Contributions pour les réseaux de neurones

Nos travaux s'inscrivent dans l'axe de recherche des réseaux de neurones utilisés en tant qu'approximateurs universels de fonctions. Dans ce chapitre, nous allons présenter les objectifs mais aussi le cheminement qui nous a permis de construire un algorithme d'apprentissage incrémental particulièrement adapté à notre domaine applicatif qu'est la radiothérapie externe.

3.1 Algorithme de construction incrémentale du réseau de neurones

L'un des points de départ de nos travaux a consisté en la réalisation d'un état de l'art des différentes techniques d'apprentissage des réseaux neuronaux. Ce travail a eu pour but d'identifier l'algorithme d'apprentissage le plus efficace dans le cadre de notre étude liée à la radiothérapie externe.

La solution que nous avons retenue consiste à associer un algorithme d'apprentissage de type RPROP (voir 2.4.1) avec une architecture de réseau du second ordre de type HPU (voir 2.3.2). Cette association d'optimisation nous a permis de mettre au point un algorithme d'apprentissage efficace et rapide mais qui ne nous permet pas de garantir que le réseau final obtenu possède un nombre minimal de neurones. Comme on peut le voir sur les différentes représentations des courbes de dépôt de doses, les caractéristiques des courbes servant de référence à l'apprentissage sont de posséder des zones relativement constantes encadrées par des zones à fortes pentes.

Nous nous sommes attachés à mettre au point un algorithme de construction incrémentale. Cet algorithme a pour objectif de conserver les optimisations précédentes tout en limitant le nombre de neurones sur la couche cachée du réseau.

Le principe de notre algorithme, schématisé à la figure 3.1 et détaillé de manière complète dans l'algorithme 3.1, est de réaliser un apprentissage de type RPROP sur une architecture de réseau de type HPU. Le nombre d'entrées, comme de sorties, est défini en fonction des données d'apprentissage. Le nombre de neurones sur la couche cachée (NCC)

Algorithme 3.1 Algorithme d'apprentissage incrémental

```
Entrées:
  Réseau Res                  // réseau à entraîner
  Réel E_courant, Précision   // erreur du réseau et précision souhaitée
Variables:
  Réseau Res_prec             // réseau intermédiaire
  Réel E_prec                 // erreur du réseau intermédiaire
  Booléen Continue            // indique si le processus continue
Résultats:
  Réseau Res                  // le réseau entraîné
  Réel E_courant              // l'erreur finale après entraînement

  // boucle générale du processus d'apprentissage
  Répéter
    // copie du réseau courant
    Res_prec ← Res
    E_prec ← E_courant
    // apprentissage du réseau courant
    E_courant ← ApprentissageRPROP(Res)
    // détermination de l'ajout d'un nouveau neurone ou d'arrêt du processus
    Continue ← Faux
    Si E_courant > Précision Alors
      Si E_courant < E_prec Alors
        Si NCC <NCC_max Alors
          AjoutNeuroneCC(Res) ;
          Continue ← Vrai
        Finsi
      Sinon
        Res ← Res_prec
      Finsi
    Finsi
  Tant que Continue = Vrai
```

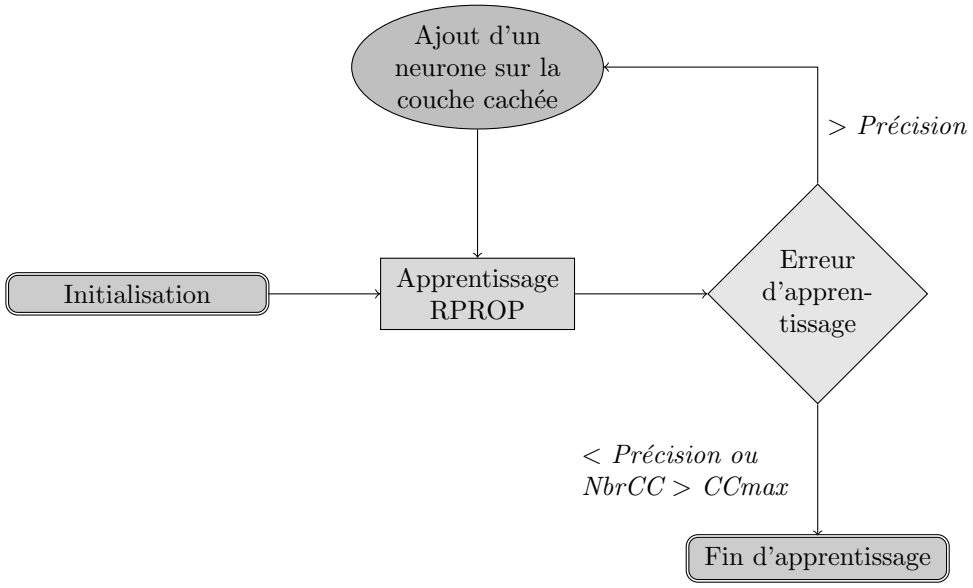

FIG. 3.1 – Principe de notre algorithme d'apprentissage incrémental

est, quant à lui, choisi en fonction de l'expérience mais toujours le plus petit possible. Ce nombre de neurones sur la couche cachée évoluera au cours de l'apprentissage. Une fois le réseau construit, l'apprentissage proprement dit est lancé. Il s'agit de présenter au réseau les différents ensembles du domaine d'apprentissage et de corriger les différents poids reliant les neurones entre eux de manière à minimiser l'erreur sur le neurone de sortie. À partir du moment où la diminution de l'erreur de sortie stagne et dans le cas où l'erreur finale obtenue ($E_{courant}$) est supérieure à celle décidée (*Précision*), un nouveau neurone est ajouté sur la couche cachée. Ce nouveau neurone est inséré dans le réseau avec des poids nuls afin de ne pas apporter d'erreur supplémentaire, comme il est illustré dans la figure 3.2.

Ensuite, l'apprentissage est de nouveau lancé jusqu'à la prochaine période de stagnation. Ce processus d'apprentissage (*ApprentissageRprop*()) et d'ajouts de neurones sur la couche cachée ($ajoutNeuroneCC()$) alterne tant que l'erreur voulue n'est pas atteinte ou qu'un ajout de neurones sur la couche cachée n'améliore plus le résultat du réseau de neurones.

Comme les autres algorithmes d'apprentissage de réseaux de neurones, celui-ci incorpore un mécanisme permettant de contrôler que le réseau n'entre pas dans un état de sur-apprentissage. De même, une limite maximum (NCC_{max}) pour le nombre de neurones possible sur la couche cachée a été incluse dans le processus d'apprentissage afin de limiter la taille du réseau de neurones, et ainsi de limiter le temps d'apprentissage dans les cas où la convergence est très lente.

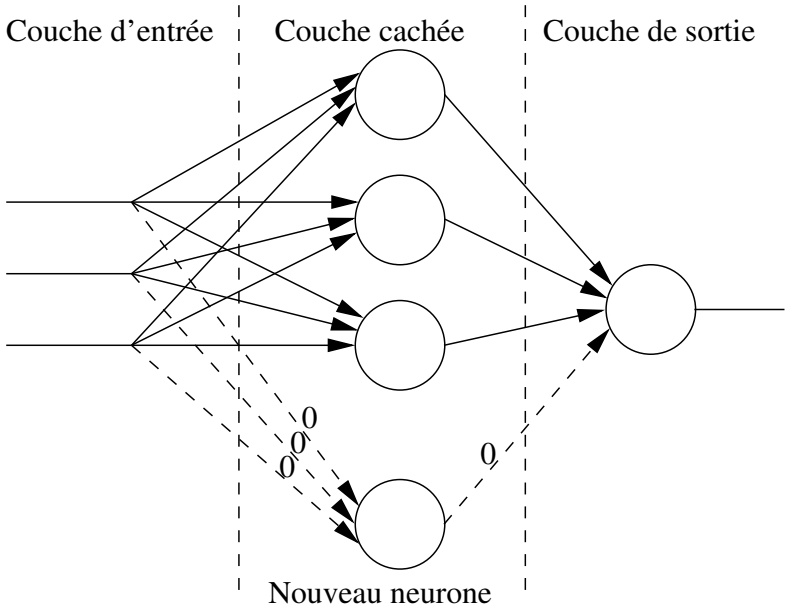

FIG. 3.2 – Ajout d'un nouveau neurone sur la couche cachée du réseau.

Comme il est montré dans le chapitre 5, les performances atteintes par cet algorithme d'apprentissage sont très bonnes pour un intervalle d'apprentissage expérimental mais restent insuffisantes dans le cadre d'une mise à l'échelle. Afin de permettre la prise en charge des domaines d'apprentissage complets, il est nécessaire de mettre en place des techniques de parallélisation de l'apprentissage.

