FACULTE DES SCIENCES ET TECHNIQUES
COLLEGE SANTE ET COLLEGE SCIENCES ET TECHNIQUES
MASTER 1 BIOINFORMATIQUE

université de BORDEAUX

Rapport : Projet Align2SeqJS

ANNE Rudy
BELIARD Aurelien
DUQUENNE Emeline
PERDRIAU Aurore

Client :
Jean-Christophe Taveau

Code de l'application :
https://github.com/crazybiocomputing/align2seq.git

10 mai 2015

Table des matières

Introduction

Les algorithmes d'alignement par paire (ou deux à deux) sont des algorithmes permettant de trouver le meilleur alignement possible entre deux séquences, qu'elles soient nucléiques ou protéiques.

Les régions d'intérêt communes repérées signifient qu'il existe des relations fonctionnelles ou structurales entre les deux séquences. Elles peuvent également démontrer une relation évolutive, c'est-à-dire qu'une des deux séquences est antérieure à la seconde.

Le résultat final de l'alignement ressort sous la forme de deux lignes, chacune représentant une séquence où les éléments sont alignés par paire. Pour que le résultat obtenu soit optimal, des pénalités de gaps sont insérées : elles permettent de signaler des insertions ou des délétions possibles dans l'évolution. Suivant le même principe, lorsque les éléments des séquences à une même position sont différents, des mismatchs indiquent des mutations ponctuelles.

La méthode des alignements par paire est plus utilisée que celle des alignements multiples dans le cas de calcul rapide ne nécessitant pas une précision optimale des résultats (exemple : recherche de séquences à très forte similarité par rapport à une séquence donnée).

Notre projet, nommé Align2Seq, va ainsi avoir comme but d'apprendre le fonctionnement de deux de ces algorithmes (Needleman et Wunsch ainsi que Smith et Waterman) aux étudiants de troisième année de Licence en Biologie de Bordeaux, et ce de manière interactive et pédagogique.

CHAPITRE I

Analyse

I.1 Contexte

Le projet Align2Seq est à visée pédagogique. Il s'agira d'une application web (utilisation des technologies HTML [1] /CSS [2] /JavaScript [3]) utilisée par les futurs étudiants de dernière année de licence Biologie/Santé afin de leur permettre d'appendre et de comprendre le fonctionnement des algorithmes d'alignement par paire. L'application devra donc être facile d'utilisation, rapide et claire.

Les algorithmes utilisés peuvent être basés sur différentes méthodes de programmation : pour le projet Align2Seq, la programmation dite dynamique est retenue. Pour cette méthode de programmation, chacune des positions de séquence est analysée et il lui est attribué un score ou un gap suivant la circonstance. Pour les séquences protéiques, les matrices de substitution sont utilisées tandis que pour les séquences nucléiques, ce sont des matrices de score.

La somme calculée à partir de ces matrices est soit positive, soit négative ou alors bénéficie d'une pénalité de gap variant selon sa position dans la séquence. Deux algorithmes seront utilisés au sein de l'application Align2Seq : Needleman et Wunsch pour l'alignement de type global, et Smith et Waterman pour l'alignement de type local.

La complexité des algorithmes effectués en programmation dynamique est généralement de l'ordre de O(nm) avec n et m la longueur de chacune des séquences. Cela leur confère une rapidité d'exécution suffisamment élevée pour qu'ils soient utilisés par plusieurs utilisateurs au même instant. Pour l'application Align2Seq, il sera fait en sorte que la complexité de ces algorithmes soit similaire à celle-ci.

I.1.1 Algorithme de Needleman et Wunsch

L'algorithme de Needleman et Wunsch a été créé par Saul B. Needleman et Christian D. Wunsch en 1970 [4]. Cet algorithme de type global va produire un alignement des deux séquences dans leur intégralité. Il fonctionne en plusieurs étapes.

1. La première séquence est placée en première ligne d'un tableau, et la deuxième séquence en première colonne. On laisse une case vide au début de chaque séquence.
2. Choix des scores pour chacun des cas possibles : le match, le mismatch et les indels (lorsque la lettre est face à un gap).
3. Remplir le tableau grâce à ces scores : l'intersection des cases vides laissées auparavant correspond à un zéro.
4. Remplir chaque case comme étant le score maximal, c'est-à-dire la somme maximale entre le score de la case et l'un des scores autour de cette case (soit la case au-dessus, à gauche ou au-dessus à gauche). La case comportant ce score environnant est retenue.
5. Une fois tous les scores remplis, l'algorithme doit indiquer le chemin de l'alignement idéal entre les deux séquences. Il s'agit de partir de la case en bas à droite et de remonter jusqu'au 0 de la case de départ (en haut à gauche), tout ceci en passant par les scores retenus précédemment (en choisissant à chaque case le score le plus élevé possible). Si plusieurs choix égaux se présentent (si plusieurs des cases avaient le même score), on choisit de remonter vers l'une d'entre elles de manière définie à l'avance (la diagonale, ou la verticale si la première n'est pas accessible).

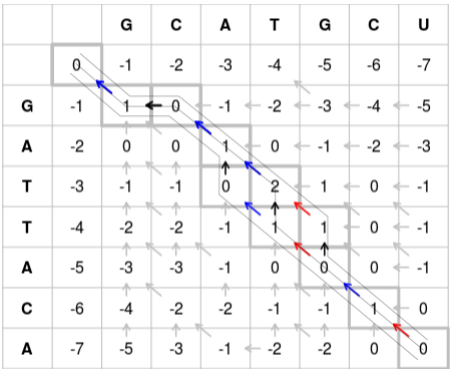


FIGURE I.1 – Exemple de matrice obtenue pour les séquences protéiques GCATGCU et GATTACA, avec alignements globaux dessinés.

De cette façon, plusieurs choix différents d'alignements globaux entre deux séquences sont possibles.

I.1.2 Algorithme de Smith et Waterman

L'algorithme de Smith et Waterman a été créé par Temple F. Smith et Michael S. Waterman en 1981 [5]. Contrairement à l'algorithme précédent, celui-ci va nous permettre d'obtenir des alignements locaux. Son fonctionnement est similaire à l'algorithme de Needleman et Wunsch : les seules différences étant qu'une fois tous les scores remplis, le chemin de l'alignement idéal s'arrête dès que le score atteint 0, et que le début du chemin ne débute pas forcément à la case en bas à droite.

1. Calcul de la matrice des scores d'alignement avec remise à 0 des scores négatifs
2. Construction de l'alignement à partir de la matrice des scores

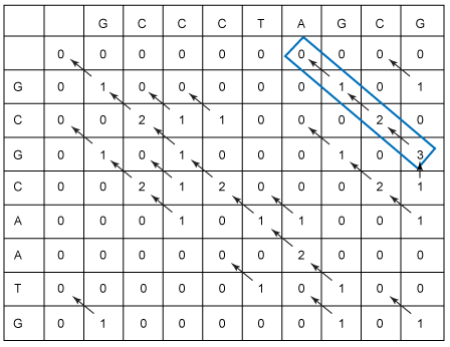


FIGURE I.2 – Exemple de matrice obtenue pour les séquences protéiques GCCCTAGCG et GCGCAATG, avec alignement local indiqué.

I.1.3 Matrices de substitution

Pour comparer deux séquences protéiques, on utilise différentes matrices de substitution, qui recensent les 20 acides aminés possibles. Les scores vont différer selon la similarité des deux acides aminés comparés. Deux types de matrices sont principalement utilisées :

- Les matrices PAM [6] (ou de Dayhoff, figure 3), signifiant "probability of acceptable mutations", sont basées sur les distances évolutives entre espèces. Elles sont suivies d'un numéro représentant les taux de mutation attendus entre acides aminés.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	2	-2	0	0	-2	0	0	1	-1	-1	-2	-1	-1	-3	1	1	1	-6	-3	0	0	0	0	-8
R	-2	6	0	-1	-4	1	-1	-3	2	-2	-3	3	0	-4	0	0	-1	2	-4	-2	-1	0	-1	-8
N	0	0	2	2	-4	1	1	0	2	-2	-3	1	-2	-3	0	1	0	-4	-2	-2	2	1	0	-8
D	0	-1	2	4	-5	2	3	1	1	-2	-4	0	-3	-6	-1	0	0	-7	-4	-2	3	3	-1	-8
C	-2	-4	-4	-5	12	-5	-5	-3	-3	-2	-6	-5	-5	-4	-3	0	-2	-8	0	-2	-4	-5	-3	-8
Q	0	1	1	2	-5	4	2	-1	3	-2	-2	1	-1	-5	0	-1	-1	-5	-4	-2	1	3	-1	-8
E	0	-1	1	3	-5	2	4	0	1	-2	-3	0	-2	-5	-1	0	0	-7	-4	-2	3	3	-1	-8
G	1	-3	0	1	-3	-1	0	5	-2	-3	-4	-2	-3	-5	0	1	0	-7	-5	-1	0	0	-1	-8
H	-1	2	2	1	-3	3	1	-2	6	-2	-2	0	-2	-2	0	-1	-1	-3	0	-2	1	2	-1	-8
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5	2	-2	2	1	-2	-1	0	-5	-1	4	-2	-2	-1	-8
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6	-3	4	2	-3	-3	-2	-2	-1	2	-3	-3	-1	-8
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5	0	-5	-1	0	0	-3	-4	-2	1	0	-1	-8
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6	0	-2	-2	-1	-4	-2	2	-2	-2	-1	-8
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9	-5	-3	-3	0	7	-1	-4	-5	-2	-8
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6	1	0	-6	-5	-1	-1	0	-1	-8
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2	1	-2	-3	-1	0	0	0	-8
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3	-5	-3	0	0	-1	0	-8
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17	0	-6	-5	-6	-4	-8
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	-2	-3	-4	-2	-8
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4	-2	-2	-1	-8
B	0	-1	2	3	-4	1	3	0	1	-2	-3	1	-2	-4	-1	0	0	-5	-3	-2	3	2	-1	-8
Z	0	0	1	3	-5	3	3	0	2	-2	-3	0	-2	-5	0	0	-1	-6	-4	-2	2	3	-1	-8
X	0	-1	0	-1	-3	-1	-1	-1	-1	-1	-1	-1	-1	-2	-1	0	0	-4	-2	-1	-1	-1	-1	-8
*	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	1

FIGURE I.3 – Matrice PAM250

- Les matrices BLOSUM [7] (ou de Henikoff, figure 4), signifiant "blocks of amino acid substitution matrix", sont basées sur l'identité entre différents blocs courts de séquences protéiques. Elles sont suivies d'un numéro représentant les taux d'identité entre ces blocs.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-2	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

FIGURE I.4 – Matrice BLOSUM62

Pour comparer deux séquences nucléotidiques, une simple utilisation de trois scores pourrait suffire, avec un quatrième score dans le cas où le nucléotide est proche de l'autre (les couples A/G et C/T). Cependant, pour des raisons de cohérence avec la comparaison protéique, nous avons utilisé trois matrices appelées EDNA disponibles sur le site d'EMBOSS [8]. Créées par Todd Lowe en 1992 , ces matrices permettent une meilleure analyse des différences entre séquences, sachant qu'elles intègrent, en plus des 4 nucléotides habituels (A,C,G,T), toutes les ambiguïtés possibles (par exemple M pour signaler un A ou un C).

I.2 Etat de l'existant

Le client souhaite que les algorithmes de Needleman et Wunch (alignement global) et de Smith et Waterman (alignement local) soient implémentés dans l'application. Ces deux algorithmes d'alignement par paire, très couramment utilisés dans le domaine de la biologie, ont déjà été développés dans la plupart des langages.

Cependant, la plupart des applications créées n'ont aucun but pédagogique et ne produisent que des résultats bruts sans visualisation du fonctionnement de l'algorithme. Certaines ont tout de même un mode de visualisation : elles seront donc un support d'observation important pour la réalisation de l'application Align2Seq.

I.2.1 Une application Java : Ba-Ba

L'application existante la plus performante trouvée à ce jour s'appelle Ba-Ba [9] (Basic-Algorithms-of-Bioinformatics Applet). Elle possède un mode pas à pas et permet de modifier de nombreux paramètres (pénalités de gap, de mismatch, choix des matrices). Le plus court chemin est visualisé par des flèches qui apparaissent également au fur et à mesure que l'utilisateur fait défiler les résultats. Cependant, Ba-Ba est réalisé en Java, ce qui entraîne des problèmes de compatibilité avec certains navigateurs web ainsi que des erreurs lors du chargement ou de l'affichage de la page.

