Transformations et recalages

Sommaire

7.1 Introduction

La présentation de la méthode d'identification utilisée pour ce travail ne peut se faire sans présenter au préalable la technique de **Recalage d'Images Déformables (RID)**. Celle-ci a de nombreuses applications dans le domaine biomédical [Sar00], car elle est particulièrement adaptée aux mesures *in vivo* des déformations. Ce chapitre vise à présenter deux notions fondamentales nécessaire à cette technique : le **Recalage d'Images Numériques (RIN)** et la **Déformation d'Images Numériques (DIN)**. Le terme de recalage est un terme dont la signification varie selon les auteurs, il peut désigner la détermination de transformations permettant la correspondance entre deux images mais aussi l'application de ces transformations à une des images.

Le RIN est le procédé qui permet de déterminer les transformations existantes entre deux images c'est-à-dire la détermination des fonctions de mise en correspondance entre les deux images [Kyb01]. Plusieurs applications du RIN sont possibles ; les transformations déterminées peuvent être appliquées directement à l'image comme dans le cas d'alignement d'images dans un repère comme réalisé dans le Chapitre 5 pour le recalage des IRM mais il peut aussi être utilisé dans le simple but de déterminer les transformations et ainsi en extraire des données sur les déformations. C'est ce deuxième aspect qui nous intéresse dans la suite de ce chapitre.

Un des objectifs du travail présenté est de mettre au point un modèle numérique de jambe pouvant être utilisé notamment pour la simulation de la contention. Cependant pour la construction de ce modèle, les lois de comportement des tissus de la jambe doivent être identifiées. Pour cette identification nous avons choisi de comparer des images IRM du mollet en coupe tranverse à l'état déformé (sous l'effet de la contention) et non-déformé. Une possibilité de comparaison a été évoquée dans un chapitre précédent [Chapitre 5], en utilisant la Corrélation d'Images Numériques. Or après plusieurs essais, nous avons conclu que cette technique n'était pas adaptée au problème posé car d'une part elle dépend fortement du choix de la taille des fenêtres et d'autre part le contraste local des IRM n'est pas suffisant d'où l'idée de faire en correspondance globale avec la RID. Nous présenterons dans ce chapitre la méthode de **RID** en vue de l'identification.

7.2 Transformations d'image

Le Recalage d'Images Numériques (RIN) permet de mettre en correspondance deux images *i.e.* déterminer les fonctions de correspondance ou transformations liant les images. Dans ce chapitre nous restreignons la présentation aux transformations géométriques.

Les transformations géométriques sont réparties en deux classes : les représentations globales et les représentations locales. Si ce sont les transformations locales qui vont

nous être utiles, on peut néanmoins citer quelques unes des tranformations globales comme les similitudes, les transformations affines, les transformations polynômiales, *etc* ...[Sar00]. Au vu des images qui sont à traiter dans notre cas (IRM du mollet à différents états de déformation), les transformations globales ne sont pas adaptées, car les images déformées ne le sont pas de manière homogène, ce qui justifie alors l'utilisation de transformées locales.

La transformation locale la plus simple est la **transformation linéaire par morceaux**. Les tranformations locales se basent sur une triangulation du domaine d'étude [ZF03]. Une transformation linéaire par morceaux indique qu'il existe pour chaque triangle défini du domaine une transformation linéaire (ou affine) le reliant au même triangle dans la seconde image. Le principe des transformations est local mais soulignons que sa mise en œuvre nécessite la continuité de la transformation, ce qui lui confère une définition globale.

L'application de ces transformations à une image est appelée la **déformation d'images numériques**, assimilée au procédé inverse du recalage. Le recalage utilise deux images pour fournir une transformation alors que la déformation d'images utilise une image et une transformation pour fournir une image déformée. Autrement dit, si on applique la transformation déterminée par le recalage à l'image initiale, elle doit être identique à la seconde image qui a été nécessaire pour le recalage. Cette dernière remarque, malgré son évidence, est à la base de la méthode implémentée par nos soins sous Matlab®, que nous exposerons un peu plus tard. À titre d'exemple, une application de cette méthode pour un problème mécanique et plus précisément biomécanique est présenté dans [VPW05].

