MODELES ET METHODES DE SIMULATION DE CONTROLE NON-DESTRUCTIF PAR ULTRASON MASSIVEMENT PARALLELES

Jason LAMBERT[1], Gilles ROUGERON[1], Lionel LACASSAGNE[2]

1 CEA LIST, CEA Saclay - Digiteo Labs, PC120, 91191 Gif-sur-Yvette cedex, France.

2 Laboratoire de Recherche en Informatique, Université Paris-Sud, F-91405 Orsay, France

RESUME

Dans le cadre du CND ultrasonore par capteurs multiéléments, on utilise de plus en plus couramment des simulations d'inspection pour mettre au point de nouveaux contrôles ou de nouveaux capteurs, pour qualifier des méthodes, pour interpréter des résultats complexes, ou pour la formation. Dans ce contexte, la demande de temps de calcul réduits et si possible interactifs est forte. Or aujourd'hui la puissance des architectures massivement parallèles (manycore et multicore sur architecturesGPP, *general purpose processors*, et GPU, *graphic processing unit*) permet de l'envisager.

Au sein de la plateforme CIVA, a été développé un outils de simulation de champ interactive reposant sur un modèle de pinceaux basé sur une résolution numérique de formulations analytiques des trajets optique de la propagation de l'onde ultrasonore à travers la pièce inspectée.

Ce poster présente, dans un premier temps, les modèles utilisés pour les simulations interactives, leur domaine d'application, ainsi que les spécificités de leur implémentation sur architectures GPP. Puis sont donnés quelques résultats de validité et performance sur différents cas d'applications industriels.

INTRODUCTION

La simulation de champ, c'est-à-dire du faisceau rayonné par un capteur, est un outil devenu indispensable dans la conception, la validation et l'interprétation de contrôles. C'est également une donnée d'entrée des calculs d'interaction faisceau/défaut. L'objectif du calcul de champ est d'obtenir, en chaque point de la zone de champ échantillonnée, le signal correspondant au déplacement du au faisceau émis.

Une simulation interactive est souhaitable pour permettre une meilleureanalyse des problématiques rencontrées et pour offrir une facilité d'utilisation. Simuler le comportement du faisceau à travers la pièce nécessite un grand nombre de calculs. Le recours aux architectures informatiques modernes permet de disposer d'une puissance de calcul suffisante pour obtenir des performances interactives sur des configurations industrielles d'inspection simples.

Le modèle utilisé au sein de la plateforme CIVA repose sur le modèle des pinceaux, modélisant l'onde par de nombreux pinceaux émis par le capteur. Afin de de tirer parti des capacités des matériels disponibles sur des plateformes de type station de travail, il est nécessaire de viser une algorithmie régulière qui se prête à la parallélisation et aux calculs intensifs.

Dans un premier temps, le modèle de calcul de champ utilisé sera présenté, en particulier ses spécificités afin d'adresser les architectures parallèles et les contraintes imposées au domaine d'application. Ensuite les détails d'une implémentation de référence seront présentés, ainsi que les différentes optimisations apportées en vue de tirer parti des instructions vectorisées. Enfin, des performances seront exposées sur un jeu de configurations de tests proches des cas d'utilisations industrielles.

PRESENTATION DU MODELE DES PINCEAUX RAPIDE

Modèle des pinceaux

Le modèle des pinceaux estime, en un point de champ, la contribution élémentaire des différents points sources correspondant à un capteur échantillonné surfaciquement. Ce modèle décrit dans la littérature [1] permet d'approximer l'onde émise en un point source comme un cône de sommet la source se propageant dans les matériaux, appelé « pinceau ».

La propagation de l'onde dans les matériaux se fait en suivant la loi de Snell-Decartes au passage de chaque interface, en fonction des caractéristiques des milieux de part et d'autre de l'interface (vitesses de propagation du type d'onde considéré). Ainsi on peut mettre en équation le chemin axial décrivant localement cette onde sous la forme d'un pinceau comme illustré par la Figure 1.

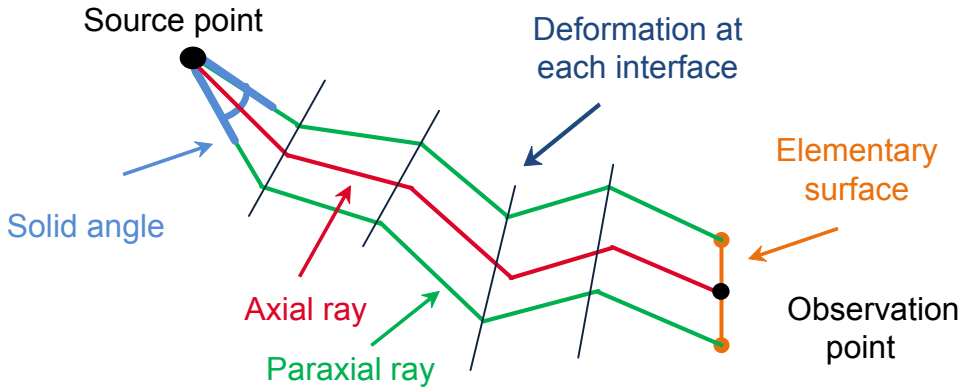

Figure 1 Propagation d'un pinceau

Afin de modéliser par une algorithmie régulière le calcul de champ, l'application se restreint actuellement aux pièces homogènes isotropes décrites par une géométrie constituée de surfaces planes. Ainsi pour calculer le trajet de l'onde émise entre un point émetteur S et le point P dans une pièceimmergée, en mode direct, on pose le problème tel que sur la Figure 2.

