



Le logiciel Echo_light : Optimisation et Validation

Nathalie Bartoli

► To cite this version:

Nathalie Bartoli. Le logiciel Echo_light : Optimisation et Validation. RT-0277, INRIA. 2003.
<inria-00071207>

HAL Id: inria-00071207

<https://hal.inria.fr/inria-00071207>

Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

*Le logiciel Echo_light :
Optimisation et Validation*

Nathalie BARTOLI

N° 0277

Mars 2003

_____ THÈME 4 _____

 *apport
technique*



Le logiciel Echo_light : Optimisation et Validation

Nathalie BARTOLI*

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Caiman

Rapport technique n° 0277 — Mars 2003 — 39 pages

Résumé : Le logiciel d'électromagnétisme Echo_light permet de coupler plusieurs géométries axisymétriques en utilisant les formulations intégrales. Ce document regroupe les modifications effectuées dans le code Echo_light d'Alcatel Space : optimisation et parallélisation des différents scripts shell.

Mots-clés : équations intégrales, calcul itératif, géométries pluri-axisymétriques

Travail effectué dans le cadre d'un contrat entre l'Inria-Cermics et Alcatel Space

* INRIA- CERMICS

The logiciel `Echo_light`: Optimisation and Validation

Abstract: The software `Echo_light` is used in electromagnetism to couple some axisymmetrical geometries by using integral formulations. This document reports all modifications done to improve the code `Echo_light` of Alcatel Space: optimization and parallelization of shell scripts

Key-words: integral equations, iterative computation, multi-axisymmetrical geometries

1 Contexte de l'étude

Alcatel Space dispose de plusieurs logiciels de calcul de rayonnement électromagnétique. Nous nous intéressons en particulier à deux logiciels qui ont pour but de résoudre les équations de Maxwell en régime harmonique dans un espace de dimension 3 dans le cas particulier d'objets à symétrie de révolution. Nous nommerons les logiciels `Mono-axi` et `Echo_light`.

`Mono-axi` permet de calculer le rayonnement de plusieurs objets en dimension 3 avec un même axe de révolution, c'est un code axisymétrique. La méthode utilise les équations intégrales, ce qui permet de réduire d'une dimension d'espace la taille du problème, et dans ce cas particulier d'axisymétrie, seul un maillage 1D de la génératrice de l'objet est nécessaire.

`Echo_light` couple plusieurs éléments rayonnants axisymétriques mais dont les axes de symétrie peuvent être différents : c'est un code pluri-axisymétrique. L'algorithme est un processus itératif au cours duquel les objets émettent et reçoivent des ondes jusqu'à l'obtention d'un régime harmonique couplé. Le point de départ d'`Echo_light` est donné par des formulations intégrales indépendantes, calculées par `Mono-axi` pour chaque axe de symétrie ; il s'agit ensuite, au fil des itérations, de faire rayonner les éléments entre eux. Le couplage entre différents éléments rayonnants est étudié de manière théorique dans [2].

`Mono-axi` servira de code de référence lors des validations.

Une partie de mon travail post-doctoral concerne l'accélération de la convergence du couplage des différentes formulations intégrales en proposant une version optimisée de l'algorithme `Echo_light` dans le but d'étendre son utilisation. Ce travail est présenté dans ce document et comprend quatre grandes parties :

1. la présentation et l'état des lieux du logiciel `Echo_light` en février 2002
2. l'optimisation de l'algorithme
3. les premières validations et la convergence de l'algorithme optimisé
4. la parallélisation du script et le cluster de PC.

Cette étude a été menée en étroite collaboration avec Christelle Roubion du Département Recherche d'Alcatel Space.

2 Présentation du logiciel Echo_light

A mon arrivée (février 2002), j'ai récupéré une première version du logiciel `Echo_light` avec différents cas tests de validation. Cette version du logiciel est **mono-source, mono-itérative et non optimisée**. Deux types de cas tests ont été proposés :

- une source (guide d'onde alimenté par un seul mode) et plusieurs objets métalliques alignés selon le même axe. Les résultats en terme de puissance rayonnée sont comparés à ceux donnés par `Mono-axi`.

- une source (guide d'onde alimenté par un seul mode) et plusieurs objets diélectriques vides (permittivité $\varepsilon_r = (1,0)$ et perméabilité $\mu_r = (1,0)$). Les résultats doivent être identiques à ceux de la source isolée.

3 Evolutions du logiciel `Echo_light`

3.1 Algorithme initial

L'algorithme du logiciel `Echo_light` est présenté brièvement. Rappelons que cette version du logiciel a été validée dans le cadre du septet d'antennes cieres et a fait l'objet d'un rapport au sein d'Alcatel Space [1]. Nous avons repris les mêmes notations que celles utilisées dans ce rapport, à savoir, que les sources sont considérées comme des objets, et donc le nombre d'objets est au moins égal au nombre de sources. Dans la suite, lorsque rien ne sera précisé, le terme "objets" se rapportera à l'ensemble "sources + objets", dans le cas contraire nous utiliserons le terme "objets uniquement". Les fichiers d'entrée et de sortie du logiciel sont nombreux, nous allons en particulier exploiter le fichier qui permet de visualiser la puissance rayonnée et la phase en co et cross polarisation.

L'algorithme général d'`Echo_light` est décrit ci-après.

0: *Initialisation des maillages 2D et 3D de chaque objet dans le repère de tous les autres objets*

1: *Boucle pour i objet uniquement*

Calcul de la source

calcul du courant sur la source

calcul du champ de la source sur la grille de visualisation

Calcul de l'objet i

détermination du champ proche issu de la source sur l'objet i

démodulation du champ sur l'objet i

calcul du courant sur l'objet i

calcul du champ sur la grille de visualisation

champ proche issu de l'objet i sur la source (retour sur la source)

démodulation du champ sur la source

calcul du courant sur la source ¹

champ de la source sur la grille de visualisation ¹

1: *fin de la boucle pour i*

2: *Boucle pour i objet uniquement*

3: *Boucle pour j objet uniquement*

Influence de l'objet i sur l'objet j

champ proche issu de l'objet i sur l'objet j

démodulation du champ sur j

courant sur l'objet j généré par i

champ de l'objet j sur la grille de visualisation

Influence de l'objet j sur l'objet i

champ proche issu de l'objet j sur l'objet i

démodulation du champ sur i

courant sur l'objet i généré par j

champ de l'objet i sur la grille de visualisation

3: *fin de la boucle sur j*

2: *fin de la boucle pour i*

Sommation des champs sur la grille de visualisation

Remarque 3.1.1 *Dans ce premier algorithme, il faut souligner plusieurs points :*

- *une seule source est considérée*
- *une seule itération est effectuée*
- *le retour des objets sur la source est néanmoins pris en compte*

1. Cette contribution de la source sera prise en compte seulement dans la deuxième itération du nouvel algorithme, on se reportera à la remarque 3.2.2.

- *les courants des objets utilisés dans le calcul des interactions (influence des objets) sont ceux calculés à partir du courant sur la source, il n'y a donc pas de mise à jour des courants sur les objets.*

Le temps de calcul est effectivement très long pour différentes raisons :

- pour calculer le courant sur un objet, on calcule l'interaction de cet objet avec les autres, c'est à dire que chaque champ proche est calculé, puis démodulé.
- à partir de ce courant, on calcule le champ sur la grille de visualisation, et donc ce calcul est fait autant de fois que le nombre d'interactions objet-objet.

3.2 Algorithme proposé

Nous présentons l'algorithme sous sa forme optimisée.

0: Initialisation des maillages 2D et 3D de chaque objet dans le repère de tous les autres objets

1: Boucle sur les itérations k

Calcul sources

2: Boucle pour i source

→ si $k = 1$ on calcule le courant (k) sur la source à partir du champ incident

→ si $k \neq 1$, on met à jour le courant (k) à partir du retour sur la source ($k - 1$)

à partir du courant (k), on calcule le champ de la source sur la grille de visualisation

2: fin de la boucle sur i source: on dispose du courant (k) pour toutes les sources

Calcul objets

3: Boucle pour i source ou objet

interaction = 0

4: Boucle pour j source avec $j \neq i$

Somme des champs créés par la source j à partir des courants (k) sur l'obstacle i

interaction = interaction + 1

4: fin de la boucle sur j

5: Si le nombre interaction $\neq 0$

démodulation du champ créé par les sources sur i

calcul du courant sur i

→ si i est un objet: calcul du courant (k) puis du champ sur la grille de visualisation

→ si i est une source: on stocke l'information source-source (k)

5: fin du test sur les interactions

3: fin de la boucle pour i : on dispose $\left\{ \begin{array}{l} \text{du courant } (k) \text{ sur les objets créés par les sources} \\ \text{du courant } (k + 1) \text{ dû aux interactions entre les sources} \end{array} \right.$

