

SPECIFICATION TECHNIQUE DE BESOIN

Version : **0.5**

Date : **15.02.2014**

Rédigé par : **Delphine Meyrieux**

Relu par : **Julien Szlamowicz, Tony Coriolle, Ibrahima Sory Barry,
Clément Etendard, Timothée Guegan**

Approuvé par : **Christophe Carré**

Signature :

MISES A JOUR

[illegible]

Ce document est destiné à traduire les besoins des utilisateurs du logiciel et à établir une référence pour sa validation. Son but est de recenser les principales exigences que l'équipe de développement s'engage à satisfaire dans le cadre du projet.

1. Objet :

L'objectif principal de ce projet est de développer un logiciel permettant de décomposer de grands nombres entiers en facteurs premiers grâce à l'algorithme de Dixon.

L'objectif technique est de proposer à l'utilisateur deux méthodes d'exécution par le biais de deux architectures pour en comparer les performances : Une méthode d'exécution **sérialisée** programmée en Sage et une seconde **parallélisée** programmée en C++. Cette dernière devra être exécutée en utilisant l'architecture CUDA. Il est pour cela nécessaire de maîtriser entièrement CUDA, le projet portant essentiellement sur l'implémentation parallélisée.

Le but final de ce projet sera de comparer la vitesse d'exécution de l'algorithme entre les deux plateformes afin de démontrer que l'architecture CUDA permet d'effectuer des calculs complexes dans un temps beaucoup plus court.

2. Documents applicables et de référence :

Pour cette version, les documents de références sont :

- Le sujet du projet proposé par M. Carré.

3. Terminologie et sigles utilisés :

CUDA : fonctionnalité proposée sur les cartes graphiques récentes du constructeur NVIDIA, permettant d'utiliser la puissance de calcul de ces dernières pour paralléliser et exécuter des tâches extrêmement rapidement.

4. Exigences fonctionnelles :

1. Présentation des fonctionnalités du produit

Le projet s'articule en 4 parties distinctes, nous les différencierons de la manière suivante :

- Sage (SG)
- CUDA (CD)
- User Interface (UI)
- Miscellaneous (MC)

Ci-après, le tableau récapitulatif des fonctionnalités principales (backlog).

Tableau des fonctionnalités principales

[illegible]

2. Description des fonctionnalités

Dans la suite, la complexité sera pondérée en utilisant les premiers termes de la suite de Fibonacci pour qu'il n'y ait pas de valeurs trop similaires et ainsi pouvoir mettre en évidence les plus importants :

- Très Faible : 1
- Faible : 2
- Moyen : 3
- Elevé : 5
- Très Elevé : 8
- Critique : 13

<SG 1> Algorithme de Dixon (Sage).

CudaFactor	
Module : Sage	
<p>Description de la fonctionnalité :</p> <p>Implémenter l'algorithme de Dixon en Sage pour permettre à l'utilisateur de décomposer des nombres en facteurs premiers.</p> <p><u>Entrée</u> : nombre N, borne B</p> <p><u>Sortie</u> : Liste des facteurs premiers du nombre passé en entrée.</p> <p><u>Début</u> :</p> <p>P <- ensemble des premiers < B</p> <p>k <- card(P)</p> <p>R <- ensemble vide</p> <p>Div <- ensemble vide</p> <p>Tester la primalité de N</p> <p><u>Tant que</u> $\prod_{i \in Div} Div_i \neq N$ <u>faire</u></p> <p> // élimination des facteurs triviaux</p> <p> <u>Pour</u> x choisi aléatoirement dans $[\sqrt{N} ; N - 1]$ <u>faire jusqu'à</u> m > k + 1</p> <p> y <- $x^2 \bmod N$</p> <p> <u>Si</u> y est B-friable et y pas dans Div <u>alors</u></p> <p> Ajouter (x,y) à R</p> <p> m <- card(R)</p> <p> <u>Fin si</u></p> <p> <u>Finpour</u></p> <p> <u>Pour</u> $1 \leq i \leq m$ <u>faire</u></p> <p> $v_{i,p} \text{ tq } y_i = \prod_{p \in P} p^{v_{i,p}}$</p> <p> <u>finpour</u></p> <p> M <- matrice $(v_{i,p} \bmod 2)_{p \in P, 1 \leq i \leq m}$</p> <p> E <- vecteur non nul noyau de M</p> <p> u <- $\prod_{1 \leq i \leq m} x_i^{2e_i}$</p> <p> v <- $\prod_{p \in P} p^{\frac{1}{2} \sum_{1 \leq i \leq m} v_{i,p} e_i}$</p> <p> <u>si</u> pgcd(u-v,N) n'est pas dans {1,N} <u>alors</u></p> <p> Ajouter u-v à Div</p> <p> <u>Sinon si</u> pgcd(u+v,N) n'est pas dans {1,N} <u>alors</u></p> <p> Ajouter u+v à Div</p> <p> <u>Sinon</u></p> <p> Erreur</p> <p> <u>finsi</u></p> <p> <u>Fintantque</u></p> <p>Retourner Div</p> <p><u>Fin</u></p>	<p>Identificateur :</p> <p>SG 1</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>N est un entier relatif.</p>	<p>Priorité :</p> <p>Indispensable</p>
<p>Conditions d'acceptation :</p> <p>Vérification et approbation du consultant technique (client).</p> <p>Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.</p> <p>Temps d'exécution optimisé au maximum : $O(L_N(1/2; 2))$</p>	<p>Complexité :</p> <p>13</p>

