
Ecole Doctorale 364 : Sciences Fondamentales et Appliquées

***N° attribué par la bibliothèque***

|__|__|__|__|__|__|__|__|__|__|__|

# T H E S E

Pour obtenir le grade de

## DOCTEUR DE L'ECOLE DES MINES DE PARIS

Spécialité "**Mécanique Numérique**"

Présentée par

**M. Tien Tho DO**

# OPTIMISATION DE FORME EN FORGEAGE 3D

***Directeur de thèse : Lionel FOURMENT***

Soutenue publiquement le 04 juillet 2006 devant le jury composé de :

| | |
|---|---|
| M. Alain POTIRON | Rapporteur |
| M. Piotr BREITKOPF | Rapporteur |
| M. Fabrice SCHMIDT | Examinateur |
| M. Ying Qiao GUO | Examinateur |
| M. Abderrahmane HABBAL | Examinateur |
| M. Richard DUCLOUX | Invité |
| M. Lionel FOURMENT | Directeur de thèse |

# TABLE DES MATIERES

# INTRODUCTION GENERALE

Le forgeage est actuellement un procédé de mise en forme très utilisé dans les secteurs mécaniques. Il s'effectue à chaud ou à froid par déformation plastique (grande déformation). Les pièces ainsi obtenues présentent une résistance supérieure à celles fabriquées avec d'autres procédés (usinage, fonderie…) mais pour un coût souvent plus élevé.

La mise au point d'une gamme de forgeage est difficile car les pièces doivent vérifier des propriétés de géométrie et de métallurgie très précises. En conséquence, plusieurs essais de forgeage en grandeur réelle sont souvent nécessaires afin d'obtenir la pièce finale souhaitée. Pour des raisons financières, le forgeron se doit alors de minimiser le nombre d'essais car la mise au point d'un tel essai est très coûteuse. Idéalement, le forgeron se doit alors de "réussir du premier coup". Ce n'est donc pas une tâche simple.

Depuis une vingtaine d'années, la simulation numérique des procédés de mise en forme constitue un moyen d'investigation privilégié pour éviter ces expériences coûteuses. Elle suscite de très grands intérêts. Elle permet au forgeron d'une part de prédire l'écoulement de matière durant le procédé et d'autre part d'avoir accès à tout un panel d'informations difficilement accessibles par l'expérience. Elle améliore la compréhension des phénomènes simulés et permet d'analyser un procédé, avant qu'il ne soit réellement mis en oeuvre. Les idées nouvelles peuvent ainsi être testées rapidement, sans mobiliser l'outil de production. De plus, le gain en coût et en rapidité par rapport à une approche exclusivement expérimentale est énorme. En conséquence, lorsque l'on veut définir un nouveau procédé de mise en forme, la simulation numérique est couramment utilisée comme une aide à la conception.

Toutefois, la complexité des pièces et des exigences de qualités augmente. Un nombre élevé d'essais numériques est nécessaire tandis que le temps pour effectuer une telle simulation est souvent très élevé. En 3D, il peut durer de quelques heures à quelques jours, selon la complexité de la pièce et selon les calculateurs utilisés.

Pour que le procédé de mise en forme satisfasse aux demandes des clients (produit final sans défaut majeur ou avec bonne qualité métallurgique, augmentation de la durée de vie des outils, diminution des coûts de fabrication etc.) en évitant les essais numérique coûteux en terme de temps de simulation, il faut l'optimiser. Cela constitue donc le sujet de ce travail de recherche : l'Optimisation de forme en forgeage 3D.

Depuis plusieurs années, de nombreuses études ont été menées sur l'optimisation de forme en forgeage [*Kusiak et al. 1989*], [*Chen et al. 1995*], [*Balan 1996*], [*Vieilledent 1999*], [*Castro el al. 2000*], [*Fourment et al. 2001*], [*Antonio al. 2002*], [*Chung et al. 2003*], etc. Toutefois, ces études sont principalement en 2D. Elles ne permettent pas d'aborder la majorité des cas réels là où les pièces produites sont de vraies formes 3D. Ce travail a pour but de palier ce défaut. Il fait suite à plusieurs thèses sur ce sujet qui se sont focalisée sur le calcul du gradient des fonctions coût et l'utilisation d'un algorithme local à base de gradient pour l'optimisation, essentiellement en 2D. Nous abordons ici véritablement l'optimisation en 3D, et recherchons des algorithmes globaux permettant d'éviter les extrema locaux, sans pour autant se traduire par un trop grand nombre de calculs.

L'objectif de ce travail de thèse est donc de résoudre les problèmes d'optimisation de forme en forgeage en 3D en utilisant des algorithmes robustes et efficaces (on se limite typiquement à 50 calculs de la fonction coût à chaque optimisation) qui peuvent localiser l'extremum global. Dans cette perspective, des sous objectifs sont définis tels que suit : l'étude des algorithmes d'optimisation, la paramétrisation de forme, la définition de fonctions coûts adéquates ainsi que le calcul de leur gradient. C'est ce que l'on propose de faire dans ce document.

Le premier chapitre de cette thèse présente un état de l'art des méthodes d'optimisation dans un cadre assez général.

**Le second chapitre présente la première de nos contributions, une nouvelle approche d'hybridation pour construire des algorithmes d'optimisation plus efficaces et plus robustes.** Cette approche est différente de celle plus traditionnelle où un algorithme évolutionnaire succède à un algorithme à direction de descente pour identifier l'optimum global du problème. L'idée de cette approche est la suivante. On laisse un algorithme évolutionnaire diriger le processus global de minimisation. Lors de chaque génération, un faible nombre d'individus particuliers sont judicieusement sélectionnés pour y calculer le coût exact (et son gradient). Ces calculs servent à fabriquer une approximation qui est utilisée pour évaluer l'ensemble des individus de la génération courante. Cette approche est très avantageuse car elle permet de mieux découvrir l'espace de recherche avec un coût de calcul très faible.

Après des rappels du problème mécanique modélisé dans FORGE3® et ceux sur la méthode de l'Etat Adjoint, le troisième chapitre expose **la seconde partie de nos contributions. Elle concerne une amélioration de la fonction coût pour mieux détecter le défaut de repli et une extension du calcul du gradient par la méthode de l'Etat Adjoint.** Dans le cadre du forgeage multi-passes, nous étudions en effet le paramètre "forme de l'outils de préforme". Cette extension permet de s'affranchir les limites du travail de Laroussi [*Laroussi 2003*] et de ses spécificités du calcul du gradient.

Les deux chapitres suivant, IV et V présentent l'application de ces algorithmes, de l'amélioration des fonctions coûts ainsi que du calcul de gradient à deux problèmes caractéristiques de forgeage, respectivement l'optimisation de la géométrie de la préforme et celle des outils de préforme.

# Chapitre I

# ETAT DE L'ART DES ALGORITHMES D'OPTIMISATION

## I.1. INTRODUCTION

L'optimisation de forme a fait l'objet de nombreux travaux [*Kusiak et al. 1989*], [*Balan 1996*], [*Vieilledent 1999*], [*Castro el al. 2000*], [*Antonio al. 2002*], etc. Grâce à ces études, la performance et l'efficacité des algorithmes d'optimisation ont beaucoup évoluées. Il est nécessaire de faire une présentation de ces algorithmes avant de présenter la nouvelle approche d'hybridation. Ce chapitre a cet objectif.

Un problème d'optimisation de forme peut généralement être présenté comme suit:

$$\begin{cases} Minimiser\, \Phi(\mu) \\ c_i(\mu) \le 0 \quad \forall i = 1,...,m_i \\ h_i(\mu) = 0 \quad \forall i = 1,...,m_e \\ \mu \in \Re^n \end{cases} \tag{1.1}$$

$\Phi(\mu)$ représente la fonction coût, qui peut être une combinaison de plusieurs critères ; $c_i(\mu)$ et $h_i(\mu)$ désignent les contraintes d'inégalité et d'égalité auxquelles est soumis le vecteur des paramètres à optimiser $\mu$.

Selon les circonstances (résultats expérimentaux, disponibilité du gradient de $\Phi(\mu)$, etc.) ainsi que les ressources disponibles (nombre possible d'évaluations, capacité des ordinateurs, etc.), le problème d'optimisation (1.1) pourra être résolu par différents algorithmes. Citons certains d'entre eux comme l'algorithme de gradient conjugué, les algorithmes de Newton, les algorithmes génétiques, les stratégies d'évolution, les méthodes de surface de réponse, etc. Nous pouvons classifier ces algorithmes d'optimisation en 3 catégories comme suit :

- Algorithmes à direction de descente (algorithmes à base de gradient)
- Algorithmes d'ordre 0 et algorithmes évolutionnaires
- Algorithmes hybrides

Nous allons maintenant présenter ces trois classes d'algorithmes d'optimisation.

## I.2. METHODES A DIRECTION DE DESCENTE

Dans cette section, nous introduisons une classe importante d'algorithmes de résolution des problèmes d'optimisation : les algorithmes à direction de descente. Ce sont des algorithmes qui ont obligatoirement besoin de l'information du gradient des fonctions coût pour chercher

l'optimum du problème. Leur utilisation nécessite que la fonction coût soit au moins une fois différentiable par rapport aux paramètres à optimiser. Nous en décrirons, dans la première partie de cette section, les principes généraux.

## I.2.1. Principes généraux

Pour utiliser avec cette catégorie d'algorithmes, supposons que la fonction $\Phi$ est continue et différentiable dans tout l'espace de recherche. Nous notons $\nabla\Phi(\mu)$ et $\nabla^2\Phi(\mu)$ respectivement le vecteur gradient et la matrice hessienne de la fonction coût $\Phi(\mu)$ en $\mu$. La condition nécessaire d'optimalité du problème d'optimisation (1.1) s'écrit :

$$\nabla\Phi(\mu) = 0 \tag{1.2}$$

Lorsque la fonction coût $\Phi(\mu)$ est deux fois différentiable, on peut écrire une condition suffisante d'optimalité :

$$\begin{cases} \nabla\Phi(\mu) = 0 \\ \nabla^2\Phi(\mu) \quad \text{définie positive} \end{cases} \tag{1.3}$$

Les méthodes à direction de descente ont pour objectif de calculer un vecteur $\mu$ satisfaisant la condition nécessaire d'optimalité (1.2).

Pour une fonction coût donnée $\Phi(\mu)$, on dit que $d$ est une direction de descente de $\Phi(\mu)$ en $\mu \in \Re^n$ si la relation suivante est vérifiée :

$$d.\nabla\Phi(\mu) < 0 \tag{1.4}$$

Géométriquement cela veux dire que $d$ fait avec l'opposé du gradient $-\nabla\Phi(\mu)$ un angle $\theta$ strictement plus petit que $\frac{\pi}{2}$.

Si $d$ est une direction de descente, pour tout $\lambda > 0$ suffisamment petit, en utilisant un développement de Taylor d'ordre 1 nous avons :

$$\Phi(\mu + \lambda d) < \Phi(\mu) \tag{1.5}$$

Nous voyons que $\Phi(\mu)$ décroît dans la direction $d$. Les directions de descentes sont intéressantes car, pour faire diminuer $\Phi(\mu)$, il suffit de faire un déplacement de $\mu$ le long de $d$. L'algorithme général construit une suite d'itérés $\mu^k$ approchant une solution $\mu^*$ du problème (1.1) par la récurrence :

$$\begin{cases} \mu^0 \in \Re^n \\ \mu^{k+1} = \mu^k + \lambda^k d^k \quad , \forall k \geq 0 \end{cases} \tag{1.6}$$

où $\lambda^k > 0$ est appelé le pas de descente, $d^k$ une direction de descente de $\Phi$ en $\mu^k$.

Le pas de descente doit être déterminé de telle sorte que la valeur de la fonction décroisse le plus possible. En général, un algorithme de recherche linéaire est utilisé pour trouver un pas "optimal" à chaque itération.

L'algorithme général de cette classe de méthodes d'optimisation pour résoudre un problème sans contrainte s'écrit :

Choix d'un itéré initial $\mu^0 \in \Re^n$, et d'un paramètre d'arrêt ε

*Pour $k \geq 0$*

1. *tant que $\left\|\nabla\Phi\left(\mu^k\right)\right\| > \varepsilon$, continue ;*
2. *recherche d'une direction de descente $d^k$ ;*
3. *recherche linéaire : déterminer un pas $\lambda^k > 0$ minimisant $\Phi\left(\mu^k + \lambda d^k\right)$*
4. *actualisation pour l'itération suivante : $\mu^{k+1} = \mu^k + \lambda^k d^k$ ; $k = k+ 1$*
5. *aller à l'étape 1.*

Le test d'arrêt apporté à l'étape 1 porte sur la condition nécessaire d'optimalité d'ordre 1 : $\left\|\nabla\Phi\left(\mu^k\right)\right\| \approx 0$.

Pour définir une méthode de direction de descente, il faut donc préciser deux ingrédients principaux qui sont :

- la manière de choisir la direction de descente, qui donne le nom de l'algorithme
- la méthode de recherche linéaire pour déterminer le pas de descente $\lambda$ optimal à chaque itération

Quelques méthodes correspondant à des choix particuliers ces deux ingrédients sont décrits dans les paragraphes qui suivent.

## I.2.2. Choix de la direction de descente

### I.2.2.1. Méthode de la plus forte pente

Il s'agit de la méthode à direction de descente d'ordre 1 la plus simple. L'inverse du gradient est évidemment une direction de descente si $\mu^k$ n'est pas un point stationnaire. La direction normalisée *d* de plus forte décroissance de Φ au voisinage d'un point $\mu$ est donnée par :

$$d = -\frac{\nabla\Phi(\mu)}{\left\|\nabla\Phi(\mu)\right\|} \tag{1.7}$$

La méthode du gradient est facile à mettre en œuvre. Cependant, sa convergence peut être très lente. Loin de la solution, cette direction est une bonne direction de descente. En revanche, dans le voisinage d'une solution optimale $\mu^*$ du problème, là où les termes du second ordre de $\Phi(\mu)$ en $\mu^*$ jouent un rôle plus important, on observe une diminution très lente de $\Phi$. Cela est son défaut majeur. Il existe des techniques d'accélération permettant de palier à cet inconvénient [*Minoux 1983*]. Néanmoins, elles sont fort coûteuses et peu utilisées. Dans le domaine de mise en forme, *Jensen et al.* [*Jensen et al. 1998*] se sont servis cette méthode pour la minimisation de l'usure en emboutissage.

### I.2.2.2. Méthode du gradient conjugué

L'algorithme du gradient conjugué [*Morris 1982*] peut être vu comme une légère modification de l'algorithme de la plus forte pente, pour lequel on garde en mémoire la direction de descente de l'itération précédente. La direction de descente est donnée par :

$$d^k = \begin{cases} -\nabla\Phi\left(\mu^1\right) & si \quad k=1 \\ -\nabla\Phi\left(\mu^k\right)+\beta^{k-1}d^{k-1} & si \quad k\geq 2 \end{cases} \tag{1.8}$$

Le scalaire $\beta^k$ peut prendre différentes valeurs, ce qui donne à l'algorithme des propriétés différentes. Nous présentons ici les deux méthodes qui sont très fréquemment citées dans la littérature. Elles calculent $\beta^k$ par les formules suivantes :

- Méthode de Fletcher-Reeves [*Fletcher et al. 1964*] :

$$\beta^k = \frac{\left\|\nabla\Phi\left(\mu^k\right)\right\|^2}{\left\|\nabla\Phi\left(\mu^{k-1}\right)\right\|^2} \tag{1.9}$$

- Méthode de Polak-Ribière [*Polak et al. 1969*] :

$$\beta^k = \frac{\left[\nabla\Phi\left(\mu^k\right)-\nabla\Phi\left(\mu^{k-1}\right)\right]^T\nabla\Phi\left(\mu^k\right)}{\left\|\nabla\Phi\left(\mu^{k-1}\right)\right\|^2} \tag{1.10}$$

Cet algorithmes est proposé à l'origine pour la minimisation de fonctions quadratiques, puis il est étendu à des fonctions quelconques dans [*Fletcher et al. 1964*]. Néanmoins, il cumule les erreurs d'arrondi, la convergence n'est alors assurée que si l'on procède à des réinitialisations périodiques. Certaines techniques de redémarrage ont été proposées (méthode de Beales [*Powell 1975*] par exemple). *Zabaras et al.* [*Zabaras et al. 1995*] ont utilisé cette méthode pour l'optimisation du flux de chaleur en solidification.

### I.2.2.3. Méthode de Newton

L'algorithme général de Newton est une méthode de résolution de systèmes d'équations non linéaires. Dans le cadre de l'optimisation, il est utilisé comme un algorithme de direction de descente sur la condition nécessaire d'optimalité (1.2).

Pour décrire cet algorithme, nous considérons l'approximation quadratique $Q(\mu)$ de $\Phi(\mu)$ au voisinage d'un point $\mu^k$ à l'itération *k*:

$$Q(\mu) = \Phi(\mu^k) + \nabla\Phi(\mu^k).(\mu - \mu^k) + \frac{1}{2}(\mu - \mu^k)^t.\nabla^2\Phi(\mu^k).(\mu - \mu^k) \tag{1.11}$$

On choisit alors $\mu^{k+1}$ qui minimise $Q$. Il est définit par la condition d'optimalité (1.2) ce qui conduit au système linéaire suivant pour déterminer la direction de descente $d^k$:

$$d^k = -\left[\nabla^2\Phi(\mu^k)\right]^{-1}.\nabla\Phi(\mu^k) \tag{1.12}$$

Cette méthode est reconnue comme une des plus efficaces dans la famille des algorithmes à direction de descente. Sa principale difficulté réside dans le calcul des dérivées secondes de Φ qui s'avère le plus souvent coûteux et difficile à réaliser.

Plusieurs algorithmes proposent de lever cette difficulté en utilisant une approximation de la matrice hessienne. On peut mentionner le cas particulier où Φ peut s'écrire sous forme de moindres carrés. On obtient alors une approximation du hessien en ne considérant que les produits des gradients. Cette approximation, qui est à la base des algorithmes de Gauss-Newton ou Levenberg-Marquardt [*Minkowycz 1988*], est largement utilisée en identification de paramètres rhéologiques [*Gavrus 1996*] [*Ghouati 1998*].

#### I.2.2.4. Méthodes quasi-Newton

L'idée des méthodes de quasi-Newton est de remplacer la matrice hessienne $\nabla^2\Phi(\mu^k)$ (ou son inverse) par une approximation mise à jour itérativement. La direction de descente s'écrit :

$$d^k = -\left[\tilde{H}^k\right]^{-1}.\nabla\Phi(\mu^k) \tag{1.13}$$

où la matrice $\tilde{H}^k$ doit converger vers la matrice hessienne lorsqu'on s'approche de l'optimum. Pour tout *k*, la matrice $\tilde{H}^k$ doit vérifier plusieurs conditions. D'une part, elle doit être symétrique et définie positive. D'autre part, on impose à la matrice $\tilde{H}^{k+1}$ de vérifier la relation de quasi-Newton suivante :

$$\nabla\Phi(\mu^{k+1}) - \nabla\Phi(\mu^k) = \tilde{H}^{k+1}(\mu^{k+1} - \mu^k) \tag{1.14}$$

Plusieurs méthodes pour construire la suite $\tilde{H}^k$ ont été proposées. Les deux méthodes les plus connues sont de rang 2, celle DFP de Davidon-Fletcher-Powell [*Fletcher et al. 1963*] et celle BFGS [*Broyden et al. 1970*].

Pour un problème d'extrusion, Kusiak et al. [*Kusiak et al. 1989*] ont montré l'efficacité des méthodes de quasi-Newton en comparant une méthode DFP, un algorithme de plus forte pente et une méthode d'ordre 0 le simplex. Avec de nombreuses utilisations comme dans [*Noiret et al. 1996*], [*Zhao et al. 1997*], [*Bourdin et al. 1998*], [*Vieilledent et al. 1998*], la méthode BFGS semble la plus adaptée aux problèmes d'optimisation en mise en forme.

## I.2.3. Méthodes de recherche linéaire

Une fois que la direction de descente $d^k$ à l'itération $k$ est déterminée, on effectue l'étape de recherche linéaire pour trouver la valeur optimale du pas de descente $\lambda^k$. La recherche linéaire permet aux méthodes à direction de descente de converger, la convergence n'étant pas assurée si l'on néglige cette étape. Cette dernière constitue souvent la partie la plus coûteuse en temps de calcul, car elle nécessite plusieurs évaluations de la fonction Φ. Le pas de descente $\lambda^k$ trouvé doit satisfaire deux conditions : Faire décroître Φ et ne pas être trop petit pour éviter une fausse convergence.

La recherche linéaire est un problème d'optimisation avec un seul paramètre. Il existe donc plusieurs méthodes et plusieurs critères d'arrêt de cette recherche tels que celui de Curry, celui d'Admijo ou Goldstein [*Armijo 1966*], ou celui de Wolf [*Wolf 1971*] pour BFGS. Nous pouvons citer ici quelques méthodes de recherche linéaire : celle de Newton (ou de la sécante), la méthode d'interpolation quadratique [*Fletcher 1996*], la méthode de la section d'or [*Morris 1982*] [*Chung et al. 1998*], la méthode de Brent avec gradient [*Chen et al. 1995*], la méthode d'interpolation parabolique [*Press et al. 1992*], la méthode de Davidon [*Schittkowski 1980*], la méthode de type Moré et Thuente [*Moré et al. 1994*], etc.

## I.2.4. Avantages – Inconvénients

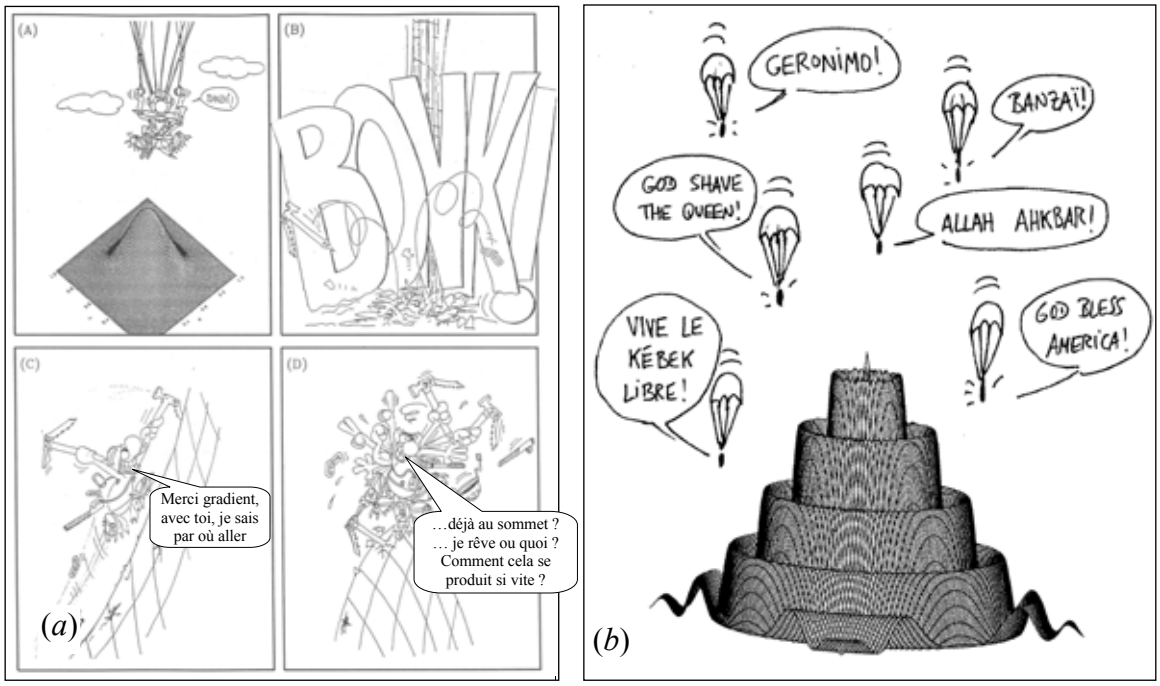



*Figure 1.1. Méthode à direction de descente: devant un problème convexe, je trouve rapidement l'optimum (a); devant un problème multi-optima : mon seigneur, envoyez à un endroit idéal (b)*

Les méthodes à direction de descente ont l'avantage de converger relativement rapidement vers un point stationnaire en utilisant les informations du gradient. Ces méthodes donc sont très favorables pour un problème convexe unimodal grâce à leur vitesse de convergence (*Figure 1.1a*). Elles nécessitent que la fonction coût soit au moins une fois différentiable par

rapport au paramètres à optimiser ce qui est un problème délicat pour des codes complexes tels qu'en 3D instationnaire. Elles ne permettent d'atteindre que le minimum le plus proche du point de départ, c'est-à-dire un minimum local. Le résultat dépend donc fortement du jeu de paramètres servant à l'initialisation (*Figure 1.1b*).

On peut aussi les utiliser pour accélérer la vitesse de convergence d'algorithmes globaux (comme les algorithmes évolutionnaires) lorsque ces algorithmes ont du mal à converger dans la région proche de l'optimum global.

Les méthodes à direction de descente ne sont donc pas le choix idéal pour résoudre des problèmes d'optimisation multi optima comme celui présenté sur la *Figure 1.1b*. Pour résoudre un tel problème, il vaut mieux utiliser des algorithmes plus globaux.

## I.3. METHODES D'ORDRE 0 ET ALGORITHMES DE MINIMISATION GLOBALE

Les méthodes d'ordre 0 ne nécessitent pas le calcul du gradient de la fonction coût. Les algorithmes les plus connus de cette catégorie sont ceux qui s'inspirent de la nature, tels que les algorithmes évolutionnaires, l'algorithme du recuit simulé, l'algorithme de la colonie des fourmis, la méthode de recherche aléatoire, la méthode du simplex, les méthodes de surface de réponse. La plupart de ces algorithmes sont des méthodes d'optimisation globales, sauf celui du simplex.

Avec ce type d'algorithmes, on cherche à générer un ou plusieurs nouveaux points, plus proches de l'optimum, uniquement à partir de la connaissance de la valeur de la fonction coût $\Phi$ d'un ou plusieurs points de l'espace des paramètres.

### I.3.1. Algorithmes évolutionnaires

Les Algorithmes Evolutionnaires (AE) constituent une discipline impliquant la simulation par un ordinateur du processus de l'évolution naturelle. Ils sont inspirés de la génétique et des mécanismes de la sélection naturelle basés sur la théorie de l'évolution de Darwin, selon laquelle la vie est une compétition où seuls les mieux adaptés survivent et se reproduisent. Ils empruntent les paradigmes de l'évolution biologique tels que la sélection, le croisement et la mutation pour chercher la solution du problème. Les AE utilisent la notion de "population d'individus", dans laquelle chaque individu représente une solution potentielle de l'espace de recherche du problème donné.

Ce sont des méthodes d'optimisation globales. Leur robustesse et leur souplesse permettent d'aborder les problèmes les plus raides. De plus, leur capacité à travailler sur des espaces de recherche non standards (non continus) ainsi que leur faible besoin d'information sur le problème (seulement la fonction coût) offrent les perspectives les plus originales et un large champ d'application. Ils ont donc été appliqués avec succès à de nombreux problèmes où les algorithmes classiques d'optimisation sont incapables de produire des résultats satisfaisants. En outre, en tant qu'algorithme à base de population, leur parallélisation est aisée : il suffit de distribuer l'évaluation de la fonction coût sur autant de processeurs que d'individus de la

population. Leur principal inconvénient est leur coût. Ils nécessitent en effet un grand nombre d'évaluations pour aboutir à l'optimum : c'est le prix qu'ils doivent payer au fait de ne pas utiliser d'autre information sur la fonction et de s'appliquer à de très larges classes de problèmes, aussi chaotiques soient ils.

Nous allons maintenant présenter de manière plus détaillée ces algorithmes. Commençons par leur historique et leurs domaines d'application.

### I.3.1.1. Historique et domaines d'application

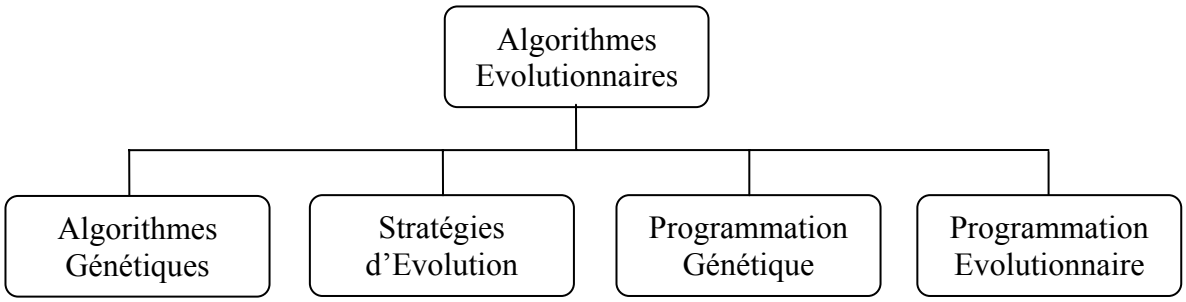



*Figure 1.2. Principales catégories des Algorithmes Evolutionnaires*

Historiquement, les AE ont été élaborés depuis les années soixante. En général, ils sont divisés en quatre catégories principales, comme présentés dans la *Figure 1.2*.

#### *I.3.1.1.1. Algorithmes Génétiques (AG)*

Les AG ont été mis au point par Holland dans les années 60 [*Holland 1962*], [*Holland 1975*] aux Etats-Unis. Ils sont ensuite raffinés et popularisés par De Jong [*De Jong 1975*], Grefenstette [*Grefenstette 1987*], Goldberg [*Goldberg 1989*]. Ces algorithmes s'appuient avant tout sur une représentation binaire des individus.

#### *I.3.1.1.2. Stratégies d'Evolution (SE)*

Les premiers efforts pour la mise en place des Stratégies d'Evolution (SE) ont eu lieu dans les années 60 en Allemagne par Rechenberg [*Rechenberg 1965*] [*Rechenberg 1973*] et Schwefel [*Schwefel 1981*]. Ces algorithmes s'appuient sur une représentation en nombres réels et de dimension fixe des individus, ainsi que sur un opérateur de mutation gaussienne. Les SE les plus performantes utilisent les mutations auto adaptatives, dans lesquelles chaque individu porte avec lui les paramètres de la mutation gaussienne qui lui sera appliquée – paramètres eux-même soumis à mutation [*Bäck 1995*].

#### *I.3.1.1.3. Programmation Génétique (PG)*

Proposée par Cramer [*Cramer 1985*], la Programmation Génétique (PG) a surtout été popularisée par Koza au début des années 90 [*Koza 1992*], [*Koza 1994*], [*Koza et al. 1999*], [*Banzhaf et al. 1998*]. Elle s'intéresse à l'évolution de programmes. Elle propose un paradigme permettant la programmation automatique d'ordinateurs par des heuristiques

basées sur les mêmes principes d'évolution que les AG. La différence entre la PG et les AG réside essentiellement dans la représentation des individus. En effet, la PG consiste à faire évoluer des individus dont la structure est similaire à celle des programmes informatiques. Koza représente les individus sous forme d'arbres, c'est-à-dire de graphes orientés et sans cycle, dans lesquels chaque noeud est associé à une opération élémentaire relative au domaine du problème. La PG est particulièrement adaptée à l'évolution de structures complexes de dimensions variables.

#### *I.3.1.1.4. Programmation Evolutionnaire (PE)*

La Programmation Evolutionnaire (PE) est introduite dans les années 60 par L. Fogel [*Fogel 1964*], [*Fogel et al. 1966*], puis étendue par Burgin [*Burgin 1973*], Atmar [*Atmar 1976*], D.B. Forgel [*Forgel 1992*] et d'autres. Elle a été conçue dans le but de faire évoluer des machines à états finis, puis a été étendue aux problèmes d'optimisation de paramètres. Cette approche met l'emphase sur la relation entre les parents et leurs descendants plutôt que sur les opérateurs génétiques. Elle a été utilisée dans de nombreux autres champs d'application. Les caractéristiques de l'évolution sont très proches de celles des SE. Contrairement aux trois autres AE classiques, la PE n'utilise pas une représentation spécifique des individus mais plutôt un modèle évolutionnaire de haut niveau, qui est associé à une représentation et à un opérateur de mutation directement appropriés au problème à résoudre.

#### *I.3.1.1.5. Champs d'application*

Les champs d'application des AE sont très vastes : en économie [*Vallée al. 2001*], en finance, en optimisation de fonctions numériques difficiles (discontinue, multimodales, bruitées) [*De Jong 1980*], en traitement d'image (alignement de photos sattelites, reconnaissance de suspects), en théorie du contrôle optimal, ou encore en théorie des jeux répétés et différentiels, en mécanique des structures [*Burczynski et al. 2001*], [*Chen 2001*], [*Papadrakakis et al. 2001*], en optimisation de forme [*Annicchiarico et al. 1999*], [*Antonio et al. 2002*], [*Chung et al. 1997*], [*Mori et al.1996*], etc. Pour plus d'information sur les applications des AE, le lecteur peut se référer à [*Oduguwa et al. 2005*]

### I.3.1.2. Vocabulaire associé

Comme les algorithmes évolutionnaires sont développés à partir de raisonnements issus de la biologie, les termes utilisés en gardant les dénominations. Pour éviter toute confusion de langage, il est nécessaire, suivant en cela Lutton [*Lutton 1999*], de leur préciser le vocabulaire. L'équivalence entre les termes biologiques et les termes d'optimisation est présentée dans le tableau *Tableau 1.1*.

En plus de ce vocabulaire, il nous faut encore distinguer entre le "Génotype" et le "Phénotype". On parle de génotype pour tout ce qui concerne les chromosomes, tandis que les solutions (les vecteurs de l'espace de recherche) constituent le phénotype. Les AE travaillent donc au niveau du génotype.

| Algorithmes Evolutionnaires | Méthodes d'optimisation |
|---|---|
| *Individu* | Solution potentielle de l'espace de recherche (vecteur des paramètres) |
| *Chromosome* | Solution codée (à partir d'une variable binaire, réelle, discrète, etc.) |
| *Gène ou Allèle* | Partie composante d'un individu (d'un chromosome) |
| *Population* | Ensemble fini (de taille N) d'individus |
| *Performance (ou fonction coût)* | Mesure de la qualité des individus basée sur la valeur de la fonction coût et permettant de comparer les individus entre eux afin de déterminer les plus et moins aptes |
| *Evaluation d'un individu* | Calcul de la performance d'un individu |
| *Croisement (ou recombinaison)* | Opérateur de reproduction appliqué aux individus de la population et qui consiste à échanger ou combiner des composantes entre plusieurs individus. |
| *Mutation* | Opérateur de modification d'un ou plusieurs gènes d'un individu dans le but d'introduire une nouvelle variabilité dans la population |
| *Sélection* | Processus du choix des individus utilisés pour la reproduction basé sur leur performance |
| *Environnement* | Espace de recherche |
| *Remplacement* | Processus de formation d'une nouvelle population à partir de l'ensemble des parents et des enfants, effectué le plus souvent sur la base de leur performance |
| *Evolution* | Un processus itératif de recherche d'un (ou plusieurs) individu optimal |
| *Génération* | Repère le moment de l'évolution |

*Tableau 1.1 : Equivalence entre « Termes biologiques » et « Termes d'optimisation »*

#### I.3.1.3. Principe de fonctionnement des AE

Le principe de fonctionnement des AE est extrêmement simple, et est présenté par l'organigramme de la *Figure 1.3*.

Pour résoudre un problème d'optimisation, on part d'un ensemble (population) de solutions potentielles (individus) arbitrairement choisies. Les performances de tous les individus de cette population sont évaluées. Sur cette base, l'application des trois opérateurs évolutionnaires de "sélection, croisement et mutation" permet de créer un nouvel ensemble d'individus, appelé "population des enfants". Cette nouvelle population doit être évaluée à son tour afin de décider les quels des enfants méritent de remplacer certains parents et de faire partie de la génération suivante. Le test d'arrêt est ensuit effectué. Si les critères sont vérifiés, on s'arrête. Autrement, on recommence le cycle jusqu'à satisfaction de ces critères.

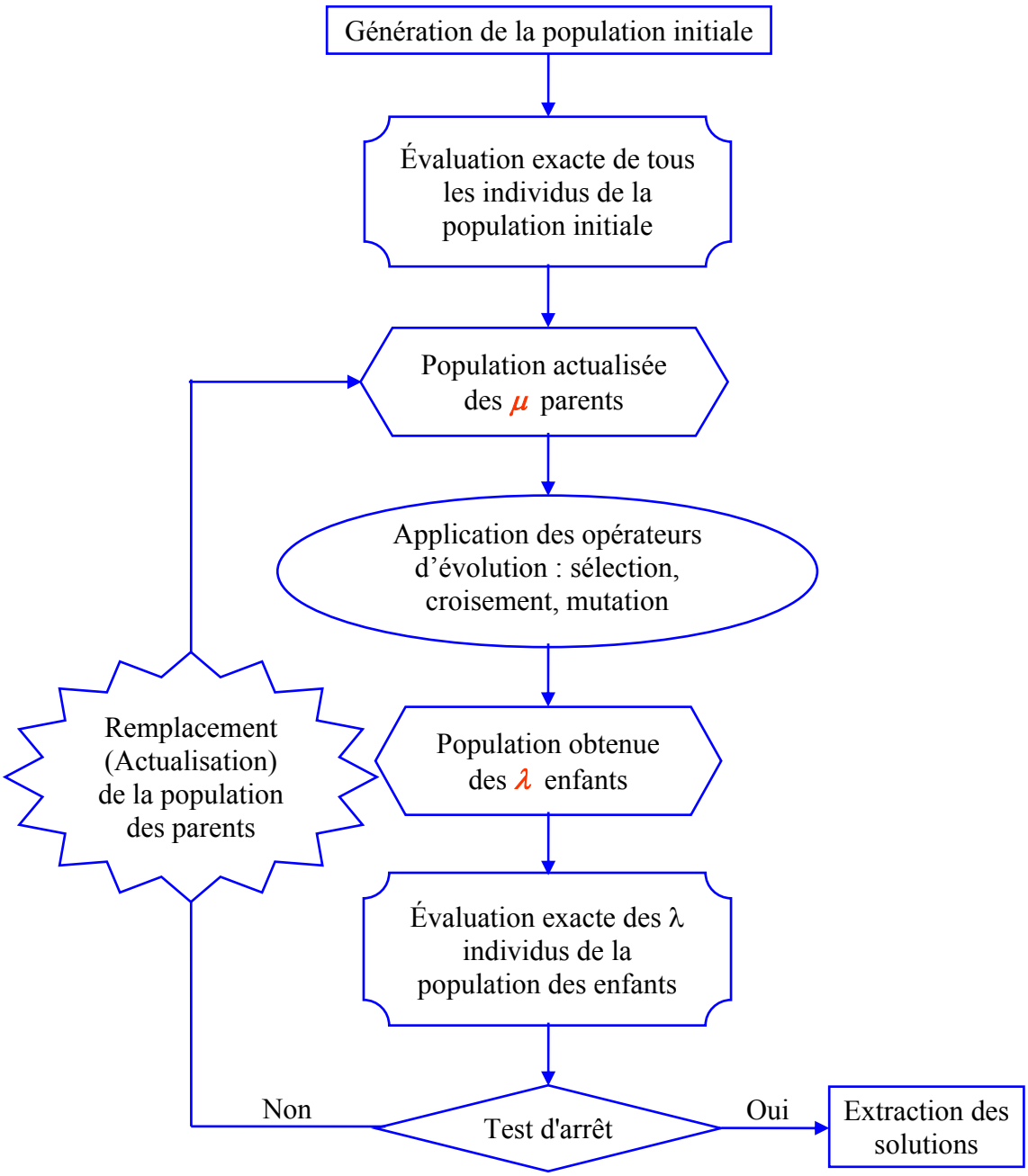



*Figure 1.3. Organigramme canonique d'un algorithme évolutionnaire*

Le critère d'arrêt des AE peut être la convergence de l'ensemble des solutions vers le même extremum ou quand le meilleur individu de la population atteint un seuil de performance fixé. Le plus souvent le processus est arrêté au bout d'un nombre d'itérations fixé a priori. Notons que la plupart des AE travaillent avec une population de taille fixe, ce qui n'est pas le cas dans la nature.

Nous allons maintenant expliquer en détails les mécanismes de fonctionnement des AE en présentant donc les AG et les SE comme nous nous en servirons pour résoudre nos problèmes d'optimisation. Pour plus de détails sur la programmation génétique et celle évolutionnaire, le lecteur peut se référer aux ouvrages mentionnés dans la partie *I.3.1.1*.

### I.3.1.4. Mécanismes de fonctionnement des AE

Pour faire fonctionner les opérateurs génétiques, les AE utilisent un codage (ou représentation) des paramètres à la place des paramètres eux-mêmes. Plusieurs codages ont été développés par différents auteurs comme le codage binaire, le codage réel, le codage de Gray, le codage de permutation, le codage d'état fini, le codage de type «arborescentes ». Les deux codages les plus utilisés sont ceux binaire (utilisé par les AG) et réel (utilisé par les SE).

#### *I.3.1.4.1. Algorithmes génétiques*

❖ **Représentation**

Les AG traditionnels utilisent le codage binaire comme représentation des solutions. Chaque individu est représenté par un vecteur binaire (ou chaîne de bits), dont chaque élément prend la valeur 0 ou 1. Ce vecteur est une concaténation des paramètres à optimiser, chaque paramètre étant transformé en une série binaire. La *Figure 1.4* présente un exemple du codage binaire d'une solution avec 3 paramètres. Chaque paramètre est représenté par une série binaire de 4 chiffres $\{0,1\}$.

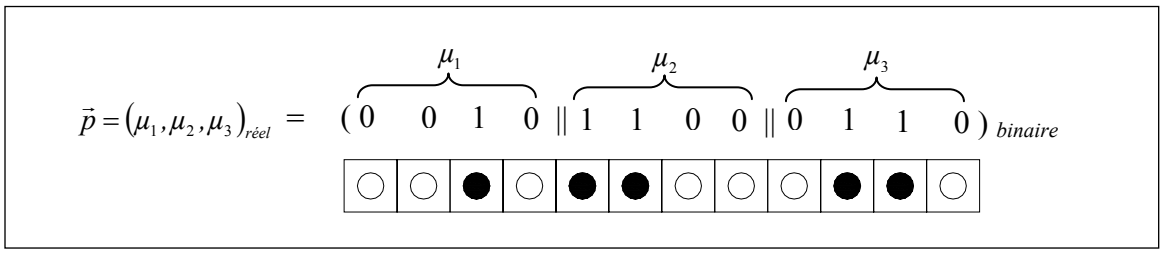



*Figure 1.4. Exemple du codage binaire d'une solution potentielle avec 3 paramètres*

Cette représentation s'adapte bien aux problèmes où les paramètres ont une représentation binaire canonique, comme les problèmes booléens. Elle s'applique aussi aux problèmes d'optimisation paramétrique continus ($\Phi : \Re^n \to \Re$), mais il est alors nécessaire de définir une technique de codage adéquate de $\Re^n$ vers $\{0,1\}^l$.

La précision du codage dépend du nombre de bits (et donc de la précision de la solution trouvée), plus *l* est grand, plus c'est précis, plus la convergence est longue.

❖ **Sélection des parents**

La sélection est un opérateur clé sur lequel repose en partie la qualité d'un algorithme génétique. Dans cette étape, les chromosomes de la population actuelle sont sélectionnés pour être les parents de la génération suivante. En accord avec la théorie de l'évolution de Darwin, les meilleurs individus doivent survivre et en créer les nouveaux. Il existe plusieurs méthodes pour choisir les meilleurs individus, par exemple la sélection proportionnelle, la sélection par tournoi, la sélection par rang, la sélection selon l'état d'équilibre, etc. Parmi celles-ci, la sélection proportionnelle et la sélection par tournoi sont les méthodes les plus utilisées. On présente ici les méthodes les plus courantes :

### ↳ *Sélection proportionnelle (Roue de loterie – Roulette selection)*

La méthode de sélection « Tirage à la roulette» introduite par [*Goldberg 1989*] est la méthode la plus connue et la plus utilisée. C'est une méthode stochastique qui reproduit une roulette de casino qui compterait autant de cases que d'individus dans la population. La largeur de la case d'un individu $\vec{x}_i$ est proportionnelle à sa performance $f(\vec{x}_i)$ et prend la valeur $\frac{f(\vec{x}_i)}{\sum_{j=1}^{N} f(\vec{x}_j)}$ . La roue est lancée, l'individu sélectionné est désigné par l'arrêt de la roue sur sa case. Pour un problème de maximisation, la performance est la valeur de la fonction coût, pour un problème de minimisation, la performance est l'inverse de la valeur de fonction coût.

L'espérance $n_i$ de la sélection d'un élément $\vec{x}_i$ de la population courante est donnée par l'expression :

$$n_i = \frac{N}{\sum_{j=1}^{N} f(\vec{x}_j)} f(\vec{x}_i) \tag{1.15}$$

où N est le nombre d'individus parents.

L'espérance maximale Max($n_i$) de l'ensemble des individus de la population est appelée la pression sélective.

Cette méthode favorise les meilleurs individus mais tous les individus ont des chances d'être choisis. Néanmoins, la méthode peut causer une perte de la diversité de la population si la pression sélective (ou l'espérance $n_i$ du meilleur individu) est élevée. De plus, sa variance est élevée.

Pour diminuer la variance, Baker a proposé en 1987 la méthode de sélection à la roulette avec reste stochastique [*Baker 1987*]. L'approche est similaire à celle de Goldberg, mais cette fois, les individus sélectionnés sont désignés par un ensemble de points équidistants. Le nombre effectif de sélections de l'individu $\vec{x}_i$ sera la partie entière inférieure ou supérieure de son espérance $n_i$.

### ↳ *Sélection par tournoi*

Une sélection par tournoi consiste à sélectionner un sous-ensemble de la population, et à ne conserver que le meilleur individu du sous-ensemble. L'opération recommence jusqu'à l'obtention du nombre d'individus requis. Au cours d'une génération, il y a autant de tournois que d'individus à sélectionner. La pression de sélection est ajustée par le nombre $q$ de participants à un tournoi. Un $q$ élevé conduit à une forte pression de sélection.

L'avantage de cette technique est qu'elle est paramétrable par la valeur de $q$, et peu sensible aux erreurs sur Φ. Par contre, sa variance est élevée [*De Jong et al. 1995*].

Une variante de la sélection par tournoi est le tournoi de Boltzmann [*Maza et al. 1993*], qui assure que la distribution des valeurs d'adaptation d'une population soit proche d'une distribution de Boltzmann. Pour une compétition entre une solution courante $\vec{x}_i$ et une solution alternative $\vec{x}_j$, $\vec{x}_i$ gagne avec la probabilité $\frac{1}{1+e^{\left(\frac{\Phi(\vec{x}_i)-\Phi(\vec{x}_j)}{T}\right)}}$, où T est la température de sélection.

↳ *Sélection par rang*

C'est une sélection qui n'est pas directement proportionnelle à la valeur de la fonction mérite, mais dépend plutôt du rang de l'individu dans la population. Le classement par rang a principalement pour but d'éviter que les individus avec des fonctions de mérite très élevées ne perturbent le processus stochastique en obtenant un nombre de sélections trop important.

↳ *Sélection par troncature*

Cette sélection consiste à choisir de manière déterministe les T% meilleurs individus d'une génération pour générer la suivante. Pour plus de détails sur ces méthodes de sélection et pour les autres méthodes de sélection, on peut se référer à la partie C.2 de [*Bäck et al. 1997*].

❖ **Remplacement**

L'étape de remplacement sert à déterminer quels individus parmi les parents de la génération courante et leurs enfants, seront les parents de la génération suivante. A la différence de l'étape de sélection, durant laquelle des individus peuvent être sélectionnés plusieurs fois, lors de l'étape de remplacement, un individu est ici sélectionné une fois – et il survit alors à la génération suivante – ou pas du tout et il disparaît définitivement de l'évolution en cours. Plusieurs stratégies de remplacement sont présentées dans la littérature pour les AG :

↳ *Remplacement générationnel*

La nouvelle population est composée uniquement des enfants. On fait disparaître tous les individus de la population courante. L'inconvénient majeur de cette approche est la perte de meilleur individu, si on ne le conserve pas systématiquement dans la nouvelle population.

↳ *Remplacement élitiste*

La nouvelle génération garde certaines "bonnes" solutions (sélection élitiste) de la génération courante et est complétée par des enfants.

↳ *Remplacement continu*

Des enfants, choisis aléatoirement, remplacent de façon régulière les individus les moins performants de la génération courante.

❖ **Croisement**

Le croisement est l'opérateur principal des AG. C'est un opérateur génétique relatif à plusieurs individus parents (souvent deux). Son rôle consiste à combiner les génotypes des individus pour en produire un nouveau. Il fait partie du mécanisme de convergence de l'AG, qui permet de concentrer la population autour des meilleurs individus. On distingue plusieurs types de croisements possibles. Les plus utilisés sont :

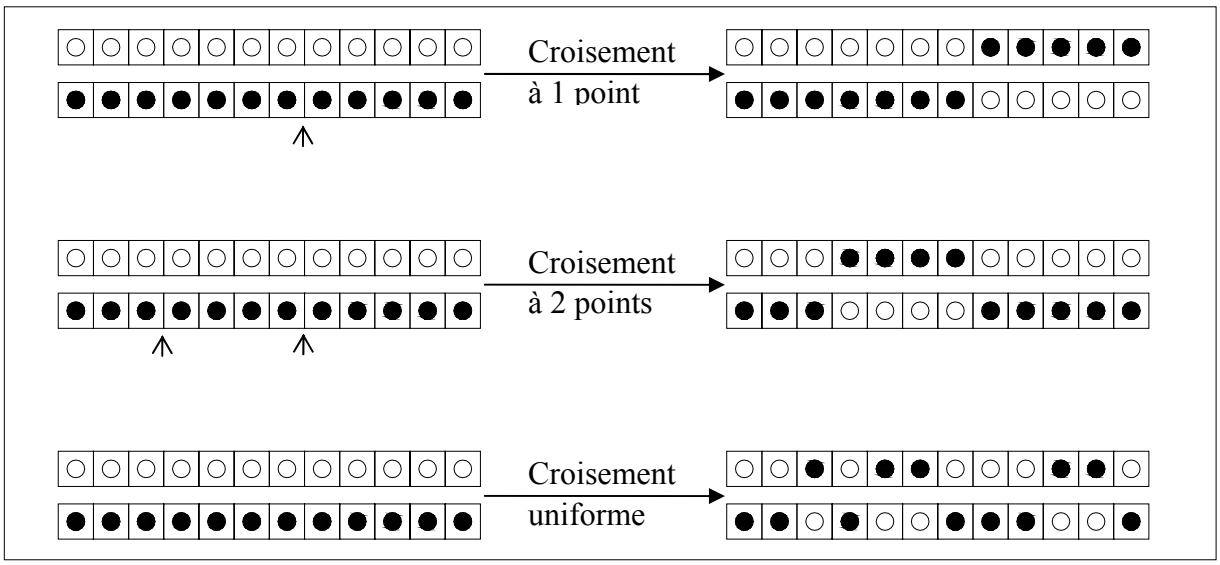



*Figure 1.5. Méthodes de recombinaison (croisement) utilisées par l'AG*

- *Croisement à 1 point* [*Holland 1962*]

Le croisement à un point est l'opérateur de croisement le plus simple et le plus classique. Il consiste à choisir aléatoirement un point de coupure, puis à subdiviser le génotype de chacun des parents en deux parties de part et d'autre de ce point. Les fragments obtenus sont alors échangés pour créer les génotypes des enfants (*Figure 1.5*).

- *Croisement à multipoints* [*De Jong et al. 1991*]

Le croisement multipoints est une généralisation du croisement à un point. Au lieu de choisir un seul point de coupure, on en sélectionne *k*, aléatoirement. Dans le croisement multipoints, les points de coupure sont fixés par avance. La *Figure 1.5* représente un croisement multipoints (deux points dans l'exemple).

- *Croisement uniforme* [*Syswerda 1989*]

Ce type de croisement est la généralisation du croisement multipoints. Dans le croisement uniforme, chaque gène d'un enfant est choisi aléatoirement entre les gènes des parents ayant la même position dans le chromosome, avec une probabilité de 0,5 s'il y a deux parents. Le second enfant est construit en prenant les choix complémentaires du premier enfant. Un exemple du croisement uniforme est aussi présenté sur la *Figure 1.5*.

### ❖ Mutation

Les AG utilisent l'opérateur de mutation comme moyen de préserver la diversité de la population. La mutation utilisée est binaire. Elle inverse aléatoirement les bits du génotype, avec une faible probabilité, typiquement de 0,01 à 0,001. La *Figure 1.6* nous montre un exemple de la mutation binaire.

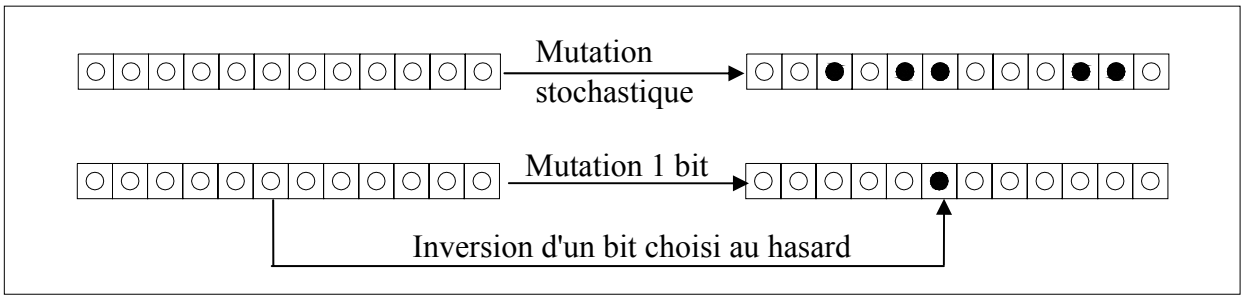



*Figure 1.6. Méthodes de mutation utilisées par le AG*

- *Mutation stochastique (bit flip)* :

Etant la plus employée avec le codage binaire, cette méthode de mutation consiste à inverser indépendamment chaque bit du chromosome. Un test sur le taux de mutation est effectué pour chacun des bits du chromosome : en cas de succès, le bit testé est alors inversé.

- *Mutation 1 bit*

Un bit du chromosome est choisi au hasard. Sa valeur est alors inversée.