3.2 La parallélisation

Le principe utilisé pour réaliser la parallélisation de l'algorithme d'apprentissage est le suivant : puisque l'apprentissage d'un domaine de données restreint est pleinement satisfaisant et puisque l'apprentissage d'un domaine complet nécessite un temps trop important, il peut être intéressant de décomposer le domaine d'apprentissage en sous-domaines.

Le principe est ensuite de construire un réseau de neurones pour chacun des sous-domaines composant l'ensemble d'apprentissage. Il faut ensuite concevoir un mécanisme permettant d'utiliser cet ensemble de sous-réseaux de neurones de manière à obtenir un méta-réseau pour le domaine complet.

3.2.1 Principe de décomposition du domaine

Le principe de décomposition du domaine d'apprentissage est élémentaire. Dans cette première version de l'algorithme d'apprentissage parallèle, il est possible de construire des ensembles d'apprentissage en décomposant de manière équivalente les différentes entrées choisies. Ce choix de découpage, s'il s'avère simpliste à première vue, a été ef-

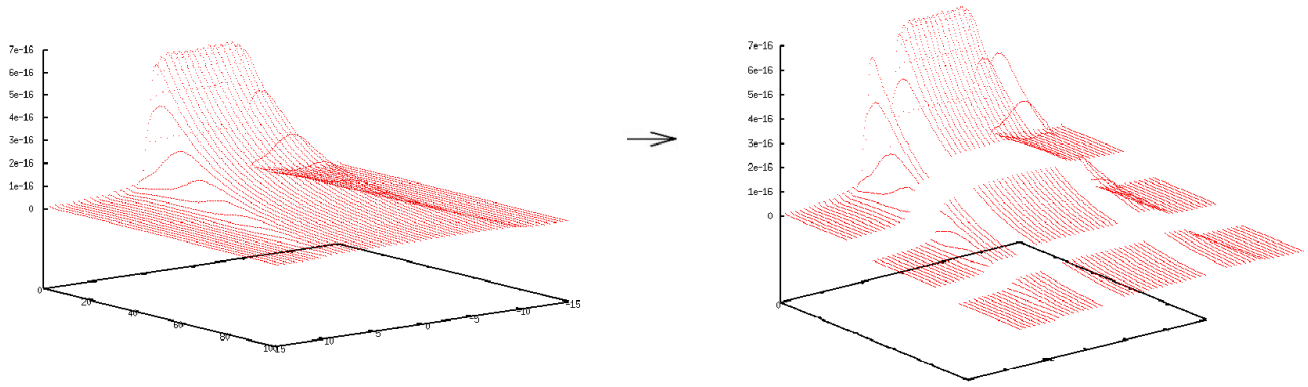

FIG. 3.3 – Décomposition d'un domaine d'apprentissage en 3x3 sous-ensembles

fectué en rapport à notre application. La plupart des paramètres de notre application ont un domaine assez restreint et seuls les paramètres liés à la description de l'environnement recouvrent un domaine suffisamment étendu pour être découpés. De plus, comme la grille de discrétisation de notre environnement est construite de manière uniforme, un découpage régulier permet de construire des ensembles de données d'apprentissage de tailles équivalentes. La figure 3.3 présente neuf sous-ensembles d'apprentissage qui, regroupés, représentent la courbe de dépôt de doses dans un volume homogène composé d'eau. Le premier intérêt de la décomposition de domaine est de réaliser des apprentissages avec des ensembles de taille relativement restreinte. Un autre intérêt, non négligeable, est que la complexité de chaque sous-domaine est très inférieure à celle du domaine complet, ce qui rend leur apprentissage beaucoup moins complexe.

3.2.2 Rôle et importance du recouvrement

L'inconvénient apporté par ce principe de construire n sous-réseaux de neurones à la place d'un réseau unique vient de la gestion des interfaces entre chaque sous-réseau. Si les réseaux de neurones ont la capacité de généralisation sur un domaine d'apprentissage fini, ils n'ont pas la même précision sur l'ensemble du domaine. Précisément, les zones se trouvant en limite de domaine présentent une erreur supérieure à la moyenne du réseau. Le fait de multiplier le nombre de réseaux augmente les zones de fortes erreurs lors de la généralisation, et donc, nuit à la précision générale.

Le principe mis en place pour diminuer ce nouvel apport d'erreur est d'augmenter la taille du domaine d'apprentissage lié à chaque sous-réseau. Ainsi, chaque sous-réseau a un domaine d'apprentissage supérieur à son domaine d'application et les zones à fort pourcentage d'erreur ne sont pas utilisées dans le calcul du réseau global. La figure 3.4 présente la mise en place du recouvrement pour le sous-domaine central de la décomposition présentée à la figure 3.3. Le taux de recouvrement doit être choisi à bon escient afin de minimiser l'erreur aux jointures des sous-réseaux tout en limitant au maximum la zone de recouvre-

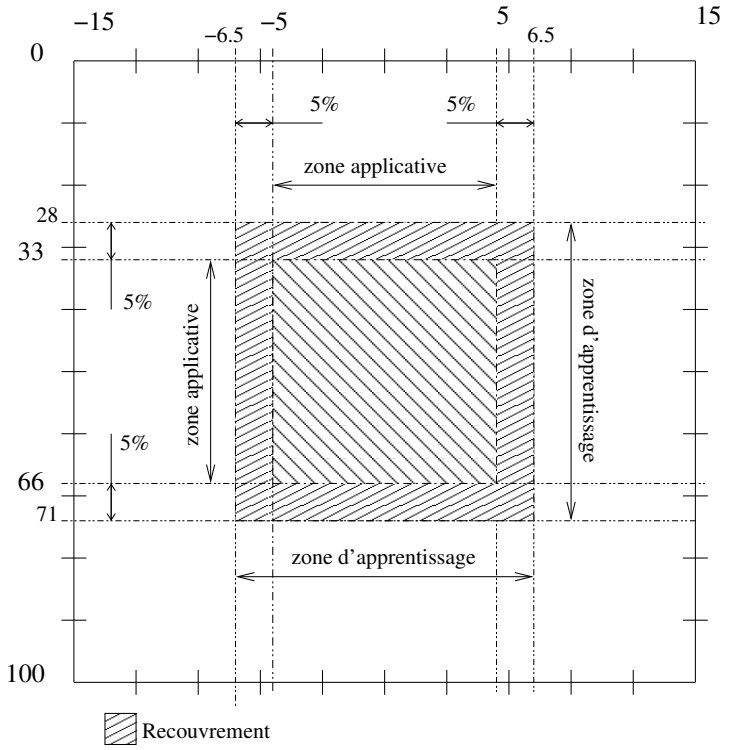

FIG. 3.4 – Mise en place du recouvrement

ment. Cela est important de minimiser l'augmentation du temps d'apprentissage due à l'augmentation de la taille du domaine d'apprentissage.

3.2.3 L'algorithme parallèle d'apprentissage incrémental

L'algorithme parallèle d'apprentissage incrémental est mis au point pour être utilisé sur une grappe de calcul. Il est donc conçu pour prendre en compte les caractères spécifiques à cette architecture. Le premier de ces caractères est qu'une grappe de calcul peut être composée de machines hétérogènes et non fiables. Il a donc fallu mettre en place un algorithme nécessitant peu de points de synchronisation et prévoir des mécanismes permettant la reprise d'un apprentissage dans le cas où le nœud le réalisant subirait une défaillance.

L'algorithme mis en place est basé sur un modèle maître-esclave. L'algorithme 3.2 décrit la partie maître et l'algorithme 3.3, la partie esclave. La première phase active de ces algorithmes est réalisée par le maître. Dans un premier temps il met en place les structures lui permettant d'enregistrer les sous-réseaux de neurones (*Reseau*) ainsi que la structure lui permettant de contrôler le bon déroulement de l'apprentissage (*EtatG*). *EtatG* liste pour chacun des nœuds : son état ($ATTENTE, APPENCOURS, PANNE$), l'indice du sous-réseau en cours d'apprentissage, ainsi que la date du dernier message reçu. Ces différentes informations sont stockées dans une structure détaillée dans la section 3.2.4.