<SG 2> Passerelle C++/Python (Sage).

CudaFactor	
Module : Sage	
<p>Description de la fonctionnalité :</p> <p>Implémentation d'une passerelle entre C++ et Python afin de pouvoir lancer l'algorithme Sage depuis l'interface graphique de l'application.</p>	<p>Identificateur :</p> <p>SG 2</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>Précondition : Avoir implémenté et vérifié la version générale de l'algorithme en Sage.</p>	<p>Priorité :</p> <p>Important</p>
<p>Conditions d'acceptation :</p> <p>Lance bien l'algorithme en Sage</p>	<p>Complexité :</p> <p>8</p>

<SG 3> Heuristiques (Sage).

CudaFactor	
Module : Sage	
<p>Description de la fonctionnalité :</p> <p>Implémenter les heuristiques de l'algorithme de Dixon pour permettre à l'utilisateur d'adapter l'algorithme en fonction de ses besoins.</p> <p>Liste des heuristiques :</p> <ul style="list-style-type: none"> • Crible quadratique (optimisé pour des nombres à moins de 100 chiffres) (QS) • Factorisation par crible sur les corps de nombres généralisé (GNFS) (optimisé pour des nombres à plus de 100 chiffres) • Fractions continues (CFRAC) <p>Algorithme du crible quadratique :</p> <p>Notons p_1, \dots, p_k les nombres premiers de la base des facteurs avec $K \approx \frac{\frac{1}{2}B}{\log B}$</p> <p>On suppose $T \geq B$.</p> <p>(1) Pour $x = 0$ à T, poser $v[x] \leftarrow 1$</p> <p>(2) Pour $i = 1$ à K, faire</p> <p>(a) Poser $p \leftarrow p_i, e \leftarrow 1$</p> <p>(b) Faire $R \leftarrow \text{racines_de_f}(N, pe)$</p> <p>(c) Si tous les éléments de R sont $> T$, passer à la prochaine valeur en 2</p> <p>(d) Pour tout $x \in R$, Faire</p> <p style="padding-left: 20px;">Tant que $x \leq T$</p> <p style="padding-left: 20px;">$v[x] \leftarrow v[x] \cdot p$</p> <p style="padding-left: 20px;">$x \leftarrow x + p^e$</p> <p>(e) Faire $e \leftarrow e + 1$ et retourner en 2.b</p> <p>(3) Renvoyer $(x, f(x))$ pour tous les $x = 0, \dots, T$ tels que $v[x] = f(x)$</p> <p>Les deux autres algorithmes font partie de la recherche pour le moment.</p>	<p>Identificateur :</p> <div style="border: 1px solid black; padding: 20px; text-align: center; height: 200px;"> <p>SG 3</p> </div>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>Précondition : Avoir implémenté et vérifié la version générale de l'algorithme en Sage.</p>	<p>Priorité :</p> <div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Facultatif</p> </div>
<p>Conditions d'acceptation :</p> <p>Vérification et approbation du consultant technique (client)</p> <p>Temps d'exécution optimisé pour chaque heuristique $L_N(1/2, 1)$.</p> <p>Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.</p>	<p>Complexité :</p> <div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>8</p> </div>

<CD 1> Implémentation de l'algorithme du PGCD (CUDA).