#### *I.3.1.4.2. Stratégies d'Evolution*

Contrairement aux AG, la représentation utilisée par les SE est celle de vecteurs à valeurs réelles des paramètres ( $\vec{p} \in \Re^n$ ) lorsqu'elles sont utilisées pour des problèmes d'optimisation continus. Rechenberg [Rechenberg 1994] a proposé d'associer à chaque valeur de la solution un vecteur auxiliaire qui détermine comment faire varier cette solution [*Schwefel 1981*] et permet l'auto-adaptation. Un individu consiste donc maintenant en le vecteur des paramètres d'objet $\vec{p}$ et celui de paramètres stratégiques $\vec{\sigma}$ - l'écart type de mutation :

$$\vec{a} = (\vec{p}, \vec{\sigma}) \tag{1.16}$$

❖ **Sélection et Remplacement**

La sélection d'un ou plusieurs parents pour générer les nouveaux individus dans les SE est semblable à celle des AG. En revanche, le remplacement est complètement déterministe, ce qui lui donne un rôle clef dans l'évolution en guidant la recherche vers les meilleurs individus. Pour produire les nouveaux individus, il opère en sélectionnant les $\mu$ ($1 < \mu < \lambda$) premiers individus par ordre de performance et remplace les anciens parents par :

- l'union de $\mu$ parents et $\lambda$ enfants : schéma appelé stratégie $(\mu + \lambda)$
- l'ensemble des $\lambda$ enfants : schéma appelé stratégie $(\mu, \lambda)$

La stratégie ($\mu+\lambda$) est élitiste. Elle garantie une amélioration monotone de la performance de la population, mais elle peut converger prématurément car elle s'adapte mal à un éventuel changement d'environnement. En revanche, avec la stratégie ($\mu$ , $\lambda$), la valeur de la meilleure performance peut décroître si tous les enfants sont inférieurs aux parents, mais l'algorithme est plus flexible vis-à-vis des changements d'environnement. De plus, la régression des meilleures performances peut aider le processus de recherche à sortir des régions d'attraction des optima locaux pour aller explorer ailleurs.

### ❖ Croisement

Le croisement est facultatif pour les SE. Il n'est pas obligatoirement présent dans la génération des nouveaux individus. Avec le codage réel, le croisement qui échange les informations entre les parents peuvent sembler le même que dans le cas du codage binaire des AG, mais il existe une différence fondamentale [*Séfrioui 1998*] : avec le codage réel, le point de coupure tombe nécessairement entre deux composantes du vecteur des paramètres alors que pour le codage en binaire le point de coupure peut tomber à l'intérieur d'une composante. On peut aussi réaliser des croisements intermédiaires comme présentés dans la formule (1.17). La recombinaison peut être différente des paramètres d'objet et pour les paramètres stratégiques. Un exemple de règles de recombinaison pour le vecteur des paramètres d'objet $\vec{p}=(p_1,p_2,...,p_i,...,p_N)$ est présenté dans [*Bäck et al. 1993*]:

$$p'_i=\begin{cases} p_{S,i} & \textit{sans recombinaison} \\ p_{S,i} \;\; ou \;\; p_{T,i} & \textit{recombinaison discrète} \\ p_{S,i}+\chi.(p_{T,i}-p_{S,i}) & \textit{recombinaison intermédiaire} \\ p_{S_i,i} \;\; ou \;\; p_{T_i,i} & \textit{recombinaison discrète globale} \\ p_{S_i,i}+\chi_i.(p_{T_i,i}-p_{S_i,i}) & \textit{recombinaison intermédiaire globale} \end{cases} \qquad (1.17)$$

Les indices *S* et *T* représentent les deux individus choisis aléatoirement dans la population des $\mu$ parents. $\chi\in[0,1]$ est une variable aléatoire uniforme; la valeur $\chi=1/2$ est souvent utilisée. Avec la recombinaison globale, pour chaque composante $p'_i$ de l'individu enfant $\vec{p}'$ les parents $S_i$, $T_i$ et la variable $\chi_i$ sont déterminés indépendamment. Empiriquement, la recombinaison discrète sur les paramètres d'objet et celle intermédiaire sur les paramètres stratégiques semblent donner les meilleurs résultats. La recombinaison sur les paramètres stratégiques s'avère obligatoire pour le mécanisme de « recombinaison » des SE fonctionne [*Bäck et al. 1993*].

### ❖ Mutation gaussienne auto-adaptative

Contrairement aux AG, l'opérateur de mutation est toujours présent dans l'évolution des SE (tandis que le croisement est facultatif). La mutation garantit la globalité de la recherche : c'est le principal opérateur d'exploration. Les SE peuvent donc fonctionner avec une population d'un seul individu.

Toutefois, lorsque l'opérateur de mutation a une intensité variable (comme c'est le cas pour l'opérateur de mutation gaussienne auto-adaptatif décrit ci-dessous), la mutation peut aussi

être un opérateur d'exploitation. Dans le cadre de génotypes réels, l'opérateur de mutation le plus efficace et le plus utilisé est la mutation gaussienne auto-adaptative, que nous allons maintenant détailler.

Le principe de base de la mutation gaussienne est d'ajouter un bruit gaussien centré $N(0,\sigma)$ aux variables que l'on désire faire muter :

$$p'_i = p_i + N(0,\sigma) \tag{1.18}$$

Tout l'art réside dans le choix du paramètre $\sigma$, la variance de la perturbation gaussienne, et définissant donc l'intensité de la mutation suivant une loi normale : environ 67% des tirages seront compris entre $-\sigma$ et $+\sigma$ et plus de 99% entre $-3\sigma$ et $+3\sigma$. Il y a également une probabilité strictement positive de tirer tout nombre réel positif ou négatif car la "queue" de la distribution ne s'annule jamais. Tous les ingrédients sont donc présents pour pouvoir faire de cette mutation un opérateur soit d'exploration (grandes valeurs de $\sigma$) soit d'exploitation (petites valeurs de $\sigma$). Il faut par contre trouver la bonne loi de mise à jour de $\sigma$ pour un bon compromis entre ces deux comportements.

Une première idée fut de faire muter plus fortement les mauvais individus (puisqu'il ne sert à rien de les exploiter, autant explorer davantage) et faiblement les bons individus (pour exploiter l'espace de recherche autour d'eux). Cette idée, utilisée dans les premiers temps de la Programmation évolutionnaire [*Fogel et al. 1966*], se révéla difficile à mettre en oeuvre dans le cadre de variables réelles.

La première approche vraiment adaptative, c'est-à-dire dans laquelle la décision est prise par rapport à la situation courante, fut la célèbre règle des 1/5 de Rechenberg [*Rechenberg 1973*]. La mutation considérée est isotrope. Son principe consiste à augmenter la valeur de l'écart type $\sigma$ de la mutation lorsque trop de mutations sont réussies, c'est-à-dire lorsque trop d'enfants ont une performance meilleure que celle des parents (trop d'exploitation), et réciproquement de diminuer la valeur de l'écart type $\sigma$ de lorsque pas assez de mutations sont réussies (trop d'exploration). La règle 1/5 s'utilise de la manière suivante : on commence par se fixer un temps d'observation T (correspondant à un nombre de générations), et toutes les T générations, on calcule le taux $\tau$ de mutations réussies. Si $\tau$ est supérieur à 0,2 alors $\sigma$ est augmenté d'un facteur 1,22 , sinon $\sigma$ est diminué d'un facteur 0,83.

Cette approche possède quelques faiblesses. Elle ne prend pas en compte les caractéristiques locales des performances du fait que la même valeur de $\sigma$ est utilisée pour toute la population et pour toutes les composantes du génotype. Pour pallier ce défaut, ainsi que pour se débarrasser élégamment de la tâche fastidieuse du réglage des paramètres de la mutation, Rechenberg [*Rechenberg 1973*] et Schwefel [*Schwefel 1981*] ont proposé de rendre "la mutation auto-adaptative" : chaque individu possède ses propres paramètres de mutation, qui sont eux-mêmes sujets à mutation, avant d'être utilisés pour la mutation des variables elles-mêmes.

Avec la mutation auto-adaptive, l'étape de remplacement sélectionne les individus avec leurs paramètres de mutation. Les individus qui survivront seront ceux qui auront à la fois les

bonnes valeurs pour les variables du problème (sur lesquelles se fait la sélection), mais aussi ceux qui auront les bonnes valeurs des paramètres de mutation, sinon ils seront immanquablement dépassés par d'autres, mieux adaptés aux caractéristiques locales du paysage de performance. De manière imagée, lorsque le gradient est fort dans une direction, il faut faire de "petites" mutations, et inversement. Avec cette méthode, on constate que :

– des mutations successives avec des paramètres de mutation aberrants ne peuvent pas être constamment réussies

– les individus qui survivent longtemps sont le fruit de nombreuses mutations successives réussies et doivent donc avoir de "bons" paramètres de mutation.

Il existe trois types de mutations auto-adaptatives :

– La mutation isotropie, dans laquelle chaque individu possède un scalaire $\sigma$ qui est utilisé pour l'ensemble des composantes du vecteur lors de la mutation. Plus précisément, la mutation de l'individu $(\vec{p},\sigma)$ s'effectue en deux temps : on commence par faire muter l'écarte type $\sigma$ suivant une loi log-normale (pour des raisons de symétrie multiplicative autour de 1), puis par faire muter des variables $p_i$, en utilisant la nouvelle valeur de $\sigma$.

$$\begin{aligned} &\sigma' = \sigma.exp(\tau\, N(0,1)) \\ &\forall i \qquad p'_i = p_i + N(0,\sigma) \end{aligned} \tag{1.19}$$

où $\tau$ est un paramètre de la SE.

– La mutation anisotrope, dans laquelle les paramètres de la mutation sont un vecteur de valeurs qui représente les écarts types dans chacune des directions canoniques. La mutation s'effectue aussi en deux temps,

$$\begin{aligned} &\forall i \qquad \sigma'_i = \sigma_i.exp(\tau\, N(0,1) + \tau' N_i(0,1)) \\ &\forall i \qquad p'_i = p_i + N(0,\sigma_i) \end{aligned} \tag{1.20}$$

où $\tau$ et $\tau$' sont aussi des paramètres (de second ordre) de la SE.

– Les mutations corrélées dans lesquelles les paramètres de la mutation sont une matrice de covariance complète, et que nous ne détaillerons pas ici, le lecteur peut se référer à [*Auger 2004*] pour plus de détails.

Les valeurs préconisées par Schwefel [*Schwefel 1981*], basées sur des études théoriques de la fonction sphère en grande dimension, sont les suivantes :

$$\tau \propto \frac{1}{\sqrt{2n}} \qquad \tau' \propto \frac{1}{\sqrt{2\sqrt{n}}} \tag{1.21}$$

## I.3.2. Méthodes de surface de réponse

Le plus souvent les méthodes de surface de réponse sont des méthodes d'optimisation basées sur les plans d'expériences. A l'origine, les plans d'expériences sont créés pour s'appliquer à l'expérimentation. Grâce à eux, l'expérimentateur peut répondre aux questions « comment sélectionner les expériences à faire, quelle est la meilleure stratégie » pour :

- Aboutir le plus rapidement possible aux résultats espérés avec une bonne précision, en évitant des expériences inutiles
- Conduire à la modélisation et à l'optimisation des phénomènes étudiés

Une littérature abondante existe sur les plans d'expériences, mais dans le cas d'expérimentation numérique, tous les aspects liés aux erreurs de mesure sont sans objet.

Dans le domaine d'optimisation numérique, un plan d'expériences peut être utilisé comme un support ou une étape préliminaire à l'optimisation par les méthodes de surface de réponse. Grâce au plan d'expériences, nous pouvons éviter les évaluations de la fonction coût inutiles et économiser le temps de résolution.

Le principe des méthodes d'optimisation par surface de réponse consiste à remplacer la résolution du problème d'optimisation réel par celle de problèmes approchés. Le schéma général de la résolution du problème d'optimisation par les méthodes de surface de réponse est présenté sur la *Figure 1.7*.

Grâce à une base de données composée de plusieurs points (solutions) déjà évalués, on approxime la fonction Φ (ainsi que les contraintes et le gradient) sur l'espace des paramètres d'optimisation par des fonctions mathématiques. Ces approximations conduisent donc au problème d'optimisation approché (1.22):

$$\begin{cases} Minimiser\, \Phi^{appr}(p) \\ c_i^{appr}(p) \le 0 \quad \forall i = 1,...,m_i \\ h_i^{appr}(p) = 0 \quad \forall i = 1,...,m_e \\ p \in \Re^n \end{cases} \tag{1.22}$$

Le problème d'optimisation approché (1.22) pourrait être résolu avec tous les types d'algorithmes d'optimisation comme ceux à direction de descente, les algorithmes évolutionnaires, les algorithmes hybrides, etc. Quelque soit l'algorithme retenu, le temps de la résolution du problème (1.22) est négligeable devant le celui de la résolution du problème réel (en une milliseconde, les ordinateurs peuvent réaliser facilement mille évaluations de la fonction coût ainsi que des contraintes et le gradient du problème approché). La création de la base de données initiale ou l'échantillonnage (Design of Experiments – DOE en anglais) et la méthode pour approximer la fonction objectif sont les éléments principaux qui caractérisent les méthodes de surface de réponse. La qualité des solutions obtenues est en grande partie fonction de ces éléments. Une description détaillée des plans d'expériences et les méthodes d'approximation est décrite dans l'ouvrage [*Myers et al. 2002*].

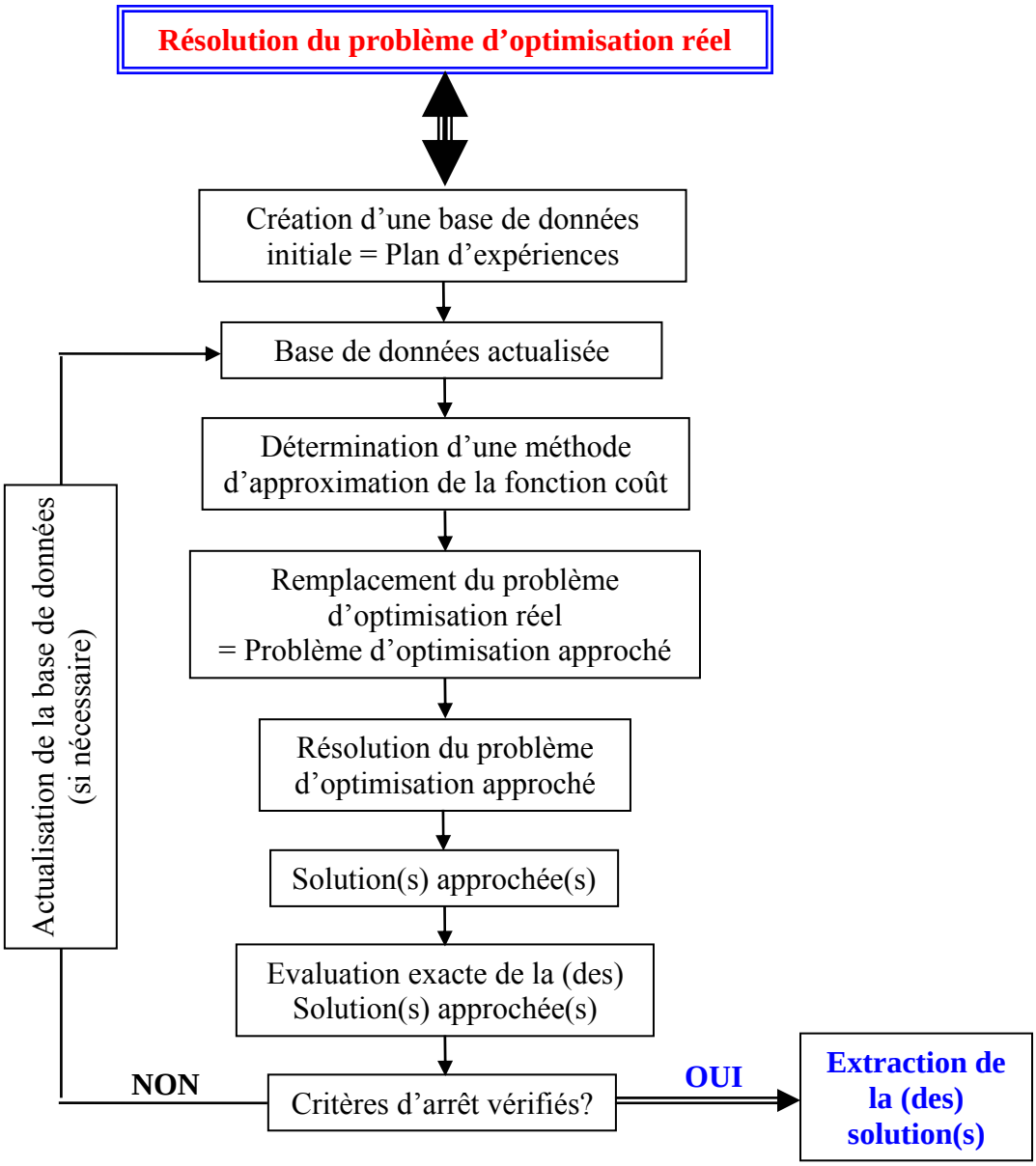



*Figure 1.7. Résolution d'un problème d'optimisation par la méthode des surfaces de réponse*

Dans le domaine de la mise en forme, les méthodes de surface de réponse ont été employées pour résoudre différents problèmes d'optimisation.

Pour l'optimisation des procédés de mise en forme des métaux, *Bonte et al.* [*Bonte et al. 2005a*], [*Bonte et al. 2005b*] et [*Bonte 2005*] ont utilisé une méthode de surface de réponse et ont obtenu des résultats très satisfaisants. A chaque itération d'optimisation, *Bonte et al.* ont créé une nouvelle base de données en utilisant la méthode d'échantillonnage "Latin Hypercube Design" (LHD) [*Santner et al. 2003*] [*Martin et al. 2004*] et le plan factoriel complet. La fonction coût (ainsi que les contraintes et les gradients) est approximée grâce à cette base de données. Pour faire cela, ils construisent sept méta-modèles à base de différentes régressions polynomiales et de la méthode de krigeage. Puis, un test de précision est effectué pour ces méta-modèles afin d'identifier le meilleur. Une fois que l'approximation de fonction coût réalisée, ils ont utilisé l'algorithme SQP (Sequential Quadratique Programming) [*Haftka et al. 1992*] pour obtenir la solution approchée optimale. Afin d'éviter le fait de tomber dans un

optimum local, ils ont choisi de lancer l'algorithme SQP à partir de chaque point de la base de données. Ils ont pris la meilleure solution approchée parmi celles obtenues par les différentes optimisations. Puis une autre évaluation exacte devait être réalisée pour obtenir la réponse réelle du problème à l'itération d'optimisation. Si la solution optimisée est satisfaisante, ils s'arrêtent. Sinon, ils recommencent l'algorithme. Cependant, à l'étape échantillonnage, ils créent un nouveau plan d'expérience en tenant compte de l'existence des points déjà évalués aux itérations précédentes pour avoir une approximation plus précise.

Une approche similaire à celle présentée précédemment est utilisée par [*Beauchesne et al. 2005*] pour faire l'optimisation du procédé d'hydroformage. Cependant, au lieu de 7 métamodèles différents, seule l'approximation à base de krigeage non linéaire est utilisée pour approcher la fonction coût.

Ayad at al. [*Ayad at al. 2005*] ont aussi utilisé la méthode des surfaces de réponse pour l'optimisation du procédé de ségrégation de poudre lors du moulage par injection métallique. La stratégie utilisée consiste en 3 étapes principales : d'abord utiliser un plan d'expérience de type Taguchi [*Sado et al. 1991*] pour déterminer les paramètres de comportement du matériau et du procédé les plus importants afin de diminuer la dimension du problème d'optimisation. Ensuite, ils construisent une approximation basée sur la méthode des moindres carrés mobiles [*Belytschko et al. 1996*], puis s'en servent pour chercher la solution optimale approchée du problème grâce à un algorithme génétique. Enfin, afin d'améliorer la recherche de l'optimum, et de localiser correctement celui-ci, ils ont développé une méthode adaptative de type moindres carrés mobiles en raffinant l'espace de recherche.

Une approche similaire à celle de Ayad et al. est utilisé par Ben Ayed et al. [*Ben Ayed et al. 2005*] pour l'optimisation des efforts de serre-flan en emboutissage. La méthode des moindres carrés mobiles est utilisée pour approximer la valeur de fonction coût. Un algorithme d'optimisation de type SQP est ensuite utilisé pour résoudre le problème approximé à partir de plusieurs points de départ pour trouver l'optimum global du problème.

Naceur et al. [*Naceur et al. 2004*] ont utilisé une méthode de surface de réponse à base d'approximation diffuse [*Nayroles et al. 1991*] pour faire l'optimisation du procédé d'emboutissage, suivant une approche semblable.

Pour l'optimisation du procédé de pliage (créer les pièces de sécurité), Bahloul et al. [*Bahloul et al. 2005*] ont tenté d'utiliser la méthode d'approximation de type « réseau de neurones » combinée avec un algorithme évolutionnaire. Un plan d'expériences de Taguchi est utilisé pour l'étape d'échantillonnage et pour sélectionner les paramètres les plus importants du problème d'optimisation.

## I.3.3. Autres méthodes d'optimisation d'ordre 0

### I.3.3.1. Méthodes de recherche aléatoire/probabiliste

La méthode de recherche aléatoire consiste à tirer aléatoirement, à chaque itération un point dans l'espace de recherche. La valeur de fonction objectif $\Phi$ est ensuite évaluée en ce point et

comparée à celle du point de départ. Si elle est meilleure, cette valeur est enregistrée, ainsi que la solution correspondante, et le processus continue. Sinon on repart du point de départ et on recommence le procédé, jusqu'à ce que les conditions d'arrêt soient atteintes. Le grand avantage de cette méthode est sa simplicité. Le temps de calcul en constitue une grande faiblesse.

#### I.3.3.2. Méthodes du simplex

Cette méthode d'ordre 0 déterministe a été introduite par Nelder et Mead [*Nelder et al. 1965*].

Supposons que la fonction coût $\Phi$ ait $n$ paramètres. On définit un simplex comme étant une figure géométrique (polygone, triangles, etc.) de volume non nul contenant (n+1) sommets. Donc, à chaque itération de l'algorithme du simplex, ($n$+1) points sont utilisés pour déterminer un pas d'essai. Les points $p_i$ sont ordonnés de manière à avoir $\Phi(p_1) \leq \Phi(p_2) \leq .... \leq \Phi(p_{n+1})$. Des nouveaux points sont obtenus en utilisant de très simples opérations algébriques, qui se traduisent par des transformations géométriques élémentaires (réflexion, contraction, expansion, et multicontraction appelée aussi rétrécissement), et ces points sont acceptés ou rejetés en fonction de leur valeur de la fonction objectif. Le simplex se transforme, il s'étend, se contracte, à chaque mouvement. Ainsi il s'adapte à l'allure de la fonction, jusqu'à ce qu'il s'approche de l'optimum. A chaque transformation, le plus mauvais point courant $x_i$ est remplacé par le nouveau point déterminé.

La méthode du simplex n'utilise que des valeurs ponctuelles de la fonction coût et ne nécessite pas l'estimation du gradient. Cette méthode peut donc être utilisée pour la recherche du minimum d'une fonction coût non-différentiable. Elle semble efficace tant que le nombre de paramètres est petit [*Kusiak 1989*]. Lorsque le nombre de paramètres est supérieur à trois, elle semble mal adaptée du point de vue du coût, et devient moins intéressante que les méthodes à direction de descente [*Nouatin 2000*] [*Vielledent 1999*] [*Kusiak 1989*].

Ohata et al. [*Ohata et al. 1998*] ont utilisé cet algorithme pour résoudre un problème d'optimisation d'un procédé de mise en forme 3D de plaques, en deux opérations et avec trois paramètres à optimiser. Coupez et al. [*Coupez et al. 1999*] l'ont utilisé pour l'optimisation du profil du champ de vitesse dans un procédé d'injection 3D.

Nakamashi et al. [*Nakamashi et al. 1998*] ont proposé une étude comparative entre la méthode du simplex et la méthode heuristique du recuit simulé [*Aarts 1989*], sur le même cas que Ohata et al., mais pour sept paramètres. De plus, il est à noter que, même si généralement l'algorithme fonctionne bien, il existe des cas où la méthode ne converge pas. Des exemples de stagnation en des points non-stationnaires ont en effet été décrits dans [*Mc Kinnon 1998*] dans des cas de minimisation de fonctions strictement convexes.

#### I.3.3.3. Méthode du recuit simulé

Cette méthode d'optimisation a été mise au point en 1983 par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi [*Kirkpatrick et al. 1983*].

Le recuit simulé est une méthode d'optimisation stochastique tirant son origine d'un processus thermodynamique. Cette méthode est issue d'une analogie avec le phénomène physique de refroidissement lent d'un corps en fusion, qui le conduit à un état solide de basse énergie. Il faut abaisser lentement la température, en marquant des paliers suffisamment longs, pour que le corps atteigne l'équilibre thermodynamique à chaque palier de température. Pour les matériaux, cette basse énergie se manifeste par l'obtention d'une structure régulière, comme les cristaux dans l'acier. L'analogie exploitée par le recuit simulé consiste à considérer la fonction $\Phi$ à minimiser comme fonction d'énergie, et une solution $p$ peut être considérée comme un état donné de la matière dont $\Phi(p)$ est l'énergie. Le recuit simulé exploite généralement le critère défini par l'algorithme de Metropolis et al. [*Metropolis et al. 1953*] pour l'acceptation d'une solution obtenue par perturbation de la solution courante.

Des études théoriques du recuit simulé ont pu montrer que sous certaines conditions, l'algorithme du recuit convergeait vers un optimum global. Ce résultat est important car il nous assure que le recuit simulé peut trouver la meilleure solution, si on le laisse chercher indéfiniment. Les principaux inconvénients du recuit simulé résident dans le choix des nombreux paramètres, tels que la température initiale, la loi de décroissance de la température, les critères d'arrêt ou la longueur des paliers de température. Ces paramètres sont souvent choisis de manière empirique.

#### I.3.3.4. Algorithmes de colonie des fourmis

Le comportement des insectes sociaux est caractérisé par l'auto-organisation. Les individus communiquent en changeant les propriétés locales de leur environnement, et, par le biais de ce moyen de communication limité, une sorte d'intelligence collective émerge. Marco Dorigo [*Dorigo et al. 1996*] de l'Université Libre de Bruxelles a inventé l'algorithme à colonies de fourmis lorsqu'il a observé des fourmis dans leur chemin de recherche de la nourriture. Celles-ci ont la capacité de trouver le chemin le plus court entre leur nid et une source de nourriture, en contournant les obstacles qui jonchent leur chemin.

L'idée générale de l'algorithme de colonie de fourmis est d'imiter le comportement coopératif d'une colonie de fourmis naturelles à l'aide des fourmis artificielles se déplaçant à travers le graphe qui représente le problème à résoudre. Le principe est le suivant : les fourmis cherchent de la nourriture et se déplacent de façon quasi aléatoire. Tout au long de leur déplacement, elles laissent derrière elles une substance chimique appelée phéromone. Cette substance a la propriété de s'évaporer au cours du temps et a pour but de guider les fourmis vers leur objectif. Une fois cet objectif atteint (dans notre cas, la nourriture trouvée), les fourmis rentrent au nid en empruntant le même chemin qu'à l'aller, grâce à leur trace de phéromone. Celle-ci s'en trouve renforcée. Plus une trace de phéromone est concentrée, plus elle va attirer les fourmis. Au fil du temps, on va donc constater l'émergence du plus court chemin vers la nourriture grâce au renforcement de la trace de phéromone.

Dans la nature, nous pouvons remarquer que les fourmis se déplacent "en ligne", suivant le chemin des fourmis précédentes. Pour mieux comprendre le phénomène et l'algorithme, faisons une simple expérience en plaçant maintenant un obstacle sur cette ligne, de manière à ce que le contournement de l'obstacle par un côté soit nettement plus court que par l'autre

(*Figure 1.8*). Au bout d'un certain temps nous observons que toutes les fourmis contourneront l'obstacle par le côté le plus court.

Le phénomène s'explique de la manière suivante : les fourmis déposent en marchant des marqueurs chimiques appelés phéromones. Les autres fourmis suivent le chemin tracé par ces phéromones. Plus un chemin est marqué, plus elles ont tendance à le suivre. Lorsque l'on pose l'obstacle, le chemin de phéromones est coupé, et les fourmis choisissent au hasard l'un ou l'autre côté pour contourner l'obstacle. Comme un chemin est plus court, plus de fourmis auront franchi l'obstacle en passant par ce côté. Par exemple, sur 6 fourmis, 3 fourmis pourront être passées par le côté court alors que les 3 fourmis étant passées par le côté long n'auront pas encore fini de contourner l'obstacle. Une fourmi arrivant en sens inverse verra donc plus de phéromone sur le chemin qui passe par le côté court, et prendra donc ce chemin.

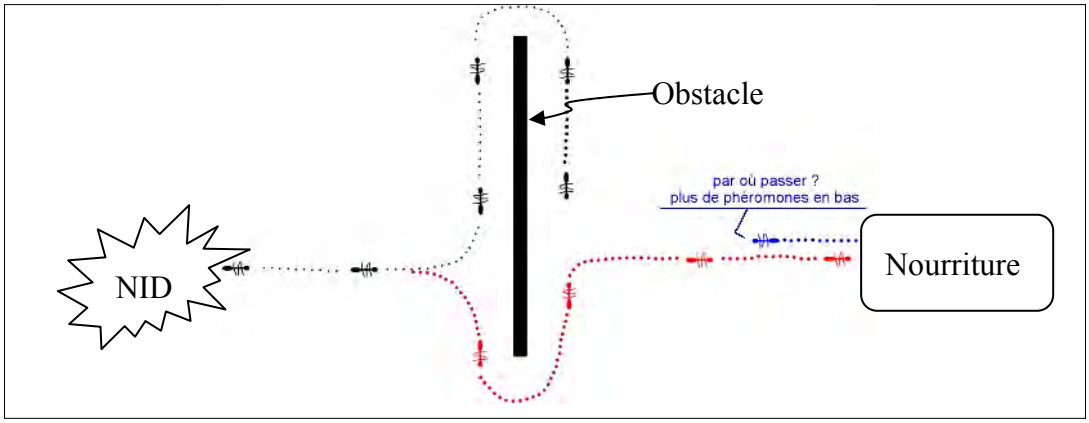



*Figure 1.8. Comportement des fourmis lors de franchir l'obstacle*

L'algorithme de colonies de fourmis a été à l'origine principalement utilisé pour produire des solutions quasi-optimales au problème du voyageur de commerce, puis, plus généralement, aux problèmes d'optimisation combinatoire. On observe depuis ses débuts que son emploi se généralise à plusieurs domaines, depuis l'optimisation continue jusqu'à la classification ou encore le traitement d'image. On peut trouver une liste des applications de plusieurs variantes de cet algorithme à la page 51 de la thèse de Roux [*Roux 2001*].

## I.4. METHODES HYBRIDES

L'hybridation des algorithmes est pour objectif de mélanger de manière harmonieuse deux ou plusieurs méthodes distinctes afin de ne retenir que les caractéristiques les plus intéressantes de chacune de ces méthodes.

L'approche d'hybridation la plus connue est celle entre un algorithme évolutionnaire et un algorithme à direction de descente [*Oulladji et al. 2003*]. Le principe de cette approche d'hybridation est assez simple. Il consiste à lancer une recherche au niveau global avec un AE, puis passer à la recherche locale avec un algorithme à direction de descente pour affiner le résultat. Cela nécessite donc d'effectuer une répartition des tâches. L'AE se charge de détecter les régions de l'espace de recherche qui sont susceptibles de se révéler les plus intéressantes. Puis, l'algorithme à direction de descente prend comme point de départ les

meilleures solutions trouvées par l'AE, et s'attache à les affiner aussi rapidement possible. Pourtant, il est délicat de décider le moment de transition ou à partir duquel l'algorithme à direction de descente doit prendre le relais et faire son travail. En effet, si cela se fait trop tôt, il y a de fortes chances pour que l'algorithme touche sa fin par la convergence vers un optimum local. Au contraire, si la transition se produit trop tard, on perd en temps de calcul car les avantages de l'algorithme à direction de descente ne sont pas pleinement exploités.

L'hybridation peut aussi être réalisée entre un AE et une méthode d'approximation. Avec cette version, la méthode d'approximation est utilisée pour accélérer la convergence de l'AE. L'idée est donc de remplacer la fonction objectif par une fonction approchée. Cette approximation peut utiliser l'information du gradient (comme les deux nouveaux algorithmes hybrides que nous avons développés) ou non (comme la SE avec Métamodèle de Emmerich et al. [*Emmerich et al. 2002*] ou les algorithmes proposé par Jin et al. [*Jin et al. 2000*] , [*Jin et al. 2001*], etc.). Avec cette approche, au sein d'une génération de l'AE, on peut avoir une partie des individus évalués avec la fonction objectif et l'autre partie avec la fonction approchée, ou bien toute la population évaluée par la fonction approchée.

Cette deuxième approche d'hybridation sera présentée par la description des deux nouveaux algorithmes hybrides que nous avons développés dans le chapitre II et celle de la SE avec Métamodèle (SE-Meta) dans le prochain paragraphe.

## I.5. ALGORITHMES D'OPTIMISATION EMPLOYES

Nous avons testés 5 différents algorithmes pour résoudre nos problèmes d'optimisation de forgeage : BFGS, SCPIP, la SE-Meta et les deux algorithmes d'optimisation que nous avons développé. Parmi les 5 algorithmes utilisés, nous ne revenons pas sur l'algorithme BFGS car il est tout à faire classique. *Les deux nouveaux algorithmes hybrides, qui sont notre contribution originale aux méthodes d'optimisation, font l'objet du chapitre qui suit*. Cette section présentera les deux algorithmes SCPIP et la SE-Meta.

### I.5.1. Algorithme SCPIP

Nous ne présentons pas les détails du SCPIP car il est assez complexe, nous présentons ici seulement son principe général de fonctionnement. L'algorithme complet est détaillé dans [*Zillober 2002*].

L'algorithme SCPIP [*Zillober 2002*] est un algorithme de type quasi Newton. Il est une combinaison de la méthode d'approximation convexe SCP (Sequential Convex Programming) avec la méthode de résolution par point intérieur (IP – Interior Point) pour chercher l'optimum global du problème (lorsque le problème est multi optima). Il a besoin du gradient de la fonction coût pour fonctionner. Construit sur la base de la méthode des asymptotes mobiles [*Svanberg 1987*], la méthode SCP remplace le problème d'optimisation original, difficile, par une série de sous-problèmes distincts, convexes et plus aisés à résoudre, et qui sont construits au cours des itérations :

- La fonction objectif et les contraintes d'inégalités sont remplacées par des approximations convexes,

- Les contraintes d'égalités sont linéarisées.

Ces sous-problèmes sont ensuite résolus par l'application de la méthode de point intérieur, qui est présentée dans [*Zillober 2001*].

Cet algorithme est considéré comme une méthode robuste pour résoudre des problèmes avec contraintes et fortement non-linéaires.

## I.5.2. Stratégie d'évolution avec Méta-modèle

La Stratégie d'Evolution avec Métamodèle (SE-Métamodèle) a été développée à l'université de Dortmund et nous a été proposé dans le cadre du projet COST 526 APOMAT. Elle est une stratégie d'évolution utilisant une fonction d'approximation (métamodèle) dans le but de diminuer le nombre d'évaluations exactes de la fonction coût. Elle utilise donc l'organigramme d'un algorithme évolutionnaire canonique avec deux différences mises en relief sur la *Figure 1.9* :

- au lieu de commencer avec une population initiale normale, il utilise la méthode d'échantillonnage aléatoire pour créer une base de données initiale qui contient *2*n* (n est le nombre de paramètres à optimiser) de points initiaux. Ces points initiaux donc sont choisis dans tout l'espace de recherche.

- il évalue exactement seulement *20%* la population d'enfants à chaque génération et enrichit la base de données en y ajoutant ces individus évalués.

Nous allons donc maintenant expliquer le métamodèle, ce qui est le point clé de cette méthode d'optimisation.

Le Méta-modèle utilisé est une forme de surface de réponse qui permet d'approcher les valeurs de la fonction coût aux points où elle n'a pas encore été évaluée, à partir des valeurs déjà calculées. Ce modèle contient également une estimation de l'erreur d'approximation. Il permet ainsi d'éviter l'évaluation exacte de 80% des individus, a priori les moins performants d'une population, en tenant compte cette erreur d'approximation.

Le métamodèle présenté dans cette section est celui basée sur la méthode de Krigeage, comme méthode d'interpolation spatiale. Le Krigeage porte le nom de son fondateur, l'ingénieur minier D.G. Krige [*Krige 1951*], qui a développé des méthodes statistiques empiriques afin de déterminer la distribution spatiale de minerai à partir d'un ensemble de forage. C'est cependant le français Matheron [*Matheron 1963*] qui a formalisé l'approche en utilisant les corrélations entre forages pour la répartition spatiale. Dans les années récentes, le Krigeage était utilisé en géostatistiques [*Wackernagel 1998*] et en optimisation [*El-Bektagy et al. 1999*], [*Ralte 1998*], [*Trosset et al. 1997*].

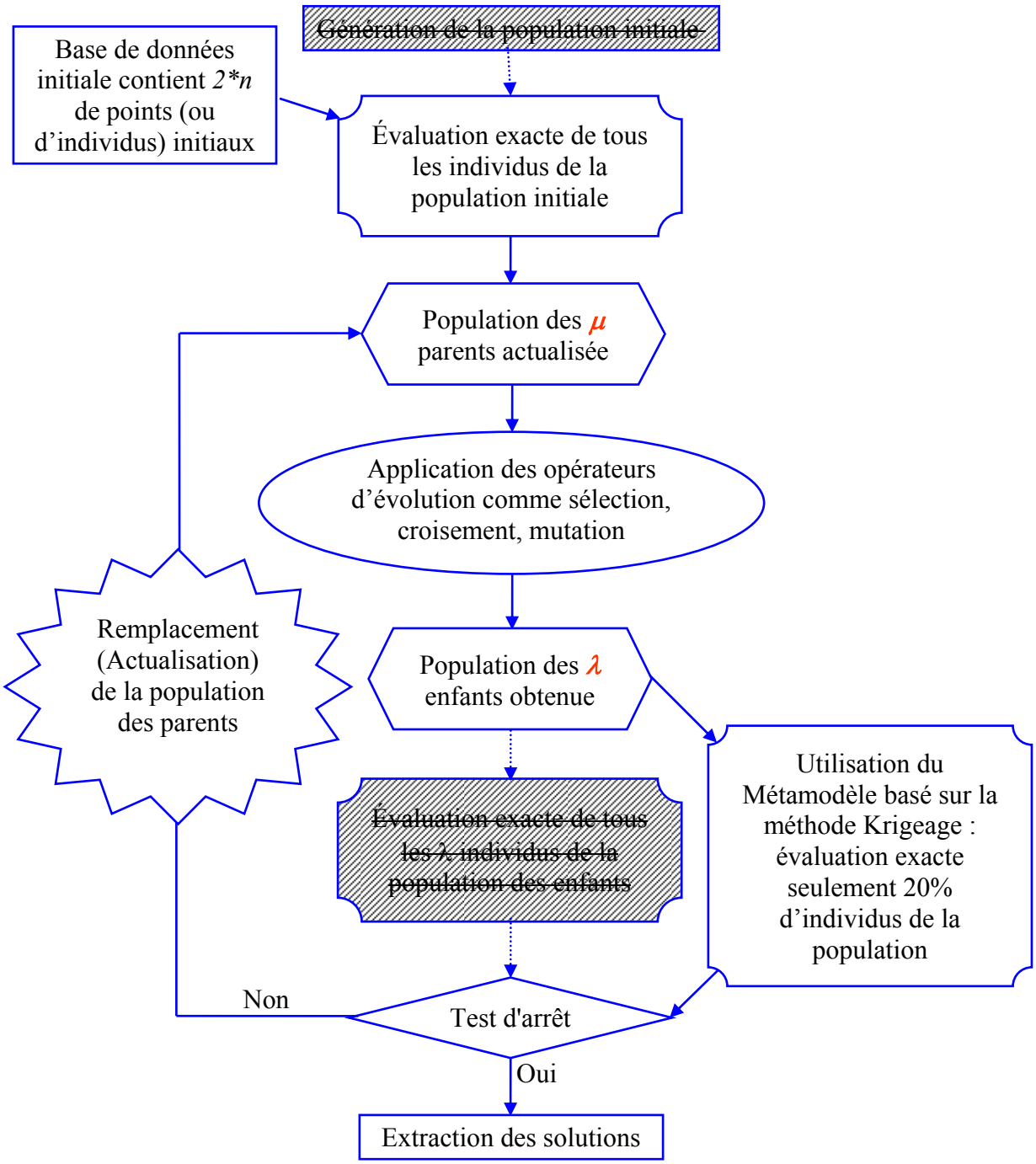



*Figure 1.9. Organigramme de la SE-Méta basée sur la méthode Krigeage*

Supposons que nous ayons une base de données composée de plusieurs points $x_1,...,x_m \in \Re^n$ déjà évalués et que $y = [y_1,...,y_m] = [f(x_1),...,f(x_m)]$ sont les résultats des évaluations exactes effectuées à chaque point.

En utilisant la méthode Krigeage, on approxime la valeur de la fonction objectif $f(x)$ d'un point $x$ de l'espace de recherche par la fonction approchée $\hat{f}(x)$. Cette fonction $\hat{f}(x)$ est une réalisation Gaussienne stochastique isotrope de moyenne inconnue $\beta$ et de covariance de forme :

$$c(u,v) = \sigma^2 r_\theta(u, v) \tag{1.23}$$

où $r_\theta(u, v)$ est la fonction de corrélation Gaussienne définie par $r_\theta(u, v) = e^{-\theta\|u-v\|^2}$ avec $\theta$ le paramètre gaussien inconnu, $u, v \in \Re^n$ et $\sigma^2$ la variance inconnue.

La construction de la fonction approchée à base du krigeage revient maintenant à l'estimation des trois inconnus $\beta$, $\sigma^2$ et $\theta$ [*Santner et al. 2003*]. L'idée de base de leur estimation est de minimiser l'erreur quadratique de prédiction suivante :

$$MSPE = E\left[\hat{f}(x) - f(x)\right]^2 \tag{1.24}$$

Cette idée ressemble à celle de minimiser l'erreur au sens des moindres carrés dans une méthode de surface de réponse. Il est courant d'utiliser la méthode de "*l'Estimation de la Probabilité Maximale (EPM)*" (Maximum Likelihood Estimation) pour estimer les inconnues $\beta$, $\sigma^2$ et $\theta$. Cette méthode repose sur la même hypothèse que celle de Krigeage, c'est-à-dire que les mesures expérimentales (les points déjà évalués) résultent d'un processus Gaussien stochastique. La maximisation de la fonction de probabilité par rapport aux inconnues $\beta$, $\sigma^2$ et $\theta$ donne la valeur de l'approximation $\hat{f}(x)$ en tout point *x,* qui est le "Meilleur Prédicteur Linéaire Objectif" (Best Linear Unbiased Predictor – BLUP) de la réponse $f(x)$ [*Martin et al. 2003*] [*Lophaven et al. 2002*] :

$$\hat{f}(x) = \hat{\beta} + \left(y - I\hat{\beta}\right)^T R\left(\hat{\theta}\right)^{-1} r\left(x;\hat{\theta}\right) \tag{1.25}$$

où $R\left(\hat{\theta}\right)$ est la matrice symétrique $m \times m$ de composante $R_{ij} = r_{\hat{\theta}}(x_i, x_j)$ et $r\left(x;\hat{\theta}\right)$ le vecteur dimension *m* de composantes $r_i = r_{\hat{\theta}}(x_i, x)$.

Il faut noter que $\beta$ et $R(\theta)$ ne dépendent que de $\theta$ et sont remplacés par leurs estimateurs $\hat{\beta}$ et $R\left(\hat{\theta}\right)$ (on suppose $\theta$ connu). On peut démontrer que $\hat{\beta}$ est l'estimation généralisée au sens des moindres carrés (Generalized Least Squares estimate) du coefficient de régression $\beta$ [*Lophaven et al. 2002*] [*Martin et al. 2003*] [*Santner et al. 2003*]. Il prend la forme suivante :

$$\hat{\beta} = \left[I^T R\left(\hat{\theta}\right) I\right]^{-1} I^T R\left(\hat{\theta}\right)^{-1} y \tag{1.26}$$

Similairement à l'estimation de $\beta$, on peut aussi utiliser la méthode EPM pour estimer la variance $\hat{\sigma}^2$, ce qui donne :

$$\hat{\sigma}^2\left(\hat{\theta}\right) = \frac{1}{m}\left[y - \hat{\beta}\right]^T R\left(\hat{\theta}\right)^{-1}\left[y - I\hat{\beta}\right] \tag{1.27}$$

Le paramètre gaussien $\hat{\theta}$ (valeur optimale de $\theta$) peut maintenant être estimée par la méthode EPM [*Martin et al. 2003*] en minimisant la quantité suivante :

$$\Pi = m\,log\left[\hat{\sigma}^2\left(\hat{\theta}\right)\right] + log\left[det\, R\left(\hat{\theta}\right)\right] \tag{1.28}$$

Une fois $\hat{\theta}$ connu, $\hat{\beta}$ et $\hat{\sigma}^2$ peuvent être calculés facilement en utilisant les équations (1.26) et (1.27). Maintenant, le BLUP $\hat{f}(x)$ de $f(x)$ est donné par l'équation (1.25). La valeur estimée $\hat{f}(x)$ est assortie d'une erreur (au sens des moindres carrés) [*Martin et al. 2003*] qui est estimée par :

$$\hat{MSE}(x) = \sigma^2 - \sigma^2 \begin{bmatrix} I & r(x;\hat{\theta})^T \end{bmatrix} \begin{bmatrix} 0 & I \\ I & R(\hat{\theta}) \end{bmatrix} \begin{bmatrix} I \\ r(x;\hat{\theta}) \end{bmatrix} \tag{1.29}$$

Cette erreur s'annule au cas où *x* se confond à un point déjà évalué. La *Figure 1.10* présente un exemple d'approximation $\hat{y} = \hat{f}(x)$ par la méthode de Krigeage en une dimension (pour un seul paramètre à optimiser). Dans cet exemple, la base de données contient 3 points déjà évalués.

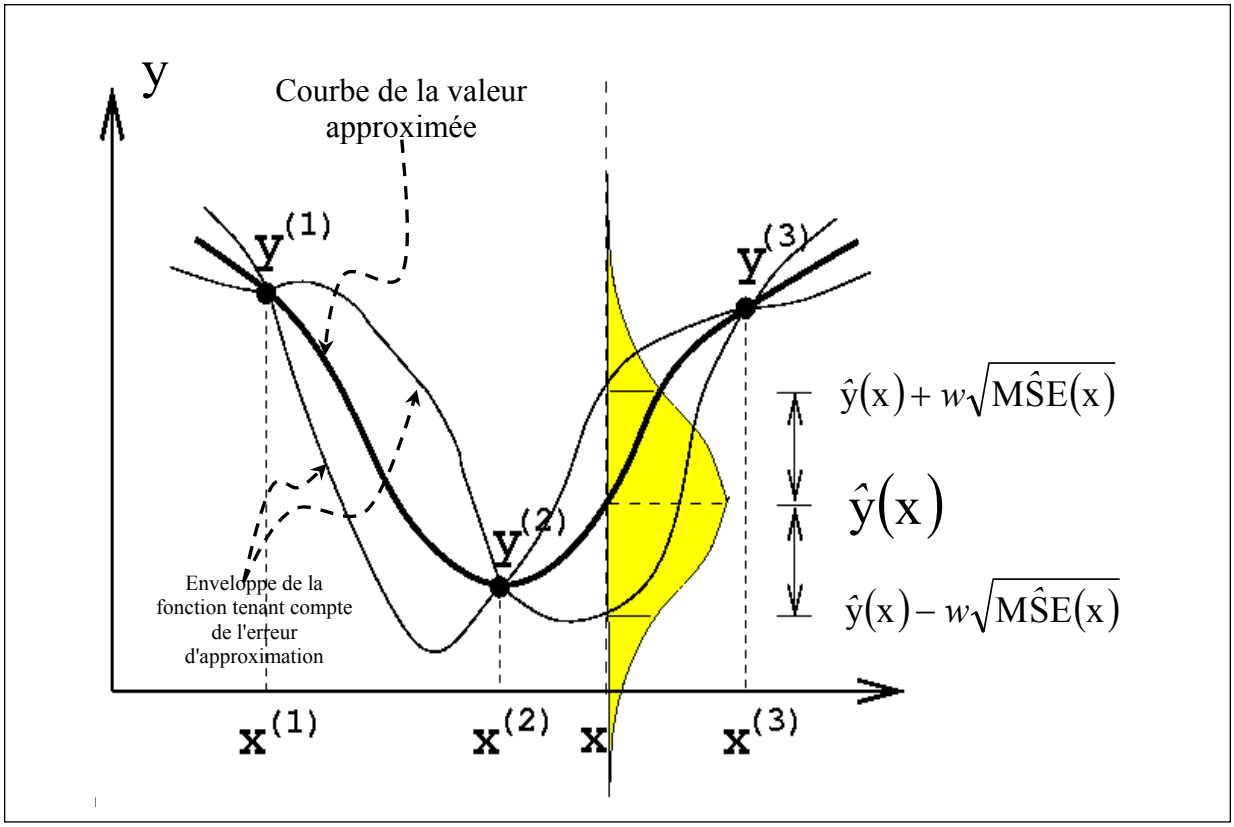



*Figure 1.10. Approximation par la méthode de Krigeage*

La précision de l'approximation dépend donc de la densité des points déjà évalués. Dans les zones avec une haute densité, l'approximation est plus précise et la valeur de l'erreur $\hat{MSE}$ est petite. Pour améliorer la qualité de l'approximation, de nouveaux points sont évalués et ajoutés à chaque génération à la base de données pour l'enrichir. La précision de l'approximation est alors augmentée, et les points choisis pour l'évaluation exacte ont plus de chance d'être les meilleurs individus de la génération.

Le choix des individus exactement évalués pour le méta modèle de krigeage est réalisé en utilisant l'expression (1.30). Ce critère est basé sur la valeur estimée de la fonction coût (1.25), corrigée par l'erreur d'approximation (1.29).

$$S_c(x) = \hat{f}(x) - w\sqrt{M\hat{S}E(x)} \qquad (1.30)$$

Le paramètre $w$ ($w > 0$) a pour but d'équilibrer l'influence du terme d'erreur d'approximation. Si le problème d'optimisation est supposé raide, une grande valeur de $w$ peut améliorer la capacité de localiser l'optimum global. Au contraire, une valeur basse de $w$ va conduire l'algorithme vers un optimum local plus rapidement. La valeur de $w$ = 1,0 est utilisée pour la SE-Meta dans cette étude.

Il existe une différence entre l'utilisation de l'approximation simple de la fonction coût (1.25) et celle basée sur le critère (1.30). Avec $\hat{f}(x)$, seuls les candidats les plus prometteurs sont choisis. Avec le critère (1.30) le choix est étendu pour explorer les zones d'améliorations potentielles pas encore explorées, en prenant compte l'erreur d'approximation, tout en examinant les individus ayant un potentiel de bonnes performances. La *Figure 1.11* présente un exemple, en une dimension, de la règle de sélection des individus ayant le meilleur potentiel avec le critère (1.30). Seuls les 20% des individus ayant la valeur de $S_c(x) = \hat{f}(x) - w\sqrt{M\hat{S}E(x)}$ la plus basse sont choisis.

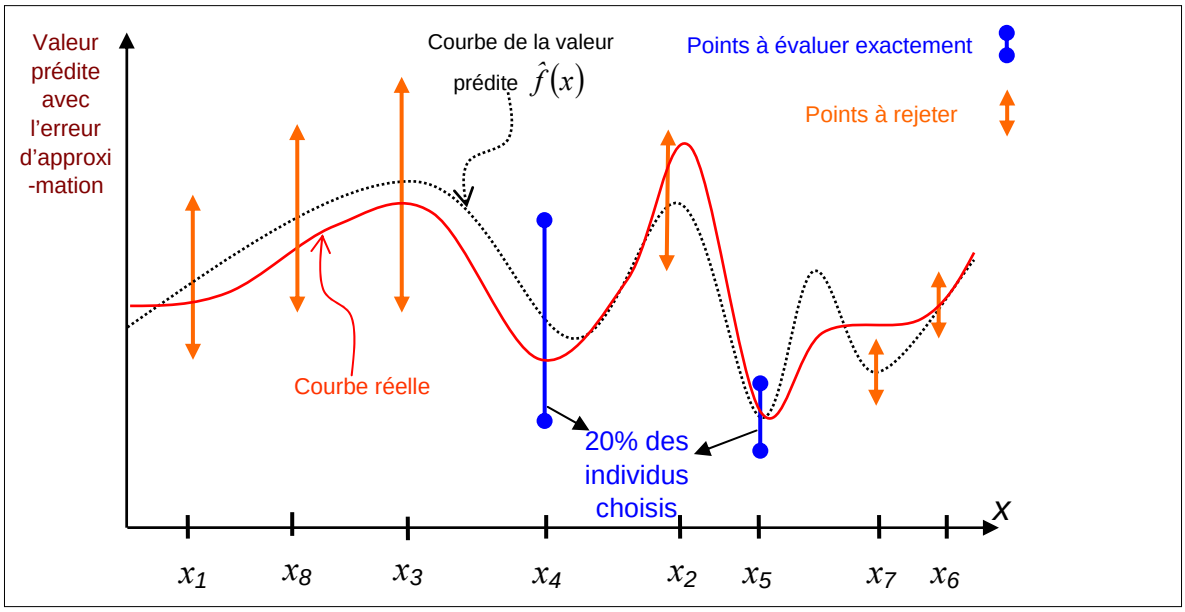



*Figure 1.11. Règle de sélection des individus ayant le meilleur potentiel, utilisée par le Métamodèle*

Le métamodèle de krigeage utilisé dans ce travail est basé sur les $k$ voisins les plus proches de chaque point. La valeur de k varie entre 15 et 20. L'augmentation de k améliore légèrement l'approximation, mais aussi le temps de calcul des approximations.

Cette SE-Meta a été donc appliquée avec succès pour résoudre plusieurs problèmes d'optimisation difficiles comme l'optimisation d'ailes d'avions [*Emmerich et al 2002*]. Dans le

cadre de ce travail, nous l'utilisons pour résoudre des problèmes d'optimisation de forme du forgeage 3D.

## I.6. CONCLUSION

Nous avons présenté dans ce chapitre une vue générale sur les différents algorithmes d'optimisation existants. Dans le chapitre qui suit, nous allons décrire les deux nouveaux algorithmes hybrides que nous avons développés pour l'optimisation globale.

# Chapitre II

# NOUVEAUX ALGORITHMES HYBRIDES POUR L'OPTIMISATION GLOBALE

## II.1. INTRODUCTION

Ce chapitre constitue l'une des contributions principales de ce travail de thèse. Il présente une nouvelle approche d'hybridation pour construire des algorithmes d'optimisation plus efficaces et plus robustes.