Echange objets

6: Boucle pour i source ou objet

interaction = 0

7: Boucle pour j objet uniquement

Somme des champs créés par les objets j à partir des courants (k) sur l'obstacle i

interaction = interaction + 1

7: fin de la boucle sur j

8: Si le nombre interaction $\neq 0$

démodulation du champ créé par les sources sur i

calcul du courant sur i

→ si i est un objet: calcul du courant ($k + 1$) puis du champ sur la grille de visualisation

→ si i est une source: on additionne ce nouveau courant ($k + 1$) à source-source(k)

8: fin du test sur les interactions

6: fin de la boucle pour i : on dispose $\left\{ \begin{array}{l} \text{du courant } (k + 1) \text{ dû aux interactions entre objets} \\ \text{du courant } (k + 1) \text{ dû aux retours des objets sur les sources} \end{array} \right.$

Somme des champs sur la grille de visualisation

1: fin de la boucle sur les itérations k

Remarque 3.2.1 Dans cette première optimisation, nous avons essayé d'inclure plusieurs points :

- le choix du nombre d'itérations. Pour cela, nous avons pris la convention suivante :
chaque émission de la source définit le départ d'une itération.
- le choix du nombre de sources
- pour calculer le courant sur un objet : on fait la liste des objets qui interagissent avec celui-ci, on calcule et on somme les différentes contributions (maillage 3D), on effectue la démodulation du champ proche pour revenir à la formulation en 2D-axi, on a ainsi le second membre de notre système que l'on inverse pour déterminer le courant
- les courants sont mis à jour à la fin de chaque itération pour l'itération suivante
- les champs sur la grille de visualisation sont sommés au fur et à mesure des interactions entre les objets et des itérations
- un effort a été fait sur la simplification des commandes shell.

Un dernier point est à préciser concernant la comparaison des deux algorithmes.

Remarque 3.2.2 L'algorithme initial est en fait une itération de l'algorithme optimisé, auquel il faut ajouter le retour sur la source. En effet, ce retour n'est pris en compte dans le nouvel algorithme que lors de la deuxième itération.

Cette remarque est à garder en mémoire lors de la comparaison des temps de calcul entre les deux algorithmes.

4 Utilisation et lancement du nouvel algorithme

Le script principal nécessite des arguments d'appel supplémentaires par rapport à la version initiale :

4 arguments obligatoires

- le nombre de sources
- le nombre d'objets (incluant le nombre de sources)
- le nombre d'itérations
- un entier 0 ou 1 qui permet de visualiser ou non les champs juste après le retour sur la source, dans le but de comparer les courbes avec l'algorithme initial

1 argument optionnel qui gère la fin des itérations

- 0 si on veut effectuer toutes les itérations sans utiliser le critère d'arrêt, ce critère sera détaillé un peu plus loin.
- 1 (par défaut) si on arrête les itérations dès que la convergence est atteinte, ou dès que le nombre d'itérations demandées a été effectué.

5 Validation du nouvel algorithme

5.1 Une source isolée

La source est modélisée par un cornet à symétrie de révolution représenté sur la figure 1. Elle est alimentée par un mode guidé, elle éclaire alors en polarisation 1 au mode 1.

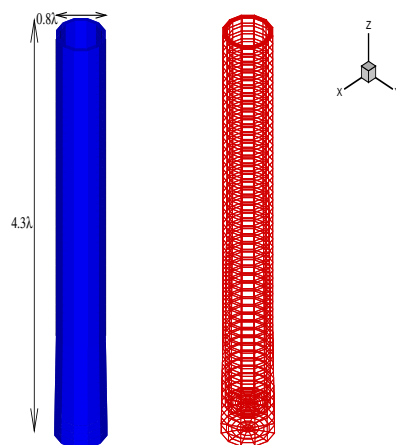


FIG. 1 – *Géométrie et maillage de la source .*

Pour cette configuration, nous présentons deux courbes relatives à la puissance rayonnée et à la phase obtenues par le code de référence `Mono-axi`. Ces courbes où la source est isolée, seront les références pour les validations suivantes.

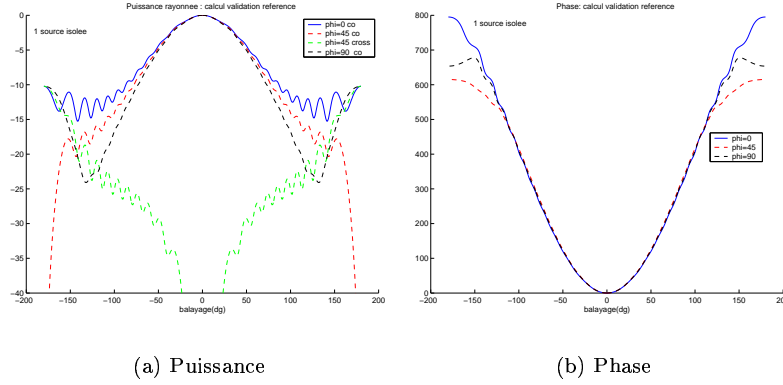


FIG. 2 – Courbes de référence pour la source isolée.

5.2 Une source et un objet diélectrique vide

Nous reprenons la source précédente et nous ajoutons sur le même axe de révolution \hat{z} un objet faussement diélectrique, c'est à dire de permittivité égale à 1. La géométrie est représentée sur la figure 3.

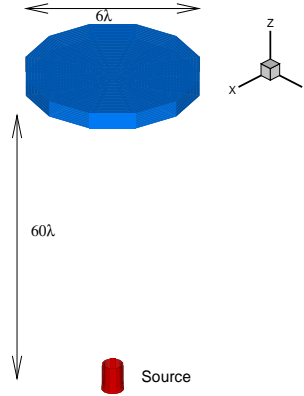


FIG. 3 – Géométrie des deux objets.

En théorie, cette configuration est identique à la configuration avec la source isolée. C'est précisément cela que nous voulons vérifier au niveau numérique. La puissance rayonnée

représentée sur la figure 4 est bien identique à la courbe de référence. La courbe relative à la phase est différente car le calcul effectué dans chacun des deux logiciels utilise une routine différente : il faudrait modifier le dérouleur de phase utilisé dans *Echo_light*. Nous ne nous sommes pas attardés sur ce point pour le moment : dans tout ce qui suit, les courbes de phase sont simplement données pour information et ne sont pas à comparer avec la courbe de référence de *Mono-axi*.

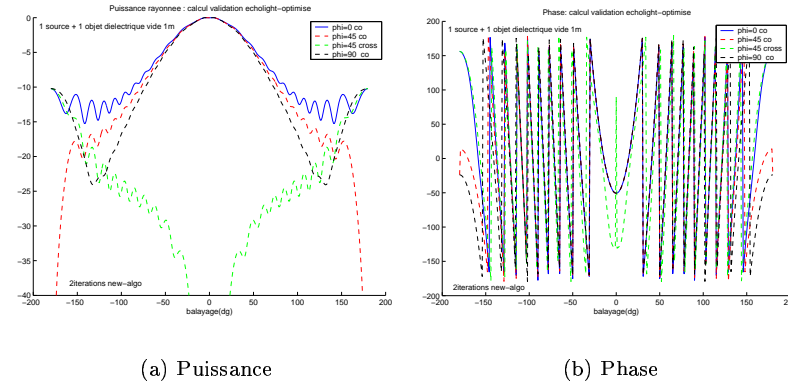


FIG. 4 – Convergence après deux itérations vers la solution de la source isolée.

Au fur et à mesure du processus itératif, c'est à dire pour l'itération courante K , nous vérifions si la convergence est atteinte ou non, en calculant le rapport des champs sur la grille de visualisation à partir du critère :

$$\frac{\max |E_k|}{\sum_{\ell=1}^{\ell=K} \max |E_{\ell}|} \text{ pour } k = 1..K \text{ itérations} \quad (1)$$

ou encore, pour avoir ce rapport en dB

$$20 \log \left\{ \frac{\max |E_k|}{\sum_{\ell=1}^{\ell=K} \max |E_{\ell}|} \right\}. \quad (2)$$

Ce critère est calculé à la fin de chaque itération, à l'exception de la première itération, et nous décidons en fonction d'un paramètre de seuil de poursuivre ou non les itérations.

Nous interprétons ce critère à partir du tableau suivant, dans le cas des deux objets.

	rapport/dB
itération 1	-9.5^{-5} dB
itération 2	-232 dB

TAB. 1 – Nombre d'itérations et convergence pour deux objets.

Au travers de cet exemple, seule la première itération suffit pour retrouver la courbe de référence. La contribution apportée par l'itération 2 est totalement négligeable, de l'ordre de -200 dB.

Remarque 5.2.1 *Un test d'arrêt a été mis en place : si à la fin de l'itération ce rapport est inférieur à 0.05 (fixé dans le script), soit une contribution de l'ordre de -60 dB, le script stoppe les itérations. Il est possible via un argument supplémentaire en entrée du script de poursuivre les itérations jusqu'au bout, indépendamment de la valeur de ce critère de convergence.*

5.3 Une source et deux objets diélectriques vides

Nous reprenons la configuration suivante en ajoutant un deuxième objet faussement diélectrique comme l'indique la figure 5.