CudaFactor	
Module : CUDA	
Description de la fonctionnalité : Implémentation parallélisée de l'algorithme du PGCD en C++ sur CUDA : Algorithme d'Euclide Etendu.	Identificateur : <div>CD 1</div>
Préconditions et Conditions de déclenchement/d'arrêt Le nombre nb est bien un entier.	Priorité : <div>Indispensable</div>
Conditions d'acceptation : Résultat correct : le nombre renvoyé est le plus grand diviseur commun du nombre nb passé en argument.	Complexité : <div>8</div>

<CD 2> Fonction liste des nombres premiers (CUDA).

CudaFactor	
Module : CUDA	
Description de la fonctionnalité : Implémentation parallélisée de la fonction permettant de construire la liste des nombres premiers inférieurs à la borne passée en argument en CUDA.	Identificateur : <div>CD 2</div>
Préconditions et Conditions de déclenchement/d'arrêt Le nombre borne passé en argument est bien un entier.	Priorité : <div>Indispensable</div>
Conditions d'acceptation : Résultat correct : liste de nombres premiers strictement inférieurs à la borne passé en argument.	Complexité : <div>8</div>

<CD 3> Remplissage de l'ensemble R (CUDA).

CudaFactor	
Module : CUDA	
<p>Description de la fonctionnalité :</p> <p>Implémentation parallélisée de la fonction permettant de remplir l'ensemble R contenant les couples (x,y) nécessaires au bon fonctionnement de l'algorithme. k = nombre de nombres premiers inférieurs à la borne tant que m < k + 1 faire choisir x aléatoirement entre \sqrt{N} et $N - 1$ $y = x^2 \bmod N$ ajouter (x,y) dans R m = cardinal de R fin tant que</p>	<p>Identificateur :</p> <div style="border: 1px solid black; padding: 20px; text-align: center;">CD 3</div>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <div style="border: 1px solid black; height: 30px;"></div>	<p>Priorité :</p> <div style="border: 1px solid black; padding: 5px; text-align: center;">Indispensable</div>
<p>Conditions d'acceptation :</p> <p>Résultat correct : ensemble de couples (x,y) où y est le carré de x modulo N et le cardinal de cet ensemble est le même que celui de la liste des nombre premiers construite en <CD 2></p>	<p>Complexité :</p> <div style="border: 1px solid black; padding: 20px; text-align: center;">8</div>

<CD 4> 3^{ème} boucle de l'algorithme (CUDA).

CudaFactor	
Module : CUDA	
<p>Description de la fonctionnalité :</p> <p>Implémentation parallélisée de la fonction de la 3^{ème} boucle de l'algorithme avec création de la matrice et construction des nombres u et v permettant de trouver un diviseur premier de nb.</p> <p>M <- matrice $(v_{i,p} \bmod 2)_{p \in P, 1 \leq i \leq m}$</p> <p>E <- vecteur non nul noyau de M</p> <p>$u \leftarrow \prod_{1 \leq i \leq m} x_i^{2e_i}$</p> <p>$v \leftarrow \prod_{p \in P} p^{\frac{1}{2} \sum_{1 \leq i \leq m} v_{i,p} e_i}$</p> <p><u>si</u> pgcd(u-v,N n'est pas dans {1,N} <u>alors</u></p> <p>Ajouter u-v à Div</p> <p><u>Sinon si</u> pgcd(u+v,N) n'est pas dans {1,N} <u>alors</u></p> <p>Ajouter u+v à Div</p> <p><u>Sinon</u></p> <p>Erreur</p> <p><u>finsi</u></p>	<p>Identificateur :</p> <p>CD 4</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p>	<p>Priorité :</p> <p>Indispensable</p>
<p>Conditions d'acceptation :</p> <p>Résultat correct : ensemble Div contenant les u+v et u-v facteurs premiers de nb.</p>	<p>Complexité :</p> <p>13</p>

<CD 5> Algorithme de Dixon : recomposition (CUDA).