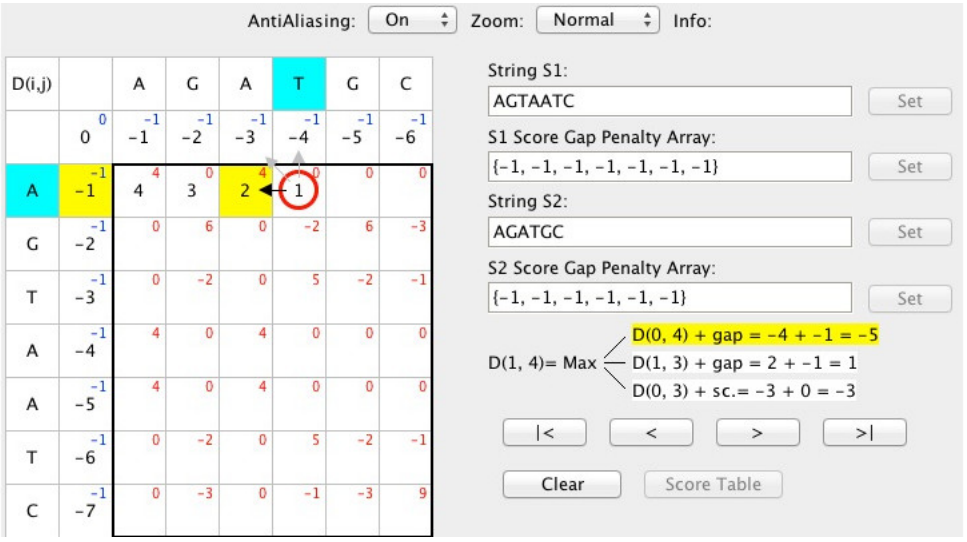


FIGURE I.5 – Application Ba-Ba en cours d'exécution

I.2.2 Une application JavaScript : Ds9a

Une autre application : Ds9a [10], développée cette fois-ci en JavaScript, permet de visualiser la construction de la matrice de score qui se construit dynamiquement au fur et à mesure que l'utilisateur renseigne les deux séquences à aligner. Les différentes pénalités sont également modulables et le plus court chemin à emprunter apparaît sous la forme d'une coloration rouge (le choix du chemin n'est pas visualisable point par point). Cette application n'est malheureusement pas complète : en effet, elle ne permet pas l'utilisation de matrices de substitution (BLOSUM et PAM), lesquelles permettent de réaliser un alignement de bien meilleure qualité sur des séquences protéiques.

I.3 Analyse des besoins

On distingue deux types de besoins : les besoins fonctionnels et les besoins non fonctionnels. Les besoins fonctionnels sont les fonctionnalités proposées à l'utilisateur permettant d'enrichir son utilisation. Les besoins non fonctionnels sont les spécifications techniques demandées au programme.

I.3.1 Analyse des besoins fonctionnels

Les besoins fonctionnels peuvent être séparés en deux sous-catégories : les besoins essentiels sont les fonctionnalités indispensables au fonctionnement de l'application, tandis que les besoins non essentiels sont des fonctionnalités envisageables qui pourraient améliorer l'application mais ne sont pas indispensables à son fonctionnement.

I.3.1.1 Analyse des besoins essentiels

Align2Seq doit permettre d'aligner deux séquences en utilisant différents algorithmes. Celui de Needleman et Wunsch ainsi que celui de Smith et Waterman seront implémentés. Sachant que de nombreux autres existent, il faudra donc qu'ils soient implémentables sans modifier tout le reste de la structure.

L'application devra bénéficier d'un mode d'affichage pas à pas afin d'expliquer le fonctionnement des algorithmes à l'utilisateur. Son interface devra être claire et propre pour faciliter la compréhension de l'algorithme utilisé.

L'utilisateur devra avoir la possibilité de comparer les séquences nucléiques ou protéiques de son choix, bien que leur longueur devra être limitée. Il sera également en mesure de modifier les pénalités de gap à appliquer lors du calcul de l'alignement.

I.3.1.2 Analyse des besoins non essentiels

La plus grande partie des matrices existantes devra être introduite dans l'application pour permettre de diversifier les exemples pédagogiques.

Des fonctions load et save seraient intéressantes pour que l'utilisateur puisse charger un fichier de séquences et enregistrer les résultats obtenus après l'alignement de celles-ci.

Les séquences peuvent être rédigées en différents formats (texte, fasta ...). Il faudrait que les principaux formats puissent être traités.

Les sources de l'application pourraient être mises en ligne afin que toute personne le souhaitant puisse étudier l'implémentation de l'algorithme pour celle-ci. De nouveaux algorithmes pourraient ainsi être ajoutés par la suite.

I.3.2 Analyse des besoins non fonctionnels

L'un des objectifs de notre projet est de respecter les bonnes pratiques de la programmation. Pour cela, un gestionnaire de version de type Git sera utilisé pour permettre une programmation collaborative.

Les langages HTML5 et CSS3 ont été choisis pour mettre en forme Align2Seq en respectant les standards du W3C [11] en matière de programmation web.

Le JavaScript a également été sélectionné pour implémenter les algorithmes car il est bien adapté aux technologies du Web et il se marie bien au HTML5 et au CSS3.

L'explication des autres choix de langage sera faite à la section suivante.

Align2Seq sera une application *responsive*, c'est à dire adaptable à la taille de la fenêtre de n'importe quel support (ordinateur, tv, portable, tablette...). Cela permettra de l'utiliser quelque soit le lieu.

I.4 Le choix du langage

I.4.1 Comparatif entre les langages

Le projet Align2Seq est à but pédagogique. Il est en effet question de réaliser une application permettant de visualiser le fonctionnement pas à pas de deux algorithmes d'alignement de séquences (l'un étant local, l'autre global). Pour ce faire, plusieurs langages de programmation peuvent être utilisés.

Le choix du langage dépend de plusieurs paramètres, le premier étant de déterminer l'utilisation qui sera faite de l'application. Align2Seq est une application web, le choix du langage s'est donc fait entre Java et JavaScript. Le tableau ci-dessous liste les différents éléments pris en compte lors de la décision finale qui s'est porté sur le JavaScript.

TABLE I.1 – Tableau de comparaisons entre Java et JavaScript

Fonctionalités	Java	JavaScript
Langage	Compilé	Interprété
Type	Statique	Dynamique
Applications	Autonomes	Effectuées sur navigateur
Puissance des outils	Très bonne	Bonne
Librairies	Très nombreuses	Nombreuses
Objet	POJO	JSON

Le langage Java permet de créer des applications autonomes tandis que celles créées en JavaScript sont dépendantes d'un navigateur web. Align2Seq étant utilisé uniquement en ligne, le JavaScript est donc dans le cas présent indiqué.

Les outils de Java tout comme son système de débogage sont plus puissants que ceux proposés par JavaScript qui reste tout de même compétent. Les librairies de JavaScript sont également moins nombreuses : cependant, la seule qui sera utilisée pour ce projet est Node.js afin d'effectuer le paquetage des donnés. JQuery est une librairie très intéressante lorsque le langage de programmation utilisé est JavaScript : en effet, cette librairie permet de réduire considérablement le temps de code. Cependant, après discussion avec notre client, elle ne sera pas utilisée pour ce projet.

I.4.2 Choix des versions des langages web

La version de JavaScript utilisée sera ECMAScript 5, la version 6 n'étant pas standardisée pour tous les navigateurs web alors que l'application devra être fonctionnelle pour tous. Deux autres langages seront nécessaires : le HTML dans sa dernière version (HTML5) pour la création de la page web et le CSS 3 pour la mise en page de cette dernière.

CHAPITRE II

Conception

II.1 Architecture du programme

II.1.1 Schéma de l'architecture

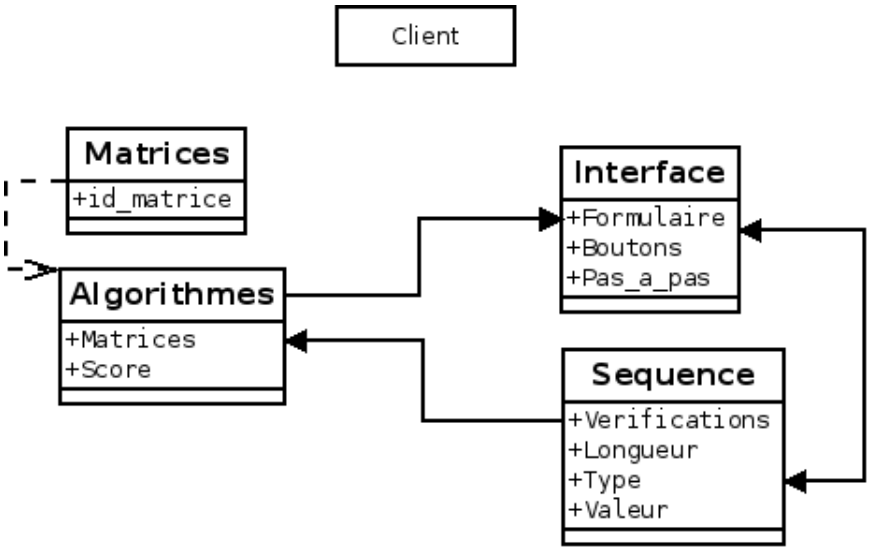


Figure II.1 – Architecture JavaScript de Align2Seq

Tous les prototypes sont implémentés du côté client car le travail est effectué sur de petites séquences. En effet, une liaison à un serveur web entraînerait un ralentissement du chargement de la page car de nombreuses interactions auraient lieu entre l'interface client et le serveur web.

Les flèches signifient que les différents éléments sont reliés :

- Rectangle : chaque rectangle correspond à un module contenant l'ensemble des fonctions ou objets qui lui sont liés.
- Pointillés : Algorithmes a une dépendance envers Matrices
- Simple : les résultats d'Algorithmes sont envoyés vers Interface et les valeurs entrées dans Sequence sont utilisées dans Algorithmes
- Double : il y a une interaction réciproque entre Interface et Sequence

II.1.2 Architecture logiciel

Pour ce projet, l'une des exigences est de rendre facile l'ajout d'algorithmes d'alignement autres que ceux de Needleman et Wunsch et de Smith et Waterman. Il faut donc veiller à la bonne modularité du code. Pour cela, la notion de `prototype`, équivalent de `class` en Java, va permettre de pouvoir dissocier la partie moteur de l'application de la partie affichage.

Une fonction `display` sera chargée des opérations d'affichage en HTML et un prototype `algorithm` utilisera les données saisies dans le formulaire afin d'effectuer les calculs correspondant selon l'algorithme choisi.

Ainsi, l'affichage et les calculs algorithmiques fonctionnent de façon dissociée, ce qui permettra à un autre développeur d'ajouter son propre algorithme d'alignement sans avoir à modifier l'affichage.

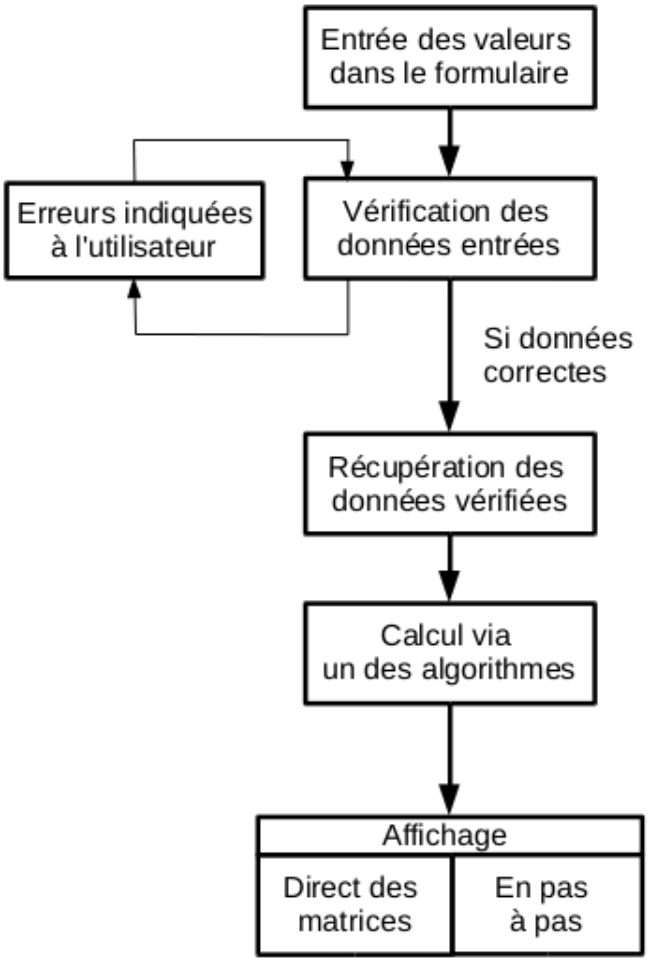


FIGURE II.2 – Schéma pipeline d'Align2Seq

II.2 Align2Seq : une application JavaScript

II.2.1 Les prototypes en JavaScript

Le langage de programmation JavaScript étant orienté objet, il permet donc de créer des objets manipulés par différents prototypes. En effet, on ne parle pas de classes comme dans le langage Java ou C++ : cependant, le principe d'héritage existe.

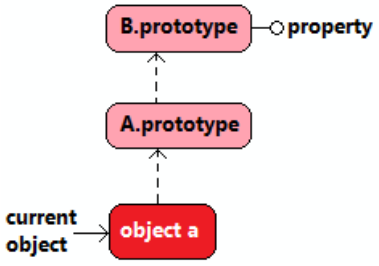


FIGURE II.3 – Exemple du principe de l'héritage [12]

II.2.2 Agencement de l'application

Un objet `Algorithm` sera implémenté. Il contiendra les éléments communs aux deux algorithmes traités.

