, in order to verify that the two programs give the same result in term of best score. We will run the test for sequences in Eq. 19 for various gap penalties and substitution matrices.

Sequences	**Sub. matrix**	**Gap penalty**		**Alignment**	**Score**
NGPIRDLLLGKD STIAPALISS	**BLOSUM45**	**open**	**extended**		
SIB's software		2	2	NGPIRDLLLGKD S-TIAPALISS-	-1
My software				NGPIRDLLLGKD S-TIAPALI-SS	-1
				NGPIRDLLLGKD S-TIAPALISS-	
SIB's software		4	1	N--GPIRDLLLGKD STIAP---ALIS-S	-2
My software				NGPIRDLLLGKD S-TIAPALIS-S	-2
SIB's software		12	2	NGPIRDLLLGKD STIAPALI--SS	-18
My software				NGPIRDLLLGKD STIAPALI--SS	-18

Sequences	Sub. matrix	Gap penalty		Alignment	Score
NGPIRDLLLGKD STIAPALISS	BLOSUM50	open	extended		
SIB's software		2	2	NGPIRDLLLGKD S-TIAPALISS-	-1
My software				NGPIRDLLLGKD S-TIAPALI-SS	-1
				NGPIRDLLLGKD S-TIAPALISS-	
SIB's software		4	1	N--GPIRDLLLGKD STIAPA--LI--SS	-1
My software				N--GPIRDLLLGKD STIAPA---LI-SS	-1
				N--GPIRDLLLGKD STIAPA---LISS-	
SIB's software		12	2	NGPIRDLLLGKD STIAPALI--SS	-19
My software				NGPIRDLLLGKD STIAPALI--SS	-19

Sequences	Sub. matrix	Gap penalty		Alignment	Score
NGPIRDLLLGKD STIAPALISS	BLOSUM62	open	extended		
SIB's software		2	2	NGPIRDLLLGKD S-TIAPALIS-S	0
My software				NGPIRDLLLGKD S-TIAPALIS-S	0
SIB's software		4	1	NGPIRDLLLGKD S-TIAPALISS-	-3
My software				NGPIRDLLLGKD S-TIAPALISS-	-3
SIB's software		12	2	NGPIRDLLLGKD STIAPALI--SS	-19
My software				NGPIRDLLLGKD STIAPALI--SS	-19

Sequences	Sub. matrix	Gap penalty		Alignment	Score
NGPIRDLLLGKD STIAPALISS	BLOSUM80	open	extended		
SIB's software		2	2	N--GPIRDLLLGKD STIAP---ALISS-	-2
My software				NGPIRDLLLGKD S-TIAPALI-SS	-8
				NGPIRDLLLGKD S-TIAPALISS-	
SIB's software		4	1	N--GPIRDLLLGKD STIAP---ALISS-	-2
My software				N--GPIRDLLLGKD STIAP---ALI-SS	-9
				N--GPIRDLLLGKD STIAP---ALISS-	
SIB's software		12	2	NGPIRDLLLGKD ST-IAPALISS-	-26
My software				NGPIRDLLLGKD STIAPALI--SS	-25

Sequences	Sub. matrix	Gap penalty		Alignment	Score
NGPIRDLLLGKD STIAPALISS	PAM250	open	extended		
SIB's software		2	2	NGPIRDLLLGKD ST-IAPALISS-	2
My software				Error (array out of bound)	-
SIB's software		4	1	NGPIRDLLLGKD ST-IAPALISS-	0
My software				NGPIRDLLLGKD STIAPALI--SS	0
SIB's software		12	2	NGPIRDLLLGKD STIAPALI--SS	-10
My software				NGPIRDLLLGKD STIAPALI--SS	-10

1.8.2 Second group of comparisons of the new code with SIB's software

We will now use sequences in paragraph 1.7.4 with various substitution matrices and gap penalties.

Sequences	Sub. matrix	Gap penalty		Alignment	Score
NGPIRDLLLGKD STIAPALISS	BLOSUM45	open	extended		
SIB's software		2	2	NGPIRDLLLGKD S-TIAPALISS-	-1
My software				NGPIRDLLLGKD S-TIAPALI-SS	-1
				NGPIRDLLLGKD S-TIAPALISS-	
SIB's software		4	1	N--GPIRDLLLGKD STIAP---ALIS-S	-2
My software				NGPIRDLLLGKD S-TIAPALIS-S	-2
SIB's software		12	2	NGPIRDLLLGKD STIAPALI--SS	-18
My software				NGPIRDLLLGKD STIAPALI--SS	-18

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	BLOSUM45	open	extended		
SIB's software		10	2	PS-TIAPALISS-- PNGPIRDLLLGKDL	-10
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	-10
				PS-TIAPALISS-- PNGPIRDLLLGKDL	
SIB's software		4	1	PSTIAPA---LISS-- PN--GPIRDLLLGKDL	5
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	5
				PS-TIAPALISS-- PNGPIRDLLLGKDL	