CudaFactor	
Module : CUDA	
<p>Description de la fonctionnalité :</p> <p>Implémenter la reconstruction de l'algorithme suivant à partir des morceaux construits en <CD 1>, <CD 2>, <CD 3> et <CD 4>.</p> <p><u>Entrée</u> : nombre N, borne B</p> <p><u>Sortie</u> : Liste des facteurs premiers du nombre passé en entrée.</p> <p><u>Début</u> :</p> <p>P <- ensemble des premiers < B</p> <p>k <- card(P)</p> <p>R <- ensemble vide</p> <p>Div <- ensemble vide</p> <p>Tester la primalité de N</p> <p><u>Tant que</u> $\prod_{i \in Div} Div_i \neq N$ <u>faire</u></p> <p> // élimination des facteurs triviaux</p> <p> Pour x choisi aléatoirement dans $[\sqrt{N} ; N - 1]$ <u>faire jusqu'à</u> m > k + 1</p> <p> Création d'un thread</p> <p> y <- $x^2 \bmod N$</p> <p> <u>Si</u> y est B-friable et y pas dans Div <u>alors</u></p> <p> Ajouter (x,y) à R</p> <p> m <- card(R)</p> <p> <u>Fin si</u></p> <p> <u>finpour</u></p> <p> Pour 1 <= i <= m <u>faire</u></p> <p> $v_{i,p} \text{ tq } y_i = \prod_{p \in P} p^{v_{i,p}}$</p> <p> <u>finpour</u></p> <p> M <- matrice $(v_{i,p} \bmod 2)_{p \in P, 1 \leq i \leq m}$</p> <p> E <- vecteur non nul noyau de M</p> <p> $u \leftarrow \prod_{1 \leq i \leq m} x_i^{2e_i}$</p> <p> $v \leftarrow \prod_{p \in P} p^{\frac{1}{2} \sum_{1 \leq i \leq m} v_{i,p} e_i}$</p> <p> <u>si</u> pgcd(u-v,N n'est pas dans {1,N} <u>alors</u></p> <p> Ajouter u-v à Div</p> <p> <u>Sinon si</u> pgcd(u+v,N) n'est pas dans {1,N} <u>alors</u></p> <p> Ajouter u+v à Div</p> <p> <u>Sinon</u></p> <p> Erreur</p> <p> <u>finsi</u></p> <p> <u>Fintantque</u></p> <p>Retourner Div</p> <p><u>Fin</u></p>	<p>Identificateur :</p> <p>CD 5</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>N est un entier relatif.</p>	<p>Priorité :</p> <p>Indispensable</p>
<p>Conditions d'acceptation :</p> <p>Résultat correct satisfaisant après soumission au client. Temps d'exécution optimisé au maximum : $O(L_N(1/2; 2))$</p> <p>Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.</p>	<p>Complexité :</p> <p>13</p>

<CD 6> Heuristiques (CUDA).

CudaFactor	
Module : CUDA	
<p>Description de la fonctionnalité :</p> <p>Implémenter et paralléliser les heuristiques de l'algorithme de Dixon en C++ sur CUDA pour permettre à l'utilisateur de décomposer des nombres en facteurs premiers.</p> <p>Liste des heuristiques :</p> <ul style="list-style-type: none"> • Crible quadratique (optimisé pour des nombres à moins de 100 chiffres) (QS) • Factorisation par crible sur les corps de nombres généralisé (GNFS) (optimisé pour des nombres à plus de 100 chiffres) : utilisant de la théorie algébrique des nombres et reposant sur la factorisation d'idéaux dans des anneaux d'entiers • Fractions continues (CFRAC) <p><u>Algorithme du crible quadratique</u> : cf. SG2</p>	<p>Identificateur :</p> <p>CD 6</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>Précondition : Avoir implémenté et vérifié la version générale de l'algorithme pour CUDA.</p>	<p>Priorité :</p> <p>Facultatif</p>
<p>Conditions d'acceptation :</p> <p>Vérification et approbation du consultant technique (client)</p> <p>Temps d'exécution optimisé pour chaque heuristique $L_N(1/2,1)$.</p> <p>Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.</p>	<p>Complexité :</p> <p>8</p>

<UI 1> Choix de la méthode d'exécution.