L'idée de cette approche est la suivante : on laisse un AE diriger le processus global de minimisation, en faisant évoluer des générations successives d'individus. Mais, lors de chaque évaluation de génération, des individus particuliers, dits maîtres, sont judicieusement sélectionnés pour y calculer le coût exact et son gradient. Ceci suppose bien sûr que l'on puisse disposer de routines de calcul du coût et du gradient, dont on se sert avec parcimonie. Ces calculs, coûteux dans le contexte habituel de la mécanique numérique, sont faits pour un faible nombre d'individus – maîtres – et servent à fabriquer une approximation qui sera utilisée pour évaluer l'ensemble des individus restants de la génération courante.

L'utilisation d'approximation de fonctions objectifs n'est évidement pas une nouvelle idée, les surfaces de réponse ou les réseaux neuronaux, largement répandus en optimisation, en sont une bonne illustration. L'originalité de notre contribution vient du fait que l'on utilise le gradient pour enrichir l'approximation et des techniques de classification – agglomérat habituelles en traitement statistique de données, pour identifier les meilleurs individus maîtres, pour lesquels le coût exact et le gradient sont calculés.

Le fruit de la nouvelle approche hybride proposée consiste en deux algorithmes hybrides que nous appelons respectivement *Algorithme Génétique avec Métamodèle utilisant le Gradient et basé sur l'Agglomération (AG-MGA) ou basé sur l'interpolation de Liszka-Orkisz (AG-MGO).*

## II.2. ALGORITHME GENETIQUE AVEC METAMODELE UTILISANT LE GRADIENT ET BASE SUR L'AGGLOMERATION (AG-MGA)

Comme SE-Meta, les deux algorithmes "l'AG-MGA" et "l'AG-MGO" utilisent aussi l'organigramme d'un AE canonique. La seule modification apportée réside dans l'étape d'évaluation des individus de la population, comme montrée sur la *Figure* 1.1.


*Figure 1.1. Organigramme d'un algorithme évolutionnaire avec Métamodèle*

A chaque génération, au lieu d'évaluer exactement les individus de la population, l'AG-MGA et l'AG-MGO les approximent en utilisant des Méta-modèles. Ce sont ces Méta-modèles qui entraînent la différence entre ces deux algorithmes.

Dans cette section, nous expliquons d'abord le fonctionnement du Méta-modèle de l'AG-MGA, qui est basé sur une méthode d'agglomération et une interpolation discontinue d'ordre 1 donnée.

Considérons une population qui évolue suivant un AE (cette étude utilise celui nommé Pikaïa [*Charbonneau 2002*]). Pour faciliter la compréhension, nous allons présenter le méta-modèle à travers l'exemple d'un problème d'optimisation à 2 paramètres ($\mu_1$, $\mu_2$). A une génération $g$, supposons qu'on a une population $P_g$ de $N_{ind}$ individus distribués dans l'espace de recherche comme présentés sur la *Figure 1.2*.

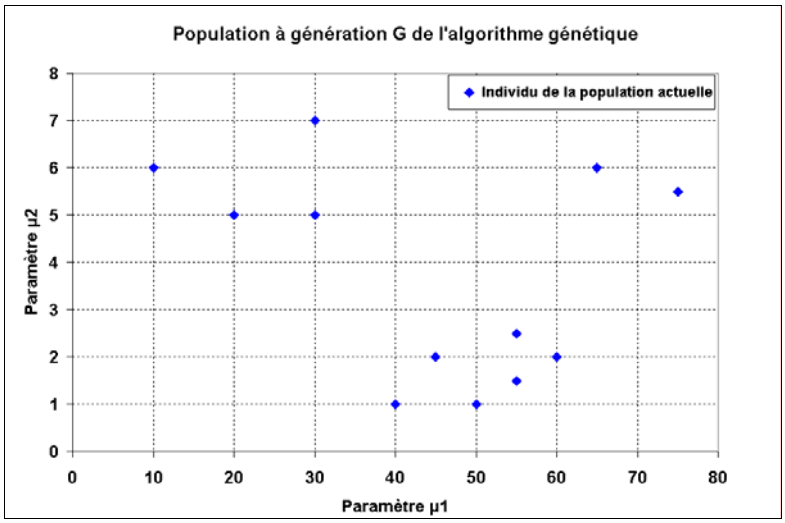



*Figure 1.2. Population exemplaire dans l'espace de recherche à 2 paramètres à la génération g*

Ce méta-modèle consiste en trois étapes principales :

- D'abord, il applique un algorithme pour regrouper la population courante en un nombre prédéfini *nbpeva* de sous populations (agglomérats) contenant chacune les "voisins les plus proches entre eux". Le nombre *nbpeva* représente le nombre d'évaluations exactes autorisées à chaque génération. Un algorithme très simple et largement répandu pour réaliser une telle partition est celui des K-moyennes :

  1) choisir *nbpeva* points initiaux distincts quelconques dans la population $P_g$ comme points maîtres initiaux

  $$x_M^k \in P_g \quad \text{pour} \quad k = 1,...,nbpeva$$

  2) initialiser les agglomérats (ou les sous populations) $C_k \subset P_g$ en associant à chaque point $x_M^k$ les points restants de la population qui sont

plus proches (en distance euclidienne) de lui que de tous les autres points maîtres $x_M^j$ , $j \neq k$

3) calculer les nouveaux barycentres de tous les agglomérats :

pour $k = 1,...,nbpeva$ $\qquad x^k_{M,nouveau} = \frac{1}{c_k}\sum_{j=1}^{c_k} x_j$

où $c_k$ est le nombre d'individus appartenants à l'agglonérat $C_k$

4) recommencer les étapes 2 et 3 jusqu'à la stationnarité.

La *Figure 1.3* montre un exemple du regroupement de la population précédente en 3 agglomérats.

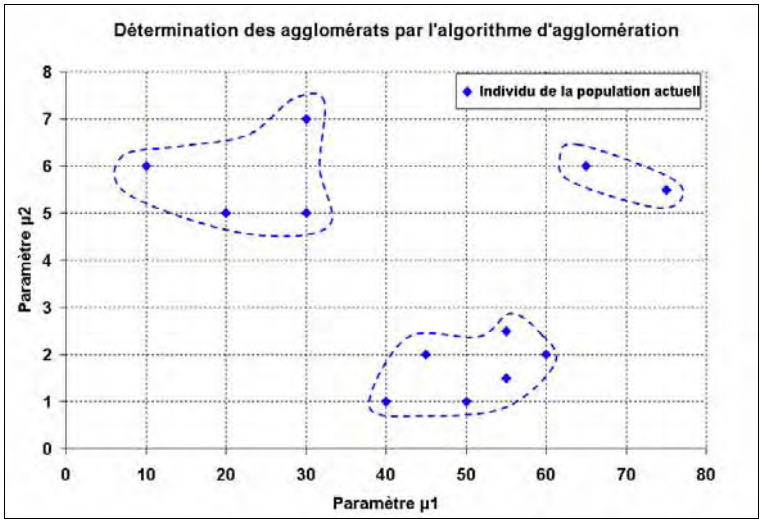



*Figure 1.3. Regroupement des individus dans les agglomérats*

- Ensuite, la valeur de la fonction coût Φ et de son gradient sont calculés exactement aux points maîtres, les centres de gravité de ces agglomérats (les points ronds en rose sur la *Figure 1.4*). Il faut noter que ces points ne sont pas des individus de la population considérée.

- Enfin, la valeur de tous les individus de la population est approximée par une interpolation linéaire locale obtenue (voir *Figure 1.5* pour une représentation 1D) selon :

  1) Pour chaque individu *i*, identifier l'agglomérat $C_j$ auquel il appartient

  2) Faire l'approximation $P_1$ d'ordre 1 en $X_i$ suivante :

$$\tilde{\Phi}(X_i) \approx \Phi(X_j) + \nabla\Phi(X_j)(X_i - X_j) \qquad (2.1)$$

  Cette approximation P1 étant différente pour chaque agglomérat, elle est globalement discontinue sur l'ensemble du domaine.

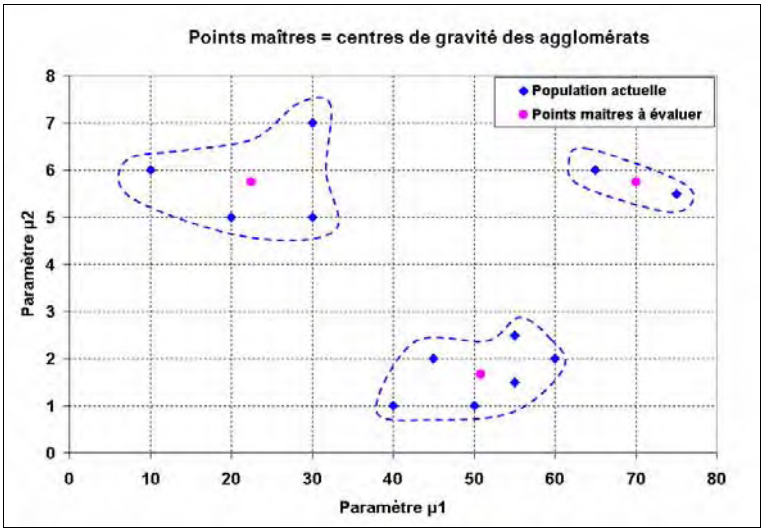



*Figure 1.4. Points maîtres et leurs agglomérats*

Une fois tous les individus évalués, l'AE se charge de faire évoluer la génération courante vers la suivante.

Avec ce méta-modèle, quelque soit la taille de la population, la fonction coût n'est évaluée qu'en un nombre fixe et réduit de points. Cela nous permet donc de travailler avec des populations de très grandes tailles à des coûts maîtrisés et ainsi de mieux explorer l'espace de recherche.

## II.3. ALGORITHME GENETIQUE AVEC METAMODELE UTILISANT LE GRADIENT ET BASE SUR L'INTERPOLATION DE LISZKA-ORKISZ (AG-MGO)

L'algorithme AG-MGA que nous venons de décrire présente deux inconvénients: une interpolation discontinue et un méta-modèle sans mémoire. En effet,

- Il utilise une interpolation discontinue d'un agglomérat à l'autre. Ainsi, un individu situé à mi-chemin entre deux points maîtres peut prendre deux valeurs très différentes. Cependant, durant la convergence de l'AE, de tels individus ont de moins en moins de chance d'exister car les individus s'agglutinent autour des barycentres. La *Figure 1.5* présente cet inconvénient pour une interpolation en 1D : le problèmes est qu'il y a un saut sur l'évaluation de la fonction coût qui pourrait perturber la convergence de l'AE et donc de la méthode (en tous cas, la ralentir).

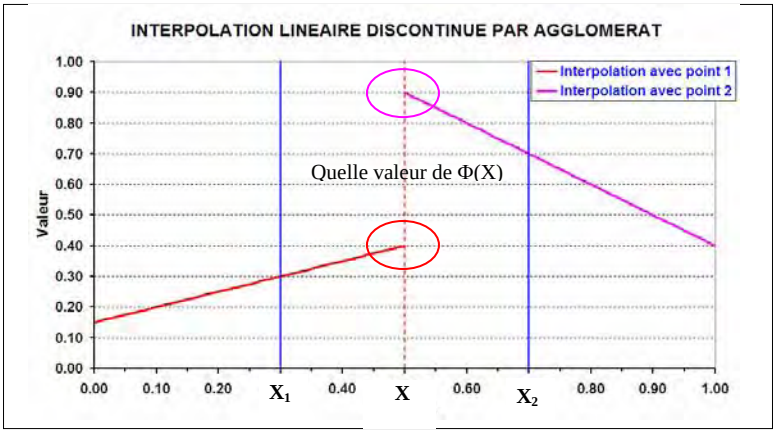



*Figure 1.5. Interpolation linéaire discontinue par agglomérat : quelle valeur le point prend–t–il à mi-chemin ?*

- Le second inconvénient vient du fait que l'AG-MGA ne sert pas des calculs des générations précédentes. Ces informations peuvent être précieuses pour améliorer la qualité des approximations dans le cas où la population ne déplace pas de manière importante entre chaque génération. Lorsque par exemple, d'une génération à l'autre, les agglomérats sont disjoints, les barycentres de la génération précédente (les seuls évalués exactement) ne sont plus pertinents pour une bonne évaluation P1 locale à l'intérieur des nouveaux agglomérats.

Pour pallier à ces deux défauts, nous avons introduit l'approche *Algorithme Génétique avec Métamodèle utilisant le Gradient basé sur l'interpolation de Liszka-Orkisz (AG-MGO)* avec un méta-modèle plus proche des surfaces de réponses. Au lieu d'effacer les informations des générations précédentes, il réutilise toutes les informations disponibles pour construire une approximation plus précise et plus fiable de la valeur de la fonction coût.

Supposons donc qu'à la génération courante, on dispose de *Nbpcalc* points maîtres, notés $X_k^A$ – les anciens, dont le critère et le gradient exacts ont déjà été calculés aux générations précédentes (à la génération 0, *Nbpcalc* = 0).

Nous décidons ensuite du nombre *Nbpeva* d'évaluations exactes que nous nous permettons à l'étape courante. Les individus sélectionnés, selon une stratégie que nous décrirons plus loin, seront notés $X_j^N$ - les nouveaux.

Nous allons d'abord décrire comment utiliser les $N_M$ = *Nbpeva* + *Nbpcalc* individus maîtres pour réaliser l'interpolation, puis comment choisir un placement optimal des nouveaux points $X_j^N$.

## II.3.1. Interpolation de Liszka-Orkisz.

La méthode d'interpolation de Liszka-Orkisz est utilisée pour construire le méta-modèle. Dans cette section, on ne distingue pas les nouveaux points maîtres des anciens, car les approximations ne sont faites qu'une fois que tous les nouveaux points ont été évalués.

On possède $N_M$ = *Nbpcalc* + *Nbpeva* points maîtres en lesquels la fonction coût et son gradient sont évalués : $\begin{cases} \Phi(X_j)_{j=1...N_M} \\ \nabla\Phi(X_j)_{j=1...N_M} \end{cases}$

Supposons que l'individu *i* de la population est approximé au point *j* maîtres par :

$$\Phi(X_i) \approx \Phi(X_j) + \nabla\Phi(X_j)\Delta X_{ij} \qquad \forall X_j, j = 1,...,N_M \tag{2.2}$$

où $\Delta X_{ij} = X_i - X_j$

La méthode de Liszka-Orkisz [*Liszka et al. 1980*] consiste en la recherche de la valeur de l'approximation $\hat{\Phi}(X_i)$ de la fonction coût en $X_i$, qui minimise l'erreur moyenne quadratique $E(\hat{\Phi}(X_i))$ des approximations linéaires effectuées en tous les points maîtres. Elle s'écrit :

$$E(\hat{\Phi}(X_i)) = \frac{1}{2}\sum_{j=1}^{N_M}\left(\frac{\Phi(X_i) - \Phi(X_j) - \nabla\Phi(X_j)\Delta X_{ij}}{(\Delta X_{ij})^2}\right)^2 \tag{2.3}$$

Par un calcul simple, nous trouvons que :

$$\hat{\Phi}(X_i) = \frac{\sum_{j=1}^{N_M}\frac{\Phi(X_j) + \nabla\Phi(X_j)\Delta X_{ij}}{(\Delta X_{ij})^4}}{\sum_{j=1}^{N_M}\frac{1}{(\Delta X_{ij})^4}} \tag{2.4}$$

Cette approximation est continue. Elle ressemble aux moindres carrés mobiles, si ce n'est qu'ici on considère tous les points sans utiliser une fonction de pondération qui s'annule en dehors d'un voisinage du point. On remarque toutefois le poids en $\frac{1}{(\Delta X_{ij})^4}$ qui remplace.

La *Figure 1.6* en présente un exemple en 1D. On n'a plus de problème comme présenté dans la *Figure 1.5*. La précision augmente avec le nombre de points maîtres pris en compte, ce qui permet d'envisager un enrichissement progressif de méta-modèle au cours des générations, obtenant ainsi un algorithme à mémoire.

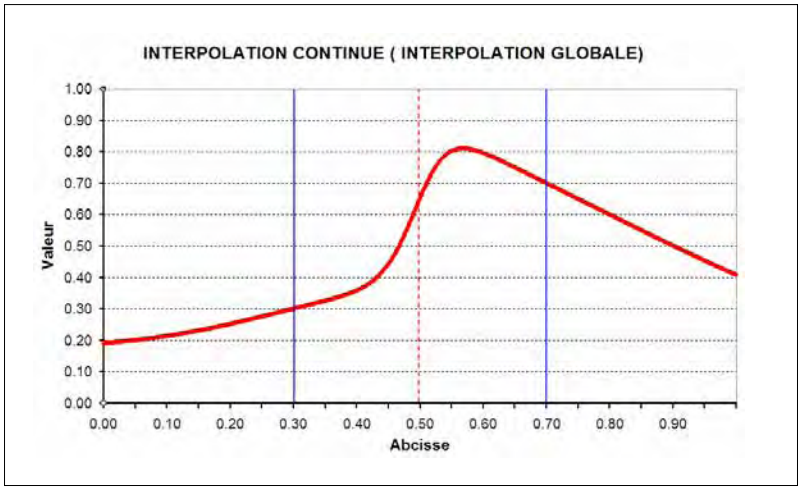



*Figure 1.6. Interpolation continue basée sur la méthode de Liszka-Orkisz*

Dans ce qui suit, nous nous intéressons au placement "au mieux" des nouveaux points maîtres, en tenant compte de l'existence des anciens points maîtres déjà évalués.

## II.3.2. Placement optimal des nouveaux points maîtres

Durant l'évolution de la population, les *Nbpeva* nouveaux points maîtres à évaluer exactement sont sélectionnés de façon à minimiser l'erreur d'approximation en tous les individus de la population :

$$Erreur = \sum_{i=1}^{Nbind} \left( \Phi(X_i) - \hat{\Phi}(X_i) \right)^2 \qquad (2.5)$$

où *Nbind* est le nombre d'individus de la population et $\hat{\Phi}(X_i)$ est calculé par l'équation (2.4), nous avons :

$$Erreur = \sum_{i=1}^{Nbind} \left( \Phi(X_i) - \frac{\sum_{j=1}^{N_M} \frac{\Phi(X_j) + \nabla\Phi(X_j)\Delta X_{ij}}{(\Delta X_{ij})^4}}{\sum_{j=1}^{N_M} \frac{1}{(\Delta X_{ij})^4}} \right)^2 \qquad (2.6)$$

ou encore $$Erreur = \sum_{i=1}^{Nbind} \left( \frac{\sum_{j=1}^{N_M} \frac{\Phi(X_i) - (\Phi(X_j) + \nabla\Phi(X_j)\Delta X_{ij})}{(\Delta X_{ij})^4}}{\sum_{j=1}^{N_M} \frac{1}{(\Delta X_{ij})^4}} \right)^2$$

En supposant que le Hessien est borné et défini positif. On a :

$$\Phi(X_i) - \Phi(X_j) + \nabla\Phi(X_j)\Delta X_{ij} \cong \Delta X_{ij} \nabla^2\Phi(X_j)\Delta X_{ij} \le C\left\|\Delta X_{ij}\right\|^2 \tag{2.7}$$

d'où $$Erreur \cong \sum_{i=1}^{Nbind} \left( \frac{\sum_{j=1}^{N_M} \frac{\Delta X_{ij} \nabla^2\Phi(X_j)\Delta X_{ij}}{(\Delta X_{ij})^4}}{\sum_{j=1}^{N_M} \frac{1}{(\Delta X_{ij})^4}} \right)^2 \le C \sum_{i=1}^{Nbind} \left( \frac{\sum_{j=1}^{N_M} \frac{1}{(\Delta X_{ij})^2}}{\sum_{j=1}^{N_M} \frac{1}{(\Delta X_{ij})^4}} \right)^2 \tag{2.8}$$

où C est la borne de $\left\|\nabla^2\Phi(X)\right\|$.

La minimisation de l'erreur d'approximation est maintenant approchée par la minimisation de la fonction suivante :

$$\Pi(X_j) = \sum_{i=1}^{Nbind} \left( \frac{\sum_{j=1}^{N_M} \frac{1}{(\Delta X_{ij})^2}}{\sum_{j=1}^{N_M} \frac{1}{(\Delta X_{ij})^4}} \right)^2 \tag{2.9}$$

Comme $N_M$ = *Nbpcalc* + *Nbpeva*, la fonction à minimiser qui tient en compte de l'existence des points maîtres déjà évalués prend la forme:

$$\Pi(X_j) = \sum_{i=1}^{nbind} \left( \frac{\sum_{k=1}^{nbpcalc} \frac{1}{(\Delta X_{ik})^2} + \sum_{j=1}^{nbpeva} \frac{1}{(\Delta X_{ij})^2}}{\sum_{k=1}^{nbpcalc} \frac{1}{(\Delta X_{ik})^4} + \sum_{j=1}^{nbpeva} \frac{1}{(\Delta X_{ij})^4}} \right)^2 \tag{2.10}$$

La minimisation de cette fonction nous amène donc à un sous problème d'optimisation qui est assez complexe à résoudre. Le temps d'une évaluation de cette fonction étant négligeable, nous utilisons un AG pour résoudre ce sous problème. Un autre AG est donc intégré dans le programme pour ce problème de minimisation. En pratique, nous utilisons l'AG de Carroll [*Carroll 1997*], et les nouveaux points à évaluer sont identifiés facilement. Les figures suivantes nous montrent des exemples en 2D de dispositions des nouveaux points à évaluer exactement :

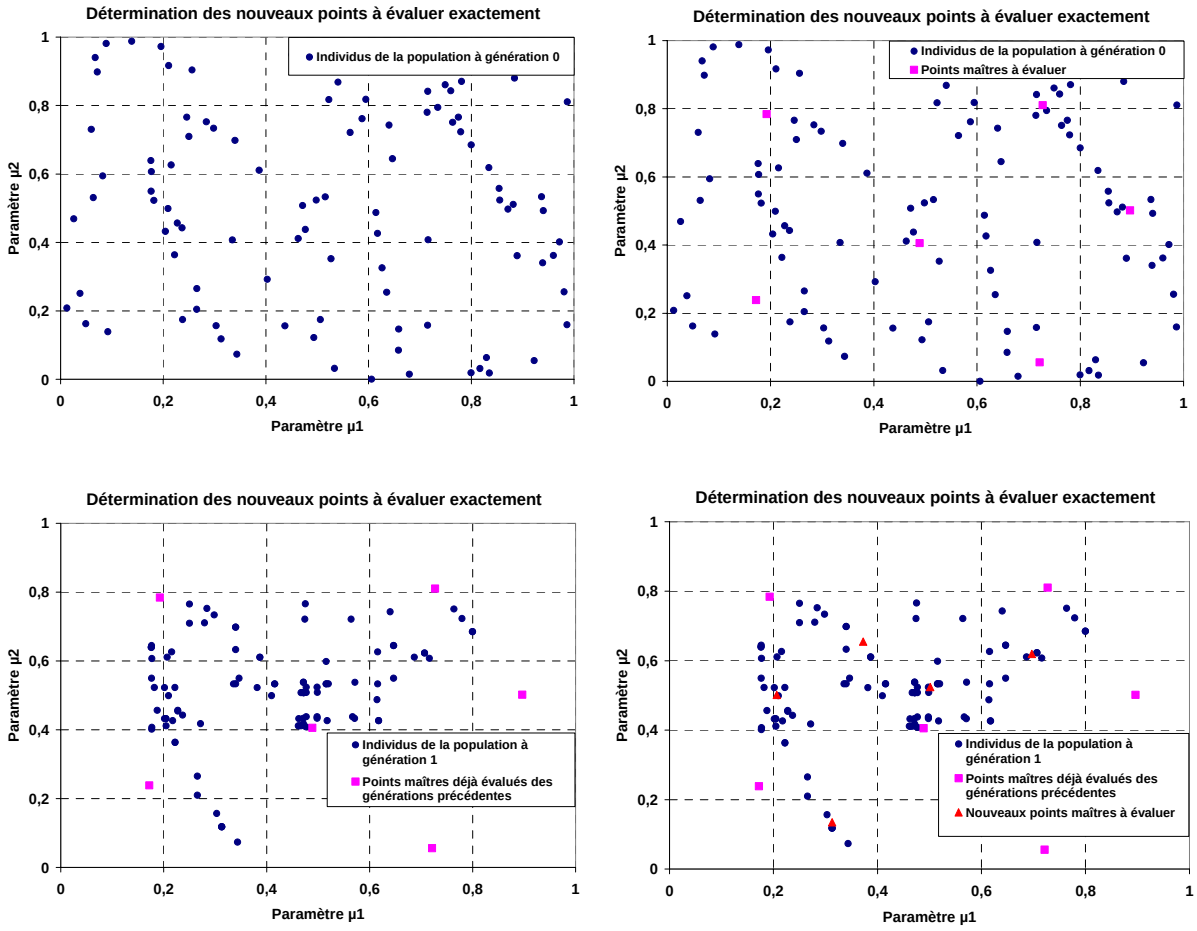



*Figure 1.7. Disposition des individus, des anciens points maîtres et des nouveaux points maîtres.*

## II.4. CONCLUSION

Nous avons présenté, dans ce chapitre, une approche d'optimisation hybride, dont le principe est piloter l'optimisation par un algorithme global, généralement gourmand en évaluations, et de tirer profit de la disponibilité des gradients pour fournir à ce dernier des approximations très peu coûteuses de la fonction coût. La contribution originale de ce travail concerne le choix des points dits maîtres pour lesquels le coût exact et le gradient sont calculés. On obtient ainsi une méthode d'optimisation globale dont on maîtrise à l'avance le coût en calculs.

Cette méthode a été implémentée avec un algorithme génétique comme pilote, de deux façons: la première, sans mémoire, utilise une approximation discontinue. La seconde, avec mémoire des évaluations exactes précédentes, utilise l'approximation continue de Liszka-Orkisz. Nous allons donc les appliquer au forgeage.

# Chapitre III

# CALCUL DU GRADIENT DES FONCTIONS COUTS PAR LA METHODE DE L'ETAT ADJOINT – EXTENSION AU FORGEAGE MULTI-PASSES

## III.1. INTRODUCTION

Une manière de résoudre efficacement un problème d'optimisation consiste à utiliser le gradient des fonctions coûts par rapport aux paramètres à optimiser et à recourir à des algorithmes à base de gradient. Cependant, le calcul du gradient est complexe dans le cadre de la simulation du forgeage 3D par des logiciels éléments finis, notamment celui par rapport aux paramètres de forme. Les verrous principaux de ce problème ont été levés dans [*Laroussi 2003*]. Dans son travail, Laroussi a implémenté une première version de la méthode de l'état adjoint qui permet de calculer le gradient de certaines fonctions coûts par rapport aux paramètres de type "forme du lopin initial".

Cependant, ce type de paramètres est lié par la contrainte de volume constant, qui n'est pas facile à traiter. De plus, le calcul du gradient des fonctions coûts par rapport au paramètre "forme du lopin initial" est "spécifique" pour chaque problème. Nous ne pouvons pas utiliser ce calcul d'un cas à l'autre. Eviter la contrainte de volume constant et s'affranchir les spécificités du calcul du gradient est donc une tâche envisagée dans ce contexte. Etendre ce calcul aux problèmes de forgeage multi-passes est donc une bonne solution pour achever cette tâche. Cette extension constitue une contribution importante de cette thèse et fait l'objet de ce chapitre.

Dans un premier temps (paragraphe III.2), ce chapitre décrit le problème mécanique du forgeage résolu dans le code éléments finis FORGE3® (développé au CEMEF). Il présente différentes fonctions coûts retenues dans le cadre de ce travail, ***avec une amélioration apportée sur la fonction coût pour mieux détecter le défaut de repli***. Il rappelle aussi la méthode de l'Etat Adjoint implémentée dans ce code pour calculer le gradient des fonctions coûts.

Dans un second temps (paragraphe III.3), il expose ***le cœur de la contribution apportée avec une extension du travail de Laroussi [Laroussi 2003] sur le calcul du gradient pour le forgeage multi-passes pour l'optimisation de formes des outils***. Il présente aussi ***la validation de ce calcul du gradient***, en comparant les valeurs obtenues par la méthode de l'Etat Adjoint avec celles obtenues par la méthode des différences finies**.**

Enfin, ce chapitre se termine par un bilan des améliorations apportées.

# III.2. CALCUL DU GRADIENT DE LA FONCTION COUT PAR LA METHODE DE L'ETAT ADJOINT DANS LE LOGICIEL FORGE3®

## III.2.1. Expression du problème mécanique de forgeage et des équations implémentées dans le logiciel FORGE3®

Cette première partie présente la modélisation des procédés de mise en forme par forgeage utilisée dans le code éléments finis FORGE3®. Nous formulerons tout d'abord le problème mécanique de forgeage comme un problème continu avec les équations de conservation, les lois de comportement et les conditions aux limites. Ensuite, nous décrirons le problème discrétisé issu du problème continu précédent, en introduisant la formulation éléments finis. Puis, nous présenterons la gestion du contact et les méthodes de résolution utilisées. Enfin, quelques remarques seront faites sur le remaillage et le transport des variables.

### III.2.1.1. Formulation du problème continu

Le problème mécanique de la mise en forme par forgeage est classique, c'est pourquoi nous nous limiterons ici à exposer les équations de base dans un cadre isotherme. Le lecteur intéressé pourra se référer à [*Cescutti 1989*] pour plus de détails sur la formulation mécanique et à [*Soyris 1990*] pour l'aspect anisotherme.

Mettons nous à l'échelle macroscopique et considérons le cas du forgeage à chaud. Le lopin à forger est considéré comme un domaine continu et homogène, $\Omega$ de frontière $\partial\Omega$. Pour la modélisation du problème, on choisit d'exprimer les équations en tout point de $\Omega$ en fonction du champ de vitesse $v$ et du champ de pression $p$. Ceci conduit à l'utilisation d'une formulation à deux champs (vitesse et pression). Afin de ne pas alourdir les équations, dans le cadre du forgeage des métaux à chaud, les hypothèses suivantes sont posées :

- les actions de l'inertie et de la gravité sont négligeables devant celles des déformations plastiques.
- la composante élastique de la déformation est négligeable par rapport à celle plastique, le matériau est donc supposé purement viscoplastique.
- le matériau forgé est donc incompressible.

#### *III.2.1.1.1. Equations d'équilibre*

L'écoulement de la matière lors de la mise en forme doit vérifier les principes fondamentaux de la mécanique des milieux continus. Avec les hypothèses précédentes, ces principes s'écrivent mathématiquement comme suit :

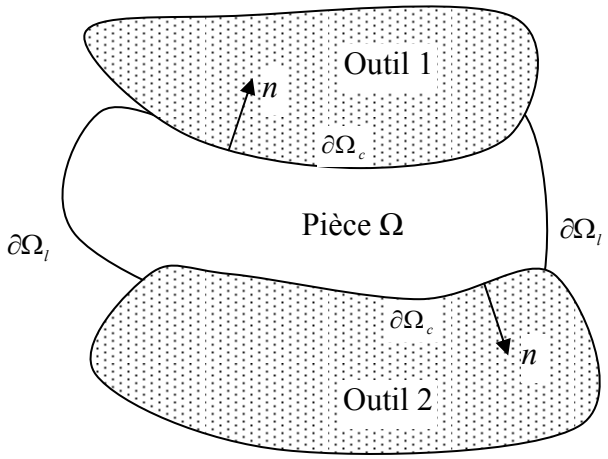
- l'équilibre dynamique $div\sigma = 0$ (3.1)
- l'incompressibilité $divv = 0$ (3.2)

où $\sigma$ est le tenseur des contraintes de Cauchy décomposé en : $\sigma = s - pI$

$s$ est le déviateur des contraintes, $I$ le tenseur identité et $p$ la pression hydrostatique définie par $p = -\frac{1}{3} Trace(\sigma)$

#### III.2.1.1.2. Conditions aux limites

Pour considérer les conditions aux limites, on décompose la surface de la pièce forgée (ou du domaine $\Omega$) $\partial\Omega$ en deux parties distinctes (voir *Figure 3.1* pour une représentation en 2D) :




$\partial\Omega = \partial\Omega_l \cup \partial\Omega_c$

où $\partial\Omega_l$ surface libre

$\partial\Omega_c$ surface en contact avec l'outillage

*Figure 3.1. Définition des surfaces et de la normale à la pièce*

Sur la surface libre $\partial\Omega_l$, la pièce n'est soumise à aucune contrainte, ce qui est traduit par :

$$\sigma n = 0 \tag{3.3}$$

avec $n$ la normale sortante à la pièce (voir *Figure 3.1*)

Sur la surface en contact avec l'outillage $\partial\Omega_c$, deux types de conditions sont imposés:

- une condition tangentielle (condition de frottement) qui permet d'exprimer la valeur de la cission de frottement $\tau$ définie par :

$$\tau = \sigma n - (\sigma n.n)n \tag{3.4}$$

  Cette condition est associée à une loi tribologique qui sera présentée plus loin.

- une condition de contact unilatéral ou de Signorini, qui exprime la non pénétration de la pièce dans l'outillage. Elle est décrite par les équations suivantes :

$$\begin{cases} (v - v_{outil}).n \le 0 \\ \sigma_n \le 0 \\ (v - v_{outil}).n\,\sigma_n = 0 \end{cases} \tag{3.5}$$

  où $v_{outil}$ est la vitesse de l'outil et $\sigma_n = \sigma n.n$ est la pression normale de contact. La condition de contact est portée par la normale à la surface de la pièce.

#### III.2.1.1.3. Lois de comportement

##### Loi rhéologique

Pour modéliser l'écoulement du matériau, on utilise une loi de comportement de type Norton-Hoff. Cette loi s'écrit :

$$s = 2K\left(\sqrt{3}\dot{\bar{\varepsilon}}\right)^{(m-1)}\dot{\varepsilon} \tag{3.6}$$

où $K$ est la consistance du matériau, $m$ la sensibilité à la vitesse de déformation, $\dot{\varepsilon}$ le tenseur des vitesses de déformation calculé par : $\dot{\varepsilon} = \frac{1}{2}\left(\nabla v + {}^{T}\nabla v\right)$, $\dot{\bar{\varepsilon}}$ la vitesse de déformation généralisée donnée par : $\dot{\bar{\varepsilon}} = \sqrt{\frac{2}{3}\dot{\varepsilon} : \dot{\varepsilon}}$

La consistance K du matériau peut-être fonction de la déformation généralisée $\bar{\varepsilon}$ et de la température. Cependant, on se place ici dans le cadre isotherme et on considère que le matériau ne présente pas d'écrouissage. Pour plus de détails sur les équations de comportement, le lecteur peut se référer à [*Rappaz et al. 1998*] et [*Aliaga 2000*].

##### Loi de frottement

La loi tribologique de type Norton est utilisée pour traduire l'action du frottement. Cette loi est bien adaptée aux fortes pressions de contact caractérisant les procédés de forgeage. Elle s'écrit comme suit :

$$\tau = -\alpha_f K\left\|\Delta v_g\right\|^{\left(p_f - 1\right)}\Delta v_g \tag{3.7}$$

où $\Delta v_g$ est la vitesse de glissement de l'outil : $\Delta v_g = v - v_{outil} - \left[\left(v - v_{outil}\right).n\right]n$ (3.8)

$\alpha_f$ est le coefficient de frottement et $p_f$ le coefficient de sensibilité à la vitesse de glissement. Souvent, on choisit $p_f = m$.

#### III.2.1.1.4. Système d'équations à résoudre

En rassemblant les équations précédentes avec la décomposition de $\sigma$ en fonction de $s$ et $p$, on aboutit au système d'équations suivant :

$$\begin{cases} div\, s - grad\, p = 0 & \text{sur } \Omega \\ div(v) = 0 & \text{sur } \Omega \\ \tau = -\alpha_f K\left\|\Delta v_g\right\|^{m-1}\Delta v_g & \text{sur } \partial\Omega_c \\ \left(v - v_{outil}\right).n \le 0 \quad \text{sur } \partial\Omega_c \quad \text{et} \quad \sigma n = 0 \quad \text{sur } \partial\Omega_l \end{cases} \tag{3.9}$$

#### *III.2.1.1.5. Forme faible du problème continu:*

Pour obtenir la formulation faible du problème, on multiplie les équations locales (3.9) par des fonctions tests et les intègres. Pour l'instant, afin de ne pas alourdir la présentation avec les termes de contact, on introduit les contraintes que doivent vérifier le champ de vitesse dans l'espace des fonctions tests. Enfin, les fonctions tests qu'on utilise dans les formulations qui suivent évoluent dans les espaces $V_0^{ca}$ et $P$ définis par :

$$\begin{aligned} V^{ca} &= \left\{ v \in \left(H^1(\Omega)^3\right);\ \ div\, v = 0 \text{ sur } \Omega;\ (v - v_{outil}).n \le 0 \text{ sur } \partial\Omega_c \right\} \\ V_0^{ca} &= \left\{ v \in \left(H^1(\Omega)^3\right);\ \ div\, v = 0 \text{ sur } \Omega;\ v.n \le 0 \text{ sur } \partial\Omega_c \right\} \\ P &= L^2(\Omega) \end{aligned}$$

Le Principe des Puissances Virtuelles nous permet d'écrire la formulation faible à partir du problème fort (3.9) comme suit :

Trouver $(v, p) \in (V^{ca}, P)$ tel que $\forall (v^*, p^*) \in (V_0^{ca}, P)$

$$\begin{cases} \int_\Omega s : \dot{\varepsilon}(v^*) d\Omega - \int_\Omega p\, div(v^*) d\Omega - \int_{\partial\Omega_c} \tau . v^*\, dS = 0 \\ \int_\Omega p^*\, div(v) d\Omega = 0 \end{cases} \tag{3.10}$$

Une formulation plus détaillée avec les démonstrations de l'existence et de l'unicité de la solution est présentée dans [*Traoré 2001*] par exemple.

En vue d'une résolution par la méthode des éléments finis, il est indispensable d'écrire le système (3.10) sous sa forme discrétisée. Cela sera l'objectif du prochain paragraphe.

### III.2.1.2. Formulation du problème discrétisé

La discrétisation du problème précédent est réalisé en temps et en espace.

#### *III.2.1.2.1. Discrétisation en temps*

L'intervalle de temps du procédé [$t_0$, $t_{fin}$], où $t_0$ et $t_{fin}$ désignent respectivement le temps initial et final de la simulation de forgeage, est discrétisé en plusieurs pas de temps. Le logiciel FORGE3® utilise le schéma d'intégration temporel de type Euler explicite comme suit:

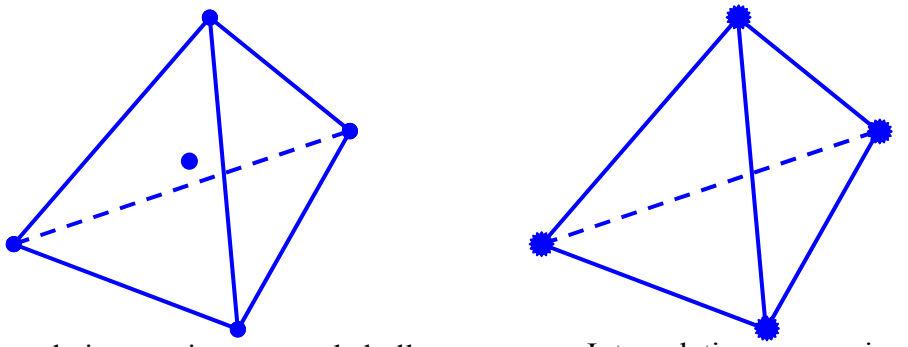
$$X^{t+\Delta t} = X^t + \Delta t\, V^t \tag{3.11}$$

L'utilisation d'un tel schéma revient à faire l'hypothèse que la vitesse est constante sur l'incrément temps $\Delta t$.

La discrétisation temporelle est adoptée. Il faut maintenant choisir un type d'éléments finis adéquat pour la discrétisation spatiale.

#### III.2.1.2.2. Discrétisation en espace

La discrétisation spatiale consiste à approximer les espaces d'admissibilité de la vitesse $V^{ca}$ et de la pression $P$ par des espaces discrets de dimensions finies $V_h^{ca}$ et $P_h$. Nous souhaitons bien évidemment que les fonctions $(v_h, p_h)$ convergent vers les solutions du problème continu $(v, p)$. Pour obtenir cette convergence, le problème doit être consistant, c'est-à-dire que l'espace d'approximation doit tend vers l'espace continu quand la taille de maille *h* tend vers zéro. Ce sera le cas puisque les espaces d'approximation avec lesquels nous travaillerons sont des espaces de polynômes.



Interpolation en vitesse avec la bulle  Interpolation en pression

*Figure 3.2. Elément fini mixte P1+/P1*

Pour coupler les champs de vitesse et de pression, la condition de consistance ne suffit plus, il faut lui adjoindre une condition de stabilité spatiale appelée condition de Brezzi-Babŭska [*Brezzi et al. 1991*]. Cette condition traduit le fait que les espaces $V_h^{ca}$ et $P_h$ ne peuvent pas être pris indépendamment. Compte tenu cette condition, l'élément fini mixte tétraédrique P1+/P1 [*Arnold et al. 1984*] a été utilisé et implémenté dans FORGE3® [*Coupez 1995*], pour une formulation viscoplastique incompressible. L'idée est d'enrichir l'espace des vitesses par rapport à la pression en introduisant une "bulle" au centre de l'élément. Le champ de vitesse est décomposé en une partie linéaire continue et en une partie linéaire bulle discontinue sur le tétraèdre de base. La contribution "bulle" étant linéaire et continue sur les quatre sous-tétraèdres définis par les 4 sommets et le centre de gravité du tétraèdre (voir *Figure 3.2*). La pression, quant à elle, est linéaire sur le tétraèdre.

$$P_h = \left\{ p_h \in C^0(\Omega_h);\quad \forall \omega_h \in \Gamma_h,\quad p_{h|\omega_h} \in P^1(\omega_h) \right\}$$

$$V_h = L_h \oplus B_h$$

$$L_h = \left\{ v_h \in \left(C^0(\Omega_h)\right)^3;\quad \forall \omega_h \in \Gamma_h,\quad v_{h|\omega_h} \in \left(P^1(\omega_h)\right)^3 \right\}$$

$$B_h = \left\{ b_h \in \left(C^0(\Omega_h)\right)^3;\quad \forall \omega_h \in \Gamma_h,\ b_{h|\omega_h^i} \in \left(P^1(\omega_h^i)\right)^3,\ i=1,\ldots,4 \text{ et } b_h = 0 \text{ sur } \partial\omega_h \right\}$$

où les $(\omega_h^i)_{i=1,\ldots,4}$ désignent les quatre sous-tétraèdres de $\omega_h$ ; $\Gamma_h$ est l'ensemble des éléments finis du maillage

Notons $N^l$ et $N^b$ respectivement les fonctions de base linéaires et bulle. Les champs de vitesse et de pression discrets s'écrivent :

$$\begin{aligned} v_h &= v_h^l + v_h^b = \sum_{k=1}^{nbnoe} V_k \, N_k^l + \sum_{j=1}^{nbelt} V_j \, N_j^b \\ p_h &= \sum_{k=1}^{nbnoe} P_k \, N_k^l \end{aligned} \tag{12}$$

où *nbnoe* et *nbelt* représentent respectivement le nombre de nœuds et le nombre d'éléments du maillage.

$$\begin{cases} \forall\, k = 1,\ldots,nbnoe ; \forall i = 1,\ldots,3 & \\ \quad \int_{\Omega_h} 2K\left(\sqrt{3}\dot{\bar{\varepsilon}}_h\right)^{m-1} \dot{\varepsilon}_h\left(v_h^l + v_h^b\right) : \dot{\varepsilon}_h\left(N_k^l \, e_i\right) d\Omega_h + & \\ \quad \int_{\partial\Omega_{ch}} \alpha_f K \left\| \Delta v_{gh} \right\|^{m-1} \left(\Delta v_{gh} . e_i\right) N_k \, dS_h - \int_{\Omega_h} p_h \, div\left(N_k^l \, e_i\right) d\Omega_h & = 0 \\ \forall\, e = 1,\ldots,nbelt ; \forall i = 1,\ldots,3 & \\ \quad \int_{\Omega_h^e} 2K\left(\sqrt{3}\dot{\bar{\varepsilon}}_h\right)^{m-1} \dot{\varepsilon}_h\left(v_h^l + v_h^b\right) : \dot{\varepsilon}_h\left(N_e^b \, e_i\right) d\Omega_h - \int_{\Omega_h^e} p_h \, div\left(N_e^b \, e_i\right) d\Omega_h & = 0 \\ \forall\, k = 1,\ldots,nbnoe \quad \int_{\Omega_h} N_k^l \, div\left(v_h^l + v_h^b\right) d\Omega_h & = 0 \end{cases} \tag{3.13}$$

où $e_i$ est le vecteur unitaire dans la direction *i* de l'espace.

En remplaçant les fonctions tests dans (3.10) par les fonctions de base éléments finis, nous obtenons la formulation faible discrétisée (3.13) de notre problème (à laquelle manque les équations de contact). Cette formulation est donc valable pour un champ cinématiquement admissible qui vérifie les équations de contact. Les équations de contact sont présentées dans le paragraphe suivant.

### III.2.1.3. Gestion du contact

En ce qui concerne la gestion du contact, nous nous limiterons à une description très générale, un développement plus complet se trouvant dans [*Mocellin 1999*].

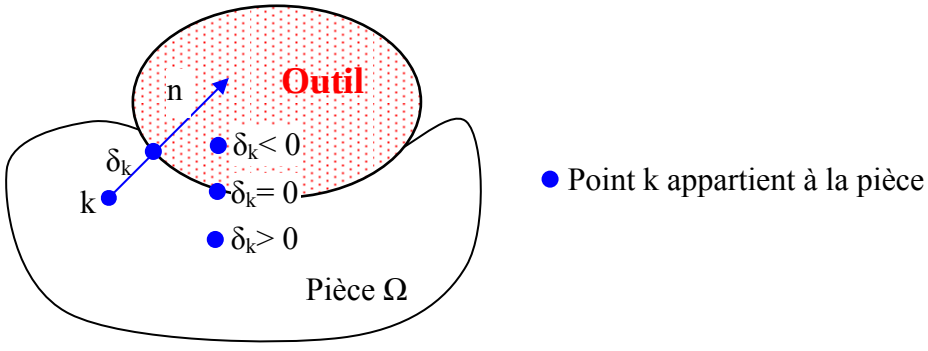



*Figure 3.3. Définition de la distance de contact δ*

En tout point du domaine $\Omega$ décrivant la pièce, notons $\delta$ la distance de ce point à la surface de l'outil. La *Figure 3.3* décrit la convention de distance qui définit $\delta$ par :

- $\delta_k = 0$ si le point *k* appartient à la surface de l'outil
- $\delta_k < 0$ si le point *k* a pénétré dans l'outil
- $\delta_k > 0$ si *k* est à l'extérieur de l'outil

La condition de non-pénétration que nous voulons imposer sur la surface revient à écrire :

$$\delta \geq 0 \quad \text{sur } \partial\Omega \tag{3.14}$$

Au cours des incréments de temps, les points de $\partial\Omega$ potentiellement en contact avec l'outil ne doivent pas pénétrer dans celui-ci. On autorise ces points, soit à rester en contact avec l'outil, soit à quitter le contact dans le sens des $\delta > 0$. Pour cela, avec un développement limité à l'ordre 1 en temps, la condition suivante est imposée :

$$\delta^{t+\Delta t} = \delta^t + \left(v_{outil} - v\right).n^t \, \Delta t + O\left(\Delta t^2\right) \geq 0 \tag{3.15}$$

Cette condition de contact unilatéral peut encore être exprimée à l'ordre 1 en temps par :

$$h(v) = \left(v - v_{outil}\right).n^t - \frac{\delta^t}{\Delta t} \leq 0 \tag{3.16}$$

Au niveau discret, l'équation (3.16) s'écrit :

$$\forall\, k \in \partial\Omega_h, \;\; h\left(V_k\right) = \left(V_k - v_{outil}\right).n_k - \frac{\delta_k}{\Delta t} \leq 0 \tag{3.17}$$

Dans le logiciel FORGE3®, le contact est traité en chaque nœud de la surface du domaine par une méthode de pénalisation décrite de manière détaillée dans [*Mocellin 1999*]. En prenant en compte le terme de contact pénalisé, nous obtenons la formulation discrétisée finale (3.18) de notre problème de forgeage :

$$\begin{cases} \forall\, k = 1,\ldots,nbnoe\,;\forall i = 1,\ldots,3 \\ \quad \displaystyle\int_{\Omega_h} 2K\left(\sqrt{3}\dot{\bar{\varepsilon}}_h\right)^{m-1} \dot{\varepsilon}_h\left(v_h^l + v_h^b\right) : \dot{\varepsilon}_h\left(N_k^l\, e_i\right) d\Omega_h + \int_{\partial\Omega_h} 1_{\partial\Omega_f}\, \alpha_f K \left\|\Delta v_{gh}\right\|^{m-1} \left(\Delta v_{gh}.e_i\right) N_k \, dS_h - \\ \quad \displaystyle\int_{\Omega_h} p_h \, div\left(N_k^l\, e_i\right) d\Omega_h + \rho 1_{\partial\Omega_c}(k) \left[\left(V_k - v_{outil}\right).n_k - \frac{\delta_k}{\Delta t}\right] n_k^i S_k = 0 \\ \forall\, e = 1,\ldots,nbelt\,;\forall i = 1,\ldots,3 \\ \quad \displaystyle\int_{\Omega_h^e} 2K\left(\sqrt{3}\dot{\bar{\varepsilon}}_h\right)^{m-1} \dot{\varepsilon}_h\left(v_h^l + v_h^b\right) : \dot{\varepsilon}_h\left(N_e^b\, e_i\right) d\Omega_h - \int_{\Omega_h^e} p_h \, div\left(N_e^b\, e_i\right) d\Omega_h = 0 \\ \forall\, k = 1,\ldots,nbnoe \quad \displaystyle\int_{\Omega_h} N_k^l \, div\left(v_h^l + v_h^b\right) d\Omega_h = 0 \end{cases} \tag{3.18}$$

où $\rho$ est un coefficient de pénalisation pris suffisamment grand, $S_k = \int_{\partial\Omega_h} N_k^l dS_h$ la surface associée au nœud $k$ , $1_{\partial\Omega_c}(k)$ l'indicatrice du contact au nœud $k$, $\partial\Omega_f$ surface de frottement et $1_{\partial\Omega f}(x)$ l'indicatrice de frottement au nœud x, avec :

$$\begin{aligned} &\partial\Omega_f = \{x \in \partial\Omega \text{ tel que } \delta \le 0\} \\ &\forall x \in \partial\Omega, \begin{cases} 1_{\partial\Omega_f}(x) = 1 & \text{si } \delta(x) \le 0 \\ 1_{\partial\Omega_f}(x) = 0 & \text{si non} \end{cases} \end{aligned} \tag{3.19}$$

$$\begin{cases} 1_{\partial\Omega_c}(k) = 1 & \text{si } (V_k - v_{outil}).n_k - \dfrac{\delta_k}{\Delta t} > 0 \\ 1_{\partial\Omega_c}(k) = 0 & \text{si non} \end{cases} \tag{3.20}$$

Nous pouvons écrire le système (3.18) sous la forme d'un système vectoriel non linéaire :

$$R(v^l, v^b, p) = 0 \tag{3.21}$$

### III.2.1.4. Méthode de résolution implémentée dans FORGE3®

Dans FORGE3®, on utilise l'algorithme de Newton-Raphson pour résoudre itérativement le système (3.21) à chaque incrément de temps. Etant donné le triplet initial $(v_{init}^l, v_{init}^b, p_{init})$, on cherche une correction $(\Delta v^l, \Delta v^b, \Delta p)$ à apporter telle que :

$$R(v_{init}^l + \Delta v^l, v_{init}^b + \Delta v^b, p_{init} + \Delta p) = 0 \tag{3.22}$$

A l'aide d'un développement de Taylor à l'ordre un, on obtient :

$$R(v_{init}^l, v_{init}^b, p_{init}) + \frac{\partial R}{\partial v^l}\Delta v^l + \frac{\partial R}{\partial v^b}\Delta v^b + \frac{\partial R}{\partial p}\Delta p = 0 \tag{3.23}$$

Ce qui conduit à la résolution du système linéaire :

$$\begin{pmatrix} \dfrac{\partial R_l}{\partial v^l} & \dfrac{\partial R_l}{\partial v^b} & \dfrac{\partial R_l}{\partial p} \\ \dfrac{\partial R_b}{\partial v^l} & \dfrac{\partial R_b}{\partial v^b} & \dfrac{\partial R_b}{\partial p} \\ \dfrac{\partial R_p}{\partial v^l} & \dfrac{\partial R_p}{\partial v^b} & \dfrac{\partial R_p}{\partial p} \end{pmatrix} \begin{pmatrix} \Delta v^l \\ \Delta v^b \\ \Delta p \end{pmatrix} = - \begin{pmatrix} R_l \\ R_b \\ R_p \end{pmatrix} \tag{3.24}$$

Il faut remarquer que le terme $\dfrac{\partial R_p}{\partial p} = 0$ [*Mocellin 1999*]. De plus, une propriété du champ « bulle » dans le cas linéaire ($m$ = 1), qui s'étend au cas non linéaire [*Perchat 2000*] donne :

$$\frac{\partial R_l}{\partial v^b} = \frac{\partial R_b}{\partial v^l} = 0 \tag{3.25}$$

Nous pouvons maintenant simplifier le système (3.24) comme suit :

$$\begin{pmatrix} \frac{\partial R_l}{\partial v^l} & 0 & \frac{\partial R_l}{\partial p} \\ 0 & \frac{\partial R_b}{\partial v^b} & \frac{\partial R_b}{\partial p} \\ \frac{\partial R_p}{\partial v^l} & \frac{\partial R_p}{\partial v^b} & 0 \end{pmatrix} \begin{pmatrix} \Delta v^l \\ \Delta v^b \\ \Delta p \end{pmatrix} = - \begin{pmatrix} R_l \\ R_b \\ R_p \end{pmatrix} \tag{3.26}$$

Si on extrait la seconde ligne du système (3.26), le champ de vitesse « bulle » peut s'exprimer en fonction du champ de pression. Finalement, le système à résoudre se met sous la forme :

$$H \begin{pmatrix} \Delta v^l \\ \Delta p \end{pmatrix} = - R_{\text{modif}} \tag{3.27}$$

où $H$ est la matrice hessienne symétrique et $R_{\text{modif}}$ le second membre modifié obtenus après l'élimination du terme « bulle ».

Dans FORGE3®, la résolution de (3.27) se fait de manière itérative par la méthode du résidu minimal préconditionné. L'intérêt d'une méthode itérative vis à vis d'une méthode directe est énorme, non seulement en terme de temps de calcul mais aussi en terme de coût de stockage [*Marie 1997*].