7.3 Interpolation

Pour chaque méthode de recalage, il est nécessaire d'en définir la méthode d'interpolation car l'une des principales difficultés rencontrées lors des transformations est la ré-assignation des niveaux de gris [Ben01]. En effet, la position d'un pixel transformé n'est pas entière. Il faut donc trouver une méthode permettant d'assigner un niveau de gris à un pixel se retrouvant à une position non-entière [Figure 7.1]. Ces méthodes sont appelées les méthodes d'interpolation dont les plus fréquemment utilisés sont la méthode des plus proches voisins et les interpolations polynômiales. On trouve aussi des méthodes de splines [Dod97]. Nous nous intéressons ici à l'interpolation par les plus proches voisins et à l'interpolation bilinéaire.

7.3.1 Interpolation par les plus proches voisins

Pour recaler une image, on reconstruit une image pixel après pixel à partir de l'image à recaler. Pour reconstruire l'intensité en un pixel donné de la nouvelle image, de position géométrique M, on considère le point matériel M_0 du solide qui est imagé en

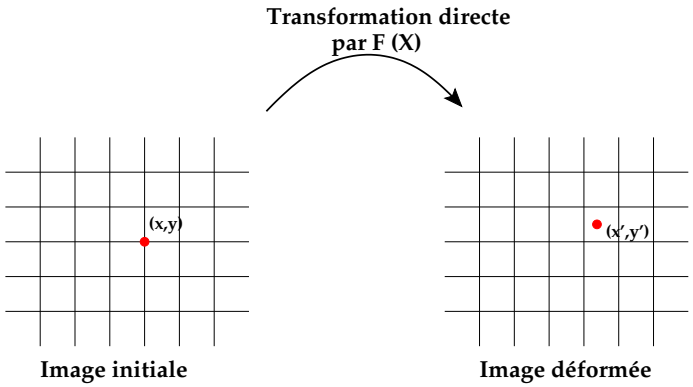


Figure 7.1 – Illustration du positionnement d'un pixel déformé

ce pixel donné. On cherche la position géométrique du même point matériel M_0 dans l'image à recaler (le point matériel n'a pas la même position car le solide imagé s'est déformé). Soit M' cette nouvelle position géométrique. Le principe du recalage est d'assigner au pixel situé en M dans l'image recalée à reconstruire la valeur du pixel le plus proche de la position géométrique M' dans l'image à recaler. Ainsi, pour chaque position M de l'image recalée à reconstruire, il faut connaître $M' = f(M)$ dans l'image à recaler, où f est la transformation géométrique. Dans notre cas, la transformation géométrique est connue ou calculée dans la lagrangienne, c'est-à-dire à l'état initial. Cela veut dire que la position géométrique M doit correspondre à la position initiale (avant déformation) du point matériel M_0 du solide imagé et la position géométrique M' doit correspondre à la position finale (après déformation) du point matériel M_0 du solide imagé. Finalement, le principe du recalage consiste donc à appliquer une transformation géométrique pour ramener une image déformée dans son état non déformé. Il faut ainsi retenir que le recalage se fait dans ce sens (état déformé vers état initial) et pas dans le sens contraire [ZF03].

7.3.2 Interpolation bilinéaire

L'interpolation bilinéaire utilise la valeur des pixels autour du pixel déformé. Soit (x, y) les coordonnées du point à interpoler, on cherche à déterminer $I(x, y)$ avec $I(x, y) = ax + by + cxy + d$ [Cas79]. En s'appuyant sur la [Figure 7.2], on peut exprimer $I(x, y)$ en fonction des autres données [Annexe B].

Ce type d'interpolation permet le *lissage* des contours là où l'interpolation par les plus proches voisins aura tendance à accentuer le phénomène de pixellisation. Une comparaison imagée des différents types d'interpolation est faite dans [ZF03].