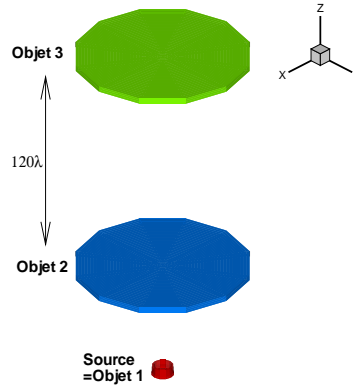


FIG. 5 – Géométrie des trois objets.

Dans cet exemple, nous devons retrouver la courbe de référence de la source isolée. Cette vérification est faite à partir de la courbe 6 relative à la puissance.

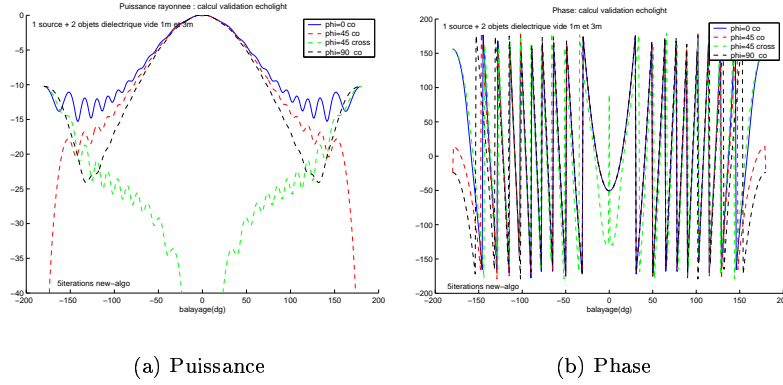


FIG. 6 – Convergence après cinq itérations vers la solution de la source isolée.

Le tableau 2 montre la convergence au fur et à mesure des 5 itérations. Au delà de la première itération, la contribution des champs est négligeable sur la grille de visualisation, largement inférieure à -200 dB.

	rapport/dB
itération 1	-7.5^{-5} dB
itération 2	-235 dB
itération 3	-468 dB
itération 4	-701 dB
itération 5	-935 dB

TAB. 2 – Nombre d'itérations et convergence pour trois objets.

Le temps nécessaire pour effectuer 5 itérations est de 4h 13. Remarquons que ce temps peut être largement réduit en n'effectuant qu'une seule itération puisque quasiment aucune information n'est apportée par les itérations suivantes.

Remarque 5.3.1 Tous les tests de validation sont effectués sur un environnement DEC et principalement sur dec1 et dec2, qui sont deux machines mono-processeur dont les caractéristiques sont données ci-après.

DEC DXP1000 21264	dec1	dec2
fréquence	600 Mhz	660 Mhz
mémoire	128 Moctets	1 Goctets

TAB. 3 – Caractéristiques des machines DEC.

Tous les temps CPU seront donnés sous forme relative par rapport au temps de référence de l'exemple considéré.

5.4 Une source et des objets décentrés

Nous modifions la configuration précédente en déplaçant les objets. La source reste à la même position et les objets 2 et 3 sont déplacés de part et d'autre de l'axe de révolution de la source comme le montre la figure 7.

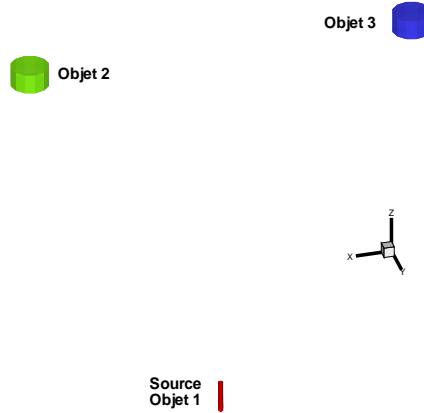


FIG. 7 – *Géométrie des objets désaxés.*

Sur la figure 8, les résultats obtenus sont conformes à notre attente, nous retrouvons la puissance rayonnée par la source isolée dans les deux cas de figure traités :

- la source et l'objet 2
- la source et les objets 2 et 3.

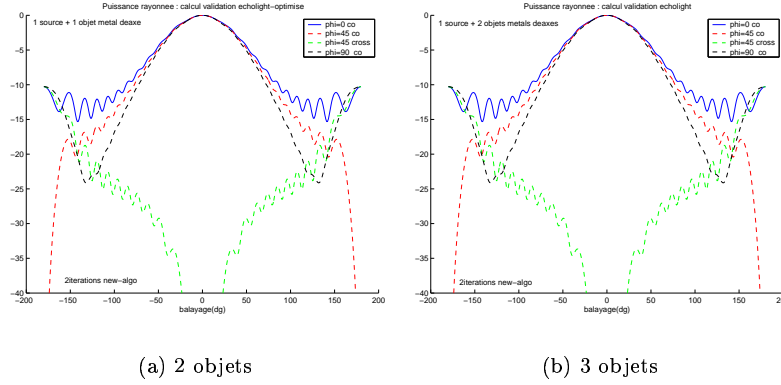


FIG. 8 – Convergence après deux itérations vers la solution de la source isolée.

	2 itérations
2 objets désaxés	12h 21min
3 objets désaxés	44h 38min

TAB. 4 – Temps CPU pour deux ou trois objets désaxés.

Pour ces deux cas tests les temps de calcul sont très longs : le nombre de modes nécessaires pour développer le champ sous forme de série est largement supérieur lorsque les axes de révolution des objets ne sont plus colinéaires. Si 2 modes suffisaient dans les exemples précédents, nous utilisons ici entre 19 et 45 modes suivant le champ à reconstituer.

Ces premiers exemples n'ont pas de grand intérêt physique sinon de valider le code `Echo_light` avec plusieurs objets pour différentes positions. Nous poursuivons la validation avec des exemples plus réalistes constitués d'objets métalliques.

5.5 Une source et un objet métallique

On considère à présent la source et un objet métallique sur un même axe, identiques à la géométrie de la figure 3. L'intérêt d'avoir un même axe de symétrie pour les deux objets est de disposer d'une courbe de référence donnée par `Mono-axi`. La puissance et la phase de référence sont représentées sur la figure 9.

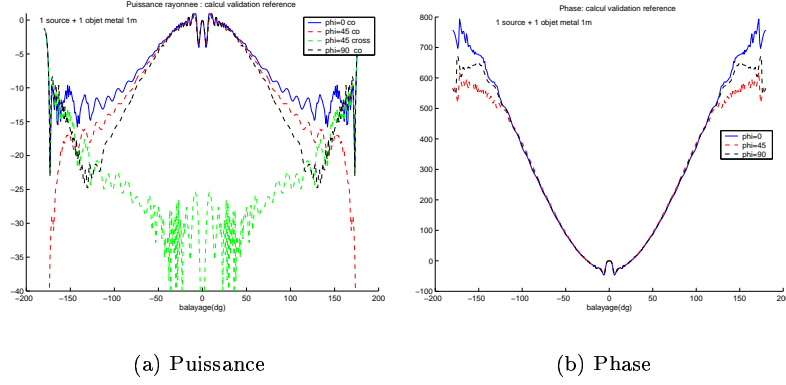


FIG. 9 – *Solution de référence pour deux objets avec Mono-axi.*

Après deux itérations, les courbes obtenues avec *Echo_light* ressemblent parfaitement aux courbes de référence comme le montre la figure 10. Nous avons reporté dans les tableaux 5 et 6 la contribution de chacune des itérations et le temps CPU.

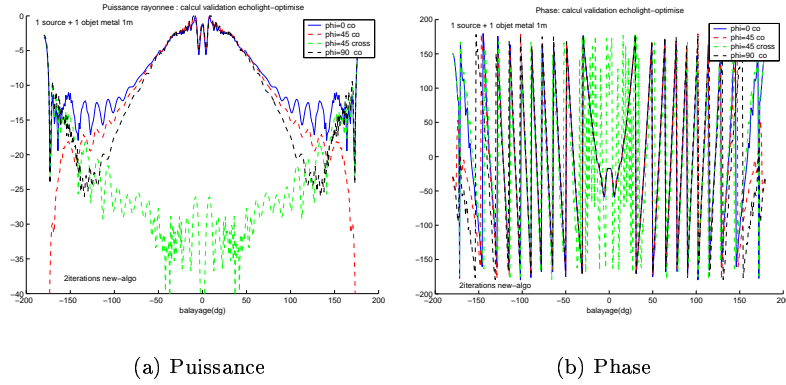


FIG. 10 – *Convergence après deux itérations vers la solution de Mono-axi.*

	rapport/dB
itération 1	-0.15 dB
itération 2	-69 dB

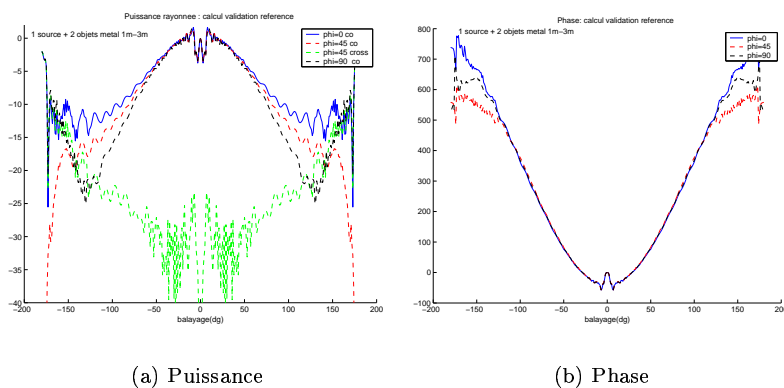
TAB. 5 – Nombre d'itérations et convergence pour deux objets.