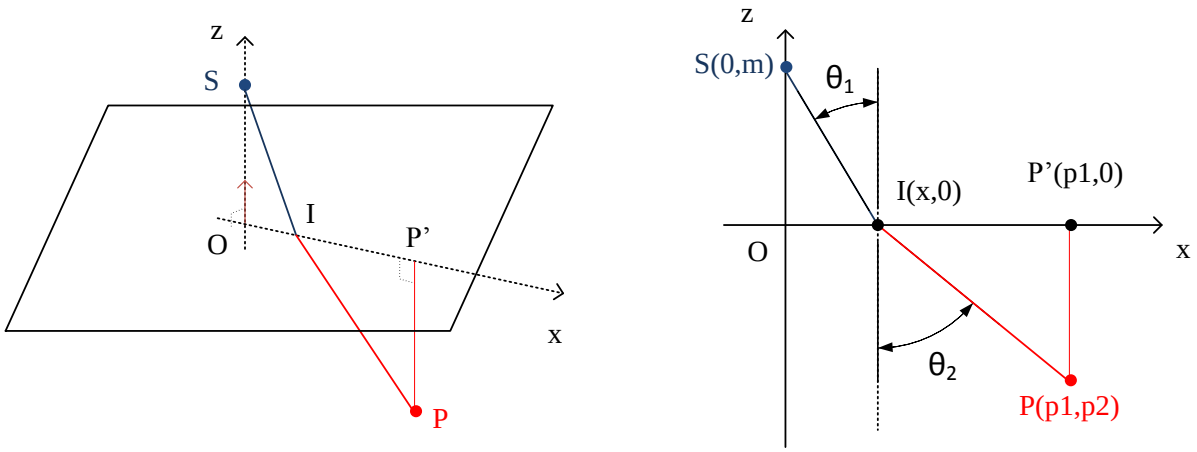

Figure 2 Recherche d'un trajet analytique - mode direct en milieu homogène isotrope

A partir de la Figure 2, on obtient $\sin(\Theta 1) = \frac{x}{\sqrt{x^2+m^2}}$ et $\sin(\Theta 2) = \frac{p_1 - x}{\sqrt{(p_1-x)^2+{p_2}^2}}$ or d'après la relation de Snell-Decartes, $\frac{v_1}{v_2} = \frac{\sin(\Theta 1)}{\sin(\Theta 2)}$.

On obtient, en posant $= \frac{v_1}{v_2}$, le polynôme $P[X]$:

$$P[X] = X^4 - 2p_1X^3 + \left(p_1^2 + \frac{v^2p_2^2 - m^2}{v^2-1}x^2\right)X^2 + \frac{2p_1m^2}{v^2-1}X - \frac{p_1^2m^2}{v^2-1} = 0$$

La recherche du trajet correspond à trouver la racine réelle d'un polynôme de degré 4 vérifiant la réalité physique du trajet (sens du trajet trouvé, conformité avec le type de capteur utilisé…). Il arrive qu'aucun trajet ne relie le point source au point de champ ; le pinceau est alors écarté. La recherche de la racine réelle supérieure à 0 de $P[X]$ s'effectue au moyen de la méthode numérique de Newton.

Une fois le trajet à l'interface déterminé, on calcule la matrice de propagation du pinceau afin d'obtenir le coefficient de divergence DF décrivant la déformation du pinceau au passage de l'interface. Dans son article de [1], N. Gengembre présente les différentes matrices de propagations possibles pour une onde ultrasonore et la manière de les calculer. Dans le cas traité ici, on obtient une expression analytique de DF. Les coefficients de Fresnel donnent enfin l'atténuation subie par l'onde au passage de l'interface.

Le calcul étant associé à une surface élémentaire du capteur dS autour du point source, l'énergie propagée est proportionnelle à cette dernière. Ainsi, il est possible de décrire le pinceau selon son amplitude (produit de DF, de dS et du coefficient de Fresnel), la direction du déplacement associée et son temps de vol associé.

Formation de la réponse impulsionnelle

Une discrétisation infinitésimale de la surface du capteur permettrait de prendre en compte les pinceaux comme des Dirac$\delta(t-\tau)$ (où τ serait le temps de vol du pinceau). Cependant, cette approche nécessiterait un temps de calcul important. On utilise plutôt un créneau d'étalementΔ_t, centré sur τ. En tenant compte des caractéristiques géométriques du trajet et de dS la surface du capteur, il est possible de déterminer l'étalement temporel du pinceau, c'est-à-dire la différence de marche de celui-ci sur le capteur. La réponse impulsionnelle résultante est une fonction rectangle de hauteur $\frac{A_\delta}{\Delta_t}$ (où $A_\delta$est l'amplitude du Dirac équivalent).

On superpose les réponses impulsionnellesélémentaires de chacune des sources ponctuelles afin d'obtenir la réponse impulsionnelle complète du traducteur. C'est lors de cette sommation qu'il est possible de prendre en compte, dans le cas d'un traducteurmultiéléments, les lois de retards appliquées aux différents éléments.

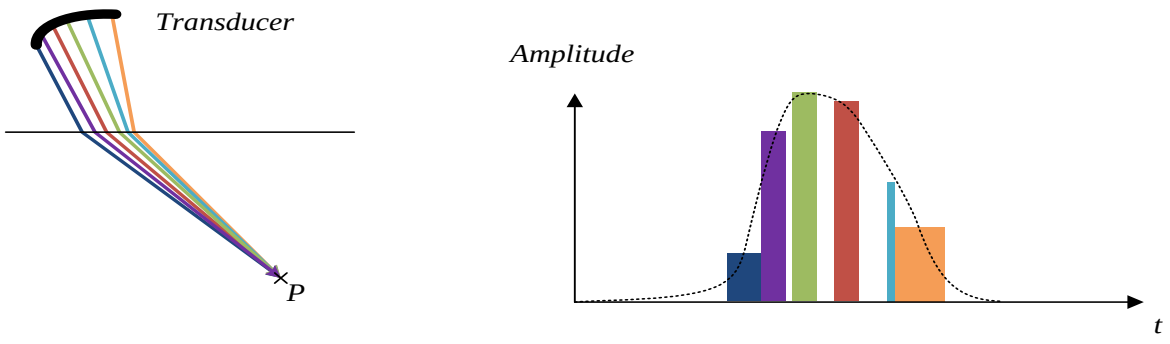

Figure 3 Sommation des contributions des pinceaux sur une réponse impulsionnelle de déplacement

Cette réponse impulsionnelle correspond au champ associé à l'émission d'un Dirac temporel par le traducteur. Pour prendre en compte le signal de l'excitation appliquée au traducteur, dit signal de référence, il faut convoluer la réponse impulsionnelle avec ce dernier. Or, comme cette réponse peut être complexe, sa partie réelle est convoluée avec le signal de référence alors que sa partie imaginaireest subitla transformée de Hilbert avec le signal de référence.