SIB's software	2	2	PS-TIAPALISS-- PNGPIRDLLLGKDL	6
My software			PS-TIAPALISS-- PNGPIRDLLLGKDL	6

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	BLOSUM50	open	extended		
SIB's software		10	2	PS-TIAPALISS-- PNGPIRDLLLGKDL	-9
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	-9
SIB's software		4	1	PSTIAPA---LISS-- PN--GPIRDLLLGKDL	8
My software				PSTIAPA---LISS-- PN--GPIRDLLLGKDL	8
SIB's software		2	2	PSTIAPA---LISS-- PN--GPIRDLLLGKDL	7
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	7

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	BLOSUM62	open	extended		
SIB's software		10	2	PS-TIAPALISS-- PNGPIRDLLLGKDL	-12
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	-12
SIB's software		4	1	PS-TIAPALISS-- PNGPIRDLLLGKDL	3
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	3
SIB's software		2	2	PS-TIAPALISS-- PNGPIRDLLLGKDL	4
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	4

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	BLOSUM80	open	extended		
SIB's software		10	2	PST-IAPALISS-- PNGPIRDLLLGKDL	-12
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	-18
SIB's software		4	1	PSTIAPA---LISS-- PN--GPIRDLLLGKDL	9
My software				PSTIAP---ALISS-- PN--GPIRDLLLGKDL	-2
SIB's software		2	2	PSTIAPA---LISS-- PN--GPIRDLLLGKDL	8
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	-2

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	**PAM250**	**open**	**extended**		
SIB's software		10	2	PSTIAPALISS--- PNGPIRDLLLGKDL	-6
My software				PSTIAPALISS--- PNGPIRDLLLGKDL	-6
SIB's software		4	1	PST-IAPALISS-- PNGPIRDLLLGKDL	5
My software				P-STIAPALISS-- PNGPIRDLLLGKDL	5
SIB's software		2	2	PST-IAPALISS-- PNGPIRDLLLGKDL	6
My software				P-STIAPALISS-- PNGPIRDLLLGKDL	6

1.8.3 Third group of comparisons of the new code with SIB's software

We will now use sequences in paragraph 1.7.5 with various substitution matrices and gap penalties.

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	**BLOSUM45**	**open**	**extended**		
SIB's software		8	4	PNGPAP---LGKDL PNGPIRDLLLGKDL	37
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	37
SIB's software		12	2	PNGPAP---LGKDL PNGPIRDLLLGKDL	39
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	39
Sequences	**Sub. matrix**	**Gap penalty**		**Alignment**	**Score**
PSTIAPALISS PNGPIRDLLLGKDL	**BLOSUM50**	**open**	**extended**		
SIB's software		8	4	PNGPAP---LGKDL PNGPIRDLLLGKDL	43
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	43
SIB's software		12	2	PNGPAP---LGKDL PNGPIRDLLLGKDL	45
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	45

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	**BLOSUM62**	**open**	**extended**		
SIB's software		8	4	PNGPAP---LGKDL PNGPIRDLLLGKDL	28

My software			PNGPAP---LGKDL PNGPIRDLLLGKDL	28
SIB's software	12	2	PNGPAP---LGKDL PNGPIRDLLLGKDL	30
My software			PNGPAP---LGKDL PNGPIRDLLLGKDL	30

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	BLOSUM80	open	extended		
SIB's software		8	4	PNGPAP---LGKDL PNGPIRDLLLGKDL	55
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	29
SIB's software		12	2	PNGPAP---LGKDL PNGPIRDLLLGKDL	57
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	31

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	PAM250	open	extended		
SIB's software		8	4	PNGPAP---LGKDL PNGPIRDLLLGKDL	24
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	24
SIB's software		12	2	PNGPAP---LGKDL PNGPIRDLLLGKDL	26
My software				PNGPAP---LGKDL PNGPIRDLLLGKDL	26

1.8.4 Fourth group of comparisons of the new code with SIB's software

In this group of comparisons, we will use sequences in 1.7.71.7.6 with various substitution matrices and various gap penalties.

Sequ.	Sub. matrix	Gap penalty		Alignment	Sc.
Eq. 29	BLOSUM 45	op.	ext.		
SIB's software		8	4	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEALELR MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV---	137
My software				"	137
SIB's software		12	2	"	137
My software				"	137

Sequ.	Sub. matrix	Gap penalty		Alignment	Sc.
Eq. 29	**BLOSUM 50**	**op.**	**ext.**		
SIB's software		4	1	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEALELR MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV---	179
My software				"	179
SIB's software		12	2	"	149
My software				"	149

Sequ.	Sub. matrix	Gap penalty		Alignment	Sc.
Eq. 29	**BLOSUM 62**	**op.**	**ext.**		
SIB's software		4	1	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEALELR MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV---	135
My software				"	135
SIB's software		12	2	"	105
My software				"	105

Sequ.	Sub. matrix	Gap penalty		Alignment	Sc.
Eq. 29	**PAM250**	**op.**	**ext.**		
SIB's software		4	1	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEALELR MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV---	115
My software				"	115
SIB's software		12	2	"	85
My software				"	85

1.8.5 Sixth group of comparison

Now we use another program for global alignment, from the Europen Bioinformatic Institute (↥). The name of this tool is EMBOSS Needle (↥) and it use the Needleman-Wunsch algorithm. It allows for only a limited set of values of gap penalties, but it has a wider range of substitution matrices. We will test again for the alignments in previous paragraphs, comparing our own software with EMBOSS Needle and SIB's software.