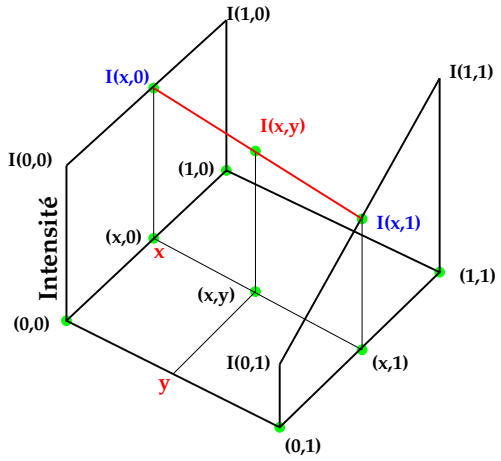


Figure 7.2 – Réprésentation graphique de l'interpolation bilinéaire (d'après [Ben01])

7.4 Utilisation du recalage d'images déformables dans un processus d'identification

7.4.1 Principe de la méthode

Le principe de l'identification est de trouver le jeu de paramètres hyperélastiques qui fera le mieux correspondre les données expérimentales aux données simulées. Il est donc nécessaire de définir une fonction qui permet de déterminer la qualité des paramètres testés. Cette fonction est appelée **fonction coût**, qui exprime l'écart entre les deux jeux de données. Avant de présenter la *fonction coût*, il faut présenter les données qui sont à comparer de manière plus précise.

Identifier les paramètres des lois de comportement consiste à trouver le jeu de paramètres qui permettra par le calcul EF de déterminer la déformation correspondant au mieux à celle existante entre les deux images du mollet respectivement comprimé et au repos. Pour comparer les déformations subies par la jambe, plusieurs possibilités s'offrent à nous.

Nous avons vu précédemment [Chapitre 5] qu'il est possible de comparer des déplacements nodaux avec des déplacements obtenus par CIN. Or, la conclusion était que l'approche trop locale de la CIN combinée au manque de contraste, pouvait engendrer des instabilités responsables du manque de robustesse de la méthode. Le problème de la CIN est que les données expérimentales font déjà l'objet d'approximations voir d'erreurs qu'on ne peut pour l'instant pas quantifier (variations de résultats en fonctions des différentes tailles de fenêtres et de zones de recherche). C'est pourquoi nous nous sommes orientés vers la méthode de **Recalage d'Images Déformables**, qui allie le recalage d'images numériques et la déformation d'images.

7.4.2 La Déformation d'Images Numériques

Il a été évoqué la possibilité, connaissant des transformations locales, de les appliquer à une image afin de la déformer numériquement. Le principe que l'on souhaite utiliser dans le cadre de l'identification, est de déterminer les tranformations numériques liées à la simulation EF et d'appliquer l'inverse de ces transformations à l'image IRM sous la contention correspondant à la simulation. Ainsi, on compare deux images à l'état non-déformé. L'une est issue de l'acquisition IRM directement et la seconde est une image IRM ayant subit une transformation inverse. Pour plus de clarté, on présente schématiquement les deux étapes précédant l'identification à proprement dite. La première étape consiste à déterminer l'ensemble des transformations F(X) pour le jeu de paramètres testé [Figure 7.3]. La seconde consiste à appliquer la transformation in-

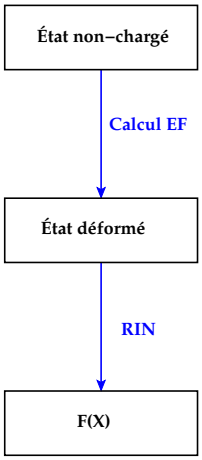


Figure 7.3 – Détermination des transformations F(X)

verse $F^{-1}(X)$ à l'image IRM de l'état déformé [Figure 7.4]. La comparaison se fera donc entre l'état non-déformé expérimental (ou initial) et l'état non-déformé numérique (ou estimé).