Enfin, pour représenter le champ à l'utilisateur, on extrait en chaque point de champ l'amplitude maximale du déplacement en moduleA_{max} et T_{max} le temps de vol associé.

Domaine d'application

Ce modèle se base sur une résolution numérique du trajet des pinceaux à travers des formulations analytiques de sa propagation dans la pièce ce qui limite le champ des configurations compatibles sur lesquelles simuler un champ ultrasonore. Ainsi, il est applicable aux configurations composées de :

- Pièces de milieux homogènes et isotropes, construite de surfaces planes ;
- Aux modes directs et rebond-fond.
- Capteurs multiéléments conformables au contact ou en immersion.

ALGORITHME DE REFERENCE

Dans un premier temps, l'algorithme naïf est un traitement que l'on peut qualifier d'horizontal, reprenant les différentes étapes prévues dans le modèle et les réalisantchacune entièrement pour l'ensemble des points de champ avant de passer à l'étape suivante. Son algorithmie est présentée par l'Algorithme 1. Sa structure réalisant chacune des étapes du calcul de la simulation de champ une à une permet d'analyser facilement et individuellementles performances de chacune.

Validation

Les résultats obtenus par cette implémentation de référence ont été validés sur un ensemble de configurations caractéristiques par rapport aux résultats obtenus par la plateforme de simulation CIVA 11. Pour l'ensemble de ces configurations, ont été vérifiées : la position et l'amplitude du maximum de déplacement sur les zones de champ ; les temps de vol associés aux valeurs champs calculés ; l'échodynamique sur la zone de champ et la taille et la forme de la tâche focale observée à -3dB (voir laFigure 4).

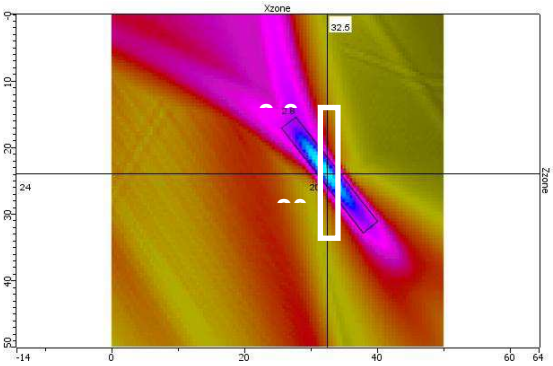

Figure 4Mesure de la tache focale à -3dB

Ces validations par rapport à CIVA 11 ont permis de valider unepremière implémentation du calcul de champ rapide qui a ensuite elle-même servi de référence aux différentes implémentations optimisées présentées plus avant. Les résultats obtenus au sein de la maquette de développement sont exportés de manière à automatiser l'analyse de validité en comparant exhaustivement les résultats d'une implémentation à l'autre.

```
1       //Calcul des pinceaux
2       for( point de champ )
3           for( mode de champ)
4               for(couple de surfaces possible) //rebond -> 1entrée, 1 fond
5                   for(élément du capteur)      //direct -> 1entrée
6                   {
7                       calcul d'un pinceau et stockage
8                           - direction
9                           - amplitude
10                          - temps de vol
11                          - étalement temporel
12                  }
13
14      //Détermination de la taille des signaux
15      taille max des signaux en mémoire
16      for( point de champ )
17      {
18          tmin,tmax locaux
19          for( mode de champ)
20              for(couple de surfaces possible)
21                  for(élément du capteur)
22                  {
23                      lecture des informations temporelles d'un pinceau
24                      application des retards
25                      mise à jours tmin,tmax locaux
26                  }
27          stockage de tmin local
28          mise à jour de la taille max des signaux en mémoire
29      }
30
31      Initialisation des signaux en mémoire
32      //Sommation et extraction des valeurs du champ
33      for( point de champ P)
34      {
35          for( mode de champ)
36              for(couple de surfaces possible)
37                  for(élément du capteur)
38                  {
39                      lecture du pinceau
40                      application des retards
41                      sommation de la contribution du pinceau sur la réponse
42      impulsionnelle de déplacement, locale au point
43                  }
44          Convolution de la R.I. avec le signal de référence
45          Extraction du module du maximum de déplacement Amax et du Tmax associé
46
47          image_tmax[P]=Tmax;
48          image_amax[P]=Amax;
49      }
```

Algorithme 1 Référence

Paramètres et configurations étudiées

A partir de cet algorithme de référence, il est possible de déterminer les paramètres numériques impactant la quantité de calcul réalisés pour une simulation de champ donnée. Afin de qualifier les performances de la simulation de champ rapide, un jeu de configurations de champ réalistes et compatible avec le modèle choisi a été développé. Ce jeu de donnée s'appuie sur une première configuration « de base » centrale qui est ensuite déclinée afin de varier indépendamment les différents paramètres pouvant impacter les performances.