Sequences	Sub. matrix	Gap penalty		Alignment	Score
NGPIRDLLLGKD STIAPALISS	**BLOSUM45**	**open**	**extended**		
SIB's software		5	1	NGPIRDLLLGKD S-TIAPALIS-S	-4
My software				NGPIRDLLLGKD S-TIAPALIS-S	-4

EMBOSS Needle			NGPIRDLLLGKD S-TIAPALIS-S	-2
SIB's software	4	1	N--GPIRDLLLGKD STIAP---ALIS-S	-2
My software			NGPIRDLLLGKD S-TIAPALIS-S	-2
EMBOSS Needle	5	1	NGPIRDLLLGKD S-TIAPALIS-S	-2

From the two tests above we can deduce that EMBOSS Needle use a gap model of this type:

$$op\_gap + ex\_gap(x-1)$$

Sequences	**Sub. matrix**	**Gap penalty**		**Alignment**	**Score**
NGPIRDLLLGKD STIAPALISS	**BLOSUM80**	**open**	**extended**		
SIB's software		9	1	N--GPIRDLLLGKD STIAP---ALISS-	-17
My software				NGPIRDLLLGKD S-TIAPALISS-	-20
EMBOSS Needle		10	1	N--GPIRDLLLGKD STIAP----ALISS	-17

From this test we can conclude that my software use a BLOSUM80 matrix with a different scale, while both SIB's software and EMBOSS use the same scale for BLOSUM80. I use the scale ln(2)/2. I will now calculate the scale used by SIB's software and by EMBOSS for BLOSUM80, from the above alignments. The alignment by SIB's software with values from BLOSUM80 with a scale ln(2)/2 gives:

N	-	-	G	P	I	R	D	L	L	L	G	K	D
S	T	I	A	P	-	-	-	A	L	I	S	S	-
0	-10	-1	0	8	-10	-1	-1	-2	4	1	-1	-1	-10

Thus, we calculate:

$$(8-2+4+1-1-1)x - 10 - 1 - 10 - 1 - 1 - 10 = 9x - 33 \Rightarrow$$
$$9x - 33 = -17 \Rightarrow x \simeq 1.778$$

The scale used by this software for BLOSUM80 is then:

$$\frac{1}{\lambda_{80}^{SIBI}} = 1.778\frac{2}{\ln(2)} = 5.12 \Rightarrow \lambda_{80}^{SIBI} \simeq 0.195$$

We will now do the same calculation for EMBOSS Needle:

N	-	-	G	P	I	R	D	L	L	L	G	K	D
S	T	I	A	P	-	-	-	-	A	L	I	S	S
0	-10	-1	0	8	-10	-1	-1	-1	-2	4	-5	-1	-1

Thus, we calculate:

$$(8-2+4-5-1-1)x-10-1-10-1-1-1=3x-24\Rightarrow$$
$$3x-24=-17\Rightarrow x\simeq 2.332$$

The scale used by this software for BLOSUM80 is then:

$$\frac{1}{\lambda_{80}^{EMB}}=2.332\frac{2}{\ln(2)}=6.73\Rightarrow\lambda_{80}^{EMB}\simeq 0.148$$

In conclusion, these three programs do not use the same scale for BLOSUM80 or do not use the same BLOSUM80. In order to better understand what kind of scale and of BLOSUM80 they use, we will perform another test.

Sequences	**Sub. matrix**	**Gap penalty**		**Alignment**	**Score**
PSTIAPALISS PNGPIRDLLLGKDL	**BLOSUM80**	**open**	**extended**		
SIB's software		9	1	PSTIAPA---LISS-- PN--GPIRDLLLGKDL	-6
My software				PS-TIAPALISS-- PNGPIRDLLLGKDL	-13
EMBOSS Needle		10	1	PSTIAP---ALISS-- PN--GPIRDLLLGKDL	-6

We will now verify the alignment with NeW_6, in order to exclude an error in this program:

P	S	-	T	I	A	P	A	L	I	S	S	-	-
P	N	G	P	I	R	D	L	L	L	G	K	D	L
8	0	-10	-2	5	-2	-2	-2	4	1	-1	-1	-10	-1

This score gives correctly -13. Thus, our program use correctly BLOSUM80 martix, this one: (↕). We will now check the alignment by SIB's LALIGN:

P	S	T	I	A	P	A	-	-	-	L	I	S	S	-	
P	N	-	-	G	P	I	R	D	L	L	L	G	K	D	L
8	0	-10	-1	0	8	-2	-10	-1	-1	4	1	-1	-1	-10	-1

This alignment gives a score of -17 if we calculate it with the above BLOSUM80 matrix. This could mean that LALIGN uses this matrix with another scale, or another matrix. In case it was the same matrix, we calculate its scale:

$$(8+8-2+4+1-1-1)x-10-1-10-1-1-10-1=17x-34\Rightarrow$$
$$17x-34=-6\Rightarrow x=1.6470$$

The scale used by this software for BLOSUM80 is then

$$\frac{1}{\lambda_{80}^{SIB}}=1.6470\frac{2}{\ln(2)}=4.75\Rightarrow\lambda_{80}^{SIB}\simeq 0.21$$

This is different from the one used in the previous test, so one possibility is that the BLOSUM80 is different from the one used by me. Otherwise we have to admit that this program change the scale according to the alignment and gap penalties. Another possibility is that the differences are (0.21 vs

0.195) are due to the attempt to obtain integer scores. In this case it could be possible that the scale adopted is ln(2)/3. In fact if I use this scale in NeW_6, it gives back the same alignment as SIB's, LALIGN with a score of -8.5. We will now check the alignment by EMBOSS:

P	S	T	I	A	P	-	-	-	A	L	I	S	S	-	-
P	N	-	-	G	P	I	R	D	L	L	L	G	K	D	L
8	0	-10	-1	0	8	-10	-1	-1	-2	4	1	-1	-1	-10	-1

This alignment gives a score of -17 if we calculate it with the above BLOSUM80 matrix. This could mean that EMBOSS uses this matrix with another scale, or another matrix. In case it was the same matrix, we calculate its scale:

$$(8+8-2+4+1-1-1)x-10-1-10-1-1-10-1=17x-34 \Rightarrow$$
$$17x-34=-6 \Rightarrow x=1.6470$$

The scale used by this software for BLOSUM80 is then

$$\frac{1}{\lambda_{80}^{EMB}}=1.6470\frac{2}{\ln(2)}=4.75 \Rightarrow \lambda_{80}^{EMB} \simeq 0.21$$

Also in this case I tend to think that EMBOSS uses a BLOSUM80 matrix scaled with a scale ln(2)/3. In order to prove this hypothesis I will now run a test where NeW_6 uses a BLOSUM80 matrix scale with a scale ln(2)/3, and I will compare the result with those given by LALIGN and EMBOSS.

Sequences	**Sub. matrix**	**Gap penalty**		**Alignment**	**Score**
NGPIRDLLLGKD STIAPALISS	**BLOSUM80**	**open**	**extended**		
SIB's software		9	1	N--GPIRDLLLGKD STIAP---ALISS-	-17
My software				N--GPIRDLLLGKD STIAP----ALISS	-19.5
				N--GPIRDLLLGKD STIAP---ALISS-	
EMBOSS Needle		10	1	N--GPIRDLLLGKD STIAP----ALISS	-17

Sequences	**Sub. matrix**	**Gap penalty**		**Alignment**	**Score**
PSTIAPALISS PNGPIRDLLLGKDL	**BLOSUM80**	**open**	**extended**		
SIB's software		9	1	PSTIAPA---LISS-- PN--GPIRDLLLGKDL	-6
My software				PSTIAPA---LISS-- PN--GPIRDLLLGKDL	-8.5
EMBOSS Needle		10	1	PSTIAP---ALISS-- PN--GPIRDLLLGKDL	-6

So, I confirm my conclusion that both EMBOSS and LALIGN use a BLOSUM80 matrix with a scale of ln(2)/3, which can be approximate by the one used by NeW_6, multiplied by 1.5.

1.8.6 Seventh group of comparison

We will now test NeW_6 and EMBOSS Needle for BLOSUM90. LALIGN doesn't have this substitution matrix.

Sequences	Sub. matrix	Gap penalty		Alignment	Score
NGPIRDLLLGKD STIAPALISS	BLOSUM90	open	extended		
My software		9	1	NGPIRDLLLGKD S-TIAPALISS-	-20
EMBOSS Needle		10	1	N--GPIRDLLLGKD STIAP----ALISS	-20

Sequences	Sub. matrix	Gap penalty		Alignment	Score
PSTIAPALISS PNGPIRDLLLGKDL	BLOSUM90	open	extended		
My software		15	5	PSTIAPALI---SS PNGPIRDLLLGKDL PSTIAPALISS--- PNGPIRDLLLGKDL	-35
EMBOSS Needle		20	5	PSTIAPALI---SS PNGPIRDLLLGKDL	-35

Sequ.	Sub. matrix	Gap penalty		Alignment	Sc.
Eq. 29	BLOSUM 90	op.	ext.		
My software		15	5	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEALELR MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV---	94
EMBOSS Needle		20	5	MS--ILKIHAREIFDSRGNPTVEVDLFTSKGLF-RAAVPSGASTGIYEAL MGFHIYEIKARQIIDSRGNPTVEADVILEDGTYGRAAVPSGASTGINEAV	94

1.9 Synopsis for main features of three programs for global alignments

In the following table, we summarize the main features of the following three programs for global alignments: NeW_6 (which is the one presented in paragraph 1.7), LALIGN by *Swiss Institute of Bioinformatics* (↑), and EMBOSS Needle, by the Europen Bioinformatic Institute (↑).

