Rapport de stage

William de Vazelhes

12 novembre 2015

Table des matières

Chapitre 1

Introduction

1.1 la CT : formation des données, rayons X, détecteurs

L'imagerie médicale est une science qui s'applique à obtenir des images des organes internes et autres éléments des organismes vivants, afin d'aider les diagnostics et les opérations chirurgicales. La tomographie est un type d'imagerie médicale dans lequel on cherche à obtenir une image 3D d'organes. L'objet de cet article est la tomographie à rayons X : elle s'effectue en faisant une série de radios à rayons X (projections) tout autour du patient. Les rayons X sont des rayons énergétiques, ils passent à travers les tissus, sont atténués par ces derniers, et arrivent sur des détecteurs numériques ici, laissant une projection dans laquelle chaque pixel est la somme des atténuations (liée à la densité) des points contenus sur le rayon source-pixel considéré. On récupère donc dans un ordinateur les différentes images en rayons X projetées. Nous verrons dans la sous-section suivante comment reconstruire l'image 3D à partir des projections. Pour faire une image en CT, il faut donc une source de rayons X, et un détecteur. La source à rayons X ainsi que le détecteur tournent autour de la zone à tomographier, de manière simultanée : le centre de rotation doit être situé sur l'axe détecteur/source. Le faisceau de rayons X peut être parallèle ou bien en forme de cône. Ici nous prendrons des rayons en forme de cône. Le mouvement de la source peut être hélicoïdal ou bien une succession de rotations décalées le long de l'axe de rotation. Nous prendrons une succession de rotations, car cela permet de conserver les invariances en polaire comme nous le verrons par la suite. Les données sont alors acquise sous forme d'un sinogramme : une matrice contenant chacune des projections effectuées.

1.2 La reconstruction :

1.2.1 Méthode générale

La reconstruction est l'étape qui consiste à appliquer des opérations sur le sinogramme acquis, de manière à retrouver la carte des atténuations du corps du patient (images des organes en 3D), qui donne le sinogramme obtenu après opérations de projection. La reconstruction est donc un problème inverse. Il existe deux grands types de reconstruction :
- la reconstruction directe : il existe une fonction mathématique, la transformée de Radon, qui prend en entrée le sinogramme et renvoie en sortie une reconstruction de l'objet reconstruite. Cette reconstruction est assez bonne, mais elle provient d'un modèle mathématique simple, qui ne tient

pas compte de tous les paramètres physiques réels apparus lors des projections (échantillonnage, bruit, polychromaticité de la source de rayons X...).
- la reconstruction itérative : cette méthode est plus lente que la précédente, mais a un gros avantage : on peut prendre en compte de nombreux paramètres physiques de la projection à rayons X. Cette méthode est itérative car à chaque itération on se rapproche un peu plus de l'image réelle, avec un pourcentage de tolérance accepté. Pour cela, on cherche à maximiser une fonction de vraisemblance qui prend en argument l'image courante estimée et renvoie un nombre qui mesure la vraisemblance que cette image soit la bonne. Maximiser cette fonction revient à minimiser une fonction f appelée critère. Ce critère est composé d'une part d'un critère d'adéquation et d'autre part d'un critère de pénalisation. Le critère de pénalisation, aussi appelé critère de régularisation, permet de s'assurer que l'image soit "cohérente" avec la réalité : on force les irrégularités (oscillations par exemple) à disparaître en prenant un critère de pénalisation qui croît lorsque les différences premières de l'image et/ou leur gradient sont trop élevés en certains points. Le critère d'adéquation mesure la vraisemblance de l'estimée à former les projections obtenues. Mathématiquement, on choisit la plupart du temps de calculer la neg log-vraisemblance du compte de photons obtenus aux projections N, sachant que l'on a l'objet estimé $\boldsymbol{\mu}$, comme ci-dessous, avec $\mathbf{J}_{adeq}$ le critère d'adéquation.

$$J_{adeq} = -\ln(\mathbf{N}|\boldsymbol{\mu}) \tag{1.1}$$

Dans notre cas, le critère peut se mettre sous une certaine forme mathématique, dont nous détaillerons les étapes. Tout d'abord, il faut établir un modèle de formation des données pour le calcul de l'adéquation des mesures aux projections de l'estimée.
Dans un premier temps, nous considérons que les rayons qui arrivent sur l'objet sont monochromatiques (une seule fréquence). La loi qui décrit l'absorption des rayons par l'objet est la loi de Beer-Lambert. Pour un fantôme continu, la loi de Beer-Lambert monochromatique est alors la suivante :

$$\lambda_i = N_0 e^{\int_\Delta -\boldsymbol{\mu}(s)ds}$$

Avec λ_i le paramètre de la loi de Poisson qui donne le compte de photons en un pixel i du détecteur, Δ_i la droite source-détecteur i, N_0 est le nombre de photons émis dans un faisceau source-détecteur (les faisceaux sont triangulaires mais ils sont tellement fins que l'on peut les considérer comme des droites). On peut cependant discrétiser cette loi en considérant une matrice de projection $\mathbf{P}$, qui appliquée à $\boldsymbol{\mu}$ donnera un vecteur $\boldsymbol{\lambda}$ dont chaque coefficient approche très raisonnablement l'intégrale. La formule précédente devient alors :

$$\boldsymbol{\lambda} = N_0 e^{-\mathbf{P}\boldsymbol{\mu}}$$

avec $\mathbf{P}$ la matrice de projection. Pour plus d'informations sur la matrice de projection, voir les travaux de Yves Goussard, et les sections **??** et **??**.

Expression du critère Lorsque l'on calcule la neg-log-vraisemblance d'obtenir le compte de photons $\mathbf{N}$ sachant que l'on a l'objet estimé $\boldsymbol{\mu}$ comme expliqué précédemment, on obtient la formule suivante :

$$J_{adeq}(\boldsymbol{\mu}) = -\log(\Pr(\mathbf{N}|\boldsymbol{\mu}))$$

Avec :

$$\Pr(\mathbf{N}_i|\boldsymbol{\mu}) = \frac{-\lambda_i^{\mathbf{N}_i}}{\mathbf{N}_i!} e^{-\lambda_i}$$

Donc, puisque les $\mathbf{N}_i$ sont indépendants, $\Pr(\mathbf{N}|\boldsymbol{\mu}) = \prod_i \Pr(\mathbf{N}_i|\boldsymbol{\mu})$:

$$J_{adeq}(\boldsymbol{\mu}) = -\log(\prod_i \frac{-\lambda_i^{\mathbf{N}_i}}{\mathbf{N}_i!} e^{-\lambda_i})$$

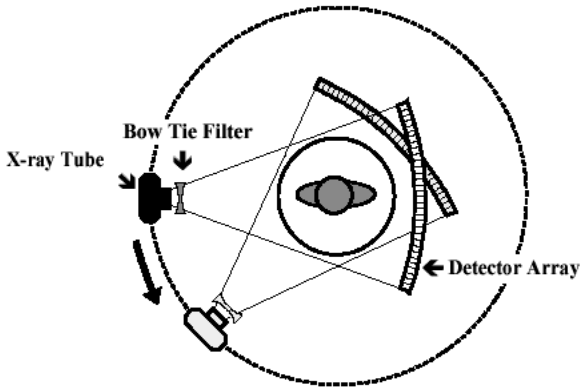
En développant et en éliminant les composantes indépendantes de μ (qui ne rentrent pas en compte dans la procédure d'optimisation) :

$$J_{adeq}(\boldsymbol{\mu}) \propto \sum_i (e^{-\mathbf{P}_{(i,\cdot)}\boldsymbol{\mu}} + e^{-p_i}\mathbf{P}_{(i,\cdot)}\boldsymbol{\mu})$$

La justification de ce calcul, le calcul du gradient de cette fonction, et les approximations dont nous parlerons juste ci-dessous ont été faites dans les articles d'Yves Goussard comme [**?**].

Différents types de critère Après avoir obtenu ce critère, on peut choisir de le garder tel quel, ce sera le critère Poissonnien, ou de le simplifier. On peut montrer qu'en faisant un développement limité à l'ordre 2 avec $\mathbf{N}_i \approx \boldsymbol{\lambda}_i$, on obtient un critère semblable à une erreur des moindres carrés ($\|\boldsymbol{y} - \mathbf{P}\boldsymbol{\mu}\|_{\boldsymbol{\Sigma}}{}^2$, avec $\boldsymbol{y} = \log(\frac{\mathbf{N}_0}{\mathbf{N}})$, où $\mathbf{N}_0$ est le nombre total de photons, et $\boldsymbol{\Sigma}$ est une matrice de pondération diagonale de la norme, dont les i-èmes coefficients diagonaux valent e^{-y_i}), ceci est ce que l'on aurait obtenu pour un modèle Gaussien de formation des données, on nommera donc cette approximation : critère Gaussien à pondération diagonale (qui peut se simplifier en critère Gaussien simple pour des très petits comptes de photons (cf. Goussard 13) ([**?**]).

FIGURE 1.1: Shéma d'un tomographe



1.2.2 Avantages de la matrice de projection polaire

Pour effectuer les opérations de projection lors du calcul de la valeur du critère et de son gradient, on utilise une matrice de projection $\mathbf{P}$ qui, multipliée par l'objet $\boldsymbol{\mu}$, renvoie un vecteur de données de sinogramme $\mathbf{N}$, dont chaque coefficient est le nombre de photons atténués par les pixels de l'objet situés sur la droite source-détecteur. Cette matrice est dépendante de la géométrie de représentation de l'objet, c'est à dire de comment sont ordonnés les coefficients dans les vecteurs $\boldsymbol{\mu}$ et $\mathbf{N}$. Yves Goussard s'est rendu compte que si l'on représentait l'objet en coordonnées polaires, cela créait des invariances dans la matrice de projection $\mathbf{P}$: cette matrice est alors bloc-circulante (ceci est dû

au fait que pour projeter l'objet tourné d'un certain angle, il suffit de permuter des blocs, si l'on adopte une représentation polaire) . Cela a pour conséquences de réduire considérablement le temps de calcul et l'espace mémoire utilisés à chaque calcul de $\mathbf{P}\boldsymbol{\mu}$, soit à chaque itération, car comme on le verra par la suite chaque itération de l'optimisation du critère doit évaluer le critère et son gradient, et donc faire des opérations de projection $\mathbf{P}\boldsymbol{\mu}$.

Chapitre 2

Objectifs

Lors de mon stage de six mois, j'ai suivi deux objectifs.

2.1 Etude de la reconstruction itérative

Tout d'abord, j'ai étudié les étapes de la reconstruction tomographique : quelles sont les différentes décisions à prendre et paramètres à choisir ? Comment utiliser les outils de simulation numérique à ma disposition ? Mieux comprendre ces étapes m'a permis de mettre au point un environnement de travail facile à utiliser pour lancer des simulations en tomographie, et pour détailler les effets des réglages. La première partie de ce rapport expose donc la mise en place d'un environnement de travail sur Matlab où l'on reliera les paramètres de manière à garder le problème invariant quelque soit l'échelle (on reviendra sur cette invariance d'échelle dans le paragraphe **??**). Puis j'étudie aussi les différents critères de reconstruction pour faire ressortir les compromis essentiels qui ressortent lors des choix du critères.

2.2 Amélioration : prise en compte du polychromatisme

Après avoir atteint le premier objectif, il m'était désormais beaucoup plus facile de tenter d'améliorer le modèle de reconstruction déjà existant. Le but de mon stage était d'implémenter un critère polychromatique dans le modèle déjà développé, afin d'améliorer la précision de la reconstruction notamment en présence de métaux, comme nous l'expliquerons par la suite. La deuxième partie du rapport explique donc à la fois les développements mathématiques à mettre en oeuvre dans le critère pour prendre en compte le polychromatisme, mais aussi les implémentations dans Matlab de ce critère.