Ces axes sont les suivants :

- Le nombre de mode de champ
- Le nombre de points capteur
- Le nombre de points de champ
- La fréquence d'échantillonnage temporel (dépendant du signal de référence choisi, ce qui influe sur la taille des signaux en nombre d'échantillons)
- La complexité géométrique (le nombre de surfaces et de fonds influe sur le nombre de pinceaux calculés)
- Un dernier axe pouvant influer les performances du calcul est lamanière dont sont réparti les étalements temporels des pinceaux d'un même point : s'il y a ou non accès concurrent aux mêmes échantillons temporels de la réponse impulsionnelle. Pour faire varier ce paramètre, on joue sur les lois de retard en focalisant ou non le champ : plus le faisceau est focalisé et plus les contributions des pinceaux sont temporellement simultanés.

Ainsi on définit le jeu de configurations suivantes à partir d'une configuration de base qui évolue selon ces axes. Celui-ci est présenté par la Figure 5.

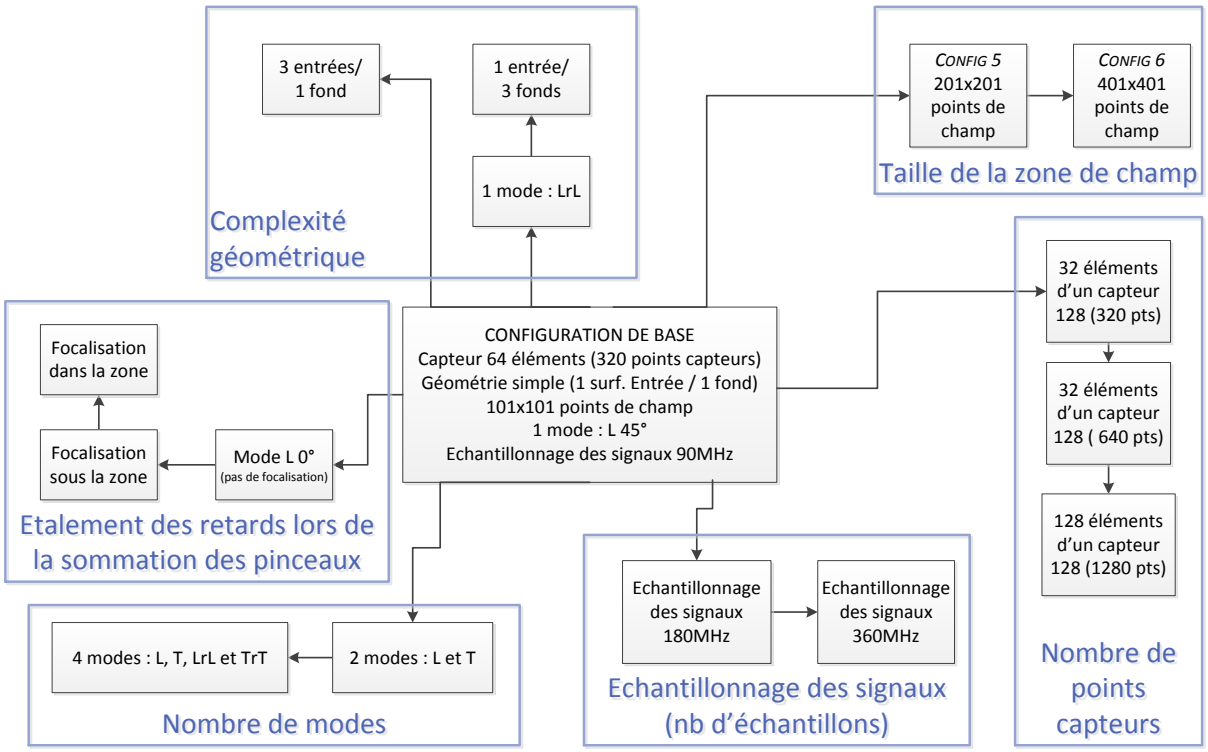

Figure 5 Arbre du jeu de configurations étudiées

OPTIMISATIONS

L'algorithme de référence présenté ci-avant est de type « *embarassingly parallel* » : à l'échelle des points de champ, les différentes taches de calcul peuvent se diviser aisément sur plusieurs threads au moyen d'*OpenMP*. Cependant, même sur des architectures puissantes, le multithreading n'est pas suffisant pour obtenir une performance interactive avec 5,4 images par secondes sur la configuration de base sur une machine disposant de 2 processeurs E5-2697v2 cadencés à 2.70GHz.

Afin de qualifier les différentes étapes du calcul de champ, la Figure 6 présente la répartition du temps passé dans chacune des étapes de calcul par rapport au temps total d'exécution de la simulation de champ sur l'ensemble des configurations. On y constate que le travail d'optimisation va se focaliser sur l'étape de calcul des pinceaux et celle de sommation des contributions et traitement du signal.