Dans la pratique, le recalage (détermination des transformations) se fait non pas sur l'image directement mais sur des points géomériques. L'avantage de l'utilisation de simulations EF est la présence des nœuds du maillage. Nous avons vu précédemment que l'utilisation de fonctions définies par morceaux était indispensable et pour cela la triangulation du domaine est nécessaire. La définition des nœuds comme points géométriques de référence assure une triangulation cohérente avec le calcul. Pour déterminer les transformations, les coordonnées des nœuds du maillage initial sont comparées aux nœuds du maillage déformé par la fonction *cp2tform* de Matlab®. Ensuite intervient la déformation d'images numériques. Une fois les transformations déterminées, leurs inverses sont appliquées à l'image IRM déformée expérimentale *via* la fonction *imtransform* de Matlab® pour laquelle l'interpolation bilinéaire est choisie. L'image résultante est une image *non-déformée estimée (ou calculée)*, elle est à comparer à l'image IRM sans contention.

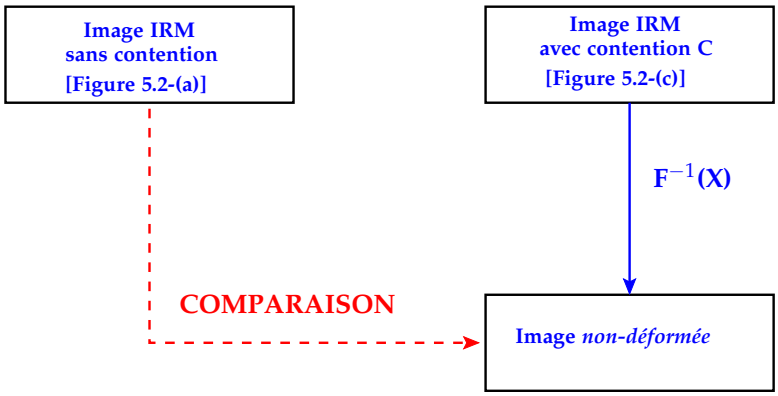


Figure 7.4 – Application de la transformation inverse $F^{-1}(X)$ et comparaison des images

Ce sont les paramètres matériaux qui déterminent les transformations. En effet, si ceux testés montrent un écart minimal entre l'image expérimentale initiale et l'image non-déformée calculée alors les paramètres testés sont les paramètres hyperélastiques réels car les transformations concordent. L'écart entre ces deux images est défini par une fonction coût, permettant d'évaluer la qualité des paramètres identifiés.

7.4.3 Présentation de la fonction coût utilisée

À présent que les entités à comparer ont été définies, il reste à présenter en quoi consiste cette comparaison. Deux possibilités ont été explorées lors d'essais préliminaires. Il s'agit ici de définir une **Fonction Coût** dont l'utilité sera présentée plus en détails par la suite. Cette fonction doit représenter l'écart entre les deux images. La minimisation de cette fonction permettra alors de déterminer les paramètres mécaniques des tissus à identifier.

La manière la plus simple de définir la fonction coût f_c est d'exprimer une différence au sens des moindres carrés. Soit la fonction coût ainsi définie,

$$f_c = \frac{1}{NP}\sum_{i=1}^{NP}(I_i^{exp} - I_i^{sim})^2 \tag{7.1}$$

avec NP le nombre de pixels et I_i le niveau de gris du pixel i pour chacune des images. On présente un exemple d'image de l'écart quadratique entre les images IRM expérimentales à différents états de déformation [Figure 7.5]. Sur cette figure, l'échelle des couleurs réprésente l'écart quadratique entre les deux images, au sens des niveaux de gris et en chaque pixel de l'image. Ce type d'image permet de se rendre compte de la localisation géométrique des zones d'écart important. Cependant, nous avons vu dans le Chapitre 5 que le modèle de simulation utilisé ne comportait pour le moment que deux matériaux distincts correspondant à la graisse et aux muscles. Il est donc probable qu'une fonction coût se basant uniquement sur leurs contours soit suffisante. De plus

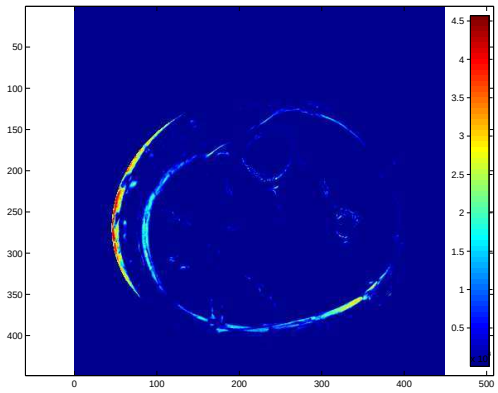


Figure 7.5 – Écart quadratique entre deux images

l'intensité entre les deux images à comparer varie. La fonction coût portant sur l'ensemble des images est donc mise en défaut. Or les contours sont stables vis-à-vis de cet artéfact.

