Stratégies avancées d'optimisation et analyse inverse basées sur l'utilisation de la PGD

Chady Ghnatios, Francisco Chinesta, Adrien Leygue, Pierre Villon, Piotr Breitkopf, Arnaud Poitou

CSMA 2011
10e Colloque National en Calcul des Structures
9-13 Mai 2011, Presqu'île de Giens (Var)

Strategies avancées d'optimisation et analyse inverse basées sur l'utilisation de la PGD.

C. Ghnatios[1], F. Chinesta[1], A. Leygue[2],
P. Villon[3], P. Breitkopf[3], A. Poitou[2]

[1] *EADS Corporate Foundation International Chair-Ecole Centrale de Nantes, France, {Chady.Ghnatios,Francisco.Chinesta}@ec-nantes.fr*
[2] *Ecole Centrale de Nantes, France, {Adrien.Leygue,Arnaud.Poitou}@ec-nantes.fr*
[3] *Université de Technologie de Compiègne, France, {Pierre.Villon,Piotr.Breitkopf}@utc.fr*

Résumé — Dans le domaine de l'optimisation des procédés, la nature implicite de la fonction coût par rapport aux paramètres d'optimisation implique des évaluations successivent de la variable à otpimiser, ce qui génère des calculs prohibitifs. Dans ce travail, on abordera une nouvelle méthode qui consiste à tranformer le problème en explicite, ceci en calculant off-line la solution du modèle paramétré où toutes les sources de variabilités ont été introduites comme des extra-coordonnées, et ensuite réaliser le calcul inverse pour explorer la solution.

1 Introduction

Dans un problème d'optimisation on souhaite calculer un certain nombre de paramètres pour minimiser une certaine fonction coût qui est classiquement évaluée en faisant un certain choix de paramètres. Si cette dernière n'est pas admissible, on corrige le choix des paramètres, avec une stratégie adéquate, et on recalcule la solution du problème. Ainsi, pour chaque choix de paramètres testés il faut résoudre le problème pour ensuite évaluer la fonction coût. Optimisation et discrétisation vont ensemble. Une possibilité intéressante consiste à introduire tous les paramètres du modèle comme extra-coordonnées dans le modèle qui devient ainsi multi-dimensionnel, mais comme nous venons de voir cela n'entraine aucune difficulté majeure quand on procède à sa résolution en utilisant la PGD [6].

La PGD ou « Proper Generalized Decomposition » est une nouvelle méthode de calcul numérique qui consiste à décrire les variables du modèle en forme séparée [6]. Ainsi la complexité résultante évolue linéairement avec le nombre de dimensions du modèle [5], au contraire des méthodes de discrétisation basées sur la construction d'un maillage de l'espace où la difficulté évolue exponentiellement avec le nombre de dimensions du problème. Avec la PGD, si on a un problème 3D, par exemple, on pourra séparer la solution comme une suite de solutions de problèmes de dimension plus réduite ($3\times$1D ou 1D+2D par exemple). Pour cela, la PGD combine représentation séparée et algorithme de construction des fonctions intervenant dans la forme séparée de la solution.

Ainsi, une fois le problème direct résolu, nous aurons accès à la solution en tout point, tout instant et pour tout choix possible des paramètres. Il faut signaler que l'obtention des sensibilités, dérivées de la fonction coût par rapport aux paramètres est explicite. Maintenant, l'optimisation ne nécessite aucune nouvelle résolution puisque la solution est connue pour tout choix des paramètres. Le problème passe donc à un problème explicite. Nous pourrions imaginer une résolution off-line et une optimisation on-line. Plusieurs méthodes d'optimisation et de descente dans la direction du gradient ont été testées. Nous abordons ici l'optimisation par l'utilisation des moindres carrés mobiles, et par la méthode de descente de Lebenberg-Marquardt. On montrera aussi la possibilité de réaliser le calcul inverse et la reconfiguration du système en temps réel.