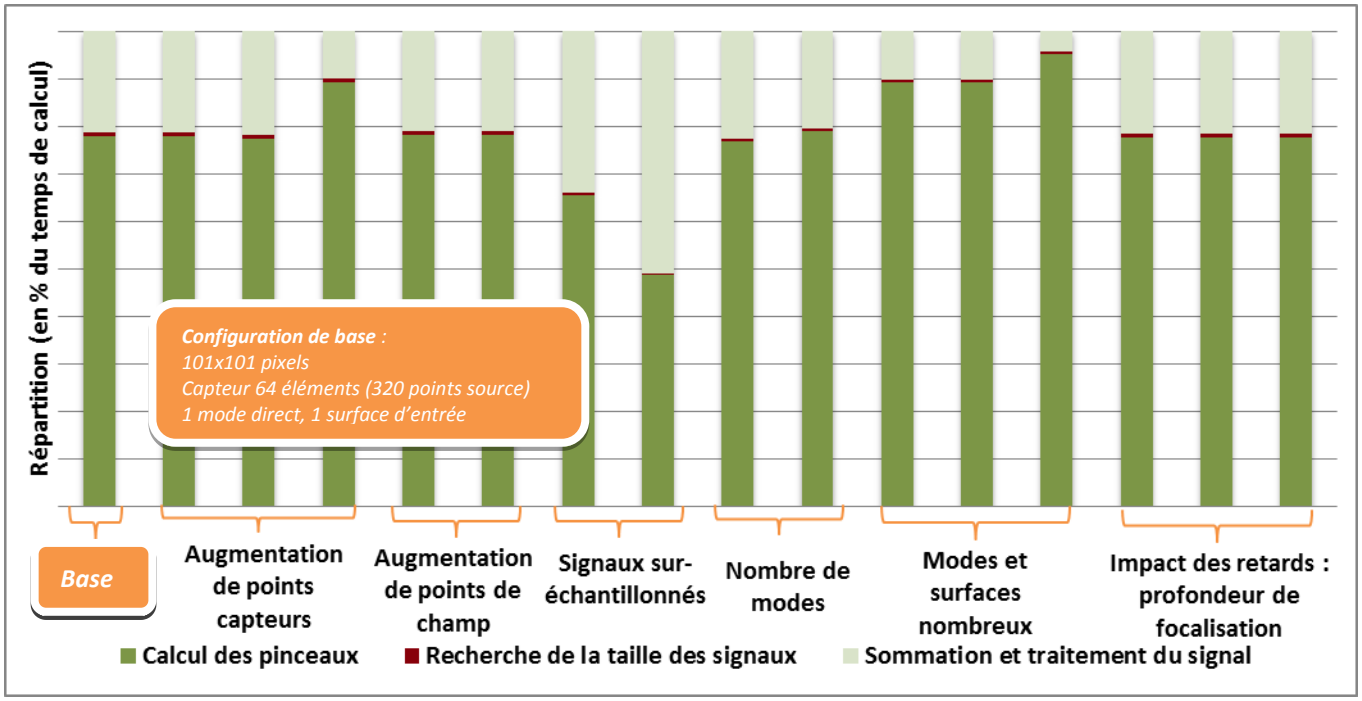

Figure 6 Répartition du temps de calcul total (en %)

L'enjeu des optimisations présentées est de tirer au mieux parti des capacités matérielles : en particulier tirer parti des caches et limiter le stockage des ressources temporaires d'un côté, et de l'autre bénéficier des capacités de vectorisations disponibles sur les processeurs.Plus particulièrement, les optimisations apportées visent les architectures disposant d'instructions SIMD (*Single Instruction Multiple Data*) SSE 4.2 (registres et instructions de 128bits, 4 flottants/2 doubles). Ces instructions sont basées sur le modèle SIMD, c'est-à-dire que la même instruction va être appliquée à plusieurs données distinctes. En pratique, ces données sont regroupée dans des registres SIMD, aussi appelés *vecteurs SIMD*; le nombre de données distinctes au sein d'un de ces vecteurs étant sa *largeur*.

Calcul vectorisé des pinceaux

Principe

A partir de l'implémentation de référence, une nouvelle implémentation a été développée. Bien que permettant de gagner théoriquement une accélération de l'ordre de la largeur d'un vecteur SIMD, dans la pratique plusieurs problèmes peuvent survenir et rendre nul l'effort de vectorisation réalisé. Sans entrer dans les détails, ces écueils peuvent être :

1. la divergence de certains algorithmes entrainant le travail sur des vecteurs partiels (en masquant les données inutilisées) et entrainant le besoin d'exécuter toutes les branches parcourues par l'ensemble des données du vecteur ;
2. la nécessité, sur la plupart des architectures, d'aligner ses données en mémoire et de lire où écrire un vecteur entier en mémoire ;
3. sur certaines architectures, certaines instructions ne sont pas disponibles et leur absence doit être contournée ;
4. sur certaines architectures, les vecteurs SIMD entiers et flottants n'ont pas la même largeur, entrainant un surcout en cas d'utilisation simultanée des deux types de données (par exemple un vecteur d'indices dans un tableau et des vecteurs de calculs).

Appliqué au cas du calcul des pinceaux, les instructions SIMD sontutilisées pour calculer simultanément plusieurs pinceaux. Afin d'éviter la majorité des écueils du SIMD, les mesures suivantes sont prises :

1. Le calcul des pinceaux sera vectorisé entre un point et plusieurs éléments du capteur, pour un mode et un couple de surface donnée. Ainsi, les pinceaux sont suffisamment proches pour minimiser le risque de divergence qui peut survenir aux différents niveaux de l'algorithme (résolution numérique du trajet ; angle du pinceau sur la surface pouvant être sur ou sous critique).
2. Les données des éléments du capteur sont stockées de manière à être alignées en mémoire (structure de tableaux).
3. et 4. La vectorisation a été réalisée au moyen de la librairie Boost.SIMD [2] fournissant une surcouche aux instructions vectorielles et détectant leur présence ou non sur une architecture donnée pour fournir un équivalent scalaire en cas d'absence, permettant une stabilité numérique sur des plateformes variées ;

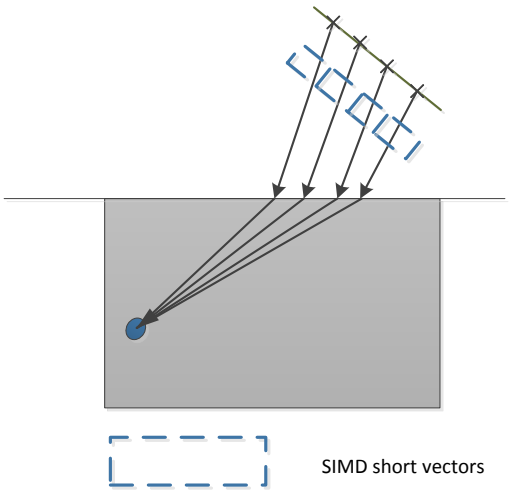

Figure 7 Vectorisation du calcul des pinceaux

Résultats

La vectorisation du calcul des pinceaux permet d'obtenir, sur l'ensemble des configurations, une accélération de l'ordre de x3,9 à x4,3 par rapport aux précédentes versions scalaires.

