République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Informatique
Spécialité :
Bioinformatique

Thème :
Alignement de séquences

Encadré par :
BOUKHEDOUMA Saida

Présenté par :
TALEB Youcef
AMARI Lokmane

Soutenu le :
21/05/2025

Introduction

La bio-informatique est une discipline interdisciplinaire qui combine les sciences biologiques, les mathématiques et l'informatique afin d'analyser et interpréter les données biologiques. L'une des tâches fondamentales en bio-informatique est l'**alignement de séquences**, qui permet de comparer des séquences d'ADN, d'ARN ou de protéines afin d'identifier des similarités, souvent porteuses d'informations fonctionnelles ou évolutives.

Ce mini-projet a pour objectif principal la compréhension et la mise en œuvre de différentes méthodes d'alignement de séquences biologiques. Nous étudierons d'abord l'**alignement par paire** en mettant en œuvre deux algorithmes classiques : ***Needleman-Wunsch*** pour l'alignement global, et ***Smith-Waterman*** pour l'alignement local. Nous analyserons leur performance et leur complexité expérimentale à travers plusieurs tests.

Nous aborderons ensuite l'**alignement multiple** de séquences, qui permet de comparer plus de deux séquences simultanément. Deux approches seront étudiées : une approche progressive (basée sur la notion de profil), et une autre selon un paradigme différent. Nous comparerons les performances de ces deux approches sur la base du score d'alignement et du temps d'exécution.

Ce travail nous permettra de mieux comprendre les limites et les défis liés à l'alignement de séquences, tout en consolidant nos compétences en algorithmique et en programmation appliquées à la bio-informatique.

1 Notions de base

1.1 Séquences biologiques

Une **séquence biologique** est une suite linéaire d'éléments appelés nucléotides (pour l'ADN et l'ARN) ou acides aminés (pour les protéines). Ces séquences codent pour l'information génétique et sont essentielles dans l'étude du vivant.

- **ADN (Acide Désoxyribonucléique)** : composé de quatre bases : A (adénine), T (thymine), C (cytosine), G (guanine).
- **ARN (Acide Ribonucléique)** : similaire à l'ADN mais avec U (uracile) à la place de T.
- **Protéines** : séquences d'acides aminés, souvent représentées par une chaîne de lettres utilisant l'alphabet à 20 caractères (un pour chaque acide aminé).

1.2 Alignement de séquences

L'**alignement de séquences** vise à comparer deux ou plusieurs séquences pour en identifier les similarités. Ces similarités peuvent refléter des relations fonctionnelles, structurelles ou évolutives.

On distingue :

- **Alignement global** : alignement sur toute la longueur des séquences (Needleman-Wunsch).

- **Alignement local** : alignement de sous-parties optimales (Smith-Waterman).
- **Alignement multiple** : alignement simultané de plusieurs séquences.

1.3 Matrices de similarité

Les **matrices de similarité** (comme *BLOSUM62* pour les protéines) permettent de quantifier la ressemblance entre caractères. À chaque paire de symboles, la matrice associe un score basé sur des observations biologiques.

- Un score élevé signifie une substitution fréquente ou favorable.
- Un score négatif signifie une substitution peu probable ou défavorable.

1.4 Pénalité de gap et principe de scoring

Lors de l'alignement de séquences, il est fréquent d'introduire des **gaps** (ou *indels*) afin d'optimiser la correspondance entre les éléments des séquences. Chaque gap est associé à une **pénalité**, généralement un score négatif, qui diminue la qualité globale de l'alignement. Dans ce projet, une pénalité fixe de **-2** est appliquée à chaque insertion ou suppression.