Chapitre 3

Outils de travail

3.1 Formation des données et reconstruction (projection, tomographie) avec l'environnement Matlab : description

Nous décrivons ici comment former un fantôme, choisir et adapter un critère aux données, et effectuer une reconstruction, ainsi que ce qui motive notre choix de paramètres.

3.2 Introduction

Dans l'optique du premier objectif (comprendre l'influence des paramètres entrant en jeu dans une reconstruction tomographique), nous présentons ici les différentes étapes d'une simulation, à savoir la création du fantôme (censée simuler un objet réel, mais nous verrons qu'il convient de choisir des paramètres tels que la résolution ou le nombre et la nature des lésions), la projection tomographique (où de nombreux paramètres sont à choisir), et les paramètres de la méthode de reconstruction.

3.3 Fantôme

Le Fantôme XCAT représente l'objet à reconstruire. C'est une structure, qui contient les informations suivantes, que l'on se doit de choisir :

Position des tranches à simuler : On se place la plupart du temps à z = 900mm (une tranche qui contient de l'os et des aspérités pour mieux voir la résolution de la reconstruction).

Epaisseur des tranches à simuler : On prendra la plupart du temps des tranches de 1mm lors de ce stage.

Nombre de tranches à simuler : Pour des simulations plus rapides on pourra prendre une seule tranche, mais avec plusieurs tranches (5 ou 6, pour ne pas avoir des simulations trop longues) on pourra vérifier que notre travail s'applique bien à la 3D.

Résolution : La résolution est le nombre de pixels d'une tranche. On pourra la faire varier entre 160 x 160 et 320 x 320 selon que l'on préfère rapidité de la simulation ou précision des images.

Lésions : On peut créer des lésions sur le fantôme, on verra par la suite que cela est utile notamment pour tester le comportement de la reconstruction face à des lésions métalliques. Il faut alors préciser la position, le rayon (les lésions sont sphériques), et la composition de la lésion (pourcentage de tel et tel matériau). Par exemple une boule d'Aluminium à 100 % en z= 900, de rayon 1cm. Si l'on veut observer uniquement des lésions, il faudra centrer les tranches dans une zone "vide" (par exemple en z=3000mm).

Autres Paramètres : Il existe aussi d'autres paramètres que l'on choisira arbitrairement (les mêmes pendant tout le stage) car ils ne sont pas très importants : le nom du patient : 'TestXCAT', le sexe : 'Male', la présence de bras :'0'.

3.4 Projections

On peut désormais effectuer une tomographie de ce fantôme. Il convient donc de choisir les paramètres de la tomographie comme pourrait le faire un médecin ayant à choisir un détecteur, une source etc (même si en pratique la plupart des paramètres sont configurés). Mais il faut aussi choisir les paramètres dûs au fait que l'on fait une simulation numérique. On donnera plus de détails sur ce dernier point dans la partie correspondante **??**.

3.4.1 Paramètres propres à la tomographie

On distinguera les paramètres fixes, structurels du tomographe, et les paramètres variables, qu'un médecin est susceptible d'ajuster lors de chaque scan.

Paramètres structurels du tomographe

Distance centre-source : On garde celle donnée par le simulateur de tomographe Siemens utilisé lors du stage.

Distance centre-détecteurs : idem

Nombre de détecteurs : — dans la direction z : = autant que le nombre de tranches
— dans la direction tangentielle : autant que le nombre de pixels du côté d'une tranche

Nombre de projections : 2 fois le nombre de détecteurs. Ces deux dernières relations s'assurent que : :`2*nbPixels*nbPixels = nDetectors*nProjections`, pour avoir deux fois plus de mesures que d'inconnues.

Ouverture angulaire de la source et taille des détecteurs (dans la direction z et dans les autres) : déduit de `nDetectors, AngleTotal, RadiusFocus, RadiusDetector`. On calcule l'ouverture angulaire de la source à partir de la taille d'une tranche (pour que le faisceau frappe toute la tranche), et la taille des détecteurs à partir de cette ouverture et de la distance source-détecteurs à l'aide d'une relation de Thalès, de manière à avoir une géométrie de tomographie mise à l'échelle.

De plus, on fait un calcul qui va faire en sorte d'ajouter des tranches de manière à ce que le faisceau de rayons croise totallement toutes les tranches lors de la projection, et l'on soustraira les tranches ajoutées dans la reconstruction. Voir figure **??** pour plus de détails.

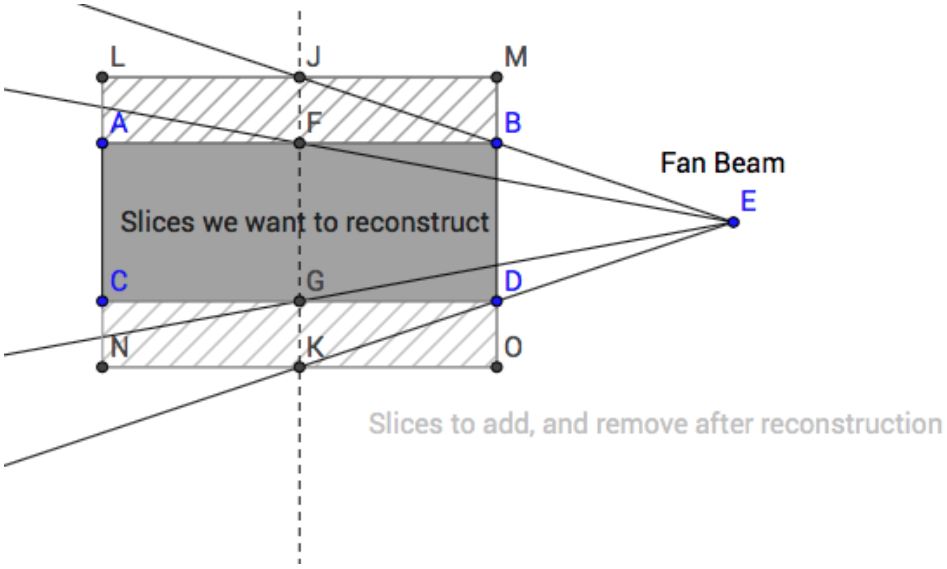


FIGURE 3.1: Rajouter des tranches pour entièrement prendre en compte celles voulues

Paramètres variables

mAs : Il s'agit de l'intensité de courant injectée dans la source que l'on souhaite avoir dans chaque détecteur : elle est proportionnelle au compte de photons par seconde et par cm^2 arrivant sur le détecteur.

3.4.2 Paramètres propres à la simulation numérique

Nombre de rayons par détecteurs : Le programme XCAT ne considère pas qu'un faisceau continu entre dans un détecteur (par défaut, il génère un seul rayon par détecteur). Pour se rapprocher de cela, on peut spécifier le nombre de rayons par détecteur (plus on en met, plus cela ressemblera à un faisceau continu. J'ai pris 20 car même si cela est long à générer, on a à générer le fantôme qu'une seule fois et cela n'influe pas sur le temps des reconstructions, donc autant prendre une assez grande valeur). Les rayons se partageront alors l'intensité par détecteur définie précédemment.

Fichier de source On doit choisir une source de rayons X. Choisir une source revient à choisir son spectre, c'est à dire la répartition de l'intensité lumineuse selon les différentes énergies. Elle se présente sous la forme d'un vecteur à deux colonnes (abscisses et ordonnées) et autant de lignes que de points du spectre : les énergies en abscisse et en keV, et les intensités correspondantes en ordonnées (en comptes de photon par cm^2 et par s).

3.5 Reconstruction

3.5.1 Géométrie

La géométrie de l'objet à reconstruire, c'est à dire la manière dont est échantillonné cet objet, peut être soit cylindrique soit cartésienne. Si elle est cylindrique on prendra comme dimensions les mêmes que lors de la génération du fantôme. Si elle est cylindrique on prendra un échantillonnage en z identique, un échantillonnage angulaire égal au nombre de projections (`nThetas=nProjections`), et un échantillonnage radial égal à `nRhos`$= \lfloor \frac{\texttt{nPixels}}{\texttt{nRhos}} \rfloor$, de manière à conserver le nombre d'inconnues entre les coordonnées cartésiennes et les coordonnées cylindriques.

3.5.2 Critère

Comme nous l'avons vu précédemment (cf **??**), nous devons choisir, pour reconstruire l'image, un critère à optimiser. Il est constitué d'un critère d'adéquation et d'un critère de régularisation. Dans la suite du rapport, nous verrons justement l'influence des différents choix pour ces critères. Globalement ce qu'on peut retenir en première approximation est que le critère Gaussien est plus facile à optimiser mais rend moins compte de la réalité physique du bruit que le Poissonnien. On retiendra aussi que le critère de pénalisation L2L1 conserve mieux les contours que le critère de pénalisation L2.

3.5.3 Matrice de projection

Lors de l'implémentation de la matrice de projection, il convient de choisir le nombre de rayons que l'on projette sur un détecteur, ainsi que la géométrie de cette matrice. La géométrie de la matrice doit être en adéquation avec le mode de représentation des données : polaires, ou cartésiennes. Ces matrices de projections sont en effet des matrices qui vont dicter, pour chaque pixel de l'observation $\mathbf{N}$, quelle est la contribution géométrique (sans tenir compte des exponentielles et des coefficients en tous genres) de chaque pixel de l'objet $\boldsymbol{\mu}$.

3.5.4 Méthode d'optimisation

De nombreux travaux on été effectués pour savoir quelle est la meilleure méthode d'optimisation pour ce type de problème. On reprendra celui de Goussard 13 [**?**], à savoir l'algorithme l-bfgsb. On devra choisir les différents paramètres pour l'optimisation. Les plus importants sont les suivants :

Tolérance sur la norme du gradient Lors de la procédure de minimisation, le gradient diminue globalement pour tendre dans l'idéal vers 0. On peut donc fixer une condition finale sur la norme du gradient : lorsque la norme du gradient projeté (normalisée en divisant par la norme du gradient initial) est inférieure à `ftol` on arrête l'algorithme.

Nombre max d'itérations On peut choisir un nombre maximal d'itérations. Cela est utile lorsque l'on veut effectuer une simulation rapide et que l'on ignore combien d'itérations elle prendra. En revanche il devient beaucoup plus difficile de comparer les résultats car la norme du gradient projeté peut être très différente entre les reconstructions à comparer.

Contrainte de positivité L'algorithme l-bfgsb offre la possibilité d'imposer une contrainte de positivité : c'est à dire que l'optimisation se fera en sachant que l'image est positive. C'est une

optimisation sous contrainte. En revanche la contrainte de positivité ne s'applique plus normalement à l'estimée courante si l'on utilise un préconditionneur, car ce dernier est un changement de variable (soit $\mathrm{P}(\boldsymbol{\mu})$ ce changement de variable, $P(\boldsymbol{\mu}) > 0$ n'impose pas forcément $\boldsymbol{\mu} > 0$).