Algorithme 3.2 Algorithme parallèle d'apprentissage incrémental (Maître)

```
Entrées:
  Réseau [ ] TReseau // Meta-réseau à entraîner
  Entier nbSr,nbProc  // Nombre de sous-réseaux, de nœuds disponibles
Variables:
  EtatG eProc          // Structure décrivant l'état des différents noeuds
  EnsApp [ ] ensApp // Tableau décrivant les différents ensembles d'apprentissage
  Entier [ ] listeApp  // Tableau contenant la liste des apprentissages non effectués
  Entier taille        // Nombre d'apprentissages restant
  Entier sd            // Indice du sous-réseau en cours de traitement
  Booléen appEncours // État de l'apprentissage
Résultats:
  Réseau [ ] TReseau // Meta-réseau entraîné

  // Initialisation des structures globales
  eProc ← new [nbProc] EtatG
  // Boucle générale du processus d'apprentissage
  Tantque appEncours = Vrai Faire
    Pour i de 0 à nbProc−1 Faire
      Si messageRecu(i,msg) Alors
        Si msg = ATTENTE Alors
          taille ← evalueApprentissageRestant(listeApp)
          Si taille > 0 Alors
            sd ← apprentissageSuivant(listeApp)
            avertissementEnvoiApp(i)
            envoiDonnées(i, sd)
            envoiSousReseau(TReseau[sd])
            indiqueApprentissageEnCours(EtatG,i,sd)
          Sinon
            envoiMsgAttente(i)
          Finsi
        Sinon Si msg = FINAPP Alors
          TRéseau[i] ← receptionReseau(i)
          envoiMsgAttente(i)
        Sinon Si msg = ENCOURS Alors
          misAJourTemoinVivant(i, EtatG)
        Sinon Si msg = SAUVEGARDE Alors
          TReseau[i] ← receptionReseau(i)
        Finsi
      Finsi
    Fin pour
    verificationVitalite(EtatG, listeApp)
    appEncours ← verificationFinApp(EtatG, listeApp)
  Fin tantque
  sauvegarde(Réseau)
```

Une fois que ces structures sont initialisées, le maître entre dans la phase de distribution/sauvegarde des différents réseaux composant le domaine complet d'apprentissage.

Lors d'une boucle d'apprentissage, le maître écoute chacun des différents nœuds et réagit en fonction du message reçu. Dans le cas où le maître reçoit un message indiquant que le nœud est dans une phase inactive, phase d'$ATTENTE$, le maître contrôle s'il lui reste des apprentissages non effectués ; dans ce cas, il envoie au nœud un sous-domaine d'apprentissage et le sous-réseau correspondant. Il est important de noter que le sous-réseau envoyé par le maître n'est pas obligatoirement un nouveau réseau non appris mais peut être un réseau temporaire issu d'un nœud ayant subi une défaillance. Dans ce cas, le nœud reprend un apprentissage en utilisant le dernier point de sauvegarde récupéré avant la panne. Ainsi, la perte de temps due à la défaillance d'un nœud est limitée.

L'algorithme "esclave" réalise les opérations complémentaires au maître : suite à l'annonce de l'envoi d'un apprentissage, le nœud réceptionne les données d'apprentissage (*donApp*) puis le réseau de neurones associé. Ensuite, le nœud effectue l'apprentissage proprement dit du réseau en utilisant l'algorithme d'apprentissage incrémental décrit précédemment (voir la section 3.1). Durant cette étape d'apprentissage, le nœud signifie son état au maître en envoyant à intervalles réguliers l'évaluation de son erreur d'apprentissage ainsi que, à intervalles un peu moins fréquents, une copie de sauvegarde de son réseau de neurones résultat.

Durant cet apprentissage, le maître continue de lancer sur l'ensemble des nœuds disponibles la suite des sous-réseaux. Une fois qu'il n'y a plus de nœud disponible, le maître attend qu'un nœud ait fini son apprentissage pour lui en redonner un nouveau. Le maître et les nœuds bouclent ainsi tant que l'ensemble des sous-réseaux n'a pas été complètement entraîné. Une fois ceci fait, le maître envoie un message aux différents nœuds leur indiquant la fin du processus global, puis sauvegarde la structure complète des différents sous-réseaux.

Algorithme 3.3 Algorithme parallèle d'apprentissage incrémental (Esclave)

```
Entrées:
  Réseau Res        // Réseau à entraîner
  EnsApp donApp // Domaine d'apprentissage à utiliser pour l'apprentissage
Variables:
  Msg msg               // Message indiquant le type d'action à réaliser par le noeud
  Booléen appEncours // État de l'apprentissage
Résultats:
  Réseau Res        // Réseau entraîné

  // Boucle d'écoute du maître
  Tantque appEncours = Vrai  Faire
    Si  msgRecu(msg) Alors
      Si msg = APP Alors
        // Lancement d'une phase d'apprentissage
        receptionDonneeApp(donApp)
        receptionReseau(Res)
        apprentissageIncremental(Res, donApp)
        envoiFinApp()
        envoiReseau(Res)
        msg ← ATTENTE
      Sinon Si msg = ATTENTE Alors
        // Cas où le maître n'a pas de tâche à réaliser
        attente(cstT)
        envoitVivant()
      Sinon Si msg = FINAPP Alors
        // Fin du processus d'apprentissage global
        appEncours ← Faux
      Finsi
    Finsi
  Fin tantque
```

3.2.4 Tolérance aux pannes

L'algorithme étant destiné à être mis en place sur une grappe de calcul utilisant des machines non fiables, il est nécessaire de construire un mécanisme assurant que même en cas de perte d'un nœud, l'apprentissage n'est pas perdu. Le principe utilisé consiste à ce que chaque nœud envoie régulièrement un message d'activité au maître pendant les phases d'apprentissage.

```
class Etat{
   int proc;      // Rang du processus (sert aussi de d'indice pour MPI).
   int indSD;     // Indice du sous-domaine en cours d'apprentissage;
   TypeEtat etat; // Definit l'état du processus.
   double erreur; // Erreur de l'apprentissage (si TypeEtat == APP).
   time_t dApp;   // Date de début d'apprentissage
   time_t dSauv;  // Date dernière sauvegarde
   time_t dViva;  // Date de message de vivacité
  }
```

FIG. 3.5 – Structure permettant la description de l'état d'un nœud

Sur le maître, il est mis en place, comme mentionné précédemment, une structure, décrite dans la figure 3.5 permettant de connaître l'état de chacun des nœuds. Au cours de la réalisation d'un apprentissage, le nœud envoie au maître à intervalles réguliers une sauvegarde de son réseau en cours d'apprentissage. Ces envois sont réalisés à plusieurs moments de l'apprentissage ; à chaque fois qu'un nouveau neurone est ajouté sur la couche cachée, et aussi toutes les n itérations durant la phase d'apprentissage Rprop. La valeur de n est un paramètre modifiable dans la configuration de l'apprentissage. À chaque envoi, le maître modifie la valeur du paramètre $dSauv$. De plus, le nœud envoie à chaque boucle d'apprentissage un message indiquant sa vitalité. La date de réception de ce message est ensuite enregitrée à l'aide du paramètre $dViva$ dans la structure associée au nœud sur le maître. Pour détecter la défaillance d'un nœud, il suffit de vérifier que la durée entre la date du dernier message et le moment de vérification ne dépasse pas un certain seuil. Pour ce faire, il suffit de comparer la valeur du paramètre $dViva$ avec la date courante. Si le seuil est dépassé, le nœud est indiqué comme défaillant en modifiant la valeur du paramètre *etat*, et le maître ne cherchera plus à lui envoyer de message. Si un apprentissage était en cours sur ce nœud, celui-ci est replacé dans la liste des apprentissages restant à accomplir. Le nœud qui reprendra cet apprentissage utilisera comme base le dernier point de sauvegarde enregistré.

Si un nœud indiqué comme défaillant reprend contact avec le maître et que le sous-domaine dont il avait la charge a déjà été redistribué à un autre nœud, les deux nœuds sont en concurrence tant qu'aucun ne termine son apprentissage. La raison pour laquelle les deux nœuds sont gardés en concurrence vient du fait que si un nœud a déjà été signalé

comme défaillant, il peut être classé dans les éléments non-fiables. Il est donc risqué, dans un premier temps, de redonner une charge de calcul à un tel élément. Un fois la convergence de l'apprentissage atteinte sur l'un des deux nœuds, un message est envoyé à l'autre pour lui indiquer qu'il peut stopper son apprentissage et se remettre en position d'attente.

3.2.5 Activation du réseau parallèle

L'utilisation de plusieurs réseaux de neurones à la place d'un réseau unique modifie la fonction d'activation classique. L'algorithme 3.4 décrit le mécanisme mis en place pour prendre en charge cette architecture particulière. L'ensemble des sous-réseaux représentant le réseau parallèle est contenu dans un tableau (*TReseau*).

Algorithme 3.4 Activation du réseau de neurones parallèle

Entrées:
Réseau [ ] TReseau // *Meta-réseau entraîné*
Variables:
Entier taille // *Indique le nombre de réseaux contenus dans le meta-réseau*
Résultats:
Booléen trouve // *Indique la validité de l'activation*
Réel sortie // *Valeur de l'activation du méta-réseau*

```
trouve ← Faux
taille ← nbreSousReseau(TReseau)
// Boucle sur les sous-réseaux tant qu'aucun n'a pu répondre aux entrées passées en paramètres
Tantque ((i < taille) et (trouve = Faux)) Faire
  trouve ← activeReseau(TReseau[i],Entrees)
  i = i + 1
Fin tantque
// Un sous-réseau correspond au domaine des entrées
Si trouve = Vrai Alors
  sortie ← valSortie(TReseau[i])
Finsi
```

La fonction d'activation consiste à appeler itérativement sur chacun des sous-réseaux leur propre fonction d'activation (activeReseau(réseau[i],Entrees)). La boucle s'arrête dès qu'un sous-réseau a pu s'activer sur les entrées (*Entrees*) passées en paramètre à sa propre fonction d'activation. Une fois qu'un sous-réseau a pu s'activer, ses résultats sont recopiés dans le vecteur de sortie (*Sortie*). La fonction retourne ensuite un booléen permettant de savoir si les entrées présentées au réseau de neurones correspondaient bien à son domaine d'apprentissage.