Nous proposons donc une fonction coût qui représente la distance moyenne entre les contours considérés. Plus précisément, pour chaque pixel appartenant au contour, défini par la segmentation, non-déformé expérimental, on recherche le pixel du contour non-déformé simulé le plus proche. L'écart ainsi défini l'est au sens d'une distance physique. On réalise ensuite la moyenne sur l'ensemble des distances minimales de chacun des pixels du contour initial.

7.5 Conclusion

Dans ce chapitre nous avons abordé les notions et méthodes relatives au recalage d'images déformables ou recalage élastique d'images, qui est la méthode envisagée dans le processus d'identification des paramètres matériaux de la jambe. En effet le recalage d'images déformables conjugue le recalage d'images numériques *i.e.* la détermination des transformations (géométriques dans notre cas) entre deux images et la déformation élastique d'images. L'identification se doit de comparer des données de type expérimentales et des données issues de simulations EF. Dans le cas présent les données expérimentales sont représentées sous la forme d'image IRM sans contention *i.e.* à l'état non-déformé et les données simulées sont introduites par le recalage d'images effectué entre les nœuds du maillage EF initial et ceux du maillage déformé (par la simulation). Les transformations ainsi déterminées sont donc fonction des paramètres du comportement et leurs inverses sont appliquées à une image déformée expérimentale (sous une contention). Ce procédé revient à comparer des transformations, il est donc

possible par une optimisation appropriée de déterminer les transformations simulées les plus proches de celles ayant lieu entre les deux images expérimentales. La *fonction coût* définie compare les images expérimentales et *simulées* dans leur état non-déformé. Il est possible de comparer des images dans l'état déformé mais il est toujours préférable d'utiliser les transformées inverses pour déformer une image comme précisé dans [ZF03]. Plusieurs *fonctions coût* ont été envisagées mais au vu du peu d'informations contenues dans les images, plus précisément à l'intérieur du muscle, et du nombre de matériaux contenus dans le modèle EF, nous avons choisi d'utiliser une fonction coût permettant de ne représenter que l'écart entre les contours des deux matériaux présents dans le modèle EF.

L'identification des paramètres matériaux est une optimisation de la *fonction coût i.e.* la recherche d'un minimum pour cette fonction. Il reste donc à implémenter ou simplement utiliser des algorithmes d'optimisation afin de minimiser celle-ci de manière efficace (rapide, précise et robuste). On vérifiera par la suite la validité de la méthode proposée ainsi que l'efficacité de l'optimisation sur un cas-test pour finalement l'appliquer au modèle *biofidèle*.

Chapitre 8

Présentation de la méthode d'identification

Sommaire

8.1 Introduction

Grâce au recalage d'images déformables, nous sommes à présent en mesure de déformer une image numériquement et de la comparer à une image de type "expérimentale". Cette comparaison est faite par la définition d'une *fonction coût* qui est une fonction décrivant l'écart entre les 2 images. L'identification étant la détermination du jeu de paramètres des lois de comportement permettant de faire correspondre ces deux images, elle est le résultat d'une minimisation de la fonction coût.

Dans ce chapitre, nous nous intéressons à deux algorithmes d'optimisation qui sont l'algorithme de **Newton-Raphson** et l'algorithme de **Nelder-Mead**. Après avoir décrit le principe de ces algorithmes, nous allons les appliquer à un cas-test afin d'évaluer les difficultés qui pourront être rencontrées lors de l'identification des propriétés mécaniques des tissus biologiques mous.