2 Calcul off-line

Dans cette partie, on introduit les idées générales en relation avec le processus d'optimisation à travers l'analyse d'un modèle thermique simple. Malgré l'apparente simplicité de cet exemple, la même stratégie pourra être appliquée pour traîter des scénarios beaucoup plus complexes. Considérons le four

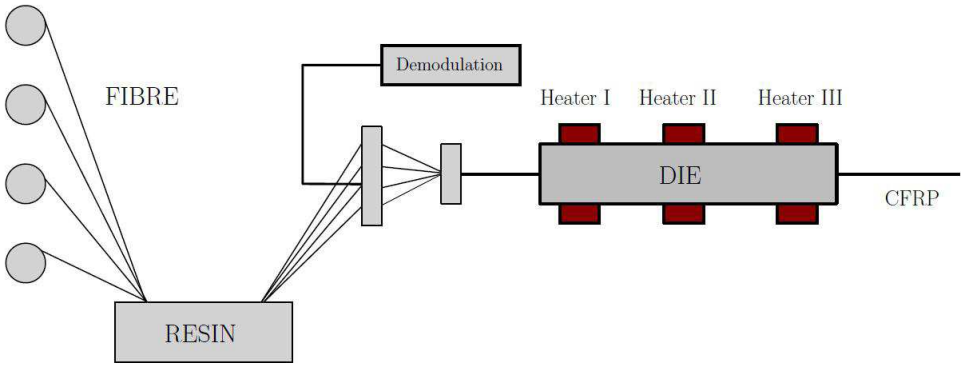

FIGURE 1 – Illustration du procédé de pultrusion

à trois étages d'échauffement de la figure (1). Ce dernier illustre un shémas simplifié de la pultrusion, un procédé de mise en oeuvre en continu de tubes et profilés en plastique renforcé. L'optimisation du choix des températures des étages d'échauffements constitue une tâche principale dans l'optimisation de ce procédé. En plus, pour des années les ingénieurs reposaient sur l'expérience et le savoir faire pour traiter ce problème [7]. Par contre, avec l'augmentation de la demande sur ce genre de procédé [1], il va falloir trouver des méthodes plus pertinentes pour réalier l'optimisation efficace de la pultrusion.

2.1 Introduction à la PGD

Le champ de température stationnaire $u(\mathbf{x})$ est donné en tout point du four $\mathbf{x} = (x,y) \in \Omega \subset \mathcal{R}^2$ par l'équation 2D de la chaleur, avec les termes de convection - diffusion et un éventuel terme source Q. Le champ de vitesse est unidirectionnel partout $\mathbf{v}^T = (v,0)$, et donc l'équation se réduit à :

$$v\frac{\partial u}{\partial x} - k\Delta u - Q = 0 \tag{1}$$

avec k la conductivité du matériaux utilisé.Pour résoudre l'équation (1), nous utiliserons des conditions aux limites variables. En effet les températures du four, ainsi que la vitesse de déplacement de la résine seront introduites comme coordonnées supplémentaires du problème. Ainsi, les conditions aux limites s'écrivent :

$$\begin{cases} u(0,y,\theta_1,\theta_2,\theta_3,v) = u_0 \\ u(x,y,\theta_1,\theta_2,\theta_3,v) = \theta_1 \ pour \ x \in [\varepsilon; L_1 - \varepsilon] \ et \ (y = 0 \ ou \ y = h) \\ u(x,y,\theta_1,\theta_2,\theta_3,v) = \theta_2 \ pour \ x \in [L_1 + \varepsilon; L_2 - \varepsilon] \ et \ (y = 0 \ ou \ y = h) \\ u(x,y,\theta_1,\theta_2,\theta_3,v) = \theta_3 \ pour \ x \in [L_2 + \varepsilon; L_3 - \varepsilon] \ et \ (y = 0 \ ou \ y = h) \\ \nabla T(L_3,y,\theta_1,\theta_2,\theta_3,v) = 0 \end{cases} \tag{2}$$

avec L_1, L_2 et L_3 les distances entre l'entrée du four et la sortie des étages de réchauffement 1, 2 et 3 respectivement, $\theta_1, \theta_2 et \theta_3$ les températures respectives de cees derniers. Pour la résolution, la PGD capable de surmonter la malédiction de multi dimenstionnalité [2] est utilisée puisqu'elle combine la représentation séparée à une technique de résolution par point fixe. Elle porte quelques ressemblances avec la méthode LATIN (représentation séparée de la solution par rapport à chaque variable [10]). On cherche alors la solution sous la forme :

$$u(x,y,\theta_1,\theta_2,\theta_3,v) = \sum_{i=1}^{i=N} F_i(x,y) \times \Theta_{1i}(\theta_1) \times \Theta_{2i}(\theta_2) \times \Theta_{3i}(\theta_3) \times V_i(v) \tag{3}$$