### III.2.1.5. Remaillage et transport des variables dans FORGE3®

FORGE3® a pour but de simuler des procédés de mise en forme en mettant en jeu de grandes déformations. Sous l'action des outils, la matière se déforme et par conséquent le maillage aussi, puisque FORGE3® adopte une description Lagrangienne. Or, les résultats d'estimation de l'erreur a priori de la méthode éléments finis montrent que les éléments du maillage doivent être de forme aussi proche que celle d'éléments équilatéraux. La précision de méthode éléments finis dégénère avec la qualité du maillage. Donc, pour réserver une bonne qualité de maillage durant la simulation, Coupez [*Coupez 1991*] a développé un algorithme de remaillage automatique dans FORGE3®. Dès que le maillage est de mauvaise qualité, ou à intervalles fixes, un nouveau maillage de meilleure qualité est généré. Il est nécessaire de transporter les variables d'un maillage à l'autre. Pour passer de l'ancienne à la nouvelle configuration, nous définissons un opérateur linéaire de transport noté $J$ par la relation (3.28) :

$$X' = JX \tag{3.28}$$

L'utilisation d'une opération de remaillage nécessite le développement d'une méthode de transport des variables d'histoire depuis l'ancienne configuration $\Omega$ jusqu'à la nouvelle $\Omega'$. Il existe deux types de variables :

- Les variables P0 (constantes sur chaque élément) comme par exemple la déformation équivalente $\bar{\varepsilon}$, la partie déviatorique $s$ du tenseur des contraintes, … calculée aux points d'intégration (centre de l'élément).

- Les variables P1 (variables nodales) comme la vitesse, la pression …

FORGE3® utilise la technique d'interpolation inverse pour le transport des variables, qui est décrite dans le paragraphe qui suit.

#### III.2.1.5.1. Transport d'une variable nodale P1

Le transport des variables nodales s'effectue en deux étapes :

- recherche de l'élément $e$ de l'ancien maillage d'appartenance de chaque nœud $k'$ du nouveau maillage

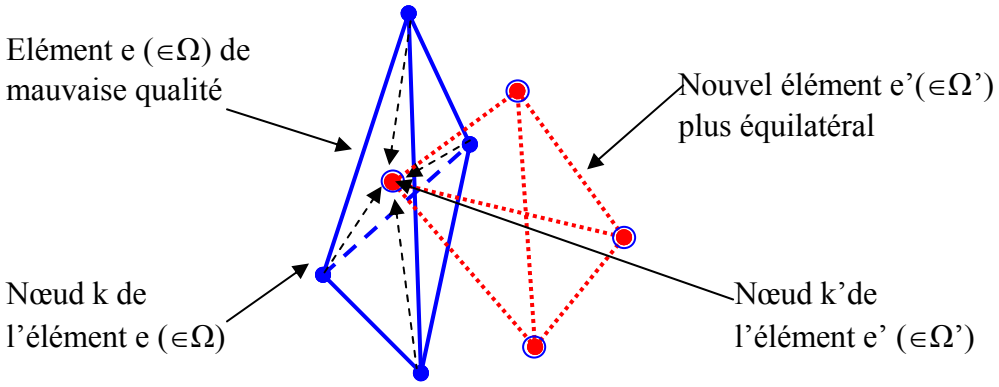



*Figure 3.4. Transport par interpolation inverse*

- interpolation des valeurs nodales depuis les sommets de l'élément $e$ d'appartenance de l'ancien maillage $\Omega$ jusqu'au nœud $k'$ du nouveau maillage $\Omega'$ (*Figure 3.4*). Cette interpolation est effectuée en utilisant les fonctions de forme de l'ancien maillage.

#### III.2.1.5.2. Transport d'une variable élémentaire P0

Le transport des variables P0 par élément est un peu plus délicat. Il est réalisé en deux étapes. Soit $f$ la quantité P0 à transporter.

- dans un premier temps, nous ramenons le transport P0 à un transport P1 (*Figure 3.5*). Pour cela, nous calculons une interpolation nodale $f_n$ à partir des valeurs aux éléments $f_{elt}$ par la méthode des moindre-carrés, en minimisant la fonctionnelle :

$$\Pi_{L^2}(f_n) = \int_\Omega (f - \bar{f})^2 dw = \left\| \sum_{n=1}^{Nbnoe} f_n N_n - \sum_{elt=1}^{Nbelt} f_{elt} 1_{elt} \right\|_{L^2}^2 \qquad (3.29)$$

où $N_n$ désigne les fonctions de base éléments finis et $1_{elt}$ la fonction indicatrice valant un dans l'élément considéré et zéro en dehors.

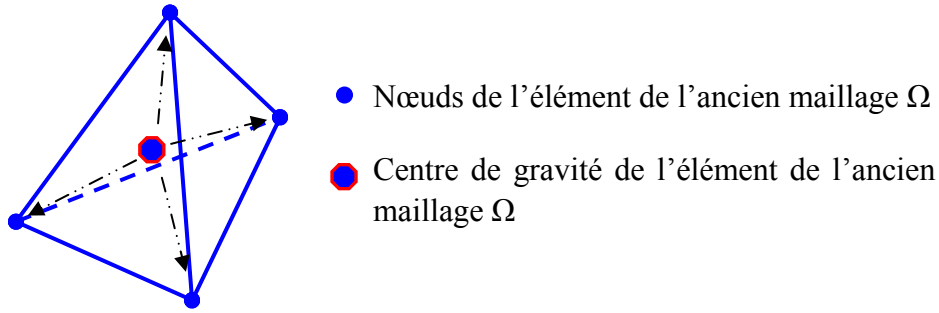



*Figure 3.5. Passage d'une interpolation P0 à une interpolation P1*

- dans une deuxième étape, les quantités nodales $f_n$ sont transportées par la procédure de transport de P1 décrite précédemment (*Figure 3.6*). Ensuite, il faut ramener les variables au centre de l'élément nouveau

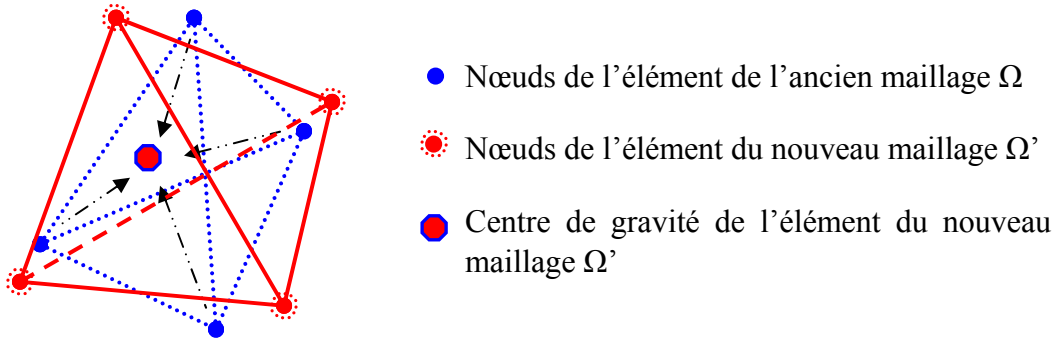



*Figure 3.6. Transport P1 des variables au centre de gravité de l'élément du nouveau maillage Ω'*

Nous venons de décrire le problème mécanique de forgeage et sa résolution au sein de FORGE3®. Dans notre travail, ce logiciel est utilisé pour calculer la valeur des fonctions coûts (ou critères d'optimisation) ainsi que ses gradients, qui seront ensuite utilisés dans nos problèmes d'optimisation. Ces fonctions coûts sont présentées dans le paragraphe qui suit.

## III.2.2. Fonctions coûts

Différentes fonctions coûts ont été implémentées dans le logiciel FORGE3® dans le cadre du travail de Laroussi [*Laroussi 2003*] comme l'énergie totale de mise en forme, la mesure non-qualité de la surface, le remplissage de la matrice finale. Dans ce travail, nous allons présenter seulement deux fonctions coûts utilisées dans nos problèmes d'optimisation respectivement l'énergie totale et la mesure non-qualité de la surface ou de "repli".

### III.2.2.1. Fonction coût énergie

La fonction coût énergie totale de forgeage a été introduite dans [*Fourment et al. 1996*]. Elle représente l'énergie dépensée pour déformer la matière de la configuration $\Omega_{t_0}$ à la configuration $\Omega_{t_{fin}}$. L'expression de cette fonction coût est la suivante :

$$\Phi_{ene} = \int_{t=t_0}^{t_{fin}} \left[ \underbrace{\int_{\Omega_t} \sigma : \dot{\varepsilon}\, dw}_{déformation} + \overbrace{\int_{\partial\Omega_{ct}} \tau.v\, ds}^{frottement} \right] dt \tag{3.30}$$

Dans le cadre d'une loi de comportement de Norton-Hoff et d'une loi de frottement de Norton, cette fonction s'écrit :

$$\Phi_{en} = \int_{t_0}^{t_{fin}} \left( \int_{\Omega_t} K\left(\sqrt{3}\dot{\bar{\varepsilon}}\right)^{m+1} dw + \int_{\partial\Omega_{ft}} \alpha_f K \left\| \Delta v_g \right\|^{p_f+1} ds \right) dt \tag{3.31}$$

où la quantité $\int_{\Omega_t} K\left(\sqrt{3}\dot{\bar{\varepsilon}}\right)^{m+1} dw$ est la puissance de déformation et la quantité $\int_{\partial\Omega_{ft}} \alpha_f K \left\| \Delta v_g \right\|^{p_f+1} ds$ la puissance de frottement.

### III.2.2.2. Fonction coût repli

Le repli est un défaut majeur du forgeage. Un repli apparaît lorsque les frontières $\Omega_t$ de la pièce se chevauchent. A la fin du procédé, la fissure créée représente une zone de faiblesse inacceptable. Ce défaut est donc synonyme de rebut de la pièce. L'élimination de ce défaut est un problème clé de la conception du procédé. Deux questions se posent : Comment savoir quand le repli apparaît ? Et comment s'en éloigner?

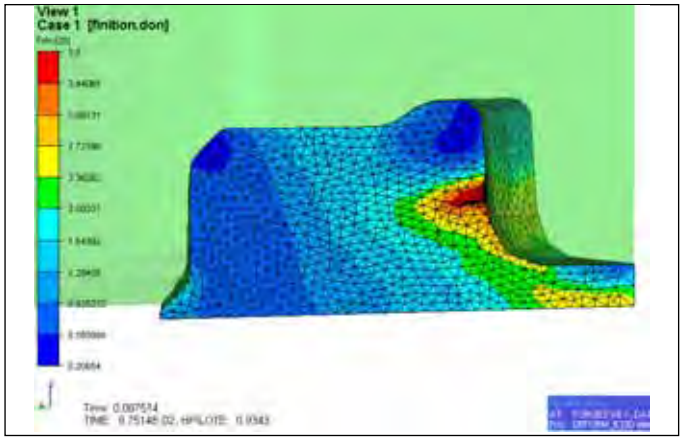



*Figure 3.7. Isovaleurs de $\dot{\bar{\varepsilon}}$ lors de la formation d'un repli de matière*

Pour répondre à la première question, une fonction coût $\Phi_{repli}$ est utilisée, mais sa définition est délicate. On constate que dans la zone de repli, la matière est soumise à de forts cisaillements. Des valeurs anormalement élevées de $\dot{\bar{\varepsilon}}$ sont obtenues, alors même que le repli

n'est pas encore formé, ni détecté (parfois plus de 10 fois supérieures à la valeur moyenne sur le reste de la surface libre – comme le montre la *Figure 3.7*).

Grâce à cette propriété, des fonctions coût $\Phi_{repli}$ basées sur la vitesse de déformation généralisée ont été proposées. Vielledent [*Vieilledent 1999*] a proposé une fonction de la forme :

$$\Phi_{repli1} = \int_{t_0}^{t_{fin}} \left( \int_{\partial\Omega_l} \dot{\bar{\varepsilon}} \, ds \right) dt \tag{3.32}$$

Pourtant, cette fonction ne pouvait être utilisée que localement avec des boites prédéfinies supposant de connaître la zone de formation du repli. Dans son travail, M. Laroussi [***Laroussi 2003***] a modifié cette formule en ajoutant un coefficient $\lambda$ afin de mieux détecter des replis. La fonction coût repli prend la forme :

$$\Phi_{repli2} = \int_{t_0}^{t_{fin}} \left( \int_{\partial\Omega_l} \left(\dot{\bar{\varepsilon}}\right)^{\lambda} ds \right) dt \tag{3.33}$$

Le coefficient $\lambda$ va nous permettre, en lui attribuant des valeurs supérieures à 1, d'amplifier les valeurs élevées de $\dot{\bar{\varepsilon}}$ dans les zones de repli.

Deux tests d'écrasement entre tas plats d'un cube entaillé (*Figure 3.8.a*) et d'un cube normal (*Figure 3.8.b*) ont été effectués. La dimension des cubes est de 50x50x50mm, la hauteur d'écrasent est de 20% de la hauteur initiale. On s'attend à ce que la valeur de la fonction coût repli soit beaucoup plus élevée dans le cas du cube entaillé qu'avec le cube plein.

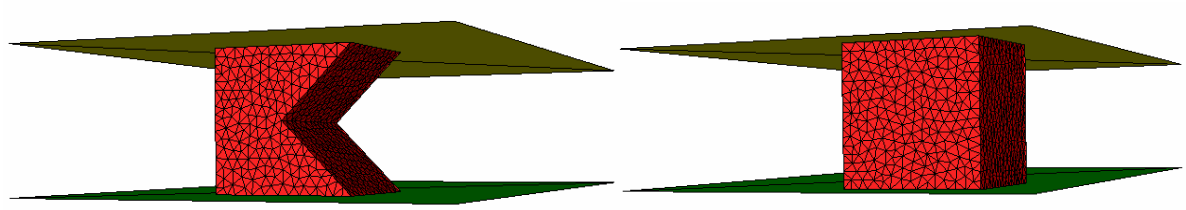


*Figure 3.8. Tests d'écrasement d'un cube entaillé (a) et d'un cube plein (b)*

En effet, dans le cas du cube entaillé, la zone de repli est petite donc la surface d'intégration des valeurs de $\dot{\bar{\varepsilon}}$ est faible. Le *Tableau 3.1* présente la comparaison des valeurs de $\Phi_{repli}$ obtenues. Le critère a une signification globale uniquement pour $\lambda \geq 4$. Autrement dit, le repli est détecté seulement pour des valeurs de $\lambda$ supérieures à 4.

| $\lambda$ | *1* | *2* | *3* | *4* |
|---|---|---|---|---|
| $\Phi_{repli}$ *CUBE PLEIN* | *2106* | *4687* | *10476* | *23510* |
| $\Phi_{repli}$ *CUBE ENTAILLE* | *1490* | *3419* | *9674* | *35427* |

*Tableau 3.1 : Valeur de la fonction coût « repli » non-adimensionnalisée pour différentes valeurs de $\lambda$*

Avec cette fonction coût, nous voyons bien que plus $\lambda$ est grand, mieux la potentialité de repli est identifiée. Cependant, on ne peut utiliser une valeur de $\lambda$ aussi grande que l'on veut, et on s'aperçoit que cette valeur dépend du problème.

Pour s'affranchir de ce problème, on adimensionalise la fonction coût afin de pouvoir utiliser des valeurs plus grandes de $\lambda$ (si nécessaire) et pour essayer d'avoir un critère quantitatif d'apparition de repli. La fonction proposée prend la forme suivante :

$$\Phi_{repli}^{a\,dim} = \frac{1}{t_{fin} - t_0} \int_{t_0}^{t_{fin}} \left( \frac{1}{S_{libref}} \int_{\partial\Omega_{lib}} \left( \frac{\dot{\bar{\varepsilon}}}{\dot{\bar{\varepsilon}}_{car}} \right)^{\lambda} ds \right)^{1/\lambda} dt \tag{3.34}$$

où $\dot{\bar{\varepsilon}}_{car} = \frac{v_{ref}}{h_{ref}}$ est vitesse de déformation généralisée caractéristique

$v_{ref}$ est la vitesse initiale de l'outil de forgeage , $h_{ref}$ la course de forgeage, $S_{libref}$ la surface libre référence (= surface libre initiale)

Avec la nouvelle formule, nous avons effectué les mêmes tests d'écrasement de cubes. Le *Tableau 3.2* regroupe les valeurs de $\Phi_{repli}^{a\,dim}$ pour différentes valeurs de λ.

| $\lambda$ | 1 | 2 | 3 | 4 | 5 | 10 |
|---|---|---|---|---|---|---|
| $\Phi_{repli}^{a\,dim}$ CUBE PLEIN | 1.040 | 1.103 | 1.154 | 1.209 | 1.272 | 1.666 |
| $\Phi_{repli}^{a\,dim}$ CUBE ENTAILLE | 0.511 | 1.027 | 1.633 | 2.306 | 2.959 | 5.336 |

*Tableau 3.2 : Valeur de la fonction coût « repli » adimensionnalisée pour différentes valeurs de $\lambda$*

On constate que dans le cas du cube normal (sans repli) la valeur de la fonction coût de repli ne dépend pas beaucoup de la valeur de $\lambda$. Elle est proche de 1. La vitesse de déformation généralisée est relativement homogène sur la surface libre. Il n'y a pas une grande influence de $\lambda$ dans ce cas. Au contraire, dans le cas du cube entaillé, $\lambda$ joue un rôle important pour amplifier des valeurs anormalement grandes de $\dot{\bar{\varepsilon}}$ dans les zones du repli. On constate que, à partir de $\lambda = 3$, on obtient déjà une différence notable entre les deux cas. Ces résultats nous permettent de dire que la nouvelle fonction coût repli (3.34) donne des résultats plus globaux. Elle fonctionne bien en détectant le repli avec des valeurs de $\lambda \geq 3$. Cela donne une idée pour savoir s'il y a repli ou pas : on fait le rapport des fonctions replis obtenues avec $\lambda = 1$ et $\lambda = 10$ (ou 5). Si ce rapport est grand, il y a probablement un repli.

Dans les parties qui suivent, nous allons utiliser cette nouvelle formule avec une valeur de $\lambda = 10$.

Une fois que le repli est détecté, il nous faut encore répondre à la deuxième question « Comment faire pour l'éliminer ? ». Cela est un des objectifs principaux de ce travail de thèse. Comme nous l'avons mentionné dès le début de ce chapitre, l'utilisation du gradient des

fonctions coûts par rapport aux paramètres à optimiser pour utiliser des algorithmes à base de gradient est une manière de résoudre efficacement ce problème d'optimisation. Le calcul du gradient est complexe mais les verrous principaux de ce problème ont été levés dans [*Laroussi 2003*] en utilisant la méthode de l'Etat Adjoint. Le prochain paragraphe va décrire cette méthode de l'Etat Adjoint implémentée dans FORGE3® pour le calcul du gradient des fonctions coûts.

## III.2.3. Calcul du gradient par la méthode de l'Etat Adjoint

### III.2.3.1. Equations de la méthode de l'Etat Adjoint

Nous avons différentes fonctions coûts $\Phi(\mu)$ qui représentent différents critères du problème d'optimisation. L'objectif de ce paragraphe est de décrire la méthode de l'Etat Adjoint utilisée pour calculer le gradient de ces fonctions coût $\Phi$ par rapport aux paramètres d'optimisation $\mu$, qui peuvent être de deux types :

- paramètres du procédé (vitesse de forgeage, coefficient de frottement, …)
- paramètres de forme (forme du lopin, forme des outils, dimension, …)

Pour un procédé de forgeage instationnaire composé de $N$ incréments de temps, la loi de comportement étant sans "histoire", les fonctions coût $\Phi(\mu)$ et le résidu $R(\mu)$ du problème de forgeage, après discrétisation par éléments finis, sont des fonctions dépendant uniquement de $\mu$, de la position $X$ des nœuds du maillage et de $V$ et $P$. A un incrément donné $i$, nous pouvons écrire:

$$\begin{aligned} R_i(\mu) &= R_i(\mu, X_i(\mu), V_i(\mu), P_i(\mu)) \\ \Phi(\mu) &= \Phi(\mu, X(\mu), V(\mu), P(\mu)) \end{aligned} \tag{3.35}$$

Avec

$$\begin{aligned} X(\mu) &= (X_i(\mu))_{i=0,\dots,N} \\ V(\mu) &= (V_i(\mu))_{i=0,\dots,N-1} \\ P(\mu) &= (P_i(\mu))_{i=0,\dots,N-1} \end{aligned}$$

La méthode de l'Etat Adjoint consiste à définir un Lagrangien $\Lambda$ pour chaque fonction coût $\Phi(\mu)$ en introduisant le vecteur des états adjoints $(\lambda_i)_{i=0,N-1}$ de la manière suivante :

$$\Lambda(\mu,\lambda) = \Phi(\mu, X(\mu), V(\mu), P(\mu)) + \sum_{i=0}^{N-1} {}^T \lambda_i R_i(\mu, X_i(\mu), V_i(\mu), P_i(\mu)) \tag{3.36}$$

Si $(V_i(\mu), P_i(\mu))$ vérifie les équation du problème $R_i(\mu, X_i, V_i, P_i) = 0$ à chaque incrément de temps ($i = 0,\dots, N-1$), nous avons :

$$\Lambda(\mu) = \Phi(\mu) \quad \text{et donc} \quad \frac{d\Lambda}{d\mu}(\mu) = \frac{d\Phi}{d\mu}(\mu) \qquad \forall \mu \tag{3.37}$$

Le calcul du gradient de $\Phi(\mu)$ est donc équivalent à celui de $\Lambda$. On omet les $(\mu)$ pour simplifier les notations. Aussi :

$$\frac{d\Lambda}{d\mu} = \frac{d\Phi}{d\mu} + \sum_{i=0}^{N-1} {}^T\lambda_i \frac{dR_i}{d\mu} \tag{3.38}$$

avec
$$\frac{d\Phi}{d\mu} = \frac{\partial\Phi}{\partial\mu} + \sum_{i=0}^{N} \frac{\partial\Phi}{\partial X_i}\frac{dX_i}{d\mu} + \sum_{i=0}^{N-1}\left(\frac{\partial\Phi}{\partial V_i}\frac{dV_i}{d\mu} + \frac{\partial\Phi}{\partial P_i}\frac{dP_i}{d\mu}\right) \tag{3.39}$$

et
$$\forall i = 0, N-1 \qquad \frac{dR_i}{d\mu} = \frac{\partial R_i}{\partial\mu} + \frac{\partial R_i}{\partial X_i}\frac{dX_i}{d\mu} + \frac{\partial R_i}{\partial V_i}\frac{dV_i}{d\mu} + \frac{\partial R_i}{\partial P_i}\frac{dP_i}{d\mu} \tag{3.40}$$

Considérons d'abord le cas où aucun remaillage n'est opéré, l'équation (3.11) peut être réécrite :

$$\forall i = 0, N \quad X_i = X_0 + \sum_{j=0}^{i-1} V_j \Delta t_j \tag{3.41}$$

La différentiation de cette équation nous donne :

$$\forall i = 0, N \quad \frac{dX_i}{d\mu} = \frac{dX_0}{d\mu} + \sum_{j=0}^{i-1} \frac{dV_j}{d\mu}\Delta t_j + \overbrace{\sum_{j=0}^{i-1} V_j \frac{d\Delta t_j}{d\mu}}^{\approx 0} \tag{3.42}$$

Nous supposons que le vecteur des paramètres n'a aucune influence sur le pas de temps $\left(\frac{d\Delta t_j}{d\mu} \approx 0\right)$, ce qui est le cas lorsque le pas de temps est constant, et qui est une bonne approximation autrement [*Balan 1996*] [*Vielledent 1999*].

L'égalité (3.42) est injectée dans les équations (3.39) et (3.40) dans le but d'éliminer les dérivées $\left(\frac{dX_i}{d\mu}\right)_{i=1,\ldots,N}$. Après quelques transformations, nous obtenons :

$$\begin{aligned}
\frac{d\Lambda}{d\mu} = {} & \frac{\partial\Phi}{\partial\mu} + \sum_{i=0}^{N}\frac{\partial\Phi}{\partial X_i}\frac{dX_0}{d\mu} + \sum_{i=0}^{N-1} {}^T\lambda_i\left(\frac{\partial R_i}{\partial\mu} + \frac{\partial R_i}{\partial X_i}\frac{dX_0}{d\mu}\right) + \sum_{i=0}^{N-1}\left(\frac{\partial\Phi}{\partial P_i} + {}^T\lambda_i\frac{\partial R_i}{\partial P_i}\right)\frac{dP_i}{d\mu} \\
& + \sum_{i=0}^{N-2}\left(\frac{\partial\Phi}{\partial V_i} + {}^T\lambda_i\frac{\partial R_i}{\partial V_i} + \Delta t_i\left(\sum_{j=i+1}^{N}\frac{\partial\Phi}{\partial X_j} + \sum_{j=i+1}^{N-1} {}^T\lambda_j\frac{\partial R_j}{\partial X_j}\right)\right)\frac{dV_i}{d\mu} \\
& + \left(\frac{\partial\Phi}{\partial V_{N-1}} + \Delta t_{N-1}\frac{\partial\Phi}{\partial X_N} + {}^T\lambda_{N-1}\frac{\partial R_{N-1}}{\partial V_{N-1}}\right)\frac{dV_{N-1}}{d\mu}
\end{aligned} \tag{3.43}$$

Le coût de calcul des termes $\frac{dV_i}{d\mu}$ et $\frac{dP_i}{d\mu}$ dépend du nombre de paramètres $\mu$. La méthode de l'état adjoint a pour but d'éviter cette dépendance au nombre de paramètres. A chaque

incrément de temps *i*, on choisit donc l'Etat Adjoint $\lambda_i$ permettant d'éliminer ces deux termes. Pour cela, le vecteur des états adjoints $(\lambda_i)_{i=0,...,N-1}$ doit vérifier les équations suivantes :

- élimination de $\frac{dV_i}{d\mu}$

$$\begin{cases} {}^T\lambda_{N-1}\dfrac{\partial R_{N-1}}{\partial V_{N-1}} = -\left(\dfrac{\partial \Phi}{\partial V_{N-1}} + \Delta t_{N-1}\dfrac{\partial \Phi}{\partial X_N}\right) \\ \forall i = N-2,...,0 \quad {}^T\lambda_i\dfrac{\partial R_i}{\partial V_i} = -\left(\dfrac{\partial \Phi}{\partial V_i} + \Delta t_i\sum_{j=i+1}^{N}\dfrac{\partial \Phi}{\partial X_j} + \Delta t_i\left(\sum_{j=i+1}^{N-1}{}^T\lambda_j\dfrac{\partial R_j}{\partial X_j}\right)\right) \end{cases} \tag{3.44}$$

- élimination de $\frac{dP_i}{d\mu}$

$$\forall i = N-1,...,0 \quad {}^T\lambda_i\frac{\partial R_i}{\partial P_i} = -\frac{\partial \Phi}{\partial P_i} \tag{3.45}$$

Afin de simplifier les notations, nous introduisons une variable intermédiaire $\Gamma$, définie par :

$$\forall i = -1,...,N-1 \quad \Gamma_i = \frac{\partial \Phi}{\partial X_N} + \sum_{j=i+1}^{N-1}\left(\frac{\partial \Phi}{\partial X_i} + {}^T\lambda_i\frac{\partial R_i}{\partial X_i}\right)^T \tag{3.46}$$

ou encore :

$$\begin{cases} \Gamma_{N-1} = \dfrac{\partial \Phi}{\partial X_N} \\ \forall i = N-1,...,0 \quad \Gamma_{i-1} = \Gamma_i + \dfrac{\partial \Phi}{\partial X_i} + {}^T\lambda_i\dfrac{\partial R_i}{\partial X_i} \end{cases} \tag{3.47}$$

En utilisant des variables compactées définies par $W_i = \begin{pmatrix} V_i \\ P_i \end{pmatrix}$ et $\overline{\Gamma}_i = \begin{pmatrix} \Gamma_i \\ 0 \end{pmatrix}$, les équations (3.44) et (3.45) peuvent être condensées en le système suivant :

$$\lambda_i = -\left(\frac{\partial R_i}{\partial W_i}\right)^{-1\,T}\left(\frac{\partial \Phi}{\partial W_i} + \Delta t_i\overline{\Gamma}_i\right) \quad \forall i = N-1,...,0 \tag{3.48}$$

L'équation (3.47) montre que la variable $\overline{\Gamma}_i$ est déterminée à partir de la variable $\overline{\Gamma}_{i+1}$. Il en résulte que $\overline{\Gamma}_i$ doit être calculée incrémentalement dans le sens inverse du temps, c'est-à-dire de $i = N - 1$ à $i = 0$. Le vecteur des états adjoints $\lambda_i$ doit donc être calculé de la même manière (équation (3.48)). Ceci représente la difficulté majeure pour implémenter la méthode de l'état adjoint dans le cadre de problèmes instationnaires [*Fourment et al. 2001*] [*Chung et al. 2003*]**.** Une fois le vecteur des états adjoints $(\lambda_i)_{i=0,...,N-1}$ calculé, on obtient :

$$\frac{d\Phi}{d\mu}(\mu) = \frac{\partial \Lambda}{\partial \mu}(\mu,\lambda) = \frac{\partial \Phi}{\partial \mu} + \sum_{i=0}^{N-1}{}^T\lambda_i\frac{\partial R_i}{\partial \mu} + \Gamma_{-1}\frac{dX_0}{d\mu} \tag{3.49}$$

où $X_0$ représente les coordonnées des nœuds du maillage initial de la simulation FORGE3®

Nous constatons que la méthode de l'état adjoint est caractérisée par :

- le fait que le vecteur des états adjoints est déterminé dans le sens inverse du temps, en commençant par le dernier incrément.
- le fait que l'analyse de sensibilité doit donc être réalisée après la simulation du procédé de forgeage. Elle nécessite donc le stockage des données de chaque incrément de calcul pour le calcul inverse.
- le fait qu'à chaque incrément de temps, elle nécessite la résolution d'autant de systèmes linéaires qu'il y a de fonctions coût. Le coût de la méthode est donc bien indépendant du nombre de paramètres.

### III.2.3.2. Remaillage et transport des variables adjointes

Considérons maintenant une simulation du forgeage avec des remaillages.

#### *III.2.3.2.1. Influences du remaillage sur les équations de l'Etat Adjoint*

Supposons qu'un remaillage intervienne à l'incrément $r$ $(0 \le r \le N-1)$ pour générer un nouveau maillage de meilleure qualité $\Omega_h^r{}'$. Notons $J_r$ l'opérateur linéaire de transport permettant de passer de l'ancienne à la nouvelle configuration :

$$X_r' = J_r X_r \tag{3.50}$$

On doit transporter le terme $\frac{dX}{d\mu}$ de l'ancien maillage vers le nouveau comme pour la méthode directe [*Balan 1996*]. La manière la plus simple est d'utiliser l'opérateur de transport par interpolation inverse $J$ déjà utilisé dans FORGE3® :

$$\frac{dX_r'}{d\mu} = \underbrace{\frac{dX_r'}{dX_r}}_{J_r} \frac{dX_r}{d\mu} \tag{3.51}$$

Pour $r \le i$, l'équation (3.42) doit être remplacée par :

$$\forall i = r,\ldots,N-1 \qquad \frac{dX_i}{d\mu} = J_r\left(\frac{dX_0}{d\mu} + \sum_{j=0}^{r-1} \frac{dV_j}{d\mu}\Delta t_j\right) + \sum_{j=r}^{i-1} \frac{dV_j}{d\mu}\Delta t_j \tag{3.52}$$

Cette modification transforme l'équation (3.43) en la suivante :

$$\begin{aligned}\frac{d\Lambda}{d\mu}(\mu) &= \frac{\partial \Phi}{\partial \mu} + \sum_{i=0}^{N-1} {}^T\lambda_i \frac{\partial R_i}{\partial \mu} + \left[\Gamma_{-1} - \Gamma_{r-1}(1-J_r)\right]\frac{dX_0}{d\mu} \\ &+ \sum_{i=0}^{r-1}\left(\frac{\partial \Phi}{\partial V_i} + {}^T\lambda_i \frac{\partial R_i}{\partial V_i} + \Delta t_i\left[\Gamma_i - \Gamma_{r-1}(1-J_r)\right]\right)\frac{dV_i}{d\mu} \\ &+ \sum_{i=r}^{N-1}\left(\frac{\partial \Phi}{\partial V_i} + {}^T\lambda_i \frac{\partial R_i}{\partial V_i} + \Delta t_i \Gamma_i\right)\frac{dV_i}{d\mu} + \sum_{i=0}^{N-1}\left(\frac{\partial \Phi}{\partial P_i} + {}^T\lambda_i \frac{\partial R_i}{\partial P_i}\right)\frac{dP_i}{d\mu}\end{aligned} \tag{3.53}$$

Nous constatons que l'opérateur de transport s'applique uniquement à la variable $\Gamma$. S'il y a un remaillage à l'incrément $i$, il suffit de modifier le calcul de $\Gamma$ comme suit pour que l'équation (3.47) reste valable :

$$\forall i = N-1,...,0 \quad \begin{cases} \Gamma_{i-1} = \Gamma_i + \dfrac{\partial \Phi}{\partial X_i} + {}^T\lambda_i \dfrac{\partial R_i}{\partial X_i} & \left[\text{pas de remaillage à l'incrément } i\right] \\ \Gamma_{i-1} = \left(\Gamma_i + \dfrac{\partial \Phi}{\partial X_i} + {}^T\lambda_i \dfrac{\partial R_i}{\partial X_i}\right) J_i & \left[\text{remaillage à l'incrément } i\right] \end{cases} \tag{3.54}$$

D'un point de vue pratique, les remarques suivantes sont faites :

- Premièrement, la dérivée $\dfrac{dX_0}{d\mu}$ n'est utilisée qu'à l'incrément 0, donc elle n'a pas besoin d'être transportée au cours des remaillages comme dans la méthode de différentiation directe.

- Deuxièmement, on constate que l'opérateur de transport intervient sous sa forme transposée ${}^TJ$, pour calculer la variable ${}^T\Gamma$ ($\Gamma$ intervient lui aussi sous sa forme transposée dans le calcul de la variable adjointe $\lambda$ - voir l'équation (3.48)). S'il y a remaillage à l'incrément $r$, nous avons :

$$\begin{aligned} &\Gamma_{r-1} = \Gamma_{r-1}' J_r \\ \text{et donc} \quad &{}^T\Gamma_{r-1} = {}^TJ_r \, {}^T\Gamma_{r-1}' \end{aligned} \tag{3.55}$$

Il faut remarquer que le calcul des variables $\Gamma_i$ et $\lambda_i$ se fait dans le sens inverse du temps, alors que l'opérateur de transport $J_r$ va de l'ancienne configuration $\Omega_h$ vers la nouvelle $\Omega_h'$.

Remarque : Lorsque des remaillages sont effectués, il faut stocker toutes les données concernant l'ancien et le nouveau maillage (connectivités volumiques et surfaciques, nombre de nœuds, d'éléments, de faces, …). Ce stockage est impératif car ces informations sont utilisées lors de la résolution "adjointe".

#### *III.2.3.2.2. Transport des variables adjointes*

Il faut noter d'abord que le transport des données est réalisé par interpolation inverse. L'équation (3.50) est donc réécrite sous sa forme scalaire :

$$\forall k' \in \Omega', \quad X'^{k'} = \sum_{l \in \vartheta(k')} J^{k'l} X^{l} \tag{3.56}$$

où $\vartheta(k')$ représente l'ensemble des nœuds $l$ de l'élément $e$ de l'ancien maillage $\Omega$ qui contient le nœud $k'$ du nouveau maillage $\Omega'$.

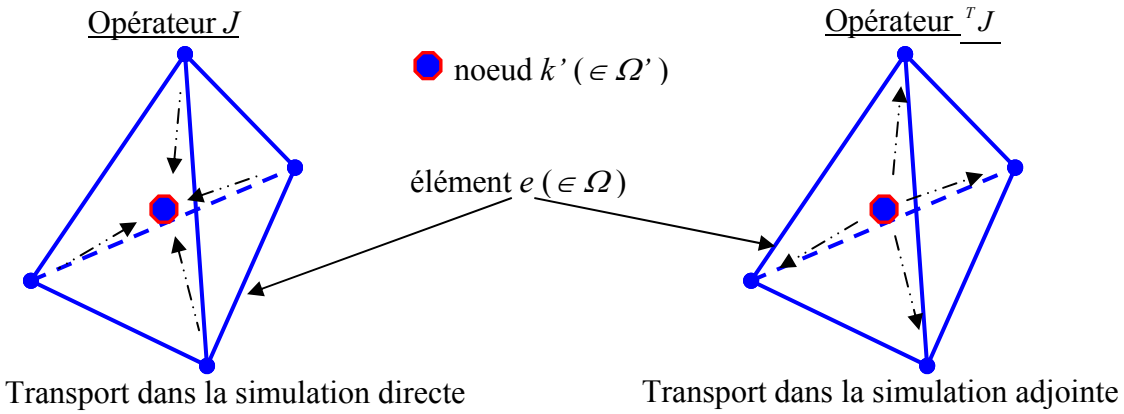



*Figure 3.9. Opérateur de transport pour la simulation directe et le calcul de l'Etat Adjointe*

Dans le cadre de la méthode de l'état adjoint, nous voulons transporter la variable nodale ${}^{T}\Gamma$ (variable P1) du nouveau maillage $\Omega'$ vers l'ancien maillage $\Omega$. L'équation (3.55) s'écrit :

$$\forall l \in \Omega, \quad {}^{T}\Gamma^{l} = \sum_{\substack{k' \in \Omega' \\ \text{tel que: } l \in \vartheta(k')}} J^{k'l\,T}\,\Gamma'^{k'} \tag{3.57}$$

La valeur $\Gamma'^{k'}$ au nœud $k'$ est connue car lors du calcul de l'adjoint, $\Omega'$ est le maillage courant (*Figure 3.9*). On peut trouver l'algorithme pour calculer la valeur de ${}^{T}\Gamma$ sur le maillage $\Omega$ dans [*Laroussi 2003*].

Le coût de calcul, ainsi que la précision de cet opérateur ${}^{T}J$, sont tout à fait similaires à ceux de l'opérateur $J$ de la simulation directe.

### III.2.3.3. Technique semi-analytique pour calculer les dérivées partielles

Pour calculer les différentes dérivées partielles qui apparaissent dans les équations de la méthode de l'état adjoint ($\frac{\partial \Phi}{\partial \mu}$, $\frac{\partial R_i}{\partial \mu}$, $\frac{\partial R_i}{\partial X_i}$, $\frac{\partial \Phi}{\partial X_i}$ et $\frac{\partial \Phi}{\partial W_i}$), nous retenons une technique semi-analytique. Elle présente l'avantage d'être flexible, facile à implémenter et précise. Nous utilisons des quotients différentiels décentrés à droite, comme par exemple :

$$\forall j = 1,...,nbpar \quad \frac{\partial \Phi}{\partial \mu_j} = \lim_{\|\Delta\mu_j\| \to 0} \frac{\Phi(\mu + \Delta\mu_j, X, W) - \Phi(\mu, X, W)}{\|\Delta\mu_j\|} \tag{3.58}$$

où
$$\forall j = 1,...,nbpar \quad \Delta\mu_j = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \varepsilon \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ \vdots \\ j \\ \vdots \\ nbpar \end{matrix} \tag{3.59}$$

où $\varepsilon$ est une petite perturbation prise en général égale à $10^{-6}$ en valeur relative.

Le coût de calcul des dérivées $\frac{\partial \Phi}{\partial \mu}$ et $\frac{\partial R_i}{\partial \mu}$ n'est pas grand car il est proportionnel au nombre de paramètres (qui est petit devant le nombre d'éléments du maillage). En revanche, pour le calcul des dérivées $\frac{\partial \Phi}{\partial X_i}, \frac{\partial \Phi}{\partial W_i}$ et $\frac{\partial R_i}{\partial X_i}$, le coût est proportionnel au nombre de nœuds du maillage si la programmation est effectuée de la même manière. Il résulte que pour un maillage moyen ce coût devient rapidement exorbitant. Pour ce problème, une méthode de programmation basée sur la décomposition de $\Phi$ et $R_i$ au niveau des éléments et des faces du maillage a été retenue. Le point clé consiste à remarquer que lorsqu'on perturbe un nœud *k* du maillage, l'influence de cette perturbation se limite aux éléments et aux faces contenant ce nœud *k*. La perturbation est locale et l'assemblage n'est fait que pour les éléments appartienne le nœud k.

Le surcoût du calcul des dérivées par la méthode semi-analytique est ainsi indépendant du nombre de nœuds. Ce coût est extrêmement raisonnable puisqu'il ne traduit que par un nombre limité d'assemblages de résidu et d'évaluations de la fonction coût. Il est très largement inférieur au coût de calcul nécessaire à la résolution d'un système linéaire 3D.

Notons enfin que la dérivée $\frac{\partial R_i}{\partial X_i}$ est une matrice dont le nombre de coefficients est proportionnel à (*nbnoe*)². Stocker une telle matrice est très vite problématique, dès que le nombre de nœuds devient grand. Or, seul le produit de cette dérivée avec le vecteur des états adjoint $\lambda_i$ apparaît dans les équations (voir équation (3.47)). Ainsi, on calcule directement la quantité ${}^T\lambda_i \frac{\partial R_i}{\partial X_i}$ au niveau d'un élément ou d'une face, comme expliqué précédemment.

### III.2.3.4. Traitement des termes de contact et de frottement

Considérons le calcul du terme $\frac{\partial R}{\partial X}$ par la méthode semi-analytique. On décompose le résidu en vitesse en différentes contributions comme suit :

$$R = R_{rheo} + R_{frott} + R_{incomp} + R_{contact} \tag{3.60}$$

avec : $R_{rheo} = \int_\Omega s : \dot{\varepsilon}^* dw$ est le terme volumique contenant la loi de comportement du matériau,

$R_{incomp} = -\int_\Omega p\, div v^* dw$ est le terme volumique lié à la condition d'incompressibilité

$R_{frott} = -\int_{\partial\Omega} 1_{\partial\Omega_f}\, \tau . v^* ds$ est le terme surfacique contenant la loi de frottement $\tau$

$R^k_{contact}$ est la contribution du contact au nœud $k$, donnée par

$$R^k_{contact} = \rho 1_{\partial\Omega_c}(k)\left[ (V_k - v_{outil}).n_k - \frac{\delta_k}{\Delta t} \right] n_k S_k \qquad \forall k \in \partial\Omega_h \tag{3.61}$$

Pour calculer les dérivées $\frac{\partial R_{contact}}{\partial X}$ et $\frac{\partial R_{frott}}{\partial X}$, nous devons différentier l'inégalité de contact unilatéral par rapport aux coordonnées du maillage. Ce problème a été abordé par de nombreux auteurs [*Balan 1996*] [*Chung et al. 1998*] [*Vieilledent 1999*] [*Zabaras et al. 2000*] [*Stupkiewicz et al. 2002*] [*Chung et al. 2002*] [*Stupkiewicz et al. 2003*]. Les fonctions indicatrices du contact et du frottement apparaissent dans les termes de la condition de contact unilatéral sont des fonctions de type « partie positive ». La particularité de ces fonctions est qu'elles peuvent présenter des points de non-dérivabilité, comme le montre la *Figure 3.10* avec la fonction $[x]^+$ non dérivable en 0 :

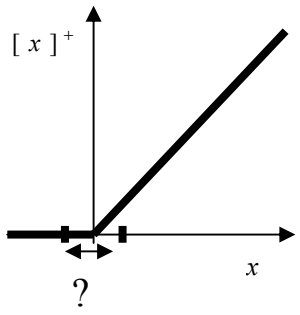



*Figure 3.10. Fonction $[x]^+$ non dérivable en 0 et pour laquelle la dérivée à droite est choisie*

On voit bien que $[x]^+$ accepte une dérivée à droite et une dérivée à gauche. On choisit la dérivée à droite pour $1_{\partial\Omega_c}$ et $1_{\partial\Omega_f}$ avec les raisonnements suivants:

- physiquement, après perturbation, le nœud a plus de raisons de rester en contact que de quitter systématiquement le contact. On suppose donc que la surface de contact $\partial\Omega_c$ reste inchangée ainsi on retient le choix des travaux antérieurs de plusieurs auteurs : [*Balan 1996*] [*Chung et al. 1998*] [*Vieilledent 1999*] [*Zabaras et al. 2000*] [*Stupkiewicz et al. 2002*] [*Chung et al. 2002*] [*Stupkiewicz et al. 2003*] et [*Laroussi 2003*]. Cela revient à considérer la condition de contact unilatéral

comme étant bilatérale au moment de la différentiation. L'inégalité devient égalité et on a :

$$\frac{\partial \mathbf{1}_{\partial\Omega_c}}{\partial \mathrm{X}} = \frac{\partial \mathbf{1}_{\partial\Omega_f}}{\partial \mathrm{X}} = 0 \tag{3.62}$$

- ce choix est cohérent avec celui du terme $\frac{\partial [h]^+}{\partial v}$, effectué pour calculer la contribution du contact à la matrice hessienne dans FORGE3® au sein de l'algorithme de Newton-Raphson généralisé (au contact unilatéral).

On écrit maintenant le résidu en vitesse en faisant intervenir la dépendance implicite des fonctions indicatrices du contact et du frottement, ainsi que la distance de contact $\delta$ et la normale $n$ comme suit :

$$R(\mu, X, W)) = \widetilde{R}(X, W, 1_{\partial\Omega_c}, 1_{\partial\Omega_f}, \delta, n, \mu) \tag{3.63}$$

où $\quad 1_{\partial\Omega_c} = 1_{\partial\Omega_c}(\mu, X, W), \quad 1_{\partial\Omega_f} = 1_{\partial\Omega_f}(\mu, X), \quad \delta = \delta(\mu, X), \quad n = n(\mu, X)$ (3.64)

Nous sommes en mesure d'écrire la dérivée $\frac{\partial R}{\partial X}$ comme suit :

$$\frac{\partial \mathrm{R}}{\partial \mathrm{X}} = \left.\frac{\partial R}{\partial X}\right|_{\mathbf{1}_{\partial\Omega_c}, \mathbf{1}_{\partial\Omega_f}, \delta, n} + \frac{\partial \mathrm{R}}{\partial \delta}\frac{\partial \delta}{\partial \mathrm{X}} + \frac{\partial \mathrm{R}}{\partial \mathrm{n}}\frac{\partial \mathrm{n}}{\partial \mathrm{X}} + \underbrace{\frac{\partial \mathrm{R}}{\partial \mathbf{1}_{\partial\Omega_c}}\frac{\partial \mathbf{1}_{\partial\Omega_c}}{\partial \mathrm{X}}}_{=0} + \underbrace{\frac{\partial \mathrm{R}}{\partial \mathbf{1}_{\partial\Omega_f}}\frac{\partial \mathbf{1}_{\partial\Omega_f}}{\partial \mathrm{X}}}_{=0} \tag{3.65}$$

Ainsi, lorsque l'on calcule le terme $\frac{\partial R}{\partial X}$ par la technique semi-analytique en effectuant des perturbations locales sur $X$ :

- la condition de contact ne change pas, donc les surfaces de contact $\partial\Omega_c$ et de frottement $\partial\Omega_f$ restent inchangées

- seules les distances de contact $\delta$ et les normales $n$ sont perturbées, et sont à recalculer

Pour le calcul du terme $\frac{\partial R}{\partial \mu}$, le même raisonnement est appliqué:

$$\frac{\partial R}{\partial \mu} = \left.\frac{\partial R}{\partial \mu}\right|_{1_{\partial\Omega_c}, 1_{\partial\Omega_f}, \delta, n} + \frac{\partial R}{\partial \delta}\frac{\partial \delta}{\partial \mu} + \frac{\partial R}{\partial n}\frac{\partial n}{\partial \mu} + \underbrace{\frac{\partial R}{\partial 1_{\partial\Omega_c}}\frac{\partial 1_{\partial\Omega_c}}{\partial \mu}}_{=0} + \underbrace{\frac{\partial R}{\partial 1_{\partial\Omega_f}}\frac{\partial 1_{\partial\Omega_f}}{\partial \mu}}_{=0} \tag{3.66}$$

Si la fonction coût $\Phi$ fait intervenir des termes de contact et/ou de frottement, pour calculer les termes $\frac{\partial \Phi}{\partial \mu}, \frac{\partial \Phi}{\partial X} et \frac{\partial \Phi}{\partial W}$, on adoptera la même démarche que pour le calcul de $\frac{\partial R}{\partial X}$, en utilisant la décomposition suivante :

$$\Phi = \Phi_{volumique} + \Phi_{frott} + \Phi_{contact} \tag{3.67}$$

où $\Phi_{volumique}$ est la contribution volumique de $\Phi$

$\Phi_{frott}$ est la contribution due au frottement de $\Phi$

$\Phi_{contact}$ est la contribution due au contact de $\Phi$

# III.3. CALCUL DU GRADIENT POUR LE FORGEAGE MULTI-PASSES ET POUR L'OPTIMISATION DE FORME D'OUTILS

## III.3.1. Problème lié au type de paramètres de forme du lopin initial : contrainte de volume constant à traiter et spécifique

Dans son travail de thèse, Laroussi [*Laroussi 2003*] a implémenté une première version de la méthode de l'état adjoint décrite précédemment dans le logiciel FORGE3® pour certaines fonctions coûts ($\Phi_{ener}$ énergie totale…). Cela lui a permis d'obtenir le gradient par rapport aux paramètres de type "forme du lopin initial" et avec une bonne précision. Cela nous permet d'utiliser les algorithmes à base de gradient (BFGS, SCPIP, …) ainsi que des algorithmes hybrides (AG-MGO, AG-MGA…) pour résoudre le problème d'optimisation d'engrenage qui va être décrit dans la partie d'application.

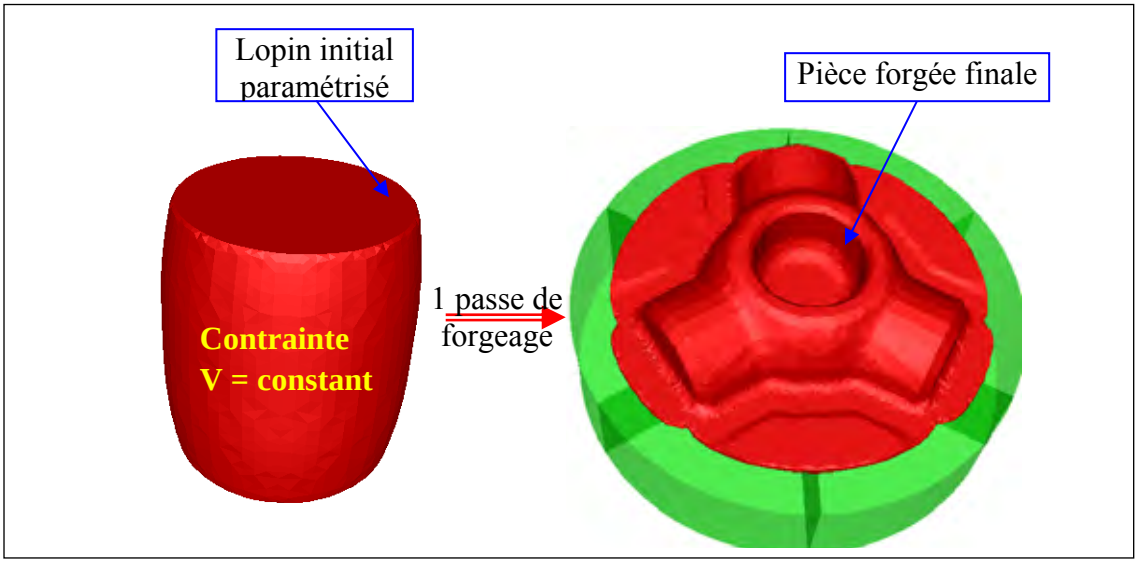



*Figure 3.11. Exemple du cas forgeage à 1 passe avec le lopin initial paramétré*

Cependant, ce type de paramètres est lié par la contrainte de volume constant, qui n'est pas facile à traiter. La méthode la plus simple est d'éliminer un des paramètres en utilisant la fonction analytique qui relie les paramètres. Cette fonction est facile à expliciter si la pièce est de forme simple (axisymétrique,…) et paramétrée par des polygones de bas degré. Au contraire, si la pièce est de forme compliquée (vrai 3D) ou paramétrée par des courbes de haut degré (par exemple une Bspline), la fonction analytique ne peut être explicitée. Cela n'empêche pas de la prendre en compte [*Vieilledent 1999*] mais le problème d'optimisation, et sa solution, dépendent du choix du paramètre éliminé, ce qui n'est pas parfaitement satisfaisant.

De plus, avec l'approche retenue dans [*Laroussi 2003*] le calcul du terme $\frac{dX_0}{d\mu}$ est très spécifique et dépendant de chaque cas d'optimisation. Le calcul du gradient des fonctions coûts par rapport au paramètre "forme du lopin initial" est donc "spécifique" pour chaque problème. Nous ne pouvons pas utiliser ce calcul d'un problème à l'autre.

Eviter la contrainte de volume constant et s'affranchir les spécificités du calcul du gradient est donc envisagée dans ce contexte. Cela constitue donc une contribution importante de cette thèse.

## III.3.2. Paramétrisation des outils de préforme

Pour pouvoir aborder une gamme plus vaste de problèmes d'optimisation, nous nous orientons vers un autre type de paramètre de forme : la forme des outils dans le cadre du forgeage multi-passes.