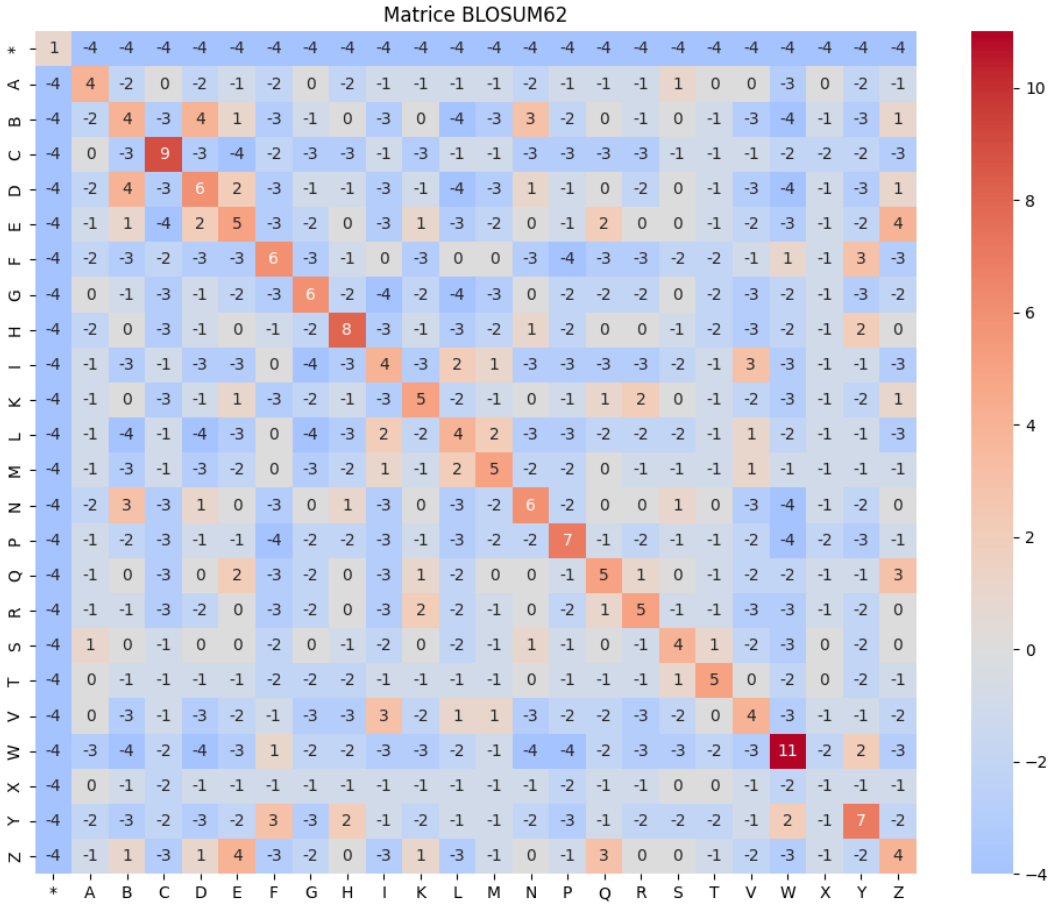
Le **score d'un alignement** est la somme :

- des scores de correspondance/mismatch obtenus depuis une matrice de similarité (par exemple BLOSUM62),
- des pénalités de gaps.

Un score élevé indique un alignement de meilleure qualité, reflétant un degré de similarité plus élevé entre les séquences comparées.

Dans ce projet on va utiliser la matrice BLOSUM62 et une penalite de gap (**-2**)

1.5 Matrice BLOSUM62



Visualisation de la matrice BLOSUM62 utilisée pour les alignements

2 Alignement par paire de séquences

2.1 Motivations et cas d'applications

L'alignement par paire de séquences est une tâche fondamentale en bio-informatique. Elle permet de comparer deux séquences (ADN, ARN ou protéines) afin d'identifier des similarités significatives qui peuvent révéler une relation fonctionnelle, structurale ou évolutive.

Voici quelques cas d'applications :

- **Recherche de gènes homologues** entre espèces différentes.
- **Détection de mutations** dans les séquences d'ADN, utile en médecine personnalisée.
- **Prédiction de la fonction** d'une protéine en la comparant à une base de données de protéines connues.
- **Alignement de séquences expérimentales** avec des séquences de référence (génomique comparative).

L'alignement par paires est également une étape préliminaire importante dans des tâches plus complexes comme l'alignement multiple ou la construction d'arbres phylogénétiques.

2.2 Exemples d'alignement deux à deux

Voici deux exemples concrets d'alignement entre deux séquences de taille réduite, l'un en alignement global et l'autre en alignement local.

Alignement global (Needleman-Wunsch)

Séquences utilisées :

- `Seq1` : ACGT
- `Seq2` : AGT

Résultat de l'alignement :

```
A C G T
A - G T
```

Score total : `4` (calculé à partir de BLOSUM62 et pénalité de gap -2)

Alignement local (Smith-Waterman)

Séquences utilisées :

- `Seq1` : TCGTAG
- `Seq2` : GTAG

Résultat de l'alignement local :

```
G T A G
G T A G
```

Score local maximal : `8`

2.3 Algorithmes d'alignement

2.3.1 Algorithme de Needleman-Wunsch (Alignement global)

L'algorithme de **Needleman-Wunsch** permet d'effectuer un alignement **global** entre deux séquences. Il est basé sur la programmation dynamique, où une matrice est remplie de manière à maximiser le score d'alignement global.