Chapitre 4

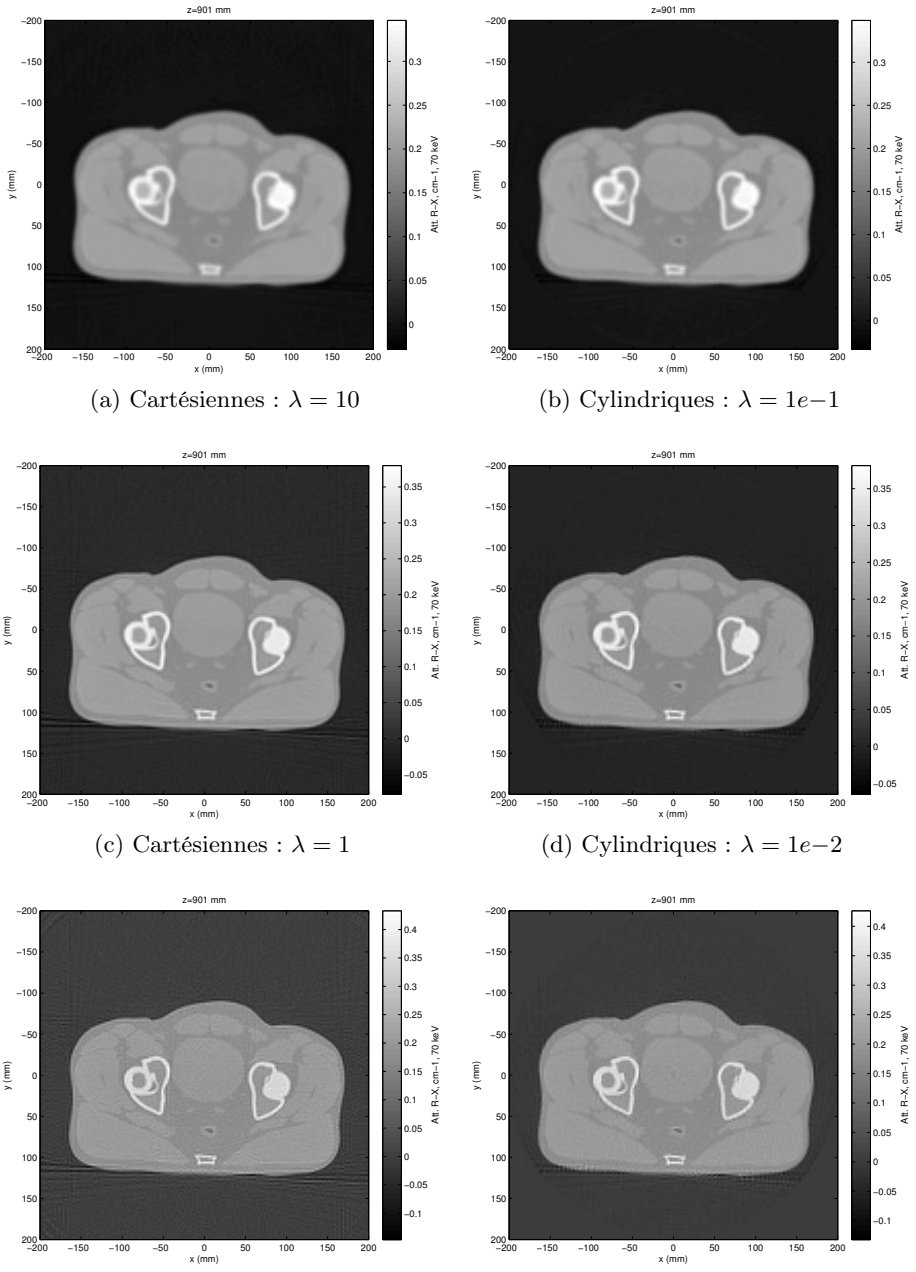
Discussions préliminaires à propos du critère

4.1 Mise en évidence du rôle des paramètres du critère : exemple du critère Gaussien

4.1.1 Influence de lambda

Objectifs : Montrer que plus lambda est grand, plus les bruits et artefacts sont atténués, et montrer que le lambda ne s'applique pas de la même manière en polaires et cartésiennes.

Génération des résultats : On génère 3*2 images pour une échelle de λ variant de $1e-1$ à 10 (pour les coordonnées cartésiennes) (zone de variation sensible des résultats), et pour trouver le lambda identique entre polaires et cartésiennes on regarde à l'oeil nu les images 2 à 2 et on essaye de faire qu'elles soient le plus identiques possibles. On utilise ici le critère d'adéquation Gaussien en guise d'exemple car c'est le plus simple, et la pénalisation GradL2 uniquement car on étudiera GradL2L1 dans la partie suivante.



(a) Cartésiennes : $\lambda = 10$

(b) Cylindriques : $\lambda = 1e{-}1$

(c) Cartésiennes : $\lambda = 1$

(d) Cylindriques : $\lambda = 1e{-}2$

(e) Cartésiennes : $\lambda = 1e{-}1$

(f) Cylindriques : $\lambda = 1e{-}3$

FIGURE 4.1: Associations d'images pour lesquelles λ a le même effet, pour `PenalGrad_L2`.

Analyse On voit que plus λ augmente, plus les hautes fréquences de l'image sont détruites. L'on voit aussi qu'une valeur de 1 (resp.$1e-2$) en coordonnées cartésiennes (resp. polaires) est une bonne valeur pour obtenir une image ni floue ni bruitée. Nous résumons les résultats dans le tableau suivant :

TABLE 4.1: Résumé des résultats

λ	Type de reconstruction	Commentaires	Nombre d'itérations	Temps moyen par itération (s)
10	Cartésiennes	Un peu floue	78	1.15
1	Cartésiennes	Bonne image	50	"
$1e{-}1$	Cartésiennes	Quelques artefacts	190	"
$1e{-}1$	Polaires	Un peu floue	42	0.7
$1e{-}2$	Polaires	Bonne image	78	"
$1e{-}3$	Polaires	Quelques artefacts	95	"

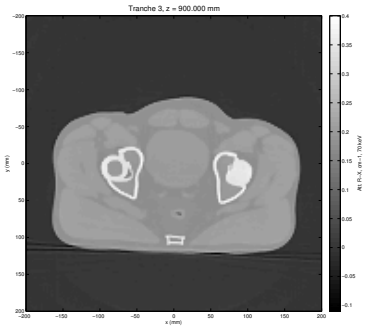
On voit qu'il y a moins d'itérations (et un temps par itération plus faible) avec la reconstruction qui utilise des coordonnées polaires et un préconditionnement, pour une même qualité d'images. Ceci suggère que les coordonnées polaires offrent une alternative intéressante à explorer en reconstruction itérative.

4.1.2 Influence du type de pénalisation

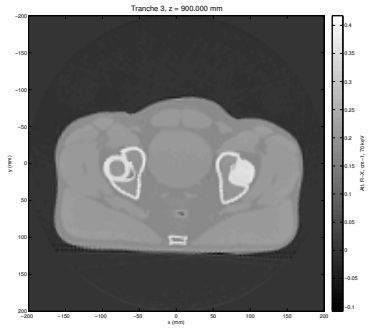
Objectifs : Une autre manière de régulariser l'image est d'utiliser le critère de régularisation L2L1 : `PenalGradObj_L2L1`. Le critère utilisé jusqu'à présent était le critère L2 : `PenalGradObj_L2`. La différence entre les deux est que le précédent essaye de minimiser la somme quadratique des différences premières de l'image, alors que `PenalGradObj_L2L1`, au lieu de mettre au carré chaque différence première, la transforme par une fonction linéaire quadratique (de la forme $f(x) = \sqrt{x^2 + \delta^2} - \delta$, où le delta est à choisir (pendant toute la durée du stage on prendra $\delta = 5e - 3$). Ainsi, on devrait avoir une image qui garde mieux les contours si l'on applique cette régularisation (on "estompe" uniquement les zones avec peu de variations d'intensité, plus potentiellement contituées du même matériau, et on "estompe" moins les grosses variations comme au niveau des contours).

Génération des résultats : On génère les données pour une valeur $1e - 1$ de lambda pour le cartésien, car c'est la valeur pour une bonne régularisation trouvée précédemment, et elle reste bonne ici. Idem que précédemment on essaie de trouver le lambda pour avoir le même effet en polaire. On prend $5e - 3$ pour le δ car c'est la valeur recommandée. On va donc montrer cette affirmation par des simulations.

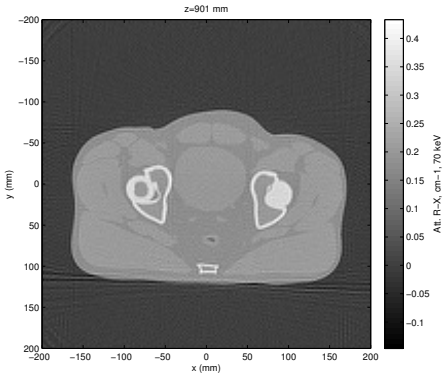
Analyse On a donc réussi à obtenir le même type de résultats que dans le paragraphe précédent, à savoir trouver les λ donnant des images équivalentes en coordonnées cartésiennes et en coordonnées polaires, mais cette fois -ci avec une régularisation L2L1



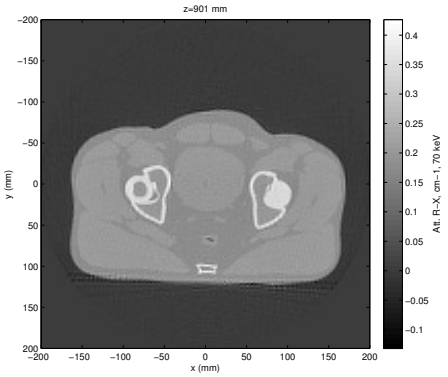
(a) Cartesian : $\lambda = 1e{-}1$, `PenalGrad_L2L1`



(b) Cylindriques : $\lambda = 1e{-}3$, `PenalGrad_L2L1`



(c) Cartesian : $\lambda = 1e{-}1$, `PenalGrad_L2`



(d) Cylindriques : $\lambda = 1e{-}3$, `PenalGrad_L2`

FIGURE 4.2: Associations d'images pour lesquelles λ a la même effet

4.2 Le critère Poissonnien : analyse approfondie

Le critère Poissonnien, expliqué brièvement précédemment (cf), est le critère qui rend vraiment compte de la réalité physique de la formation des données. Il est donc plus susceptible de rendre des images précises et plus robustes face au bruit du détecteur. Cependant, il est plus difficile à minimiser que sa version simplifiée en critère Gaussien ou Gaussien pondéré. Avec l'augmentation de la puissance de calcul des ordinateurs, et grâce à la méthode de reconstruction en coordonnées polaires de Yves Goussard, qui accélèrent les calculs et permettent donc une plus grande complexité d'opérations, il nous a semblé pertinent de nous intéresser plus précisément à ce critère.

Introduction Le modèle Poissonnien suppose que le nombre de photons arrivant en chaque pixel d'une projection est le résultat d'une expérience de Poisson de paramètre λ rendant compte de l'atténuation du rayon source-pixel considéré. Le modèle Poissonnien initial est monochromatique : on considère pour simplifier que les rayons X frappant l'objet ne sont constitués que d'une fréquence. La fonction à minimiser, aussi appelée critère, est la neg log-vraisemblance d'avoir les projections observées, sachant qu'on a l'objet courant estimé. Elle prend alors la forme suivante :

$$f(\boldsymbol{\mu}) = -\ln(P(\mathbf{N}|\boldsymbol{\mu}))$$

avec f : la fonction à minimiser (critère), $\boldsymbol{\mu}$ l'image estimée, $\mathbf{N}$ le vecteur des comptes de photons observés, qui suit pour chaque pixel une loi de paramètre λ qui se calcule grâce à la loi de Beer lambert, comme suit :

$$\lambda = N_0 e^{-\int_\Delta \mu(s)ds}$$

avec : μ : l'atténuation au point s de la droite Δ source-pixel considéré

Objectifs : J'ai voulu effectuer des tests avec cette méthode de reconstruction Poissonnienne, et j'ai rencontré plusieurs problèmes. Nous verrons lesquels, et comment améliorer le modèle de Poisson en tenant compte de la polychromaticité des rayons. Les problèmes rencontrés sont les suivants :

- Oscillations au niveau des contours de l'objet en Cartésiennes
- Mauvaise convergence de l'algorithme
- Préconditionnement

De plus, on finira le document en justifiant la nécessité d'adopter un critère Poissonnien tenant compte de la polychromaticité pour la reconstruction.