Au niveau des sous-réseaux de neurones, l'activation du réseau reste classique, comme on peut le constater dans l'algorithme 3.5. La première phase effectuée par cet algorithme est de vérifier son mode de fonctionnement. Dans le cas où le réseau de neurones serait en cours d'apprentissage, la fonction contrôlerait la conformité des entrées (*Entrees*) en se basant sur la définition du domaine d'apprentissage comprenant le recouvrement (*absolu*) alors que dans le cas d'une activation du réseau en mode $UTILISATION$, le domaine de référence est le domaine $applicatif$. La seconde étape consiste à calculer les corrélations des entrées pour satisfaire l'architecture HPU. Il ne reste plus qu'à propager les valeurs d'entrées sur la couche cachée puis sur la couche de sortie pour obtenir le résultat. La fonction d'activation retourne un booléen permettant de savoir si les données passées en paramètre appartiennent ou non au domaine de définition du réseau.

Algorithme 3.5 Activation du réseau de neurones

Entrées:
  Réseau reseau // *Réseau entraîné*
  Réel [ ] Entrées // *Entrées devant être activées*
  DomaineDef absolu // *Domaine de définition pour l'apprentissage*
  DomaineDef applicatif // *Domaine de définition pour l'utilisation*
  TypeMessage mode // *Type d'activation souhaité*
Variables:
  Booléen appartient // *Permet de vérifier l'appartenance à un domaine de définition*
  Réel [ ] EntréesC // *Entrées artificielles + Entrées initiales*
Résultats:
  Réel [ ] Sorties // *Valeur de sortie du réseau après activation*
  Booléen appartient // *indique la validité de l'activation*

```
  Si mode = APPRENTISSAGE Alors
    appartient ← appartientDomaine(réseau, Entrées, absolu)
  Sinon Si mode = UTILISATION Alors
    appartient ← appartientDomaine(réseau, Entrées, applicatif)
  Finsi
  Si appartient = Vrai Alors
    entréesC ← calculCorrelation(réseau, Entrées)
    activationCoucheCachee(réseau, EntréesC)
    Sorties ← activationCoucheSortie(réseau)
  Finsi
  Retourne appartient
```

Comme il a déjà été mentionné, le réseau de neurones a été préalablement entraîné avec des données provenant uniquement de milieux homogènes. Le réseau ne peut donc pas directement gérer les comportements particuliers apparaissant aux changements de matériaux dans le cas d'une évaluation dans un milieu hétérogène. C'est pourquoi nous avons dû mettre en place un algorithme spécifique, permettant de calculer les courbes de doses dans des milieux hétérogènes en utilisant le réseau précédemment entraîné.

Chapitre 4

Algorithme d'évaluation de doses

Lorsqu'un faisceau irradiant entre dans une cellule, seule une partie de l'énergie contenue dans le faisceau est déposée dans la cellule traversée. Une seconde partie du faisceau est déviée à l'ensemble des cellules environnantes (phénomène de diffusion de la dose). Enfin, la dernière partie du faisceau traverse la cellule sans réaliser d'interaction. Les deux principaux paramètres qui influencent la courbe du dépôt de doses sont le spectre en énergie du faisceau incident, et la densité de la matière composant la cellule irradiée.

Actuellement, la majorité des traitements radio-thérapeutiques réalisés est effectuée à l'aide d'un ensemble d'irradiations utilisant toutes le même faisceau. Nous avons donc focalisé notre étude sur le paramètre évolutif pouvant être rencontré, à savoir la densité des milieux traversés. Nous avons toutefois gardé la possibilité d'incorporer dans notre système la fonctionnalité permettant d'utiliser plusieurs énergies de faisceau irradiant. Cette prise en charge de paramétrages différents de la source irradiante se limite à ajouter à notre réseau de neurones une nouvelle entrée caractérisant cette information complémentaire.

Ces algorithmes sont élaborées en s'appuyant sur le principe que la valeur de la dose déposée est un phénomène continu. De plus, nous avons comme seconde hypothèse que l'élément prépondérant pour le dépôt de dose se caractérise par la ou les densités de l'environnement étudié. En effet, la densité des matériaux a une influence importante sur la courbe de distribution de doses, comme il est possible de le voir dans la figure 4.1 (gauche) présentant deux courbes de doses dans deux milieux homogènes différents en fonction de la profondeur. En effet, plus la densité d'une matière est importante, plus la dose absorbée est importante, et donc, plus la décroissance de la courbe de dépôt de dose est rapide. Quant à la continuité des courbes de distribution de doses, elle est garantie dans tous les cas d'irradiation, même lorsqu'il y a présence d'hétérogénéité des milieux.

Afin de faciliter les comparaisons entre nos fichiers références et nos fichiers calculés, le format de fichier utilisé lors de notre étude pour la description des milieux est le format egsphant. Ce format de fichier a été mis au point pour le logiciel EgsNrc [7], il permet de mettre en place une discrétisation uniforme du milieu étudié. Notre travail a tout d'abord porté sur l'évaluation des milieux en deux dimensions pour être ensuite généralisé aux

milieux tridimensionnels. Seuls les voisins directs (selon les axes $\overrightarrow{x}$, $\overrightarrow{y}$ et $\overrightarrow{z}$) du voxel en cours d'évaluation sont pris en compte lors de l'évaluation de la dose déposée, c'est pourquoi toutes les interfaces entre milieux peuvent être considérées comme des interfaces soit longitudinales, soit perpendiculaires au faisceau irradiant.

4.1 Phénomène physique aux interfaces entre matériaux

Une interface perpendiculaire se produit lorsqu'il existe une suite de matières perpendiculaires à l'axe du faisceau irradiant. Du fait de la continuité de la valeur de la dose déposée, la valeur de dose est la même de part et d'autre de l'interface. En général, il y a un petit artefact dû soit à un excès, soit à un défaut de contamination électronique, mais, au vu de son peu d'importance, il sera ignoré au cours de cette étude. Algorithmiquement, une interface perpendiculaire peut être vue comme une simple juxtaposition de courbes de rendement en effectuant les décalages nécessaires au maintien du critère de continuité de la courbe. Les figures 4.1 illustrent ce phénomène. La partie de gauche présente deux courbes de rendement en fonction de la profondeur dans le milieu, une dans l'eau et l'autre dans le titane. La figure de droite présente, quant à elle, une unique courbe de rendement dans un milieu hétérogène composé d'eau puis de titane. On peut voir que cette courbe peut être construite en utilisant dans une première partie, la courbe de rendement dans un milieu homogène composé d'eau, puis dans la seconde partie, celle calculée dans un milieu de titane en effectuant le décalage en profondeur indiqué dans la figure de gauche. Notre travail consistera à construire un algorithme permettant d'effectuer ce traitement de manière automatique quels que soient les deux milieux concernés.

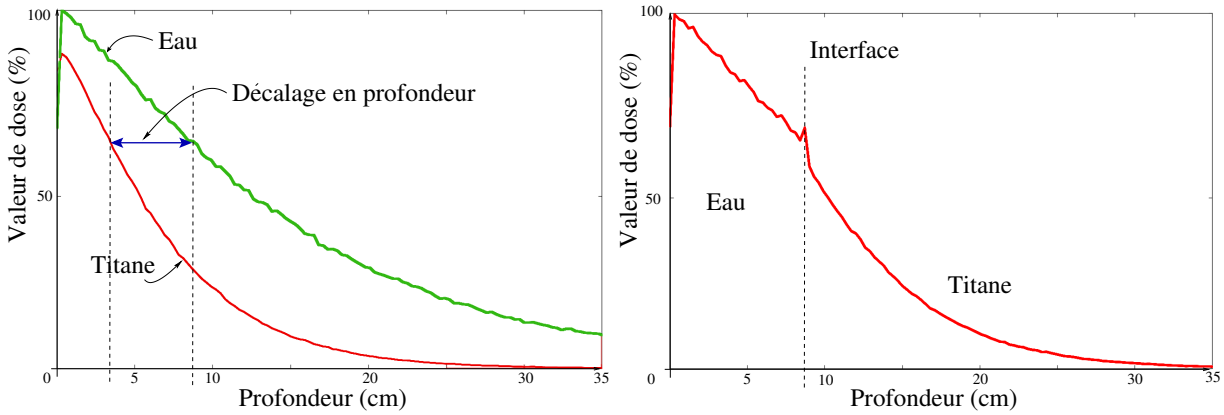

FIG. 4.1 – Rendements en profondeur dans des milieux homogènes composés d'eau ou de titane (figure de gauche). Rendement en profondeur dans un milieu hétérogène composé d'eau puis de titane (interface à 8.7 cm) (figure de droite)