Étapes principales :

1. Initialisation de la première ligne et de la première colonne avec des pénalités de gap.
2. Remplissage de la matrice en choisissant à chaque case le maximum entre :

- un match/mismatch (diagonale),
- un gap horizontal (gauche),
- un gap vertical (haut).

3. Reconstruction de l'alignement optimal en effectuant un *backtracking* depuis la dernière case.

Cet algorithme garantit l'obtention d'un alignement optimal sur l'intégralité des deux séquences.

2.3.2 Algorithme de Smith-Waterman (Alignement local)

L'algorithme de **Smith-Waterman** réalise un alignement **local**, c'est-à-dire qu'il identifie les sous-séquences les plus similaires à l'intérieur de deux séquences complètes. Il est également basé sur la programmation dynamique.

Étapes principales :

1. Initialisation de la matrice avec des zéros.
2. Remplissage de la matrice avec la formule de récurrence, en imposant que les scores négatifs deviennent 0 (aucune pénalité de début d'alignement).
3. L'alignement optimal est le chemin de score maximal dans la matrice, suivi d'un backtracking jusqu'à une case de valeur 0.

L'alignement local est utile pour détecter des régions similaires dans des séquences très différentes.

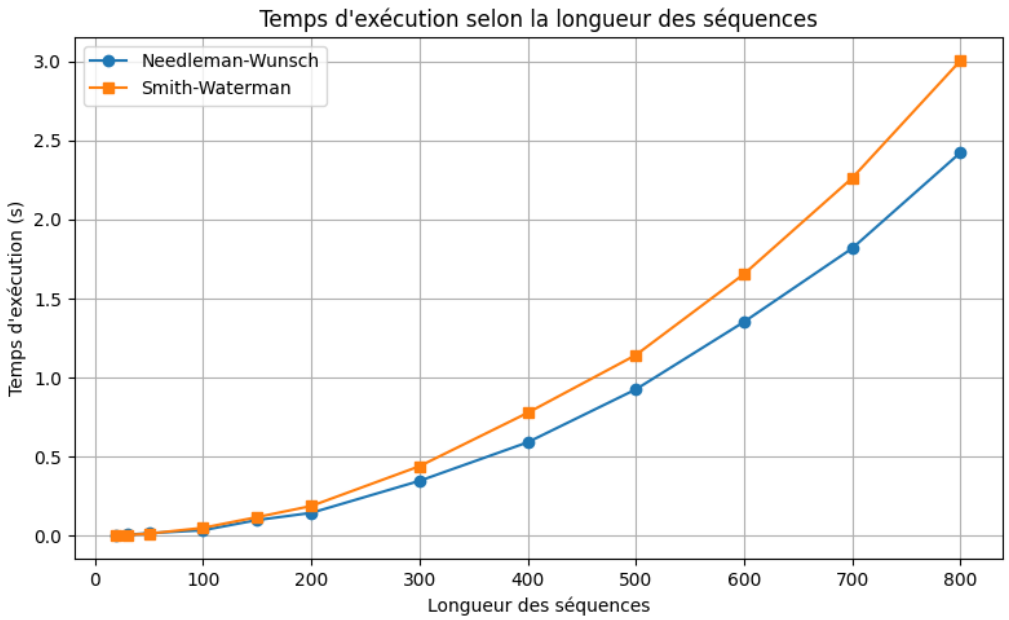
2.4 Résultats expérimentaux

Cette section présente les résultats expérimentaux obtenus pour les algorithmes de Needleman-Wunsch (alignement global) et Smith-Waterman (alignement local). Nous incluons d'abord quelques exemples d'alignements de séquences sous forme de captures d'écran, suivis d'un tableau récapitulatif des scores et temps d'exécution obtenus pour des séquences de longueurs croissantes. Enfin, deux graphiques illustrent l'évolution du temps d'exécution et du score d'alignement en fonction de la longueur des séquences.