Deux prototypes hériteront des caractéristiques de l'objet précédemment cité : `smithwaterman` et `needlemanwunsch`. Ceux-ci posséderont chacun une méthode de calcul de l'alignement qui leur sera propre ainsi qu'une méthode de calcul de la matrice score. Les deux algorithmes effectuent le chemin en prenant toujours le meilleur score d'alignement : cependant, l'algorithme de Needleman et Wunsch implique de toujours commencer le chemin sur la dernière case de droite pour remonter vers la première case de gauche, tandis que l'algorithme de Smith et Waterman effectue le calcul du chemin en partant de la case ayant le meilleur score (le score maximal), quel que soit son emplacement dans la matrice. Ainsi, plusieurs chemins sont possibles pour chacun des algorithmes (cependant, dans le cas de Needleman et Wunsch, un seul d'entre eux sera affiché, à cause de la sélection préalable d'une seule case en cas de choix possible de direction).

Align2Seq étant une application web, il sera nécessaire de réaliser une page web claire et graphique à l'aide des technologies du web (HTML5 et CSS3) permettant de gérer l'affichage des différents éléments de résultat. L'affichage de la matrice de score ainsi que du chemin à parcourir sera dynamique (affichage pas à pas, c'est à dire élément par élément) : cela implique que la page HTML soit remplie dynamiquement par les résultats produits depuis les différents prototypes des algorithmes. Il y aura donc un autre module rédigé en JavaScript qui récupérera les résultats d'alignement produits par les prototypes des différents algorithmes et des matrices

de score produites par le prototype initial `Algorithm`.

Cette méthode d'implémentation permettra d'implémenter d'autres prototypes pour d'autres algorithmes par la suite comme par exemple la méthode des quatre Russes [13].

Deux modes d'affichage seront disponibles. Le premier mode affichera les matrices de score et de chemin remplies : pour ce faire, un prototype `Display` sera implémenté. Le second mode est dit pas à pas : la matrice se remplira d'un élément supplémentaire à chaque clic (action de l'utilisateur), en commençant par la matrice de score. Pour chaque case de la matrice de score complétée, la case précédente contiendra la somme maximale ainsi que le chemin par lequel cette somme est obtenue. Pour le mode pas à pas, il est également important de pouvoir retourner en arrière : c'est pourquoi deux fonctions seront implémentées : une fonction `next` pour avancer et une fonction `prev` pour retourner à l'étape précédente.

Ces différentes fonctionnalités seront déclenchées lors de clics sur des boutons de type magnétoscope qui seront disponibles à la fin du formulaire.

II.3 Contrôles

II.3.1 Les choix de l'utilisateur

Il sera primordial de contrôler les entrées et les choix de paramètres effectués par l'utilisateur avant de lancer un alignement quelconque, afin de s'assurer du bon déroulement du programme et de la cohérence des résultats.

Il ne serait par exemple pas souhaitable que l'utilisateur puisse par inadvertance sélectionner une matrice protéique alors que les séquences renseignées sont de type nucléique, ou bien que l'utilisateur puisse entrer une pénalité de gap négative (elles le sont, mais on entre toujours un chiffre positif par convention) ou encore, dans le cas des séquences protéiques, qu'une lettre saisie par l'utilisateur ne corresponde à aucune protéine.

Les exemples d'erreurs de saisie qui pourraient être commises par l'utilisateur sont nombreux. Il nous faudra donc réfléchir au développement de structures de contrôle afin de ne procéder au traitement des données qu'une fois les données entrées en paramètres correctes. En cas d'erreur, il faudrait avertir l'utilisateur afin qu'il puisse changer l'élément erroné : cela implique un enregistrement des données du formulaire sans rafraîchissement de la page.

II.3.2 Les matrices

Comme dit précédemment, notre projet ayant une visée pédagogique, la longueur des séquences à traiter sera relativement courte (15-20 caractères au plus).
Aucun serveur web ne sera utilisé, ce qui implique que les matrices devront se trouver en dur dans l'application et codifiées de façon à pouvoir être utilisées.

De nombreuses matrices protéiques existent, mais seulement trois pour les matrices nucléiques sont présentes : il faudra donc effectuer un parsage des fichiers contenant les matrices et placer le résultat dans un fichier `.json` qui sera utilisé lors des calculs effectués au sein des algorithmes (un fichier par type de matrice).

II.4 Les modes et affichages associés

II.4.1 Normal

Le mode normal sera celui qui montrera à l'utilisateur toutes les matrices permettant d'obtenir le ou les alignements optimaux lors de la comparaison de deux séquences (protéiques ou nucléiques).

Ce mode sera lancé lors d'un clic sur le bouton play du magnétoscope que l'on retrouve dans le formulaire.

Pour calculer les scores et les sommes nécessaires à l'obtention des différentes matrices qui permettront par la suite de réaliser le meilleur alignement avec des flèches colorées, les éléments entrés dans le formulaire seront récupérés dans des variables utilisées par les différents prototypes (`algorithm`, `needlemanwunsch` et `smithwaterman`).

Tous les calculs seront faits en arrière-plan : ainsi, l'utilisateur n'y aura pas accès. De cette façon, pour mieux comprendre le fonctionnement des différents algorithmes, il sera nécessaire d'avoir recours à un affichage pas à pas des différentes étapes représentant les matrices jusqu'à l'affichage final du chemin optimal par des flèches colorées. Cependant, dans le mode normal, cela n'est pas visible car le chemin ne s'affiche qu'en une fois.

II.4.2 Pas à pas

L'affichage en mode pas à pas joue un rôle prépondérant dans cette application car c'est lui qui va permettre à l'utilisateur de comprendre étape par étape comment se construisent la matrice somme et la matrice chemin et quels sont les calculs qui ont conduit à l'aboutissement de ces matrices.

Le mode pas à pas devra être décomposé en trois grandes étapes : l'affichage de la matrice score couplé à celui de la matrice somme, cette dernière comprenant le meilleur score calculé pour chaque cellule ainsi que la flèche associée à ce score. La dernière étape est l'affichage de la matrice chemin, où les flèches des cellules correspondant à l'alignement optimal seront colorées d'une couleur spécifique.

L'utilisateur aura la possibilité de passer à l'étape suivante ou de revenir à l'étape précédente respectivement grâce aux boutons `avance rapide »|` et `retour rapide |«` qui seront présents dans le formulaire.

Au moment de l'initialisation du mode pas à pas, l'utilisateur se verra afficher une matrice vide comportant en première ligne la séquence numéro 1 et en première colonne la séquence numéro 2 puis, au moyen d'un bouton `next >|`, l'utilisateur verra alors se remplir d'un score la première case en haut à gauche de la matrice. Dans le même temps, à droite de la matrice, apparaîtra sous forme textuelle les calculs prenant en compte les valeurs réelles des cellules avoisinantes pour calculer la somme maximale de la cellule courante. La flèche correspondant à ce score sera également affichée dans ce bloc.

De la même manière, à chaque nouveau clic sur le bouton `next >|`, l'utilisateur verra apparaître le contenu de la cellule suivante ainsi que les calculs associés, mais aussi les valeurs de somme et de chemin indiquées pour le tour précédent dans la case précédente en lieu et place du score.

La matrice se remplit ainsi de gauche à droite et de haut en bas.

Une fois le remplissage de la matrice somme terminée, si l'utilisateur clique une nouvelle fois sur le bouton `next >|`, cela lance l'affichage de la matrice chemin. Les cellules de cette matrice ne contiendront que les flèches associées au score optimal calculé dans la matrice somme. Le chemin optimal sera indiqué par une coloration des flèches dans les cellules concernées. Dans le cas où il y aurait plusieurs chemins possibles, l'utilisateur sera en mesure de les visualiser grâce à des doubles ou triples flèches mais un seul sera choisi d'office par l'application (en premier lieu la diagonale, puis la verticale et enfin l'horizontale).

II.5 Les outils de production

II.5.1 Utilisation de Node.js

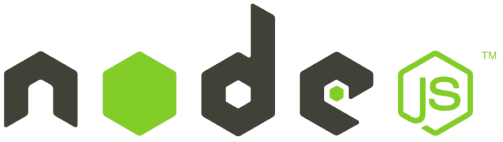


FIGURE II.4 – Logo de la plateforme Node.js

Node.js [14] est une plate-forme logicielle libre créée par Ryan Lienhart Dahl et implémentée en JavaScript. Ses utilisations sont nombreuses et avant tout orientées vers les applications réseaux. Elle sert principalement à faire tourner des serveurs web : cependant, Align2Seq étant uniquement implémentée côté client, Node.js aura donc dans le cas présent une toute autre utilité.

Cette plate-forme peut servir de gestionnaires de modules. Elle permet en effet de créer la documentation JavaScript, de compacter le code ou encore de réaliser la minification de celui-ci (obtention du code JavaScript sans caractère dit inutile comme les retours à la ligne ou les commentaires, afin de le rendre le plus court et le moins lourd possible) via des modules téléchargeables.

Pour Align2Seq, Node.js sera utilisé pour la réalisation du paquetage avec les éléments précédemment cités. La plate-forme sera également utilisée afin de produire la documentation de l'application et d'effectuer tous les services de maintenance du code de l'application.

II.5.2 Outils utilisés

Node.js possède de nombreux outils dont les fonctionnalités varient de l'un à l'autre. Afin d'automatiser les tâches répétitives que sont la minification ou encore la compaction de notre code qui seront nécessaires à chaque étape afin d'avoir un code rapide au fur et à mesure de la réalisation, Grunt [15] sera utilisé.

Après le traitement du code par l'outil Grunt, deux fichiers différents seront produits : l'un avec l'extension .js (le fichier Java-Script initial) et l'autre portant l'extension -min.js. Ce second fichier sera la version minifiée du code, laquelle sera utilisée dans le HTML afin de produire une application très rapide lorsqu'elle sera utilisée par une classe entière au même moment.

Afin de pouvoir exploiter les matrices protéiques (BLOSUM et PAM) et les matrices nucléiques (EDNA) trouvées sur le site EMBOSS, un parser en python sera réalisé et permettra d'obtenir un fichier JSON.

Le fichier JSON [16] sera exploité dans le formulaire pour pouvoir choisir la matrice souhaitée. Dans ce fichier, chaque matrice, associée à son nom, sera placée dans une liste : ainsi, les différentes valeurs seront facilement réutilisables lors des calculs effectués au sein des algorithmes.

CHAPITRE III

RÉALISATION

III.1 Interfaces

III.1.1 Formulaires

La première étape de réalisation de l'application est de créer un formulaire dans lequel les différents éléments nécessaires au calcul de l'alignement seront saisis par le client en fonction de ses besoins.

Ce formulaire est constitué de plusieurs parties : le choix du gap (simple ou multiple), les deux séquences à comparer ou encore le choix de l'algorithme. Une fois que tous les choix sont effectués, il est possible de lancer le calcul des algorithmes et d'afficher le résultat via un bouton situé en fin de formulaire.

Afin d'éviter toute erreur de saisie pouvant entraîner des erreurs lors des calculs d'alignement, le client ne pourra actionner le bouton de lancement de l'application qu'une fois tous les champs du formulaire remplis convenablement.

Ce formulaire permet également de choisir un mode pas à pas qui sera détaillé dans la suite de ce rapport. Pour ce mode, plusieurs fonctionnalités sont ajoutées au formulaire : un bouton d'avance rapide et un autre de retour rapide pour passer d'une étape à l'autre. Deux autres boutons permettent d'afficher le contenu des matrices cellule par cellule ou de les retirer en fonction de celui qui est actionné.

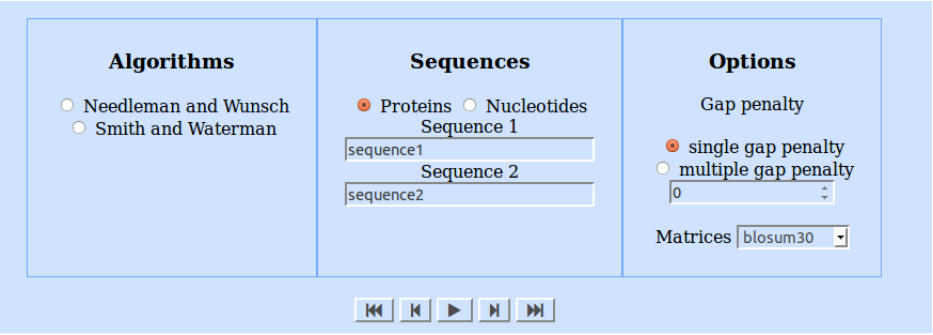


FIGURE III.1 – Formulaire au lancement de la page

III.1.2 Affichages

Deux types d'affichage sont disponibles. Le premier ressort les résultats finaux d'un coup en deux tableaux tandis que le second permet de visualiser pas à pas le fonctionnement des différents calculs et la création de tous les chemins possibles entre les deux séquences comparées.