Génération des données et paramètres invariants Ci-dessous nous décrivons les paramètres des fantômes et de la reconstruction qui ne varient pas, et que l'on a gardés pour tous les tests :

- Résolution des fantômes : 160x160 pixels
- Mode de formation des projections : compte de photons
- Tranche à reconstruire : unique, en z=900mm, de taille 40x40 cm, d'épaisseur 1mm
- Les matériaux sont plongés dans un fantôme d'eau : une boule de 40cm de diamètre, centrée sur la tranche à reconstruire
- Le critère est, sauf à la fin pour l'étude du beam hardening, toujours monochromatique Poissonnien
- Le spectre de rayons X polychromatique est échantillonné en 20 points (+2 pour les pics) (ceci n'est utile qu'à la fin du rapport)
- Régularisation : L2, dont le coefficient est précisé avant chaque expérience

4.2.1 Oscillations en Cartésiennes

On a vu lors des reconstructions que les images en coordonnées cartésiennes présentaient des oscillations sur le contour du fantôme d'eau. Ceci n'est vrai que pour le critère Poissonnien, et peut être expliqué par le fait que la reconstruction des données suppose un modèle de rayons comme un faisceau épais et continu (comme un vrai tomographe), alors qu'avec le logiciel XCAT de création de fantômes et de sinogrammes on a projeté des fantômes avec un seul rayon par détecteur. Pour mettre en évidence cela, on a simulé les données avec plusieurs rayons par détecteurs (en prenant soin de ne pas prendre le même que dans la matrice de projection pour ne pas faire de crime inverse (on en prend 5 alors que les matrices en prennent 3)), et l'on a exposé les résultats ci-dessous :

Paramètres invariants de la simulation

- Images générées : fantômes avec deux boules de fer de diamètre 2 cm et deux boules d'aluminium de rayon 6 cm (voir figure **??**).
- Tranche reconstruite : coupe la boule en son centre
- Coordonnées : cartésiennes
- Régularisation : 0.001 (pour avoir un contour non flou mais ne pas avoir trop d'artefacts en forme de raies)
- Colormap pour le résultat : [0, 0.5]

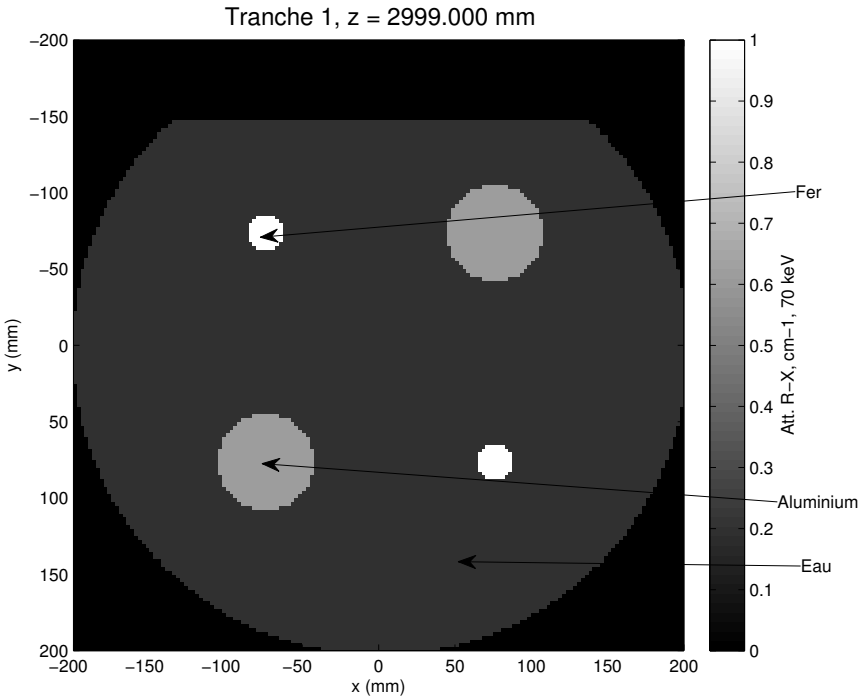
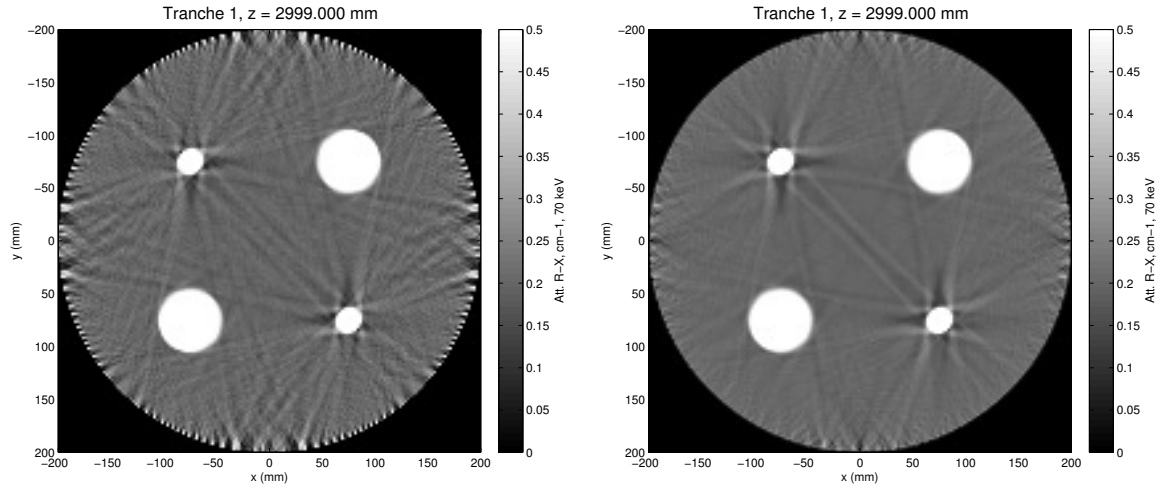


FIGURE 4.3: Schéma du fantôme généré

Résultats : comparaison entre un seul et 5 rayons par détecteur



(a) 1 rayon par détecteur

(b) 5 rayons par détecteur

FIGURE 4.4: Influence de la formation des données sur la reconstruction finale

TABLE 4.2: Indicateurs de convergence

Ini	Iter	f(x)fin	Rel.desc.	|Pg(x)|	Coût	Raison de la convergence
0	644	1.28e4	7.11e-16	1.39e-6	37	abn
0	793	1.39E28e4	0	2.7e-6	12	nconv

Indicateurs de convergence : dernière ligne affichée par lbfgsb lors de la reconstruction :

- Ini : Intensité de l'image d'initialisation (uniforme)
- Iter : Nombre d'itérations effectuées
- f(x)fin : Valeur finale du critère
- Rel. desc. : Différence entre les deux dernières valeurs du critère
- |Pg(x)| : Valeur absolue du gradient
- Coût : coût en termes de calcul pour la dernière itération (nombre de pas lors de la recherche, pour la dernière itération)
- réponses possibles pour la convergence de l'algorithme
  - tol : Norme du gradient projeté inférieure à la tolérance
  - abn : "abnormal termination in line search"
  - nconv : "Null relative descent stops the solve, have we converged, though ?"

Solution :

Il faut donc augmenter le nombre de rayons par détecteurs lors de la génération du sinogramme. On voit que les réultats sont meilleurs (meilleure image, peut-être même meilleure convergence, ce qui est encourageant quand on sait que plus il y a de rayons plus on se rapproche d'une image susceptible d'être reconstruite avec un vrai tomographe.)

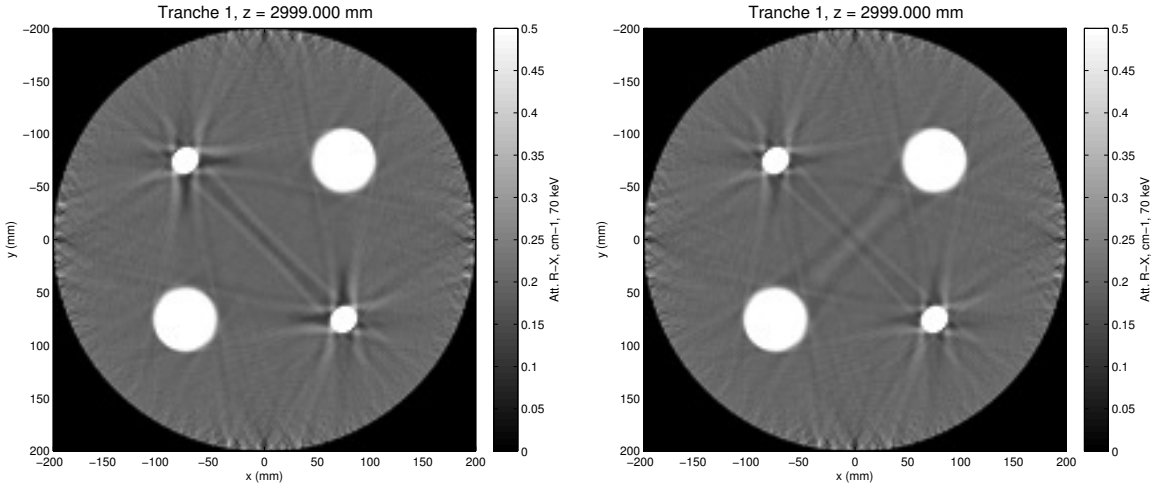
4.2.2 Conditionnement du critère Poissonnien

Affirmation Au cours des tests de reconstruction avec le critère Poissonnien, il nous a semblé que ce dernier était mal conditionné, ce qui cause une convergence lente sur la fin ou inexacte, faisant écho à la thèse de Benoît Hamelin paragraphe 2.3 : « De plus, l'algorithme de De Man et al. (2001) repose sur la modélisation de l'incertitude sur les mesures par la loi de Poisson. Le problème de reconstruction qui en résulte est mal conditionné [...]. » [**?**] Nous allons détailler les test effectués qui nous poussent à affirmer cela. Pour cela, nous allons nous intéresser aux valeurs finales des indicateurs de la convergence de l'algorithme l-bgsfb. En effet l'on peut contrôler l'arrêt de l'algorithme de minimisation en fixant par exemple une tolérance à atteindre sur la norme du gradient projeté du critère : plus cette norme est petite, plus l'on force l'algorithme à converger loin.

Influence des paramètres du critère Dans ces test nous changeons la tolérance demandée sur le gradient du critère. Nous verrons que pour une tolérance de 10^{-5}, les images sont assez différentes, alors qu'il y a bien une image finale, que l'on peut obtenir si l'on force plus la convergence en diminuant la tolérance jusqu'à 10^{-8}. Les paramètres de la simulation sont les mêmes qu'en **??**, et avec un nombre de rayons par détecteurs de 8.

Résultats

Tolérance 1e-5 Pour cette tolérance, on observe surtout que les images sont assez différentes, et que l'algorithme a atteint le seuil de tolérance sans encombres.



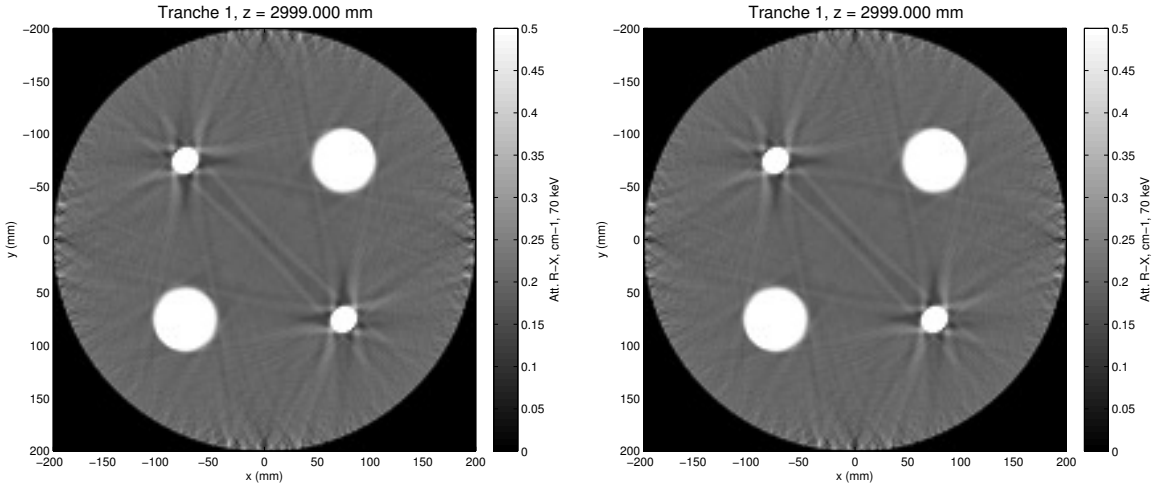
(a) Initialisation de $\mu = 0$ (b) Initialisation de $\mu = 20$

FIGURE 4.5: Tolérance 1e-5 : 2 images différentes

TABLE 4.3: Indicateurs de convergence

Ini	Iter	f(x)fin	Rel.desc.	|Pg(x)|	Coût	Raison de la convergence
0	341	1.28e4	7.44e-10	6.13e-4	1	tol
20	983	1.28e4	6.41e-10	4.82e-4	1	tol

Tolérance 1e-8 Sur ces images, on observe que les images sont assez identiques, comme si on avait pu "forcer" la convergence en mettant une tolérance plus faible. Cependant, l'algorithme semble moins performant lorsqu'il atteint ces limites de précisions car on n'a pas réussi à atteindre la tolérance de 10-8 (trop d'itérations effectuées, cf "Coût").