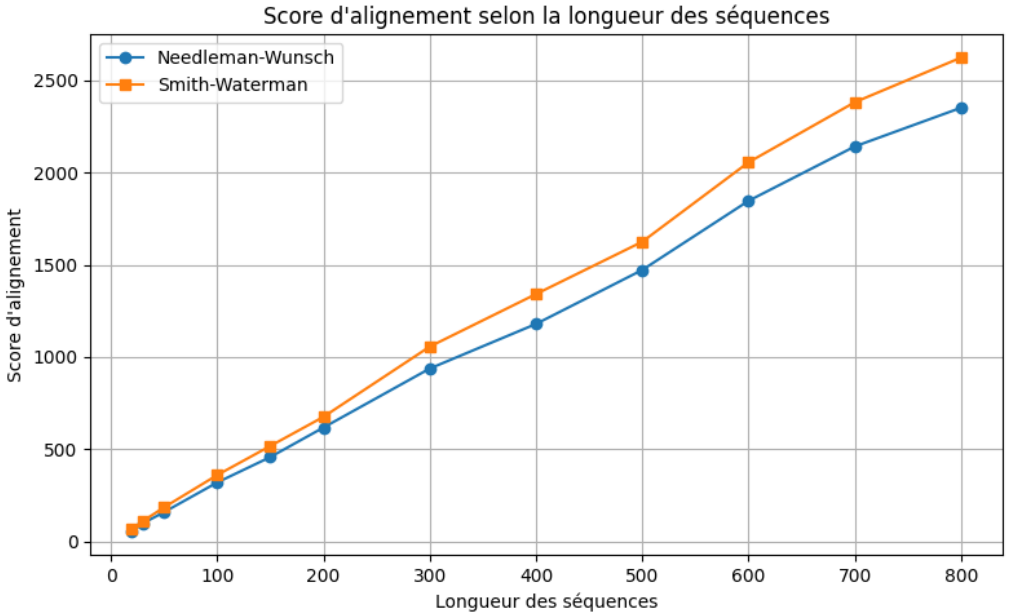
```
Les sequances:
ACTTGCCGTC
ACGTAGGATTTGGGGA

[LOCAL] Smith-Waterman
Score : 27.0
AC-TTGCCG-T
ACGTAG--GAT

[GLOBAL] Needleman-Wunsch
Score : 15
AC-TT-GC--C--GTC
ACGTAGGATTTGGGGA
########################################
Les sequances:
GACTAATGTGTATACGGGGAGCG
TTAATCGTCCTGGAAAGCT

[LOCAL] Smith-Waterman
Score : 60.0
TAATGTGTATAC-GGGGAGC
TAAT-CG--TCCTGGAAAGC

[GLOBAL] Needleman-Wunsch
Score : 54
GACTAATGTGTATAC-GGGGAGCG
-T-TAAT-CG--TCCTGGAAAGCT
########################################
Les sequances:
TTAATCGTCCTGGAAAGCT
CTGTGCTGATGATAAAGTGGTCATCGGACGAGA

[LOCAL] Smith-Waterman
Score : 60.0
TTAA-TCGTC--CTGGA-AAG
TAAAGTGGTCATC-GGACGAG

[GLOBAL] Needleman-Wunsch
Score : 42
---T--T-A--AT--C---GTC--CTGGA-AAGCT
CTGTGCTGATGATAAAGTGGTCATC-GGACGAG-A
########################################
```

```
les sequences de longueurs 20:
        TCGTCCTTCAACGGCTGATA
        TCCCCTAGTGGTGGTAAGAA
Needleman-Wunsch: (temps: 0.0079, score: 56)
        TCGTCCTTC-AACGGCTGAT----A
        TC--CC--CTAGTGG-TGGTAAGAA
Smith-Waterman: (temps: 0.0108, score: 66.0)
        TCGTCCTTC-AACGGCTGAT
        TC--CC--CTAGTGG-TGGT
########################################
les sequences de longueurs 30:
        CCGGGAATTATGGATCTCAAACGCTTTGTC
        ACAACCGTTTTGGACTTCCCAGTGCCATAC
Needleman-Wunsch: (temps: 0.0212, score: 86)
        -C--CGGGAATTATGGA--TCTCAAACGCTTTGTC
        ACAAC-CG--TTTTGGACTTC-CCAGTGCCAT-AC
Smith-Waterman: (temps: 0.0120, score: 97.0)
        C--CGGGAATTATGGA--TCTCAAACGCTTTGTC
        CAAC-CG--TTTTGGACTTC-CCAGTGCCAT-AC
########################################
les sequences de longueurs 50:
        TACTCTCTCTATAGTGGCGACGCAGCGGGAGACCTGGCCTAGCCCTGGGG
        CAAGATGTCACCTTAGAATTCATAAGAGCCATCAGTATGGCAGTAGCGCA
Needleman-Wunsch: (temps: 0.0360, score: 125)
        -TA-C--TCTCTCTATAG--TGGC----GA-CGCAGCGGGAGACCTGGCC-TAGCCCTGGGG
        CAAGATGTCAC-CT-TAGAAT-TCATAAGAGC-CATC---AGT-ATGGCAGTAGCGC----A
Smith-Waterman: (temps: 0.0438, score: 155.0)
        A-C--TCTCTCTATAG--TGGC----GA-CGCAGCGGGAGACCTGGCC-TAGC-C
        AGATGTCAC-CT-TAGAAT-TCATAAGAGC-CATC---AGT-ATGGCAGTAGCGC
########################################
les sequences de longueurs 100:
        GGAGGATGTTGCCCCTACTATGTAGGTAAGACCCCAACATGGGGGCTGAGTTAAGCCCAAGCCGGCAATCACGTGCGTGGTAGAACCAGTGCTGGTTCCC
        GACTTGAAGTGTTCTGCCCAGCTGCGAATTGGAATACCACATTTGCGTCCTTCAAGACCGCTGCAATCGGGTTGCGGGTTCCTTCTTCATGGAGCCAAAG
Needleman-Wunsch: (temps: 0.1449, score: 326)
        G----GAGGATG-T-TGCCC--CTAC-TATGTAGGTAAGACCCCAACATGGGGGCTGAGTTAAGCC--CAAG-CCG--GCAATCACG-TGCGTGGTAGAACCAG-TGC-TGGTTCC--C-
        GACTTGAAG-TGTTCTGCCCAGCTGCGAAT-T-GG-AATA--CC-ACAT--TTGC---G-T---CCTTCAAGACCGCTGCAATCGGGTTGCG-GGT---TCCTTCTTCATGGAGCCAAAG
Smith-Waterman: (temps: 0.2567, score: 369.0)
        G----GAGGATG-T-TGCCC--CTAC-TATGTAGGTAAGACCCCAACATGGGGGCTGAGTTAAGCC--CAAG-CCG--GCAATCACG-TGCGTGGTAGAACCAG-TGC-TGGTTCC
        GACTTGAAG-TGTTCTGCCCAGCTGCGAAT-T-GG-AATA--CC-ACAT--TTGC---G-T---CCTTCAAGACCGCTGCAATCGGGTTGCG-GGT---TCCTTCTTCATGGAGCC
```