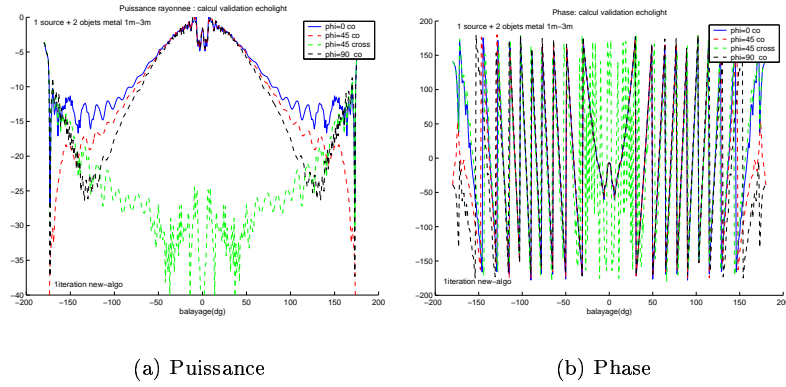
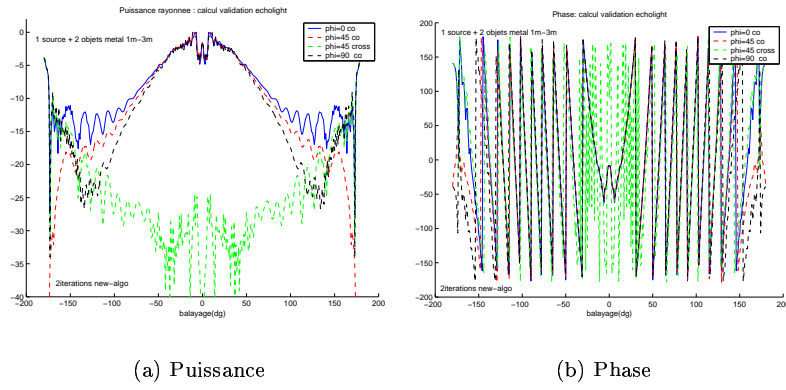
Deux objets	2 itérations
	31min

TAB. 6 – Temps CPU pour deux objets.

5.6 Une source et deux objets métalliques

Nous reprenons la géométrie présentée à la figure 5 qui couple la source à deux objets métalliques. La solution de référence est donnée par `Mono-axi` et nous regardons les courbes obtenues avec `Echo_light` après une ou plusieurs itérations.

FIG. 11 – Solution de référence pour trois objets avec `Mono-axi`.

FIG. 12 – *Convergence après une itération.*FIG. 13 – *Convergence après deux itérations.*

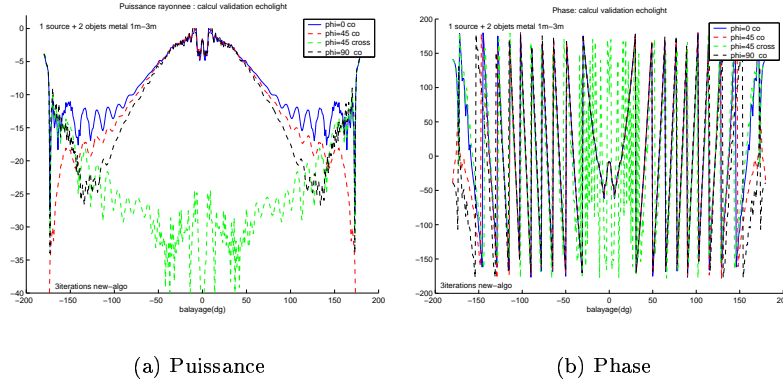


FIG. 14 – Convergence après trois itérations.

	rapport/dB
itération 1	-0.1 dB
itération 2	-71 dB
itération 3	-146 dB

TAB. 7 – Nombre d'itérations et convergence pour trois objets.

Trois objets	1 itération	1 itération retour source	2 itérations	3 itérations
	t	$1.38t$	$2.38t$	$3.76t$

TAB. 8 – Temps CPU pour trois objets.

Le tableau 8 reporte les temps CPU de manière relative en fonction du nombre d'itérations imposées et de l'algorithme utilisé. Précisons que

- 1, 2 ou 3 itérations sont obtenues avec le nouvel algorithme `Echo_light`
- 1 itération avec le retour sur la source est imposé par l'algorithme initial (version non optimisée), se reporter à la remarque 3.2.2.

Si nous comparons par exemple la courbe de co-polarisation pour $\varphi = 0$, selon le critère calculé en norme L^2 entre deux itérations l et k ($k > l$)

$$\text{erreur relative sur le champ } E \text{ en } \% = 100 \frac{\|E(k) - E(l)\|_2}{\|E(k)\|_2}$$

alors nous avons

erreur relative entre itération 1 et itération 3= 0.0036%
 erreur relative entre itération 2 et itération 3= 0.0001%

ce qui nous conforte sur la convergence du logiciel `Echo_light`.

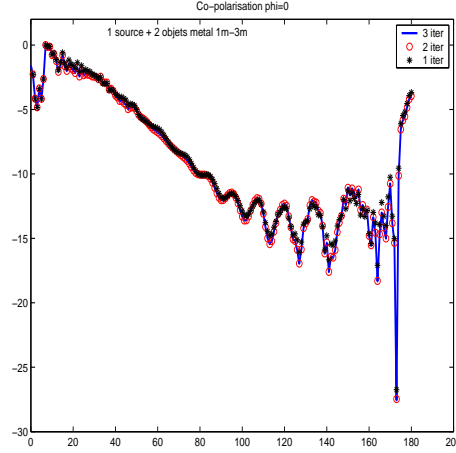
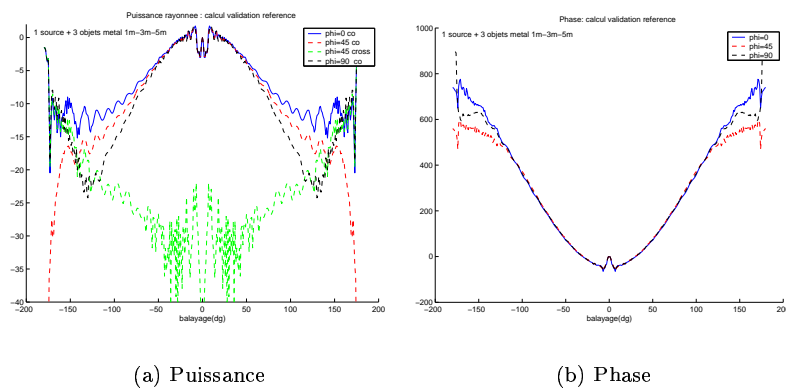
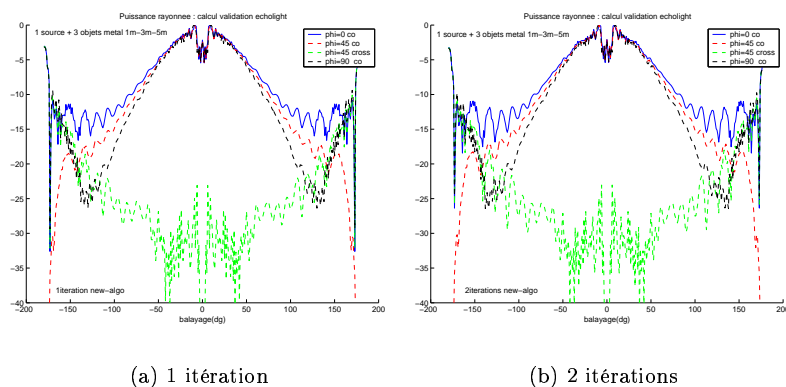


FIG. 15 – Convergence de la courbe de co-polarisation $\varphi = 0$.

5.7 Une source et trois objets métalliques

A la géométrie présentée à la figure 5, nous rajoutons un objet métallique sur l'axe \hat{z} à 5 mètres de la source. Comme précédemment, nous comparons les résultats de `Mono-axi` et d'`Echo_light`.