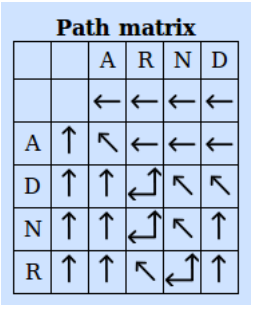


FIGURE III.2 – Tableau représentant le chemin

Le mode pas à pas permet, comme dit précédemment, de visualiser au fur et à mesure la construction du chemin. Il est donc séparé en 3 étapes : la première affiche les éléments du tableau score un par un, la seconde (couplée à la première) affiche la valeur du tableau somme et du chemin dans la case précédente, et la dernière remonte le chemin élément par élément.

Un bouton `preview <|` permet au client de retourner à la vue de la cellule précédente. A l'inverse, un bouton `next |>` permet d'afficher le contenu de la cellule suivante. Le bouton de retour en arrière est d'une grande importance pour bien comprendre le fonctionnement des flèches (horizontale plutôt que verticale par exemple).

III.2 Background

III.2.1 Algorithmes

Les deux algorithmes sur lesquels l'application Align2Seq se base ont une structure commune : un objet `algorithm` est donc créé afin de rassembler les éléments communs aux calculs d'alignement suivant les algorithmes de Needleman et Wunsch ainsi que de Smith et Waterman.

Chaque algorithme est donc représenté comme un nouvel objet ayant les caractéristiques de celui précédemment cité (grâce à l'héritage). Deux fonctions sont implémentées dans chacun des algorithmes : elles permettent de calculer les différents scores suivant les pénalités de gap attribuées et de calculer l'alignement des deux séquences. Au final, la complexité de ces algorithmes est pour chacun de O(nm) avec n et m la longueur des séquences.

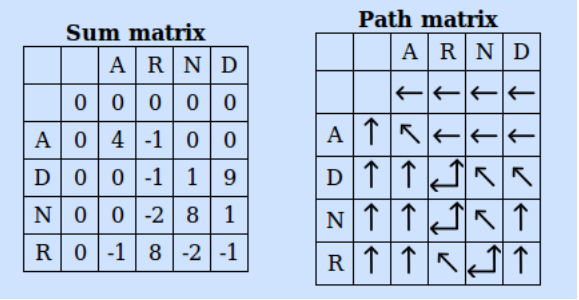
```
1 function needlemanwunsch()
2 {
3   for (key in algorithm.prototype) {
4     needlemanwunsch.prototype[key] = algorithm.prototype[key];
5   }
6 }
7 needlemanwunsch.prototype.score = function (matrix,matscore, matpath,
    matsumdia...)
```

Dans l'exemple ci-dessus, la fonction `needlemanwunsch()` hérite des éléments(key) provenant de l'objet `algorithm` lequel est caractérisé par le mot-clef prototype. La fonction score, quand à elle, va hériter de `needlemanwunsch()` : elle effectuera donc tous ses calculs à partir des éléments trouvés dans l'objet initial `algorithm`.

III.2.2 Les fonctions d'affichage

Deux fonctions d'affichage sont disponibles comme explicité précédemment, et deux fichiers permettent de moduler ces deux affichages.

La première fonction, `display`, gère l'affichage général des résultats. Elle permet de visualiser directement le contenu complet des matrices de score et du chemin (représenté par des flèches et obtenu par le calcul des sommes maximales) ainsi que le ou les chemin(s) représentant le ou les meilleur(s) alignement(s) possible(s) des deux séquences comparées. Cette fonction récupère les différents éléments placés dans des listes. Le résultat obtenu après le lancement de cette fonction est le suivant :



		A	R	N	D
	0	0	0	0	0
A	0	4	-1	0	0
D	0	0	-1	1	9
N	0	0	-2	8	1
R	0	-1	8	-2	-1

		A	R	N	D
		←	←	←	←
A	↑	↖	←	←	←
D	↑	↑	↵	↖	↖
N	↑	↑	↵	↖	↑
R	↑	↑	↖	↵	↑

FIGURE III.3 – Affichage en mode normal

Afin que ces matrices ne se dupliquent pas en cas de clic supplémentaire par erreur sur le bouton de lancement de la fonction, le contenu des tableaux est supprimé lorsqu'il y a déjà des éléments au sein de ceux-ci. Ces tableaux sont dimensionnés en fonction de la longueur des séquences entrées par l'utilisateur et créées de la façon suivante : les méthodes inserRow et insertCell permettent respectivement d'ajouter le nombre de lignes en fonction de la longueur de la séquence 2 et le nombre de cellules en fonction de la longueur de la séquence 1.

```
1  function display(){
2
3    document.getElementById("matrixsum").removeChild(matrixsum.childNodes[0]);
4
5    var matrixs=document.getElementById("matrixsum");
6
7    for (var i =0;i<=(size2-1);i++){
8      matrixs.insertRow(i);
9      for(var j=0;j<=(size1);j++){
10       matrixs.rows[i].insertCell(j);
11     }
12   }
```

L'application devant être adaptable à la taille de petits écrans comme ceux des tablettes, la longueur des séquences est donc limitée à 15 caractères. En effet, au-delà de cette limite, les tableaux ont des tailles trop importantes et ne sont pas visualisables dans leur totalité. En cas de réduction de la fenêtre du navigateur, un ascenseur horizontal apparaît en bas de page pour naviguer.

III.3 Le mode pas à pas

III.3.1 Visualisation des matrices

La visualisation des différentes matrices se fait en plusieurs étapes. La première consiste à afficher la matrice de score (elle se remplit d'une case à chaque clic). Une fois la case complète, chaque nouveau clic en remplace le score par la somme maximale et la flèche (diagonale, horizontale ou verticale) correspondante pour visualiser la création des chemins potentiels, tout en affichant le score de la case suivante.

Une fois toutes les flèches remplies, il est possible de n'afficher que le meilleur chemin (représentant le meilleur alignement) avec des flèches rouges lorsqu'il s'agit de l'algorithme de Needleman et Wunsch. Les alignements pour l'algorithme de Smith et Waterman sont représentés de la même façon : ainsi tous les chemins possibles sont aisément visualisables par l'utilisateur.

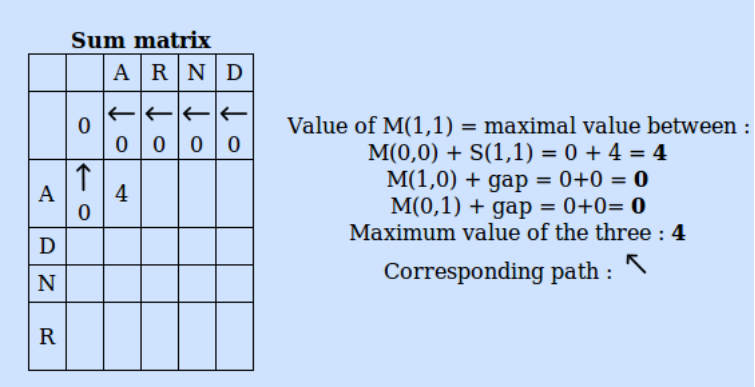


FIGURE III.4 – Mode pas à pas en cours de fonctionnement

III.3.2 Equations

L'application Align2Seq ayant un but pédagogique, il est intéressant de pouvoir visualiser les différents calculs permettant de réaliser la matrice somme en parallèle du remplissage de celle-ci par les flèches correspondantes au fur et à mesure.

La matrice est placée dans un tableau (matrice score puis somme avec flèches correspondantes) et les calculs permettant l'obtention des sommes se situent sur la droite de ce tableau.

Les équations pour les deux algorithmes traités dans cette application sont du type suivant :

$$\text{M(i,j)=max} \begin{cases} M(i-1,j-1)+S(i,j) \\ M(i,j-1)+gap \\ M(i-1,j)+gap \end{cases}$$

M(i,j) correspond à la cellule dans laquelle se trouve l'utilisateur, S au score dans la case indiquée, i et j respectivement à la position dans la séquence 1 et 2.

La valeur maximale des trois calculs précédents est récupérée et permet par la suite de réaliser la matrice somme qui sera à l'origine du chemin et du meilleur alignement possible des deux séquences entrées au départ.

Les gaps peuvent être multiples : le calcul reste dans ce cas le même. Cependant, il doit prendre en compte le changement de gap en fonction de la cellule dans laquelle se trouve l'utilisateur : les résultats en sont donc affectés, tout comme les chemins.

III.4 Les vérifications et tests

III.4.1 Formulaire

Les fonctions d'affichage, qu'elles concernent le mode normal ou le mode pas à pas, ne doivent pas pouvoir être exécutées tant que le formulaire n'est pas totalement et convenablement rempli. Des vérifications sont effectuées sur les boutons permettant le lancement des fonctions `display` et `timelapse` car ce sont elles qui exécutent les deux modes.

Ces vérifications consistent à afficher un message d'erreur lorsque l'utilisateur essaie de lancer les fonctions citées sans avoir rempli le formulaire convenablement ou en totalité.

L'utilisateur ne pourra donc pas obtenir de résultats incohérents du fait d'une mauvaise saisie du formulaire.

Une autre vérification consiste à limiter la taille des séquences entrées afin que l'affichage soit toujours correct lorsque l'application est utilisée sur des écrans de petite taille. Si l'une des séquences entrées par l'utilisateur est trop grande, alors un message d'erreur le lui indique. Après plusieurs essais, la longueur maximale retenue pour les séquences est de 15 caractères. Une longueur minimale placée à 2 caractères est également incluse.

```
1 function verif () {
2   if ((verif_check_algo()===true)&&(verif_check_type_seq()===true)&&(
        check_content_seq()===true)&&(verif_choice_gap()===true)){
3     return true;
4   }
5   else{
6     alert("complete the form before submit it");
7     return false;
8   }
9 }
```

Ci-dessus, un exemple de code permettant de vérifier que le formulaire est bien rempli dans sa totalité avant de pouvoir actionner le bouton de lancement des différentes fonctions d'affichage.

III.4.2 Affichage

Certaines cellules doivent être remplies dès le moment de l'affichage, tandis que d'autres doivent se remplir uniquement lors du clic sur le bouton `next |>` dans le mode pas à pas. Il faut donc effectuer des tests afin que le déroulement de l'affichage ne se fasse qu'à partir de la cellule (i=1,j=1). De la même façon, lors d'un clic sur le bouton `prev <|`, celle-ci doit s'arrêter lorsque l'utilisateur atteint la cellule précédemment citée. Ci-dessous, un exemple de remplissage dans la fonction `display`

```
1  if(i>=1 && j===1){
2    for (scoring in matscore){
3      matrixsum.rows[i].cells[j].innerHTML=matscore[scoring];
4      j++;
5      if(j%(size1+1)===0){
6        i++;
7        j=1;
8      }
9    }
10   i=1;
11 }
```

III.4.3 Compatibilité

L'application Align2Seq est une application web, elle doit donc être compatible avec un maximum de navigateurs car elle a également un rôle pédagogique, ce qui signifie qu'elle sera utilisée sur différentes machines.

Voici un tableau récapitulatif des différents navigateurs et des versions à partir desquels fonctionnent cette application :

Logo des navigateurs web					
Compatibilité	√	√	√	×	√
Version	41 pour Chromium	37	A partir de 9		

FIGURE III.5 – Tableau représentatif des navigateurs compatibles avec l'application

Ce tableau permet de voir que l'application fonctionne sur la plupart des navigateurs web couramment utilisés. Cependant, lorsque ces navigateurs sont anciens, elle ne fonctionne pas toujours en raison de certaines méthodes utilisées lors de la réalisation du code.

III.5 Perspectives

Cette application pourrait inclure à l'avenir d'autres algorithmes d'alignement de séquence, il faudrait alors retravailler la modularité du code. On pourrait imaginer créer un prototype `Algorithm` possédant les éléments principaux nécessaires aux calculs uniquement (récupération des éléments du formulaire). Les différents algorithmes dépendraient donc de ce prototype par la suite. De cette façon, il y aurait moins d'éléments en paramètres des différentes fonctions, ce qui rendrait le code plus modulable et réutilisable.

L'utilisateur choisit lui-même si les séquences entrées sont protéiques ou nucléiques (le mode par défaut étant protéique), même si une vérification est utilisée par rapport aux lettres indiquées dans les séquences. Cependant les matrices même nucléiques utilisent une grande nombre de lettres dont la plupart sont communes à celles utilisées par les matrices protéiques : le choix pourrait alors être erroné. Une perspective pourrait être de limiter les séquences nucléiques aux quatre lettres de base (A,T,C ou G) et donc d'adapter les matrices nucléiques à ce choix.

Actuellement, toutes les séquences comprises entre 2 et 15 caractères peuvent être indiquées, même si elles sont de tailles différentes. Cependant, dans la version actuelle, si les tailles des deux séquences sont différentes, la plus grande des séquences doit être la séquence 1, sous peine de ne pas avoir de résultats corrects. Il serait donc utile de modifier cette exception, et permettre que la séquence la plus grande soit la séquence 2 tout comme la séquence 1, au choix de l'utilisateur.

L'application, comme dit précédemment, ne fonctionne pas toujours sur les anciennes versions des navigateurs web. Ne sachant pas la version utilisée par les utilisateurs, il serait intéressant de revoir le code pour le rendre compatible avec des versions antérieures de certains navigateurs.