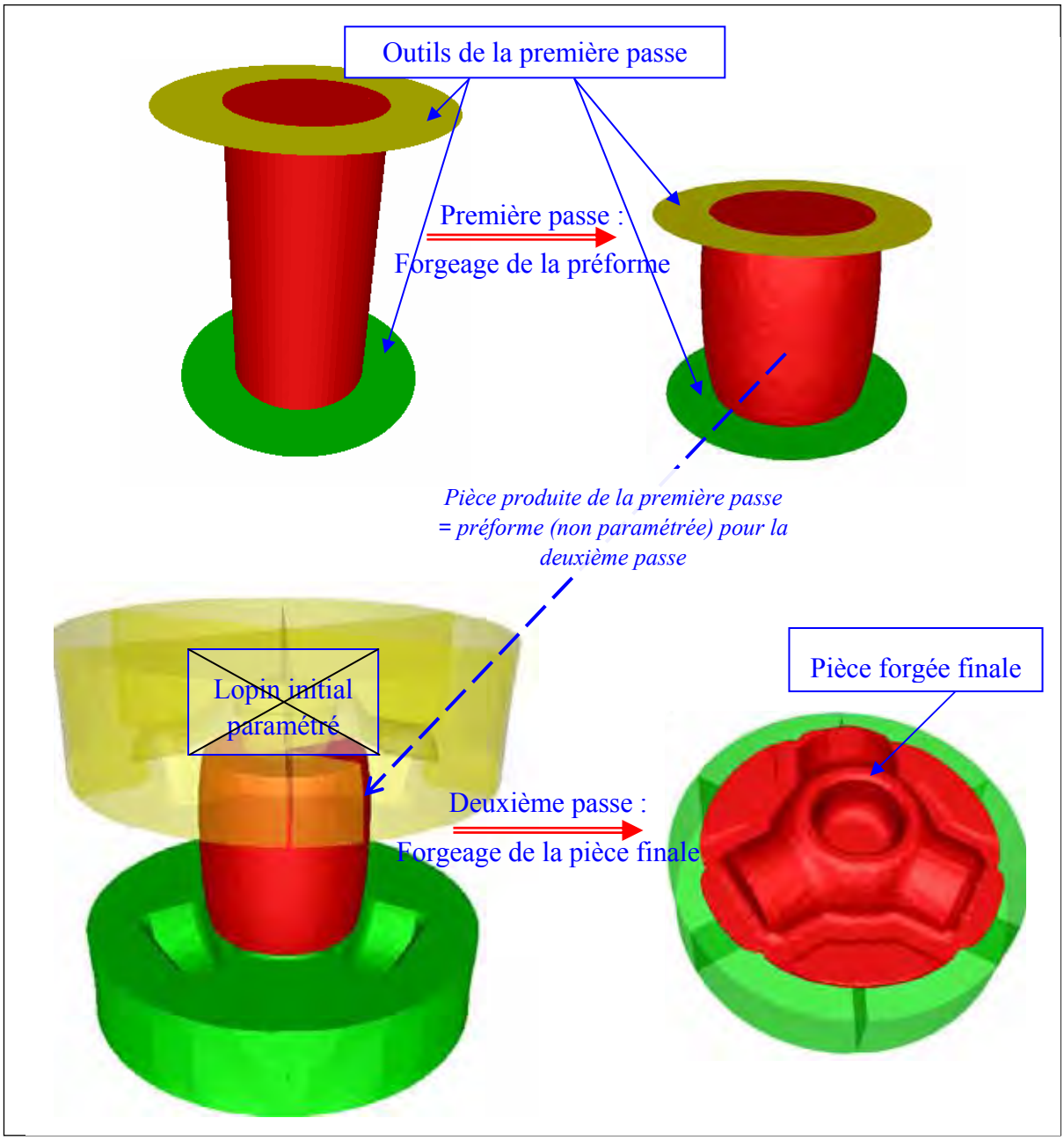



*Figure 3.12. Exemple de forgeage en 2 passes avec outils de préforme paramétrés*

Le forgeage multi-passes consiste à réaliser une pièce de forme compliquée en plusieurs opérations et avec des outils différents. La *Figure 3.12* nous montre un exemple en deux

passes pour forger un triaxe à partir d'un lopin initial cylindrique. La pièce produite à l'étape actuelle est la préforme de l'étape suivante. La complexité des outils augmente à chaque opération de forgeage. Cette méthode de forgeage permet d'obtenir des pièces de forme compliquées à partir des lopins initiaux de formes très simples comme par exemple un lopin cylindrique (*Figure 3.12*).

Dans le cadre du forgeage en deux passes, la forme des outils de préforme est paramétrisée en considérant celle du lopin initial (lopin de la première opération de forgeage) comme fixée. Pour les cas avec un nombre d'opération supérieur à deux, les outils paramétrisés pourraient être ceux de la première opération ou ceux des opérations intermédiaires, selon la nature du problème.

La forme des outils de forgeage peut être paramétrée par deux approches selon sa complexité :

- par de vraies surfaces 3D, si la forme des outils est très complexe,
- par des courbes ou des polygones si l'outil est axisymétrique ou de forme régulière suivante une direction quelconque.

La paramétrisation des outils 3D est compliquée et laborieuse. De plus, dans de nombreux cas, la préforme est axisymétrique et seulement la forme finale est véritablement 3D. Dans un premier temps, nous considérons la paramétrisation des outils de forgeage de forme axisymétrique. Nous avons donc choisi les courbes Bsplines d'ordre 4 (degré 3) [*Vielledent 1999*] pour paramétrer ces formes.

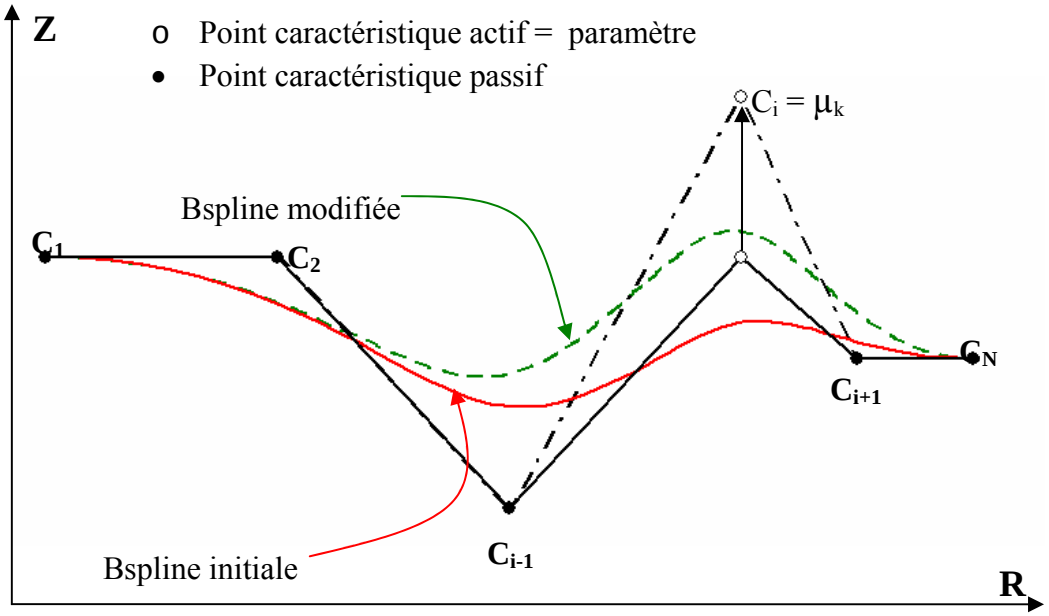



*Figure 3.13. Exemple de courbe Bspline initiale et de courbe Bspline modifiée*

Une courbe Bspline est définie par un contour polygonal comportant $N$ ($N \geq 4$) sommets $C_1,\ldots, C_N$, ou points caractéristiques ou encore points de contrôle. Ces sommets peuvent être fixes ou mobiles. Les sommets mobiles sont les paramètres d'optimisation. Le mouvement de ces sommets peut s'effectuer dans une direction quelconque, mais nous nous limitons dans le cas où les sommets ne se déplacent que dans la direction Z. La *Figure 3.13* présente un

exemple de courbes Bsplines : la courbe initiale (en rouge) et sa forme modifiée (en verte) après changement de la position d'un point de contrôle. Les Bsplines sont choisies car elles présentent certains avantages : elles sont continues et souples, nécessitent peu de paramètres pour définir des géométries complexes en 2D, et elles sont faciles à contrôler.

Plusieurs étapes doivent encore être effectuées pour créer un outil depuis un contour polygonal initial. La *Figure 3.14* présente un exemple de ces étapes lors de la création d'un outil de préforme pour le forgeage d'un triaxe. Grâce à la symétrie du triaxe, nous étudions seulement un sixième de la moitié supérieur du triaxe. A partir d'un contour polygonal donné (*Figure 3.14.a*) qui contient plusieurs points de contrôle, certains étant des paramètres de forme, nous créons d'abord la Bspline correspondante (*Figure 3.14.b*). Ensuite, il faut effectuer une extrusion de la courbe Bspline suivant la direction axisymétrique et créer une matrice de connectivité pour générer le maillage de surface en 3D de l'outil (*Figure 3.14.c*). Puis, il faut lier les paramètres de contrôle à ce maillage pour générer un nouvel outil de forgeage (*Figure 3.14.d*). Enfin, cet outil sera utilisé dans le calcul du forgeage pour la valeur des fonctions coûts.

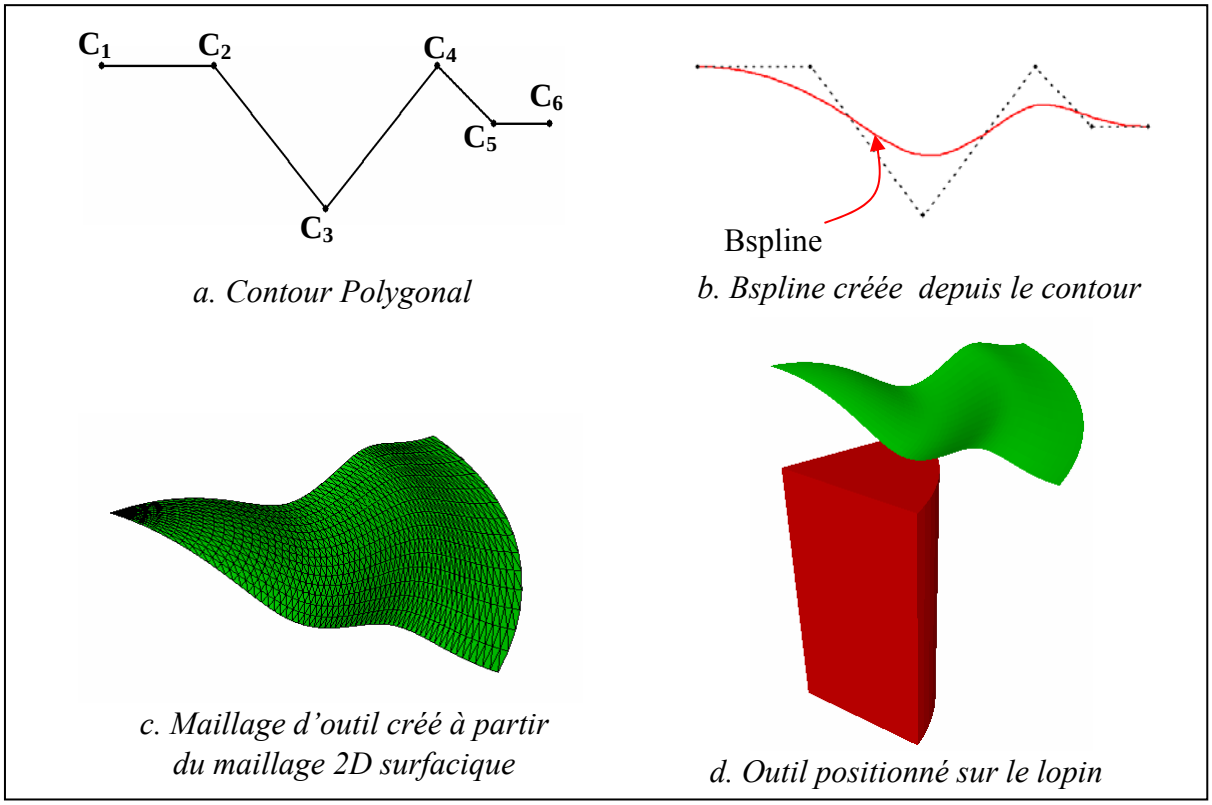



*Figure 3.14. Création de l'outil de préforme pour le forgeage d'un triaxe à partir d'un contour polygonal*

### III.3.3. Calcul du gradient par rapport au paramètre de forme des outils

Les équations de base de la méthode de l'Etat Adjoint sont les mêmes lorsque l'on calcule le gradient des fonctions coûts par rapport au paramètre "forme des outils". Les changements interviennent dans l'expression de la dérivée totale.

Reprenons l'équation (3.49) pour calculer la dérivée totale d'une fonction coût $\Phi$, nous avons :

$$\frac{d\Phi}{d\mu}(\mu) = \frac{\partial \Phi}{\partial \mu} + \sum_{i=0}^{N-1} {}^{T}\lambda_i \frac{\partial R_i}{\partial \mu} + \Gamma_{-1} \frac{dX_0}{d\mu} \tag{3.68}$$

où rappelons-le $X_0$ représente les coordonnées des nœuds du maillage du lopin initial.

Le calcul des gradients présenté précédemment est assez spécifique aux paramètres de forme du lopin initial. La modification d'un paramètre $\mu$ quelconque dans le jeu des paramètres actuel engendre le changement de toutes les coordonnées $X_0$ du maillage du lopin initial. Par contre, le résidu $R_i$ ne dépend pas explicitement de ce type de paramètres :

$$\frac{dX_0}{d\mu} \neq 0 \quad et \quad \frac{\partial R_i}{\partial \mu} = 0 \tag{3.69}$$

La dérivée totale de $\Phi$ est alors donnée par :

$$\frac{d\Phi}{d\mu}(\mu) = \frac{\partial \Phi}{\partial \mu} + \Gamma_{-1} \frac{dX_0}{d\mu} \tag{3.70}$$

Pour le forgeage multi-passes, en revanche on optimise la forme des outils de forgeage en considérant celle du lopin initial comme fixée ($X_0$ fixée). Cette fois, le résidu dépend explicitement des paramètres, mais pas de la préforme initiale et nous avons :

$$\frac{\partial R_i}{\partial \mu} \neq 0 \quad et \quad \frac{dX_0}{d\mu} = 0 \tag{3.71}$$

L'équation de la dérivée totale de $\Phi$ devient dans ce cas :

$$\frac{d\Phi}{d\mu}(\mu) = \frac{\partial \Phi}{\partial \mu} + \sum_{i=0}^{N-1} {}^{T}\lambda_i \frac{\partial R_i}{\partial \mu} \tag{3.72}$$

Nous devons donc maintenant calculer $\frac{\partial R_i}{\partial \mu}$ qui intervient dans la dérivée totale de $\Phi$. Ce terme est lui aussi calculé par différences finies. Dans (3.60), un seul terme de $R_i$ dépend explicitement de la géométrie des outils, c'est le terme de contact (3.61). En effet, les distances de contact $\delta_k(X,\mu)$ et les normales $n_k(X,\mu)$ dépendent explicitement de la variation du paramètre $\mu$. Ainsi, la dérivée du résidu à chaque incrément $i$ se réduit à celle du terme de contact :

$$\frac{\partial R_i}{\partial \mu} = \frac{\partial R_{i,contact}}{\partial \mu} \tag{3.73}$$

Nous calculons donc le terme $\frac{\partial R_{i,contact}}{\partial \mu}$ à chaque incrément $i$ par la méthode de différences finies. Pour cela, on effectue des perturbations des différentes composantes de $\mu$ d'une

quantité $\Delta\mu$ en supposant que les surfaces de contact $\partial\Omega_c$ et de frottement $\partial\Omega_f$ restent inchangées. On recalcule seulement les distances de contact $\delta_k^{pert}(X,\mu+\Delta\mu)$ et les normales $n_k^{pert}(X,\mu+\Delta\mu)$ perturbées en effectuant une nouvelle analyse de contact pour chaque configuration d'outillage perturbée. Toutefois, la perturbation des paramètres de forme des outils ainsi que l'analyse de contact à un incrément i du calcul de l'Etat Adjoint sont délicates. Elles nécessitent des précautions qui sont décrites sur la *Figure 3.15*.

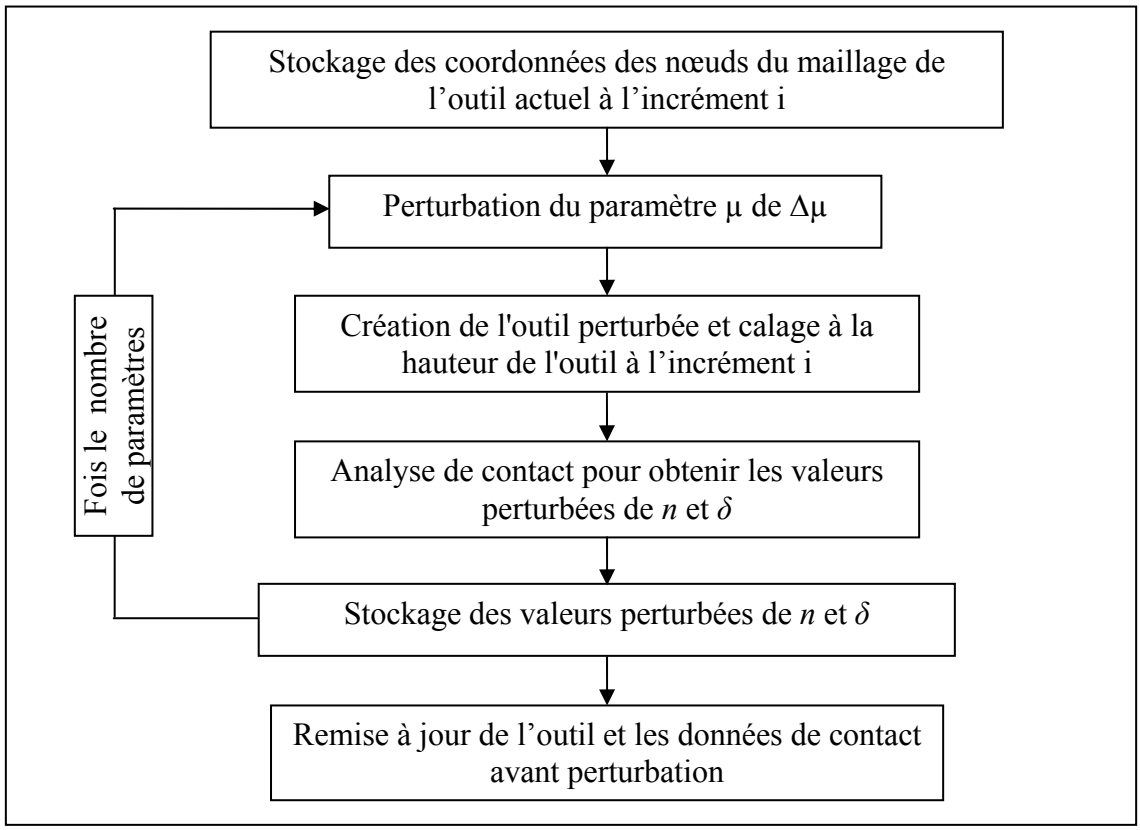



*Figure 3.15. Algorithme pour calculer les valeurs perturbées de n et δ*

Une fois les valeurs perturbées de *n* et *δ* obtenues, la valeur de $\frac{\partial R_i^k}{\partial\mu}$ est calculée par :

$$\forall k = 1,...,Nbnoe;\forall i = N-1,...,0$$

$$\frac{\partial R_i^k}{\partial\mu} = \frac{\partial R_{i\,contact}^k}{\partial\mu} = \frac{\rho 1_{\partial\Omega_c}(k)\left\{\left[\left(V_k - V_{outil}\right).n_k^{pert} - \frac{\delta_k^{pert}}{\Delta t}\right]n_k^{pert} - \left[\left(V_k - V_{outil}\right).n_k - \frac{\delta_k}{\Delta t}\right]n_k\right\}S_k}{\Delta\mu} \tag{3.74}$$

En ce qui concerne $\frac{\partial\Phi}{\partial\mu}$ et les fonctions coûts utilisées, le terme $\Phi_{contact}$ de (3.67) est nul, de sorte que $\frac{\partial\Phi}{\partial\mu}=0$. Dans un cas plus général $\frac{\partial\Phi_{contact}}{\partial\mu}$ serait calculé de la même manière que $\frac{\partial R_{contact}}{\partial\mu}$

### III.3.4. Transport des variables de l'état adjoint entre deux opérations de forgeage

Comme nous l'avons mentionné précédemment, le calcul de l'état adjoint est effectuée dans l'ordre inverse du temps. Il n'est déclenché que lorsque toutes les simulations directes du FORGE3® sont terminées.

Pour le forgeage en une passe, la gestion des variables de l'Etat Adjoint est présentée dans [*Laroussi 2003*]. En revanche, pour le forgeage multi-passes, la dérivée totale est le résultat de plusieurs calculs séquentiels de l'état adjoint (chaque opération de forgeage est un calcul adjoint complet comme pour le forgeage en une passe). Donc, nous devons transporter les variables entre deux opérations séquentielles. Lors de l'analyse les équations, nous constatons que seulement la variable $\Gamma$ avec la valeur de la dérivée totale $\left.\frac{d\Phi}{d\mu}\right|^{K}$ après chaque opération K ont besoin d'être transportées. Supposons que le forgeage consiste en M passes. Après un calcul de l'état adjoint de la passe K, pour poursuivre avec celui à la passe K-1, les données sont transportées en plusieurs étapes comme présentées sur la *Figure 3.16*.

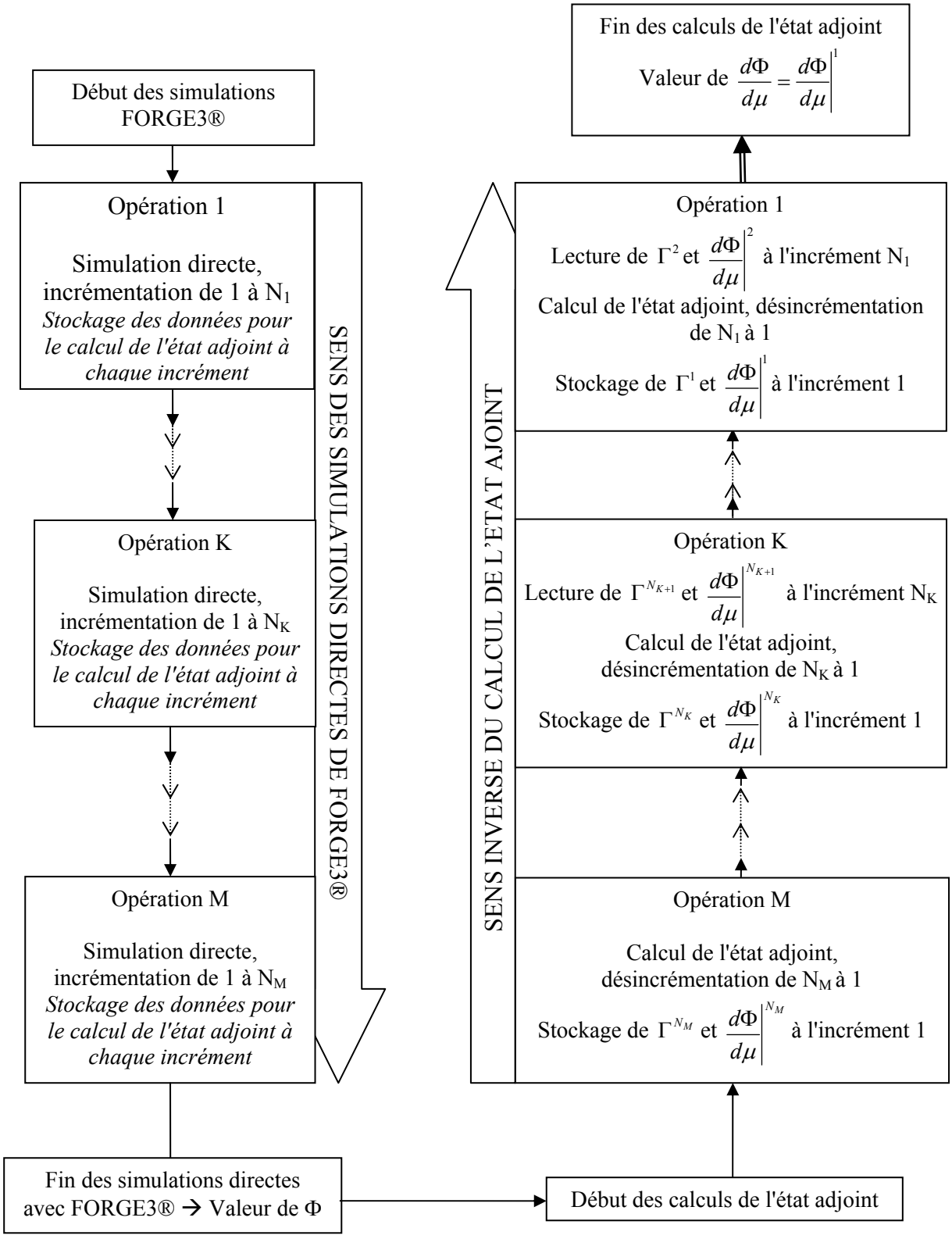



*Figure 3.16. Sens des simulations directes et des calculs de l'état adjoint dans un cas de forgeage composé de M passes avec transfert des données entre deux passes.*

## III.3.5. Validation du calcul du gradient pour le forgeage multi-passes

La validation du calcul de ces dérivées par rapport aux paramètres de type "préforme du lopin initial" a été faite dans [*Laroussi 2003*]. Ici, nous présentons la validation du calcul du gradient par rapport au paramètre de la forme des outils, dans le cas du forgeage multi-passes, avec outil axisymétrique paramétré par des fonctions Bsplines.

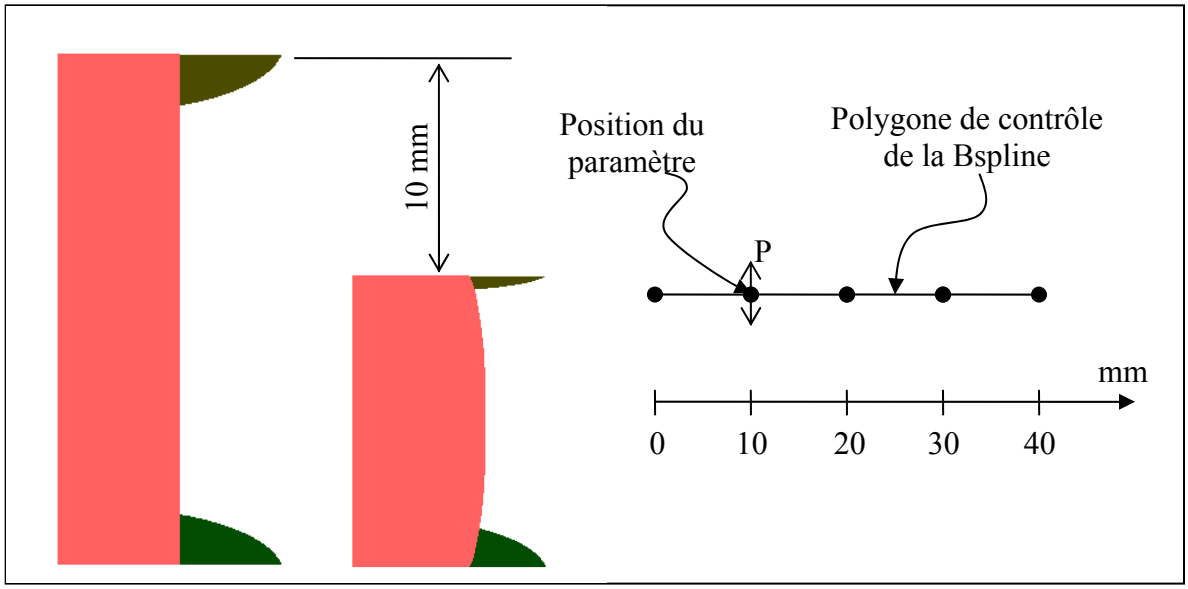



*Figure 3.17. Test d'écrasement d'un lopin cylindrique et position du paramètre sur le polygone de contrôle de la Bspline*

Nous validons d'abord le nouveau type de paramètre "forme d'outil" en réalisant un test d'écrasement d'un lopin cylindrique. Comme ce lopin est symétrique, nous simulons seulement l'écrasement d'un sixième de ce lopin. La paramétrisation de l'outil avec les courbes Bspline a été présentée dans la section *III.3.2*. La position du paramètre considéré est présentée sur la *Figure 3.17* (le rayon du lopin est de 20mm). La course d'écrasement est de 10mm (la hauteur du lopin est de 90mm). La valeur du gradient obtenue est comparée à celle d'un calcul par différences finies.

Pour valider le calcul du gradient dans le cas multi-passes, nous décomposons le test d'écrasement précédent en deux opérations de forgeage, chaque passe ayant une course de forgeage de 5mm (*Figure 3.18*). Dans ce cas, nous paramétrons les outils des deux opérations de forgeage. Si le calcul du gradient est précis, la valeur calculée doit être la même que celle obtenue en une seule passe de 10mm. La validation va être effectuée pour les deux fonctions coûts "énergie totale" du forgeage et "défaut de repli".

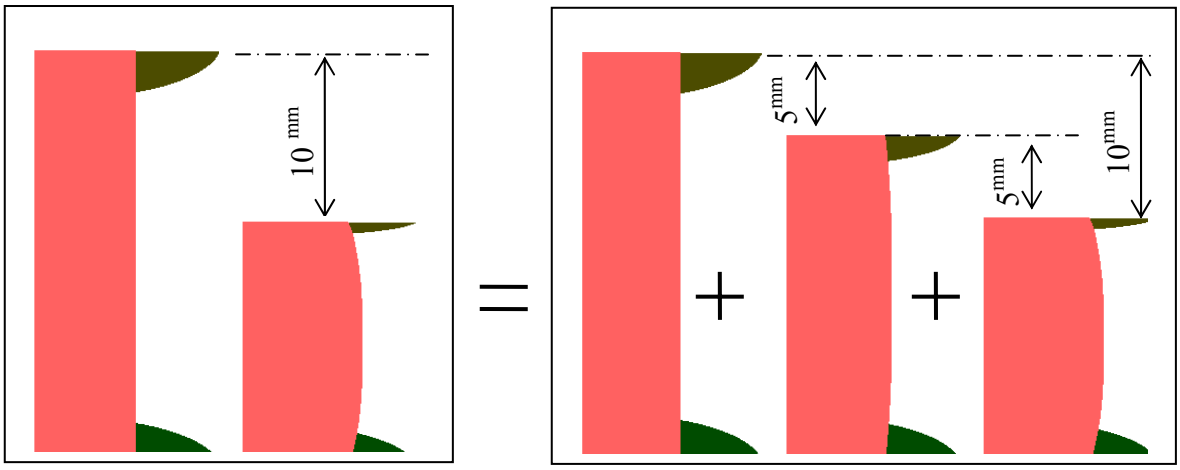



*Figure 3.18. Décomposition du test d'écrasement pour valider le calcul du gradient lors du forgeage multi-passes*

### III.3.5.1. Validation pour la fonction coût énergie totale

Les résultats du calcul des dérivées de la fonction coût énergie totale de mise en forme par rapport au paramètre "forme d'outils de préforme" sont présentés dans le *Tableau 3.3*. Ils sont obtenus en utilisant la méthode de l'Etat Adjoint ainsi que la méthode des différences finies. Pour la méthode des différences finies, la perturbation choisie est égale à $10^{-5}$ en valeur relative. Nous voyons que l'erreur relative entre les dérivées obtenues avec ces deux méthodes de calcul dans le forgeage à une passe est faible (de l'ordre de 0.2%) et de l'ordre de grandeur de l'erreur du calcul Elément Finis. Le gradient de la fonction coût énergie totale issu de la méthode de l'Etat Adjoint est donc précis.

| *Valeur de dérivée de $\Phi_{ener}$ donnée par méthode* | $\frac{d\Phi}{d\mu}$ | $\frac{d\Phi/d\mu - d\Phi/d\mu\big\|_{DF\,1\,passe}}{d\Phi/d\mu\big\|_{DF\,1\,passe}}$ | $\frac{d\Phi/d\mu - d\Phi/d\mu\big\|_{EA\,1\,passe}}{d\Phi/d\mu\big\|_{EA\,1\,passe}}$ |
|---|---|---|---|
| *Différences finies 1 passe* | $7.80x10^{12}$ | *0%* | *0.20%* |
| *Etat Adjoint 1 passe* | $7.78\,x10^{12}$ | *-0.20 %* | *0%* |
| *Etat Adjoint 2 passes* | $7.78\,x10^{12}$ | *-0.19%* | *0.012%* |

*Tableau 3.3 : Comparaison des dérivées de la fonction coût « énergie totale » obtenues par la méthode de l'Etat Adjoint et par la méthode des différences finies.*

En comparant les dérivées obtenues avec la méthode de l'Etat Adjoint dans les deux cas forgeage, en une passe et en deux passes, nous obtenons presque la même valeur (l'ordre de grandeur de l'erreur relative est de 0.012%). Si nous comparons la dérivée obtenue avec la méthode de l'Etat Adjoint en 2 passes avec celle obtenue par la méthode des différences finies, nous obtenons aussi une précision de 0.19% (l'ordre de la précision des calculs éléments finis).

### III.3.5.2. Validation pour la fonction coût repli

Pour la fonction coût « repli », les mêmes tests sont effectués. Les dérivées obtenues sont présentées dans le *Tableau 3.4*.

| *Valeur de dérivée de Φ donnée par méthode* | $\frac{d\Phi}{d\mu}$ | $\frac{d\Phi/d\mu - d\Phi/d\mu\big\vert_{DF\,1\,passe}}{d\Phi/d\mu\big\vert_{DF\,1\,passe}}$ | $\frac{d\Phi/d\mu - d\Phi/d\mu\big\vert_{EA\,1\,passe}}{d\Phi/d\mu\big\vert_{EA\,1\,passe}}$ |
|---|---|---|---|
| *Différences finies 1 passe* | $-1.024 \times 10^{-1}$ | *0%* | *-0.58%* |
| *Etat Adjoint 1 passe* | $-1.03 \times 10^{-1}$ | *0.58 %* | *0%* |
| *Etat Adjoint 2 passes* | $-1.028 \times 10^{-1}$ | *0.39%* | *-0.19%* |

*Tableau 3.4 : Comparaison des dérivées de la fonction coût « repli » obtenues par la méthode de l'Etat Adjoint avec celles obtenue par la méthode des différences finies.*

Le *Tableau 3.4* nous permet de valider le calcul du gradient de la fonction coût «repli». En une passe de forgeage, lors de la comparaison des valeurs du gradient obtenues par la méthode de l'Etat Adjoint et par celle des différences finies, nous voyons que la dérivée est précise (l'erreur relative inférieure à 1%). En comparant les dérivées calculées par la méthode de l'Etat Adjoint dans les deux cas de forgeage en une passe et en deux passes, nous obtenons une erreur relative de 0.19%. Cette erreur est bien satisfaisante pour notre calcul.

## III.4. BILAN ET CONCLUSIONS

En conclusion, dans le cadre de ce travail, nous voyons que :

- Une fonction coût adimensionalisée qui permet de mieux détecter le défaut de repli, est proposée.
- Un autre type de paramètres de forme moins spécifique, et qui permet d'aborder une gamme plus vaste de problème d'optimisation de forme en forgeage, est introduit. C'est celui de la "forme de l'outil de préforme".
- Le calcul du gradient des deux fonctions coûts "énergie totale" et "défaut de repli" est donc étendu pour ce type de paramètre. Le gradient obtenu est précis, pour le forgeage en une et plusieurs passes.

On pourra donc utiliser ces résultats dans la résolution des problèmes d'optimisation de forme.

# Chapitre IV

# BENCHMARK : OPTIMISATION DE LA PREFORME POUR LE FORGEAGE D'UNE ROUE DENTEE

## IV.1. Introduction

Un exemple typique de forgeage est de fabriquer des engrenages (*Figure 4.1*) qui sont souvent utilisés dans les moteurs, les voitures, les avions, etc. Ils permettent le transfert de mouvements et de forces importantes d'une partie à l'autre. Ce sont donc des pièces à très haute résistance mécanique.

*Figure 4.1. Exemple d'engrenages*

Le forgeage des engrenages est un domaine très compétitif car ce sont souvent des pièces de grande production forgées à froid ou à mi-chaud le plus souvent et donc :

- la durée de vie des outils de forgeage est souvent courte car ils sont intensivement sollicités, alors que leur fabrication est très coûteuse.
- l'énergie nécessaire pour le forgeage d'une telle pièce est importante.

Par conséquence, prolonger la durée de vie des outils et minimiser l'énergie de mise en forme reste un enjeu important, qui peut être formulé en problème d'optimisation et en utilisant des logiciels de simulation numérique.

Dans ce chapitre, nous considérons le cas de forgeage d'une roue dentée droite proposé par la société ASCOFORGE et qui est retenu comme problème caractéristique d'optimisation de forme en forgeage, c'est-à-dire comme benchmark d'optimisation.

# IV.2. Description du cas d'optimisation

## IV.2.1. Présentation du cas de forgeage d'un triaxe

Ce cas consiste à forger une roue dentée de dix dents à mi-chaud à partir d'un lopin initial de forme axisymétrique simple. La *Figure 4.2* présente la forme de la pièce initiale (*Figure 4.2a*) et d'une pièce forgée finale (*Figure 4.2b*).

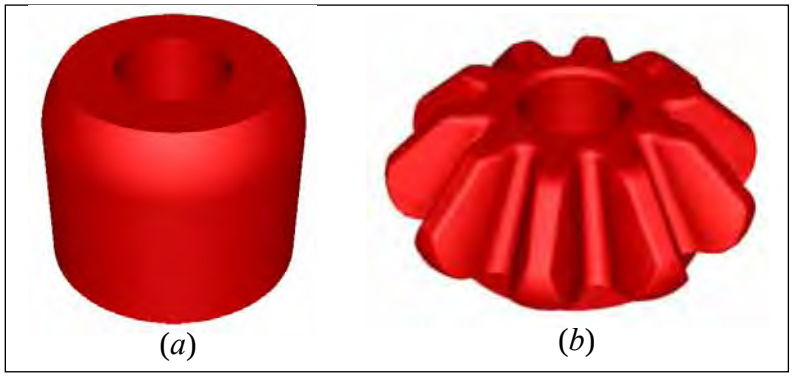



*Figure 4.2. Lopin initial (a) et roue dentée (b)*

L'outillage de forgeage est présenté sur la *Figure 4.3a* et la *Figure 4.3b*. Le lopin est écrasé par la matrice et l'éjecteur supérieurs. Dans son mouvement de translation verticale, la matrice supérieure entraîne la matrice inférieure. De même, l'éjecteur supérieur entraîne avec lui la tige centrale. Le dernier outil, la douille éjectrice, est fixe et ne sert qu'à caler le lopin.

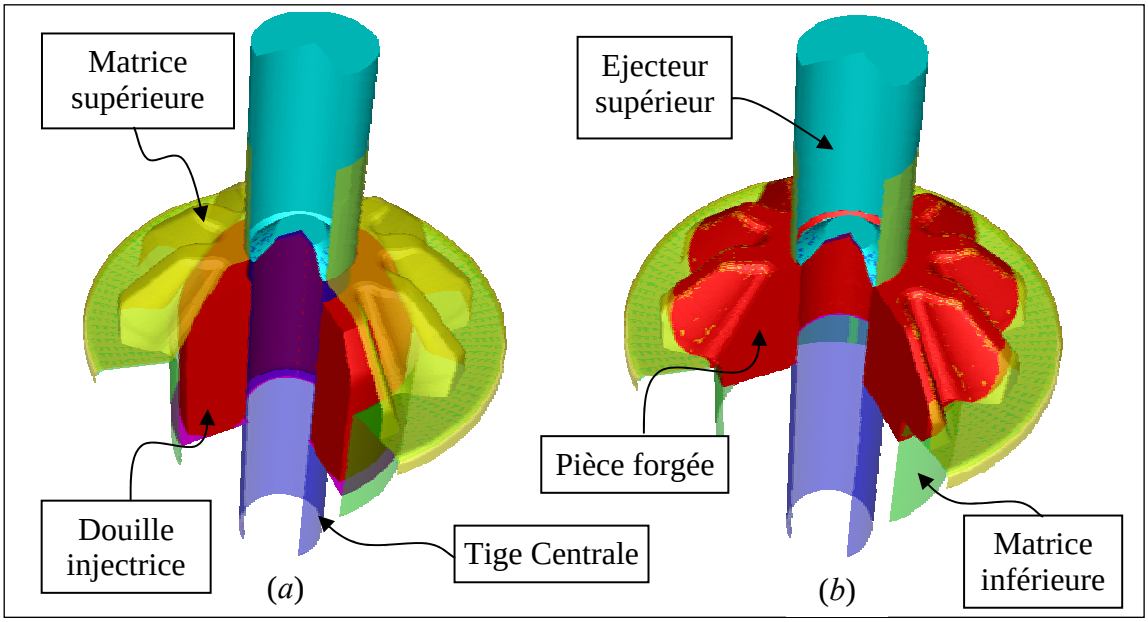



*Figure 4.3. Pièces initiale (a) et finale (b) avec les outils de forgeage*

La hauteur du lopin initial est de 45 *mm* et après écrasement elle est de 21 *mm*, ce qui représente une déformation de quasiment 50%.

La symétrie de la roue dentée permet de modéliser seulement un vingtième de la pièce (comme présenté sur la *Figure 4.4*) correspondant à une demi dent.

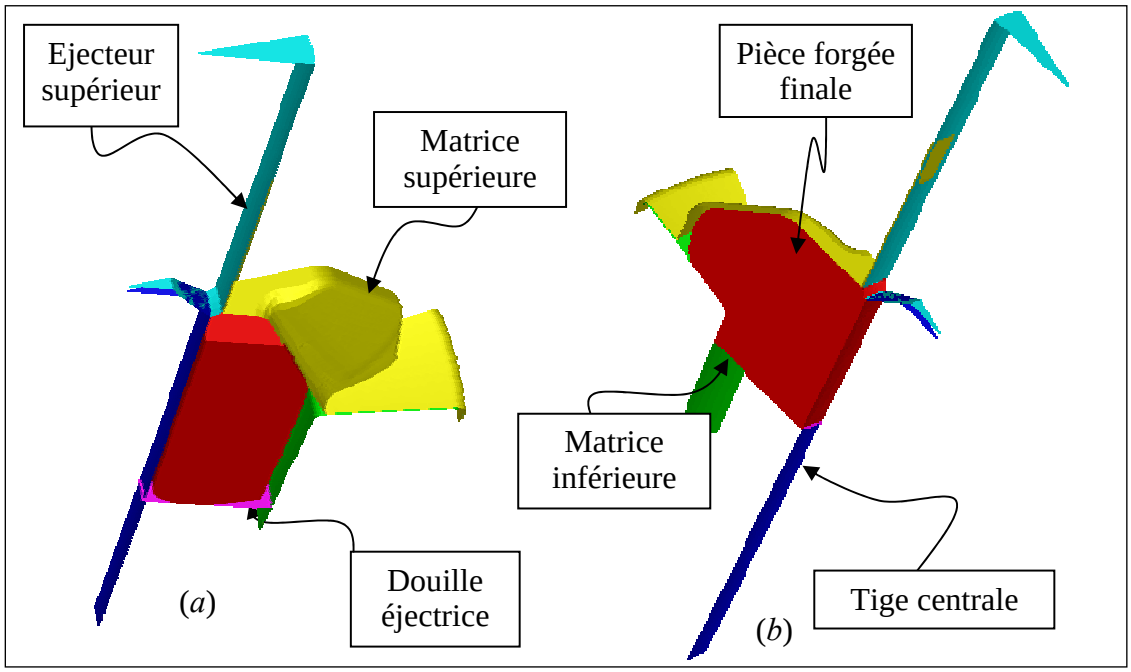



*Figure 4.4.Un vingtième de la préforme à forger (a) et une demi dent (b) de la roue dentée forgée avec les outils de forgeage.*

## IV.2.2. Paramétrisation de la préforme

Lors de la paramétrisation de la préforme, une difficulté réside dans la prise en compte de la contrainte du volume constant, quelque soit le jeu de paramètres.

M. Laroussi [*Laroussi 2003*] a donc proposé d'utiliser une paramétrisation plus simple, de type polynomial. Le contour du plan radial est un polygone fermé (*Figure 4.5a*) composé de :

- sept portions (parties polynomiales) distinctes
  - quatre portions I, II, IV et VI sont des segments droits.
  - trois portions III, V et VII sont des polynômes de degré 2.

  Ces portions doivent satisfaire les conditions de raccordement (le contour doit être lisse) [*Laroussi 2003*].

- sept points de contrôle de coordonnées $(r_i, z_i)_{i=1,\ldots,7}$ dans le repère radial ($r$, $z$). Les mouvements de ces points sont les paramètres à optimiser. Les points retenus pour l'optimisation sont les points 2, 3 et 4. Leurs déplacements en constituent les paramètres (*Figure 4.5b*), de la manière suivante:
  - le paramètre $\mu_1$ est le déplacement vertical du point 2 (dans la direction z).

- le paramètre $\mu_2$ est le déplacement horizontal du point 3 (dans la direction r). Notons que dans la direction z, le déplacement du point 3 est lié à celui du point 2 de telle sorte que la portion II reste un segment parfaitement horizontal.
- le paramètre $\mu_3$ est le déplacement vertical du point 4 (dans la direction z).
- le paramètre $\mu_4$ est le déplacement horizontal des points 4 et 5 (identiques) dans la direction r (la portion IV doit être toujours un segment vertical).

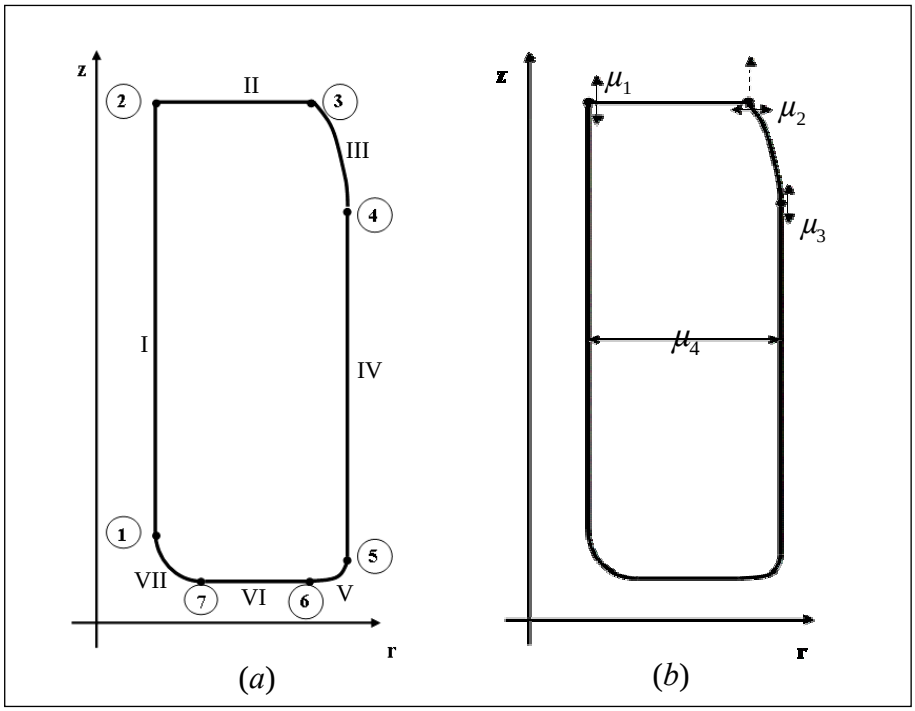



*Figure 4.5. Paramétrisation 2D de la préforme axisymétrique de la roue dentée*

Avec cette paramétrisation, 4 paramètres caractérisent la préforme. Le problème d'optimisation est celui de minimisation d'une fonction coût Φ (effort de forgeage, énergie totale de mise en forme, défaut de repli…) sous la contrainte de volume constant :

$$\begin{cases} Min \ \ \Phi(\mu_1,\mu_2,\mu_3,\mu_4) \\ \vartheta(\mu_1,\mu_2,\mu_3,\mu_4) = \vartheta_0 = \text{constant} \end{cases} \qquad (1)$$

où $\vartheta_0$ est le volume initial.

La modification des paramètres de forme $(\mu_1,\mu_2,\mu_3,\mu_4)$ entraîne inévitablement un changement du volume de la préforme qui doit pourtant rester constant.

Il existe plusieurs méthodes pour traiter cette contrainte mais la méthode la plus simple est de l'éliminer en écrivant une condition de liaison sur un des paramètres. Dans notre cas, le paramètre qui permet le plus naturellement d'éliminer cette contrainte est $\mu_4$. Le problème (1) peut donc être reformulé de la manière suivante :

$$\begin{cases} Min \ \ \Phi(\mu_1,\mu_2,\mu_3) \\ \mu_4 = f(\mu_1,\mu_2,\mu_3) \end{cases} \tag{2}$$

où l'expression de *f* est déterminée par la relation équivalente $\vartheta(\mu_1,\mu_2,\mu_3,\mu_4)=\vartheta_0$. Les détails de cette expression sont présentés dans [*Laroussi 2003*]. Le problème d'optimisation est ainsi un problème à trois paramètres $(\mu_1,\mu_2,\mu_3)$. On peut aussi lui ajouter les conditions de bords appliquées aux paramètres de conception.

$$Min \quad \Phi(\mu_1,\mu_2,\mu_3) \tag{3}$$

$$\text{avec} \quad \begin{cases} 40 \le \mu_1 \le 46 \\ 18 \le \mu_2 \le 22 \\ 28 \le \mu_3 \le 34 \end{cases}$$

## IV.2.3. Génération automatique du maillage de la préforme

Le processus de génération des maillages de préforme 3D à partir du contour 2D est présenté dans la *Figure 4.6*.

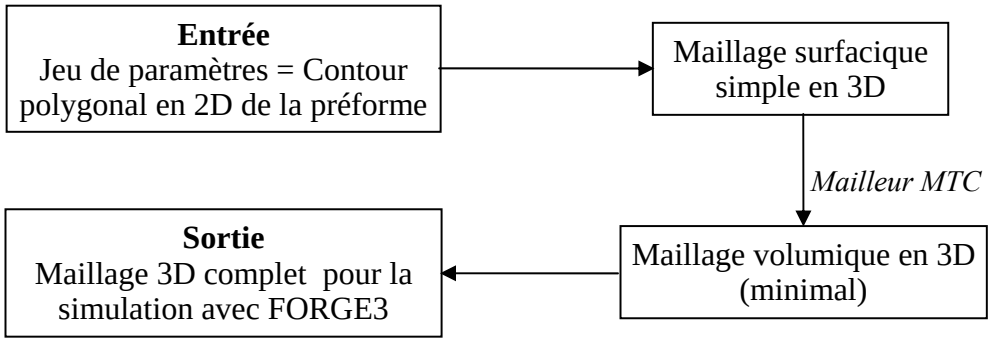



*Figure 4.6. Module de génération automatique du maillage de la préforme*

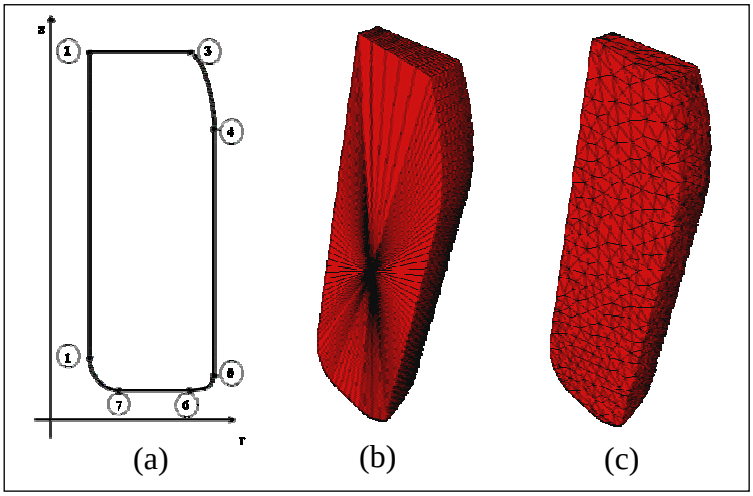



*Figure 4.7. Contour polynomial 2D (a) – Maillage surfacique simple 3D (b) – Maillage surfacique et volumique (c)*

A partir du jeu de paramètres, un contour polygonal 2D fermé (*Figure 4.7a*) est obtenu. Ensuite, on génère un maillage surfacique simple en 3D en extrudant ce contour dans la direction axisymétrique suivant un angle $\theta = 18°$ (correspondant à un vingtième du lopin) et en créant la connectivité entre les nœuds (*Figure 4.7b*). Le mailleur MTC permet alors d'obtenir le maillage volumique de la préforme 3D. Il manque encore des informations (comme la topologie de la frontière, les plans de symétrie, etc.) qui font l'objet de la dernière étape (*Figure 4.7c*).

## IV.2.4. Fonctions coûts et benchmark d'optimisation

Du point de vue industriel, on cherche à optimiser la géométrie de la préforme pour augmenter la durée de vie des outils de forgeage. Une approche de ce problème consiste à minimiser les efforts subis par les outils durant tout le forgeage, tout en évitant la formation de défauts majeurs de type repli de matière. Cela nous conduit à formuler le benchmark suivant : minimiser la fonction coût "énergie totale" de forgeage (équation 3.31) ou/et la fonction coût " repli" (équation 3.34) vis-à-vis des trois paramètres de forme.

Pour réduire le temps de calcul et faciliter la mise en place des méthodes d'optimisation, nous décidons de simplifier le comportement du matériau et de le supposer linéaire. Ainsi, sur un PC Pentium IV (2.4Ghz – 512Mo RAM) la durée d'une simulation complète sans calcul du gradient est de 45 minutes, et est de 60 minutes avec calcul du gradient.

Enfin, nous limitons le nombre d'évaluations de la fonction coût à 50.

Nous obtenons ainsi un problème caractéristique du forgeage 3D que nous utilisons pour comparer les différents algorithmes d'optimisation entre eux.

## IV.2.5. Processus d'optimisation automatique

L'algorithme d'optimisation est présenté sur la *Figure 4.8*.

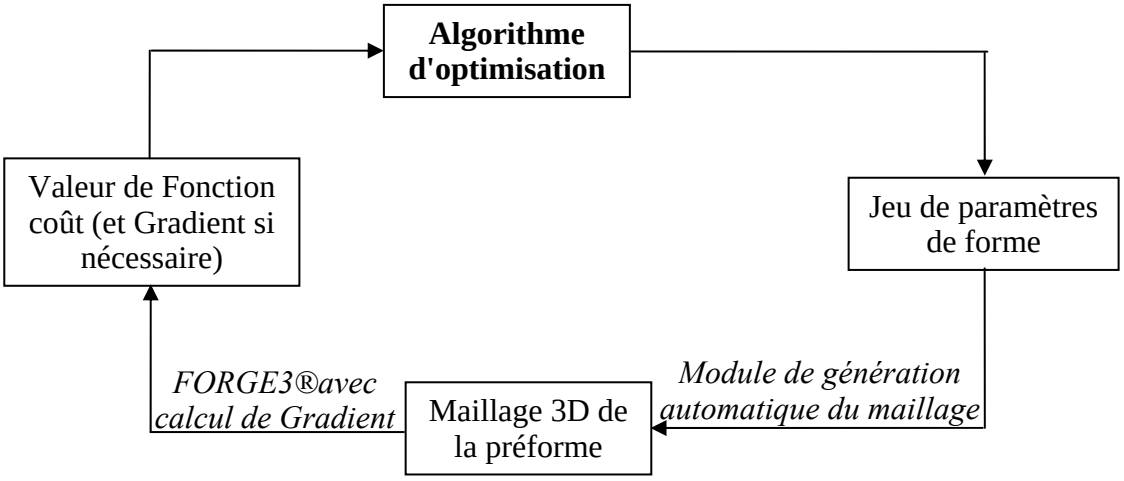



*Figure 4.8. Processus d'optimisation automatique pour résoudre le problème de benchmark*

# IV.3. Résultats d'optimisations

Cette section présente les résultats d'optimisation obtenus. La préforme utilisée par ASCOFORGE correspond au jeu de paramètres suivant ($\mu_1 = 44,60mm$ ; $\mu_2 = 21,65mm$ ; $\mu_3 = 32,33mm$). Elle est prise comme référence. Notons que cette préforme est le fruit de l'expérience et non d'une optimisation particulière. Par ailleurs, nous avons pu constater que les géométries optimisées avec un comportement linéaire ou viscoplastique sont assez différentes. Cette préforme n'a donc pas de qualités particulières, mais elle n'est pas non plus arbitraire. C'est donc une simple référence.