FIG. 16 – *Solution de référence pour trois objets avec Mono-axi.*FIG. 17 – *Puissance rayonnée en fonction des itérations dans le cas de 4 objets.*

	rapport/dB
itération 1	-0.11 dB
itération 2	-70 dB

TAB. 9 – *Nombre d'itérations et convergence pour quatre objets.*

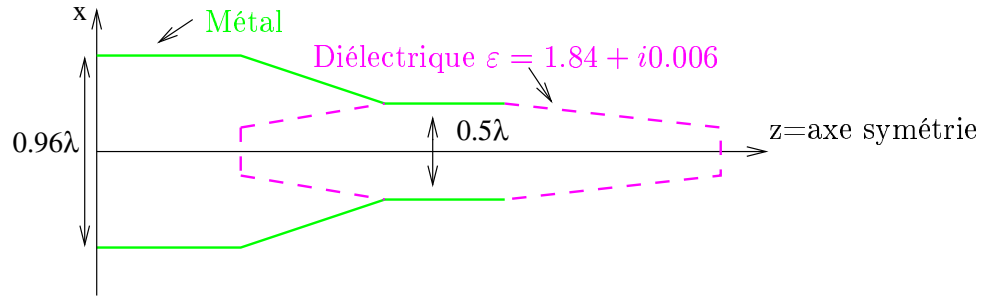
Quatre objets	1 itération	1 itération retour source	2 itérations
	t	$1.43t$	$2.29t$

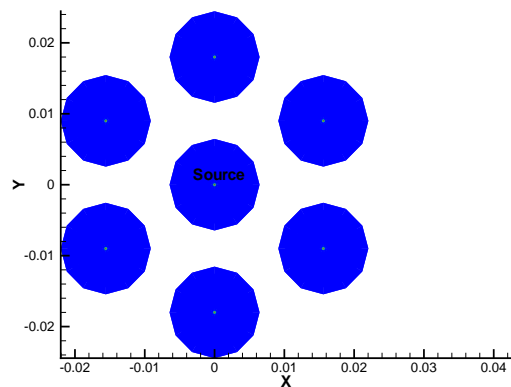
TAB. 10 – *Temps CPU pour quatre objets.*

De la même manière que dans le tableau 8, le tableau 10 compare l'ancien et le nouvel algorithme.

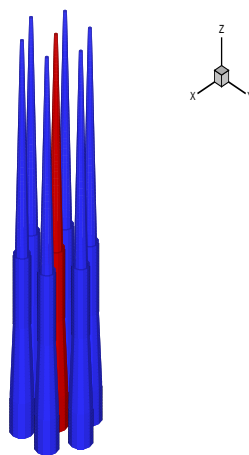
5.8 Le septet avec une alimentation monosource

Le septet est composé de sept antennes cieres dont la forme est représentée en deux dimensions sur la figure 18. Une antenne cierge est placée au centre et les six autres antennes sont positionnées en cercle autour de la première comme le montre la figure 19. Seule l'antenne centrale est alimentée. La fréquence est de 20.2 GHz et la distance centre à centre entre la source et les bougies environnantes est de 18mm soit 1.2 longueur d'onde.

FIG. 18 – *Schéma d'une antenne cierge.*

FIG. 19 – *Coupe des 7 antennes cierges.*

L'antenne centrale qui est alimentée apparaît sur la figure 20 avec la couleur rouge.

FIG. 20 – *Géométrie des 7 antennes cierges.*

Ce dispositif a fait l'objet d'une campagne de mesures réalisées par Alcatel Space en 2000. Nous souhaitons approcher au mieux les courbes de mesure de la figure 21 et comparer les temps de calcul avec la première version d'Echo_light.

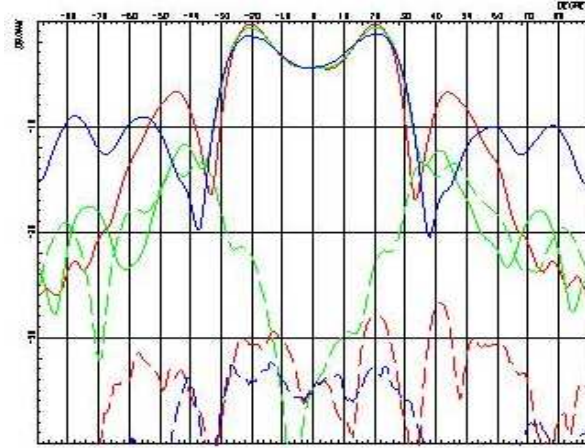


FIG. 21 – *Mesures du septet d'antennes.*

La légende relative aux courbes de mesure de la figure 21 est donnée ci-dessous.

- rouge pour le plan de coupe $\phi=0$, en co-polarisation avec le trait continu et en cross-polarisation avec le trait pointillé
- vert pour le plan de coupe $\phi=45$ en co-polarisation avec le trait continu et en cross-polarisation avec le pointillé
- bleu pour le plan de coupe $\phi=90$ en co-polarisation avec le trait continu et en cross-polarisation avec le trait pointillé

La correspondance des couleurs est la même pour les résultats des simulations avec `Echo_light` sur les figures 22 et 23. Les trois courbes en co-polarisation sont données, par contre pour la polarisation croisée, seule la puissance rayonnée dans le plan $\phi=45$ est représentée.

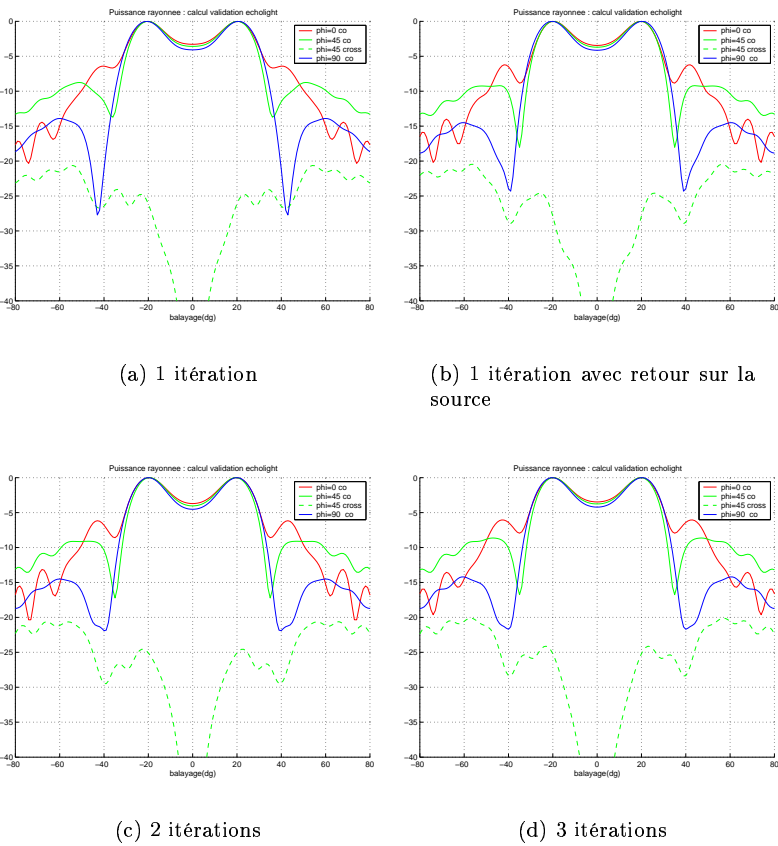


FIG. 22 – Puissance rayonnée par le septet monoalimé.

On remarque que la courbe relative à la co-polarisation dans le plan $\phi=90$ (couleur bleue, trait continu) à tendance à se rapprocher de la courbe des mesures au fur et à mesure des itérations, notamment au niveau des angles de balayage -40° et 40° où le rayonnement mesuré est de l'ordre de -20dB .

	rapport/dB
itération 1	-0.4 dB
itération 2	-45 dB
itération 3	-57 dB

TAB. 11 – Nombre d'itérations et convergence pour le septet.

Nous observons dans le tableau 11 que notre critère d'arrêt est vérifié après 3 itérations. Cependant, si nous regardons la puissance rayonnée entre les angles -80 et 80 degrés de manière à comparer les courbes avec les mesures effectuées (figure 21), nous remarquons sur la figure 23 qu'une seule itération suffit déjà pour retrouver des courbes proches de celle des mesures. Dans ce cas, effectuer une deuxième itération n'apporte pas beaucoup plus d'informations et surtout coûte très cher en temps de calcul : sur le cluster de PC, il faut environ 2.3 fois plus de temps pour deux itérations. Les temps CPU sont détaillés un peu plus loin dans le tableau 17. Dans la suite du manuscrit, nous ferons varier la distance entre la source et les bougies environnantes pour étudier le comportement de la convergence.