	NeW_6	LALIGN	EMBOSS Needle
Algorithm	Needleman-Wunsch		
Gap model	$op\_gap + ex\_gap(x-1)$		$op\_gap + ex\_gap(x)$
End gap penalty	As above. Can't be excluded.	Can be included or excluded. Values can be selected independently from those of the gaps.	Can be included or excluded.

BLOSUM45	$\lambda = \frac{\ln(2)}{3} \sim 0.231$ (↑)		
BLOSUM50	$\lambda = \frac{\ln(2)}{3} \sim 0.231$ (↑)	To verify	$\lambda = \frac{\ln(2)}{3} \sim 0.231$ (↑)
BLOSUM62	$\lambda = \frac{\ln(2)}{2} \sim 0.346$ (↑)	To verify	$\lambda = \frac{\ln(2)}{2} \sim 0.346$ (↑)
BLOSUM80	$\lambda = \frac{\ln(2)}{2} \sim 0.346$ (↑)	Probably use a scale: $\lambda = \frac{\ln(2)}{3}$	Probably use a scale: $\lambda = \frac{\ln(2)}{3}$
BLOSUM90	$\lambda = \frac{\ln(2)}{2} \sim 0.346$ (↑)	Not implemented	$\lambda = \frac{\ln(2)}{2} \sim 0.346$ (↑)
PAM250	$\lambda = \frac{\ln(2)}{3} \sim 0.231$(↑)	To verify	$\lambda = \frac{\ln(2)}{3} \sim 0.231$(↑)

2 Referencie

Agrawal, A. 2009. Sequence-specific sequence comparison using pairwise statistical significance. *PhD dissertation.* s.l. : Iowa State University, 2009.

Agrawal, A, Brendel, VP and Huang, X. 2008. Pairwise statistical significance and empirical determination of effective gap opening penalties for protein local sequence alignment. *Int J Comput Biol Drug Des.* 2008, Vol. 1, 4, pp. 347-67.

Altschul, SF. 1991. Amino acid substitution matrices from an information theoretic perspective. June 5, 1991, Vol. 219, 3, pp. 555-65.

Altschul, SF and Gish, G. 1996. Local alignment statistics. *Methods Enzymol.* 1996, Vol. 266, pp. 460-480.

Altschul, SF, et al. 2001. The estimation of statistical parameters for local alignment score distributions. *Nucleic Acids Res.* Jan 15, 2001, Vol. 29, 2, pp. 351-61.

Andersen, HP, Nielsen, M and Lund, O. 2006. Prediction of residues in discontinuous B-cell epitopes using protein 3D structures. *Protein Sci.* Nov 2006, Vol. 15, 11.

Argos, P. 1987. A sensitive procedure to compare amino acid sequences. *J Mol Biol.* Jan 20, 1987, Vol. 193, 2, pp. 385-96.

Bairoch, A. 1991. PROSITE: a dictionary of sites and patterns in proteins. *Nucleic acids research.* April 25, 1991, pp. 2241–2245.

Bairoch, A, Bucher, P and Hofmann, K. 1997 . The PROSITE database, its status in 1997. *Nucleic acids research.* January 1, 1997 , Vol. 25, 1, pp. 217-21.

Bastide, L, et al. 2002. Interferon and the 2-5A/Pathway. [book auth.] P Englebienne and K De Meirleir. *Chronic fatigue syndrome, a biological approach.* Boca Raton : CRC press, 2002, 1, pp. 1-3.

Berzofsky, JA. 1985. Intrinsic and extrinsic factors in protein antigenic structure. *Science.* Sep 6, 1985, Vol. 229, 4717, pp. 932-40.

Beyond Myalgic Encephalomyelitis/Chronic Fatigue Syndrome: Redefining an Illness. **IOM. 2015.** Washington (DC) : National Academies Press (US), 2015.

Bordoni, PG. 1999. *Lezioni di Mecanica Razionale.* s.l. : Zanichelli, 1999.

Brenu, EW, et al. 2011. Immunological abnormalities as potential biomarkers in Chronic Fatigue Syndrome/Myalgic Encephalomyelitis. *J Transl Med.* May 2011, Vol. 9, 81.

Caligiuri, M, et al. 1987. Phenotypic and functional deficiency of natural killer cells in patients with chronic fatigue syndrome. *J Immunol.* Nov 15, 1987, Vol. 139, 10, pp. 3306-13.

Callister, SM, et al. 2016 May. Detection of IFN-γ Secretion by T Cells Collected Before and After Successful Treatment of Early Lyme Disease. *Clin Infect Dis.* May 15, 2016 May, Vol. 62, 10, pp. 1235-41.