Les modifications apportées par les interfaces entre matières ne sont pas uniquement présentes dans le cas des interfaces en profondeur, mais également dans le cas des interfaces

latérales. La figure 4.2 (gauche), présentant sur une même figure deux profils pour des milieux composés respectivement d'eau et de titane, permet de mettre en évidence le comportement de ce type d'interfaces. Le profil de rendement d'un milieu présentant une interface latérale est présenté à la figure 4.2 (droite). Dans ce contexte, l'influence de l'interface est due au phénomène de diffusion de la dose dans l'environnement irradié. Cette influence se traduit dans une première approximation par un lissage sur la largeur du milieu. Comme pour les interfaces en profondeur, si les différences de milieux sont vraiment importantes, il apparaît des artefacts sur l'interface ; mais, pour les mêmes raisons que dans le cadre des interfaces en profondeur, ces comportements ne sont pas pris en compte dans cette étude et devront faire l'objet d'une étude ultérieure pour affiner la précision de la méthode.

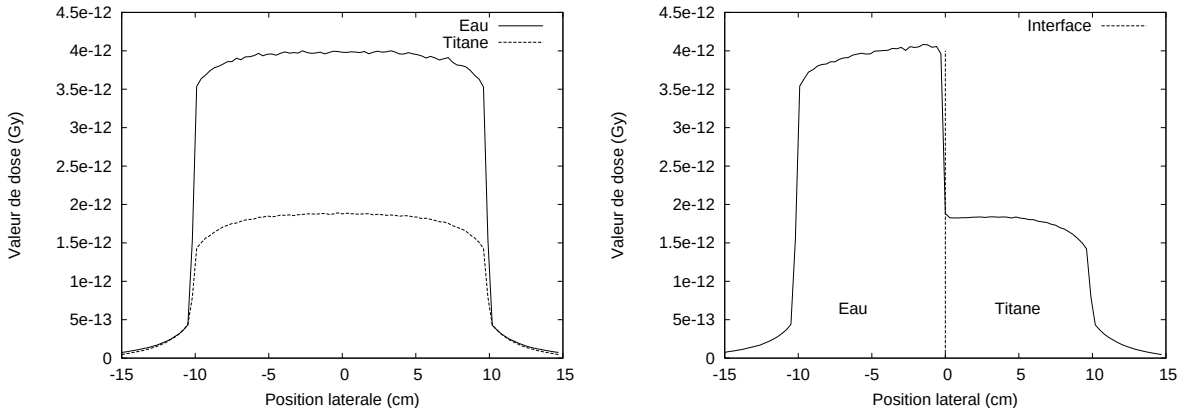

FIG. 4.2 – Profils pour une distribution de doses dans des milieux homogènes composés d'eau et de titane (figure de gauche). Profil pour une distribution de doses dans un milieu hétérogène composé d'eau puis de titane (figure de droite)

4.2 Algorithme d'évaluation de doses

L'algorithme d'évaluation a pour objectif de calculer la distribution de dose pour un milieu en trois dimensions. La caractéristique innovante de cet algorithme est de se baser sur un réseau de neurones pour évaluer les doses déposées dans les milieux hétérogènes. Le but est donc de mettre en place des mécanismes permettant l'utilisation de ce réseau de neurones malgré l'hétérogénéité des environnements étudiés. Cette utilisation est permise en calculant pour chaque voxel étudié, sa position corrigée. Cette correction est établie en prenant en compte le parcours du faisceau irradiant entre la sortie de la tête de l'accélérateur et le voxel étudié. Cet algorithme est composé de la prise en charge de plusieurs points particuliers que sont : la gestion des différentes interfaces rencontrées ainsi que le type d'irradiation du voxel étudié (cas d'un voxel appartenant ou non au faisceau irradiant comme il est décrit dans la figure 4.3).

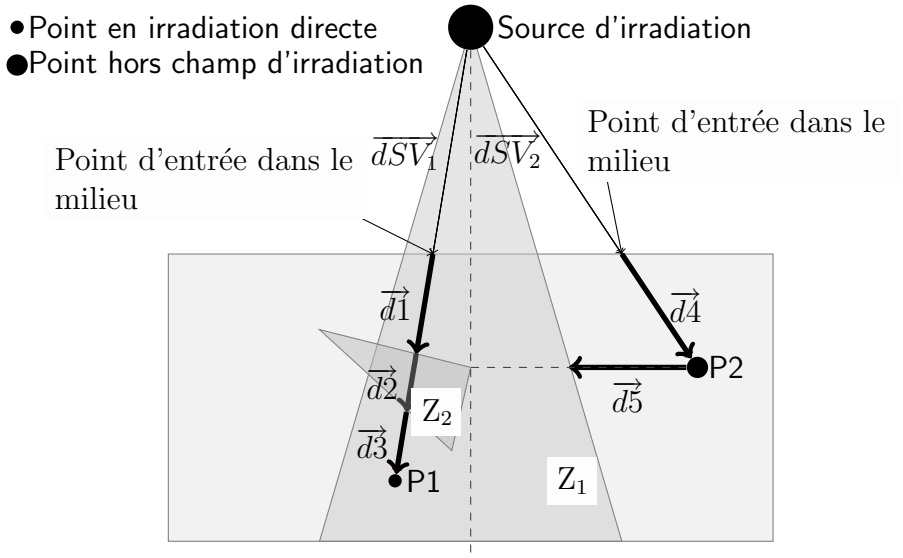

FIG. 4.3 – Construction des *chemins* d'irradiation

La spécificité de la méthode proposée vient de ce qu'elle est basée sur le mixage d'un réseau neuronal avec un algorithme de calcul mis au point en utilisant l'expérience et la validation, sans faire appel à des algorithmes coûteux reproduisant le comportement physique des éléments étudiés.

4.2.1 Détection du type d'irradiation principale

Lors de l'évaluation de la dose déposée dans un voxel par un faisceau irradiant, le fait que le voxel soit sous influence directe ou non du faisceau est un critère fondamental pour le calcul de celle-ci ; c'est pourquoi nous avons mis en place deux algorithmes spécifiques qui peuvent prendre en charge chacune de ces situations. Pour déterminer si un point appartient au cône d'irradiation, il faut vérifier que l'angle formé entre le vecteur cible-source et le vecteur directeur de la source est inscrit ou non dans l'angle définissant la divergence du faisceau irradiant.

Point dans le champ d'irradiation

Lorsqu'un voxel appartient à la zone directe d'irradiation, le processus d'évaluation suit le cheminement détaillé dans l'algorithme 4.1. La première étape de cet algorithme est de calculer les coordonnées du point en cours d'évaluation, en fonction du repère lié à la position de la source d'irradiation. Le fait d'utiliser les coordonnées liées à la source et non pas liées au milieu permet d'utiliser directement le réseau de neurones quelle que soit la position de la source par rapport au milieu. En effet, de cette manière, le point étudié se retrouve toujours dans des conditions similaires à celles qui ont été utilisées pour réaliser l'entraînement du réseau. Une fois le point positionné, il faut calculer le vecteur entre la source et le point en cours d'évaluation ($VecDir$). Ce vecteur est ensuite pris pour

calculer le chemin parcouru par le faisceau irradiant pour atteindre la cible. Le chemin (*che*) est une structure sous forme de liste répertoriant l'ensemble des milieux traversés, elle est décrite dans la figure 4.4.

```
class Position {
   Point posA;   // Position du point repere absolu.
   Point posS;   // Position du point repere source.
   double dens;  // Densite du milieu
}
class Chemin  {
   Position [ ] chemin;  // Liste des positions du chemin.
   double dist;          // Distance entre la source et milieu.
   bool dansRayon;       // Précise si l'entrée du chemin appartient
                         // au faisceau irradiant.
}
```

FIG. 4.4 – Structures Chemin et Position

Cette structure permet de détailler pour chaque milieu traversé, la position d'entrée (dans les deux repères en utilisant les champs *posA* et *posS*), la densité du milieu traversé (à l'aide du champ *dens*), ainsi que la distance parcouru dans le milieu (dans le champ *dist*). Le champ *dansRayon* permet d'enregistrer si le chemin appartient ou non au champ irradiant. Le chemin est construit à partir du point cible, il faut donc l'inverser avant de passer à la dernière étape du processus qui consiste à évaluer la dose dans la zone cible.