CHAPITRE IV

Remerciements

Nous remercions le professeur Jean-Christophe Taveau pour son aide, ses conseils et sa disponibilité, ainsi que pour la possibilité de travailler dans sa salle informatique.
Merci également au CREMI pour l'utilisation plus que fréquente de ses salles en libre-service.
Nos remerciements vont enfin à notre jury pour la lecture de notre rapport et l'utilisation de notre application pédagogique.

Bibliographie

[1] http ://www.w3.org/TR/html5/

[2] http ://www.w3.org/Style/CSS/

[3] http ://www.w3.org/standards/webdesign/script

[4] Needleman, Saul B. ; and Wunsch, Christian D. (1970)"A general method applicable to the search for similarities in the aminoacid sequence of two proteins".Journal of Molecular Biology 48 (3) : 443-53.

[5] Smith, Temple F. ; and Waterman, Michael S. (1981). "Identification of Common Molecular Subsequences". Journal of Molecular Biology 147 : 195–197.

[6] Dayhoff, M. O. ; Schwartz, R. M. ; Orcutt, B. C. (1978). "A model of evolutionary change in proteins". Atlas of Protein Sequence and Structure 5 (3) : 345–352.

[7] Henikoff, S. ; Henikoff, J.G. (1992). "Amino Acid Substitution Matrices from Protein Blocks". PNAS 89 (22) : 10915–10919.

[8] www.ebi.ac.uk/

[9] Norman Casagrande : http ://sourceforge.net/projects/baba/

[10] Hubert Bert : http ://ds9a.nl/nwunsch/nwunsch.js

[11] http ://www.w3.org/

[12] http ://thoughtsonscripts.blogspot.fr/2011/12/javascript-prototype.html

[13] https ://www.youtube.com/watch ?v=cYJrMUvJQGc

[14] https ://nodejs.org/

[15] http ://gruntjs.com/

[16] http ://www.json.org/

ANNEXE A

Code de l'application

Printed by Aurore

mai 09, 15 12:13 **algorithm.js** Page 1/4

```
/*
 *  align2seq: Pairwise alignements algorithms in JavaScript, html5, and css3
 *  Copyright (C) 2015
 *
 *  This file is part of align2seq.
 *
 *  align2seq is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  align2seq is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with align2seq.  If not, see <http://www.gnu.org/licenses/>
 *
 * Authors:
 * Rudy Anne
 * Aurelien Beliard
 * Emeline Duquenne
 * Aurore Perdriau
 */

/** First file executed after data treatment, creation of matrices for applicati
on of the chosen algorithm
@constructor
@param {string} sequence1 - The first sequence entered by the user, in the first
 line of the score matrix
@param {string} sequence2 - The second sequence entered by the user, in the firs
t column of the score matrix
@param {array} matrix - Substitution matrix chosen by the user
@param {string} type_seq - If the sequences are proteins or nucleotides
@algo {string} - If the algorithm used is S&W or N&W (for the moment)
@gap {number} - Gap penality */
function algorithm(sequence1,sequence2,matrix,type_seq,algo,gap)
{
        //initialization of the values
        this.seq1 = sequence1;
        this.seq2 = sequence2;
        this.len1 = this.seq1.length;
        this.len2 = this.seq2.length;
        this.lenmax= Math.max(this.len1,this.len2);
        this.lenmin=Math.min(this.len1,this.len2);
        this.matrix = matrix;
        this.type_seq=type_seq;
        this.gap=gap;
        this.place = 0;
        this.s1 = [];
        this.s2 = [];
        this.letters=[];
        this.matseq = [];
        this.matscore = [];
        this.matpath = [];
        this.matpatharrows = [];
        this.matpatharrowsalign = [];
        this.matsumdia=[];
        this.matsumhor=[];
        this.matsumvert=[];
        this.matsumtot=[];
```

mai 09, 15 12:13 **algorithm.js** Page 2/4

```
        this.i;
        this.j;
        this.size1=len1+1;
        this.size2=len2+2;
        this.algo = algo;
        this.listalign=[];
        this.matseq[0] = "-";

        //division of the sequences into separate letters
        s1 = this.seq1.split("");
        for (var elems1 = 0; elems1 <= this.len1; elems1++) {
                this.matseq.push(s1[elems1]);
        }
        this.matseq[this.len1 + 1] = "-";
        s2 = this.seq2.split("");
        for (var elems2 = 0; elems2 < this.len2; elems2++) {
                this.matseq.push(s2[elems2]);
        }

        //creation of the gap tables
        if (isNaN(this.gap) === false ){
                var gapsimple=this.gap;
                var gap =[];
                for(var i=0;i<size1;i++){
                        gap.push(gapsimple);
                }
                this.gap=gap;
        }
        this.gap2=gap;
        if (len1!=len2){
                if (len2<len1){
                        gap2=[];
                        for(var j=0;j<(this.lenmin+1);j++){
                                gap2.push(gap[j]);
                        }
                this.gap2=gap2;
                }
        }

        //selection of the good gap for the each comparison and the letters depe
nding the selected matrix, and calculation of the score
        for (j = this.len1 + 1; j <= ((this.len1 + this.len2) + 1); j++) {
                for (i = 0; i <= this.len1; i++) {
                        if ((this.place<(this.len1+1))){
                                this.gapplace=this.gap[place];
                        }
                        else{
                                this.gapplace=this.gap[this.place%(this.len1+1)]
;
                                this.gapplace2=this.gap2[Math.floor(this.place/(
this.len1+1))];
                        }
                        if (this.type_seq=="protein"){
                                this.letters=["A","R","N","D","C","Q","E","G",
"H","I","L","K","M","F","P","S","T","W","Y","V","B","Z","X",  "*"];
                        }
                        else{
                                if (this.matrix=="EDNAFULL"){
                                        this.letters=["A","T","G","C","S","W",
"R","Y","K","M","B","V","H","D","N","U"];
                                }
                                else{
```