Sommation des contributions des pinceaux et traitement du signal

Principe

Cette phase du calcul de champ se décompose en plusieurs sous étapes consécutives.

- Les contributions des pinceaux sommées sur les réponses impulsionnelles de déplacement (en prenant en compte les lois de retards éventuels appliquées aux éléments d'un capteur multiélément). Il y a ainsi trois réponses impulsionnelles sur les coordonnées dans le domaine réel et trois réponses impulsionnelles sur les coordonnées imaginaires.
- Les réponses impulsionnelles sont convoluées avec le signal de référence (convolution de la partie réelle, transformée de Hilbert de la partie imaginaire).
- Le module du déplacement est obtenu par la norme 2.
- L'enveloppe du module est calculée et son maximum est recherché, c'est l'amplitude du champ de déplacement retenue pour former l'image de champ. Le temps de vol associé contribue à l'image de temps de vol.

Sur l'ensemble de ces étapes, un travail de vectorisation a été appliqué afin d'en améliorer les performances, en particulier les étapes de convolution, du calcul du module et du calcul de l'enveloppe. La vectorisation de la phase de sommation sera détaillée dans un paragraphe ultérieur de cet article.

Résultats

Sur l'ensemble des configurations, les performances de ces différentes étapes sont mesurées indépendamment ce qui permet de mettre en lumière les accélérations obtenues sur chaque étape de calcul.

Table 1 Performances de la vectorisation du traitement du signal

	Configuration base	**Jeu de configurations**
Convolution (sans FFT)	x 9,4	x 10,5
Calcul du module	x 4,4	x 3,1
Enveloppe (avec FFT MKL)	x 1,5	x 1,8

Sur ces étapes de traitement du signal, le calcul des Transformations de Fourier Rapide (FFT), nécessaire aux convolutions et aux calculs d'enveloppe, fait appel à la bibliothèque *Intel Math Kernel Library* (MKL) qui fournit une implémentation optimisée de ces opérations pour les architectures des processeurs Intel. C'est pourquoi les performances du calcul de l'enveloppe présentent des gains en deçà du gain théorique de 4 (vecteurs SIMD SSE 4.2 et calculs flottants simple précision) : le gain ne concerne qu'une sous partie du temps de calcul de l'enveloppe ; les opérations de FFT étant déjà optimisées par la MKL ne s'accélèrent pas plus.

Le calcul du module se comporte bien, avec un gain proche du gain théorique.

Enfin, le calcul de convolution utilise beaucoup d'opérations inter registre de type « *shuffle* » pour mélanger les éléments des registres entreeux; ces opérations sont très rapide et permettent ainsi de mélanger les informations de plusieurs registres entreeux efficacement. Ces opérations rendent particulièrement efficace cette phase de calcul convolution avec un gain sur-linéaire.

La vectorisation des traitements des signaux de la seconde partie du calcul de champ a donné de très bons résultats, tirant parti des capacités des instructions SIMD.

Vectorisation de sommations

La seconde étape du calcul de champ nécessitant beaucoup de calculs est la sommation des différents pinceaux afin de former la réponse impulsionnelle, en chaque point de champ, tout en prenant en compte les retards imposés. Plusieurs approches sont envisageable pour vectoriser le calcul.

Sommation « horizontale » des signaux

Une première approche consiste à vectoriser la sommation d'un pinceau à la fois. Les vecteurs SIMD sont utilisés « horizontalement » : les différents éléments de ces derniers contiennent les données correspondantes à des échantillons temporels contigus. Pour un point de champ, une boucle va parcourir l'ensemble des pinceaux reliant ce point de champ au capteur. Pour chaque pinceau, la sommation se fait « horizontalement », avec un calcul scalaire des informations du pinceau retardé, avant de les répartir dans une succession de vecteurs SIMD qui sont ajoutés à la réponse impulsionnelle résultat. Le code vectorisé se préoccupe alors principalement des calculs de bornes temporelles pour vérifier que les informations du pinceaudans les vecteurs SIMD soient alignés en mémoire.

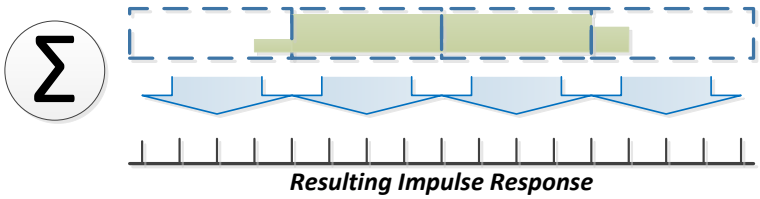

Figure 8 Sommation "horizontale"

Sommation « verticale » des signaux

Une seconde approche, transposée de la précédente, utilise des vecteurs SIMD non plus « horizontalement » mais « verticalement » par rapport au signal résultat : chacun de ces vecteurs est alors utilisé pour représenter des contributions de multiples pinceaux aux réponses impulsionnelles, pour un pas de temps donné.

Ainsi, dans cette implémentation, pour un point de champ, les informations concernant plusieurs pinceaux sont obtenues dans un ensemble de registres SIMD. On va ensuite parcourir l'ensemble des pas de temps concernés par les pinceaux « actifs » afin de créer un vecteur contenant les informations correspondantes au pas de temps traité : un zéro lorsque le pinceau de ne contribue pas, sa contribution sinon. Puisque la contribution des différents pinceaux est additive, on procèdera à une réduction des éléments du vecteur SIMD pour obtenir la valeur scalaire à ajouter au signal résultat. Le code SIMD s'attache à calculer les contributions de plusieurs pinceaux en même temps.