La première position du chemin, associée au vecteur cible-source, permet de déterminer le point d'entrée du faisceau dans le milieu. La position de ce point permet de calculer la distance générale entre le milieu étudié et la source, ce qui correspond au vecteur d1 sur la figure 4.7. Le processus itère ensuite sur chacune des différentes positions enregistrées dans la structure *che*. Toujours en se référant à la même figure, la structure *che* contient trois ensembles représentant les vecteurs $\overrightarrow{d1}, \overrightarrow{d2}$ et $\overrightarrow{d3}$. A chaque position, le processus évalue la valeur de dose avant le passage d'interface en utilisant le réseau de neurones. Puis, à l'aide de l'algorithme de passage d'interfaces longitudinales et de la densité du milieu suivant (contenu dans la structure chemin *che*), l'algorithme calcule la position équivalente à la valeur de dose suivant l'interface. Ce processus itère jusqu'à la dernière position du chemin qui correspond à la zone cible.

Point hors champ

L'algorithme mis en place pour l'évaluation d'une zone cible hors du champ d'irradiation est plus simple. Pour rappel, si le processus diffère de celui utilisé pour évaluer la valeur de dose d'un voxel dans le champ d'irradiation, c'est que la dose déposée dans un tel contexte ne provient pas directement de la source, mais de l'énergie diffusée par les autres zones du milieu. La valeur de dose dans ces zones hors champ reste souvent négligeable. Pouvoir la quantifier avec une relative précision est intéressant uniquement

Algorithme 4.1 Algorithme d'évaluation en irradiation directe

Entrées:
Vecteur VecDir // *Vecteur unitaire entre la source et la zone cible*
Réseau Res // *Réseau de neurones entraîné*
Réel [ ] Position // *Position de la zone cible*
Variables:
Chemin che // *Chemin entre la source et la zone cible*
Entier rang // *Position dans le chemin*
Résultats:
Réel valDose // *Valeur de dose dans la zone cible*

// *Construction du chemin entre la cible et la source jusqu'à l'entrée dans le milieu*
che ← calculChemin(vecDir, Position)
rang ← 0
// *Évaluation de l'évolution du dépôt de dose au cours du chemin*
PositionCourante ← dernierePositionChemin(che)
Tantque PositionCourante ≠ Position **Faire**
  rang ← rang + 1
  // *Recherche de la nouvelle position en utilisant l'algorithme 4.4*
  PositionCourante ← recherchePositionInterfaceLong(che, rang, dosePrec)
  // *Recherche de la distance parcourue dans la section de chemin*
  distance ← distanceRang(che, rang)
  PositionCourante ← avancePosition(PositionCourante, VecDir, distance)
  valDose ← activeReseau(Res, positionCourante)
Fin tantque

dans le cas d'un enchaînement d'irradiations. Puisque la dose est un élément cumulatif, une somme de quantités négligeables peut devenir une information substantielle à terme.

Le processus mis au point pour évaluer cette dose est décrit de manière précise dans l'algorithme 4.2. Comme pour le cas précédent, la première étape de ce processus est de calculer les coordonnées de la cible dans le repère lié à la source d'irradiation. Ensuite, le processus évalue la valeur de dose dans le milieu homogène correspondant, à l'aide de ses nouvelles coordonnées. Finalement, pour prendre en compte la diffusion de l'irradiation, nous pondérons cette valeur en utilisant la valeur de la première zone du champ d'irradiation rencontrée. Cette évaluation est schématisée dans la figure 4.7 et concerne l'évaluation du point $P2$. Le chemin pour atteindre ce point depuis la source d'irradiation est composé uniquement du vecteur $\overrightarrow{d4}$. Cette évaluation passe donc par le calcul à l'aide du réseau de neurones pour le point $P2$ dans un milieu homogène, qui pour l'exemple, est composé d'eau. Ensuite, on recherche la première valeur de dose appartenant au cône d'irradiation, en utilisant le vecteur $\overrightarrow{d5}$. Une fois cette valeur de dose évaluée, il suffit de l'utiliser pour pondérer la valeur en $P2$.

Algorithme 4.2 Algorithme d'évaluation de la dose d'une cible hors champ

Entrées:
Réel [ ] VecDir // *Vecteur entre la source et la zone cible*
Réseau Res // *Réseau de neurones entraîné*
Réel [ ] Position // *Position de la zone cible*
Variables:
Réel [ ] Vec // *Vecteur perpendiculaire dirigé vers le cône d'irradiation*
Réel [ ] positionCourante // *Position lors de la recherche du cône d'irradiation*
Booléen appartient // *Indique si le voxel étudié appartient au cône d'irradiation*
Résultats:
Réel valDose // *Valeur de dose dans la zone cible*

```
valDose ← activeReseau(Res, Position)
Vec ← calculVecteurOrtho(Position, VecDir)
// Pondération de la valeur de dose avec la première valeur de dose rencontrée du cône irradié
Tantque appartient =Faux Faire
  PositionCourante ← avancePosition(PositionCourante, Vec)
  appartient ← appartientConeIrradiation(Position)
Fin tantque
doseCourante ← activeReseau(Res, positionCourante)
dist ← distance(Position, PositionCourante)
valDose ← pondereDose(valDose, doseCourante, distance)
```

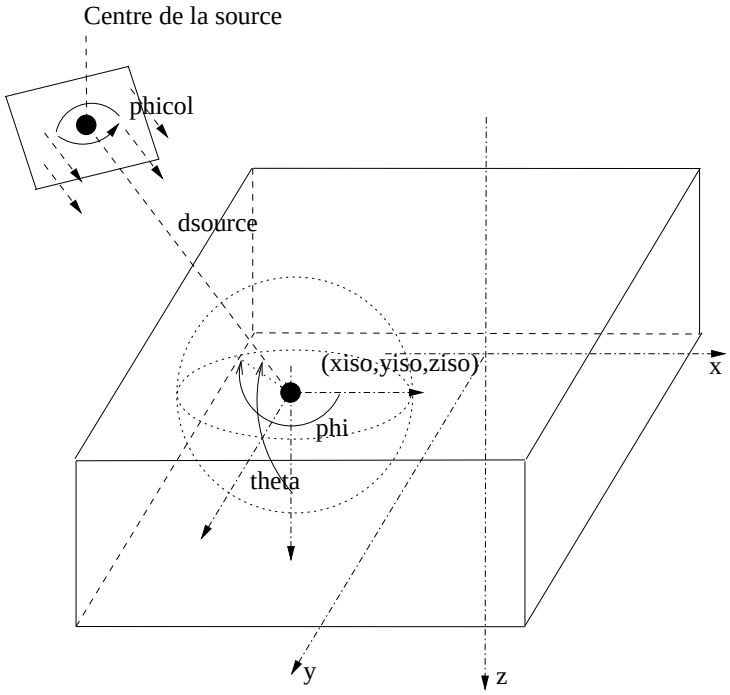

FIG. 4.5 – Positionnement de la source d'irradiation par rapport au milieu irradié

4.2.2 L'algorithme général d'évaluation des doses

Le premier rôle de cet algorithme est de replacer chacun des différents points composant le volume soumis à l'irradiation dans un contexte similaire à celui utilisé lors de l'apprentissage du réseau de neurones. Comme le montre la figure 4.5, lors de la réalisation d'une irradiation, la source peut se trouver de part et d'autre du volume irradié. L'intérêt est de pouvoir irradier un volume relativement fixe avec la plus grande liberté possible. Il est aisément compréhensible que le volume en question, à savoir un patient, a des possibilités de positionnement restreintes. Pour positionner la source avec précision, il est possible de jouer sur trois angles et sur une distance. Ces mesures sont prises en fonction d'un point de référence, le point Iso de coordonnées $(x_{iso}, y_{iso}, z_{iso})$ du volume cible à irradier, et d'un repère orthogonal (x, y, z) lié à ce point. Les angles ϕ et θ permettent de fixer la position de la source alors que la distance d_{source} donne la distance entre la zone cible et la tête de l'accélérateur. L'angle ϕ_{col} permet d'orienter la tête de l'accélérateur sur son axe d'irradiation.