Centers for disease control and prevention. 2015. Laboratory tests that are not recommended. *www.cdc.gov.* [Online] Nov 2015. [Cited:]

Chapman, SJ. 1998. *Introduction to Fortran 90/95.* s.l. : The McGraw-Hill Companies, 1998.

Chen, J, et al. 1999. Association of antibiotic treatment-resistant Lyme arthritis with T cell responses to dominant epitopes of outer surface protein A of Borrelia burgdorferi. *Arthritis Rheum.* Sep 1999, Vol. 42, 9, pp. 1813-22.

Dayhoff, MO, Eck, RV and Park, CM. 1972. *Atlas of Protein Sequence and Structure.* Washington D.C. : Nat. Biomed. Res. Found., 1972.

Dayhoff, MO, Schwartz, RM and Orcutt, BC. 1978. A model of evolutionary change in proteins. [book auth.] MO Dayhoff. *Atlas of protein sequence and structure.* Silver Spring : National Biomedical Research Foundtion, 1978, Vol. 5, pp. 345-352.

De Meirleir, K, et al. 2000. A 37 kDa 2-5A binding protein as a potential biochemical marker for chronic fatigue syndrome. *Am J Med.* Feb 99-105, 2000, Vol. 108, 2.

Demettre, E, et al. 2002. Ribonuclease L proteolysis in peripheral blood mononuclear cells of chronic fatigue syndrome patients. *J Biol Chem.* Sep 20, 2002, Vol. 277, 38, pp. 35746-51.

Dressler, F, Yoshinari, NH and Steere, AC. 1991. The T-cell proliferative assay in the diagnosis of Lyme disease. *Ann Intern Med.* 1991, Vol. 115, 7.
Echave, J, Spielman, SJ and Wilke, CO. 2016. Causes of evolutionary rate variation among protein sites. *Nat Rev Genet.* Feb 2016, Vol. 17, 2, pp. 109-21.
Ewens, W and Grant, G. 2005. Appendix B. Mathematical Formulae and Results. *Statistical Methods in Bioinformatics: An Intoduction.* s.l. : Springer, 2005.
—. 2005. Phylogeneric tree estimation. *Statistical Methods in Bioinformatics: An Introduction.* 2nd edition. New York : Springer, 2005, 15.
—. 2005. Stochastic Processes: Markov Chains. *Statistical Methods in Bioinformatics: An Introduction.* Second Edition. s.l. : Springer, 2005, 11.
—. 2005. The analysis of multiple DNA or Protein Sequences. *Statistical Methods in Bioinformatics: An Introduction.* Second Edition. s.l. : Springer, 2005, Vol. 6.
Fontain, PA, et al. 2000. Antibodies to Streptococcal Surface Enolase React with Human a-Enolase: Implications in Poststreptococcal Sequelae. *Journal of Infectious Diseases.* 2000, Vol. 182, 6, pp. 1712-1721.
Gaudino, EA, Coyle, PK and Krupp, LB. 1997. Post-Lyme syndrome and chronic fatigue syndrome. Neuropsychiatric similarities and differences. *Arch Neurol.* Nov, 1997, Vol. 54, 11, pp. 1372-6.
George, DG, Barker, WC and Hunt, LT. 1990. Mutation data matrix and its uses. *Methods in enzymology.* 1990, Vol. 183, pp. 333-51.
Ghizzetti, A and Rosati, F. 1996. *Analisi Matematica 1.* s.l. : Zanichelli, 1996.
Gibbs, AJ and McIntyre, GA. 1970. The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. *European journal of biochemistry / FEBS.* September 1970, Vol. 16, 1, pp. 1-11.
Giloteaux, L, et al. 2016. Reduced diversity and altered composition of the gut microbiome in individuals with myalgic encephalomyelitis/chronic fatigue syndrome. *Microbiome.* Jun 23, 2016, Vol. 4, 1.
Gori, L. 1999. Formule di Newton-Cotes. *Calcolo numerico.* IV. Roma : Edizioni Kappa, 1999, 7.
—. 1999. Nozioni introduttive. *Calcolo numerico.* s.l. : Edizioni Kappa, 1999, 1.
Han, Y, et al. 2014. Structure of human RNase L reveals the basis for regulated RNA decay in the IFN response. *Science.* Mar 14, 2014, Vol. 343, 6176, pp. 1244-8.
Henikoff, S and Henikoff, JG. 1992. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America.* 1992, Vol. 89, pp. 10915-10919.
—. 1991. Automated assembly of protein blocks for database searching. *Nucleic acids research.* December 11, 1991, Vol. 19, 23, pp. 6565-72.
Kaech, SM and Wherry, EJ. 2007. Heterogeneity and cell-fate decisions in effector and memory CD8+ T cell differentiantion during viral infection. *Immunity.* September 2007, Vol. 27, 3, pp. 393-405.
Karlin, S and Altschul, SF. 1990. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences of the United States of America.* March 1990, Vol. 87, 6, pp. 2264-8.
Kringelum, JV, et al. 2013. Structural analysis of B-cell epitopes in antibody:protein complexes. *Mol Immunol.* Jan 2013, Vol. 53, 1-2, pp. 24-34.
Kschischo, M, Lässig, M and Yu, YK. 2005. Toward an accurate statistics of gapped alignments. *Bull Math Biol.* Jan 2005, Vol. 67, 1, pp. 169-91.
Kumaran, D, et al. 2001. Crystal structure of outer surface protein C (OspC) from the Lyme disease spirochete, Borrelia burgdorferi. *EMBO J.* . Mar 2001, Vol. 20, 5, pp. 971-8.
Law, RH, et al. 2010. The structural basis for membrane binding and pore formation by lymphocyte perforin. *Nature. 2010 Nov 18;468(7322):.* Nov 18, 2010, Vol. 468, 7322, pp. 447-51.