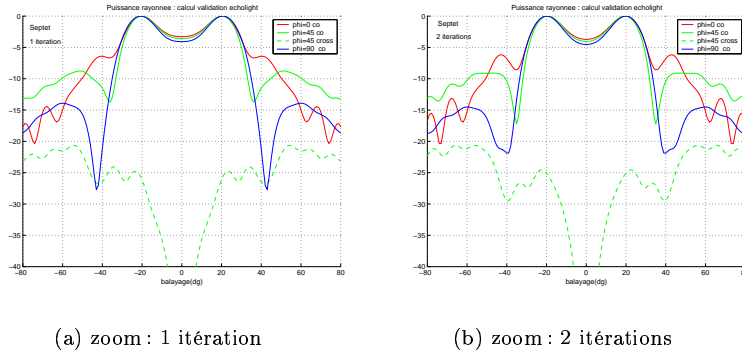


FIG. 23 – Puissance rayonnée par le septet monoalimenté (zoom sur l'intervalle $[-80^\circ, 80^\circ]$).

5.9 Une source et un objet métallique relativement proches

Nous reprenons un cas test du manuel [1]. Il s'agit de deux objets relativement proches représentés sur la figure 24. La source et le cylindre sont distants de 0.1m soit 6 longueurs d'onde pour une fréquence fixée à 18.4Ghz. Par comparaison aux cas tests précédents, les objets sont maillés avec 15 points par longueur d'onde (10 points auparavant) et le cylindre est dix fois plus proche de la source. Rappelons qu'une divergence avait été observée dans le rapport précédent entre les courbes de `Mono-axi` et `d'Echo_light`.

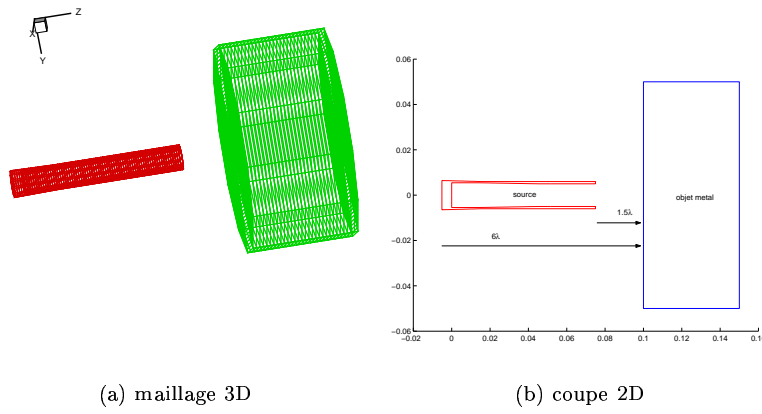


FIG. 24 – Géométrie des deux objets proches.

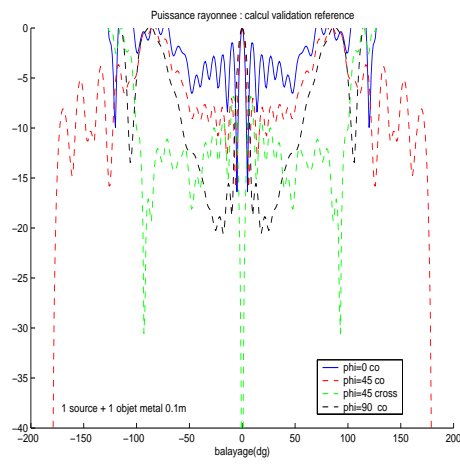


FIG. 25 – Solution de référence pour les objets proches avec Mono-axi.

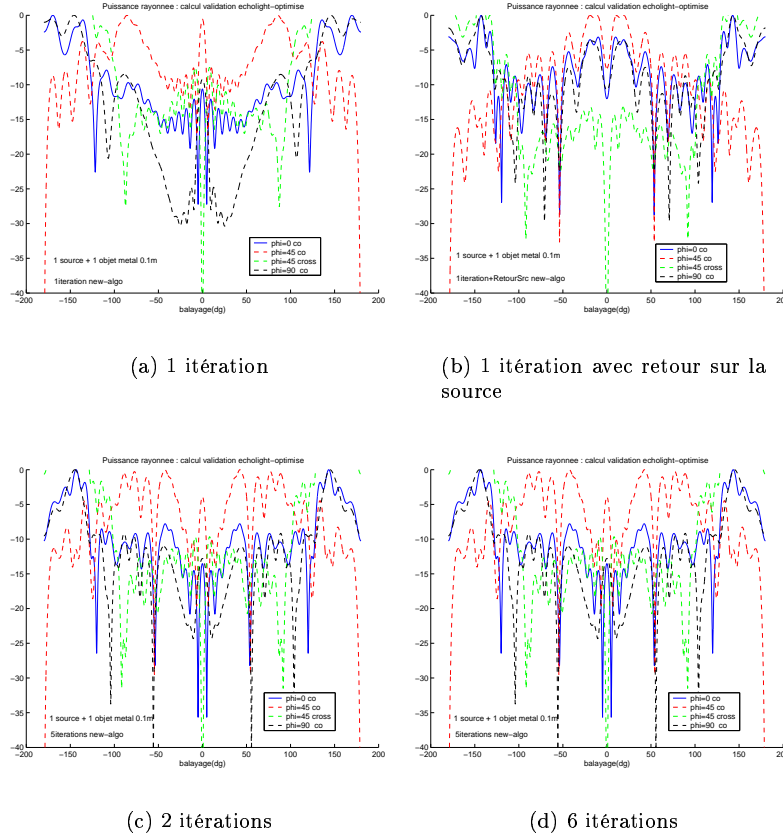


FIG. 26 – *Solution fournie par Echo_light au fur et à mesure des itérations.*

	rapport/dB
itération 1	-2.5 dB
itération 2	-0.3 dB
itération 3	-15 dB
itération 4	-32 dB
itération 5	-48 dB
itération 6	-64 dB

TAB. 12 – *Nombre d'itérations et convergence pour deux objets.*

A la vue de ces résultats, nous remarquons que la figure 26-b est identique à celle obtenue dans le manuel [1]. On peut également affirmer que la convergence a lieu, les courbes après 2 ou 6 itérations sont quasiment identiques et le tableau 12 montre que la contribution des itérations successives est de plus en plus négligeable. Cependant, les résultats d'Echo_light restent très éloignés des courbes de référence 25. Il y a deux explications à cela :

- la première est du domaine de la physique. Le guide d'onde et l'objet sont très proches, moins de deux longueurs d'onde les séparent comme l'indique la figure 24, ils se comportent alors comme deux plaques métalliques d'un condensateur où, au fur et à mesure des réflexions (quasiment totales), les lignes de champ deviennent parallèles. Le problème est donc infini et ne peut être résolu que globalement avec Mono-axi.
- la deuxième explication concerne la modélisation du problème. Etant donnée la faible distance qui sépare la source et l'objet, l'objet est capable de renvoyer des ondes de même fréquence que la source à l'intérieur du guide d'onde. L'alimentation est donc modifiée au fur et à mesure des itérations et le modèle actuel n'en tient pas compte.

L'algorithme Echo_light ne peut donc pas être appliqué dans une telle situation.

5.10 Une source et un éclateur

Pour valider notre algorithme sur deux objets proches, on considère une source et un éclateur comme l'indique la figure 27. Le choix de cette géométrie a été fait de manière à éviter les problèmes de condensateur rencontrés précédemment.

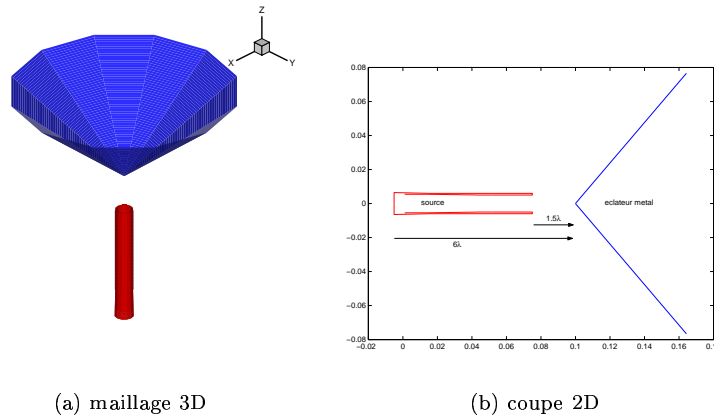


FIG. 27 – Géométrie de l'éclateur.