samedi mai 09, 2015 algorithm.js 1/2

```
                                        this.letters=["A","B","C","D","G","H",
"K","M","N","R","S","T","U","V","W","X","Y"];
                                }
                        }
                        if (this.algo=="smith_waterman"){
                                smithwaterman.prototype.score(this.matrix,this.m
atscore, this.matpath, this.matsumdia, this.matsumvert, this.matsumhor,this.mats
umtot, this.matseq[j], this.matseq[i], this.lenmax, this.place,this.gapplace,thi
s.gapplace2,this.letters);
                        }
                        else{
                                needlemanwunsch.prototype.score(this.matrix,this
.matscore, this.matpath, this.matsumdia, this.matsumvert, this.matsumhor,this.ma
tsumtot, this.matseq[j], this.matseq[i], this.lenmax, this.place,this.gapplace,t
his.gapplace2,this.letters);
                        }
                        this.place++;
                }
        }

        //substituton of the value of the path by the corresponding arrow for ea
ch case (in alignment or not)
        for (path in matpath){
                if (matpath[path]===0){
                        matpatharrows[path]="";
                        matpatharrowsalign[path]="";
                }
                else if (matpath[path]===1){
                        matpatharrows[path]="<object type=\"image/svg+xml\" data=\"..\/img\/hor
i.svg\" width=\"25 px\" height=\"25\"> error </object>";
                        matpatharrowsalign[path]="<object type=\"image/svg+xml\" data=\"..\/i
mg\/horir.svg\" width=\"25 px\" height=\"25\"> error </object>";
                }
                else if (matpath[path]===2){
                        matpatharrows[path]="<object type=\"image/svg+xml\" data=\"..\/img\/dia
g.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                        matpatharrowsalign[path]="<object type=\"image/svg+xml\" data=\"..\/i
mg\/diagr.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                }
                else if (matpath[path]===3){
                        matpatharrows[path]="<object type=\"image/svg+xml\" data=\"..\/img\/ver
t.svg\" width=\"25 px\" height=ÃM–^W\"25 px\"> error </object>";
                        matpatharrowsalign[path]="<object type=\"image/svg+xml\" data=\"..\/i
mg\/vertr.svg\" width=\"25 px\" height=ÃM–^W\"25 px\"> error </object>";
                }
                else if (matpath[path]===4){
                        matpatharrows[path]="<object type=\"image/svg+xml\" data=\"..\/img\/bih
v.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                        matpatharrowsalign[path]="<object type=\"image/svg+xml\" data=\"..\/i
mg\/bihvr.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                }
                else if (matpath[path]===5){
                        matpatharrows[path]="<object type=\"image/svg+xml\" data=\"..\/img\/bih
d.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                        matpatharrowsalign[path]="<object type=\"image/svg+xml\" data=\"..\/i
mg\/bihdr.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                }
                else if (matpath[path]===6){
                        matpatharrows[path]="<object type=\"image/svg+xml\" data=\"..\/img\/bid
v.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                        matpatharrowsalign[path]="<object type=\"image/svg+xml\" data=\"..\/i
mg\/bidvr.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
```



```
                }
                else if (matpath[path]===7){
                        matpatharrows[path]="<object type=\"image/svg+xml\" data=\"..\/img\/tri.
svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                        matpatharrowsalign[path]="<object type=\"image/svg+xml\" data=\"..\/i
mg\/trir.svg\" width=\"25 px\" height=\"25 px\"> error </object>";
                }
        }

        //calculation of the alignment(s)
        if (this.algo == "smith_waterman") {
                var result = smithwaterman.prototype.alignment(this.matpath, thi
s.matscore,this.matsumtot, this.s1, this.s2, this.len1, this.lenmax);
        }
        else{
                var result = needlemanwunsch.prototype.alignment(this.matpath, t
his.matscore,this.matsumtot, this.s1, this.s2, this.len1, this.lenmax);
        }

        //display of the alignement(s)
        if (result.length>2){
                var cpt=1;
                document.getElementById('alignment').innerHTML="";
                for(var alignseq=0;alignseq<=(result.length-1);alignseq+=2){
                        document.getElementById('alignment').innerHTML +="<div id=\"al
align\"><h3>Alignment "+cpt+"</h3><br/>"+result[alignseq]+"<br/>"+result[alignseq+1]+"<br
/></div>";
                        cpt++;
                }
        }
        else{
                document.getElementById('alignment').innerHTML ="<h3>Alignment</h3><br
>"+result[0]+"<br>"+result[1];
        }
}
```

```
/*
 *  align2seq: Pairwise alignements algorithms in JavaScript, html5, and css3
 *  Copyright (C) 2015
 *
 *  This file is part of align2seq.
 *
 *  align2seq is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  align2seq is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with align2seq.  If not, see <http://www.gnu.org/licenses/>
 *
 * Authors:
 * Rudy Anne
 * Aurelien Beliard
 * Emeline Duquenne
 * Aurore Perdriau
 */
/** Last file executed after data treatment : creation of the displayed matrices
, without step by step
@constructor
 */
 function display(){
        //elimination of the elements already in results
        matrixsum.childNodes=[];
        while(matrixsum.hasChildNodes()) {
                matrixsum.removeChild( matrixsum.childNodes[0] );
        }
        while(matrixpath.hasChildNodes()){
                matrixpath.removeChild( matrixpath.childNodes[0] );
        }

        //creation of score matrix
        var matrixs=document.getElementById("matrixsum");

        for (var i =0;i<=(size2-1);i++){
                matrixs.insertRow(i);
                for(var j=0;j<=(size1);j++){
                        matrixs.rows[i].insertCell(j);
                }
        }

        //filling of score matrix
        var matrix2=document.getElementById("matrixsum").rows;
        for (var i = 0 ; i < matrix2.length; i++) {
                var column = matrix2[i].cells;
                for (var j = 0; j < column.length ; j++) {
                        if (i>=2 && j===0){
                                for(var column in s2){
                                        matrixsum.rows[i].cells[j].innerHTML=s2[
column];
                                        i++;
                                }
                        }
                        if (i===0 && j>=2) {
```



```
                                for (var ligne in s1) {
                                        matrixsum.rows[i].cells[j].innerHTML=s1[
ligne];
                                        j++;
                                }
                        }
                        if(i>=1 && j===1){
                                for (scoring in matscore){
                                        matrixsum.rows[i].cells[j].innerHTML=mat
score[scoring];
                                        j++;
                                        if(j%(size1+1)===0){
                                                i++;
                                                j=1;
                                        }
                                }
                        i=1;
                        }
                }
        var title=document.getElementById("matrixsum").createCaption();
        title.innerHTML="<b>Sum matrix</b>";
        }

        //creation of path matrix
        var matrixp=document.getElementById("matrixpath");
        for (var i =0;i<=(size2-1);i++){
                matrixp.insertRow(i);
                for(var j=0;j<=(size1);j++){
                        matrixp.rows[i].insertCell(j);
                }
        }

        //filling of path matrix
        var matrix3=document.getElementById("matrixpath").rows;

        //crÃ©ation of lines
        for (var i = 0 ; i < matrix3.length; i++) {
                var cpt=0;
                var column = matrix3[i].cells;

                //Creation of the cells
                for (var j = 0; j < column.length ; j++) {
                        if (i>=2 && j===0){
                                for(var column in s2){
                                        matrixpath.rows[i].cells[j].innerHTML=s2
[column];
                                        i++;
                                }
                        }
                        if (i===0 && j>=2) {
                        //filling of the first line from the second cell
                                for (var ligne in s1) {
                                        matrixpath.rows[i].cells[j].innerHTML=s1
[ligne];
                                        j++;

                                }
                        }
                        if(i>=1 && j===1){
                                for (path in matpath){
                                        matrixpath.rows[i].cells[j].innerHTML=ma
tpatharrows[cpt];
```

```
                                        j++;
                                        if(j%(size1+1)===0){
                                                i++;
                                                j=1;
                                        }
                                        cpt++;
                                }
                        i=1;
                        }
                }
        var title=document.getElementById("matrixpath").createCaption();
        title.innerHTML="<b>Path matrix</b>";
        }
}
```

```
/*
* align2seq: Pairwise alignements algorithms in JavaScript, html5, and css3
* Copyright (C) 2015
*
* This file is part of align2seq.
*
* align2seq is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* align2seq is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with align2seq. If not, see <http://www.gnu.org/licenses/>
*
* Authors:
* Rudy Anne
* Aurelien Beliard
* Emeline Duquenne
* Aurore Perdriau
*/
/**
 * Function to put in the form the right matrices according to sequence type and
 the right gap penalities
 */
"use strict";

 function event_onload(){
        if(document.getElementById('nucleotide').checked!==true){
        var listmatrix=Object.keys(matrixlist);
                for (var l in listmatrix){
                        document.getElementById('choice_matrix').options[l] = new O
ption(listmatrix[l],listmatrix[l]);
                }
        }
        else{
                while (document.getElementById('choice_matrix').firstChild){
                        document.getElementById('choice_matrix').removeChild(docume
nt.getElementById('choice_matrix').firstChild);
                }
                var listEDNA=Object.keys(matrixEDNA);
                for (var l in listEDNA){
                        document.getElementById('choice_matrix').options[l] = new O
ption(listEDNA[l],listEDNA[l]);
                }
        }

        var seq1=document.getElementById("sequence1");
        var seq2=document.getElementById("sequence2");
        var enter_gap_penalty =document.getElementById("gap");

        while(enter_gap_penalty.firstChild){
                enter_gap_penalty.removeChild(enter_gap_penalty.firstChild);
        }

        if (document.getElementById('multiple').checked!==true){
                var enter_gap=document.createElement('input');
                enter_gap.setAttribute("type","number");
```



```
                enter_gap.setAttribute("min", "0");
                enter_gap.setAttribute("id", "enter_gap_penalty");
                enter_gap.setAttribute("value", "0");
                enter_gap.setAttribute("size", 2);
                enter_gap_penalty.appendChild(enter_gap);
        }
        else{
                var lengthmax=Math.max(seq1.value.length,seq2.value.length);

                for (var m =0; m <= lengthmax; m++) {
                        if (enter_gap_penalty.hasChildNodes===true){
                                var enter_gap=document.createElement('input');
                                enter_gap.insertBefore(enter_gap,enter_gap_penal
ty.lastChild);
                                enter_gap.setAttribute("type","number");
                                enter_gap.setAttribute("min", "0");
                                enter_gap.setAttribute("id", "enter_gap_penalty"+m);
                                enter_gap.setAttribute("value", "0");
                                enter_gap.setAttribute("style", "width:2em");
                        }
                        else {
                                var enter_gap=document.createElement('input');
                                enter_gap_penalty.appendChild(enter_gap);
                                enter_gap.setAttribute("type","number");
                                enter_gap.setAttribute("min", "0");
                                enter_gap.setAttribute("id", "enter_gap_penalty"+m);
                                enter_gap.setAttribute("value", "0");
                                enter_gap.setAttribute("style", "width:2em");
                        }
                }
        }
}


/**
 * Function to put in the form the right matrices according to sequence type
 */

function choose_matrix(){
        if (document.getElementById('protein').checked===true){
                while (document.getElementById('choice_matrix').firstChild){
                        document.getElementById('choice_matrix').removeChild(docume
nt.getElementById('choice_matrix').firstChild);
                }
                var listmatrix=Object.keys(matrixlist);
                for (var l in listmatrix){
                        document.getElementById('choice_matrix').options[l] = new O
ption(listmatrix[l],listmatrix[l]);
                }
        }
        else if (document.getElementById('nucleotide').checked===true){
                while (document.getElementById('choice_matrix').firstChild){
                        document.getElementById('choice_matrix').removeChild(docume
nt.getElementById('choice_matrix').firstChild);
                }
                var listEDNA=Object.keys(matrixEDNA);
                for (var l in listEDNA){
                        document.getElementById('choice_matrix').options[l] = new O
ption(listEDNA[l],listEDNA[l]);
                }
        }
}
```

```
/** Function to put in the form the right inputs for gap penalities according to
 user choice
@constructor
 */

function choose_gap_penalty(){
        var seq1=document.getElementById("sequence1").value;
        var seq2=document.getElementById("sequence2").value;
        var enter_gap_penalty =document.getElementById("gap");
        while(enter_gap_penalty.firstChild){
                enter_gap_penalty.removeChild(enter_gap_penalty.firstChild);
        }
        if (document.getElementById('single').checked===true){
                var enter_gap=document.createElement('input');
                enter_gap.setAttribute("type","number");
                enter_gap.setAttribute("min", "0");
                enter_gap.setAttribute("id", "enter_gap_penalty");
                enter_gap.setAttribute("value", "0");
                enter_gap.setAttribute("size", 2);
                enter_gap_penalty.appendChild(enter_gap);
        }
        if (document.getElementById('multiple').checked===true){
                var lengthmax=Math.max(seq1.length,seq2.length);
                for (var m =0; m <= lengthmax; m++) {
                        if (enter_gap_penalty.hasChildNodes===true){
                                var enter_gap=document.createElement('input');
                                enter_gap.insertBefore(enter_gap,enter_gap_penal
ty.lastChild);
                                enter_gap.setAttribute("type","number");
                                enter_gap.setAttribute("min", "0");
                                enter_gap.setAttribute("id", "enter_gap_penalty"+m);
                                enter_gap.setAttribute("value", "0");
                                enter_gap.setAttribute("style", "width:2em");
                        }
                        else {
                                var enter_gap=document.createElement('input');
                                enter_gap_penalty.appendChild(enter_gap);
                                enter_gap.setAttribute("type","number");
                                enter_gap.setAttribute("min", "0");
                                enter_gap.setAttribute("id", "enter_gap_penalty"+m);
                                enter_gap.setAttribute("value", "0");
                                enter_gap.setAttribute("style", "width:2em");
                        }
                }
        }
}

/**
 * Function to obtain the user values for the treatment of the sequences alignme
nt
 */
function get_value(){
        var algo;
        var type_seq;
        var li_gap=[];
        var algo_choice=document.getElementsByName("algorithm");
        for (var i=0;i<algo_choice.length;i++){
                if (algo_choice[i].checked===true){
                        algo=algo_choice[i].value;
                }
        }
```



```
        var seq_choice=document.getElementsByName("type_seq");
        for (var j=0;j<seq_choice.length;j++) {
                if (seq_choice [j].checked===true) {
                        type_seq= seq_choice[j].value;
                }
        }

        var seq1=document.getElementById("sequence1").value.toUpperCase();
        var seq2=document.getElementById("sequence2").value.toUpperCase();
        var matrix=document.getElementById("choice_matrix").options[document.getEle
mentById('choice_matrix').selectedIndex].value;
        if (document.getElementById('protein').checked===true){
                var matrix=matrixlist[matrix];
        }
        else{
                var matrix=matrixEDNA[matrix];
        }
        if (document.getElementById("single").checked===true){
                var gap=document.getElementById("enter_gap_penalty").value;
                var gap=-Math.abs(parseInt(gap,10));
                algorithm(seq1,seq2,matrix,type_seq,algo,gap);
        }
        else if (document.getElementById("multiple").checked===true){
                var max_len=seq1.length;
                if (seq2.length>max_len){
                        max_len=seq2.length;
                }