Pour lancer l'algorithme de résolution, on commence par supposer u connue jusqu'à $i = n$ et on cherche à enrichir la solution par le terme $n+1$. Dû à l'apparition des coordonnées comme produits de fonctions dans la solution, le problème devient non linéaire. La méthode de résolution repose sur un algorithme itératif à point fixe [6][3][2]. La résolution s'achève au bout de 2 minutes seulement sur un simple ordinateur portable utilisant MATLAB. Par comparaison avec la méthode des éléments finis et

pour une combinaison donnée de conditions aux limites, une erreur par la $||.||_\infty$ de moins de 10^{-5} est obtenue.

2.2 Optimisation utilisant les moindres carrés mobiles

Dans cette section on considère la solution u déjà calculée sous la forme donnée dans l'équation (3). L'objectif est de déterminer les paramètres du modèle θ_1, θ_2 et θ_3 en fixant v à une valeur constante. Pour ceci, on commence par définir une fonction coût. D'autre part, dans ce travail le seul intérêt est de mettre en évidence l'efficacité de la méthode d'optimisation proposée. On ne cherchera pas donc à déterminer une fonction coût à signification physique. Supposons la fonction à minimiser est la différence entre le profil de température de sortie $x = L$, $u(x = L, y, \theta_1, \theta_2, \theta_3)$ et un profil donné $\overline{u}(y)$:

$$\mathcal{C}(\theta_1, \theta_2, \theta_3) = \frac{1}{2} \int_0^h \left(u(x = L, y; \theta_1; \theta_2, \theta_3) - \overline{u}(y)\right)^2 dy \tag{4}$$

Le domaine des paramètres est défini par : $I = I_1 \times I_2 \times I_3$. On notera donc un point dans l'espace des paramètres par Γ. On agit d'abord en considérant un point arbitraire dans I. On calcule ensuite le gradient et la Hessienne de la fonction coût pour appliquer une méthode de descente. On réalise donc d'abord une approximation quadratique de la fonction autour du point arbitraire, puis on utilise l'algorithme de descente de Newton.

2.2.1 Les moindres carrées mobiles centrés en quelques lignes

On cherche l'approximation d'une fonction générique $u(\mathbf{x})$ en un point $\mathbf{x}$, $u^h(\mathbf{x})$, en se basant sur le valeurs nodales u_i de $u(\mathbf{x})$ au voisinage de $\mathbf{x}$. On se limite à n points $\mathbf{x}_i$. $u^h(\mathbf{x})$ s'écrit alors sous la forme :

$$\mathbf{u}^h(\mathbf{x}) = \mathbf{q}(\mathbf{x}) \cdot \alpha(\mathbf{x}) \tag{5}$$

avec $\mathbf{p}(\mathbf{x})$ une base polynomiale, par exemple : $\mathbf{q}^T(\mathbf{x}) = [1, x_1 - x, x_2 - x, (x_1 - x) \cdot (x_2 - x)]$ pour une approximation linéaire et $\mathbf{q}^T(\mathbf{x}) = [1, x_1 - x, x_2 - x, (x_1 - x) \cdot (x_2 - x), (x_1 - x)^2, (x_2 - x)^2]$ pour une approximation quadratique. Alors que $\alpha(\mathbf{x})$ est un vecteur de coefficients inconnus. Les coefficients $\alpha(\mathbf{x})$ sont définis comme étant les minimums de $J(\alpha)$:

$$J(\alpha) = \frac{1}{2} \sum_{i=1}^{n} w_i(\mathbf{x}) \left(\mathbf{q}(\mathbf{x}_i - \mathbf{x}) \cdot \alpha(\mathbf{x}) - u_i\right)^2 \tag{6}$$

On peut démontrer ([8][9]) que $\alpha(\mathbf{x})$ s'écrit sous la forme de :

$$\alpha(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{u} \tag{7}$$

avec :

$$A_{jk}(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) \cdot q_j(\mathbf{x}_i - \mathbf{x}) \cdot q_k(\mathbf{x}_i - \mathbf{x}) \tag{8}$$

$$B_{ij}(\mathbf{x}) = w_i(\mathbf{x}) \cdot q_j(\mathbf{x}_i - \mathbf{x}) \tag{9}$$