Les préformes obtenues par différents algorithmes d'optimisation sont comparées à celle de référence et entre elles, en termes d'amélioration de la valeur de fonction coût, des géométries de la préforme et du coût de calcul.

## IV.3.1. Optimisation par rapport à un seul critère

### IV.3.1.1. Objectif 1 : Energie totale du forgeage

La fonction coût est l'énergie totale de forgeage. Elle est exprimée en valeur relative. Elle est adimensionnalisée par la valeur obtenue avec la préforme de référence.

#### *IV.3.1.1.1. Résultats d'optimisation avec BFGS*

Ce benchmark est d'abord étudié avec l'algorithme de gradient BFGS. Avec un pas de descente de *1,2mm* et un rayon de recherche linéaire de *0,6mm*, nous partons d'un point choisi aléatoirement dans l'espace correspondant au jeu de paramètres ($\mu_1$*=40,0 ;* $\mu_2$*=18,7 ;* $\mu_3$*=34,0*).

L'évolution de la fonction coût et celle des paramètres sont présentées sur la *Figure 4.9.* La préforme optimisée par BFGS et la préforme de référence sont présentées sur la *Figure 4.10*.

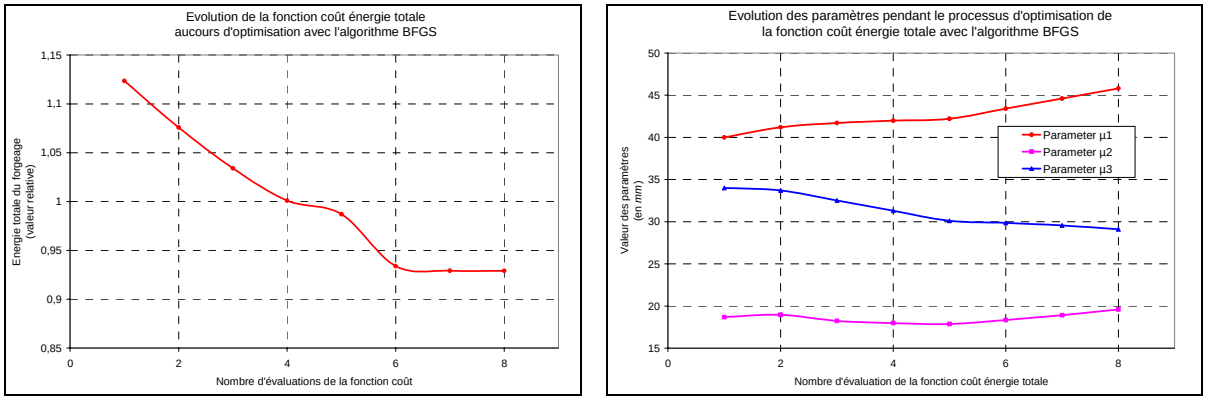



*Figure 4.9. Evolution des paramètres et de la fonction coût énergie totale au cours du processus d'optimisation avec l'algorithme de gradient BFGS.*

Avec cet algorithme, on obtient une amélioration de 7,1% de la valeur de la fonction coût. L'algorithme s'arrête après seulement 8 calculs de forgeage lorsque le critère de convergence sur l'amélioration relative de la fonction coût est obtenu.

Si on regarde l'évolution des paramètres, on observe qu'ils varient de manières très différentes. Le paramètre qui représente la hauteur du lopin, $\mu_1$ augmente sans cesse. Il balaie tout l'intervalle de recherche. Au contraire, le paramètre $\mu_3$ a tendance à décroître continûment tandis que le paramètre $\mu_2$ fluctue légèrement. Les deux paramètres $\mu_1$ et $\mu_3$ prennent des valeurs très différentes par rapport à celles du point de départ alors que le paramètre $\mu_2$ ne change pas beaucoup. Ainsi, la préforme optimisée est bien différente (plus haute et plus épaisse) de la préforme initiale, comme le montre la *Figure 4.10*.

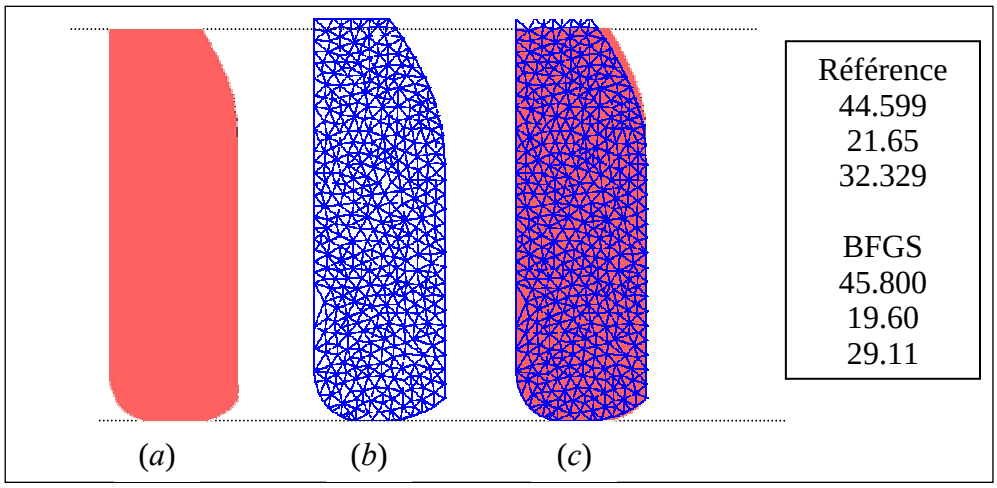



*Figure 4.10. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme de gradient BFGS (b) – Comparaison entre les deux préformes (c).*

#### *IV.3.1.1.2. Résultats d'optimisation avec SE-Meta*

La Stratégie d'Evolution avec Meta modèle (SE-Meta) présenté dans la section I.5.2 est maintenant appliquée à ce benchmark. La stratégie (2+10) (population de 10 individus dont 2 parents) est retenue avec un taux de rejet de 80% de sorte que l'on évalue exactement que deux individus à chaque génération.

Après une initialisation aléatoire pour créer la base de données initiale (avec 2**nbparam* = 6 points), la SE-Meta commence la recherche de la solution optimale. Les résultats d'optimisation sont présentés sur la *Figure 4.11*.

La SE-Meta démontre son caractère « stochastique » dans la recherche de la solution optimale. La fonction coût fluctue fortement (*Figure 4.11*). Au cours du processus d'optimisation, la SE-Meta balaie tout l'espace de recherche. Grâce au métamodèle, elle choisit uniquement les solutions ayant le meilleur potentiel pour le calcul exact. Si on observe l'évolution des paramètres (*Figure 4.11*), on constate que tous balaient leur intervalle de recherche. A la fin du processus d'optimisation, la fonction coût et les paramètres ne sont pas stabilisés : la convergence de l'AE n'est pas atteinte au bout de 50 calculs, ce qui est une valeur faible pour atteindre la convergence d'une telle stratégie d'évolution.

La meilleure solution proposée par la SE-Meta est trouvée à la 27$^{ième}$ évaluation et correspond au jeu de paramètres ($\mu_1$*=46,0 ;* $\mu_2$*=18,0 ;* $\mu_3$*=31,72*). Elle apporte une amélioration de 10,4%. La préforme correspondante est assez différente de celle de référence (plus haute et plus épaisse) et ressemble bien celle trouvée par BFGS (voir *Figure 4.10* et *Figure 4.18*).

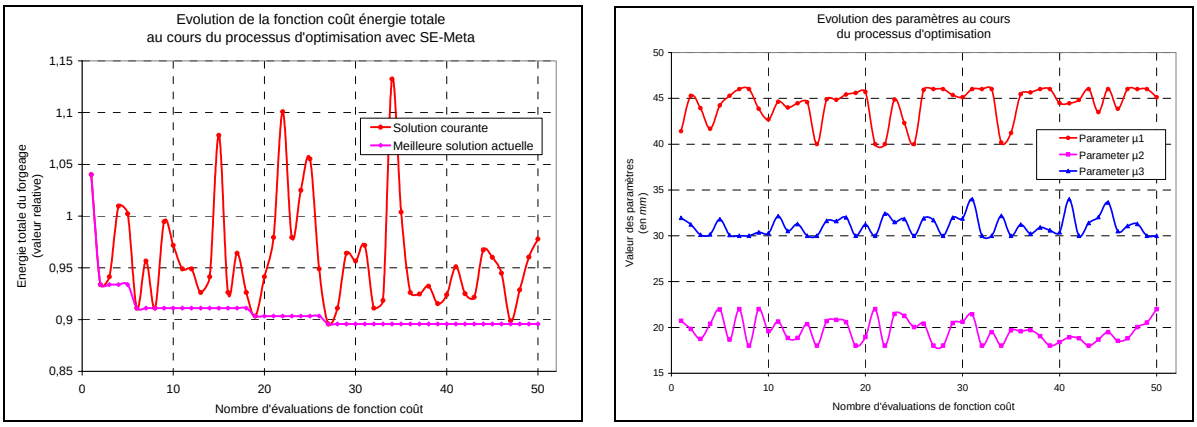



*Figure 4.11. Evolution de la fonction coût énergie totale et celle des paramètres au cours du processus d'optimisation avec la Stratégie d'Evolution avec Métamodèle, SE-Meta.*

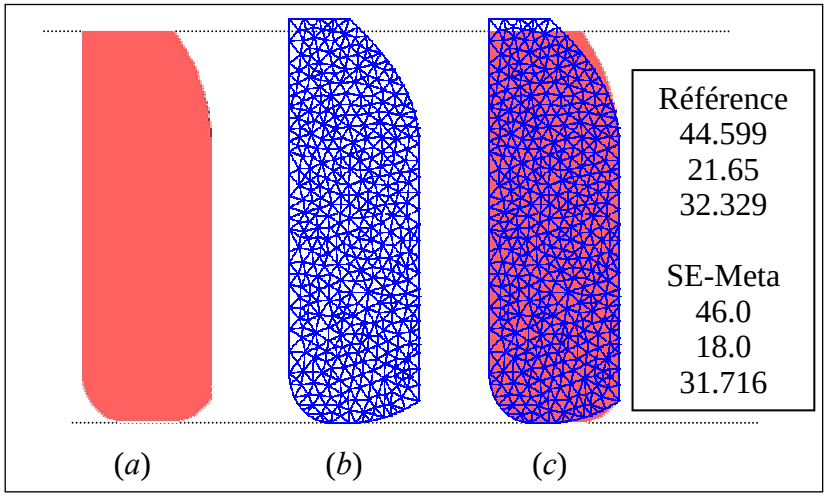



*Figure 4.12. Préforme de référence (a) – Préforme optimisée obtenue avec la SE-Meta (b) – Comparaison entre les deux préformes (c).*

#### *IV.3.1.1.3. Résultats d'optimisation avec AGMGA*

La première variante des deux algorithmes hybrides que nous avons développés, l'AGMGA présenté dans la section II.2, est maintenant employée.

Nous utilisons une population de 120 individus avec 6 évaluations exactes (et donc 6 agglomérats) par génération et un nombre maximal de 8 générations. Les résultats d'optimisation avec l'AGMGA sont présentés sur la *Figure 4.13*.

Cet algorithme donne des résultats très encourageants. En regardant l'évolution de la fonction coût (*Figure 4.13*), on observe qu'elle fluctue plus légèrement qu'avec la SE-Meta. Cette fluctuation diminue à la fin du processus. Tous les points maîtres évalués sont meilleurs (valeur relative <1) que la solution de référence (ce qui est un effet heureux du processus de génération aléatoire de la population initiale). Si on observe l'évolution des paramètres (*Figure 4.13*), on voit qu'ils varient fortement au début du processus mais qu'après cinq générations (30 calculs de fonction coût), leurs valeurs sont relativement stabilisées, et notamment le paramètre $\mu_3$.

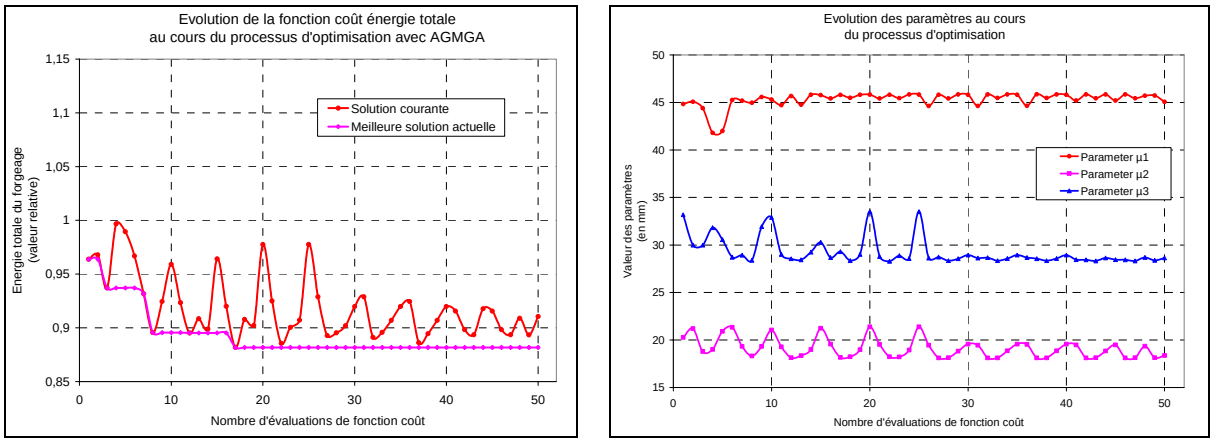



*Figure 4.13. Evolution de la fonction coût énergie totale et celle des paramètres au cours du processus d'optimisation avec l'algorithme hybride AGMGA.*

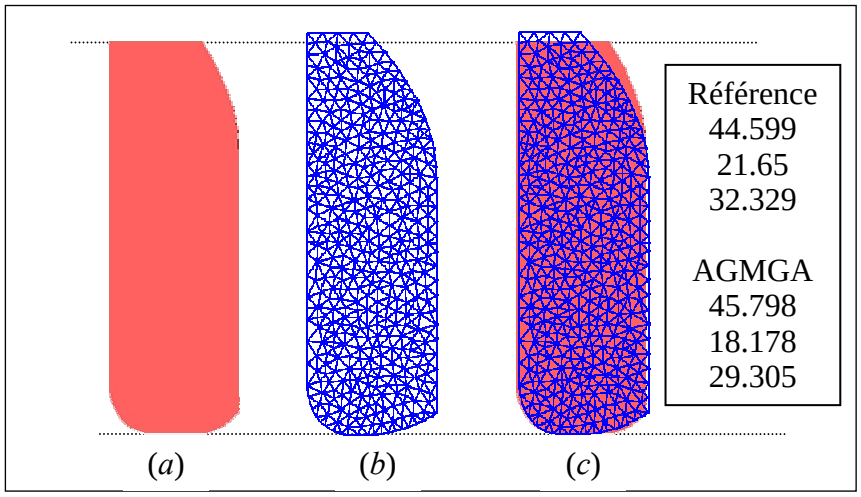



*Figure 4.14. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme hybride AGMGA (b) – Comparaison entre les deux préformes (c).*

L'AGMGA a trouvé une solution qui apporte une amélioration de 11,8%. Elle est meilleure que la solution de la SE-Meta, et elle est obtenue après seulement 17 évaluations avec le calcul du gradient (ce qui correspond à un coût en temps de calcul équivalent à celui de 23 simulations sans calcul du gradient en considérant que le calcul du gradient représente généralement 30% du coût du calcul de la simulation).

La forme obtenue, correspond au jeu de paramètres ($\mu_1$=45,80 ; $\mu_2$=18,18 ; $\mu_3$=29,31), et est présentée sur la *Figure 4.14*. Elle ressemble à la préforme proposée par la SE-Meta.

#### *IV.3.1.1.4. Résultats d'optimisation avec AGMGO*

La deuxième variante des algorithmes hybrides, l'AGMGO présentée dans la section II.3, est abordée avec également six évaluations exactes par génération et un nombre maximal de 8 générations. Les résultats obtenus sont présentés sur la *Figure 4.15*. La meilleure solution obtenue apporte une amélioration de 11,4% de la fonction coût. Cette solution (très proche de celle de l'AGMGA), correspond au jeu de paramètres ($\mu_1$=45,81 ; $\mu_2$=18,25 ; $\mu_3$=28,29), est

trouvée dès la douzième évaluation (ce qui correspond au coût de 16 simulation dans gradient en général).

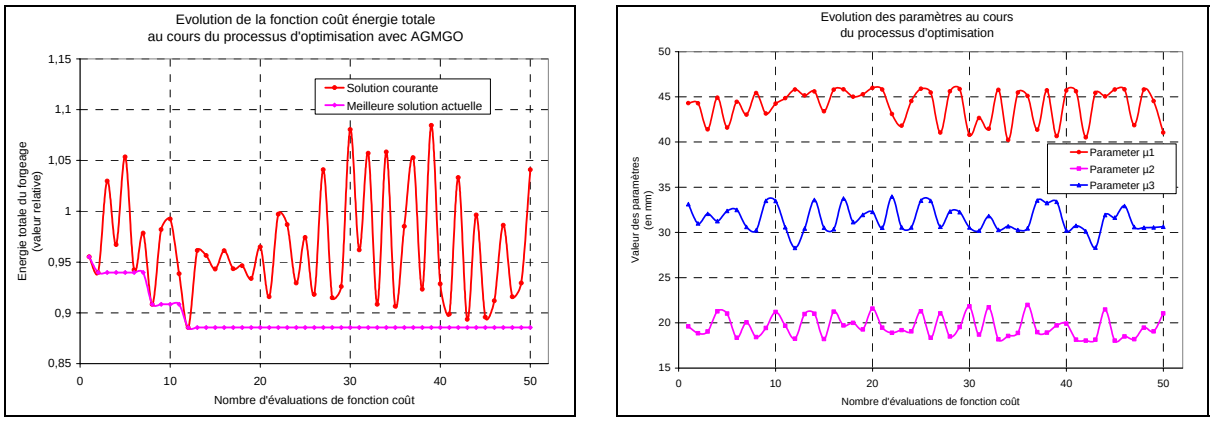



*Figure 4.15. Evolution de la fonction coût énergie totale et celle des paramètres au cours du processus d'optimisation avec l'algorithme hybride AGMGO.*

En revanche, nous observons une fluctuation très forte des valeurs de la fonction coût et des paramètres d'optimisation au cours de l'optimisation, du début jusqu'à la fin (*Figure 4.15*).

En fait, l'AGMGO essaie d'utiliser les informations disponibles (les points déjà évalués) pour avoir la meilleure interpolation de la fonction coût sur l'ensemble de l'espace paramétrique. A chaque génération, il introduit des nouveaux points maîtres dans les sections inexplorées de l'espace de recherche et pas nécessairement aux endroits les plus proches de l'optimum. Cela entraîne les fluctuations mentionnées précédemment.

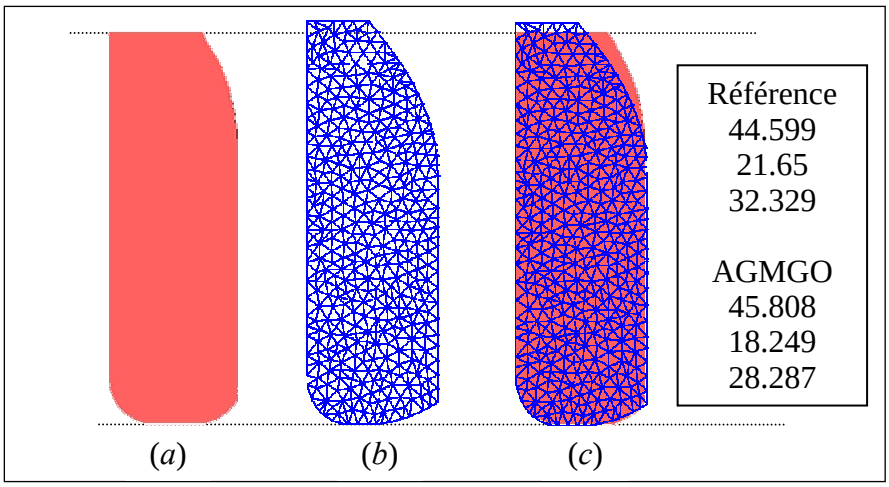



*Figure 4.16. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme hybride AGMGO (b) – Comparaison entre les deux préformes (c).*

Si on compare la préforme (*Figure 4.16*b) fournie par l'AGMGO avec celle de référence, on voit qu'elles sont très différentes et que cette préforme ressemble à la préforme proposée par l'AGMGA et la SE-Meta.

#### *IV.3.1.1.5. Synthèse des résultats d'optimisation*

Le *Tableau 4.1* présente une synthèse des résultats de ce benchmark avec la fonction coût "énergie totale". Pour tous les algorithmes d'optimisation, il indique le numéro de la simulation $N_{opt}$ où on a obtenu la meilleure solution en référence au nombre total de simulations $N_{tot}$ effectuées. Il présente la valeur des paramètres ($\mu_1$, $\mu_2$, $\mu_3$) optimaux, la valeur de la fonction coût, et l'amélioration apportée. La *Figure 4.17* synthétise les résultats de convergence des algorithmes d'optimisation employés. Les préformes optimisées et sont regroupées et comparées sur la *Figure 4.18*.

| | $N_{opt}/N_{tot}$ | $\mu_1$(mm) | $\mu_2$(mm) | $\mu_3$(mm) | $\Phi_{opt}$ | Amélioration |
|---|---|---|---|---|---|---|
| *Référence* | - | 44,60 | 21,65 | 32,33 | 1 | - |
| *BFGS* | 7/8 | 45,80 | 19,60 | 29,11 | 0,929 | -7,1% |
| *SE-Meta* | 27/50 | 46,00 | 18,00 | 31,72 | 0,896 | -10,4% |
| *AGMGA* | 17/50 | 45,80 | 18,18 | 29,31 | 0,882 | -11,8% |
| *AGMGO* | 12/50 | 45,81 | 18,25 | 28,29 | 0,886 | -11,4% |

*Tableau 4.1 : Synthèse des résultats d'optimisation avec les différents algorithmes pour le benchmark avec la fonction coût énergie totale*

On constate que tous les algorithmes arrivent à trouver une meilleure préforme que celle proposée en référence, apportant des améliorations significatives à la fonction coût (de 7% à 11 %). Les deux algorithmes hybrides montrent leurs intérêts avec plus de 11% d'amélioration.

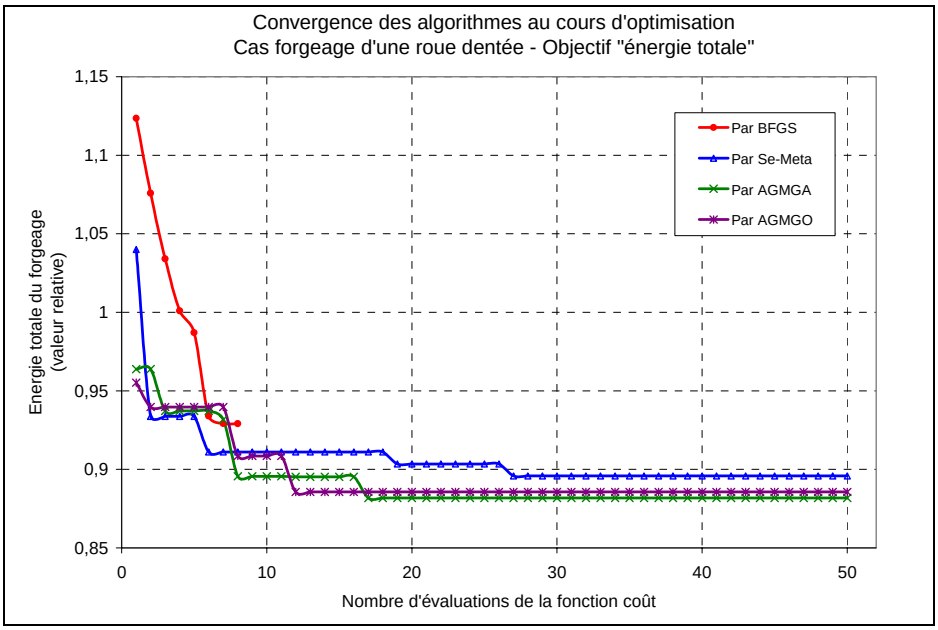



*Figure 4.17. Convergence de différents algorithmes au cours du processus d'optimisation de la fonction coût énergie totale.*

La *Figure 4.17* montre l'efficacité de l'algorithme BFGS qui fait diminuer rapidement la fonction coût mais qui stagne au bout de quelques itérations. Les algorithmes évolutionnaires montrent leur point fort en trouvant de meilleures solutions, mais ils nécessitent davantage d'évaluations de la fonction coût.

Les solutions obtenues avec les algorithmes évolutionnaires sont fort semblables. Nous pouvons juste comparer leur rapidité à trouver la solution la plus satisfaisante, et constater ainsi l'intérêt de l'hybridation.

Les différentes préformes optimisées présentées sur la *Figure 4.18* sont très proches de l'une de l'autre et bien différentes de la préforme de référence ce qui semble indiquer que ce problème admet un extremum unique. Ces préformes sont en général plus hautes et plus épaisses ce qui permet de diminuer la quantité d'énergie nécessaire au forgeage.

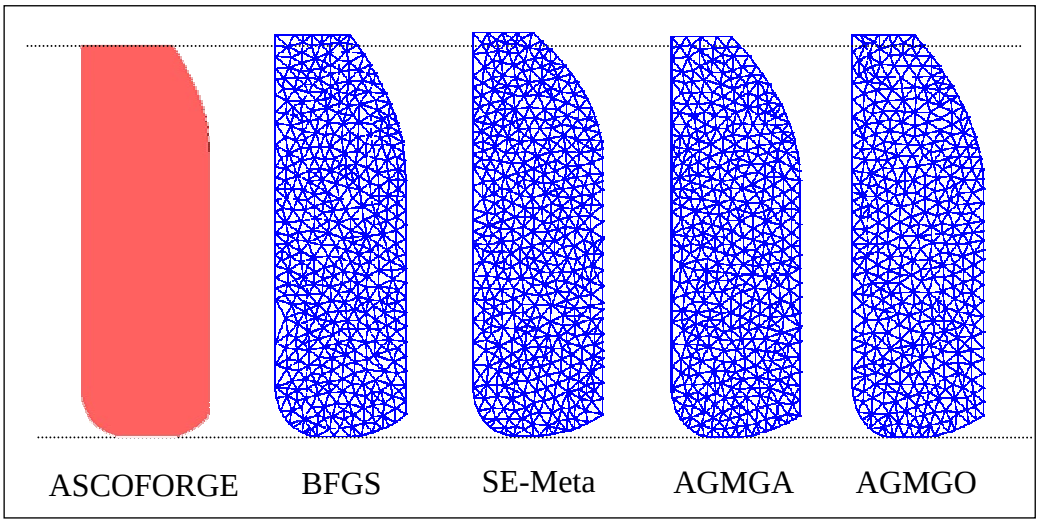



*Figure 4.18. Comparaison des préformes obtenues par les différents algorithmes d'optimisation*

En observant les valeurs des paramètres optimaux, on constate qu'elles ont tendance à aller vers le coin ($\mu_1$=46,0, $\mu_2$=18,0, $\mu_3$=28,0) de l'espace de recherche, ce qui indique que la solution à ce problème est probablement assez simple, unimodale et que la solution optimale se trouve dans le coin de l'espace.

Afin d'étudier un problème plus complexe non convexe et également caractéristique de cas rencontrés en forgeage, nous nous orientons vers une fonction coût qui combine deux objectifs antagonistes, l'énergie totale et la mesure de la non qualité de la surface (ou de défaut de repli). Avant d'aborder cette fonction coût, nous faisons quelques tests avec la fonction "défaut repli" pour étudier le comportement du problème.

#### IV.3.1.2. Objectif 2 : Mesure de la non qualité de la surface ou du défaut de repli

Dans cette section, on étudie seulement les deux algorithmes BFGS et SE-Meta. On utilise toujours la préforme proposée par l'industriel ASCOFORGE comme référence et pour laquelle la valeur de la fonction coût repli est de *2,16*.

#### *IV.3.1.2.1. Résultats d'optimisation avec BFGS*

L'évolution de la fonction coût et celle des paramètres d'optimisation avec l'algorithme BFGS sont présentées sur la *Figure 4.19*. L'algorithme BFGS part d'un point choisi aléatoirement dans l'espace de recherche et stagne rapidement. Il propose une solution ayant une valeur de la fonction de *2,03*, soit une amélioration de 6% par rapport à la préforme de référence.

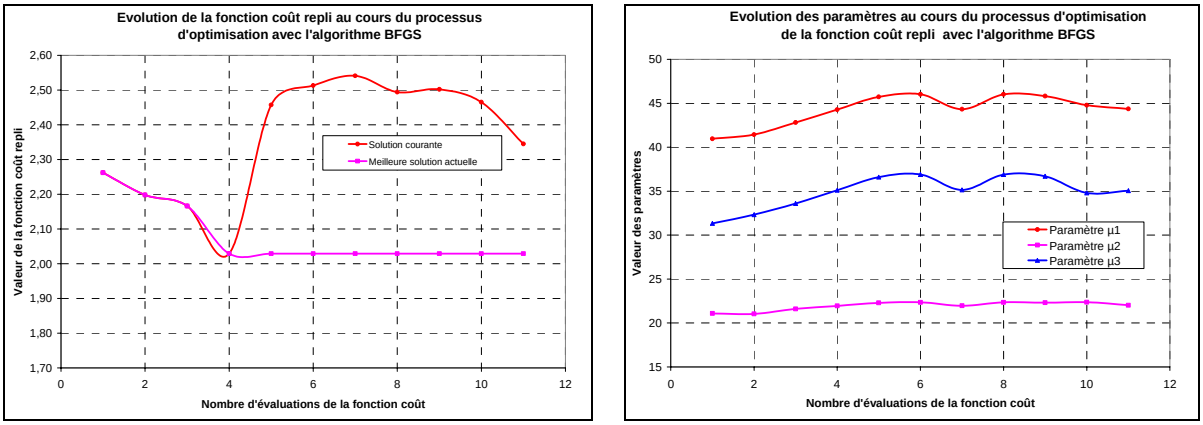



*Figure 4.19. Evolution de la fonction coût "repli" et celle des paramètres au cours du processus d'optimisation avec l'algorithme de gradient BFGS.*

En observant l'évolution des paramètres, on constate qu'ils varient fortement à l'exception du paramètre $\mu_2$. Après trois itérations, les variations semblent portées dans les mauvaises directions, elles ne peuvent plus améliorer la solution. L'algorithme semble piégé dans un minimum local.

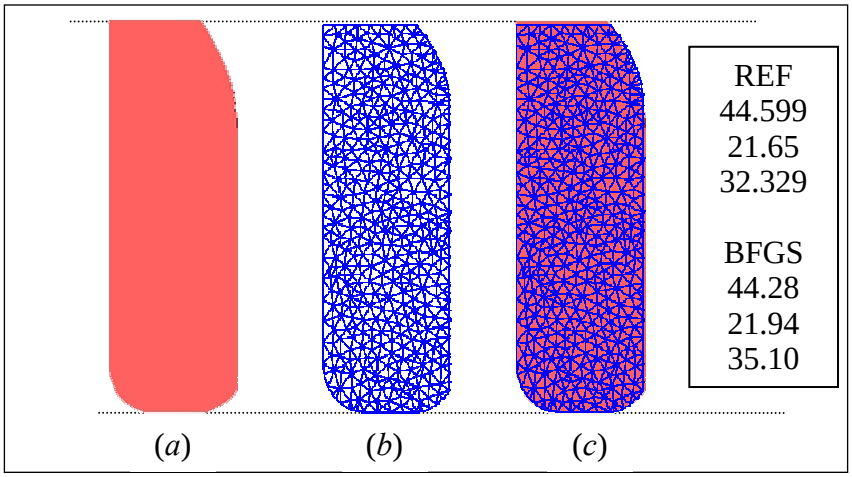



*Figure 4.20. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme de gradient BFGS (b) – Comparaison entre les deux préformes (c).*

La *Figure 4.20* présente la préforme optimale obtenue qui correspond au jeu de paramètres ($\mu_1 = 44,28mm$ ; $\mu_2 = 21,94mm$ ; $\mu_3 = 35,10mm$). On voit bien que cette préforme ressemble assez à la préforme de référence.

#### *IV.3.1.2.2. Résultats d'optimisation avec SE-Meta*

Avec la SE-Meta, les évolutions de la fonction coût et des paramètres sont présentées sur la *Figure 4.21*.

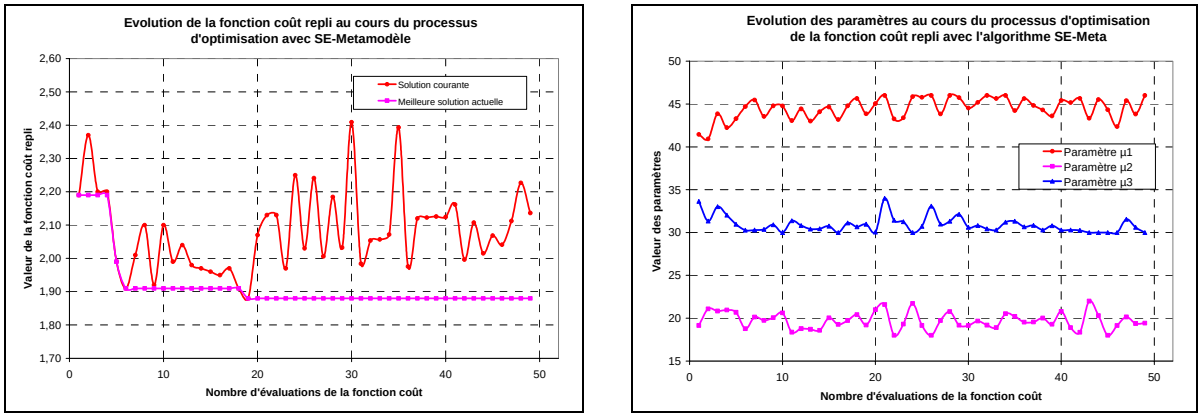



*Figure 4.21. Evolution de la fonction coût "repli" et celle des paramètres au cours du processus d'optimisation avec la Stratégie d'Evolution avec Metamodele (SE-Meta).*

La SE-Meta montre son efficacité en trouvant une préforme qui améliore la fonction coût d'une valeur relative de 13% ($\Phi_{repli\ opt}$ = 1,88), assez rapidement, à la 19$^{ième}$ évaluation.

La préforme "optimale" (*Figure 4.22*), qui correspond au jeu de paramètres ($\mu_1$=*43,87mm* ; $\mu_2$=*21,94mm* ; $\mu_3$=*35,10mm*), est bien différente de la préforme proposée par BFGS et de celle de référence. Elle est plus épaisse et moins haute. On peut aussi remarquer que cette préforme est bien différente de celle qui minimise l'énergie totale de forgeage (voir *Figure 4.18*). La combinaison des deux fonctions coûts conduira à la recherche d'un compromis entre deux objectifs antagonistes.

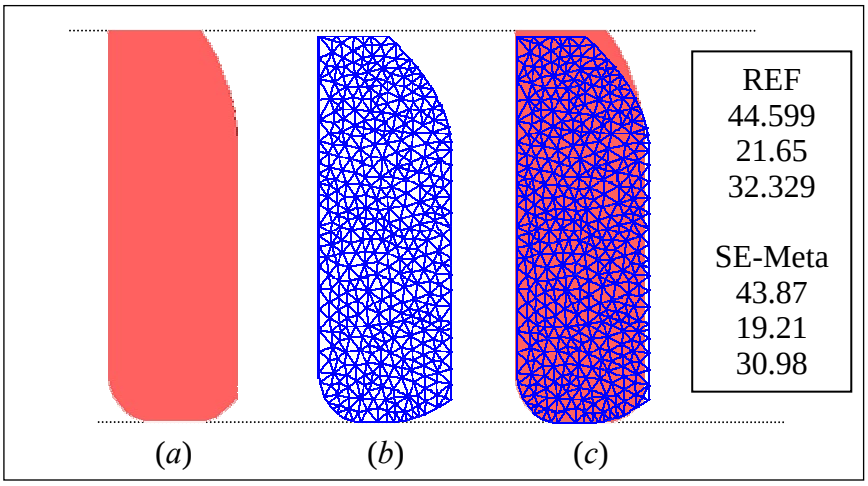



*Figure 4.22. Préforme de référence (a) – Préforme optimisée obtenue avec la SE-Meta (b) – Comparaison entre les deux préformes (c).*

## IV.3.2. Optimisation multi-critères

Afin d'obtenir un second benchmark plus caractéristique des problèmes non convexes avec plusieurs extrema, nous considérons la fonction coût qui est la combinaison des deux fonctions précédentes. Elle s'écrit :

$$\Phi_{tot} = 0.5\frac{\Phi_1}{\Phi_{1ref}} + 0.5\frac{\Phi_2}{\Phi_{2ref}}$$

avec $\Phi_1$ et $\Phi_{1ref}$ : fonction coût "énergie totale" et sa valeur de référence

$\Phi_2$ et $\Phi_{2ref}$ : fonction coût "repli" et sa valeur de référence

La préforme proposée par l'industriel ASCOFORGE est toujours utilisée comme référence. En conséquence, la valeur de sa fonction coût est 1.

### IV.3.2.1. Résultats d'optimisation avec BFGS

Les résultats d'optimisation obtenus avec l'algorithme BFGS sont présentés sur la *Figure 4.23* montrant sa faiblesse sur les problèmes non convexes avec une stagnation après seulement 3 itérations. Malgré un redémarrage de l'algorithme la descente dans la direction du gradient ne permet pas d'améliorer la fonction coût.

La meilleure solution correspond au jeu de paramètres ($\mu_1$=*40,50mm* ; $\mu_2$=*17,95mm* ; $\mu_3$=*30,05mm*), et est obtenue en trois itérations. Elle apporte une amélioration de seulement 2,7%. Si on regarde l'évolution des paramètres, on voit qu'ils varient très légèrement autour de leurs valeurs initiales. L'algorithme BFGS est très probablement coincé dans un optimum local.

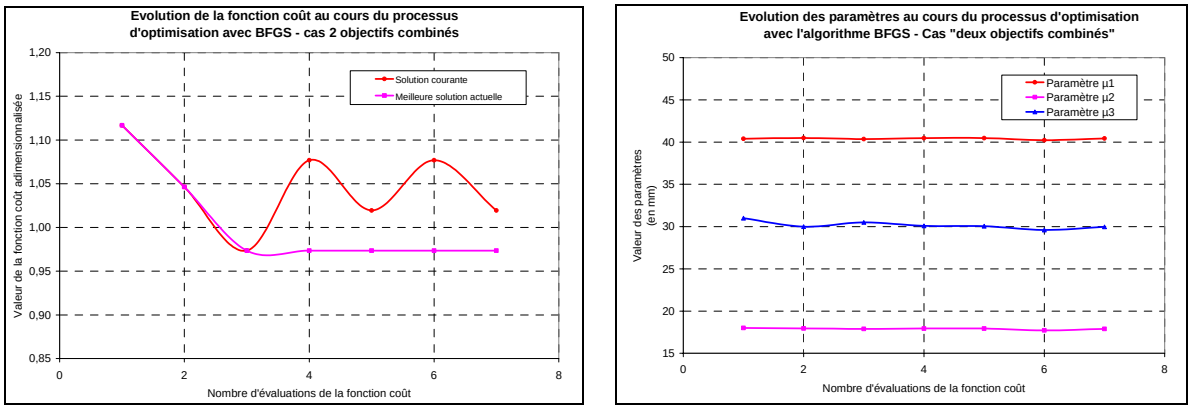



*Figure 4.23. Evolution de la fonction coût "2 objectifs combinés" et celle des paramètres au cours du processus d'optimisation avec l'algorithme de gradient BFGS.*

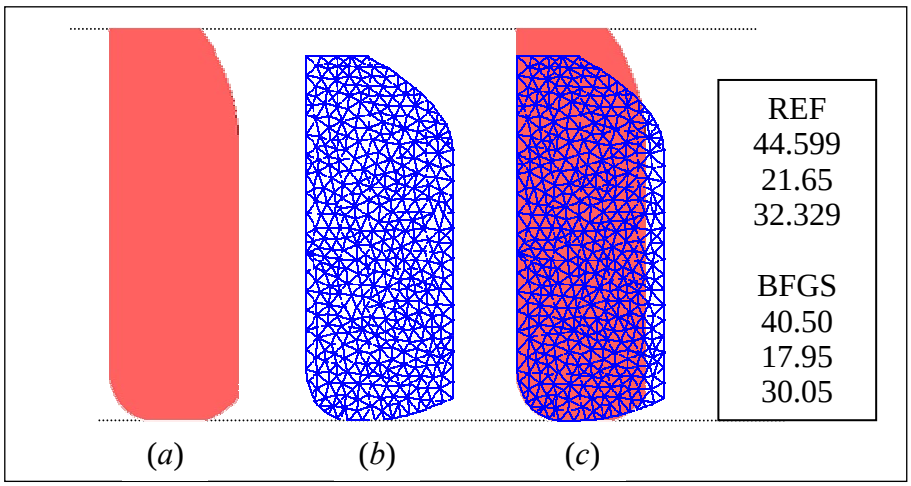



*Figure 4.24. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme de gradient BFGS (b) – Comparaison entre les deux préformes (c).*

### IV.3.2.2. Résultats d'optimisation avec SCPIP

Pour résoudre ce problème d'optimisation non convexe, nous faisons appel à l'autre algorithme d'optimisation à base de gradient, le SCPIP présenté dans la section I.5.1. En partant du même point de départ, le SCPIP arrive à faire mieux que BFGS, et améliore la fonction coût de 5,2%.

Avec cet algorithme, on constate une descente brutale (très forte) de la valeur de la fonction coût au début de l'algorithme. Puis, on observe une légère amélioration entre la deuxième et la troisième itération, qui est suivie par une stagnation de l'algorithme.

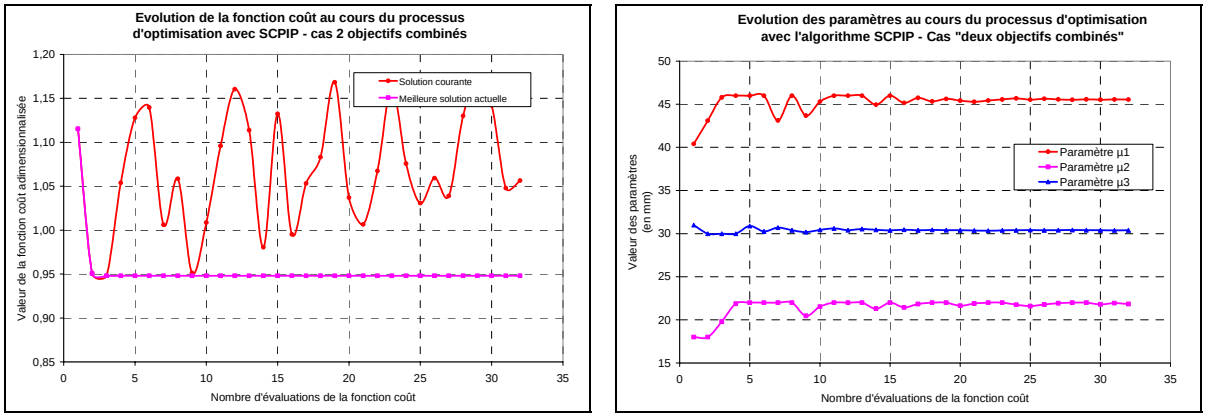



*Figure 4.25. Evolution de la fonction coût "2 objectifs combinés" et celle des paramètres au cours du processus d'optimisation avec l'algorithme SCPIP.*

Contrairement à l'algorithme BFGS dont le pas de descente est contrôlé, l'algorithme SCPIP fait varier très fortement les paramètres d'optimisation au début, puis plus légèrement par la suite.

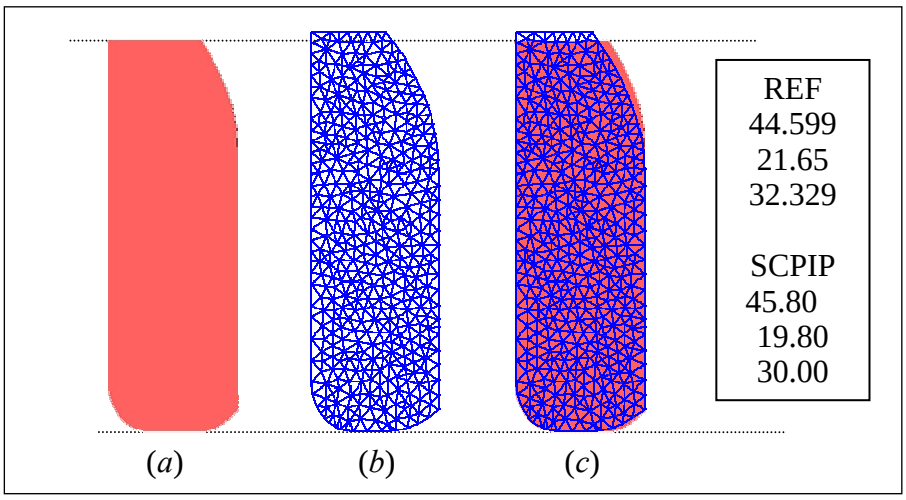



*Figure 4.26. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme de gradient SCPIP (b) – Comparaison entre les deux préformes (c).*

La préforme optimisée par SCPIP est légèrement différente de celle de référence. Elle est plus haute mais moins épaisse. Il faut remarquer que cette préforme est complètement différente de celle trouvée par BFGS, alors que les deux algorithmes utilisent le même point de départ. Ce qui met bien an relief l'incapacité de BFGS à s'extraire de l'extremum local où le choix du point de départ l'a précipité.

### IV.3.2.3. Résultats d'optimisation avec SE-Meta

La SE-Meta montre ici aussi son potentiel en trouvant une solution qui améliore la fonction coût de 9,05% (voir *Figure 4.27*).

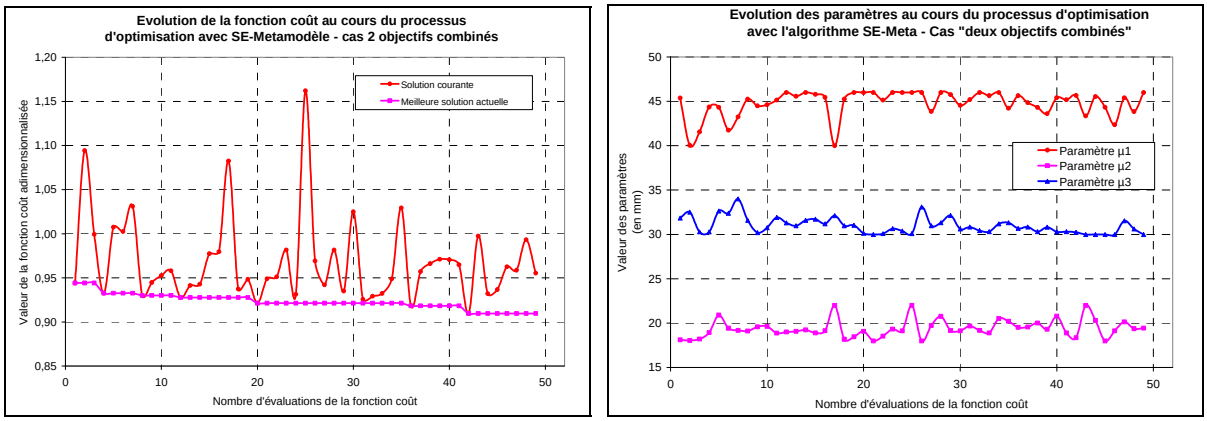



*Figure 4.27. Evolution de la fonction coût "2 objectifs combinés" et celle des paramètres au cours du processus d'optimisation avec la Stratégie d'Evolution avec Metamodele SE-Meta.*

Grâce à son caractère stochastique et un tirage chanceux, la SE-Meta a un excellent début en trouvant une solution meilleure que celle proposée par BFGS ainsi que celle de SCPIP. Cela signifie qu'un algorithme purement stochastique serait ici meilleur que les deux algorithmes de gradient, et est une indication sur le caractère multi-extrema de la fonction coût considérée. Il faut remarquer que SE-Meta commence par *2*nbparam* = 6 calculs sur des valeurs aléatoires. Ces 6 valeurs peuvent être considérées comme calculées "simultanément", sans

antériorité entre elles. Pendant le processus d'optimisation, la SE-Meta explore tout l'espace de recherche en faisant varier les paramètres dans toutes les directions (voir *Figure 4.27*), ce qui entraîne une forte fluctuation de la fonction coût. Cette exploration permet d'améliorer la solution initiale en trouvant d'autres solutions encore plus intéressantes.

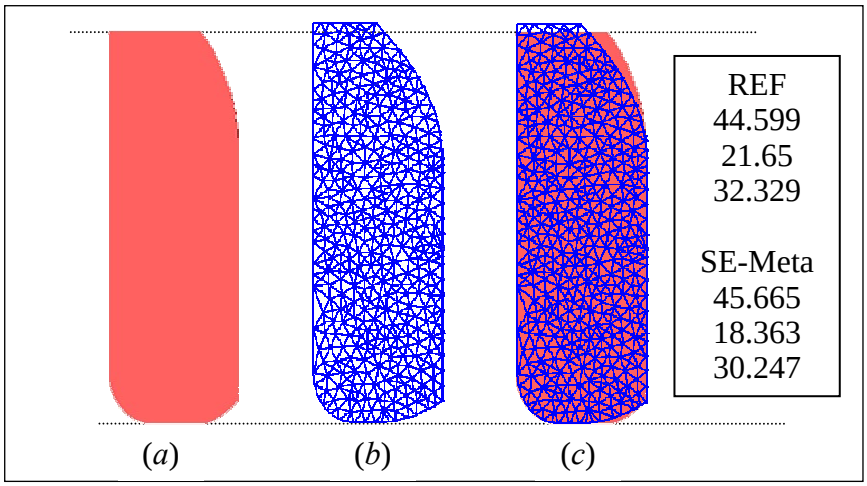



*Figure 4.28. Préforme de référence (a) – Préforme optimisée obtenue avec la SE-Meta (b) – Comparaison entre les deux préformes (c).*

La meilleure solution est trouvée à la 42$^{ième}$ évaluation, avec le jeu de paramètres ($\mu_1$=*43,87mm* ; $\mu_2$=*21,94mm* ; $\mu_3$=*35,10mm*) (voir *Figure 4.28*). Par rapport à la préforme de référence, elle est plus haute et plus pointue. Cette différence de forme permet une amélioration de la valeur de fonction coût de 9,05%.

### IV.3.2.4. Résultats d'optimisation avec AGMGA

Avec l'algorithme hybride AGMGA et une population de 120 individus avec 6 points maîtres (6 évaluations exactes par génération), plusieurs solutions intéressantes ont été trouvées. L'évolution de la fonction coût et celle des paramètres sont présentées sur la *Figure 4.29*.

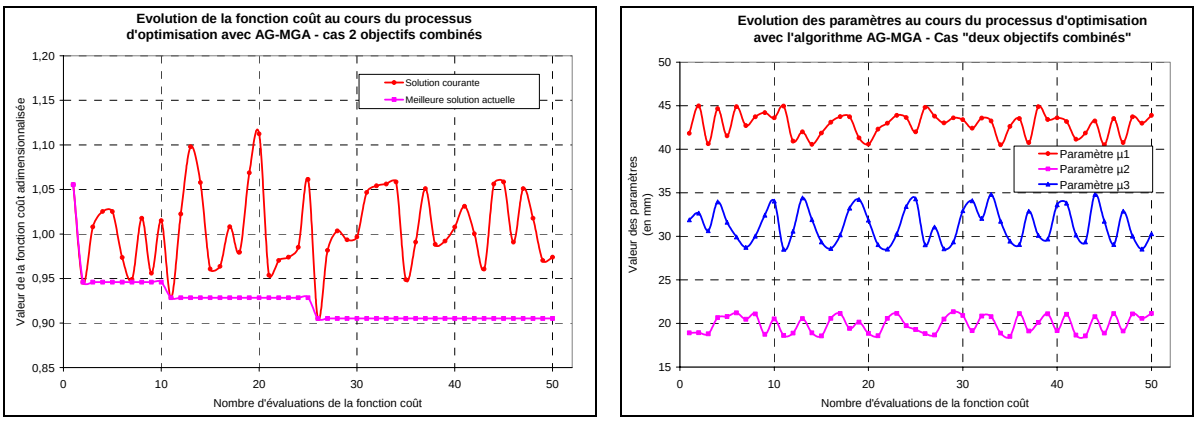



*Figure 4.29. Evolution de la fonction coût "2 objectifs combinés" et celle des paramètres au cours du processus d'optimisation avec l'algorithme hybride AGMGA.*

Parmi les bonnes solutions, celle trouvée à la 26$^{ième}$ évaluation est la meilleure, avec 9,5% d'amélioration. Elle est ainsi meilleure que celle proposée par la SE-Meta.

On note une forte fluctuation de l'évolution de la fonction et des paramètres d'optimisation (*Figure 4.29*), du début de l'optimisation jusqu'à la fin. Cela est assez différent de l'application précédente avec la fonction coût "énergie totale" seule, pour laquelle ces fluctuations se résorbent à la fin de l'optimisation. Une explication est que dans ce cas présent la fonction coût a plusieurs optima. En 50 évaluations, l'AGMGA n'arrive pas encore à stabiliser les solutions. A priori, il demandait davantage d'itérations de l'algorithme pour observer cette stabilisation.