                for (var i =0;i<(max_len+1);i++) {
                        var tmp=document.getElementById("enter_gap_penalty"+i).value
;
                        tmp=Math.abs(parseInt(tmp,10));
                        li_gap.push(-tmp);
                // li_gap contains list of gaps in digital format
                }
        algorithm(seq1,seq2,matrix,type_seq,algo,li_gap);
        }
}

/**
 * Function to verify if all values are correctly filled
*/
function verif () {
        if ((verif_check_algo()===true)&&(verif_check_type_seq()===true)&&(check
_content_seq()===true)&&(verif_choice_gap()===true)){
                return true;
        }
        else{
                alert("complete the form before submit it");
                return false;

        }
}


/**
 * Function to verify if the choice of algorithm is correctly filled
 */
function verif_check_algo(){
        var algo_checked=false;
        var algo_choice=document.getElementsByName("algorithm");
        for (var i=0;i<algo_choice.length;i++){
                if (algo_choice[i].checked===true){
```

```
                        var algo=algo_choice[i].value;
                        if (algo !== ""){
                                algo_checked=true;
                                }
                }
        }
        if (algo_checked===false){
                alert("Choose an algorithm to obtain a result.");
        }
        return(algo_checked);
}

/**
 * Function to verify if the choice of type sequence is correctly filled
 */
function verif_check_type_seq(){
        var type_seq_check=false;
        var seq_choice=document.getElementsByName("type_seq");
        var type_seq;
        for (var j=0;j<seq_choice.length;j++) {
                if (seq_choice [j].checked===true) {
                        type_seq=seq_choice[j].value;
                        type_seq_check=true;
                }
        }
        if (type_seq_check===false){
        alert("Choose a sequence type.") ;
        }
        return(type_seq_check);
}

/**
 * Function to verify if the chosen sequences are correctly filled
 */

function check_content_seq () {
        var content_seq_check=false;
        var seq2=document.getElementById("sequence1").value.toUpperCase();
        var seq1=document.getElementById("sequence2").value.toUpperCase();
        var type_seq;
        var seq_choice=document.getElementsByName("type_seq");
        for (var j=0;j<seq_choice.length;j++) {
                if (seq_choice [j].checked===true) {
                        type_seq=seq_choice[j].value;
                }
        }
        if (((2<=seq1.length)&&(seq1.length<=15))&&((2<=seq2.length<=15)&&(seq2.
length<=15))){
                if(((type_seq=="protein")&&(/^[ARNDCQEGHILKMFPSTWYVBZX]+$/.test(se
q1,seq2)))||((type_seq=="nucleotide")&&(/^[ATGCSWRYKMBVHDNU]+$/.test(seq1,seq2))))
{
                        content_seq_check=true;
                }
                else{
                        alert("the content of the sequence did not match the sequence type you have ch
ecked");
                }
        }
        else{
                alert("the sequence is too small or too long for the application");
        }
        return(content_seq_check);
```



```
}

/**
 *  Function to verify if the choice of gap penality is correctly filled
*/
function verif_choice_gap(){
        var check_gap=false;
        var choice_gap_check=false;
        var number=false;
        var numbers=false;
        var choice_nbgap=document.getElementsByName("choose_gap_penalty");
        var nb;
        var not_equal=false;
        for (var j=0;j<choice_nbgap.length;j++) {
                if (choice_nbgap[j].checked===true) {
                        nb=choice_nbgap[j].value;
                        choice_gap_check=true;
                }
        }
        if (nb=="single"){
                var gap=document.getElementById("enter_gap_penalty").value;
                var gap=-Math.abs(parseInt(gap,10));
                if(isNaN(gap)===false){
                        number=true;
                }
        }

        if(nb=="multiple"){
                var seq1=document.getElementById("sequence1").value;
                var seq2=document.getElementById("sequence2").value;
                var li_gap=[];
                var max_len=seq1.length;
                if (seq2.length>max_len){
                        max_len=seq2.length;
                }
                for (var i =0;i<max_len;i++) {
                        var tmp2=document.getElementById("enter_gap_penalty"+0).valu
e;
                        var tmp=document.getElementById("enter_gap_penalty"+i).value
;
                        tmp=parseInt(tmp,10);
                        tmp2=parseInt(tmp2,10);
                        if (tmp==tmp2) {
                                not_equal=true;
                        }
                        tmp=Math.abs(parseInt(tmp,10));
                        if (isNaN(Math.abs(parseInt(tmp,10)))===false){
                                numbers=true;
                        }
                        li_gap.push(-tmp);
                }
                if (not_equal===false){
                        alert("if you choose similar gap penalty, choose single gap penalty");
                }
        }
        if (choice_gap_check===false){
                alert("choose single gap penalty or multiple gap penalty");
        }
        if (((nb=="single")&&(number===false))||((nb=="multiple")&&(numbers===false)
)) {
                alert("you have to enter a number");
        }
```

```
        if (((number===true)||(numbers===true))&&(choice_gap_check===true)) {
                check_gap=true;
        }
        return check_gap;
}

/**
 * [Function to initiate the process and get the final result]
 */
function init_final(){
        if (verif()===true){
        get_value();
        display();
        }
}
/**
 * [Function to initiate the step by step and go to the next step]
 */
function init_next(){
        if (verif()===true){
        get_value();
        next();
        }
}
/**
 * [Function to initiate the step by step and go to the previous step]
 */
function init_prev(){
        if (verif()===true){
        get_value();
        prev();
        }
}

/**
 * [Function to initiate the step by step and go to the next state]
 */
function init_fast(){
        if (verif()===true){
        get_value();
        fastnext();
        }
}
```

```
/*
* align2seq: Pairwise alignements algorithms in JavaScript, html5, and css3
* Copyright (C) 2015
*
* This file is part of align2seq.
*
* align2seq is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* align2seq is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with align2seq. If not, see <http://www.gnu.org/licenses/>
*
* Authors:
* Rudy Anne
* Aurelien Beliard
* Emeline Duquenne
* Aurore Perdriau
*/

/** Function to initiate the alignment according to Needleman and Wunsch algorit
hm
@constructor
 */
function needlemanwunsch()
{
        for (key in algorithm.prototype) {
                needlemanwunsch.prototype[key] = algorithm.prototype[key];
        }
}

/**
 * [ * Function to calculate the score according to Needleman and Wunsch algorit
hm]
 * @param  {[array]} matrix     [Substitution matrix chosen by the user]
 * @param  {[array]} matscore   [Score matrix filled by this function]
 * @param  {[array]} matpath    [Path matrix filled by the function]
 * @param  {[array]} matsumdia  [Sum matrix obtained by the diagonal case filled
 by the function]
 * @param  {[array]} matsumvert [Sum matrix obtained by the vertical case filled
 by the function]
 * @param  {[array]} matsumhor  [Sum matrix obtained by the horizontal case fill
ed by the function]
 * @param  {[array]} matsumtot  [Sum matrix obtained by the maximal value betwee
n the third previoux matrices]
 * @param  {[string]} l1         [The letter obtained by the first sequence used
 for comparison]
 * @param  {[string]} l2         [The letter obtained by the second sequence use
d for comparison]
 * @param  {[integer]} lengthseq  [Length of the sequence]
 * @param  {[integer]} place      [Place of the letter]
 * @param  {[integer]} gap        [Gap penality]
 * @param  {[integer]} gap2       [Gap penality]
 * @param  {[array]} letters    [Letters used in subsitution matrices]
 */
needlemanwunsch.prototype.score = function (matrix,matscore, matpath, matsumdia,
```



```
matsumvert, matsumhor,matsumtot,l1, l2, lengthseq, place,gap,gap2,letters) {
        var currentscore;
        var scorevert,scorehor,scoredia;
        var sumvert,sumdia,sumhor;
        var pos1,pos2;
        var placevert=place-(lengthseq+1);
        var placehor=place-1;
        var placedia=place-(lengthseq+2);
        if (place===0){
                matscore[place]=gap;
                matpath[place] = 0;
                scorevert = 0;
                scorehor = 0;
                scoredia = 0;
                matsumdia[place]=0;
                matsumvert[place]=0;
                matsumhor[place]=0;
                matsumtot[place]=gap;
        }
        else if(place<=lengthseq && place !== 0){
                matscore[place]=gap+matsumtot[place-1];
                matpath[place] = 1;
                scorevert = 0;
                scorehor = 0;
                scoredia = 0;
                matsumdia[place]=0;
                matsumvert[place]=0;
                matsumhor[place]=0;
                matsumtot[place]=gap+matsumtot[place-1];
        }
        else if (place%(lengthseq+1)===0 ){
                matscore[place]=gap2+matsumtot[place-(lengthseq + 1)];
                matpath[place] = 3;
                scorevert = 0;
                scorehor = 0;
                scoredia = 0;
                matsumdia[place]=0;
                matsumvert[place]=0;
                matsumhor[place]=0;
                matsumtot[place]=gap2+matsumtot[place-(lengthseq + 1)];
        }
        else{
                scorevert=matsumtot[placevert];
                scorehor=matsumtot[placehor];
                scoredia=matsumtot[placedia];
                for (var l in letters){
                        if (l1 === letters[l]){
                                pos1=parseInt(l,10);
                        }
                        if (l2 === letters[l]){
                                pos2=parseInt(l,10);
                        }
                }
                var lengthmat=letters.length;
                var posmatrix=(lengthmat*pos1)+pos2;
                currentscore=parseInt(matrix[posmatrix],10);
                matscore[place]=currentscore;
                sumdia=scoredia+currentscore;
                sumvert=scorevert+gap;
                sumhor=scorehor+gap2;
                matsumdia[place]=sumdia;
                matsumvert[place]=sumvert;
```

```
                matsumhor[place]=sumhor;
                var maxiscore=Math.max(sumvert,sumdia,sumhor);
                matsumtot[place]=maxiscore;
                if (maxiscore==(sumhor) && maxiscore!=(sumvert) && maxiscore!=(s
umdia)){
                        matpath[place]=1;
                }
                else if (maxiscore!=(sumhor) && maxiscore!=(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=2;
                }

                else if (maxiscore!=(sumhor) && maxiscore==(sumvert) && maxiscor
e!=(sumdia)){
                        matpath[place]=3;
                }
                else if (maxiscore==(sumhor) && maxiscore==(sumvert) && maxiscor
e!=(sumdia)){
                        matpath[place]=4;
                }
                else if (maxiscore==(sumhor) && maxiscore!=(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=5;
                }
                else if (maxiscore!=(sumhor) && maxiscore==(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=6;
                }
                else if (maxiscore==(sumhor) && maxiscore==(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=7;
                }
        }
}

/**
@constructor
 * [ Function to calculate the alignment according to Needleman and Wunsch algor
ithm
 * @param  {array} matpath   Path matrix filled by the function
 * @param  {type} matscore   Score matrix filled by this function
 * @param  {array} matsumtot Sum matrix obtained by the maximal value between th
e third previous matrix
 * @param  {array} s1        The first sequence
 * @param  {array} s2        The second sequence
 * @param  {integer} len1      Length of the first sequence
 * @param  {integer} lengthseq Length of the sequence
 */
needlemanwunsch.prototype.alignment = function (matpath, matscore, matsumtot, s1
, s2, len1, lengthseq) {
        var dep=(matsumtot.length)-1;
        var align1=[];
        var align2=[];
        var align1string="";
        var align2string="";
        var choice1,choice2,choice3;
        while (true) {
                var posseq1=(dep%(len1+1)-1);
                var posseq2=Math.floor(dep/(len1+1)-1);
                listalign.push(dep);
                        if (matsumtot[dep] === 0) {
                                listalign.pop();
```



```
                        break;
                }
                if (matpath[dep] === 1) {
                        dep = dep - 1;
                        align1.unshift(String(s1[posseq1]));
                        align2.unshift("-");
                }
                else if(matpath[dep] ===2){
                        dep = dep - (lengthseq + 2);
                        align1.unshift(String(s1[posseq1]));
                        align2.unshift(String(s2[posseq2]));
                }
                else if (matpath[dep] === 3) {
                        dep = dep - (lengthseq + 1);
                        align1.unshift("-");
                        align2.unshift(String(s2[posseq2]));
                }
                else if (matpath[dep] === 4){
                        choice1=matsumtot[dep-1];
                        choice2=matsumtot[dep - (lengthseq + 1)]
                        if (choice1>choice2){
                                dep = dep - 1;
                                align1.unshift(String(s1[posseq1]));
                                align2.unshift("-");
                        }
                        else {
                                dep = dep - (lengthseq + 1);
                                align1.unshift("-");
                                align2.unshift(String(s2[posseq2]));
                        }
                }
                else if (matpath[dep] === 5){
                        choice1=matsumtot[dep- 1];
                        choice2=matsumtot[dep - (lengthseq + 2)];
                        if (choice1>choice2){
                                dep = dep - 1;
                                align1.unshift(String(s1[posseq1]));
                                align2.unshift("-");

                        }
                        else{
                                dep = dep - (lengthseq + 2);
                                align1.unshift(String(s1[posseq1]));
                                align2.unshift(String(s2[posseq2]));

                        }
                }
                else if (matpath[dep] === 6){
                        choice1=matsumtot[dep - (lengthseq + 1)];
                        choice2=matsumtot[dep - (lengthseq + 2)];
                        if (choice1>choice2){
                                dep = dep - (lengthseq + 1);
                                align1.unshift("-")
                                align2.unshift(String(s2[posseq2]));
                        }
                        else{
                                dep = dep - (lengthseq + 2);
                                align1.unshift(String(s1[posseq1]));
                                align2.unshift(String(s2[posseq2]));
                        }
                }
                else if (matpath[dep] === 7){
```

```
                                choice1=matsumtot[dep- 1];
                                choice2=matsumtot[dep - (lengthseq + 2)];
                                choice3=matsumtot[dep - (lengthseq + 1)];
                                var maxchoice=Math.max(choice1,choice2,choice3);
                                if (maxchoice === choice2){
                                        dep = dep - (lengthseq + 2);
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift(String(s2[posseq2]));
                                }
                                else if (maxchoice === choice3){
                                        dep = dep - (lengthseq + 1);
                                        align1.unshift("-")
                                        align2.unshift(String(s2[posseq2]));
                                }
                                else{
                                        dep = dep - 1;
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift("-");
                                }
                        }
                }
                for(var el1 in align1){
                        align1string=align1string.concat(align1[el1]);
                }
                for(var el2 in align2){
                        align2string=align2string.concat(align2[el2]);
                }
        var result=[align1string,align2string];
        return result;
}
```

```
/*
 *  align2seq: Pairwise alignements algorithms in JavaScript, html5, and css3
 *  Copyright (C) 2015
 *
 *  This file is part of align2seq.
 *
 *  align2seq is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  align2seq is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with align2seq.  If not, see <http://www.gnu.org/licenses/>
 *
 * Authors:
 * Rudy Anne
 * Aurelien Beliard
 * Emeline Duquenne
 * Aurore Perdriau
 */

/** Function to initiate the alignment according to Smith and Waterman algorithm
@constructor
 */
 function smithwaterman()
 {
        for (key in algorithm.prototype) {
                smithwaterman.prototype[key] = algorithm.prototype[key];
        }
 }
/**
  * [Function to calculate the score according to Smith and Waterman algorithm
 * @param  {array]} matrix     Substitution matrix chosen by the user
 * @param  {array]} matscore   Score matrix filled by this function
 * @param  {array]} matpath    Path matrix filled by the function
 * @param  {array]} matsumdia  Sum matrix obtained by the diagonal case filled b
y the function
 * @param  {array]} matsumvert Sum matrix obtained by the vertical case filled b
y the function
 * @param  {array]} matsumhor  Sum matrix obtained by the horizontal case filled
 by the function
 * @param  {array]} matsumtot  Sum matrix obtained by the maximal value between