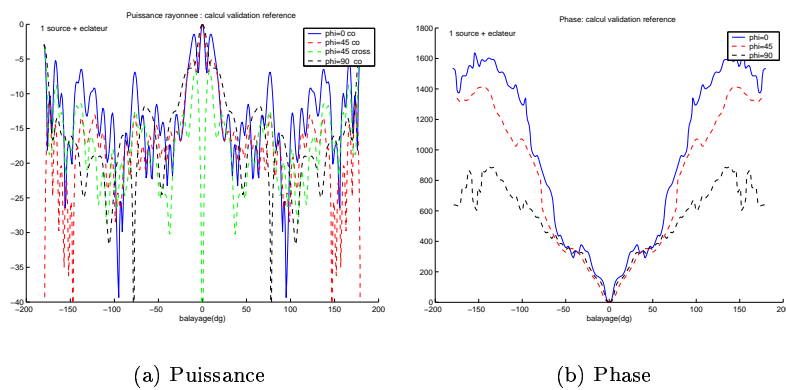
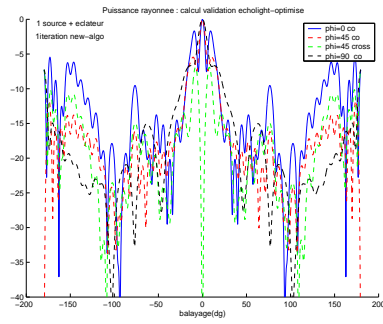
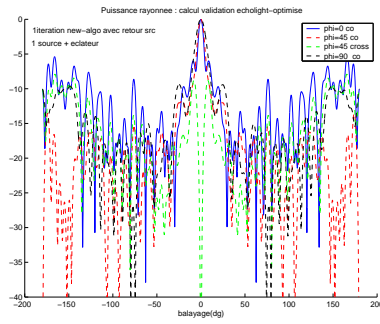


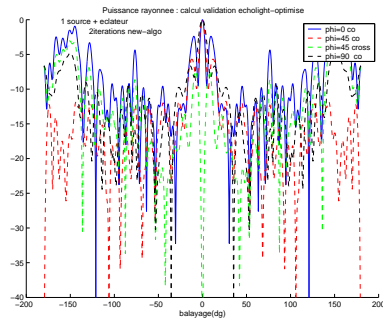
FIG. 28 – *Solution de référence pour l'éclateur avec Mono-axi.*



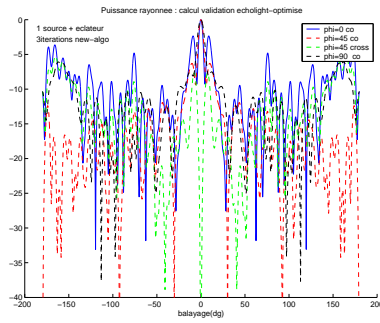
(a) 1 itération



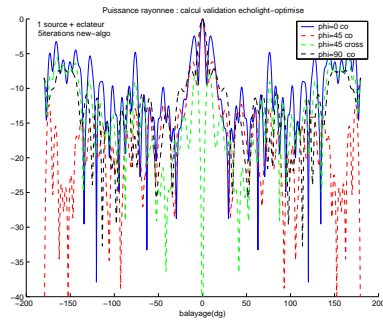
(b) 1 itération avec retour sur la source



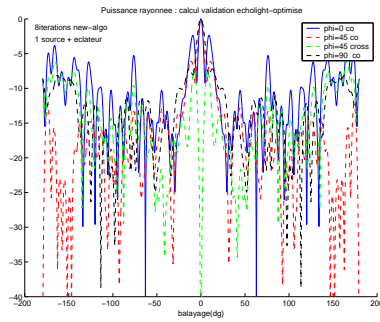
(c) 2 itérations



(d) 3 itérations



(e) 5 itérations



(f) 8 itérations

FIG. 29 – Solution fournie par Echo_light au fur et à mesure des itérations.

	rapport/dB
itération 1	6.62 dB
itération 2	-6 dB
itération 3	-16 dB
itération 4	-25 dB
itération 5	-36 dB
itération 6	-47 dB
itération 7	-57 dB
itération 8	-67 dB

TAB. 13 – *Nombre d'itérations et convergence pour deux objets.*

Deux objets	3 itérations	5 itérations	8 itérations
	t	$1.76t$	$2.92t$

TAB. 14 – *Temps CPU pour deux objets.*

Sur cet exemple où les deux objets sont très proches, la convergence est plus difficile à s'établir. Cependant, au delà de la deuxième itération, la puissance rayonnée ne varie plus beaucoup comme le montre la figure 29.

5.11 Etude de la convergence en fonction de la distance entre les objets

Nous souhaitons étudier la convergence du logiciel en fonction de la distance source-objet. A partir de l'exemple du septet d'antennes, nous choisissons deux antennes cieres avec une alimentation mono-source. La fréquence est fixée à 20.2 GHz. Nous allons faire varier la distance centre à centre entre les deux bougies, notée d sur la figure 30 et relever le nombre d'itérations nécessaires pour vérifier le critère de convergence (2) : les itérations s'arrêtent dès lors que la contribution est inférieure à -60 dB. Nous calculons ce critère dans les trois plans de coupes principaux $\phi = 0^\circ$, $\phi = 45^\circ$ et $\phi = 90^\circ$ et les valeurs sont reportées dans le tableau 15.

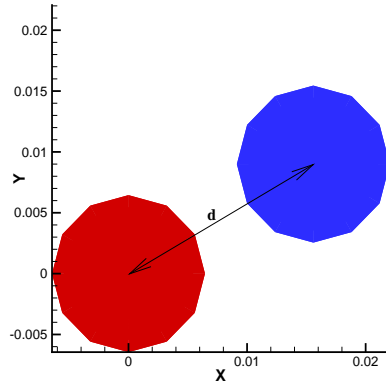


FIG. 30 – Géométrie des 2 antennes cieres.

Deux antennes cieres		$d = 1.2\lambda$	$d = 2.4\lambda$	$d = 3.6\lambda$	$d = 7.2\lambda$
Nombre d'itérations		2	2	2	2
Itération 1	$\phi = 0^\circ$	0.036 dB	-0.004 dB	-0.007 dB	0.005 dB
	$\phi = 45^\circ$	0.030 dB	-0.005 dB	-0.007 dB	0.005 dB
	$\phi = 90^\circ$	0.049 dB	-0.004 dB	-0.007 dB	0.005 dB
Itération 2	$\phi = 0^\circ$	-54.03 dB	-79.83 dB	-90.05 dB	-105.6 dB
	$\phi = 45^\circ$	-66.55 dB	-96.56 dB	-102.86 dB	-122.81 dB
	$\phi = 90^\circ$	-82.52 dB	-132.29 dB	-148.03 dB	-161.96 dB

TAB. 15 – Convergence en fonction de la distance d entre les deux antennes cieres.

Sur la figure 31, nous avons tracé la contribution de la deuxième itération en fonction de la distance d pour les trois plans de coupe principaux. Cette contribution est d'autant plus négligeable que la distance d est élevée : la convergence est alors beaucoup plus rapide.

De par sa simplicité, ce cas test nous a permis de valider la convergence de notre algorithme et de justifier le choix de notre critère d'arrêt. En fonction des applications et de la précision requise, le seuil de -60 dB imposé pour l'arrêt des itérations se doit d'être modifié.

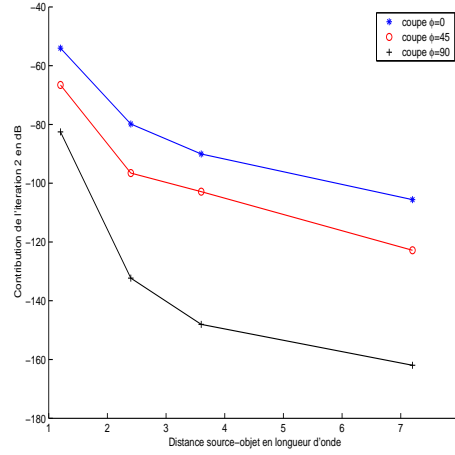


FIG. 31 – *Convergence en fonction de la distance entre les 2 antennes cieres.*

5.12 Conclusion sur la validation du logiciel Echo_light

Nous avons proposé une version multi-sources, itérative et optimisée du logiciel `Echo_light`. Cette nouvelle version a été testée sur de nombreux exemples extraits de [1]. La convergence a été observée, même si toutefois dans la plupart des situations une ou deux itérations suffisent pour retrouver la solution de référence. Certains points restent encore à régler comme la différence de phase entre `Mono-axi` et `Echo_light` ou encore les temps de calcul trop longs. Avant de considérer d'autres cas tests plus réalistes, il paraît donc primordial de s'intéresser à l'accélération d'`Echo_light`.

6 Parallélisation du script principal d'Echo_light

6.1 Un nouvel algorithme

L'idée consiste à accélérer le script d'Echo_light en utilisant les ressources de plusieurs machines. Nous allons donc paralléliser les boucles dans lesquelles les calculs sont indépendants. Par exemple le calcul des courants et des champs créés par chacune des sources est indépendant des autres calculs. Jusqu'à présent nous utilisions les ressources d'une seule machine avec l'algorithme suivant :

- Calcul sources à l'itération (k) :

1: *Boucle pour i source*

on appelle un code fortran pour calculer le courant sur chaque source

1: *fin de la boucle sur i source* : on dispose du courant (k) pour toutes les sources

- Calcul sources parallélisé à l'itération (k), on utilise autant de machines que de sources :

1: *Boucle pour i source*

on lance le calcul de la source i sur la machine j

1: *fin de la boucle sur i source* : on dispose du courant (k) pour toutes les sources

Si les ressources informatiques le permettent, l'idéal est de disposer d'autant de machines différentes que d'objets que l'on veut faire rayonner, la parallélisation est alors optimale.