8.2 Algorithmes d'optimisation envisagés

Le but de l'identification étant de trouver un jeu de paramètres minimisant la fonction coût définie précédemment, l'utilisation d'algorithmes d'optimisation est nécessaire. Nous avons choisi de ne présenter que deux algorithmes de différents types. Une méthode d'ordre 1, soit l'algorithme de **Newton-Raphson** [Jol99] et une méthode d'ordre 0, soit l'algorithme du **Simplexe** ou algorithme de **Nelder-Mead** [NM65],[Vay04].

8.2.1 Algorithme de Newton-Raphson

L'algorithme de Newton-Raphson est un algorithme de Newton multidimensionnel [Jol99]. C'est une méthode d'ordre 1 *i.e.* utilisant les gradients de la fonction coût. Cette méthode est choisie pour sa simplicité d'implémentation dans Matlab® et sa robustesse. Soit $f_c(X)$ la fonction coût à minimiser et X le vecteur constitué des paramètres à identifier, on cherche à résoudre,

$$\frac{df_c(X)}{dX} = 0 \tag{8.1}$$

Par un développement de Taylor au premier ordre au voisinage de X^n on a,

$$\left.\frac{df_c(X)}{dX}\right|_{X^{n+1}} = \left.\frac{df_c(X)}{dX}\right|_{X^n} + \left.\frac{d^2f_c(X)}{dX^2}\right|_{X^n} (X^{n+1} - X^n) \tag{8.2}$$

On peut déterminer X^{n+1} tel que l'Équation (8.2) soit nulle,

$$X^{n+1} = X^n - H^{-1} \left.\frac{df_c(X)}{dX}\right|_{X^n} \tag{8.3}$$

où H est la matrice *hessienne*. On rappelle que $H_{ij}^n = \left.\frac{d^2 f_c(X_i)}{(dX_j^n)^2}\right|_{X^n}$. Les gradients de la fonction coût sont obtenus par différences finies centrée à l'ordre 1.
À chaque itération, on determine le nouveau jeu de paramètres grâce à celui de l'itération précédente. Dans certains cas, il peut être utile d'introduire un coefficient d'amortissement d tel que,

$$X^{n+1} = X^n - d * H^{-1} \left.\frac{df_c(X)}{dX}\right|_{X^n} \qquad \text{avec} \qquad d < 1 \tag{8.4}$$

Le coefficient d'amortissement permet la diminution de la distance entre deux jeux de paramètres successifs. On s'approchera de manière moins rapide de la solution mais cela permet d'éviter l'éloignement de la solution. On peut représenter graphiquement l'algorithme de Newton-Raphson à une seule dimension [Figure 8.1]. Comme il s'agit

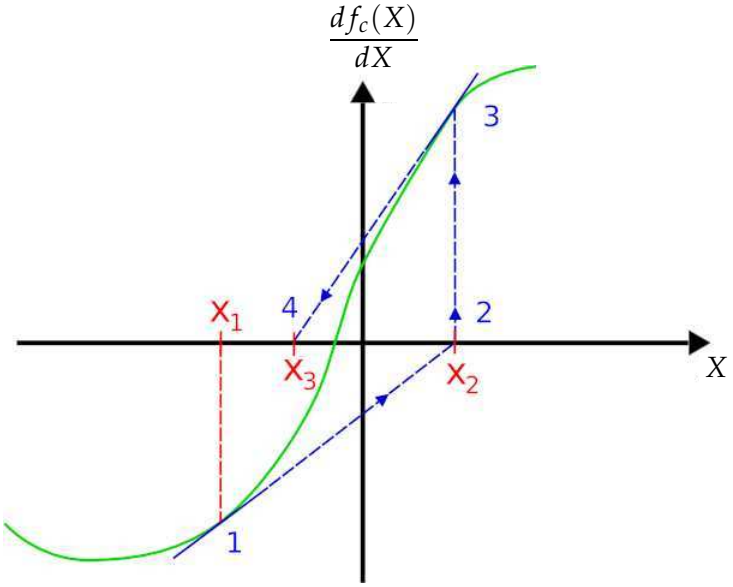


Figure 8.1 – Représention graphique de l'algorithme de Newton-Raphson

d'une procédure itérative, il est nécessaire de définir un ou plusieurs critères d'arrêt. Nous nous munissons de quatre critères.