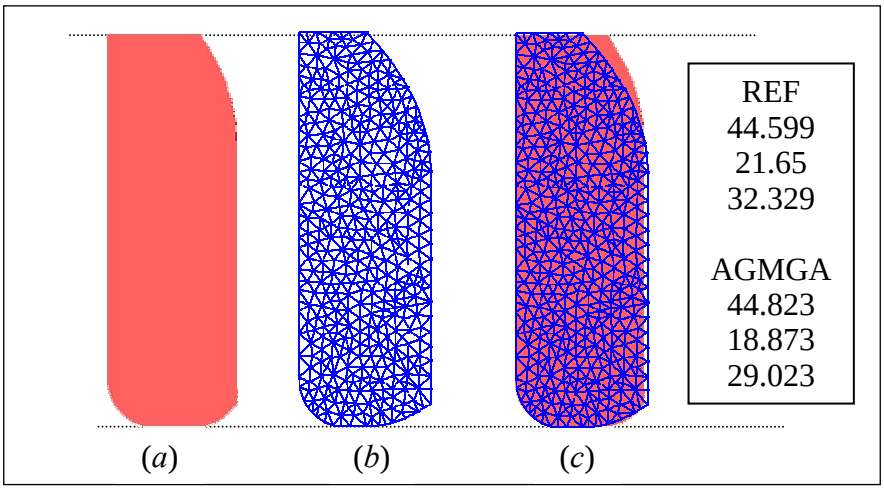



*Figure 4.30. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme hybride AGMGA (b) – Comparaison entre les deux préformes (c).*

Si on examine la préforme proposée par l'AGMGA, on constate qu'elle est différente de la préforme de référence et de celle proposée par SE-Meta. Elle est plus épaisse, plus pointue et légèrement plus haute que la première.

### IV.3.2.5. Résultats d'optimisation avec AGMGO

Le dernier algorithme est celui hybride AGMGO. Avec six évaluations par génération, l'AGMGO arrête sa recherche après 8 générations. La *Figure 4.31* présente les résultats obtenus avec cet algorithme.

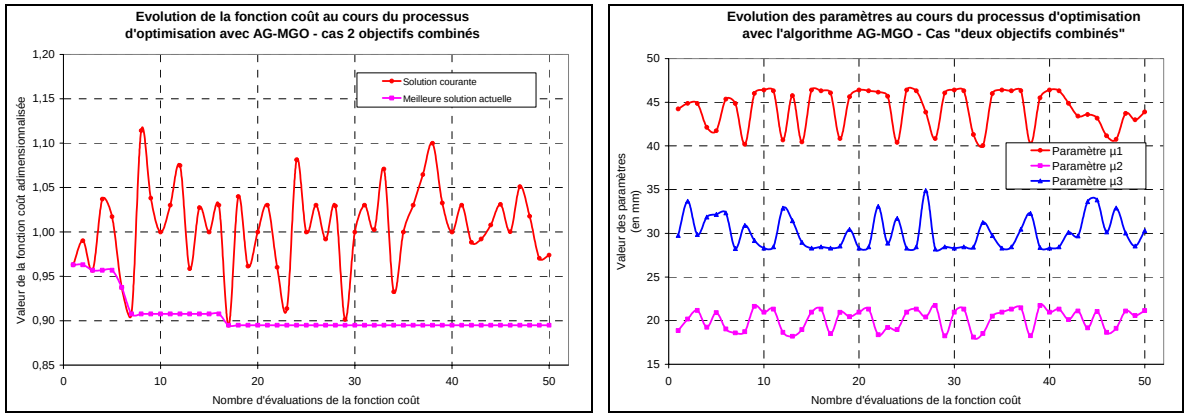



*Figure 4.31. Evolution de la fonction coût "2 objectifs combinés" et celle des paramètres au cours du processus d'optimisation avec l'algorithme hybride AGMGO.*

Comme avec les deux AE précédents, des l'initialisation (pour SE-Meta) ou la première génération (pour les AG), AGMGO trouve une solution meilleure que la référence (cequi confirme que BFGS a plutôt été malchanceux lors du tirage aléatoire de son point de départ). Pendant le processus d'optimisation, l'AGMGO explore bien tout l'espace de recherche, comme le montre l'évolution de la fonction coût et des paramètres (*Figure 4.31*) qui varient dans toutes les directions. La valeur de la fonction coût fluctue fortement pendant l'optimisation comme observé avec la fonction coût "énergie totale" seule.

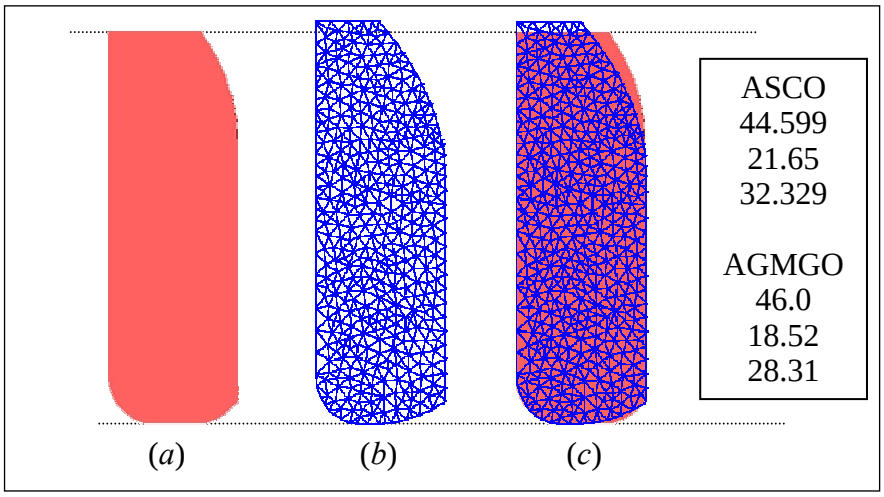



*Figure 4.32. Préforme de référence (a) – Préforme optimisée obtenue avec l'algorithme hybride AGMGO (b) – Comparaison entre les deux préformes (c).*

La meilleure solution est obtenue à la 17$^{ième}$ évaluation de la fonction coût avec une amélioration de 10,5% de sa valeur. C'est l'amélioration la plus importante obtenue pour ce cas. La préforme optimale proposée est présentée sur la *Figure 4.32*. Si on la compare à celle de référence, on note une différence : celle optimisée par l'AGMGO est plus haute et plus pointue mais elles ont presque la même épaisseur. Notons que les différentes préformes obtenues sont ainsi fort différentes, preuve du caractère chahuté et multi-optima de la fonction coût étudiée.

### IV.3.2.6. Comparaisons des algorithmes

Nous présentons une synthèse des résultats obtenus avec tous les algorithmes d'optimisation avec la fonction coût "2 objectifs combinés".

Pour chaque algorithme d'optimisation, le *Tableau 4.2* indique le numéro $N_{opt}$ de la simulation où la meilleure solution est obtenue ainsi que le nombre total de simulations $N_{tot}$ effectuées. Il présente la valeur de la fonction coût et des améliorations obtenues avec les préformes optimales. Il présente enfin le coût de chaque algorithme pour trouver une solution équivalente à celle trouvée par SE-Meta, en valeur relative par rapport à celui de la SE-Meta (on considère toujours que le calcul du gradient représente 30% du calcul de la simulation, ce qui est plutôt une extrapolation pour des problèmes plus complexes (comportement non linéaire) et avec plus de nœuds qu'ici).

Un premier regard au *Tableau 4.2* montre que tous les algorithmes trouvent une meilleure solution que le cas de référence et que la valeur des deux fonctions coûts peut être réduite de

plus de 10%. Les améliorations obtenues se divisent en 2 groupes d'amplitudes différentes. Le groupe 1 contient les algorithmes à direction de descente qui améliorent la fonction coût seulement de 3% à 5%. Le groupe 2 réunit les trois autres algorithmes évolutionnaires qui apportent des améliorations plus significatives de l'ordre de 10% grâce à leur capacité à rechercher une solution plus globale.

| | $N_{opt}/N_{tot}$ | $\Phi_{tot}$ | *Amélioration de* $\Phi_{ener}$ | *Amélioration de* $\Phi_{repli}$ | *Amélioration de* $\Phi_{tot}$ | *Coût % SE-Meta* |
|---|---|---|---|---|---|---|
| *Référence* | - | 1 | - | - | - | - |
| *BFGS* | 3/7 | 0,973 | -1,8% | -3,6% | -2,7% | Stagne |
| *SCPIP* | 3/32 | 0,948 | -6,9% | -3,5% | -5,2% | Stagne |
| *SE-Meta* | 42/50 | 0,9095 | -10,5% | -7,6% | -9,05% | 1 |
| *AGMGA* | 26/50 | 0,905 | -8,9% | -10,1% | -9,5% | 0,8 |
| *AGMGO* | 17/50 | 0,895 | -10,3% | -10,7% | -10,5% | 0,53 |

*Tableau 4.2 : Synthèse des résultats d'optimisation avec différents algorithmes pour le benchmark du forgeage d'une roue dentée avec la fonction coût 2 objectifs combinés.*

La *Figure 4.33* synthétise la convergence des algorithmes. La *Figure 4.34* regroupe les préformes proposées et celle de référence.

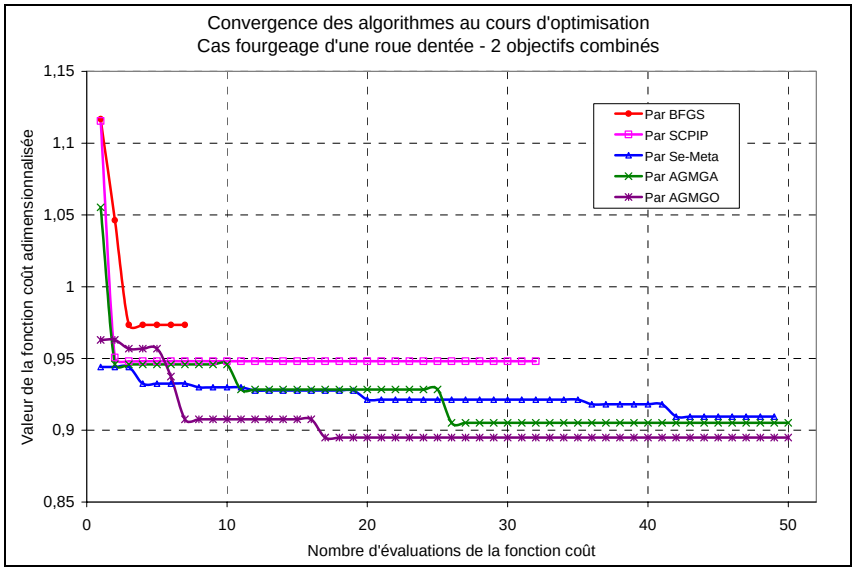



*Figure 4.33. Convergence de différents algorithmes au cours du processus d'optimisation de la fonction coût "2 objectifs combinés".*

La convergence des algorithmes présentée sur la *Figure 4.33* montre clairement que les algorithmes globaux (comme SE-Meta, AGMGA, AGMGO) sont plus performants que les locaux (comme BFGS). Pour les algorithmes locaux, les résultats dépendent fortement du point de départ mais ils sont obtenus en seulement quelques itérations. Les algorithmes

globaux ont souvent besoin de beaucoup plus d'évaluations pour atteindre leur optimum, mais on observe ici que dès la première génération, c'est-à-dire dès les 6 premiers calculs effectués pour des valeurs aléatoires des paramètres (que ce soit pour SE-Meta ou AGMGA, AGMGO) la solution obtenue est meilleure ou équivalente à celle proposée par SCPIP ou BFGS. Autrement dit, un tirage aléatoire de 6 valeurs est plus efficace que tout algorithme de gradient.

On compare maintenant les résultats obtenus avec les deux nouveaux algorithmes hybrides à celui obtenu par la SE-Meta. Dans le *Tableau 4.2*, nous constatons que la solution de la SE-Meta minimise au mieux l'énergie totale, tandis que les algorithmes hybrides minimisent préférentiellement plus le potentiel de repli. L'AG-MGO s'avère particulièrement performant en combinant une grande réduction des deux objectifs. Toutefois, la différence entre les solutions proposées par la SE-Meta et les algorithmes hybrides est assez fable comme le montre la *Figure 4.34*.

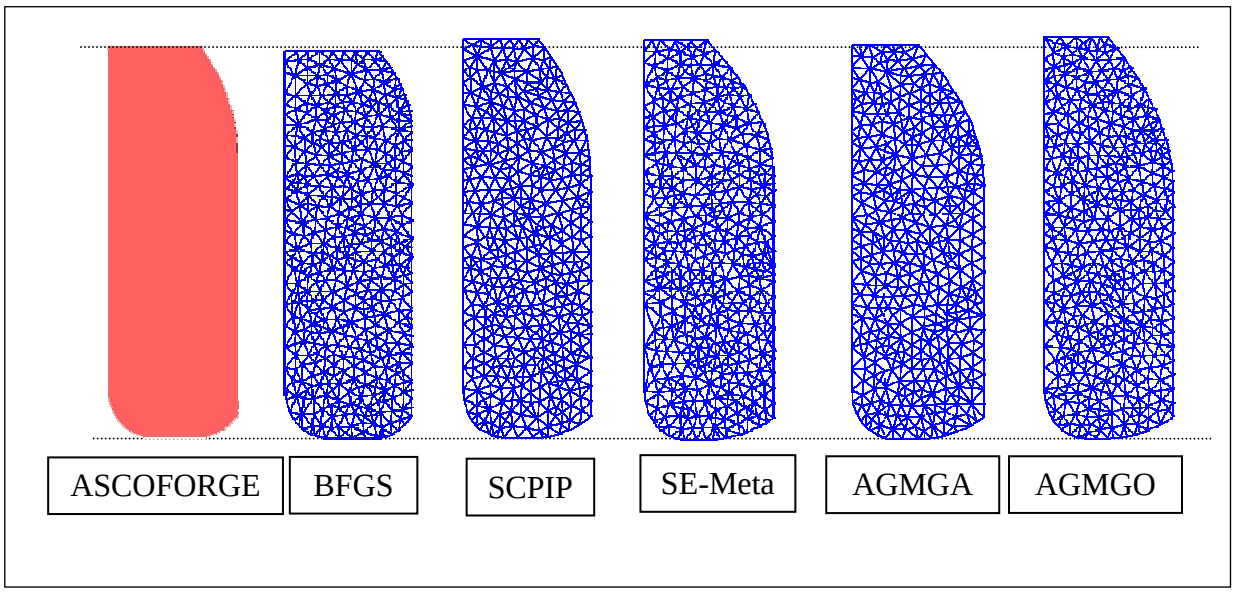



*Figure 4.34. Comparaison des préformes proposées par les différents algorithmes d'optimisation*

Une autre remarque sur le coût de l'optimisation peut être formulée. Les deux algorithmes permettent d'obtenir uns solution équivalente en beaucoup moins de temps. Notamment l'AGMGO, permet de diviser par deux ce coût des calculs, ce qui permet exercer les qualités de cette hybridation.

## IV.4. Conclusions

Différentes conclusions peuvent être tirées :

- tous les algorithmes d'optimisation trouvent de meilleures solutions que la solution de référence
- les algorithmes de type évolutionnaire sont plus performants que les algorithmes à direction de descente (comme BFGS ou SCPIP)

- les trois algorithmes évolutionnaires (AGMGA et AGMGO et SE-Meta) proposent des améliorations semblables et d'environ 10%.

- les deux algorithmes hybrides s'avèrent surtout plus rapides (jusqu'à deux fois plus rapide que la SE-Meta) pour trouver une solution intéressante.

A travers cet exemple, nous voyons bien que l'on peut appliquer les algorithmes évolutionnaires à la résolution des problèmes complexes d'optimisation de forgeage 3D avec un coût acceptable (moins de 50 simulations).

# Chapitre V

# OPTIMISATION DE LA GEOMETRIE DES OUTILS DE PREFORME

## V.1. Introduction

Le benchmark précédent nous a donné une vue générale sur le fonctionnement des algorithmes d'optimisation appliqués au forgeage. Toutefois, le choix des fonctions coûts autres que celle de l'énergie totale est assez académique. La fonction coût "défaut repli" a juste été introduite pour créer un cas d'optimisation avec nombreux extrema car il n'y a pas de repli sur la pièce forgée considérée compte tenue de la préforme dessinée et des faibles variations de celle-ci durant l'optimisation.

*Figure 5.1. Exemple de triaxe*

Afin d'étudier un vrai problème d'optimisation de forgeage (et d'utiliser l'extension du calcul du gradient que nous avons réalisée), nous examinons le cas de forgeage d'un triaxe (*Figure 5.1*).

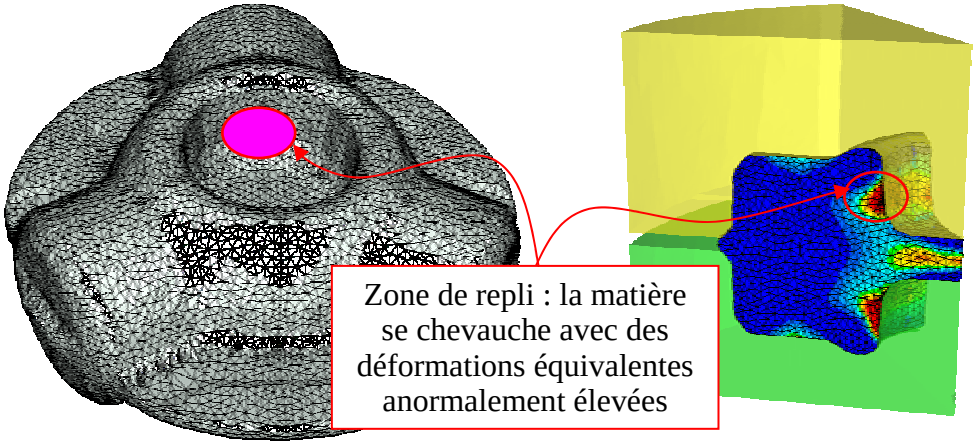



*Figure 5.2. Défaut repli sur le Triaxe forgé avec la gamme de forgeage standard*

Lors du forgeage d'un triaxe en 2 passes avec la gamme standard, on observe un repli sur la pièce finale (*Figure 5.2*). La zone de repli représente une zone de faiblesse inacceptable. Ce défaut entraîne donc le rebus de la pièce.

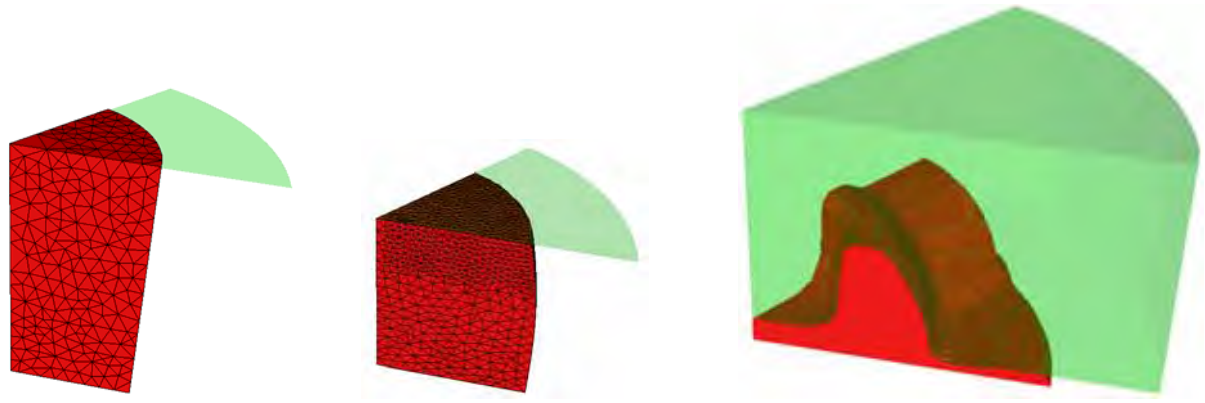
Un problème d'optimisation émerge pour éviter ce défaut : trouver la forme de l'outil de préforme pour qu'il n'y ait plus de repli sur la pièce forgée finale. Ce chapitre a pour but d'étudier ce problème d'optimisation.

# V.2. Présentation du problème d'optimisation

Dans cette section, nous allons présenter brièvement le problème d'optimisation. Nous décrirons tout d'abord le cas de forgeage du triaxe.

## V.2.1. Description du cas de forgeage d'un triaxe

Ce cas de forgeage a été présenté dans la section III.3.2. Compte tenu de la symétrie du triaxe, nous modélisons seulement la moitié supérieure d'un sixième de la pièce (un douzième du Triaxe), comme montré sur la *Figure 5.3*.



(a) (b) (c)

*Figure 5.3. Un douzième de la pièce initiale (a) – Pièce finale de la première passe = Préforme de la deuxième passe (b) – Un douzième du triaxe forgé*

Le comportement du matériau est viscoplastique. Avec un maillage de 35000 éléments (9400 nœuds), le temps nécessaire pour une simulation complète sans calcul du gradient est de 4h, et il est de 5h30' avec calcul du gradient sur un PC mono processeur à 2,4GHz.

## V.2.2. Paramétrisation de la préforme d'outillage

L'outil de préforme du forgeage d'un triaxe est axisymétrique. Il est paramétrisée par une Bspline. Nous ne présentons pas cette paramétrisation car elle a été décrite dans la section III.3.2.

## V.2.3. Fonction coût étudiée

Pour ce problème, la fonction coût repli est le seul objectif à minimiser.

# V.3. Résultats d'optimisation

Nous allons d'abord étudier ce problème d'optimisation avec deux paramètres d'optimisation pour examiner son comportement. Puis le cas est étudié avec trois paramètres. Enfin, un test sur 5 paramètres d'optimisation est réalisé.

## V.3.1. Optimisation avec 2 paramètres

### V.3.1.1. Résumé du problème

Dans le but d'examiner la nature du problème d'optimisation, nous l'étudions d'abord avec 2 paramètres. La disposition des paramètres sur le polygone de contrôle de la Bspline est présentée sur la *Figure 5.4*.

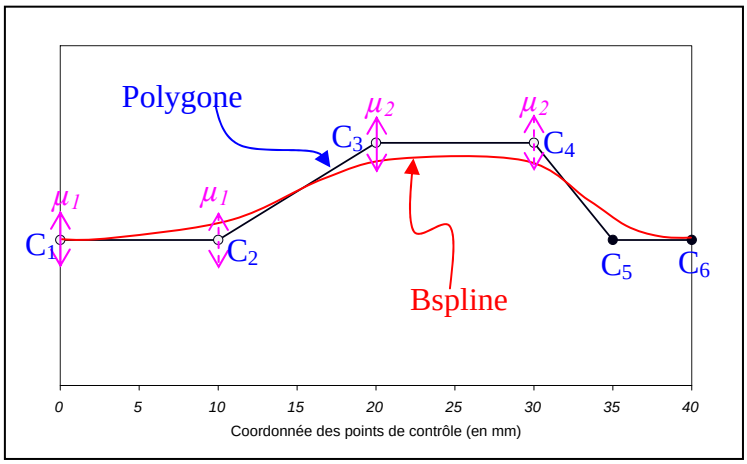



*Figure 5.4. Disposition des paramètres sur le polygone de contrôle de Bspline*

Le polygone de contrôle de la Bspline a 6 points $C_1$, $C_2$,..., $C_6$. Les 4 points $C_1$, $C_2$, $C_3$, $C_4$ peuvent se déplacer verticalement. Les deux points $C_5$ et $C_6$ sont fixes. Parmi les 4 points mobiles, $C_1$ et $C_3$ sont actifs alors que $C_2$ et $C_4$ sont passifs. C'est-à-dire $C_2$ et $C_4$ se déplacent identiquement respectivement à $C_1$ et $C_3$. Les déplacements des 2 points actifs $C_1$ et $C_3$ sont donc nos paramètres d'optimisation $\mu_1$ et $\mu_2$.

Le problème d'optimisation peut être présenté comme suit :

$$\text{Minimiser } \Phi_{repli}(\mu_1,\mu_2) \text{ avec } -10mm \le \mu_1, \mu_2 \le 20mm$$

Nous allons résoudre ce problème avec différents algorithmes: les deux algorithmes à direction de descente BFGS et SCPIP, la SE-Meta et les deux algorithmes hybrides l'AGMGA et l'AGMGO. Les résultats d'optimisation sont présentés dans le paragraphe suivant.

### V.3.1.2. Résultats d'optimisation

Le *Tableau 5.1* présente une synthèse des résultats d'optimisation obtenus. Pour tous les algorithmes d'optimisation, il indique le numéro de la simulation $N_{opt}$ où on a obtenu la meilleure solution en référence au nombre total de simulations $N_{tot}$ effectuées. Il présente la

valeur des paramètres ($\mu_1$, $\mu_2$) optimaux, la valeur de la fonction coût, et l'amélioration apportée. Ce tableau répond aussi à la question si le repli est enlevé ou non.

| | $N_{opt}/N_{tot}$ | $\mu_{1opt}$ | $\mu_{2opt}$ | $\Phi_{repli\ min}$ | ***Amélioration*** | ***OK ?*** |
|---|---|---|---|---|---|---|
| ***Cas REF*** | - | 0 | 0 | 10,49 | - | Non |
| ***BFGS*** | 7/14 | 6,99 | 18,41 | 10,20 | -2,8% | Non |
| ***SCPIP*** | 8/24 | 1,03 | 0,47 | 9,29 | -11,4% | Oui |
| ***Se-Meta*** | 37/48 | -7,87 | -10 | 8,14 | -22,4% | Oui |
| ***AGMGA*** | 16/48 | -8,47 | -4,38 | 9,13 | -13,0% | Oui |
| ***AGMGO*** | 20/48 | -5,65 | -7,52 | 9,00 | -14,2% | Oui |

*Tableau 5.1 : Synthèse des résultats d'optimisation avec différents algorithmes pour le cas de forgeage d'un triaxe avec 2 paramètres*

D'après ce tableau, tous les algorithmes réduisent la potentialité de repli par rapport à la référence. Cependant, l'algorithme itératif BFGS ne trouve qu'une solution avec une fonction coût de 10,20 soit seulement 2,8% d'amélioration. Le SCPIP présente un meilleur résultat. Il trouve un outil de préforme avec une potentialité de repli de 9,29 , qui apporte donc une amélioration de 11,4%. Dans ce cas d'optimisation, les trois algorithmes évolutionnaires démontrent encore une fois leur supériorité par rapport aux algorithmes à base de gradient.

En effet, l'optimisation basée sur la méthode SE-Meta donne la meilleure préforme optimale avec une fonction coût de 8,14 , ce qui correspond à une amélioration de 22,4% par rapport à la référence.

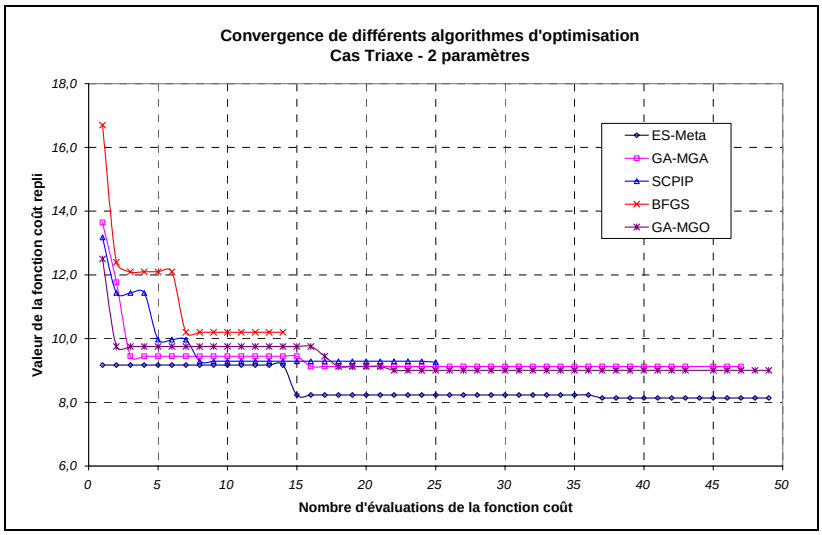



*Figure 5.5. Convergence des algorithmes d'optimisation pour le cas Triaxe – 2 paramètres*

La *Figure 5.5* présente la convergence des différents algorithmes d'optimisation employés. On observe sur cette figure que les algorithmes se comportent différemment. Le BFGS et le

SCPIP obtiennent leur solution optimale en quelques itérations seulement, tandis que les trois algorithmes évolutionnaires ont besoin de beaucoup plus (16 évaluations avec l'AGMGA, 20 avec AGMGO et 37 avec la SE-Meta). Pourtant on voit qu'au moment où le BFGS obtient sa solution optimale (à la $7^{ième}$ itération), les algorithmes évolutionnaires ont déjà trouvé de meilleures solutions.

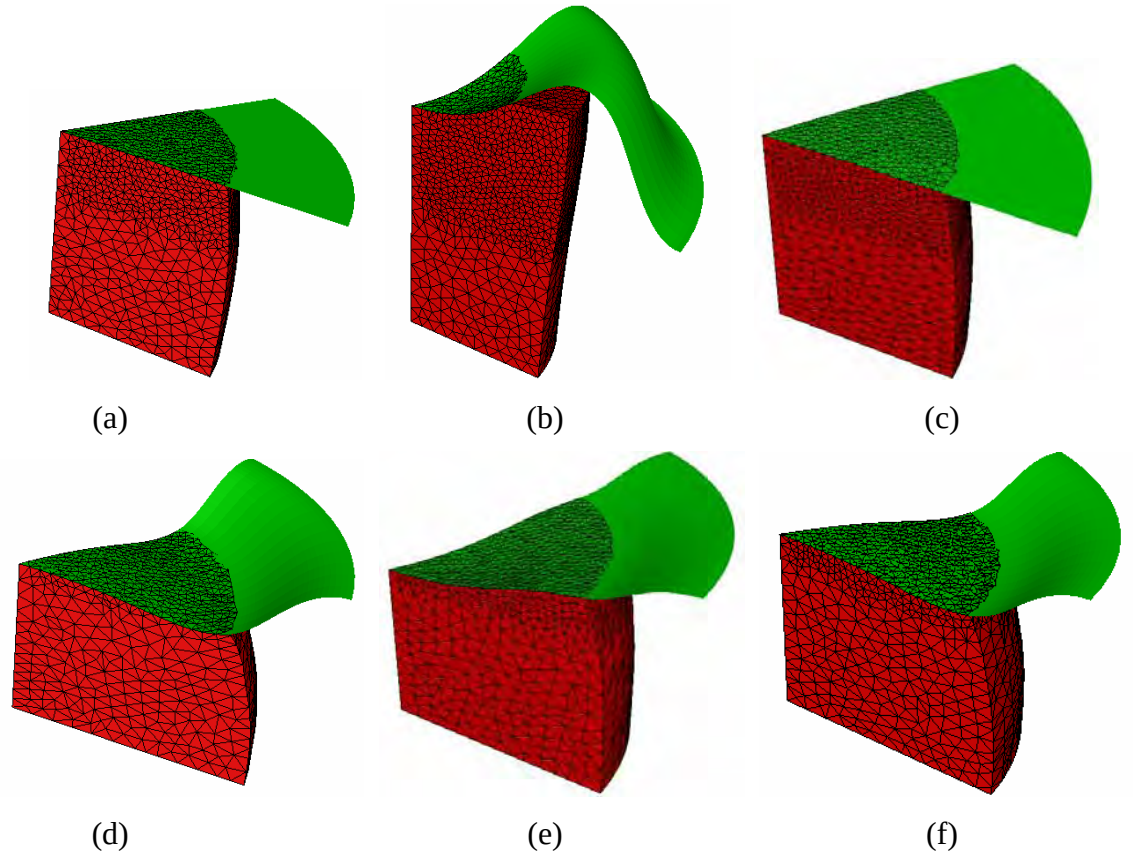



*Figure 5.6.Pièces forgées finales de la première passe avec différentes outils de préforme : (a) de référence; (b) optimisé par BFGS ; (c) SCPIP ; (d) SE-Meta ; (e) AGMGA ; (f) AGMGO*

Les formes des outils de préforme optimisés sont présentées sur la *Figure 5.6* Nous voyons que le BFGS donne un outil de préforme complètement différent des autres solutions. Pourtant, la fonction coût n'est pas beaucoup améliorée avec cette préforme. D'autre part, le SCPIP trouve une préforme légèrement différente de celle de référence. Par ailleurs, les algorithmes évolutionnaires donnent des solutions très similaires qui introduisent les améliorations les plus significatives sur la fonction coût.

Les résultats présentés montrent que tous les algorithmes arrivent à trouver une solution ayant une potentialité de repli inférieure à celle de référence. Cependant, la question principale est de savoir si le repli est bien enlevé sur la pièce finale. Une brève réponse à cette question est présentée dans le *Tableau 5.1*. La *Figure 5.7* illustre également cette réponse par les images des pièces forgées finales. Dans cette figure, la pièce forgée avec l'outil de préforme optimisé par le BFGS présente encore une zone de repli (*Figure 5.7b*). Dans cette zone, on observe une déformation équivalente beaucoup plus élevée que les autres parties de la surface libre de la pièce. Le produit final donné par le SCPIP est meilleur que celui de BFGS. Le repli diminue nettement et on obtient une déformation équivalente plus homogène (*Figure 5.7c*), sauf celle de quelques éléments en jaune qui est légèrement plus élevée.

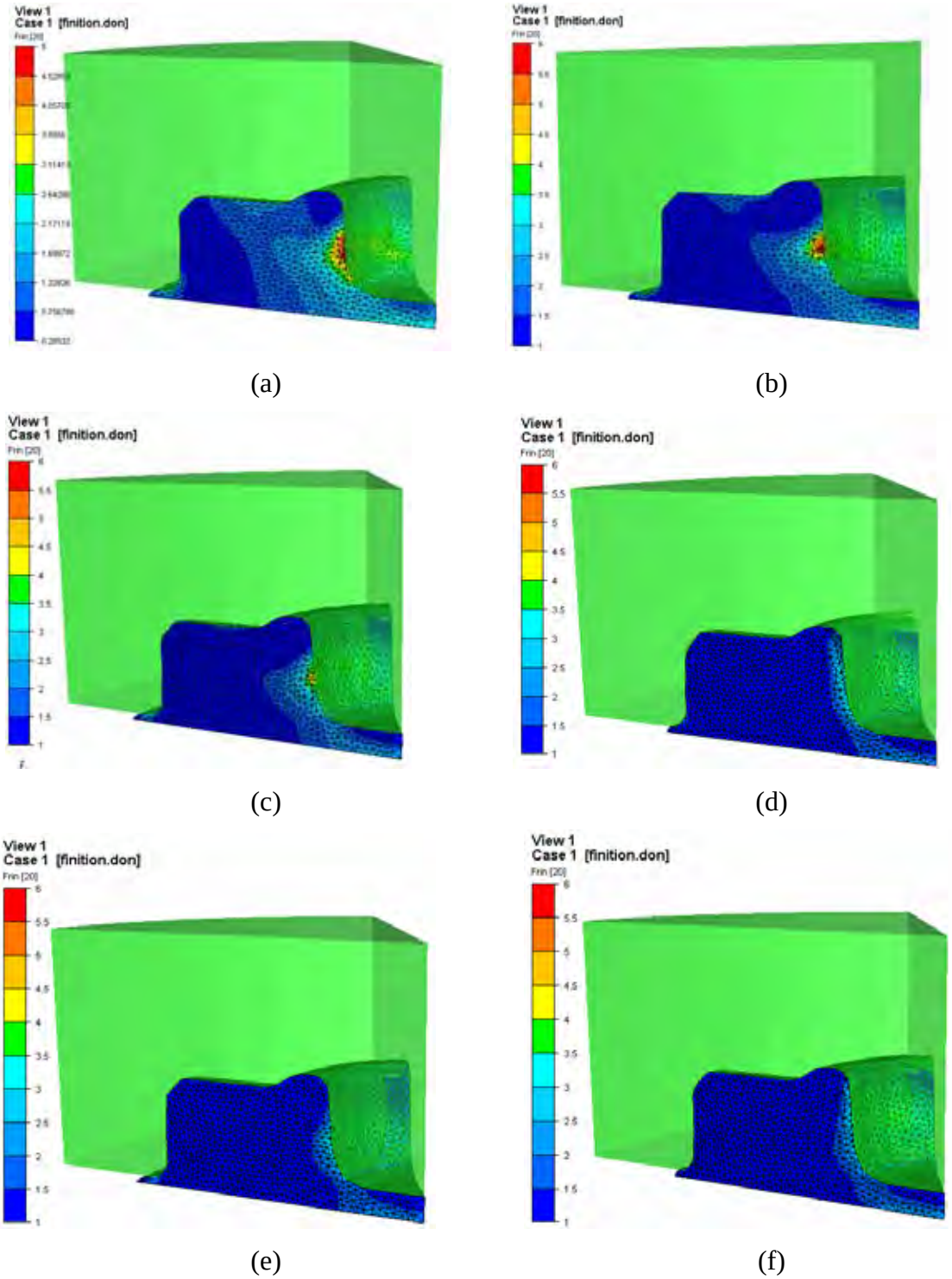



*Figure 5.7.Pièces forgées finales de la deuxième passe avec différents outils de préforme : (a) de référence; (b) optimisé par BFGS ; (c) SCPIP ; (d) SE-Meta ; (e) AGMGA ; (f) AGMGO*

On observe maintenant la pièce forgée avec l'outil optimisé par la SE-Meta (*Figure 5.7d*). On constate qu'il n'y a plus de repli. L'outil de préforme optimisé donne une pièce forgée finale avec une déformation bien homogène. Les deux algorithmes hybrides arrivent au même résultat (*Figure 5.7e,f*).

A partir de la valeur des points obtenus par les différents algorithmes d'optimisation présentés précédemment, nous construisons une surface de réponse (ou plan d'expérience) avec Matlab pour décrire la nature du problème. Cette surface est présentée sur la *Figure 5.8*. Sur la *Figure 5.9a*, nous avons la distribution des points dans l'espace. La position des solutions optimales obtenues est aussi décrite sur la *Figure 5.9b* avec les courbes des isovaleurs.

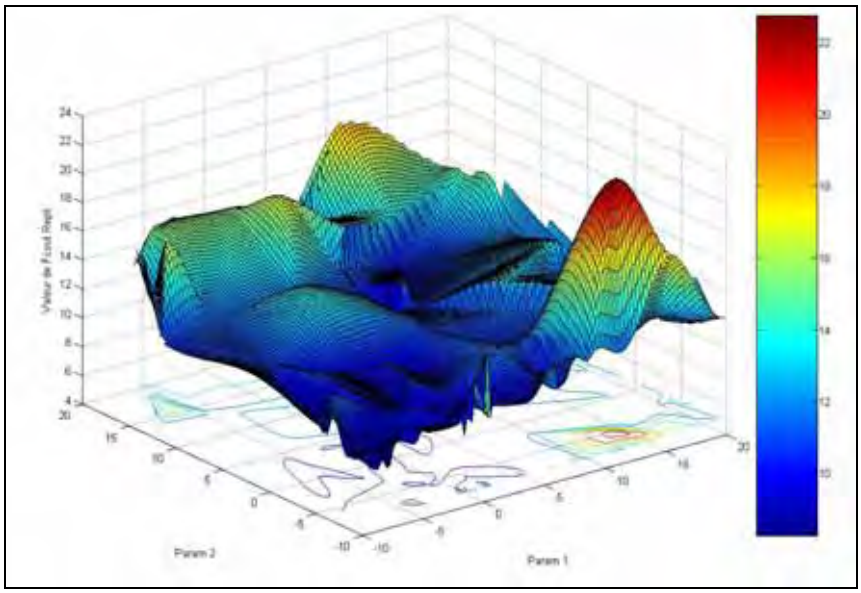
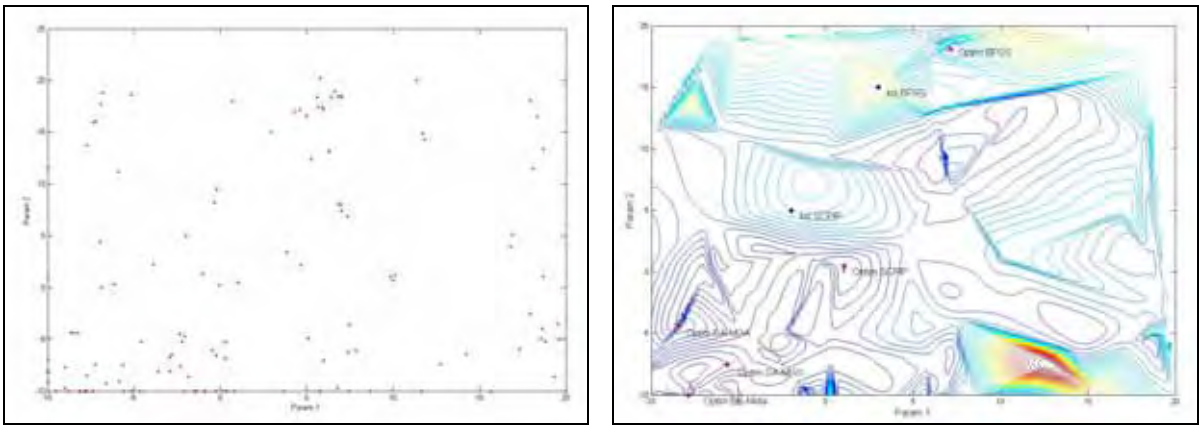


*Figure 5.8.Surface de réponse dessinée avec les valeurs des points obtenus de tous les algorithmes d'optimisation*



(a) (b)

*Figure 5.9.Distribution des points de calcul dans l'espace de recherche (a) et Localisation des solutions optimales dans l'espace de recherche avec des courbes d'isovaleurs*

Ces figures montrent que notre problème d'optimisation possède de nombreux extrema. Elles expliquent pourquoi les algorithmes à direction de descente comme le BFGS ne sont pas performant ici : ils ne trouvent que l'optimum le plus proche de leur point de départ, donc un optimum local.

Ces résultats montrent à nouveau la supériorité des algorithmes évolutionnaires pour trouver des solutions plus globales, et notamment la robustesse de la SE-Meta.

## V.3.2. Optimisation avec 3 paramètres

Afin de vérifier l'efficacité des algorithmes évolutionnaires et d'étudier l'influence du nombre de paramètres, nous essayons de résoudre le même problème avec cette fois 3 paramètres.

### V.3.2.1. Résumé du problème

Nous gardons la même paramétrisation que celle utilisée dans le cas précédent, mais parmi les quatre points mobiles, trois sont actifs ($C_1$, $C_3$, $C_4$) et sont des 3 paramètres à optimiser. Ils sont présentés sur la *Figure 5.10* sous le nom $\mu_1$, $\mu_2$, $\mu_3$.

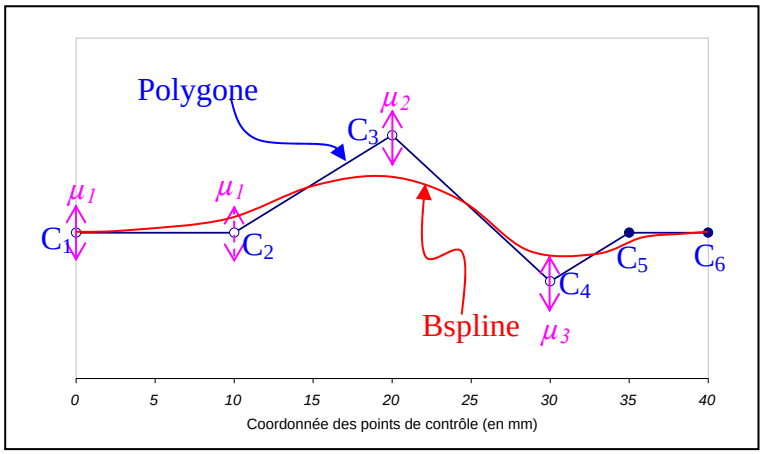



*Figure 5.10. Disposition des paramètres sur le polygone de contrôle de la Bspline*

Le problème d'optimisation est reformulé par :

$$\text{Minimiser } \Phi_{repli}(\mu_1,\mu_2,\mu_3) \quad \text{avec } -10mm \le \mu_1, \mu_2, \mu_3 \le 20mm$$

### V.3.2.2. Résultats d'optimisation

Les résultats de l'optimisation sont résumés dans le *Tableau 5.2*.

| | $N_{opt}/N_{tot}$ | $\mu_{1opt}$ | $\mu_{2opt}$ | $\mu_{3opt}$ | $\Phi_{repli\ min}$ | **Amélioration** | **OK ?** |
|---|---|---|---|---|---|---|---|
| ***Cas REF*** | - | 0 | 0 | 0 | 10.49 | - | Non |
| ***BFGS*** | 3/12 | 7.02 | 2.92 | 6.55 | 10.25 | -2.3% | Non |
| ***SCPIP*** | 9/18 | -4.25 | -6.01 | -3.07 | 9.17 | -12.6% | Oui |
| ***Se-Meta*** | 30/48 | -7.26 | -3.84 | -10 | 8.35 | -20.4% | Oui |
| ***AGMGA*** | 32/48 | -4.56 | -8.42 | -5.21 | 9.03 | -14.0% | Oui |
| ***AGMGO*** | 28/48 | -8.54 | -9.10 | -4.87 | 8.83 | -15.8% | Oui |

*Tableau 5.2 : Synthèse des résultats d'optimisation avec différents algorithmes pour cas forgeage d'un triaxe avec 3 paramètres*

Ce tableau montre que l'optimisation à 3 paramètres donne les mêmes résultats que celle à 2 paramètres. Encore une fois, on constate que tous les algorithmes réduisent la potentialité de repli par rapport à la référence. Le BFGS est encore une fois le moins performant et apporte seulement 2,3% d'amélioration. Il ne donne pas une solution satisfaisante : la pièce forgée finale présente toujours un défaut repli. La SE-Meta est encore une fois l'algorithme le plus performant dans ce problème. Elle trouve la meilleure solution optimale avec une fonction coût de valeur de 8,35 et une amélioration de 20,4%. Sur la pièce forgée finale, le défaut repli est totalement enlevé.

En partant du même point que le BFGS, le SCPIP trouve cette fois un outil de préforme avec une potentialité de repli de 9,17, qui apporte une amélioration de 12.6% avec une pièce forgée finale sans repli. SCPIP est toujours meilleur que le BFGS, mais moins robuste que les algorithmes pilotés par les AE. Les deux algorithmes hybrides donnent des solutions optimales avec des améliorations du même ordre de grandeur (de 15% environ) avec des pièces forgées finales sans repli. Pour ce cas, ces deux algorithmes sont moins performants que la SE-Meta mais ils sont plus intéressants que les algorithmes à direction de descente.

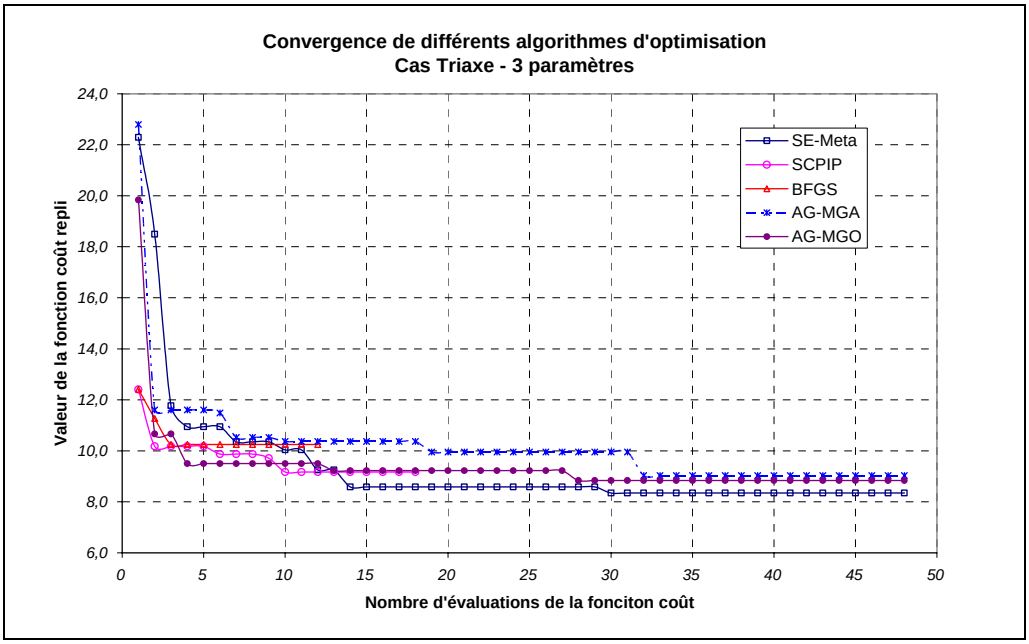



*Figure 5.11. Evolution de la fonction coût repli au cours d'optimisation– cas du Triaxe à 3 paramètres*

Les profils de convergence des algorithmes d'optimisation sont synthétisés sur la *Figure 5.11* En terme de vitesse d'obtention de la solution optimale, on voit encore une fois que les algorithmes à direction de descente trouvent très rapidement leur optimum (en moins de 10 évaluations de fonction coût). Les algorithmes évolutionnaires ont toujours besoin beaucoup plus d'évaluations. Chez les algorithmes évolutionnaires, on constate que la fonction coût de la meilleure solution courante diminue rapidement au début de l'optimisation, puis, plus lentement à la fin : pour chaque saut de la fonction coût, il faut plusieurs évaluations. En fait, ce phénomène est dû au fait que les algorithmes évolutionnaires sont initialisés par des mauvaises solutions dans ce cas. Il est donc facile à trouver une meilleure solution avec un grand saut de la fonction coût. Une fois qu'une bonne solution est obtenue, il faut explorer plus largement l'espace de recherche pour trouver une meilleure solution et le saut de la fonction coût est plus petit à la fin du processus d'optimisation.

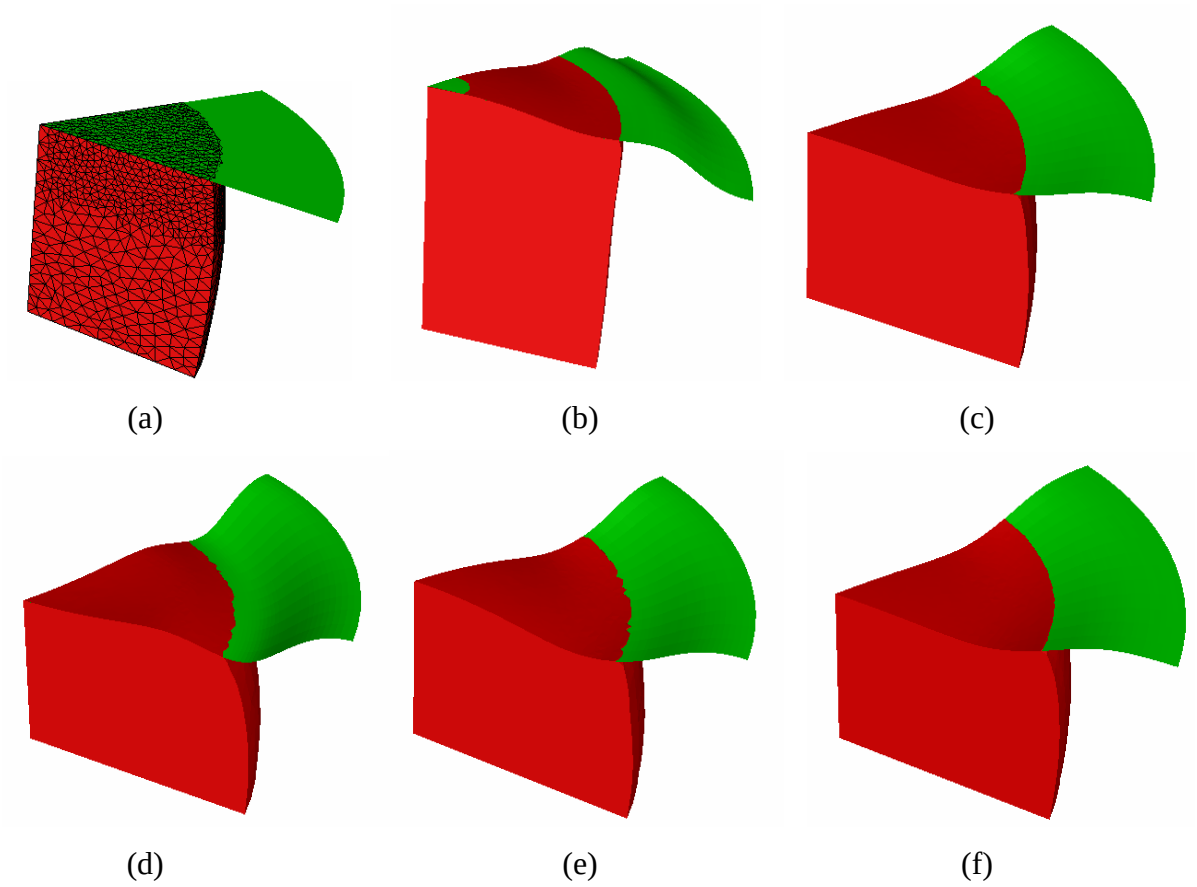



*Figure 5.12. Pièces produites de la première passe avec différentes outils de préforme : (a) de référence; (b) optimisé par BFGS ; (c) SCPIP ; (d) SE-Meta ; (e) AGMGA ; (f) AGMGO*

Comparons maintenant la forme des outils de préforme optimisées présentées sur la *Figure 5.12*. Encore une fois, l'outil de préforme proposée par le BFGS est complètement différent de ceux proposées par des autres algorithmes. La valeur de fonction coût n'est pas beaucoup améliorée. Le BFGS est coincé dans un optimum local, le repli est toujours présent sur la pièce finale. Le SCPIP trouve cette fois un outil de préforme différent de celui du cas à 2 paramètres. Sa forme est similaire aux formes obtenues par les trois AE.

Les images présentées sur la *Figure 5.13* expliquent bien la différence entre les valeurs des fonctions coûts obtenues avec les différents algorithmes d'optimisation. Sur la pièce finale obtenue avec le BFGS, nous observons toujours une zone avec une déformation équivalente anormalement élevée (zone en rouge sur la *Figure 5.13b*) de la surface libre. Le repli est présent encore sur cette zone. La valeur de la fonction coût repli est élevée. En revanche, avec les pièces finales obtenues avec les quatre algorithmes SCPIP, SE-Meta, AGMGA, AGMGO, la déformation équivalente est bien homogène sur la surface libre. La fonction coût repli prend des valeurs plus faibles que celle du BFGS. Le repli est donc enlevé pour toutes ces pièces.