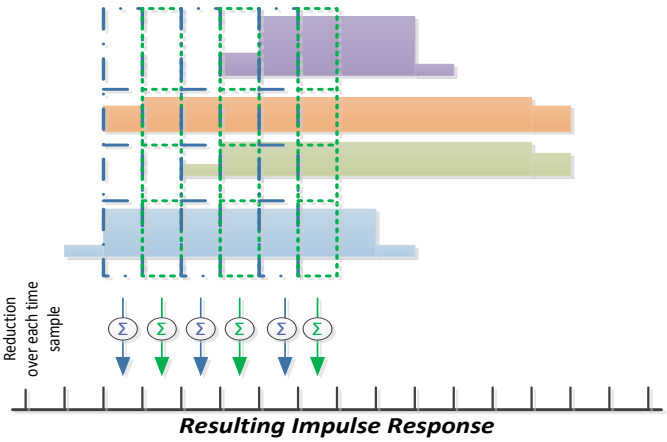

Figure 9 Sommation "verticale"

Performances

Ces deux algorithmes se montrent moins performants que le code scalaire. La sommation horizontale s'effectue, en moyenne, sur l'ensemble des configurations dans le même temps que le calcul scalaire. La sommation verticale présente des performances moindres, allant jusqu'à en moyenne 2.8x plus lentement que le code scalaire.

Ces deux algorithmes nécessitent beaucoup de traitements pour transformer les informations scalaires des pinceaux en registres SIMD pour la sommation (les calculs ne sont pas réguliers, avec d'importants calculs entier de bornes sur la réponse impulsionnelle, variables selon les pinceaux). L'algorithmie résultant est peu adaptée aux calculs SIMD.

Des améliorations sont actuellement à l'étude afin de travailler sur les données pour retrouver la régularité qui serait utile aux calculs vectorisés. Plusieurs pistes sont à l'étude, avec entre autre des prétraitements pour formater les données différemment ; l'utilisation d'un tri préalable des pinceaux afin de les présenter dans l'ordre optimal pour chaque stratégie de sommation.

De plus, sur architectures GPU, l'algorithme d'un premier portage [3] présente beaucoup de synchronisations sur une algorithmie proche de la sommation horizontale, or l'approche SIMT est obligatoire par conception : il est possible d'espérer un gain avec l'algorithmie verticale (qui ne présente plus de synchronisations).

Algorithmie du calcul par point en une étape

Une dernière piste d'optimisation a été étudiée afin de regrouper les traitements, point de champ par point de champ, pour la parallélisation multithread. L'objectif est de faire calculer à un thread le champ en un point, depuis les pinceaux jusqu'à l'obtention de l'amplitude finale. Cette approche est moins couteuse en mémoire utilisée, puisque les données temporaires ne concernent plus qu'un seul point de champ par thread et sont tout de suite jetées après utilisation. De plus, en réduisant les traitements entre chaque phase, cette approche vise à une meilleure réutilisation des caches (les pinceaux sont tout de suite sommés) tout en augmentant le grain du parallélisme (plus de traitements par threads).

Une telle approche est transverse à toutes les optimisations présentées dans ce document, ainsi il est possible de comparer des versions scalaires et vectorisées d'une telle algorithmie. En fonction de la configuration et du type d'instruction, des gains de performances variant de 10 à 30% sont observés.

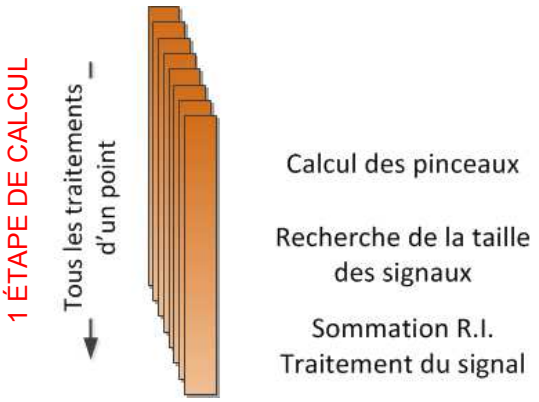

Figure 10 Algorithme en une étape

Synthèse des résultats

En combinant la vectorisation des calculs et l'algorithmie par point de champ, il est possible de former une version de l'algorithme performante dont les résultats sont présentés dans la Table 2.

Machine 2x12 cœurs architecture Ivy Bridge		Config Base		Gain SIMD	Moyenne / ∀Config		Gain SIMD
		Scalaire	SIMD		Scalaire	SIMD	
Algo de référence	Calcul des pinceaux	136,7 ms	31,8 ms	x4,3	434,7 ms	106,3 ms	x4,1
	Taille Signaux	2,0 ms	2,0 ms	x1,0	5,7 ms	5,5 ms	x1,0
	Sommation + Tr. du sig.	46,0 ms	29,5 ms	x1,6	128,7 ms	70,7 ms	x1,8
	Total algo référence	184,7 ms	63,3 ms	x2,9	569,1 ms	182,5 ms	x3,1
	Algo. en une étape	162,5 ms	**48,9 ms**	x3,3	525,3 ms	**161,9 ms**	x3,2
		x1,1	x1,1	**x3,7**	x1,1	x1,1	**x3,5**

Table 2 Synthèse des performances

Horizontalement, il est possible de lire, pour les différentes étapes, les gains obtenus par la vectorisation sur les différentes étapes de calcul de champ. Verticalement, les gains obtenus par la parallélisation par point de champ sont récapitulés pour les versions scalaires et vectorielles du code. Enfin, en rouge sont donnés les temps de la version la plus rapide combinant les deux approches, ainsi que l'accélération obtenue par rapport au code de référence initial.