(a) Initialisation de $\mu = 0$ (b) Initialisation de $\mu = 20$

FIGURE 4.6: Tolérance 1e-8 : images identiques

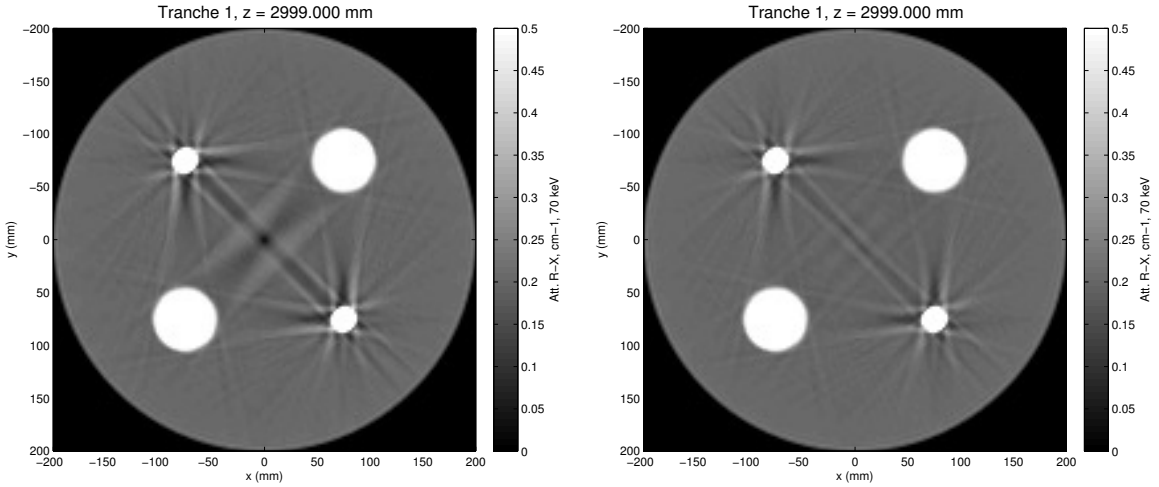
TABLE 4.4: Indicateurs de convergence

Ini	Iter	f(x)fin	Rel.desc.	|Pg(x)|	Coût	Raison de la convergence
0	816	1.28e4	3.08e-14	4.14e-6	1	abn
20	1506	1.28e4	0	3.56e-6	37	abn

4.2.3 Autres problèmes

Absence de préconditionneur en polaires

Sans préconditionneur, en polaire, on observe un petit disque sombre au centre de l'image, dû à un mauvais conditionnement, comme c'est le cas avec un critère Gaussien cf. [**?**]. On voit sur les images ci-dessous que le préconditionneur (ici diagonal de Fourier) permet de supprimer ce problème. Génération des données : on garde les 5 rayons par détecteurs Tolérance : 10-5



(a) sans préconditionneur (b) avec préconditionneur

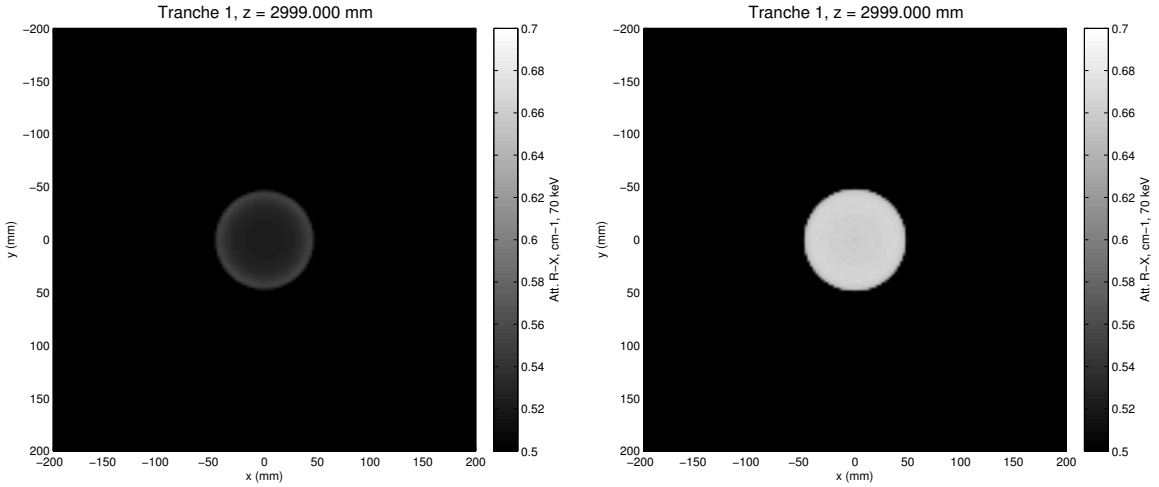
FIGURE 4.7: Influence du préconditionneur

4.2.4 Nécessité d'implémenter un critère polychromatique

Dans cette partie, on met en évidence l'effet de durcissement de rayon : on crée une boule d'aluminium et on verra que l'atténuation centrale est sous-estimée en monochromatique, mais que le critère polychromatique lui redonne sa valeur normale.

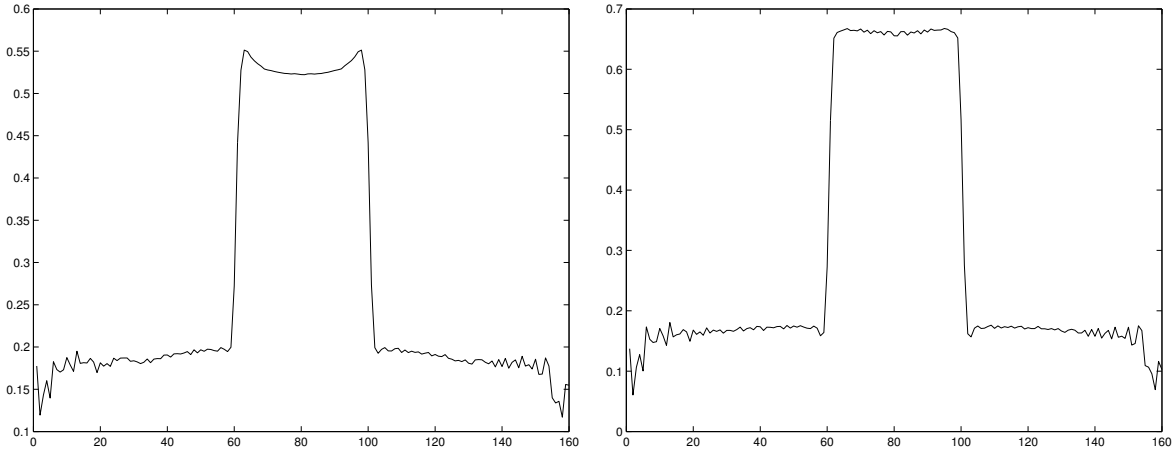
Fantôme

- Boule d'aluminium de 10 cm de diamètre, dans une boule de 40 cm de diamètre
- Tranche reconstruite : coupe la boule en son centre
- Nombre de rayons par détecteurs : 8
- Coordonnées cartésiennes
- On a "forcé la convergence" en initialisant avec une image d'intensité uniforme 0.6.
- On a recalé la colormap entre 0.5 et 0.7 pour voir l'évolution du profil d'intensité radial de manière optimale.



(a) Monochromatique (b) Polychromatique

FIGURE 4.8: Durcissement de rayon : images



(a) Monochromatique (b) Polychromatique

FIGURE 4.9: Durcissement de rayon : profils d'intensité

Analyse On remarque que l'adoption d'un critère polychromatique permet bien d'atténuer voire de supprimer le durcissement de rayon : le profil en polychromatique est bien plus plat au niveau de la boule d'aluminium (comme il doit l'être) qu'en monochromatique.

Conclusion Nous avons tout d'abord remarqué que certains artefacts étaient dûs à la projection simulée avec un seul rayon fin par détecteur. Nous avons vu qu'en corrigeant cela en moyennant plusieurs rayons sur un même détecteur pour simuler un rayon épais le problème était résolu. A partir de là nous avons pu étudier la convergence de l'algorithme de minimisation du critère Poissonnien en tant que tel. Nous avons vu que le critère Poissonnien était probablement mal conditionné

car on devait forcer l'algorithme à converger vers une image unique en mettant une tolérance très précise. Pour résoudre ce problème, qui est amplifié en coordonnées polaires, nous avons vu que nous pouvions implémenter un préconditionneur qui facilite la convergence de l'algorithme (mais on perd la contrainte de positivité). Enfin nous avons vu que la prise en compte de la polychromaticité des rayons améliorait notablement la reconstruction, notamment en corrigeant l'effet de beam hardening.

Chapitre 5

Mise en place du critère polychromatique

5.1 Expression mathématique du critère polychromatique

5.1.1 Loi de Beer-Lambert polychromatique

Loi intégrale pour un spectre continu On rappelle que l'atténuation des photons arrivant sur l'objet est donnée par la loi de Beer-Lambert. Auparavant, cette loi considérait que le spectre de la source était constitué d'une seule fréquence. En réalité, la source possède un spectre continu de fréquences, distribuées souvent selon le même motif : une partie lisse, et une partie consituée de deux pics à des fréquences caractéristiques. Ci-dessous une figure donne un spectre classique d'une source de rayons X. Les intensités correspondant aux énergies considérées sont représentées (les énergies sont en keV et les intensités en quanta par cm^2 par s.