CudaFactor	
Module : User Interface	
<p>Description de la fonctionnalité :</p> <p>L'utilisateur doit pouvoir choisir la méthode d'exécution de l'algorithme : Sage ou CUDA.</p>	<p>Identificateur :</p> <p>UI 1</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>NA.</p>	<p>Priorité :</p> <p>Indispensable</p>
<p>Conditions d'acceptation :</p> <p>L'action enregistre le choix de la méthode à exécuter et mène à l'exécution de la bonne méthode.</p>	<p>Complexité :</p> <p>2</p>

<UI 2> Choix de l'heuristique.

CudaFactor	
Module : User Interface	
<p>Description de la fonctionnalité :</p> <p>L'utilisateur doit pouvoir choisir l'heuristique de l'algorithme de Dixon qu'il souhaite utiliser.</p>	<p>Identificateur :</p> <p>UI 2</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>NA.</p>	<p>Priorité :</p> <p>Facultatif</p>
<p>Conditions d'acceptation :</p> <p>L'action enregistre bien l'heuristique choisie et mène à l'exécution du bon algorithme.</p>	<p>Complexité :</p> <p>2</p>

<UI 3> Choix du nombre à factoriser.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur doit pouvoir choisir le nombre à factoriser.	Identificateur : UI 3
Préconditions et Conditions de déclenchement/d'arrêt Une méthode d'exécution et un algorithme ont été choisis.	Priorité : Indispensable
Conditions d'acceptation : L'action enregistre le nombre à factoriser et lance la factorisation.	Complexité : 3

<UI 4> Description de l'algorithme.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur doit pouvoir avoir une description rapide de l'algorithme (ou de l'heuristique) qu'il s'apprête à utiliser pour être sûr qu'il fait un choix cohérent.	Identificateur : UI 4
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : L'utilisateur présélectionne une heuristique. Condition d'arrêt : L'utilisateur effectue une autre action.	Priorité : Facultatif
Conditions d'acceptation : Description correcte et synthétique.	Complexité : 2

<UI 5> Surveillance de l'exécution.

CudaFactor	
Module : User Interface	
<p>Description de la fonctionnalité :</p> <p>L'utilisateur peut connaître l'état d'avancement des calculs, le nombre d'instructions par seconde ou l'état général du système en temps réel et des détails comme combien de colonnes de la matrice a été trouvé dans la phase 1 de l'algorithme de Dixon.</p>	<p>Identificateur :</p> <p>UI 5</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>Condition de déclenchement : L'utilisateur lance une factorisation. Condition d'arrêt : Le nombre a été factorisé.</p>	<p>Priorité :</p> <p>Important</p>
<p>Conditions d'acceptation :</p> <p>Les valeurs rapportées sont exactes et s'actualisent en temps réel.</p>	<p>Complexité :</p> <p>5</p>

<UI 6> Affichage du rapport d'exécution.

CudaFactor	
Module : User Interface	
<p>Description de la fonctionnalité :</p> <p>L'utilisateur peut avoir un compte rendu de l'exécution à la fin de celle-ci. Le rapport est composé de temps d'exécution, liste des facteurs avec leurs puissances, le nombre d'instructions effectuées, l'entier de départ, l'algorithme utilisé, la méthode utilisée et les caractéristiques matérielles.</p>	<p>Identificateur :</p> <p>UI 6</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>Condition de déclenchement : La factorisation est terminée.</p>	<p>Priorité :</p> <p>Important</p>
<p>Conditions d'acceptation :</p> <p>Les valeurs doivent être celles rendues par l'algorithme. Présentation claire.</p>	<p>Complexité :</p> <p>5</p>

<MC 1> Générer un rapport au format *.xml*.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur peut générer un rapport d'exécution au format <i>.xml</i> .	Identificateur : MC 1
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : La factorisation est terminée.	Priorité : Facultatif
Conditions d'acceptation : Pas d'erreur de syntaxe dans le langage. Hérite des caractéristiques de l'UI 6.	Complexité : 5