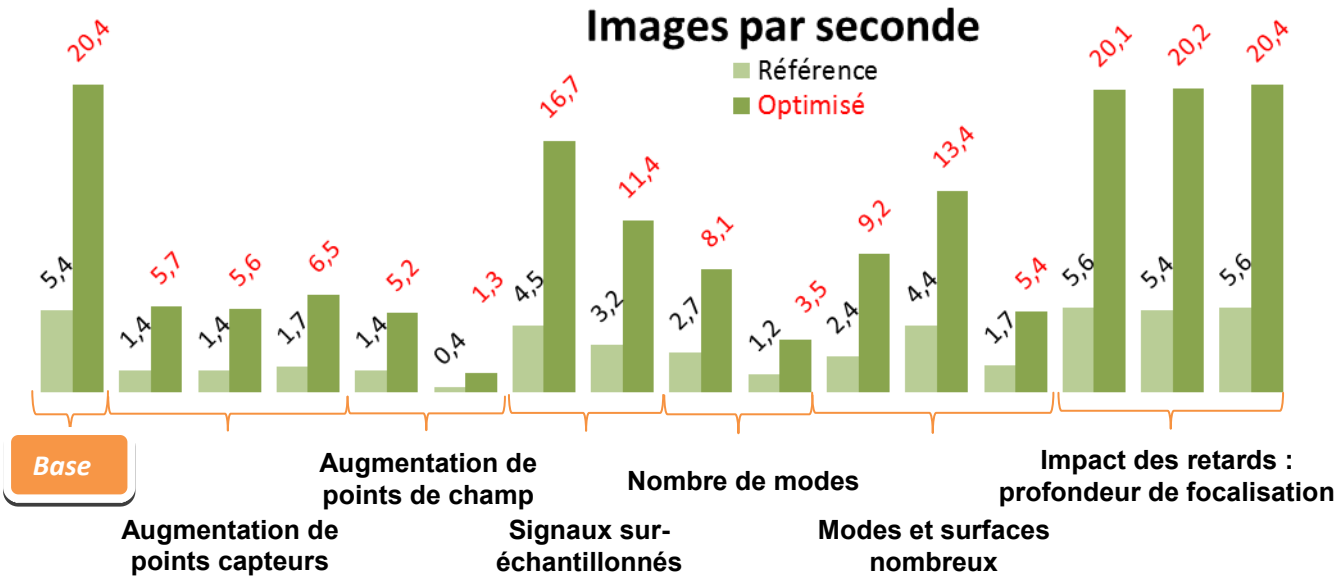

Figure 11 Performances en images par seconde sur l'ensemble des configurations sur 2xE5-2697v2

Enfin, le graphiqueFigure 11 présente les performances, en images par seconde, de l'algorithme de référence et de l'algorithme optimisé, afin d'observer si les performances interactives sont atteintes. Avec plus de 20 images par secondes sur plusieurs configurations sur une machine 2x12 cœurs @2.70 GHz (2x Xeon E5-2697v2), la simulation de champ interactive est atteinte.

Conclusion

Partant d'une implémentation de référence, validée par rapport à CIVA 11, d'un modèle restreint en domaine d'application mais présentant des qualités de régularité, des optimisations ont été apportées pour tendre vers une simulation interactive.

Sur l'ensemble des étapes un travail a été réalisé pour vectoriser le code et bénéficier des instructions SIMD au moyen de la librairie Boost.SIMD, en particulier des instructions SSE 4.2 sur architecture

Intel x86. Les performances des vectorisations du calcul des pinceaux et des étapes de traitement du signal sont bonnes. Cependant, l'étape de la sommation des contributions des pinceaux sur les réponses impulsionnelles de déplacement ne tire pas parti des instructions SIMD et une étude est en cours afin d'en déterminer l'origine. La combinaison de la vectorisation et de la parallélisation du calcul de champ par point de champ permet d'obtenir des performances interactives, avec plus de 20 images par secondes sur la configuration basique de champ.

Ce travail va servir de socle a de futurs développements permettant d'obtenir des performances encore plus importantes sur la simulation de contrôle non destructif ultrasonore, de champ et d'écho, et d'espérer obtenir des calculs interactifs sur un spectre de configurations encore plus large.Tout d'abord, l'utilisation de vecteurs SIMD plus longs, AVX et AVX2 (256bits), permettrai d'améliorer à nouveau les performances des étapes de vectorisation. De plus, l'utilisation des architectures Xeon Phi semble intéressante pour tirer parti des capacités manycore de ces coprocesseurs (vecteurs SIMD de 512bits et grand nombre de threads avec plusieurs dizaines de cœurs). Un portage GPU des codes des simulations de champ est prévu afin de bénéficier des capacités massivement parallèles de ces cartes. Enfin, un travail similaire sera réalisé sur les codes de simulation d'écho pour atteindre l'interactivité en réutilisant les résultats obtenus sur le calcul de champ [4] et [5].

Le domaine d'application sera étendu (réflexion avec conversion de mode, gestion de surfaces non planes). Les outils ainsi développés seront intégrés dans la prochaine version majeure de la plateforme CIVA.

Bibliographie

[1] N. Gengembre. Pencil method for ultrasonic beam computation. In *World Congress on Ultrasonics*, page 1533-1536, Paris, France, 2003.

[2] Pierre Estérie, Mathias Gaunard, Joel Falcou, Jean-Thierry Lapresté, and Brigitte Rozoy. Boost.simd: Generic programming for portable simdization. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, pages 431–432, New York, NY, USA, 2012. ACM.

[3] J. Lambert, L. Lacassagne, S. Le Berre, G. Rougeron, and S. Chatillon. High performance simulation of ultrasonic fields for non destructive testing. In *Proceedings of the Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013),* 2013.

[4] J. Lambert, G. Rougeron, L. Lacassagne, and S. Chatillon. A fast ultrasonic simulation tool based on massively parallel implementations. 2013.

[5] D. Romero-Laorden, O. Martinez-Graullera, C. J. Martn, M. Pérez, and L. G. Ullate. Field modelling acceleration on ultrasonic systems using graphic hardware. *Computer Physics Communications*, 182 (3): 590–599, 2011.