Li, H, et al. 2014. Autoimmune basis for postural tachycardia syndrome. *J Am Heart Assoc.* Feb 26, 2014, Vol. 3, 1.
Maes, M, Mihaylova, I and Leunis, JC. 2007. Increased serum IgA and IgM against LPS of enterobacteria in chronic fatigue syndrome (CFS): indication for the involvement of gram-negative enterobacteria in the etiology of CFS and for the presence of an increased gut-intestinal permeability. *J Affect Disord.* . Apr 2007, Vol. 99, 1-3, pp. 237-40.
Maher, KJ, et al. 2002. Quantitative fluorescence measures for determination of intracellular perforin content. *Clin Diagn Lab Immunol.* Nov 2002, Vol. 9, 6, pp. 1248-52.
Maher, KJ, Klimas, NG and Fletcher, MA. 2005. Chronic fatigue syndrome is associated with diminished intracellular perforin. *Clin Exp Immunol.* . Dec 2005, Vol. 142, 3, pp. 505-11.
McLachan, AD. 1971. Tests for comparing related amino-acid sequences. Cytochrome c and cytochrome c 551. *J Mol Biol.* Oct 28, 1971, Vol. 61, 2, pp. 409-24.
Morris, GE. 2007. Encyclopedia of life sciences. *www.els.net.* [Online] September 2007. [Cited: Aug 1, 2016.] http://www.els.net/WileyCDA/ElsArticle/refId-a0002624.html.
Mott, R. 2000. Accurate formula for P-values of gapped local sequence and profile alignments. *J Mol Biol.* Jul 14, 2000, Vol. 300, 3, pp. 649-59.
Mount, D. 2001. Alignment of pairs of sequencies. *Bioninformatics. Sequence and genome analysis.* s.l. : Cold Spring Harbour Laboratory Press , 2001, 3.
—. 2001. Database search for similar sequences. *Bioinformatics. Sequence and genome analysis.* s.l. : Cold Spring Harbour Laboratory Press, 2001, 7.
—. 2001. Glossary. *Bioninformatics. Sequence and Genome Analysis.* s.l. : Cold Spring Harbor Laboratory Press, 2001.
Needleman, SB and Wunsch, CD. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology.* March 1970, Vol. 48, 3, pp. 443-53.
Ofran, Y, Schlessinger, A and Rost, B. 2008. Automated identification of complementarity determining regions (CDRs) reveals peculiar characteristics of CDRs and B cell epitopes. *J Immunol.* . Nov 1, 2008, Vol. 181, 9, pp. 6230-5.
Openshaw, PJ. 2009. Crossing barriers: infections of the lung and the gut. *Mucosal Immunol.* . Mar 2009, Vol. 2, 2, pp. 100-2.
Parker, JM, Guo, D and Hodges, RS. 1986. New hydrophilicity scale derived from high-performance liquid chromatography peptide retention data: correlation of predicted surface residues with antigenicity and X-ray-derived accessible sites. *Biochemistry.* Sep 23, 1986, Vol. 25, 19, pp. 5425-32.
Peterson, P, Perheentupa, J and Krohn, KJ. 1996. Detection of candidal antigens in autoimmune polyglandular syndrome type I. *Clin Diagn Lab Immunol.* May 1996, Vol. 3, 3, pp. 290-294.
Piast, M, et al. 2005. Molecular evolution of enolase. 2005, Vol. 52 . Epub 2005 May 15., 2, pp. 507-13.
Ponomarenko, J, et al. 2008. ElliPro: a new structure-based tool for the prediction of antibody epitopes. *BMC Bioinformatics.* Dec 2, 2008, Vol. 9, 514.
Reese, JT and Pearson, WR. 2002. Empirical determination of effective gap penalties for sequence comparison. *Bioinformatics.* Nov 2002, Vol. 18, 11, pp. 1500-7.
Richardson, JS. 2000-2007. *The Anatomy and Taxonomy of Protein Structure.* Durham : Duke University, 2000-2007.
Robbins, H. 1955. A Remark on Stiling's Formula. *The American Mathematical Monthly.* January 1955, Vol. 62, 1, pp. 26-29.
Root-Bernstein, R. 2014. Rethinking molecular mimicry in rheumatic heart disease and autoimmune myocarditis: laminin, collagen IV, CAR, and B1AR as initial target of disease. *Frontiers in Pediatrics.* August 2014, Vol. 2, 85.
Rose, NR. 1998. The role of infection in the pathogenesis of autoimmune disease. *Seminars in immunology.* February 1998, Vol. 10, 1, pp. 5-13.