Algorithme	**Longueur**	**Score**	**Temps (s)**
Needleman-Wunsch	20	54	0.0041
Smith-Waterman	20	64	0.0042
Needleman-Wunsch	30	97	0.0073
Smith-Waterman	30	109	0.0060
Needleman-Wunsch	50	158	0.0168
Smith-Waterman	50	184	0.0141
Needleman-Wunsch	100	319	0.0355
Smith-Waterman	100	359	0.0507
Needleman-Wunsch	150	457	0.1000
Smith-Waterman	150	517	0.1185
Needleman-Wunsch	200	617	0.1453
Smith-Waterman	200	675	0.1887
Needleman-Wunsch	300	937	0.3476
Smith-Waterman	300	1056	0.4403
Needleman-Wunsch	400	1179	0.5915
Smith-Waterman	400	1341	0.7780
Needleman-Wunsch	500	1472	0.9263
Smith-Waterman	500	1625	1.1436
Needleman-Wunsch	600	1847	1.3532
Smith-Waterman	600	2056	1.6559
Needleman-Wunsch	700	2142	1.8183
Smith-Waterman	700	2382	2.2630
Needleman-Wunsch	800	2352	2.4236
Smith-Waterman	800	2624	3.0048



Temps d'exécution des algorithmes selon la longueur des séquences



Score d'alignement des algorithmes selon la longueur des séquences

2.5 Limites des algorithmes et pistes d'amélioration

Les algorithmes de Needleman-Wunsch et Smith-Waterman, bien qu'efficaces pour des séquences de taille modérée, présentent certaines limites lorsqu'on les applique à de très longues séquences.

L'une des principales limites réside dans leur complexité temporelle et spatiale de $O(n \times m)$, où n et m sont les longueurs des séquences. Cela signifie que le temps d'exécution et la mémoire nécessaire augmentent rapidement avec la taille des données, comme le confirment nos résultats expérimentaux.