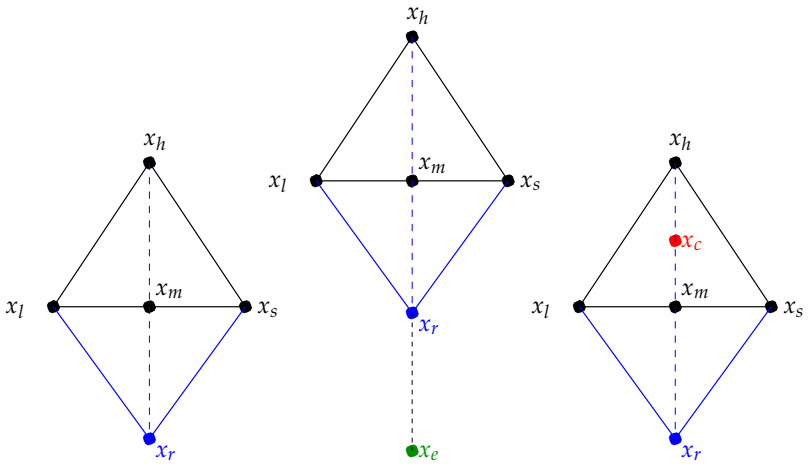
- **critère 1 :** valeur de la fonction coût en deçà de laquelle elle est considérée satisfaisante
- **critère 2 :** nombre d'itérations du calcul d'optimisation
- **critère 3 :** différence normalisée entre 2 valeurs de la fonction coût successives
- **critère 4 :** distance normalisée entre 2 jeux de paramètres testés successifs.

Dans le cas d'identification de nombreux paramètres, l'algorithme de Newton-Raphson est relativement rapide.

8.2.2 Algorithme de Nelder-Mead

La méthode de Nelder-Mead [NM65] permet une optimisation en s'affranchissant du calcul des gradients. Cette méthode est articulée autour de trois étapes qui sont la réflexion, la contraction et l'expansion. En dimension n, un simplexe est un polygone à $n+1$ sommets. Le principe est de déplacer le simplexe dans le domaine en remplaçant itérativement le plus mauvais point par un point meilleur.

On définit un simplexe à trois sommets (un triangle), on appelle x_h le sommet pour lequel la fonction coût est maximale $f_c(x_h) = f_h$, x_s le sommet où la fonction coût prend la deuxième plus grande valeur $f_c(x_s) = f_s$, x_l le sommet où la fonction coût est minimale $f_c(x_l) = f_l$. Le barycentre entre les points x_s et x_l est défini par x_m. La première étape de l'algorithme est la **réflexion**, on cherche à s'éloigner de x_s en créant x_r un nouveau sommet tel que $x_r = x_m + a(x_m - x_s)$ [Figure 8.2-*(a)*]. Soit $f_r = f_c(x_r)$, si



(a) Réflexion *(b)* Expansion *(c)* Contraction

Figure 8.2 – Les différents étapes de l'algorithme de Nelder-Mead en 2D

$f_l < f_r < f_s$, on remplace x_s par x_r. Si f_r est meilleur que le miminum actuel *i.e.* $f_r < f_l$, on essaye d'aller plus loin en créant x_e par l'**expansion** tel que $x_e = x_m + b(x_r - x_m)$ [Figure 8.2-*(b)*]. Mais si x_r n'est pas meilleur que x_s, on cherche à se rapprocher de x_l en effectuant une **contraction** *i.e.* en créant x_c tel que $x_c = x_m + c(x_h - x_m)$ [Figure 8.2-*(c)*]. Pour plus de détails concernant cet algorithme, nous invitons le lecteur à se référer à la [Figure 8.3]. D'une manière générale, les paramètres a, b et c sont 1, 2, 0,5 respectivement.

L'algorithme de Nelder-Mead est une méthode efficace et robuste mais elle est à proscrire dans le cas où de nombreux paramètres sont à déterminer (>10), on observerait alors une explosion du temps de convergence [Vay04]. On retrouve cet algorithme implémenté dans la fonction *fminsearch* de Matlab®.