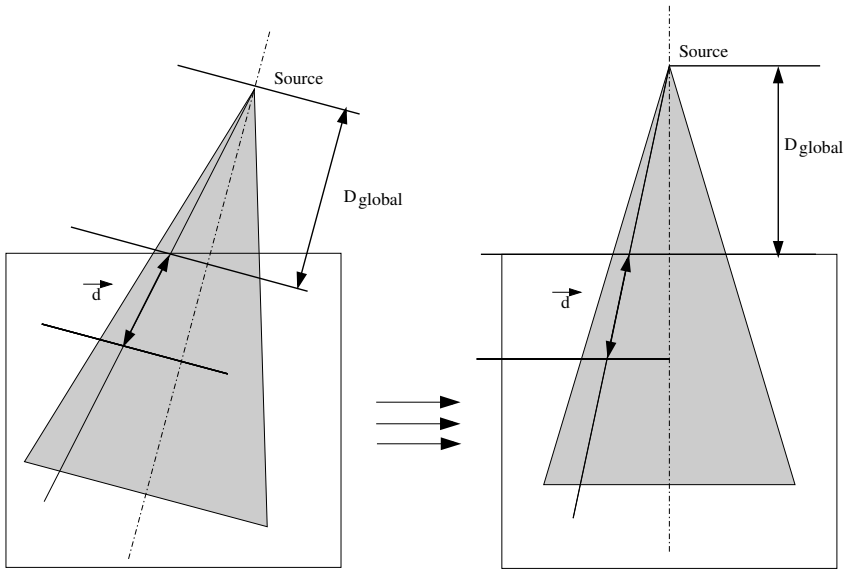

FIG. 4.6 – Calcul d'un point pour une incidence de rayon non verticale à partir d'une position quelconque

Afin de limiter le nombre de conditions à faire apprendre au réseau de neurones, nous avons décidé que tous les apprentissages se feraient dans des conditions identiques où la tête de l'accélérateur se trouverait à la verticale du volume irradié (contexte droit), c'est à dire pour une valeur de ϕ à 0°, θ à 180° et ϕ_{col} à 0° et où la cible se situerait au centre de la face supérieure du volume irradié ($x_{iso} = \frac{x_{range}}{2}, y_{iso} = \frac{y_{range}}{2}, z_{iso} = 0$). Le seul paramètre variable concernant la localisation de la source porte donc sur la distance entre la source et le milieu irradié.

La technique permettant de calculer la position virtuelle d'un point dans un contexte non droit, afin de pouvoir calculer sa dose à l'aide d'un réseau de neurones, est détaillée dans la figure 4.6. La méthode consiste à replacer le vecteur cible-source (d sur la figure) dans un contexte d'irradiation perpendiculaire au volume. Pour permettre ce replacement dans un contexte connu, deux informations sont importantes : le vecteur d, qui indique le chemin parcouru depuis la source par le rayon irradiant pour atteindre le point cible, ainsi que la distance D_{global}, indiquant, quant à elle, la distance entre la source et le milieu à irradier. À partir de ces deux informations, il est possible de replacer chacun des points indépendamment.

Une fois cette étape réalisée, et à partir des deux algorithmes précédents, l'algorithme général d'évaluation des doses a pour unique rôle de déterminer les conditions d'irradiation de la zone étudiée afin d'exécuter l'algorithme correspondant qui, lui, évalue la dose déposée.

Algorithme 4.3 Algorithme général d'évaluation des doses

Entrées:
Params params // *Structure contenant les paramètres de calcul*
Réseau Res // *Réseau de neurones entraîné*
Milieu milieu // *Structure décrivant le milieu (ensemble de voxels)*
Source source // *Structure décrivant la source d'irradiation*
Variables:
Matrice mat // *Matrice de changement de repère liée à la source*
Booléen appartient // *Indique si le voxel étudié appartient au cône d'irradiation*
Résultats:
Voxcube doseDepose // *Structure permettant d'enregistrer les doses déposées*

Pour tout voxel du milieu **Faire**
// *Évaluation de la position du voxel dans le repère source*
Position ← positionnement(voxel, mat)
appartient ← appartientConeIrradiation(Position)
Si appartient = Vrai **Alors**
doseDeposee$_{voxel}$ ← doseIrradtiationDirect(Res, Position)
Sinon
doseDeposee$_{voxel}$ ← doseHorsChamp(Res, Position)
Finsi
Fin pour

4.2.3 Gestion des interfaces perpendiculaires

L'interface perpendiculaire est l'interface la plus influente lors du parcours d'un milieu par un faisceau irradiant. Comme il est précisé dans la section 4.1, ce type d'interface apparaît lorsque l'environnement étudié comporte une suite de matériaux perpendiculaires au faisceau irradiant. Pour résoudre ce problème, il faut trouver la profondeur équivalente dans le milieu homogène correspondant à la valeur de dose précédent l'interface dans le milieu. Cette recherche de profondeur équivalente revient à évaluer le décalage décrit dans la figure 4.1(gauche). L'hypothèse mise en place lors de la réalisation de cet algorithme est que seules les interfaces longitudinales situées après le point maximum de la courbe de rendement sont prises en charge. En utilisant cette hypothèse, il est garanti que les courbes de rendement en profondeur sont exclusivement en phase décroissante, ce qui facilite la recherche de la position équivalente en la limitant à une simple montée ou descente de courbe.

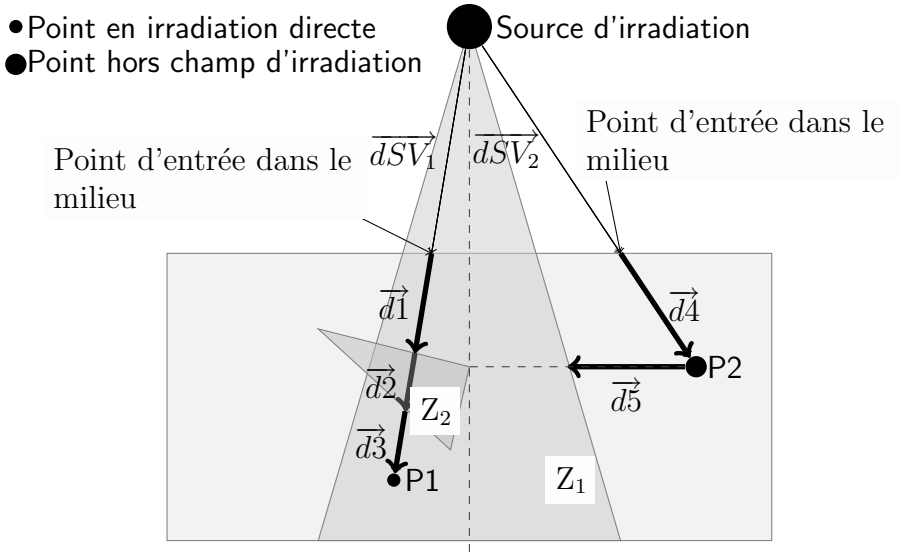

FIG. 4.7 – Milieu en cours d'irradiation

Deux cas peuvent être rencontrés en fonction de la densité des deux milieux entourant l'interface : soit le faisceau irradiant passe d'un milieu plus dense à un milieu moins dense, soit dans le cas contraire, il passe d'un milieu moins dense à un milieu plus dense. Dans les deux cas, le traitement est similaire, il suffit de parcourir le vecteur entre la source d'irradiation et la position étudiée pour trouver la profondeur correspondant à la valeur de dose précédente dans le nouveau milieu homogène virtuel. La figure 4.7 présente le détail de l'évaluation de deux zones (Z_1 et Z_2). Pour l'évaluation de la zone Z_1, nous utilisons le chemin construit à l'aide des vecteurs $\overrightarrow{d1}$, $\overrightarrow{d2}$ et $\overrightarrow{d3}$. Avant d'arriver en $P1$, le faisceau irradiant doit traverser deux interfaces perpendiculaires, apparaissant de part et d'autre de la pièce de titane qui est plongée dans le volume d'eau. Pour calculer, par exemple, la valeur de dose se trouvant directement après la première interface, il faut

connaître la valeur de dose précédent l'interface, c'est à dire à la fin du vecteur $\overrightarrow{d1}$, pour ensuite, retrouver ce niveau de dose dans un milieu composée exclusivement de titane. Cette nouvelle valeur sera la première valeur du vecteur $\overrightarrow{d2}$. La différence de densité permet d'indiquer le sens de parcours du vecteur dans le nouveau milieu afin de retrouver la position recherchée. Une fois cette nouvelle position trouvée, il est possible d'utiliser le réseau de neurones pour obtenir les valeurs de dose déposées sur la suite des positions du milieu, dans notre cas, le long du vecteur $\overrightarrow{d2}$, et ce, jusqu'à la prochaine interface rencontrée. Ces procédés sont décrits de manière détaillée dans l'algorithme 4.4.

4.2.4 Gestion des interfaces latérales

Comme il a été mentionné dans la section 4.1, le phénomène présent aux interfaces latérales peut être modélisé en première approche comme un simple lissage entre les milieux. Nous avons donc mis en place dans un premier temps un mécanisme simple de lissage entre les différentes valeurs de doses. Pour ne pas perturber le mécanisme global d'évaluation de dose, cet algorithme devait être appliqué seulement après l'évaluation complète des doses dans le milieu.