Pour y remédier, plusieurs optimisations sont envisageables :

- Utiliser des variantes en espace réduit, comme l'algorithme de Hirschberg pour Needleman-Wunsch, qui réduit la complexité mémoire à $O(n + m)$.
- Appliquer des heuristiques telles que BLAST pour éviter de parcourir toute la matrice, ce qui permet de traiter des séquences très longues avec des résultats approximatifs mais rapides.
- Paralléliser le calcul de la matrice avec des GPU ou des architectures multi-cœurs pour améliorer les performances.

En résumé, bien que ces algorithmes soient fiables pour l'alignement optimal, ils deviennent coûteux pour des séquences longues, et des approches approximatives ou distribuées peuvent alors s'avérer plus adaptées.

3 Alignement multiple de séquences

L'alignement multiple de séquences (ou MSA pour *Multiple Sequence Alignment*) est une extension de l'alignement par paire. Il consiste à aligner simultanément trois séquences ou plus afin d'identifier des régions communes, des motifs conservés, ou des similarités évolutives.

Ce type d'alignement est essentiel dans plusieurs domaines de la bio-informatique, notamment :

- L'identification de motifs conservés au sein d'une famille de gènes ou de protéines,
- La prédiction de structures secondaires ou tertiaires,
- La construction d'arbres phylogénétiques,
- L'annotation fonctionnelle de nouvelles séquences.

Contrairement à l'alignement par paire, l'alignement multiple présente une complexité computationnelle bien plus importante, car le nombre de combinaisons possibles croît exponentiellement avec le nombre de séquences. Il est donc souvent abordé avec des heuristiques ou des méthodes approximatives.

Dans cette section, nous allons :

1. Montrer des exemples d'alignement multiple,
2. Expliquer différentes approches (exacte, progressive, itérative),
3. Implémenter une méthode progressive,
4. Puis en comparer une seconde (approche alternative),
5. Et analyser leurs performances.

3.1 Exemple d'alignement multiple

Voici un exemple d'alignement multiple de quatre séquences ADN :

```
Seq1 : A T G C T A - A G C T
Seq2 : A T G C T A A A G C T
Seq3 : A T G - T A A A G C T
Seq4 : A T G C T A A - G C T
```

Dans cet alignement, on peut observer des positions conservées communes (par exemple les positions 1, 2, 3, 5, 6, 9, 10, 11) ainsi que des insertions ou décalages (gaps) dans certaines séquences.

L'objectif est de maximiser l'alignement global de toutes les séquences, en insérant des gaps là où cela améliore la cohérence générale.

3.2 Approches d'alignement multiple

L'alignement multiple de séquences est un problème plus complexe que l'alignement par paire. Il ne s'agit plus de comparer deux séquences à la fois, mais de trouver un alignement commun à un ensemble de k séquences de taille L, de manière à maximiser un score global ou une cohérence de conservation. Ce problème est fondamental en bio-informatique, mais il est également connu pour être difficile à résoudre de manière exacte à grande échelle.

Trois grandes approches sont utilisées pour aborder ce problème : l'approche exacte, l'approche progressive, et l'approche itérative.

3.2.1 Approche exacte (programmation dynamique multidimensionnelle)

L'approche exacte repose sur une généralisation de la programmation dynamique classique à plus de deux dimensions. L'idée est de construire une matrice k-dimensionnelle, où chaque dimension correspond à une des séquences à aligner. À chaque cellule de cette matrice, on calcule le meilleur score possible en fonction des chemins menant à cet état, comme dans l'alignement par paire.

Cependant, cette méthode a une complexité exponentielle en fonction du nombre de séquences : $O(L^k)$ en temps et en espace, ce qui la rend impraticable au-delà de 3 ou 4 séquences courtes. Malgré cela, elle reste une référence théorique, utilisée pour valider d'autres méthodes sur des cas tests de petite taille.

3.2.2 Approche progressive (méthode gloutonne)

L'approche progressive est la plus utilisée en pratique. Elle est basée sur une idée simple : réaliser d'abord tous les alignements par paires possibles, en calculant une matrice de distances, puis construire un arbre guide (généralement un arbre phylogénétique) représentant l'ordre dans lequel les séquences doivent être alignées.

L'alignement commence par les séquences les plus proches (selon l'arbre), puis les alignements sont ajoutés progressivement à un alignement multiple, en utilisant la notion de *profil* : un alignement de plusieurs séquences est traité comme une séquence unique contenant des fréquences de symboles pour chaque position.

L'un des exemples les plus connus de cette approche est l'algorithme ClustalW, qui reste aujourd'hui une référence pour l'alignement multiple standard.

Cette méthode est rapide et donne de bons résultats dans la majorité des cas, mais elle est sensible aux erreurs initiales : un mauvais alignement de départ peut affecter tout l'alignement final, car il n'y a pas de correction ultérieure.