the third previoux matrices
 * @param  {integer} l1         The letter obtained by the first sequence used f
or comparison
 * @param  {integer} l2         The letter obtained by the second sequence used
for comparison
 * @param  {integer} lengthseq   max Length of the 2 sequence
 * @param  {integer} place      Place of the letter
 * @param  {integer} gap        Gap penality
 * @param  {integer} gap2       Gap penality
 * @param  {char} letters      Letters used in subsitution matrices
 */
 smithwaterman.prototype.score = function (matrix,matscore, matpath, matsumdia,
matsumvert, matsumhor, matsumtot,l1, l2, lengthseq, place,gap,gap2,letters) {
        var currentscore;
        var scorevert, scorehor, scoredia;
```



```
        var sumvert, sumhor, sumdia;
        var pos1,pos2;
        var placevert = place - (lengthseq + 1);
        var placehor = place - 1;
        var placedia = place - (lengthseq + 2);
        if (place===0){
                matscore[place]=0;
                matpath[place] = 0;
                scorevert = 0;
                scorehor = 0;
                scoredia = 0;
                matsumdia[place]=0;
                matsumvert[place]=0;
                matsumhor[place]=0;
                matsumtot[place]=0;
        }
        else if(place<=lengthseq && place !== 0){
                matscore[place]=0;
                matpath[place] = 1;
                scorevert = 0;
                scorehor = 0;
                scoredia = 0;
                matsumdia[place]=0;
                matsumvert[place]=0;
                matsumhor[place]=0;
                matsumtot[place]=0;

        }
        else if (place%(lengthseq+1)===0 ){
                matscore[place]=0;
                matpath[place] = 3;
                scorevert = 0;
                scorehor = 0;
                scoredia = 0;
                matsumdia[place]=0;
                matsumvert[place]=0;
                matsumhor[place]=0;
                matsumtot[place]=0;
        }
        else{
                scorevert=matsumtot[placevert];
                scorehor=matsumtot[placehor];
                scoredia=matsumtot[placedia];
                for (var l in letters){
                        if (l1 === letters[l]){
                                pos1=parseInt(l,10);
                        }
                        if (l2 === letters[l]){
                                pos2=parseInt(l,10);
                        }
                }
                var lengthmat=letters.length;
                var posmatrix=(lengthmat*pos1)+pos2;
                currentscore=parseInt(matrix[posmatrix],10);
                matscore[place]=currentscore;
                sumdia=scoredia+currentscore;
                sumvert=scorevert+gap;
                sumhor=scorehor+gap2;
                matsumdia[place]=sumdia;
                matsumvert[place]=sumvert;
                matsumhor[place]=sumhor;
                var maxiscore=Math.max(sumvert,sumdia,sumhor);
                matsumtot[place]=maxiscore;
```

```
                if (maxiscore==(sumhor) && maxiscore!=(sumvert) && maxiscore!=(s
umdia)){
                        matpath[place]=1;
                }
                else if (maxiscore!=(sumhor) && maxiscore!=(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=2;
                }

                else if (maxiscore!=(sumhor) && maxiscore==(sumvert) && maxiscor
e!=(sumdia)){
                        matpath[place]=3;
                }
                else if (maxiscore==(sumhor) && maxiscore==(sumvert) && maxiscor
e!=(sumdia)){
                        matpath[place]=4;
                }
                else if (maxiscore==(sumhor) && maxiscore!=(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=5;
                }
                else if (maxiscore!=(sumhor) && maxiscore==(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=6;
                }
                else if (maxiscore==(sumhor) && maxiscore==(sumvert) && maxiscor
e==(sumdia)){
                        matpath[place]=7;
                }
                if (maxiscore<0){
                        matsumtot[place]=0;
                }
        }
 }

/**
 * Function to calculate the alignment according to Needleman and Wunsch algorit
hm
 * @param  {array} matpath   path matrix filled  by the function
 * @param  {type} matscore   Score matrix filled by this function
 * @param  {type} matsumtot sum matrix filled by the function
 * @param  {type} s1        The first sequence
 * @param  {type} s2        The second sequence
 * @param  {type} len1      Length of the first sequence
 * @param  {type} lengthseq max Length of the 2 sequence
 */
 smithwaterman.prototype.alignment = function (matpath, matscore, matsumtot, s1,
 s2, len1, lengthseq) {
        var val, elem, dep, valmaxpos;
        var valmax = 0;
        var l = [];
        var choice1,choice2,choice3;
        var result = [];
        for (val in matsumtot) {
                valmaxpos = matsumtot[val];
                valmax = Math.max(valmax, valmaxpos);
        }

        for (elem in matsumtot) {
                if (matsumtot[elem] === valmax) {
                        l.push(elem);
                }
```



```
        }

        for (dep in l) {
                var align1string="";
                var align2string="";
                deppos = l[dep];
                var align1 = [];
                var align2 = [];
                while (true) {
                        var posseq1=(deppos%(len1+1)-1);
                        var posseq2=Math.floor(deppos/(len1+1)-1);
                        listalign.push(deppos);
                        if (matsumtot[deppos] === 0) {
                                listalign.pop();
                                break;
                        }
                        if (matpath[deppos] === 1) {
                                deppos = deppos - 1;
                                align1.unshift(String(s1[posseq1]));
                                align2.unshift("-");
                        }
                        else if(matpath[deppos] ===2){
                                deppos = deppos - (lengthseq + 2);
                                align1.unshift(String(s1[posseq1]));
                                align2.unshift(String(s2[posseq2]));
                        }
                        else if (matpath[deppos] === 3) {
                                deppos = deppos - (lengthseq + 1);
                                align1.unshift("-");
                                align2.unshift(String(s2[posseq2]));
                        }
                        else if (matpath[deppos] === 4){
                                choice1=matsumtot[deppos-1];
                                choice2=matsumtot[deppos - (lengthseq + 1)]
                                if (choice1>choice2){
                                        deppos = deppos - 1;
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift("-");
                                }
                                else {
                                        deppos = deppos - (lengthseq + 1);
                                        align1.unshift("-");
                                        align2.unshift(String(s2[posseq2]));
                                }
                        }
                        else if (matpath[deppos] === 5){
                                choice1=matsumtot[deppos- 1];
                                choice2=matsumtot[deppos - (lengthseq + 2)];
                                if (choice1>choice2){
                                        deppos = deppos - 1;
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift("-");

                                }
                                else{
                                        deppos = deppos - (lengthseq + 2);
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift(String(s2[posseq2]));

                                }
                        }
                        else if (matpath[deppos] === 6){
```

```
                                choice1=matsumtot[deppos - (lengthseq + 1)];
                                choice2=matsumtot[deppos - (lengthseq + 2)];
                                if (choice1>choice2){
                                        deppos = deppos - (lengthseq + 1);
                                        align1.unshift("-")
                                        align2.unshift(String(s2[posseq2]));
                                }
                                else{
                                        deppos = deppos - (lengthseq + 2);
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift(String(s2[posseq2]));
                                }
                        }
                        else if (matpath[deppos] === 7){
                                choice1=matsumtot[deppos- 1];
                                choice2=matsumtot[deppos - (lengthseq + 2)];
                                choice3=matsumtot[deppos - (lengthseq + 1)];
                                var maxchoice=Math.max(choice1,choice2,choice3);
                                if (maxchoice === choice2){
                                        deppos = deppos - (lengthseq + 2);
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift(String(s2[posseq2]));
                                }
                                else if (maxchoice === choice3){
                                        deppos = deppos - (lengthseq + 1);
                                        align1.unshift("-")
                                        align2.unshift(String(s2[posseq2]));
                                }
                                else{
                                        deppos = deppos - 1;
                                        align1.unshift(String(s1[posseq1]));
                                        align2.unshift("-");
                                }
                        }
                }
                for(var el1 in align1){
                        align1string=align1string.concat(align1[el1]);
                }
                for(var el2 in align2){
                        align2string=align2string.concat(align2[el2]);
                }
                result.push(align1string);
                result.push(align2string);
        }
        return result;
}
```

```
/*
 *  align2seq: Pairwise alignements algorithms in JavaScript, html5, and css3
 *  Copyright (C) 2015
 *
 *  This file is part of align2seq.
 *
 *  align2seq is free software: you can redistribute it and/or modify

 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  align2seq is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with align2seq.  If not, see <http://www.gnu.org/licenses/>
 *
 * Authors:
 * Rudy Anne
 * Aurelien Beliard
 * Emeline Duquenne
 * Aurore Perdriau
 */

"use strict"

var nbValuesToDisplay = 0;
var nbValuesAlignToDisplay = 0;
var nbAlign;
var title;

/**
 *[First function executed after data treatment in case of step by step]
 *
 */

function next(){
        nbValuesToDisplay++;
        //Limit check value
        if(nbValuesToDisplay>=matscore.length){
                nbValuesToDisplay=matscore.length;
                nbValuesAlignToDisplay++;
                if (nbValuesAlignToDisplay>=listalign.length){
                        nbValuesAlignToDisplay=listalign.length;
                }
        }
        launch_nstep(nbValuesToDisplay);
        launch_nstep_align(nbValuesAlignToDisplay);
}
/**
 * [this function allow to return to the previous step of the step by step]
 *  */
function prev(){
        nbValuesAlignToDisplay--;
        if(nbValuesAlignToDisplay<0){
                nbValuesAlignToDisplay=0
                nbValuesToDisplay--;
        }
        launch_nstep(nbValuesToDisplay);
```



```
        launch_nstep_align(nbValuesAlignToDisplay);
}
/**
 * [this function allow to return to the next state]
 *
 */
function fastnext(){
        if(nbValuesToDisplay<=matscore.length){
                nbValuesToDisplay=matscore.length
        }
        if (nbValuesAlignToDisplay<=listalign.length && nbValuesAlignToDisplay!=
=0){
                nbValuesAlignToDisplay=listalign.length;
        }
        launch_nstep(nbValuesToDisplay);
        launch_nstep_align(nbValuesAlignToDisplay);
        nbValuesAlignToDisplay++
}
/**
 * [this function allow to return to the previous state]
  */
function fastpreview(){
        if (nbValuesAlignToDisplay!==0){
                nbValuesAlignToDisplay=0
                nbValuesToDisplay=matscore.length;
        }
        else if (nbValuesToDisplay<=matscore.length){
                nbValuesAlignToDisplay=0
                nbValuesToDisplay=0
        }
        launch_nstep(nbValuesToDisplay);
        launch_nstep_align(nbValuesAlignToDisplay);

}
/** Step by step function with next and preview possibilities
@param {[number]} nbValuesToDisplay - counter for the scoring matrix
*/


function launch_nstep(nbValuesToDisplay){
        var matrixs=document.getElementById("matrixtime");

        //The table is empty
        while (matrixs.firstChild) {
        matrixs.removeChild(matrixs.firstChild);
        }

        // Filling the array with the desired number of cells
        for (var i =0;i<=(size2-1);i++){
                matrixs.insertRow(i);
                for(var j=0;j<=(size1);j++){
                        matrixs.rows[i].insertCell(j);
                }
        }

        for(var i=0;i<matrixs.rows.length;i++){
                var currentRow = matrixs.rows[i];
                for(var j=0;j<currentRow.cells.length;j++){
                        var currentCell=currentRow.cells[j];

                        //Filling the array with the first sequence (first colum
n)
```

```
                        if (i>=2 && j===0){
                                currentCell.innerHTML=s2[i-2];
                        }

                        //Filling the array with the second sequence (first lign
e)
                        if (i===0 && j>=2){
                                currentCell.innerHTML=s1[j-2];
                        }
                }
        }

        var nbDisplayedValues= 0;
        for(var i=1;i<matrixs.rows.length;i++){
                var currentRow = matrixs.rows[i];
                for(var j=1;j<currentRow.cells.length;j++){
                        var currentCell=currentRow.cells[j];
                        if ((nbDisplayedValues>=1) && (nbDisplayedValues<nbValue
sToDisplay)){
                                var i2,j2;
                                if (j==1){
                                        i2=i-1;
                                        j2=(currentRow.cells.length)-1
                                }
                                else{
                                        i2=i;
                                        j2=j-1;
                                }
                                var previousCell=matrixs.rows[i2].cells[j2]
                                var cellprevious=nbDisplayedValues-1;
                                previousCell.innerHTML=matpatharrows[cellpreviou
s];
                                previousCell.innerHTML+=matsumtot[cellprevious];
                        }
                        //The table is filled with the assumption that it is fil
led from left to right
                        currentCell.innerHTML=matscore[nbDisplayedValues];
                        nbDisplayedValues++;

                        if (nbDisplayedValues>nbValuesToDisplay) {
                                currentCell.style.visibility="hidden";
                        }

                        if (nbDisplayedValues==matscore.length){
                                currentCell.innerHTML=matpatharrows[(matscore.le
ngth)-1]+" "+matsumtot[(matscore.length)-1];
                        }
                }
        }
    title=document.getElementById("matrixtime").createCaption();
        title.innerHTML="<b>Sum matrix</b>";
        if(nbValuesToDisplay>size1) {
                if ((nbValuesToDisplay-1)%size1!==0){
                        var cellvert=size1;
                        var celldia=size1+1;
                        var cellcurrent=nbValuesToDisplay-1;
                        var posj=(nbValuesToDisplay-1)%(len1+1);
                        var posi=Math.floor((nbValuesToDisplay-1)/(len1+1));
                        explain.innerHTML="Value of M("+posi+","+posj+") = maximal value
between : <br>";
                        explain.innerHTML+="M("+(posi-1)+","+(posj-1)+") + S("+pos
i+","+posj+") = "+matsumtot[cellcurrent-celldia]+" + "+matscore[cellcurrent]+" = " +
```



```
"<b>"+matsumdia[cellcurrent]+"</b>"+"<br>";
                        explain.innerHTML+="M("+(posi)+","+(posj-1)+") + gap = "+mat
sumtot[cellcurrent-1]+"+"+gap2[posi]+" = "+"<b>"+matsumhor[cellcurrent]+"</b>"+"<br
>";
                        explain.innerHTML+="M("+(posi-1)+","+(posj)+") + gap = "+mat
sumtot[cellcurrent-cellvert]+"+"+gap[posj]+"= "+"<b>"+matsumvert[cellcurrent]+"</
b>"+"<br>";
                        explain.innerHTML+="Maximum value of the three : <b>"+matsumtot[c
ellcurrent]+"</b><br>";
                        explain.innerHTML+="Corresponding path : "+matpatharrows[cellc
urrent]+"<br>";
                }
                else{
                        explain.innerHTML="";
                }
        }
}
/**
 * display the alignment step by step
 * @param  {integer} nbValuesToDisplayAlign number of value in the alignment
 *
 */
function launch_nstep_align(nbValuesToDisplayAlign){
        if (nbValuesAlignToDisplay>=1){
                title.innerHTML="<b>Alignment matrix</b>";
                explain.innerHTML=" ";
        }

        var matrixs=document.getElementById("matrixtime")
        var nbDisplayedValuesAlign= 0;
        for(var i=1;i<matrixs.rows.length;i++){
                var currentRow = matrixs.rows[i];
                for(var j=1;j<currentRow.cells.length;j++){
                        var currentCell=currentRow.cells[j];
                }
        }
        for(var posalign=1;posalign<=nbValuesToDisplayAlign;posalign++){
                var alignpos=listalign[posalign-1];
                console.log(alignpos);
                if (alignpos >= listalign[posalign-2]){
                        launch_nstep(matscore.length)
                }
                var posj=(alignpos%(len1+1)-1)+2;
                var posi=Math.floor((alignpos/(len1+1)-1)+2);
                var alignCell=matrixs.rows[posi].cells[posj];
                alignCell.innerHTML=matpatharrowsalign[alignpos];
        }
}
```