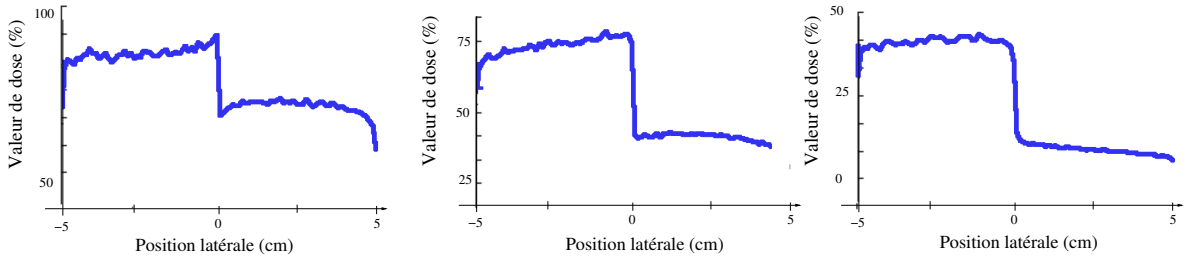

FIG. 4.8 – Forme de la brusque variation du dépôt de dose présente à l'interface latéral à 1 cm , 5 cm puis 10 cm de profondeur (de gauche à droite)

Toutefois, il apparaît que l'apport d'erreur dû aux artefacts suite aux différences de densités est plus important que dans le cas longitudinal. De plus, les effets du lissage ont un effet négatif sur l'erreur globale de la courbe d'irradiation. Dans un second temps, nous avons mis en place un mécanisme faisant appel à un réseau de neurones spécialement entraîné pour la gestion de ce type d'interface. Le principe de ce mécanisme a été testé sur un milieu en deux dimensions, et laisse présager d'un bon comportement général. Mais, suite au passage à l'évaluation des doses directement en milieux tridimensionnels, le temps nous a manqué pour explorer plus précisément cette piste.

Algorithme 4.4 Algorithme de gestion des interfaces perpendiculaires

```
Entrées:
  Réseau Res        // Réseau de neurones entraîné
  Réel [ ] Position // Position de l'interface
  Réel valDoseR     // Valeur de la dose précédent l'interface
  Réel [ ] vecDir   // Vecteur unitaire entre la source et le point en cours d'évaluation
  Réel densPrec, densSuiv // Densité précédent et suivant l'interface
Variables:
  Réel [ ] posPrec          // Position intermédiaire
  Réel tolDens, tolDose     // Tolérance de densité et de valeur de dose
  Réel valSortie, valPrec   // Valeurs résultats (courante et précédente)
                            // du réseau de neurones
  Réel coefRech             // Coefficient de recherche d'une position
  Booléen active            // Validité de l'activation
Résultats:
  Réel [ ] Position // Position modifiée prenant en compte l'interface

  Si active = Faux Alors
    Position ← positionBorneDomaine(Res, Position)
  Sinon
    posPrec ← Position
    Si densSuiv > densPrec Alors
      Tantque coefRech ≥ tolDose et valPrec ≤ valSortie et active = Vrai Faire
        posPrec ← Position
        valPrec ← valSortie
        Position ← delacementPostion(coefRech, -vecDir, Position)
        active ← activeReseau(Res, Position, valSortie)
        Si valSortie > valDoseR et active = Vrai Alors
          Position ← posPrec
          coefRec ← coefRec/2
        Finsi
      Fin tantque
    Sinon
      Tantque coefRech ≥ tolDose Faire
        posPrec ← Position
        valPrec ← valSortie
        Position ← delacementPostion(coefRech, vecDir, Position)
        active ← activeReseau(Res, Position, valSortie)
        Si valSortie ≤ valDoseR et active = Vrai Alors
          Position ← posPrec
          coefRec ← coefRec / 2
        Finsi
      Fin tantque
    Finsi
  Finsi
```

Le principe est de mettre au point un réseau de neurones sachant décrire l'artefact présent à un changement de milieu en se basant sur le rapport des deux densités encadrant l'interface entre les milieux. L'artefact présent de part et d'autre d'une interface dépend principalement de deux paramètres, la valeur de dose au niveau de l'interface et le rapport entre les densités des deux matières la composant. De plus, la forme de cet artefact n'est pas constante sur l'ensemble du rendement en profondeur, tel qu'on peut le constater sur la figure 4.8. L'idée est donc de réaliser une série d'apprentissages sur des milieux composés d'une hétérogénéité centrale, correspondant à une interface latérale. Cet ensemble est composé, pour une énergie de faisceau donnée, d'une série de matières différentes dans l'objectif de recouvrir la plus grande plage de rapports de densités possible. Ensuite, il ne reste plus qu'à construire un réseau de neurones ayant pour domaine d'apprentissage la zone d'influence de l'interface, c'est-à-dire le centre du milieu irradié.

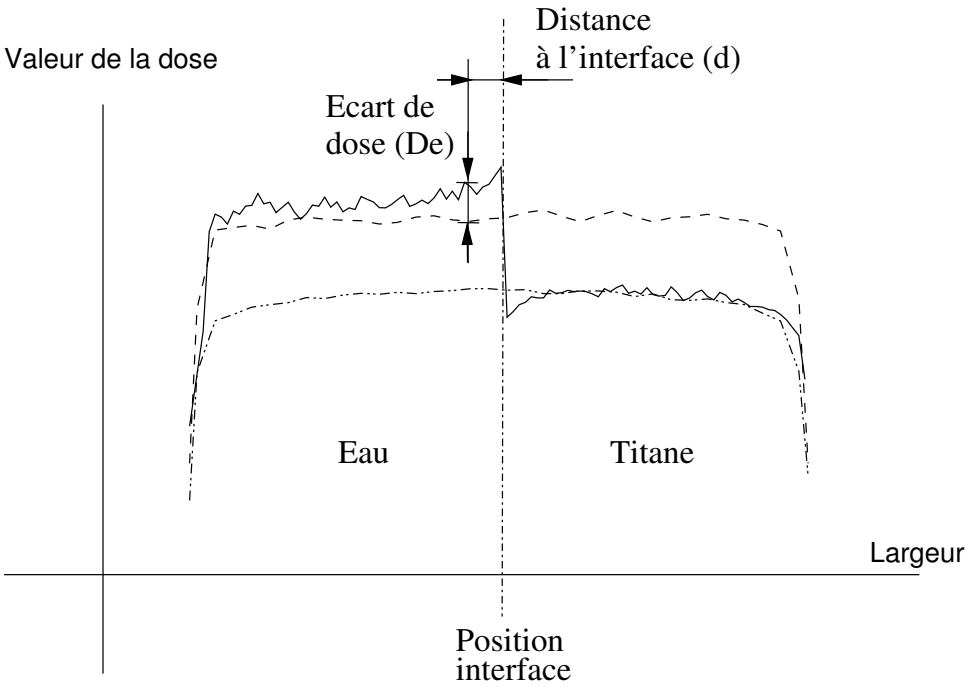

FIG. 4.9 – Rapport entre les valeurs de doses soumises ou non à une interface latérale

Comme pour notre première idée, l'application de cet algorithme est effectuée en post-traitement. La première étape consiste à repérer, sur l'ensemble du milieu irradié, les différentes zones sous l'influence d'une interface latérale. Ensuite, pour chaque zone, il faut calculer le rapport des densités composant l'interface. En utilisant le rapport de doses, et le niveau de dose de la zone étudiée, il est possible de retrouver le rapport de dose entre la dose du milieu homogène et celle présente dans le milieu sous l'influence de l'interface. La figure 4.9 présente un exemple de profils regroupant trois courbes de distribution de doses. Chacune de ces différentes courbes a été évaluée dans des milieux différents mais avec une énergie de faisceau identique. Il y a deux courbes issues de milieux homogènes (eau puis titane) et une courbe issue d'un milieu possédant une interface latérale entre deux matériaux (eau et titane). On peut clairement voir sur cette figure l'influence de l'interface sur le niveau des doses, comme le quantifie la mesure De. L'intérêt de notre algorithme est donc de retrouver cette mesure De correspondant au milieu en cours d'évaluation.

Cette recherche se fait en plusieurs étapes :

- Calcul du quotient entre les densités composant l'interface ;
- Recherche du réseau de neurones correspondant à ce rapport ;
- Recherche de la position correspondant à la position courante dans le milieu de référence (utilisation de la distance à l'interface d) ;
- Calcul du rapport correspondant à l'influence de l'interface ;
- Application de ce rapport à la zone étudiée.

Le temps nous ayant manqué pour la mise en œuvre de cet algorithme, il n'est pas présent dans la version tridimensionnelle de notre solution logicielle Neurad [5], et n'est donc pas encore évalué. Cependant, il devrait être mis en place lors de travaux ultérieurs.

Les différents algorithmes présentés dans cette section ont pour rôle de reconstruire des courbes de dépôt de doses suite à une irradiation. Comme indiqué précédemment, l'élément de base de ces algorithmes repose sur l'utilisation des réseaux de neurones. De ce fait, la précision globale de la courbe de rendement repose sur la combinaison de ces modules. L'enjeu est donc d'avoir une précision maximale de chacun des éléments afin que leur assemblage ne construise pas une courbe présentant des incertitudes supérieurs au standard imposé dans le monde médical. Le chapitre suivant présente les différents résultats obtenus au cours de ces travaux.