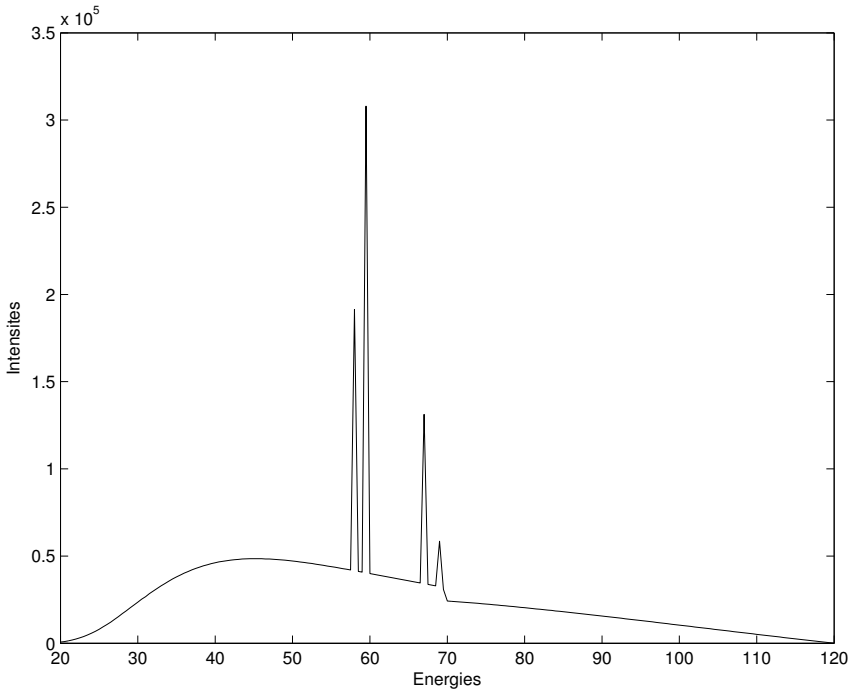


FIGURE 5.1: Source de rayons polychromatique

A chaque fréquence, le matériau répond de manière spécifique : le coefficient μ d'absorption dépend de la fréquence. L'expression de l'absorption devient donc :

$$\boldsymbol{\lambda} = \int_E b(E) e^{-\mathbf{P}\boldsymbol{\mu}(E)} dE$$

où $\boldsymbol{\lambda}$ est le vecteurs des paramètres de la loi de Poisson qui donne le compte de photons en l'ensemble dess pixel du détecteur, $b(E)$ donne le nombre de photons de la source d'énergie comprise entre E et E+dE, et $\mathbf{P}$ est la matrice de projection. Pour plus d'explications sur la matrice de projection, voir **??** et **??**.

Loi somme pour une approximation de spectre discret En pratique, le spectre sera échantillonné en k valeurs d'intensité $b * \alpha_k$ avec $\sum_k \alpha_k = 1$, correspondant aux énergies E_k. L'expression de l'absorption devient donc :

$$\boldsymbol{\lambda} = b \sum_k \alpha_k e^{-\mathbf{P}\boldsymbol{\mu}_k}$$

avec $\boldsymbol{\mu}_k = \boldsymbol{\mu}(E_k)$, et ici $b * \alpha_k$ est le nombre de photons de la source, d'énergie comprise entre E_k et E_{K+1}.

5.1.2 Approximation de l'atténuation en fonction de μ_{70}

Dépendance uniquement par rapport à phi et theta Si l'on gardait l'expression précédente pour la formation des données, on aurait un problème d'optimisation où il faudrait estimer k vecteurs $\boldsymbol{\mu}_k$ de taille $n = \texttt{nPixels}^2$ chacun, ce qui est beaucoup trop. Heureusement, DeMan a établi empiriquement une formule qui donne les $\boldsymbol{\mu}_k$ à chaque énergie d'indice k en fonction d'une atténuation de référence $\boldsymbol{\mu}_{70}$: l'atténuation $\boldsymbol{\mu}$ à 70keV. Ainsi le problème d'estimation a de nouveau autant d'inconnues que précédemment : on cherche uniquement à estimer $\boldsymbol{\mu}_{70}$: un seul vecteur de taille n. DeMan a obtenu ce résultat en décomposant l'atténuation $\boldsymbol{\mu}_k$ en deux parties (décomposition d'Alvarez-Makcovski) : une partie qui tient compte de l'effet Compton et l'autre de l'effet photoélectrique (deux effets physiques responsables de l'atténuation des rayons X).

$$\boldsymbol{\mu}_k = \underbrace{\Phi_k \phi}_{photoelectrique} + \underbrace{\Theta_k \theta}_{Compton}$$

où Φ_k et Θ_k sont des constantes dépendantes uniquement de l'énergie k, et ϕ et θ des fonctions dépendant uniquement du matériau atténuant, mais pas de l'énergie k. De Man s'est aperçu que les fonctions ϕ et θ pouvaient même s'exprimer en fonction seulement du $\mu(70keV)$ du matériau considéré.

$$\boldsymbol{\mu}_k = \underbrace{\Phi_k \phi(\boldsymbol{\mu}_{70})}_{photoelectrique} + \underbrace{\Theta_k \theta(\boldsymbol{\mu}_{70})}_{Compton}$$

Pour plus de détails, voir l'article de De Man [**?**]

Modèle pour phi et theta : interpolation avec les matériaux Pour obtenir l'expression des fonctions ϕ et θ, De Man a calculé leurs valeurs pour de nombreuses valeurs de μ_{70} caractéristiques de matériaux (métal, aluminium, eau, os ... à préciser). De Man a ensuite supposé que l'on pouvait interpoler cette fonction entre les points. Il existe d'ailleurs plusieurs méthodes d'interpolation : De Man utilise une méthode linéaire, mais nous verrons par la suite que l'on peut implémenter dans Matlab une fonction qui interpolera entre les points par une hyperbole. cf **??**

5.1.3 Expression mathématique du critère et de son gradient

Expression du critère

On rappelle que le critère à évaluer est le suivant :

$$f(\boldsymbol{\mu}) = -\ln(P(\boldsymbol{y}|\boldsymbol{\mu}))$$

avec f : la fonction à minimiser (critère), $\boldsymbol{\mu}$ l'image estimée, $\boldsymbol{y}$ le vecteur des comptes de photons observés, qui suit pour chaque pixel une loi de Poisson de paramètre λ qui se calcule grâce à la loi de Beer lambert en polychromatique, comme vu précédemment :

$$\lambda = b \sum_k \alpha_k e^{-\mathbf{P}\boldsymbol{\mu}_k}$$

On prendra à partir de maintenant $\boldsymbol{\mu}$ pour $\boldsymbol{\mu}_{70}$ L'expression du critère devient (cf. **??** pour tous les développements mathématiques) :

$$J(\mu) = sum(\tilde{\boldsymbol{y}}) + \frac{1}{b}\boldsymbol{y}^T \log \tilde{\boldsymbol{y}}$$

Expression de son gradient

$$\Delta J(\boldsymbol{\mu}) = -\phi'(\boldsymbol{\mu}) \cdot \mathbf{P}^T(\boldsymbol{e} \cdot \tilde{\boldsymbol{y}}^{\Phi}) - \theta'(\boldsymbol{\mu}) \cdot \mathbf{P}^T(\boldsymbol{e} \cdot \tilde{\boldsymbol{y}}^{\Theta})$$

Expression approximée du critère et de son gradient

Expression approximée du critère

$$J(\boldsymbol{\mu}) = \texttt{Sum}(\frac{\boldsymbol{y}}{2}. * (\mathbf{S})\hat{.}2)$$

avec

$$\mathbf{S} = \boldsymbol{s} - p\mathbf{P}\Phi(\boldsymbol{\mu}) - t\mathbf{P}\Theta(\boldsymbol{\mu})$$

avec

$$\boldsymbol{p} = \sum_k \alpha_k \Phi_k$$

$$\boldsymbol{t} = \sum_k \alpha_k \Theta_k$$

Expression de son gradient

$$\nabla_\mu J(\boldsymbol{\mu}) = (-\boldsymbol{p}\Phi'(\mu) - \boldsymbol{t}\Theta'(\mu)). * \mathbf{P}^T.(\mathbf{S}. * \boldsymbol{y})$$

5.2 Implémentation dans Matlab

5.2.1 Introduction

Pour l'implémentation dans Matlab, le principal enjeu consiste à modéliser la source pour le critère polychromatique : la source intervient en effet dans l'expression des α_k et de b. On expose ici le modèle de ré-échantillonnage de la source développé par Benoît Hamelin. On cherche à ré-echantilloner la source pour donner de la flexibilité au modèle : si les calculs sont trop longs on pourra choisir un pas d'échantillonnage plus large.

5.2.2 Ré-échantillonnage de la source

Format initial de la source

Le fichier source utilisé par XCAT se présente sous la forme d'un tableau à deux colonnes : une colonne des énergies et une colonne des intensités correspondantes à chaque énergie, avec un pas constant. Les intensités sont en compte de photons par cm^2 et par seconde ($\texttt{quanta}/(cm^2.s)$). On considèrera donc que ce format est le format de base d'une source avant ré-échantillonnage. Les intensités de ce vecteur ne prennent pas en compte l'échantillonnage : les coefficients $\alpha_k * b$ eux en revanche portent en eux le poids de l'échantillonnage. On verra ci-dessous dans l'opération de ré-échantillonnage comment obtenir ces coefficients $\alpha_k * b$ ("source" de sortie désirée) avec une discrétisation au pas voulu.

Décomposition de la source en deux parties

La source initiale est constituée de deux parties, une partie "lisse", croissante puis décroissante ("triangulaire"), et une partie discontinue, des pics à deux fréquences caractéristiques. L'idée de Benoît Hamelin pour le ré-échantillonnage du spectre est de ré-échantillonner séparément ces deux parties afin de conserver le plus d'informations sur la source (contenu haute fréquence et contenu basse fréquence).
Les pics de la source sont situés toujours aux mêmes fréquences caractéristiques. Benoît Hamelin a donc choisi une fenêtre fixe pour séparer les deux parties. On considère comme dans la partie "pics" les points de la source dont l'énergie est comprise entre 57 et 64 keV (première fréquence caractéristique), et 66 et 72 keV (deuxième fréquence caractéristique). Le reste des points est la partie "lisse".

Réechantilllonnage de la partie lisse, triangulaire

Pour ré-échantillonner cette partie, Benoît Hamelin a créé une fonction Matlab `ge_model_poly2line` qui interpole le vecteur source partie "lisse" précédent. On n'a ensuite plus qu'à préciser les nouvelles valeurs d'énergie dont on cherche les intensités, et former ainsi un nouveau vecteur source lisse ré-échantillonné.

Interpolation La fonction d'interpolation de la partie lisse `ge_model_poly2line` développée par Benoît Hamelin est constituée d'un polynôme pour approximer la première partie de la courbe, et d'une droite pour approximer la fin de la courbe. Le degré du polynôme ainsi que l'intensité de bascule entre la partie polynômiale et la partie linéaire sont automatiquement calculés de manière à

minimiser l'erreur quadratique avec les données (à savoir le vecteur contenant la partie triangulaire "lisse").

Ré-échantillonnage Une fois la partie triangulaire interpolée, on peut ré-échantillonner cette partie : il suffit de préciser le nombre d'échantillons que l'on souhaite, et les intensités correspondantes seront calculées grâce à la fonction d'interpolation `ge_model_poly2line`. Puisque la source ré-échantillonnée que l'on désire avoir en sortie est en réalité les $\alpha_k * b$, on doit aussi prendre en compte le pas d'échantillonnage. Pour le trouver on prend le pas de la source initiale, le nouveau pas, et on applique la règle de trois aux intensités.