3.2.3 Approche itérative

L'approche itérative consiste à améliorer progressivement un alignement initial en modifiant ou réalignant certaines parties des séquences de manière locale. Contrairement à l'approche progressive, elle ne dépend pas d'un ordre fixe défini au départ. Elle permet donc de corriger des erreurs d'alignement en ajustant dynamiquement les séquences pour maximiser un score global.

Des algorithmes comme SAGA (genetic algorithm), DiAlign (basé sur des fragments locaux), ou T-Coffee (qui combine alignement global et local par contraintes) utilisent des variantes de cette stratégie.

Cette méthode est souvent plus précise mais aussi plus coûteuse en temps de calcul. Elle est utile quand on recherche un alignement de haute qualité, en particulier dans le cadre d'analyses phylogénétiques ou fonctionnelles poussées.

En résumé, chaque approche présente un compromis entre précision, coût computationnel et sensibilité aux erreurs :

- **Exacte** : très précise mais inutilisable pour plus de 3–4 séquences.
- **Progressive** : rapide, simple et efficace mais sensible aux erreurs d'ordre.
- **Itérative** : plus lente mais plus robuste et flexible.

3.3 Analyse des résultats d'alignement multiple

Dans cette section, nous présentons les résultats obtenus pour l'alignement multiple progressif de différentes tailles de jeux de séquences, de 3 à 10 séquences. Nous avons mesuré pour chaque taille :

- Le temps d'exécution de l'algorithme (en secondes),
- Le score moyen d'alignement de chaque séquence comparé à la séquence consensus.

```
Séquences:  3, Temps: 0.0023s, Score moyen vs consensus: 67.67
Séquences:  4, Temps: 0.0024s, Score moyen vs consensus: 66.50
Séquences:  5, Temps: 0.0033s, Score moyen vs consensus: 63.40
Séquences:  6, Temps: 0.0037s, Score moyen vs consensus: 55.17
Séquences:  7, Temps: 0.0048s, Score moyen vs consensus: 56.29
Séquences:  8, Temps: 0.0052s, Score moyen vs consensus: 55.62
Séquences:  9, Temps: 0.0074s, Score moyen vs consensus: 57.11
Séquences: 10, Temps: 0.0070s, Score moyen vs consensus: 61.20
```

Les sequances utilité

```python
# Exemple de jeu de séquences de base (10 séquences)
base_seqs = [
    "ACGTACGTACGT",
    "ACGTCGTTACGT",
    "ACGTCGTTGCGT",
    "ACGTACGTGCGT",
    "ACGTCGTCACGT",
    "ACGTAGGTACGT",
    "ACGTAGGTGCGT",
    "ACGTTGGTGCGT",
    "ACGTAGGTACGT",
    "ACGTCGGTACGT"
]

```

Nombre de séquences	Temps d'exécution (s)	Score moyen vs consensus
3	0.0023	67.67
4	0.0024	66.50
5	0.0033	63.40
6	0.0037	55.17
7	0.0048	56.29
8	0.0052	55.62
9	0.0074	57.11
10	0.0070	61.20

Commentaires :

- Le temps d'exécution augmente globalement avec le nombre de séquences, ce qui est attendu car l'algorithme traite plus de données.
- Le score moyen d'alignement diminue en général lorsque le nombre de séquences augmente, indiquant que le consensus est moins similaire à chaque séquence individuelle en raison de la diversité accrue.
- On observe cependant une légère remontée du score pour 10 séquences, ce qui peut être dû à la composition spécifique des séquences testées.
- Ces résultats confirment la validité et la cohérence de l'implémentation de l'algorithme d'alignement multiple progressif.

3.4 Tests d'alignement avec augmentation de la longueur des séquences

```
Length: 100, Time: 0.2154s, Avg score: 293.40
Length: 150, Time: 0.4357s, Avg score: 410.20
Length: 200, Time: 0.6529s, Avg score: 611.80
Length: 300, Time: 1.3983s, Avg score: 922.00
Length: 500, Time: 3.8901s, Avg score: 1579.60
Length: 600, Time: 6.0362s, Avg score: 1809.60
```

Longueur des séquences (L)	Temps d'exécution (s)	Score moyen d'alignement
100	0.2154	293.40
150	0.4357	410.20
200	0.6529	611.80
300	1.3983	922.00
500	3.8901	1579.60
600	6.0362	1809.60

TABLE 1 – Résultats des tests d'alignement multiple progressif avec un nombre fixe de séquences (5) et des longueurs croissantes.