6.2 Comparaison des temps de calcul

Les premiers tests ont été effectués sur l'environnement DEC. Nous disposons dans le projet de 4 machines DEC et nous avons donc pu comparer les temps CPU dans le cas du rayonnement d'une source avec 3 objets alignés sur un axe à 1m, 3m et 5m respectivement, en choisissant de traiter un objet par machine.

	version séquentielle	version parallèle
2 objets, 2 itérations	t_1	t_1
3 objets, 2 itérations	t_2	$0.73t_2$
4 objets, 1 itération	t_3	$0.53t_3$
4 objets, 2 itérations	$2.29t_3$	$1.19t_3$

TAB. 16 – Temps CPU sur les machines DEC.

Lorsque l'on considère deux objets, nous n'observons aucune différence entre la version séquentielle et la version parallèle. Ce résultat était prévisible car dans cette situation aucun calcul n'est fait en même temps. La version parallèle devient très intéressante lorsque le nombre d'objets augmente. Un facteur 2 sur le temps CPU est observé dans le cas des 4

objets. Nous n'avons pas pu considérer davantage d'objets car le nombre de machines DEC disponibles ne nous le permettait pas.

6.3 L'utilisation du cluster de PC

Pour répondre à l'intérêt d'Alcatel Space sur le cluster de PC, nous avons testé le code parallèle sur le cluster de l'INRIA qui se compose de

- 19 PC bipro Pentium III à 933 Mhz
- 14 PC bipro Pentium III à 500 Mhz

reliés par un réseau Fast-Ethernet à 100 Mbit/s. Pour plus d'informations techniques sur le système, on pourra se reporter à l'adresse <http://www.inria.fr/sophia/parallel/>.

Dans ce cadre, l'utilisateur fournit le nombre et la liste des noeuds du cluster qu'il souhaite utiliser, puis lance sa requête. Une option dans le script permet de laisser le choix des noeuds au logiciel LSF. Nous rappelons que chaque noeud se compose de deux processeurs, et que le nombre de processeurs est en général fixé au nombre d'objets de manière à utiliser la parallélisation de manière optimale.

	version parallèle
2 objets, 2 itération, 2 processeurs	t
4 objets, 2 itérations, 4 processeurs	$2.12t$
septet 7 objets, 1 itération, 7 processeurs	$28.25t$
septet 7 objets, 2 itérations, 7 processeurs	$65.59t$
septet 7 objets, 3 itérations, 7 processeurs	$100.93t$

TAB. 17 – Temps CPU sur le cluster de PC.

Pour deux et quatre objets, les temps CPU du tableau 17 sont comparables à ceux de la version `rsh` sous environnement DEC, cf tableau 16. Pour donner un ordre d'idée du gain obtenu via l'utilisation du cluster de PC, il faut considérer davantage d'objets : l'exemple du septet permet d'obtenir un facteur 4 entre la version séquentielle et la version parallèle.

7 Optimisation de la routine de champ rayonné

Le calcul des champs proches E et H se fait initialement par l'appel consécutif de deux routines que l'on notera `champ - e` et `champ - h`. Chacune de ces routines détermine respectivement les coefficients E_m et H_m de la série de Fourier, pour chacune des deux polarisations. L'idée est de remplacer ces deux routines par un unique appel de la routine `champ`. On évite ainsi de redéfinir deux fois les noyaux de Green, les points de Gauss, les variables relatives à chaque élément de la frontière ..., cependant on doit dédoubler certaines variables (des scalaires complexes) en fonction de leur dépendance à E ou à H . L'amélioration du temps de

calcul est très nette : le tableau 18 reporte le temps CPU pour le calcul des champs proches avec l'utilisation de une ou de deux routines de champs rayonnés.

	2 routines	1 routine
exemple réduit : 2 modes de Fourier	t_1	$0.64t_1$
exemple 2 antennes cierges : 12 modes de Fourier	t_2	$0.64t_2$

TAB. 18 – Temps CPU pour le calcul du champ proche.

Au travers de ces deux exemples, on s'aperçoit que l'utilisation de l'unique routine `champ` permet un gain de 1.5 sur le temps CPU du calcul de champ proche, de plus ce gain est indépendant de la taille du problème et du nombre de modes de Fourier.

Nous considérons maintenant 2 itérations de l'algorithme `Echo_light` de manière à illustrer la part du temps consacré aux calculs des champs proches. Nous reprenons l'exemple de la section 5.6 avec 3 objets de même axe de symétrie : 1 source et deux objets métalliques situés respectivement à 1 m et 3m. Le temps de calcul pour effectuer 2 itérations de `Echo_light` a été noté avec la version initiale, et ce temps est comparé à celui de la nouvelle version du logiciel où une seule routine est appelé pour le calcul des champs proches.

1 source + 2objets métalliques : 2 modes de Fourier	2 routines	1 routine
1 itération	t_1	$0.71t_1$
2 itérations	t_2	$0.68t_2$

TAB. 19 – Temps CPU pour 2 itérations de `Echo_light`.

On note sur cet exemple un gain de 1.39 pour une itération, de 1.45 pour deux itérations grâce à la nouvelle routine `champ`. Le temps nécessaire au calcul des champs proches représente une part non négligeable du temps CPU global. Le facteur 1.5 obtenu pour le calcul du champ proche se retrouve à peu près sur le temps global de `Echo_light` où on observe à présent un gain moyen de 1.4.

septet	2 routines	1 routine
1 itération	t_1	$0.61t_1$
2 itérations	t_2	$0.60t_2$

TAB. 20 – Temps CPU pour 2 itérations de `Echo_light`.

Nous considérons un dernier exemple pour illustrer l'amélioration du temps de calcul. Le septet, constitué de 7 antennes cierges dont une source, représente l'exemple le plus coûteux en terme de temps CPU que nous ayons traité. La nouvelle routine de calcul de champ proche permet de diviser le temps CPU par un facteur de 1.6 pour une ou deux itérations de `Echo_light`, voir le tableau 20.

8 Conclusion

Nous avons présenté dans ce document le logiciel `Echo_light` avant et après les modifications que nous avons apportées. A présent, le logiciel permet de considérer plusieurs sources, d'effectuer plusieurs itérations pour atteindre la convergence en fournissant un critère d'arrêt. Une part importante du travail a été consacrée à l'amélioration des temps de calcul. Pour cela, nous avons modifié l'algorithme puis optimisé les scripts, une version de script parallèle a été développée et validée sur différentes architectures : dec, linux et en particulier sur le cluster de PC. Au regard des premiers tests, le gain obtenu est important : dans le meilleur des cas le temps de calcul est divisé par un facteur 2. Cependant, si l'on examine le temps d'exécution de chacune des tâches, une grande partie est dépensée par la routine de calcul de champ proche. Cette routine calcule l'interaction entre deux objets sur un maillage tridimensionnel, ce qui explique son coût élevé : une modification dans la structure de la routine a permis de diviser son temps d'exécution par 1.5 et de réduire d'autant le temps CPU global de `Echo_light`.

Nous arrêtons là nos investigations sur l'optimisation des scripts et des programmes fortran pour nous consacrer davantage à cette routine. Nous espérons développer une nouvelle méthode plus rapide pour calculer l'interaction entre deux objets suffisamment éloignés, basée sur les techniques multipolaires.

Table des matières

1	Contexte de l'étude	3
2	Présentation du logiciel Echo_light	3
3	Evolutions du logiciel Echo_light	4
3.1	Algorithme initial	4
3.2	Algorithme proposé	6
4	Utilisation et lancement du nouvel algorithme	8
5	Validation du nouvel algorithme	9
5.1	Une source isolée	9
5.2	Une source et un objet diélectrique vide	10
5.3	Une source et deux objets diélectriques vides	12
5.4	Une source et des objets décentrés	14
5.5	Une source et un objet métallique	15
5.6	Une source et deux objets métalliques	17
5.7	Une source et trois objets métalliques	20
5.8	Le septet avec une alimentation monosource	22
5.9	Une source et un objet métallique relativement proches	26
5.10	Une source et un éclateur	29
5.11	Etude de la convergence en fonction de la distance entre les objets	32
5.12	Conclusion sur la validation du logiciel Echo_light	34
6	Parallélisation du script principal d'Echo_light	35
6.1	Un nouvel algorithme	35
6.2	Comparaison des temps de calcul	35
6.3	L'utilisation du cluster de PC	36
7	Optimisation de la routine de champ rayonné	36
8	Conclusion	38

Références

- [1] A. Lahure-Lecompte. Echo_light_n, manuel de l'utilisateur. Technical report, Alcatel Space Industries, 2001.
- [2] Véronique Tirel. *Hybridation par méthode de décomposition de bord appliquée à l'étude des ondes acoustiques et électromagnétiques. Résultats théoriques et numériques*. PhD thesis, Université Paris XIII, 1998.



Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-0803