Rubinstein, ND, et al. 2008. Computational characterization of B-cell epitopes. *Mol Immunol.* . Jul 2008, Vol. 45, 12, pp. 3477-89.
Rudensky, AYu, et al. 1991. Sequence analysis of peptides bound to MHC class II molecules. *Nature.* Oct 17, 1991, Vol. 353, 6345, pp. 622-7.
Sievers, F, et al. 2011. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular systems biology.* October 11, 2011, Vol. 7, 539.
Singh, H, Ansari, HR and Raghava, GP. 2013. Improved Method for Linear B-Cell Epitope Prediction Using Antigen's Primary Sequence. *PLoS One.* May 7, 2013, Vol. 8, 5.
Smith, HO, Annau, TM and Chandrasegaran, S. 1990. Finding sequence motifs in groups of functionally related proteins. *Proceedings of the National Academy of Sciences of the United States of America.* January 1990, Vol. 87, 2, pp. 826-30.
Smith, TF and Waterman, MS. 1981. Identification of common molecular subsequences. *Journal of molecular biology.* March 25, 1981, Vol. 147, 1, pp. 195-7.
Smith, TF, Waterman, MS and Fitch, WM. 1981. Comparative biosequence metrics. *Journal of molecular evolution.* 1981, Vol. 18, 1, pp. 38-46.
Sompayrac, Lauren M. 2012. *How the Immune System Works.* Fourth Edition. s.l. : Wiley-Blackwell, 2012.
Stanek, G, et al. 2011. Lyme borreliosis: clinical case definitions for diagnosis and management in Europe. *Clin Microbiol Infect.* Jan 2011, Vol. 17, 1, pp. 69-79.
Stern, LJ and Wiley, DC. 1994. Antigenic peptide binding by class I and class II histocompatibility proteins. *Structure.* April 1994, Vol. 2, pp. 245-251.
Stormo, GD and Hartzell, GW. 1989. Identifying protein-binding sites from unaligned DNA fragments. *Proc Natl Acad Sci U S A.* Feb 1989, Vol. 86, 4, pp. 1183-7.
Suhadolnik, RJ, et al. 1997. Biochemical evidence for a novel low molecular weight 2-5A-dependent RNase L in chronic fatigue syndrome. *J Interferon Cytokine Res.* . Jul 1997, Vol. 17, 7, pp. 377-85.
Suhadolnik, RJ, et al. 1994. Changes in the 2-5A synthetase/RNase L antiviral pathway in a controlled clinical trial with poly(I)-poly(C12U) in chronic fatigue syndrome. *In Vivo.* . Jul-Aug 1994, Vol. 8, 4, pp. 599-604.
Sundstrom, P and Aliaga, GR. 1992. Molecular cloning of cDNA and analysis of protein secondary structure of Candida albicans enolase, an abundant, immunodominant glycolytic enzyme. *Journal of Bacteriology.* November 1992, Vol. 174 , 21, pp. 6789–6799.
Thornton, JM, et al. 1986. Location of 'continuous' antigenic determinants in the protruding regions of proteins. *EMBO J.* . Feb 1986, Vol. 5, 2, pp. 409-13.
Tracy, MR and Hedges, SB. 2000 . Evolutionary history of the enolase gene family. *Gene.* December 23, 2000 , Vol. 259, 1-2, pp. 129-138.
Valentine-Thon, E, Ilsemann, K and Sandkamp, M. 2007. A novel lymphocyte transformation test (LTT-MELISA) for Lyme borreliosis. *Diagn Microbiol Infect Dis.* 2007, Vol. 57, 1.
Van Regenmortel, MHV. 1996. Mapping Epitope Structure and Activity: From One-Dimensional Prediction to Four-Dimensional Description of Antigenic Specificity. *Methods.* Jun 1996, Vol. 9, 3, pp. 465-72.
von Baehr, V, et al. 2012. The lymphocyte transformation test for borrelia detects active lyme borreliosis and verifies effective antibiotic treatment. *Open Neurol J.* 2012, Vol. 6.
Wyatt, J, et al. 1993. Intestinal permeability and the prediction of relapse in Crohn's disease. *Lancet.* Jun 5, 1993, Vol. 341, 8858, pp. 1437-9.
Yu, X, et al. 2012. Autoantibody activation of beta-adrenergic and muscarinic receptors contributes to an "autoimmune" orthostatic hypotension. *Journal of the American Society of Hypertension.* Jan-Feb 2012, Vol. 6, 1, pp. 40-7.
Zhu, L, et al. 2013. Characterization of gut microbiomes in nonalcoholic steatohepatitis (NASH) patients: a connection between endogenous alcohol and NASH. *Hepatology.* Feb 2013, Vol. 57, 2, pp. 601-9.