<MC 2> Comparaison de deux rapports.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur peut comparer deux rapports de précédentes exécutions, sous forme de tableau avec une coloration des données : rouge pour les moins performantes et vert pour les plus performantes.	Identificateur : MC 2
Préconditions et Conditions de déclenchement/d'arrêt Précondition : les rapports concernent une factorisation du même nombre, les rapports sont au format <i>.xml</i>	Priorité : Facultatif
Conditions d'acceptation : Comparatif synthétique et correct des deux rapports. Mise en avant des données performantes.	Complexité : 5

<MC 3> Interruption/Reprise de l'exécution.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur peut mettre en pause la factorisation pour la reprendre plus tard, ou simplement l'interrompre.	Identificateur : MC 3
Préconditions et Conditions de déclenchement/d'arrêt NA.	Priorité : Facultatif
Conditions d'acceptation : Le programme libère correctement les ressources.	Complexité : 8

<MC 4> Conseil sur l'heuristique à utiliser.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur est conseillé sur l'heuristique à utiliser en fonction de l'entier qu'il souhaite factoriser. (Certaines heuristiques sont plus efficaces pour certains types de nombres)	Identificateur : MC 4
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : L'utilisateur a choisi un nombre à factoriser.	Priorité : Facultatif
Conditions d'acceptation : Conseil approprié à la taille de l'entier.	Complexité : 2

<MC 5> Décimal, Hexa, Binaire.

CudaFactor	
Module : Miscellaneous	
<p>Description de la fonctionnalité :</p> <p>L'utilisateur peut entrer le nombre à factoriser dans plusieurs bases : décimale, hexadécimale, binaire car il peut être amené à entrer de très grands nombres.</p> <p>Conversion des nombres entrés dans l'application.</p>	<p>Identificateur :</p> <p>MC 5</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>La base choisie doit être soit décimale, hexadécimale ou binaire.</p>	<p>Priorité :</p> <p>Important</p>
<p>Conditions d'acceptation :</p> <p>Le nombre entré et le nombre converti sont égaux.</p>	<p>Complexité :</p> <p>5</p>

<MC 6> Etude sur l'algorithme de Dixon

CudaFactor	
Module : Miscellaneous	
<p>Description de la fonctionnalité :</p> <p>Réaliser une étude préliminaire des algorithmes et heuristiques nécessaires au développement de l'application.</p>	<p>Identificateur :</p> <p>MC 6</p>
<p>Préconditions et Conditions de déclenchement/d'arrêt</p> <p>NA.</p>	<p>Priorité :</p> <p>Indispensable</p>
<p>Conditions d'acceptation :</p> <p>NA.</p>	<p>Complexité :</p> <p>8</p>

<MC 7> Optimisation du code

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : Apporter des corrections et optimisations sur les implémentations des algorithmes en SAGE et CUDA.	Identificateur : <div>MC 7</div>
Préconditions et Conditions de déclenchement/d'arrêt Précondition : Avoir implémenté et vérifié la version générale et les heuristiques de l'algorithme.	Priorité : <div>Important</div>
Conditions d'acceptation : Résultat satisfaisant après soumission au client. Code correct.	Complexité : <div>13</div>

5. Exigences opérationnelles :

Nom	Identificateur	Priorité
Une carte graphique NVIDIA supportant CUDA nous permettant le développement en CUDA	EO 1	Indispensable
Un système d'exploitation UNIX nous permettant le développement du projet ainsi que son exécution	EO 2	Indispensable

Ces exigences sont nécessaires à la réalisation de toutes les exigences fonctionnelles citées précédemment.

6. Exigences de qualité

Nom	Identificateur	Priorité
Exécution rapide de l’algorithme en SAGE	EQ 1	Indispensable
Exécution de l’algorithme en CUDA optimisée par rapport au temps d’exécution obtenu avec SAGE ce qui dépend de la parallélisation que l’on fera sur celui-ci	EQ 2	Indispensable

7. Exigences d’interface

L’interface n’étant pas une priorité pour la réussite du projet, nos exigences seront les suivantes :

Nom	Identificateur	Priorité
Simple	EI 1	Important
Fonctionnelle	EI 2	Important
Fluide	EI 3	Important
Ergonomique	EI 4	Important

8. Cas d'utilisation