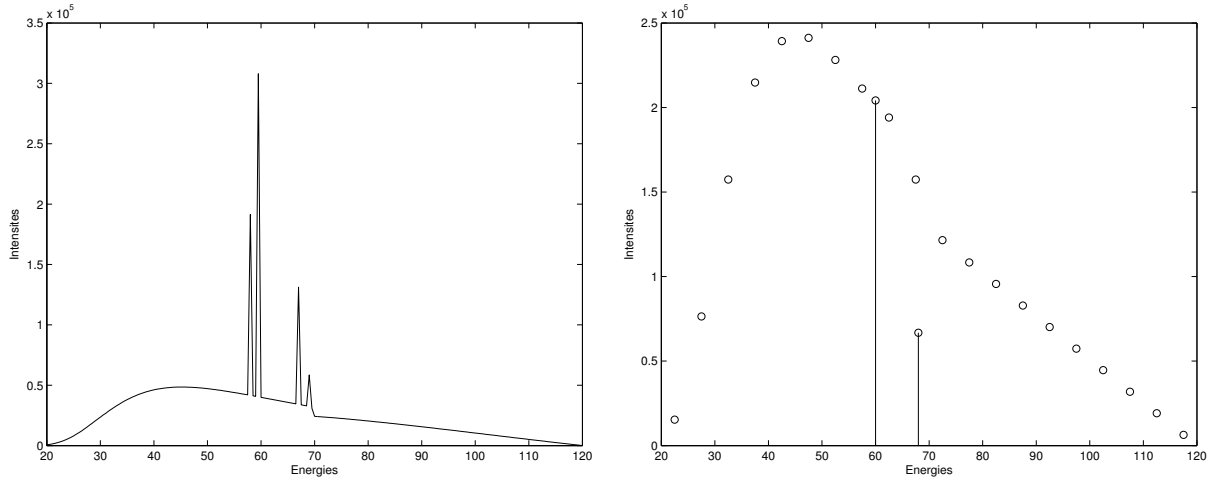
ré-échantillonnage des pics

Reste maintenant à ré-échantillonner les pics. On rappelle que l'on a deux ensembles de points "pics" correspondants au voisinage des deux fréquences caractéristiques. Benoît Hamelin transforme ces deux ensembles en deux valeurs : une pour chaque pic, qui est la somme de l'ensemble des points contenus dans le pic correspondant (et j'ai corrigé me semble-t-il une erreur en prenant soin de multiplier par la fréquence d'échantillonnage pour avoir des $\alpha_k * b$ comme désiré en sortie), de la manière suivante :

```
1 pas = sp(2,1)-sp(1,1); % rajoute
2 spike68 = sp(find(sp(:,1) > 66 & sp(:,1) < 72), :);
3 A60 = sum(ce60)*pas; % multiplication par pas rajoutee
4 ce68 =  spike68(:,2) - ge_model(spike68(:,1));
5 A68 = sum(ce68)*pas; % multiplication par pas rajoutee
```

Résumé

On a résumé ci-dessous les sources avant et après ré-échantillonnage pour un échantillonnage de 20 points pour la partie lisse, avec comme fichier source initial celui utilisé par XCAT.



(a) source initiale

(b) source ré-échantillonnée

FIGURE 5.2: source initiale et source ré-échantillonnée

On remarquera que les valeurs ne sont pas les mêmes. Cela est normal car à gauche il s'agit des valeurs brutes d'intensité de la source à chaque énergie, alors qu'à droite il s'agit de la source ré-échantillonnée avec un intervalle de 5 keV mais dont chaque valeur d'intensité porte en elle le poids de l'échantillonnage : chaque valeur est donc 5 fois plus grande que dans la figure de gauche. On remarquera de plus que les pics n'apparaissent plus vraiment sous forme de pics. Ceci est normal car n'oublions pas que ces points se rajoutent au spectre déjà existant : lors des calculs qui seront faits avec cette source, on ajoutera la valeur des pics (qui prend aussi en compte l'échantillonnage cible et l'échantillonnage source).

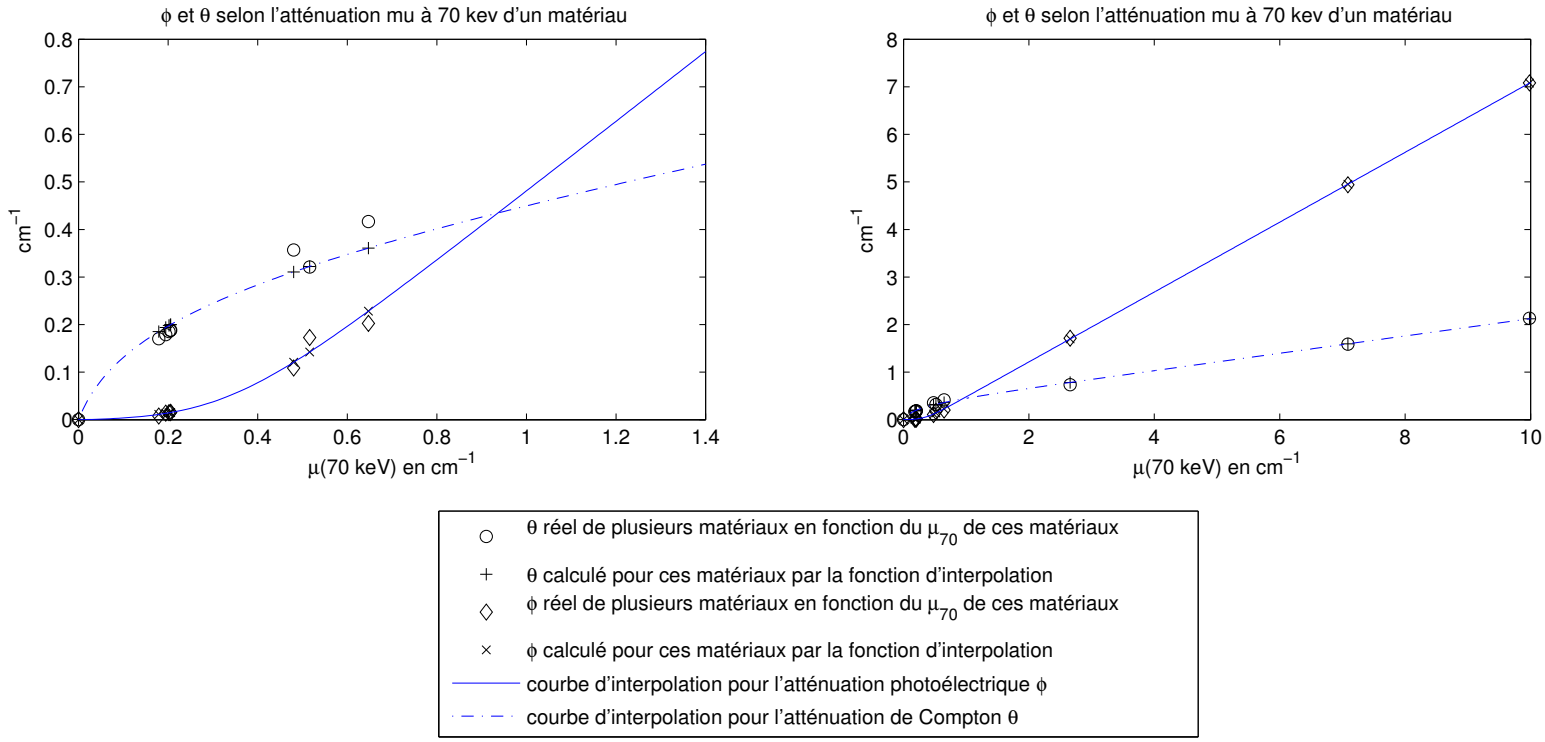
5.2.3 Modèle d'atténuation photoélectrique et Compton

On a vu précédement que l'on avait besoin de fonctions $\Phi(\boldsymbol{\mu})$ et $\Theta(\boldsymbol{\mu})$ pour calculer le critère polychromatique. On rappelle que ces fonctions rendent compte de la part d'atténuation des rayons X dues respectivement à l'effet photoélectrique et l'effet Compton. De Man a mis en évidence une méthode pour obtenir ces fonctions à partir de points déjà existants pour des matériaux connus. On peut en effet placer sur un graphe les points $\Phi(\boldsymbol{\mu}_{70kev})$ et $\Theta(\boldsymbol{\mu}_{70kev})$ pour un ensemble de matériaux à choisir (ex : eau, métal etc). Alors que DeMan interpole linéairement cet ensemble de points pour obtenir les fonctions $\Phi(\boldsymbol{\mu}_{70kev})$ et $\Theta(\boldsymbol{\mu}_{70kev})$, Benoît Hamelin utilise une interpolation hyperbolique entre ces points : il crée une hyperbole dépendant de 4 paramètres, et il trouve ces paramètres en minimisant une erreur quadratique.

Tracé des courbes

Ci-dessous on verra un aperçu des courbes tracées par cette fonction de Benoît Hamelin. Pour plus de détails, voir la thèse de Benoît Hamelin [**?**], et l'article de DeMan [**?**] :

FIGURE 5.3: Modèle d'atténuation de Compton et photoélectrique



Annexe A

Développements relatifs au critère (crédits : Yves Goussard)

A.1 Cadre monochromatique

En reprenant les notations de DeMan :
$\boldsymbol{y}_i$: compte de photons sur le détecteur i
$\boldsymbol{\mu}$: coefficients d'atténuation de l'objet à reconstruire
$\boldsymbol{y}_i$ suit une distribution de Poisson de paramètre $\hat{y}_i$
On a $\hat{y}_i = b_i e^{-\sum_j a_{ij}\mu_j}$ avec $\mathbf{P} = [a_{ij}]$ matrice de projection. En pratique, on suposera que $b_i = b$ indépendant du détecteur

$$\Pr(y_i) = \frac{\hat{y}_i^{y_i}}{y_i!} e^{\hat{y}_i}$$

Neg-log-vraisemblance de $\boldsymbol{\mu}$

$$J = \sum_i (\hat{y}_i - y_i \log \hat{y}_i) = b \sum_i (e^{-\mathbf{P}_{i,\cdot}\boldsymbol{\mu}} - \cancel{\frac{y_i}{b} \log b} + \frac{y_i}{b} \mathbf{P}_{i,\cdot}\boldsymbol{\mu}) = b \sum_i (e^{-\mathbf{P}_{i,\cdot}\boldsymbol{\mu}} + \frac{y_i}{b} \mathbf{P}_{i,\cdot}\boldsymbol{\mu}) \qquad \text{(A.1)}$$

A.2 Cadre polychromatique

Maintenant : $\hat{y}_i = b \sum_{k=1}^{K} \alpha_k e^{-\sum_j a_{ij}\mu_{jk}}$
k : indice d'énergie ou de fréquence du rayonnement. On utilise la décomposition d'Avarez-Macovski (cf. **??**), paramétrisée adéquatement, pour représenter $\boldsymbol{\mu}$:

$$\boldsymbol{\mu} = b \sum_{k=1}^{K} \boldsymbol{\phi} \Phi_k + \boldsymbol{\theta} \Theta_k$$

Φ_k, Θ_k : coefficients scalaires connus
ϕ, θ : éléments de la décomposition de μ
Puis on reparamétrise empiriquement ϕ et θ par μ_{70}, atténuation à 70 keV

$$\boldsymbol{\mu} = b \sum_{k=1}^{K} \phi(\mu_{70}) \Phi_k + \theta(\mu_{70}) \Theta_k$$

où ϕ et θ sont les fonctions scalaires d'une variable scalaire.