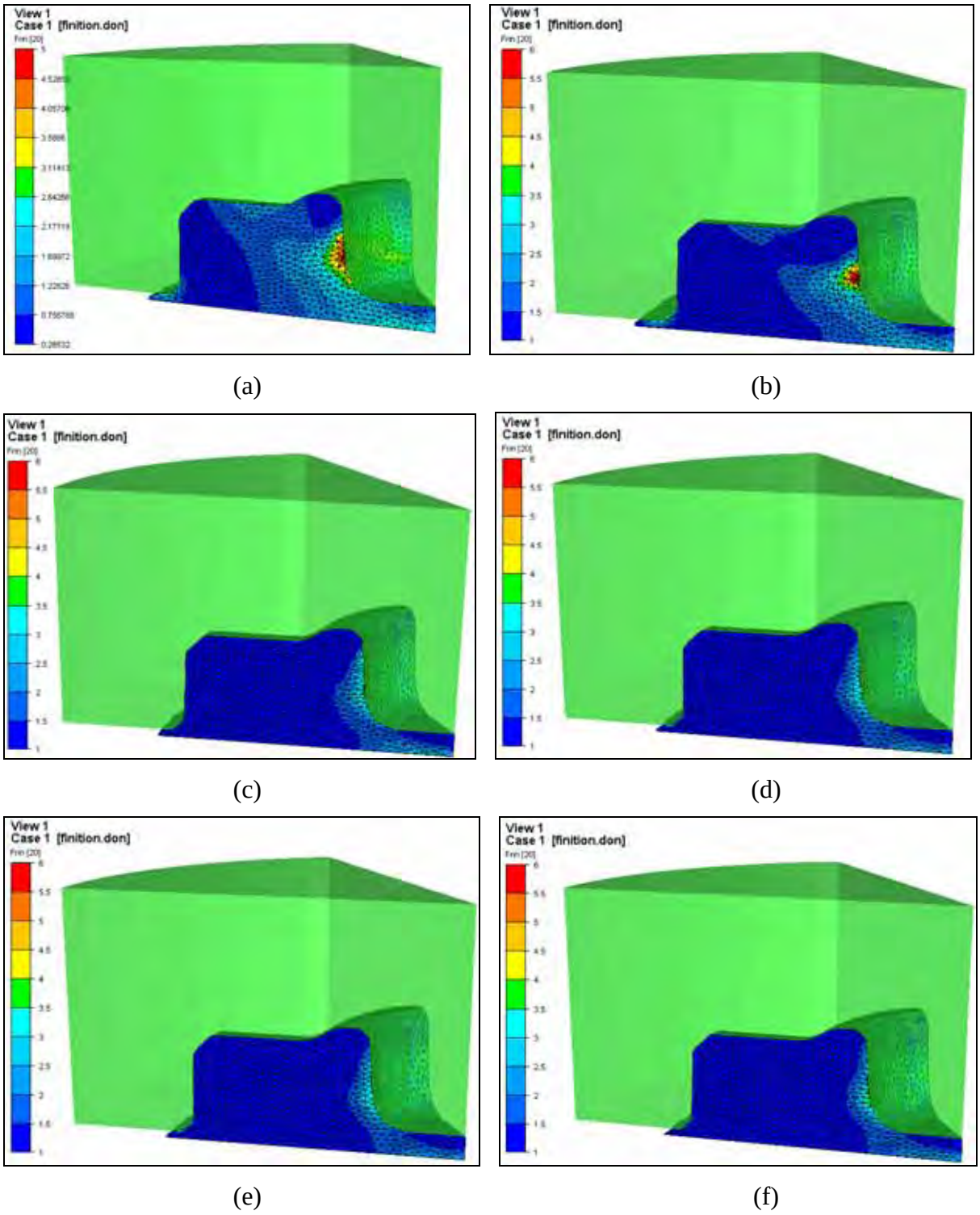



*Figure 5.13.Pièces forgées finales avec différents outils de préforme : (a) de référence; (b) optimisé par BFGS ; (c) SCPIP ; (d) SE-Meta ; (e) AGMGA ; (f) AGMGO*

### V.3.2.3. Vérification

Afin de vérifier les résultats d'optimisation, nous effectuons les calculs avec des maillages plus fins de 18000 nœuds et 75000 éléments environ. Avec ces maillages, il faut environ 14h pour une simulation complète du forgeage d'un triaxe. Les résultats de ces simulations sont présentés sur la *Figure 5.14*.

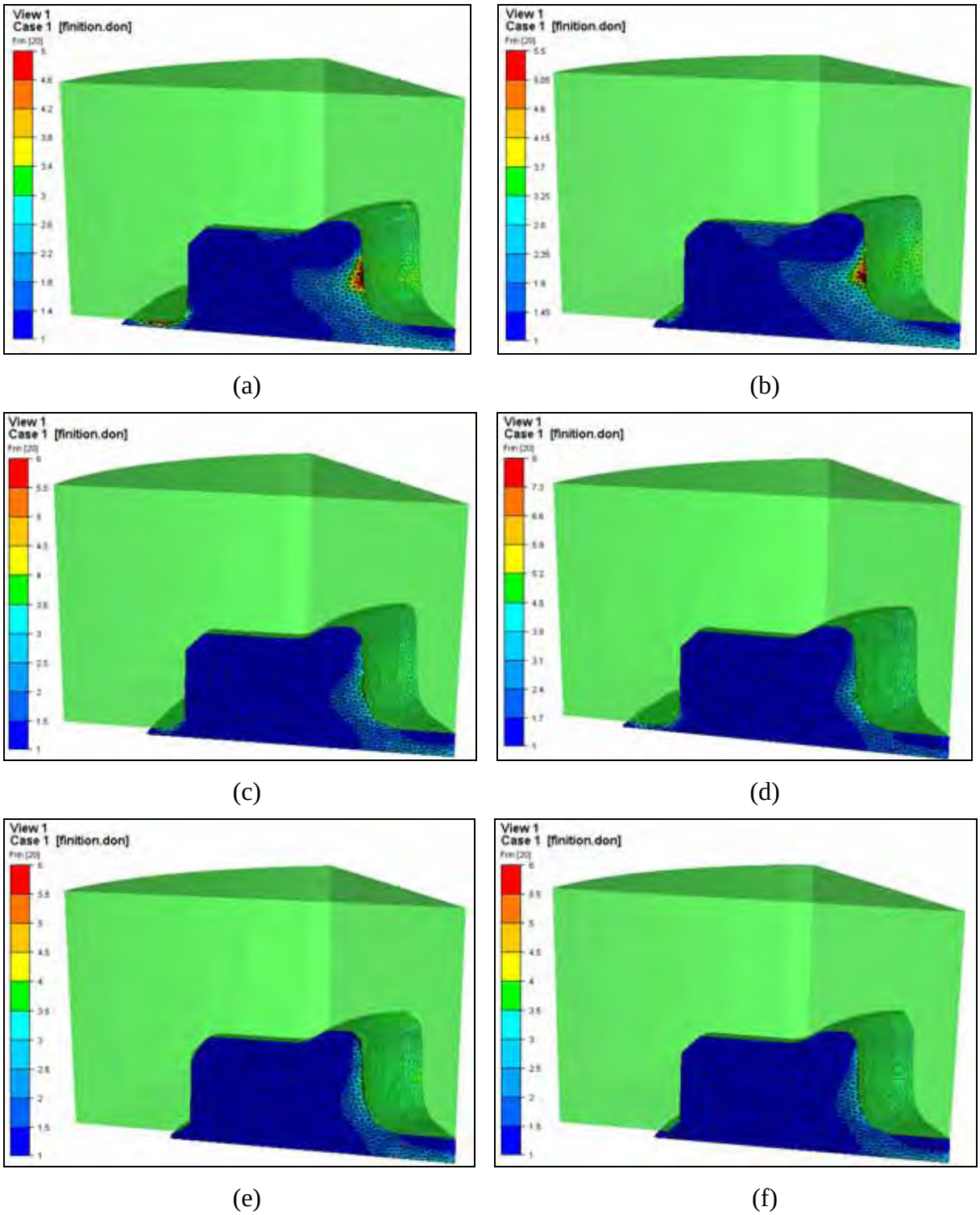



*Figure 5.14. Vérification avec maillage fin : pièces forgées finales avec différentes outils de préforme: (a) de référence; (b) optimisé par BFGS ; (c) SCPIP ; (d) SE-Meta ; (e) AGMGA ; (f) AGMGO*

Ces images confirment que les résultats d'optimisation sont fiables. Avec le maillage fin, les isovaleurs de déformation équivalente sont les mêmes que dans le cas du maillage grossier. Le défaut repli est toujours présent sur la pièce finale obtenue avec BFGS, avec une déformation équivalente bien élevée dans la zone de repli. Par contre, ce défaut n'existe plus sur les autres pièces forgées résultant des autres algorithmes. La déformation équivalente est bien homogène sur ces pièces.

Les deux problèmes d'optimisation de la forme d'outils présentés (à 2 et à 3 paramètres) sont assez compliqués avec de nombreux extrema locaux. Les algorithmes évolutionnaires montrent leur robustesse ainsi que leur performance devant ce problème. Ils proposent des solutions intéressantes, plus globales que celles des algorithmes à direction de descente. La SE-Meta paraît l'algorithme le plus robuste et trouve la meilleure solution.

### V.3.3. Avec 5 paramètres

Pour tester la capacité des AE à résoudre des problèmes d'optimisation avec un plus grand nombre de paramètres, nous étudions le même cas avec cette fois 5 paramètres, tout en conservant la contrainte de nous limiter à 50 calculs de la fonction coût.

Afin d'éviter des instabilités de types "haute fréquences" sur la forme de l'outil de préforme, nous avons utilisé un raffinement hiérarchique de la paramétrisation des formes à partir du cas avec trois paramètres.

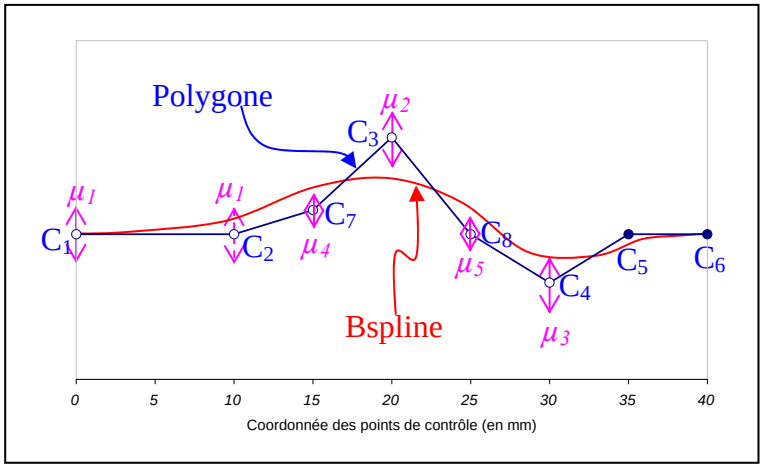



*Figure 5.15. Disposition des paramètres sur le polygone de contrôle de Bspline*

Pour cette paramétrisation, le polygone de contrôle de la Bspline a 8 sommets $C_1$, $C_2$, …, $C_8$ comme présenté sur la *Figure 5.15*. Six parmi eux sont mobiles ($C_1$, $C_2$, $C_3$, $C_4$, $C_7$, $C_8$). Cinq points parmi eux sont actifs et représentés par les paramètres $\mu_1, \mu_2, \ldots, \mu_5$, parmi ceux-ci :

- trois paramètres du problème précédent sont les paramètres principaux ($\mu_1, \mu_2, \mu_3$). Leur intervalle de variation est le même que précédemment.

- deux nouveaux paramètres supplémentaires ($\mu_4, \mu_5$) sont ajoutés pour raffiner la forme de l'outil. Leur intervalle de variation est $-5mm \leq \mu_4, \mu_5 \leq 5mm$. Les ordonnées $\overline{\mu}_4$ et $\overline{\mu}_5$ des points hiérarchiques $C_7$ et $C_8$ sont calculées par :

$$\overline{\mu}_4 = \frac{\mu_1 + \mu_2}{2} + \mu_4 \qquad et \qquad \overline{\mu}_5 = \frac{\mu_2 + \mu_3}{2} + \mu_5$$

Il s'agit des variations des positions des points $C_7$ et $C_8$ autour des positions moyennes données par $\frac{C_2 + C_3}{2}$ et $\frac{C_3 + C_4}{2}$ respectivement.

Le problème d'optimisation peut s'écrire comme suit :

$$\begin{aligned} &\textit{Minimiser} && \Phi_{repli}(\mu_1,\mu_2,\mu_3,\mu_4,\mu_5) \\ &\textit{avec} && -10mm \le \mu_1,\mu_2,\mu_3 \le 20mm \\ & && -5mm \le \mu_4,\mu_5 \le 5mm \end{aligned}$$

Pour ce cas d'optimisation, la SE-Meta permet de trouver une solution légèrement améliorée par rapport au cas à 3 paramètres mais toujours de qualité inférieure à celle obtenue avec 2 paramètres. La meilleure solution est trouvée au 16$^{ième}$ calcul avec une valeur de fonction coût de 8,20. Le défaut de repli n'existe plus sur la pièce forgée finale.

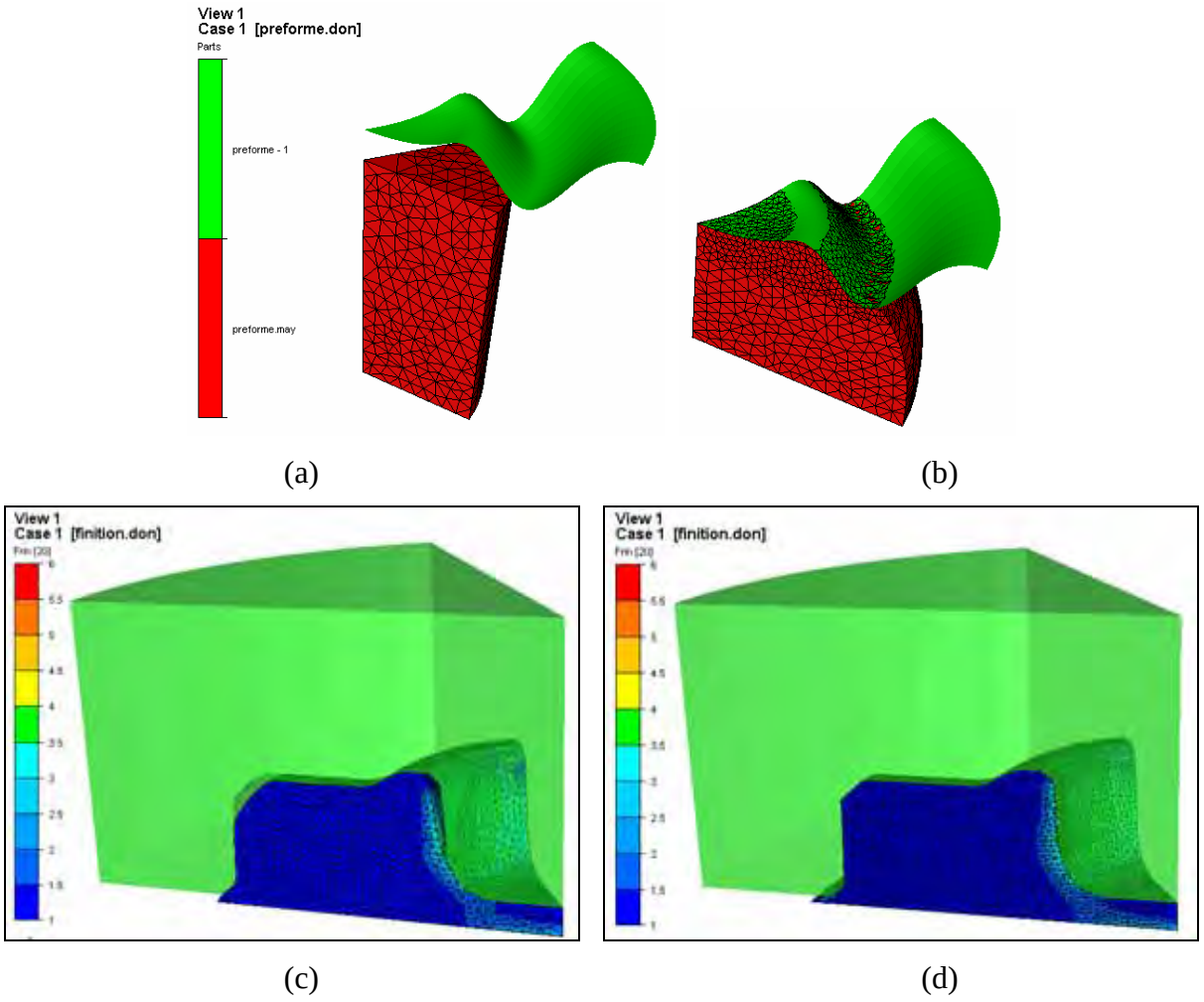



*Figure 5.16. Cas 5 paramètres : outil de préforme optimisé par SE-Meta (a) et (b); Pièce forgée finale avec maillage grossier (c); Vérification avec maillage fin (d)*

La forme de l'outil de préforme optimisée et le produit final sont présentés sur la *Figure 5.16*. En observant l'outil de préforme, nous voyons que sa forme est bien oscillante. Il est très différent des outils optimisés dans le cas avec 2 et 3 paramètres. Sur ce cas, les 3 solutions

Pour ce cas, nous avons aussi fait une vérification avec le maillage fin. Le résultat est présenté sur la *Figure 5.16d*. Nous constatons que le repli est bien enlevé, la déformation est homogène. Le résultat est bien satisfaisant.

# V.4. Conclusions

Dans ce chapitre, nous avons formulé et résolu un problème d'optimisation concret de forgeage. Différentes conclusions peuvent être tirées :

- Devant un problème multimodaux assez raide comme celui-là, l'algorithme BFGS est coincé dans un optimum local. Il est le moins performant et il n'arrive pas à trouver une solution satisfaisante dans les deux cas où il était appliqué.

- Le SCPIP est prometteur car il trouve des solutions satisfaisantes dans les deux cas. Il apporte des améliorations signifiantes (de l'ordre de 10%) sur la fonction coût.

- Les trois algorithmes évolutionnaires proposent des solutions plus intéressantes que celles des algorithmes à direction de descente. La SE-Meta semble la plus performant dans ce cas où elle apporte des améliorations les plus significatives (de 20%) de la fonction coût. Pourtant, les deux algorithmes hybrides sont toujours prometteurs avec des améliorations de l'ordre de 15% et des solutions satisfaisantes (la pièce forgée finale sans repli).

- La SE-Meta a montré sa capacité à résoudre des problèmes d'optimisation de forgeage avec plus de 3 paramètres et un nombre limité de calculs de la fonction coût. Dans tous les cas, elle trouve une préforme qui enlève le repli.

# CONCLUSIONS GENERALES

L'objectif de ce travail de thèse était de construire une chaîne automatique pour résoudre des problèmes d'optimisation de forme en forgeage 3D. Plusieurs aspects ont été abordés pendant cette étude : l'étude des algorithmes d'optimisation, la simulation numérique avec le logiciel FORGE3®, la paramétrisation de forme 3D axisymétrique, l'étude de fonctions coûts d'optimisation, le calcul du gradient des fonctions coûts par la méthode de l'Etat Adjoint dans le cas multi-passes, ainsi que les applications en forgeage.

Différents algorithmes d'optimisation ont été présentés dans le chapitre I conduisant à la proposition d'une approche d'hybridation avec deux variantes dans le chapitre II. Le principe est de faire piloter l'optimisation par un algorithme global, généralement gourmand en évaluations, et de tirer profit de la disponibilité des gradients, pour fournir à ce dernier des approximations très peu coûteuses de la fonction coût. La contribution originale de ce travail concerne le choix des points dits maîtres pour lesquels le coût exact et le gradient sont calculés. On obtient ainsi une méthode d'optimisation globale dont on maîtrise à l'avance le coût en calculs. Cette méthode a été implémentée avec un algorithme génétique, de deux façons. La première, sans mémoire, utilise une approximation discontinue. La seconde, avec mémoire des évaluations exactes précédentes, utilise l'approximation continue de Liszka-Orkisz.

Dans le chapitre III, une nouvelle fonction coût a été proposée. Elle permet donc de mieux détecter les défauts de type "repli" de la surface de la pièce forgée. Le type de paramètres "forme d'outils" est également considéré pour l'optimisation. Par rapport au paramètre "préforme du lopin initial", il apporte deux principaux avantages. Il permet aux utilisateurs d'éviter le traitement de la contrainte volume constant, et d'aborder une gamme plus vaste de problèmes d'optimisation. Une extension du calcul du gradient des fonctions coûts par rapport à ce type de paramètres dans le cadre des procédés de forgeage multi-passes a été présentée. Les principaux travaux de cette extension, qui résident dans le traitement des termes de contact et le transport des variables entre deux opérations de forgeage, ont été réalisés et permet d'obtenir le gradient des fonctions coûts avec une excellente précision (de l'ordre de la précision de la méthode d'éléments finis). Ce calcul est donc fiable.

Les résultats obtenus avec le benchmark présenté dans le chapitre IV nous permettent de tirer différentes conclusions. D'abord, le BFGS est un algorithme local. Il ne trouve que l'optimum le plus proche de son point de départ. Ensuite, les algorithmes évolutionnaires sont robustes puisqu'ils apportent des améliorations signifiantes à la fonction coût en un nombre limité de calculs (50 calculs). Ils proposent ainsi des solutions plus intéressantes que les algorithmes à direction de descentes. Sur cet exemple, les deux algorithmes hybrides fonctionnent parfaitement en proposant des solutions intéressantes très rapidement (jusqu'à deux fois plus rapide que la SE-Meta). Enfin, la conclusion la plus importante est que l'on peut appliquer les algorithmes évolutionnaires à la résolution de problèmes d'optimisation de forgeage 3D.

Le chapitre V renforce les conclusions précédentes. Le BFGS est coincé dans un optimum local lorsqu'il ne peut pas proposer une solution sans défaut de repli. Le SCPIP est aussi

prometteur avec des solutions bien meilleures et résolvant parfaitement le problème d'optimisation. Les algorithmes évolutionnaires sont toujours plus performants avec des solutions satisfaisantes (pièce forgée finale sans repli). Dans ce cas de forgeage, les deux algorithmes hybrides sont moins performants que la SE-Meta.

A l'avenir, il serait intéressant d'améliorer les algorithmes hybrides pour qu'ils soient plus efficaces par exemple en cherchant une nouvelle méthode de disposition des points maîtres ou une approximation plus précise de la fonction coût. Un autre axe de recherche pourrait être l'étude de nouvelles fonctions coûts significatives et surtout la paramétrisation de vraies formes en 3D. De plus, pour traiter une gamme de problèmes industriels plus variés, il serait utile d'étendre le calcul du gradient des fonctions coûts aux lois de comportement et de frottement plus complexes (loi de Norton-Hoff avec écrouissage, loi élasto-viscoplastique, frottement de Tresca, de Coulomb, etc.) en faisant intervenir les variables d'histoire.

# BIBLIOGRAPHIE

[*Aarts et al. 1989*] Aarts E. H. L., Korst J. *Simmulated Annealing and Boltzman Mechanics*, Wiley& Sons, 1989.

[*Aliaga 2000*] Aliaga C. *Simulation numérique par éléments finis en 3D du comportement thermomécanique au cours du traitement thermique des aciers : application à la trempe de pièces forgées ou coulées*. Thèse de Doctorat, ENSMP, CEMEF, 2000.

[*Annicchiarico et al. 1999*] Annicchiarico W., Cerrolaza M. *Finite Elements, genetic algorithms and Bsplines : a combined technique for sape optimidation*. Elsevier - Finite Elements in Analysis and Design 33, P.125–141.1999.

[*Antonio al. 2002*] Antonio C.A.C. et Dourado N. M. *Metal-forming process optimisation by inverse evolutionary search*. Journal of Materials processing Technology 121 P.403-413. 2002.

[*Armijo 1966*] Armijo L. *Minimization of Functions Having Continuous Partial Derivatives*. Pacific. J. Math., 16, pp. 1-3, 1966.

[*Arnold et al. 1984*] Arnold D.N., Brezzi F., Fortin M. *A stable finite element for the Stokes equations*. Calcolo, Vol. 21, pp ; 337-344, 1984.

[*Atmar 1976*] Atmar J.W. *Speculation on the evolution of intelligence and its possible realization in machine form*. Sc.D. Dissertation, New Mexico State University, Las Cruces, NM, 1976.

[*Auger 2004*] Auger A. *Contributions Théoriques et Numériques à l'optimisation continue par Algorithmes Evolutionnaires*. Thèse de doctorat Université Paris 6. 2004

[*Ayad et al. 2005*] Ayad G., Barriere T., Gelin J. C. *Optimization of powder segregation occurring in metal injection molding of stainless steels*. Lavoisier. Volume 8 – n° 1, 2005.

[*Bäck et al. 1993*] Bäck T., Schwefel H.-P. *An overview of evolutionary algorithms for parameter optimization*. Evolutionary Computation 1 (1) (1993) 1-23.

[*Bäck et al. 1993*] Bäck T., Rudolph G., and Schwefel H. P. *Evolutionary programming and evolution strategies: Similarities and differences*. Proceedings of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, San Diego, CA, pp.11-22, 1993.

[*Bäck 1995*] Bäck T. *Evolutionary Algorithms in Theory and Practice.* New-York, Oxford University Press, 1995.

[*Bäck et al. 1997*] Bäck T., Fogel D., and Michalewicz Z. (eds.). *Handbook of Evolutionary Computation*. Oxford University Press. 1997

[*Bahloul et al. 2005*] Bahloul R., Mkaddem A., Dal Santo Ph., Potiron A. *Sheet metal bending optimisation using response surface method, numerical simulation and design of experiments*, Congress of materials and manufacturing engineering and technology COMMENT 2005, May 16-19, Gliwice-Wista – Poland, pp 134.

[*Baker 1987*] Baker J. E. *Reducing bias and inefficiency in the selection algorithm*. In J.J. Grefenstette, editor, Proceedings of the 2nd International Conference on Genetic Algorithms, pp. 14-21. Laurence Erlbaum Associates, 1987.

[*Bălan 1996*] Bălan. T. *Optimisation de forme des outils de forgeage par méthode inverse*. Thèse de Doctorat, ENSMP, CEMEF, 1996.

[*Banzhaf et al. 1998*] Banzhaf W., Nordin P., Keller R. E., Francone F. D. *Genetic Programming: An Introduction*. Morgan Kaufmann, Inc., San Francisco, USA, 1998.

[*Beauchesne et al. 2005*] Beauchesne E., Bekkour T., Delcroix F., Faure J.M., Kayvantash K., Marro M., Safieddine M., Wauquiez C. *Optimisation du procédé de mise en forme par hydroformage*. Rapport Final du projet National de recherche OPTIMAT, 2005.

[*Belytschko et al. 1996*] Belytschko T., Krongauz K., Organ D., Fleming M., Krysl P. *Meshless methods : an overview and recent developments*, Comput. Methods Appl. Mech Engrg, vol. 139, p. 3-47, 1996.

[*Ben Ayed et al. 2005*] Ben Ayed L., Delameziere A., Batoz J.L., Knopf-Lenoir C. *Optimisation des efforts serre-flan en emboutissage pour contrôler la striction et le plissement*, Premier Congrès International Conception et Modélisation des systèmes Mécaniques, CMSM'2005, 23-25 Mars 2005, Hammamet, Tunisie.

[*Bonte et al 2005*a] Bonté M. H. A., van den Boogaard A. H., Huétink J. *Metamodelling techniques for the optimisation of metal forming processes*. In Proceedings of ESAFORM, Cluj-Napoca, Romania, 2005, pp. 155-158.

[*Bonte et al 2005*b] Bonte M. H. A., van den Boogaard A. H., Huétink J. *Solving optimisation problems in metal forming using finite element simulation and metamodelling techniques*. In Proceedings of APOMAT (Morschach, Switzerland, 2005), pp. 242-251.

[*Bonte 2005*] Bonte M. H. A. *A comparison between optimisation algorithms for metal forming processes – With application to forging*. Rapport de stage au CEMEF 2005.

[*Bourdin et al. 1998*] Bourdin J. P., Bonnafé J. P., Delmotte J., Grosjean E., Roelandt J. M. *Age Creep Forming Process Optimization for Aeronautical Structures*. Dans S. Idelsohn et al. (éditeurs), Computational Mechanics. New Trend and Applications. pp. 1-16 Barcelone, Espagne, 1998.

[*Boyère 1999*] Boyère E. *Contribution à la modélisation numérique thermo-mécanique tridimensionnelle du forgeage*. Thèse de doctorat, ENSMP, CEMEF, 1999.

[*Brezzi et al. 1991*] Brezzi F., Fortin M. *Mixed and Hybrid Finite Element Methods*. Springer-Verlarg, New-York Berlin Heidelberg, 1991.

[*Broyden 1970*] Broyden C. G. *The convergence of a class of double-rank minimization algorithms*. J. Inst. of Math. and Its Appl., 6, pp. 222-231, 1970.

[*Burczynski et al. 2001*] Burczynski T., Orantek P. *Hybrid evolutionary algorithms aided by sensitivity information in identification and structrural optimization*. ECCM-2001 June 26-29, Krakow, Poland

[*Burgin 1973*] Burgin G.H. *System identification by quasilinearization and evolutionary programming*, Journal of cybernetics, vol. 3, no. 2, pp. 56-75, 1973.

[*Carroll 1997*] Carroll D.L. *Fortran GA - Genetic Algorithm Driver V1.6.4,* Users Guide, 1997.

[*Castro et al. 2000*] Castro C.F., Costa Sousa L., Conceiçao Antonio C.A., César de Sà J. *A multilevel approach to optimization of bulk forming processes*. Conférence ECCOMAS, Barcelone, 2000.

[*Cescutti 1989*] Cescutti J.P. *Contribution à la simulation numérique du forgeage à chaud*. Thèse de Doctorat, ENSMP, CEMEF, 1989.

[*Chen et al. 1995*] Chen M. F., Maniatty A. M. *An Inverse Technique for the Optimization of Some Forging Processes*. Dans Shen et al. (éditeurs), Simulation of Material Processing: Theory, Methods and Applications, pp. 545-550, Balkema, Rotterdam, 1995.

[*Chen 2001*] Chen S-Y. *An approach for impact structure optimization using the robust genetic algorithm*. Elsevier - Finite Elements in Analysis and Design 37, pp.431- 446. 2001.

[*Chung et al. 1997*] Chung J. S., Hwang H. M. *Application of a genetic algorithm to the optimal design of the die shape in extrusion*. Journal of Materials processing Technology 72 , pp. 69-77,1997.

[*Chung et al. 1998*] Chung S.H., Hwang S.M. *Optimal process design in non-isothermal non-steady metal forming by the finite element method*. International

Journal for Numerical Methods in Engineering, Vol. 42, pp. 1343-1390, 1998.

[*Chung et al. 2003*] Chung S.H., Fourment L., Chenot J.-L., Hwang M. *Adjoint state method for shape sensitivity analysis in non-steady forming applications*. International Journal for Numerical Methods in Engineering, Vol. 57, pp. 1431-1444, 2003.

[*Coupez 1991*] Coupez T. *Grandes transformations et remaillage automatique*. Thèse de Doctorat, ENSMP, CEMEF, 1991.

[*Coupez 1995*] Coupez T. *Stable stabilized finite element for 3D forming calculation*. ENSMP, Sophia-Antipolis, 1995.

[*Charbonneau 2002*] Charbonneau P. *An introduction to genetic algorithms for numerical optimization*. NCAR Technical Note 450+IA (Boulder: National Center for Atmospheric Research), Etats-Unis, 2002.

[*Coupez et al. 1999*] Coupez T., Nouatin A. I. *Optimisation of Forming by using the Simplex Method and Preliminary Results on an Explicit 3D Viscoelastic Solution.* Dans : J. A. Covas (éditeur), *2nd ESAFORM Conference,* pp. 477-480, Guimarães, 1999.

[*Cramer 1985*] Cramer N. *A representation for the adaptive generation of simple sequential programs*. In Grefenstette, J., editor, Proceedings of an International Conference on Genetic Algorithms and the Applications, pages 183-187, Carnegie-Mellon University, Pittsburgh, PA, USA, 1985.

[*De Jong 1975*] De Jong K. *The analysis of the behaviour of class of genetic adaptative systems*. Thèse de doctorat. Department of computer Science, University of Michigan, Ann Arbor , Michigan 1975.

[*De Jong 1980*] De Jong K. *Adaptive System Design: A Genetic Approach*. IEEE Transaction on Systems, Man, and Cybernetics , Vol. SMC-10 , no. 9, Sep 1980, pp. 566-574.

[*De Jong et al. 1991*] De Jong K., Spears W. On the virtues of parameterised uniform crossover. In R. K. Belew and L. B. Booker, editors, Proceedings of the Fourth International Conference on Genetic Algorithms, pages 230-236, San Mateo, July 1991. Morgan Kaufman.

[*De Jong et al. 1995*] De Jong K., Sarma J. *On Decentralizing Selection Algorithms***.** Proceedings of the Sixth International Conference on Genetic Algorithms. pp. 17-23. Morgan Kaufmann. 1995.

[*Dorigo et al. 1996*] Dorigo M., Maniezzo V. et Colorni A. *The ant system : Optimization by a colony of cooperating agents*. Dans IEEE Transactions on Systems, Mans, and Cybernetics, vol. 1, p. 29–41, 1996.

[*Emmerich et al. 2002*] Emmerich M., Giotis A., Özdemir M., Bäck Th. and Giannakoglou K., *Metamodel-assisted evolution strategies*. In J. J. Merelo Guervos et al. (edts.): Parallel Problem Solving from Nature VII, Proc. Inte'I Conf, Granada, September 2002.

[*Fletcher et al. 1963*] Fletcher R., Powell M. J. D. *A Rapidly Convergent Descent Method for Minimization*. Computer Journal, 6, pp. 163-168, 1963.

[*Fletcher et al. 1964*] Fletcher R., Reeves C. M. *Function Minimization By Conjugate Gradient.* Computer Journal, 7, pp. 149-154, 1964.

[*Fletcher 1996*] Fletcher R. *Practical optimisation methods*, Second edition, John Wiley & Sons, Chichester, 1996.

[*Fogel 1964*] Fogel L. J. *On the organization of intellect*, Ph.D. Dissertation, UCLA, 1964.

[*Fogel et al. 1966*] Fogel L. J., Owens A.J., and Walsh M.J., *Artificial Intelligence through Simulated Evolution*, John Wiley, NY, 1966.

[*Fogel et al. 1969*] Fogel L. J. and Burgin G.H. *Competitive goal-seeking through evolutionary programming*, Final Report, Contract AF 19(628), Air Force Cambridge Research Laboratories, 1969.

[*Fogel 1992*] Fogel D. B. *Evolving Artificial Intelligence*, Doctoral Dissertation, University of California, San Diego, CA, 1992.

[*Fourment et al. 1996*] Fourment L., Balan T., Chenot J.-L. *Optimal design for non steady-state metal forming processes- I Shape optimization method- II Application of shape optimization in forging*. International Journal for Numerical Methods in Engineering, Vol. 39 (1), pp. 33-66, 1996.

[*Fourment et al. 2001*] Fourment L., Chung S.H. *Direct and adjoint differentiation methods for shape optimization in non-steady forming applications*. European Conference on Computational Mechanics, Cracow, 2001.

[*Fourment et al. 2003*] Fourment L. and Ward M. "Shape optimization for preform tool design in reverse Superplastic Forming", VII International Conference on Computational Plasticity, COMPLAS, Barcelonna, April 7-10, 2003.

[*Gavrus 1996*] Gavrus A. *Identification automatique des paramètres rhéologiques par analyse inverse*. Thèse de doctorat, ENSMP, 1996.

[*Ghouati et al. 1998*] Ghouati O., Gelin J. C. *Gradient based methods, genetic algorithms and the finite element method for the identification of material parameters.* ip.157-162, Simulation of Materials Processing : Theory, Methods and Applications, Huétink & Baaijens (eds), 1998.

[*Goldberg 1989*] Goldberg D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing company 1989.

[*Grefenstte 1987*] Grefenstette, J. J. (Ed.). *Genetic Algorithms and Their Applications*, Proceedings of the Second International Conference on Genetic Algorithms, Hillsdale, NJ: Lawrence Erlbaum, 1987.

[*Haftka et al. 1992*] Haftka R., GÄurdal Z. *Elements of structural optimization, 3rd edition*. Kluwer academic publishers, Dordrecht, Netherlands, 1992. ISBN 0-7923-1504-9.

[*Holland 1962*] Holland H. J. *Outline for a logical theory of adaptative systems*. Journal of the association of computing machinery, 3, 1962.

[*Holland 1975*] Holland H. J. *Adaptation in natural and artificial system*, Ann Harbor, The University of Michigan Press, 1975.

[*Jensen et al. 1998*] Jensen M. R., Damborg F. F., Nielsen K. B., Danckert J. *Optimization of the Draw-Die Design in Conventionnal Deep-Drawing in Order to Minimise Tool Wear*. J. Mater. Process. Technol., 83, pp. 106-114, 1998.

[*Jin et al. 2001*] Jin Y., Olhofer M., and Sendhoff B., *Managing approximate models in evolutionary aerodynamic design optimization*, in Proceedings of IEEE Congress on Evolutionary Computation, Vol. 1, Seoul, Korea, 2001, pp. 592–599.

[*Jin et al. 2000*] Jin Y., Olhofer M., and Sendhoff B., *On evolutionary optimisation with approximate fitness functions*, in Proceedings of the Genetic and Evolutionary Computation Conference GECCO, Las Vegas, Nevada, 2000, pp. 786–793.

[*Kirkpatrick et al. 1983*] Kirkpatrick S., Gelatt C. D. Jr., Vecchi M. P. *Optimization by simulated annealing.* Science 220, N°4598, pp. 671-680, 1983.

[*Koza 1992*] Koza. J. R. *Genetic Programming : On the Programming of Computers by means of Natural Evolution*. MIT Press, Massachusetts, 1992.

[*Koza 1994*] Koza J. R. *Genetic Programming II : Automatic Discovery of Reusable Programs*. MIT Press, Massachussetts, 1994.

[*Koza et al. 1999*] Koza J. R. et al. *Genetic Programming III : Automatic Synthesis of Analog Circuits*. MIT Press, Massachussetts, 1999.

[*Krige 1951*] Krige D. 1951. *A statical problem to some basic minim valuation problems on the Witwatersrand*. Journal of the Chemical, Metallurgical and Mining Society of the South Africa, 52, 119-139.

[*Kusiak et al. 1989*] Kusiak J., Thompson E. G. *Optimization techniques for Extrusion Die Shape Design*. Dans: E. G. Thompson et al. (éditeurs), *Numiform'89*, pp. 569-574. Balkema: Rotterdam, 1989.

[*Laroussi 2003*] Laroussi M. *Analyse de sensibilité 3D par la méthode de l'Etat Adjoint – Application au forgeage*. Thèse de Doctorat, ENSMP, CEMEF, 2003.

[*Liszka et al. 1980*] Liszka T. and Orkisz J. *The finite difference method at arbitrary irregular grids and its application in applied mechanics*. Comp. and Struct 11: 83-95, 1980.

[*Lophaven et al. 2002*] Lophaven S., Nielsen H., and Sondergaard J. *DACE - A MATLAB Kriging Toolbox*. Technical Report IMM-TR-2002-12, Technical University of Denmark - Department of Informatics and Mathematical Modelling, Lyngby, Denmark, 2002.

[*Lutton 1999*] Lutton E. *Les Algorithmes Génétiques* – Atelier Fractales – Ecole Centrale de Paris 1999.

[*Marie 1997*] Marie S. *Un modèle de parallélisation S.P.M.D. pour la simulation de procédés de mise en forme de matériaux*. Thèse de Doctorat, ENSMP, CEMEF, 1997.

[*Martin et al. 2003*] Martin J., and Simpson T. *A study on the use of Kriging models to approximate deterministic computer models*. In Proceedings of the ASME Design engineering Technical Conferences DETC 2003.

[*Martin et al. 2004*] Martin J., Simpson T. *On the use of Kriging models to approximate deterministic computer models*. In Proceedings of the ASME Design Engineering Technical Conferences DETC 2004.

[*Matheron 1963*] Matheron G. *Principles of Geostatistics*. Economic Geology, 58, 1246-1268. 1963.

[*Maza et al. 1993*] Maza M. de la and Tidor B. *An analysis of selection procedures with particular attention paid to proportional and bolzmann selection*. In Stefanie Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pages 124--131, San Mateo, CA, 1993. Morgan Kaufmann Publishers.

[*McKinnon 1998*] McKinnon K. I. M. *Convergence Of The Nelder-Mead Simplex Method To A Nonstationary Point*. SIAM Journal on Optimization . 1998

[*Metropolis et al. 1953*] Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H. and Teller E. *Equation of State Calculations by Fast Computing Machines*, J. Chem. Phys. 21 (1953) 1087-1092.

[*Minkowycz et al. 1988*] Minkowycz M., Sparrow W. J., Schneider G. E., Pletcher R. H. *Handbook of Heat Transfert*, Wisley&Sons, 1998.

[*Mocellin 1999*] Mocellin K. *Contribution à la simulation numérique tridimensionnelle du forgeage à chaud : étude du contact et calcul multigrille*. Thèse de Doctorat, ENSMP, CEMEF, 1999.

[*Moré et al. 1994*] Moré J. J., Thuente D. J., *Line search algorithms with guaranteed sufficient decrease*, ACM Transactions on mathematical software, 20, no 3 (1994) 286-307.

[*Mori et al. 1996*] Mori K., Yamamoto M., Osakada K. *Determinaton of hammering sequence in incremental sheet metal forming using a genetic algorithm*. Journal of Materials processing Technology 60 (1996) P.463-468.

[*Morris 1982*] Morris A. J. *Foundations of Structural Optimization: A unified approach*. Wiley Series in Numerical Methods for Engineering. Wiley&Sons, Chichester, 1982.

[*Myers et al. 2002*] Myers R. and Montgomery D. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments, 2nd ed*. John Wiley and Sons, Inc., New York, USA, 2002. ISBN 0-471-41255-4.

[*Naceur et al. 2004*] Naceur H., Ben Elechi S., Knopf-Lenoir C., Batoz J.L. *Response surface methodology for the design of sheet metal forming parameters to control springback effects using the inverse approach*, in Gosh, S. et al., Eds., "Materials Processing and Design: Modeling, Simulation and Applications", NUMIFORM'04, OSU, Columbus, Ohio, USA, June 2004, pp. 1991-1996

[*Nakamashi et al. 1998*] Nakamashi E., Honda T. *Optimum Design of Sheet Forming Process by Using Finite Element and Discretized Optimization Method*. *Int. J. of Forming Proc.*, 1(2), pp. 163-185, 1998.

[*Nayroles et al. 1991*] Nayroles B., Touzot G., Villon P., *La Méthode des Éléments Diffus*, Compte rendu à l'Academie de Sciences, 313, série II, p. 133-138, Paris, France, 1991.

[*Nelder et al. 1965*] Nelder J. A., Mead R. *A Simplex Method for Function Minimization*. Computer Journal, 7, pp. 308-313, 1965.

[*Nouatin 2000*] Nouatin O. H. *Méthode et analyse de simulation numérique d'écoulements 3D des polymères fondus- Identification de paramètres*

*rhéologiques viscoélastiques par analyse inverse*. Thèse de doctorat de l'Ecole de Mines de Paris, Cemef. 2000.

[*Noiret et al. 1996*] Noiret C., Lauro F., D. Lochegnies, Oudin J. *Optimisation by Inverse Method : Application to the Forming of Hollow Glass Items*. Dans *: J. L. Chenot et al. (éditeur), 1st ESAFORM Conference,* pp. 437-440, Sophia-Antipolis, 1998.

[*Ohata et al. 1998*] Ohata T., Nakamura Y., Katayama T., Nakamachi E., Omori N. *Improvement of Optimum Process Design System by Numerical Simulation*. *J. Mater. Process. Technol.*, 80-81, pp. 635-641, 1998.

[*Oduguwa et al. 2005*] Oduguwa V., Tiwari A., Roy R. *Evolutionary computing in manufacturing industry: an overview of recent applications*. Applied Soft Computing 5. p281-299. 2005.

[*Oulladji et al. 2003*] Oulladji L., Janka A., Désidéri J.A., Dervieux A. *Optimisation aérodynamique par algorithmes génétiques hybrides : application à la réduction d'un critère de bang sonique*. Rapport de recherche de l'INRIA – Juillet 2003.

[*Papadrakakis et al. 2001*] Papadrakakis M., Lagaros N. D. et Fragakis Y. *Parallel computational strategies for structural optimization*. ECCM-2001 June 26-29, Crascow, Poland

[*Perchat 2000*] Perchat E. *Mini-élément et factorisations incomplètes pour la parallélisation d'un solveur de Stokes 2D. Application au forgeage*. Thèse de Doctorat, ENSMP, CEMEF, 2000.

[*Polak et al. 1969*] Polak E., Ribière G. *Note sur la converge de méthode de directions conjuguées*, Rbue française d'informatique et de recherche opérationnelle, 3 (1969) 35-43.

[*Powell 1975*] Powell M. J. D. *Restart Procedure for the Conjugate-Gradient Method*. Math. Prog., 2, pp. 241-254, 1975.

[*Press et al. 1992*] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P. *Numerical recipes in Fortran*. The Art of Scientific Computing. Cambridge University Press, 2ième édition, 1992.

[*Rappaz et al. 1998*] Rappaz M., Bellet M., Deville M. *Modélisation Numérique en Science et Génie des Matériaux*. Vol. 10, Presses Polytechniques et Universitaires Romandes, Lausanne, 1998.

[*Rechenberg 1965*] Rechenberg, I. *Cybernetic solution path of an experimental problem*. Technical Report Library Translation number 1122, Royal Aircraft Establishment, Farnborough, Hants., UK. (1965).

[*Rechenberg 1973*] Rechenberg I., *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

[*Rechenberg 1994*] Rechenberg. *Evolution strategy*. In J. Zurada, R. Marks, and C. Robinson, editors, Computational Intelligence - Imitating Life, pages 147-159. IEEE Press, 1994.

[*Roux 2001*] Roux O. *La mémoire dans les algorithmes à colonie de fourmis : applications à l'optimisation et à la programmation automatique*. Thèse de doctorat. Université du Littoral Côte d'Opale. 2001.

[*Sado et al. 1991*] Sado G., Sado M.C. *Les plans d'Expériences : de l'expérimentation à l'Assurance Qualité*, Afnor Technique, ISBN 2-124-50311-1, 1991.

[*Santner et al. 2003*] Santner T., Williams B., and Notz W. *The Design and Analysis of Computer Experiments*. Springer-Verlag, New York, USA, 2003. ISBN 0-387-95420-1.

[*Schittkowski 1980*] Schittkowski K. *Nonlinear Programming Codes. Informations, Test, Performance*. Lecture Notes in Economic and Mathematical System. Springer-Verlag, Berlin, Heidelberg, New-York, 1980.

[*Schwefel 1981*] Schwefel H.-P. *Numerical Optimization of Computer Models*. JohnWiley & Sons, New-York, 1981. 1995 – 2nd edition.

[*Séfrioui 1998*] Séfrioui M. *Algorithmes Evolutionnaires pour le calcul scientifique. Application à l'électromagnétisme et à la mécanique des fluides numériques*. Thèse de doctorat de l'Université Paris 6, France, 1998.

[*Soyris 1990*] Soyris N. *Modélisation tridimensionnelle du couplage thermique en forgeage à chaud*. Thèse de Doctorat, CEMEF, 1990.

[*Srikanth et al. 2000*] Srikanth A., Zabaras N. *Shape Optimization and preform design in metal forming process*. Computational Methods in Applied Mechanics and Engineering, Vol. 190, pp. 1859-1901, 2000.

[*Stupkiewicz et al. 2002*] Stupkiewicz S., Korelc J., Dutko M., Rodič T.. *Shape sensitivity analysis of large deformation frictional contact problems*. Computer Methods in Applied Mechanics and Engineering, Vol. 191, No. 33, pp. 3555-3581, 2002.

[*Stupkiewicz et al. 2003*] Stupkiewicz S. *Augmented Lagrangian formulation and sensitivity analysis of contact problems*. Complas VII: 7th International Conference on Computational Plasticity, E. Onate and D.R.J. Owen (editeurs), CIMNE, Barcelone, 2003.

[*Svanberg 1987*] Svanberg K. The Method of Moving Asymptotes – a new method for structural optimization. Int. J. Num. Meth. Eng., 24:359-373, 1987.

[*Syswerda 1989*] Syswerda G. *Uniform crossover in genetic algorithms*. In Proceedings of the Third International Conference on Genetic Algorithms, pages 2-8. Morgan Kauffman. 1989.

[*Traoré 2001*] Traoré K. *Simulation thermomécanique du laminage circulaire. Développement d'une formulation quasi-Eulérienne tridimensionnelle sur une architecture parallèle*. Thèse de Doctorat, ENSMP, CEMEF, 2001.

[*Trosset et al. 1997*] Trosset M. W., Torczon V. *Numerical Optimization Using Computer Experiments*, Report No. TR97-02, Dept. of Comp. and App. Math., Rice University, Houston, TX, 1997.

[*Vallée et al. 2001*] Vallée T., Yildizoglu M. *Présentation des algorithmes génétique et de leurs applications en économie*. Working paper of E3i 2001.

[*Vieilledent et al. 1998*] Vieilledent D., Fourment L., Lasne P. *Toward an Automatic Preform Design Software for Axisymmetrical Forging*. Dans : *Colloque IDMME98*, pp. 483-490, Compiègne, 1998.

[*Vieilledent 1999*] Vieilledent D. *Optimisation des outils en forgeage à chaud par simulation éléments finis et méthode inverse. Applications à des problèmes industriels*. Thèse de Doctorat, CEMEF, 1999.

[*Wackernagel 1998*] Wackernagel. *Multivariate Geostatistics*, Second Edition, Springer-Verlag Heidelberg, p. 145. 1998

[*Wolfe 1971*] Wolfe P. *Convergence Condition for the Ascent Method*. SIAM Review, 13, pp.185-188, 1971.

[*Zabaras et al. 2000*] Zabaras N., Bao Y., Srikanth A., Frazier W.G. *A continuum Lagrangian sensitivity analysis for metal forming processes with applications to die design problems*. International Journal for Numerical Methods in Engineering, Vol. 48, pp. 679-720, 2000.

[*Zabaras et al. 1995*] Zabaras N., Kang S. *Control of the freezing Interface Motion in Two-Dimensional Solidification Processses using the Adjoint Method*. Int. J.Num. Meth. Engng., 38, pp. 63-80, 1995.

[*Zhao et al. 1997*] Zhao G., Wright E., Grandhi R. V. *Preform Die Shape Design in Metal Forming using an Optimization Method*. Int. J. Num. Meth. Engng., 40, pp. 1213-1230, 1997.

[*Zillober 2001*] Zillober C. *A combined convex approximation – interior point approach for large scale nonlinear programming*. Optimization and Engineering, 2(1):51-73, 2001

[*Zillober 2002*] Zillober C. *SCPIP - an efficient software tool for the solution of structural optimization problems*. Structural and Multidisciplinary Optimization, 24(5):362-371, 2002.

## Résumé

Ce travail de thèse a pour but l'optimisation de forme en forgeage 3D. Les problèmes à résoudre consistent à chercher la forme optimale du lopin initial ou des outils de préforme afin de minimiser une fonction coût Φ qui représente la demande des industriels. Ce sont souvent des problèmes multi optima dont le temps nécessaire pour une évaluation de la fonction coût est très élevé (de l'ordre de la journée). L'objectif de cette thèse est de construire un module d'optimisation automatique qui permet de localiser l'extremum global à un coût raisonnable (<50 calculs de la fonction coût par optimisation).

La simulation du procédé est effectuée avec le logiciel éléments finis FORGE3®. Les formes axisymétrique des pièces initiales ou des outils de préforme dans le cadre du forgeage multi-passes sont paramétrées utilisant des polygones quadratiques ou des courbes Bsplines. Différentes fonctions coûts sont considérées comme l'énergie totale de forgeage ou le défaut de repli. Le gradient de ces fonctions coûts est obtenu par la méthode de l'Etat Adjoint combinée avec la méthode de différentiation semi-analytique. Dans ce travail, afin d'aborder une gamme de forgeage plus vaste, ce calcul du gradient a été étendu au type de paramètre de "forme des outils de préformes" dans le cadre du forgeage multi passes.

Différents algorithmes d'optimisation ont été étudiés : un algorithme BFGS standard, un algorithme de type asymptotes mobiles, une stratégie d'évolution couplée avec une surface de réponse basée sur le Krigeage et deux nouveaux algorithmes hybrides – résultats d'une nouvelle approche hybride proposée dans le cadre de ce travail. Cette approche consiste à coupler un algorithme génétique avec une méthode de surface de réponse pour largement réduire le nombre d'évaluations de fonction coût. Tous les algorithmes étudiés sont comparés à deux problèmes caractéristiques de forgeage 3D, respectivement l'optimisation de la géométrie de la préforme et celle des outils de préforme. Les résultats obtenus montrent la faisabilité de l'optimisation de forme en forgeage 3D, c'est-à-dire qu'on obtient des résultats satisfaisant en moins de 50 simulations 3D.



*Mots clés : Algorithmes d'optimisation, Optimisation de forme, Forgeage, Etat Adjoint, Calcul du gradient, Eléments Finis, Mécanique des milieux continus, Modélisation numérique, Analyse numérique.*


**********


## Abstract

This study focuses on shape optimization in 3D forging process. The problems to be solved consist of searching the optimal shape of the initial work piece or of the preform tool in order to minimize an objective function F which represents the demand of the clients. These are often multi optima problems in which the necessary time for a cost function evaluation is used to be very long (about a day or more). This work aims at construction an optimization module that permits to localize the global optimum within a reasonable cost (less than 50 calculations of cost function per optimization).

The process simulation is carried out using FORGE3® finite element software. The axisymmetric initial shape of the work piece or die is parameterized using quadratic segments or Bspline curves. Several objective functions are considered, like the forging energy, the forging force or a surface defect criterion. The gradient of these objective functions is obtained by the adjoint-state method and semi-analytical differentiation. In this work, this gradient calculation has been extended to another type of parameter "shape of tool preform".

Different optimization algorithms are tested for 3D applications: a standard BFGS algorithm, a moving asymptote algorithm, an evolution strategie algorithm enhanced with a response surface algorithm based on Kriging and finally two new hybrid evolutionary algorithms – results of a new hybrid approach proposed during this work. This approach consists of coupling a genetic algorithm to a response surface method that uses gradient information to dramatically reduce the number of problem simulations. All studied algorithms are compared upon two 3D industrial tests, using rather coarse meshes. They make it possible to improve the initial design and to decrease the total forming energy and/or a surface defect criterion. They allow also solving an actual industrial design problem, the two steps forging of a turbo-reactor component. Numerical results show the feasibility of such approaches, i.e. the achieving of satisfactory solutions within a limited number of 3D simulations, less than fifty.



*Keywords: Optimisation algorithm, Shape optimization, Forging process, Adjoint State, Sensitivity Analysis, Gradient Calculation, Finite Element Method, Continuum mechanical, Numerical Simulation, Numerical Analysis.*