3.4.1 Commentaires :

Les résultats montrent une augmentation significative du temps d'exécution lorsque la longueur des séquences augmente. Ceci est attendu car la complexité de l'algorithme dépend directement de la longueur des séquences alignées.

De plus, le score moyen d'alignement augmente également avec la longueur des séquences, ce qui est logique puisque le score est une somme des contributions de chaque position alignée.

On observe que pour les séquences très longues (L = 500, 600), le temps d'exécution devient assez important, ce qui peut constituer une limite pratique à l'utilisation de cette méthode pour des séquences très longues ou pour un grand nombre de séquences.

En conclusion, cette méthode est adaptée pour des séquences de longueur modérée, mais son coût computationnel augmente rapidement avec la longueur, ce qui nécessite éventuellement l'utilisation de méthodes plus efficaces ou d'approximations pour des cas plus complexes.

3.5 Limites de la méthode d'alignement progressif

La méthode d'alignement progressif, bien qu'efficace et relativement simple à implémenter, présente plusieurs limites qu'il est important de souligner à l'issue de nos expérimentations :

- **Optimalité des résultats :** Le principal inconvénient de l'approche progressive est qu'elle est heuristique et ne garantit pas un alignement global optimal. Les erreurs introduites

dans les premières étapes (alignement de paires) ne peuvent pas être corrigées par la suite. Ainsi, la qualité de l'alignement dépend fortement de l'ordre dans lequel les séquences sont alignées.

- **Nombre maximum de séquences :** Nos tests ont montré que la méthode reste raisonnablement rapide jusqu'à environ 10 séquences. Au-delà, le temps d'exécution augmente rapidement, et la lisibilité ainsi que la qualité des alignements peuvent se détériorer, en particulier si les séquences sont longues.
- **Exactitude :** L'utilisation d'un consensus pour construire l'alignement global est une approximation. Elle ne prend pas en compte toutes les relations entre les séquences. Cela peut entraîner une perte d'information ou une mauvaise représentation de certaines régions conservées ou variables.
- **Performance computationnelle :** Comme l'ont montré nos résultats expérimentaux, le temps d'exécution augmente avec le nombre de séquences et la longueur des séquences. Pour des longues séquences ($L \geq 500$) ou un nombre élevé de séquences, la méthode devient peu efficace sans optimisation.
- **Dépendance à la matrice de distance initiale :** Le choix de la stratégie pour calculer la distance entre séquences (score pairwise) influence fortement l'arbre de guide utilisé, et donc l'alignement final. Une mauvaise estimation initiale peut dégrader l'alignement final.

En résumé, la méthode d'alignement progressif convient pour des cas simples avec un nombre modéré de séquences de taille moyenne. Toutefois, pour des cas complexes (grand nombre de séquences, séquences longues ou très divergentes), il peut être préférable de recourir à des méthodes plus sophistiquées comme l'alignement basé sur des profils, l'approche par programmation dynamique multiple (ex : T-Coffee) ou des algorithmes pro

Conclusion

Ce projet nous a permis d'explorer en profondeur les concepts fondamentaux de l'alignement de séquences en bioinformatique. Nous avons commencé par étudier l'alignement par paire à travers les algorithmes classiques de Needleman-Wunsch (alignement global) et de Smith-Waterman (alignement local), en les implémentant en Python avec la matrice de similarité BLOSUM62 et une pénalité de gap constante (**-2**). Des tests expérimentaux sur des séquences de tailles variables ont permis de comparer leurs performances en termes de score d'alignement et de temps d'exécution.

Nous avons ensuite étendu cette étude au cas plus complexe de l'alignement multiple de séquences, en analysant différentes approches théoriques : exacte, progressive et itérative. Une méthode d'alignement progressive a été implémentée avec succès en utilisant la notion de profil, suivie de tests expérimentaux sur un nombre croissant de séquences. Nous avons pu observer l'impact du nombre de séquences et de leur longueur sur la performance et la qualité de l'alignement.

Ce travail met en lumière les compromis inhérents à chaque méthode entre précision, rapidité et faisabilité, tout en soulignant l'importance des algorithmes heuristiques dans le traitement

de grandes quantités de données biologiques. Il ouvre la voie à des perspectives d'amélioration, telles que l'optimisation de l'espace mémoire ou l'utilisation de techniques parallèles, pour traiter efficacement des ensembles de séquences toujours plus importants.