A.3 Calcul de la log-vraisemblance

On a donc : $\hat{y}_i = b\sum_{k=1}^{K} \alpha_k e^{-(\Phi_k \mathbf{P}_{i,.}\phi(\boldsymbol{\mu})+\Theta_k \mathbf{P}_{i,.}\theta(\boldsymbol{\mu}))}$ où on omet l'indice de μ_{70} pour alléger les notations. Matriciellement :

$$\hat{\boldsymbol{y}} = b\sum_{k=1}^{K} \alpha_k e^{-\Phi_k \mathbf{P}\phi(\boldsymbol{\mu})+\Theta_k \mathbf{P}\theta(\boldsymbol{\mu})} \tag{A.2}$$

$$J(\boldsymbol{\mu}) = \sum_i (\hat{y}_i - y_i \log \hat{y}_i)$$

Par des développements analogues à ceux des cas monochromatiques, et en posant : $\tilde{\boldsymbol{y}} = \frac{\hat{\boldsymbol{y}}}{b}$

$$J(\boldsymbol{\mu}) = \cancel{b}\sum_i (\tilde{y}_i - \frac{y_i}{b}\log \tilde{y}_i) J(\boldsymbol{\mu}) = \sum_i \sum_k \alpha_k e^{-\Phi_k \mathbf{P}_{i,.}\phi(\boldsymbol{\mu})+\Theta_k \mathbf{P}_{i,.}\theta(\boldsymbol{\mu})} - \sum_i \frac{y_i}{b}\log(\sum_k \alpha_k e^{-\Phi_k \mathbf{P}_{i,.}\phi(\boldsymbol{\mu})+\Theta_k \mathbf{P}_{i,.}\theta(\boldsymbol{\mu})}) \tag{A.3}$$

L'évaluation de J peut se décomposer comme suit :

- calcul de $\mathbf{P}\theta(\boldsymbol{\mu})$ et $\mathbf{P}\phi(\boldsymbol{\mu})$ (deux projections)
- calcul du vecteur $\tilde{\boldsymbol{y}}$ selon l'équation **??**
- calcul de

$$J(\mu) = sum(\tilde{\boldsymbol{y}}) + \frac{1}{b}\boldsymbol{y}^T \log \tilde{\boldsymbol{y}}$$

A.4 Calcul du gradient de la neg-log-vraisemblance

On part de

$$J(\boldsymbol{\mu}) = \sum_i (\tilde{y}_i - \frac{y_i}{b}\log \tilde{y}_i)$$

$$\frac{\partial J(\boldsymbol{\mu})}{\partial \mu_j} = \sum_i (\frac{\partial \tilde{y}_i}{\partial \mu_j} - \frac{y_i}{b y_i}\frac{\partial \tilde{y}_i}{\partial \mu_j}) = \sum_i (1 - \frac{y_i}{b\tilde{y}_i})\frac{\partial \tilde{y}_i}{\partial \mu_j} \tag{A.4}$$

Calcul de $\frac{\partial \tilde{y}_i}{\partial \mu}$:

$$\tilde{y}_i = \sum_{k=1} K\alpha_k e^{-\Phi_k \mathbf{P}_{i,.}\phi(\boldsymbol{\mu})+\Theta_k \mathbf{P}_{i,.}\theta(\boldsymbol{\mu})}$$

On pose :

$$\tilde{y}_{i,k} = \alpha_k e^{-\Phi_k \mathbf{P}_{i,.}\phi(\boldsymbol{\mu})+\Theta_k \mathbf{P}_{i,.}\theta(\boldsymbol{\mu})} \tag{A.5}$$

et $\tilde{y_i} = \sum_{k=1\ K} \tilde{y}_{i,k}$

$$\frac{\partial \tilde{y}_{i,k}}{\partial \mu_j} = -a_{i,j}(\Phi_k \phi'(\mu_j) + \Theta_k \theta'(\mu_j))\tilde{y}_{i,k}$$

d'où $\frac{\partial \tilde{y_i}}{\partial \mu_j} = -\phi'(\mu_j)a_{i,j}\sum_k \Phi_k \tilde{y}_{i,k} - \theta'(\mu_j)a_{i,j}\sum_k \Theta_k \tilde{y}_{i,k}$ En utilisant **??**, on arrive à :

$$\frac{\partial J(\boldsymbol{\mu})}{\partial \mu_j} = -\phi'(\mu_j)\sum_i a_{i,j}(1 - \frac{y_i}{b\tilde{y}_i})(\sum_k \Phi_k \tilde{y}_{i,k}) - \theta'(\mu_j)\sum_i a_{i,j}(1 - \frac{y_i}{b\tilde{y}_i})(\sum_k \Theta_k \tilde{y_{i,k}})$$

En posant :
$\boldsymbol{e} = \{e_i = 1 - \frac{y_i}{b\tilde{y}_i}\}$
$\tilde{\boldsymbol{y}}^{\Phi} = \{\tilde{y}_i^{\Phi} = \sum_k \Phi_k \tilde{y}_{i,k}\}$
$\tilde{\boldsymbol{y}}^{\Theta} = \{\tilde{y}_i^{\Theta} = \sum_k \Theta_k \tilde{y}_{i,k}\}$
$\phi'(\boldsymbol{\mu}) = \{\phi'(\mu_j)\}$
$\theta'(\boldsymbol{\mu}) = \{\theta'(\mu_j)\}$
On peut écrire :

$$\Delta J(\boldsymbol{\mu}) = -\phi'(\boldsymbol{\mu}) \cdot \mathbf{P}^T(\boldsymbol{e} \cdot \tilde{\boldsymbol{y}}^{\Phi}) - \theta'(\boldsymbol{\mu}) \cdot \mathbf{P}^T(\boldsymbol{e} \cdot \tilde{\boldsymbol{y}}^{\Theta})$$

Les étapes du calcul du gradient peuvent donc être mises sous la forme :

- Calcul de $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{y}}^{\Phi}$ et $\tilde{\boldsymbol{y}}^{\Theta}$ conjointement, selon les équations précédentes
- Calcul de $\boldsymbol{e}$
- Calcul de $\mathbf{P}^T(\boldsymbol{e} \cdot *\tilde{\boldsymbol{y}}^{\Phi})$ et de $\mathbf{P}^T(\boldsymbol{e} \cdot *\tilde{\boldsymbol{y}}^{\Phi})$ (deux rétroprojections)
- Calcul du gradient.

Annexe B

Expression simplifiée du critère et de son gradient

B.1 Calcul de la log-vraisemblance

Rappels : On rappelle que l'on repart du critère polychromatique suivant (on garde les mêmes notations que précédemment) :
En posant : $\texttt{Proj(i,k)} = -\Phi_k \mathbf{P}_{i,.}\phi(\boldsymbol{\mu}) - \Theta_k \mathbf{P}_{i,.}\theta(\boldsymbol{\mu})$

$$J(\boldsymbol{\mu}) = \sum_i \{\sum_k \alpha_k e^{\texttt{Proj(i,k)}} - \frac{y_i}{b}\log(\sum_k \alpha_k e^{\texttt{Proj(i,k)}})\} \tag{B.1}$$

On pose $s_i = -\log\frac{y_i}{b}$.On fait alors apparaître e^{-s_i}.

$$J(\boldsymbol{\mu}) = \sum_i \{\sum_k \alpha_k e^{-s_i} e^{s_i + \texttt{Proj(i,k)}} - \frac{y_i}{b}\log(\sum_k \alpha_k e^{-s_i} e^{s_i + \texttt{Proj(i,k)}})\} \tag{B.2}$$

On pose : $h_{i,k} = s_i - \Phi_k \mathbf{P}_{i,.}\phi(\boldsymbol{\mu}) - \Theta_k \mathbf{P}_{i,.}\theta(\boldsymbol{\mu})$
L'équation devient alors : $J(\boldsymbol{\mu}) = \sum_i \{\sum_k \alpha_k e^{-s_i} e^{h_{i,k}} - \frac{y_i}{b}\log(\sum_k \alpha_k e^{-s_i} e^{h_{i,k})})$
On effectue un développement limité autour de h_i : en effet, pour des très petits comptes de photons, le paramètre de la loi de Poisson est très similaire à l'observation, d'où $y_i \approx \hat{y}_i$ donc $h_i \approx 0$ (à consolider)
On a : $e^{h_{i,k}} = 1 + h_{i,k} + \frac{h_{i,k}^2}{2}$.
d'où :

$$J(\boldsymbol{\mu}) = \sum_i \{\sum_k \alpha_k e^{-s_i}(1 + h_{i,k} + \frac{h_{i,k}^2}{2}) - \frac{y_i}{b}\log(\sum_k \alpha_k e^{-s_i}(1 + h_{i,k} + \frac{h_{i,k}^2}{2}))$$

$$J(\boldsymbol{\mu}) = \sum_i \{\sum_k \alpha_k e^{-s_i}(1 + h_{i,k} + \frac{h_{i,k}^2}{2}) - \frac{y_i}{b}\underbrace{\log(\sum_k \alpha_k(1 + h_{i,k} + \frac{h_{i,k}^2}{2}))}_{A} - \frac{y_i}{b}\log(e^{-s_i})$$

$$A = \log(\sum_k \alpha_k + \sum_k \alpha_k(h_{i,k} + \frac{h_{i,k}^2}{2}))$$

$$A = \log(1 + \sum_k \alpha_k (h_{i,k} + \frac{h_{i,k}^2}{2}))$$

En poursuivant le développement limité :

$$A = \sum_k \alpha_k (h_{i,k} + \frac{h_{i,k}^2}{2}) - \frac{1}{2}(\sum_k \alpha_k h_{i,k})^2$$

En se débarassant des termes qui ne dépendent pas de $\boldsymbol{\mu}$, on obtient la nouvelle expression du critère approximé, avec $s_i = \log(\frac{b}{y_i})$.

$$J(\boldsymbol{\mu}) = \frac{b}{2} \sum_i e^{-s_i} (s_i - (\sum_k \alpha_k \Phi_k)\mathbf{P}_{i,.}\phi(\boldsymbol{\mu}) - (\sum_k \alpha_k \Theta_k)\mathbf{P}_{i,.}\theta(\boldsymbol{\mu}))^2$$

De manière vectorielle, la dernière expression peut se simplifier en :

$$J(\boldsymbol{\mu}) = \texttt{Sum}(\frac{\boldsymbol{y}}{2}. * (\mathbf{S})\hat{}2)$$

avec

$$\mathbf{S} = \boldsymbol{s} - p\mathbf{P}\Phi(\boldsymbol{\mu}) - t\mathbf{P}\Theta(\boldsymbol{\mu})$$

avec

$$\boldsymbol{p} = \sum_k \alpha_k \Phi_k$$

$$\boldsymbol{t} = \sum_k \alpha_k \Theta_k$$

B.2 Calcul du gradient de la neg-log-vraisemblance

Calcul du gradient de cette approximation// On pose :

$$J_i(\boldsymbol{\mu}) = \frac{\boldsymbol{y}_i}{2} * (\mathbf{S}_i)^2$$

Alors :

$$\frac{\partial J_i(\boldsymbol{\mu})}{\partial \mu_j} = \boldsymbol{y}_i * \frac{\partial \mathbf{S}_i}{\partial \mu_j} * \mathbf{S}_i$$

En remplaçant :

$$\frac{\partial J_i(\boldsymbol{\mu})}{\partial \mu_j} = \boldsymbol{y}_i * (-\boldsymbol{p}\mathbf{P}_{i,j}\Phi'(\mu_j) - \boldsymbol{t}\mathbf{P}_{i,j}\Theta'(\mu_j)) * (\boldsymbol{s} - \boldsymbol{p}\mathbf{P}_{i,.}\Phi(\boldsymbol{\mu}) - \boldsymbol{t}\mathbf{P}_{i,.}\Theta(\boldsymbol{\mu}))$$

On peut réordonner les termes car on travaille ici avec des scalaires :

$$\frac{\partial J_i(\boldsymbol{\mu})}{\partial \mu_j} = (-\boldsymbol{p}\Phi'(\mu_j) - \boldsymbol{t}\Theta'(\mu_j)) * \mathbf{P}_{i,j}(\boldsymbol{s} - \boldsymbol{p}\mathbf{P}_{i,.}\Phi(\boldsymbol{\mu}) - \boldsymbol{t}\mathbf{P}_{i,.}\Theta(\boldsymbol{\mu})) * \boldsymbol{y}_i$$

Ceci est une forme qui peut s'exprimer plus simplement de manière vectorielle :

$$\nabla_\mu J(\boldsymbol{\mu}) = (-\boldsymbol{p}\Phi'(\mu) - \boldsymbol{t}\Theta'(\mu)). * \mathbf{P}^T(\boldsymbol{s} - \boldsymbol{p}\mathbf{P}\Phi(\boldsymbol{\mu}) - \boldsymbol{t}\mathbf{P}\Theta(\boldsymbol{\mu})). * \boldsymbol{y}$$

soit avec les notations précédentes :

$$\nabla_\mu J(\boldsymbol{\mu}) = (-\boldsymbol{p}\Phi'(\mu) - \boldsymbol{t}\Theta'(\mu)).*\mathbf{P}^T.(\mathbf{S}.*\boldsymbol{y})$$

Ce qui s'implémente facilement dans Matlab :

— On calcule $\boldsymbol{y}$

— On calcule $\mathbf{P}^T.(\mathbf{S}.*\boldsymbol{y})$ avec la fonction `A.RProject(S*y)`

— On calcule $(-\boldsymbol{p}\Phi'(\mu) - \boldsymbol{t}\Theta'(\mu))$