Ayant $\alpha(\mathbf{x})$, et en nottant que [8][2] :

$$\begin{cases} \alpha_1 = u^h(\mathbf{x}) \\ \alpha_2 = \frac{\delta u^h(\mathbf{x})}{\delta x_1} \\ \alpha_3 = \frac{\delta u^h(\mathbf{x})}{\delta x_2} \\ \alpha_4 = \frac{\delta^2 u^h(\mathbf{x})}{\delta x_1 \delta x_2} \\ \alpha_5 = \frac{\delta^2 u^h(\mathbf{x})}{\delta x_1^2} \\ \alpha_6 = \frac{\delta^2 u^h(\mathbf{x})}{\delta x_2^2} \end{cases} \tag{10}$$

Alors les MLS à base centrée donnent accès direct aux différentes dérivées, qui peuvent être utilisées dans un algorithme d'optimisation pour former les gradients et l'Hessiennes.

2.2.2 Solution de notre exemple

Pour illustrer l'exemple déjà décrit par la fonction coût (4), on cherche un profil de température de sortie $\bar{u}(y)$=200°C. La procédure d'optimisation donne le profil décrit par la figure (2). Le champ obtenu pour les valeurs optimales est donné par la figure (3).

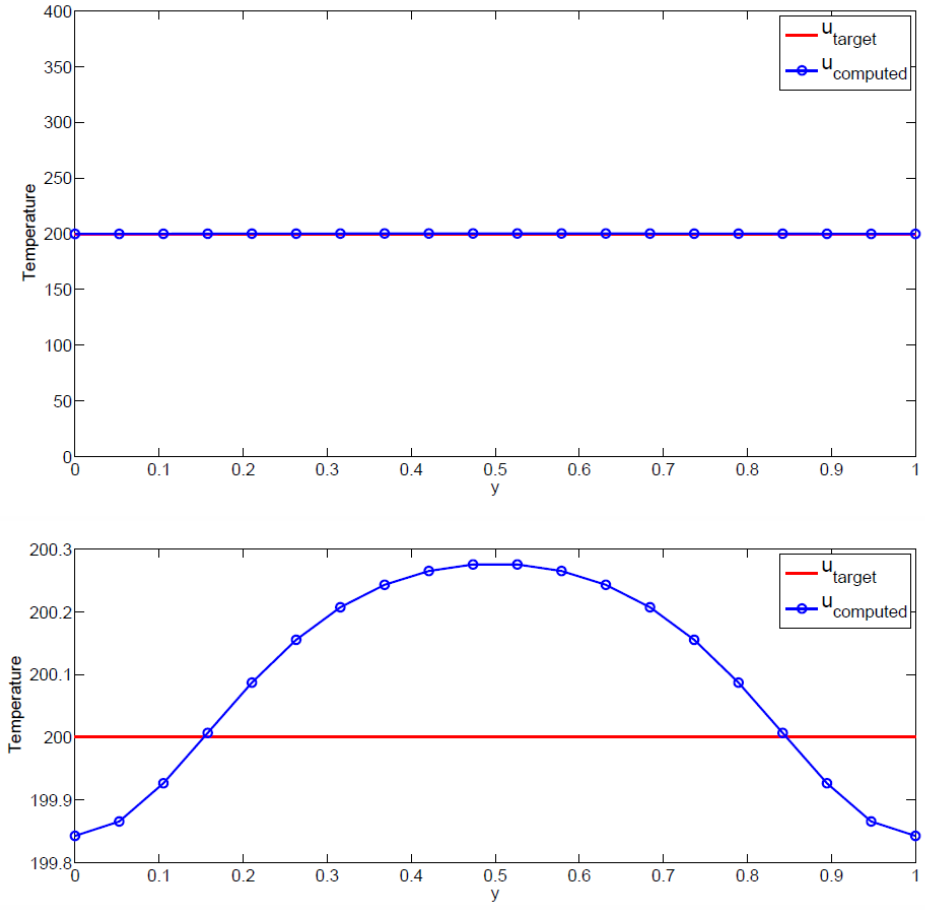

FIGURE 2 – Le profil de température à la sortie du four : en haut pour l'échelle réelle de température et en bas un zoom pour afficher l'erreur d'optimisation

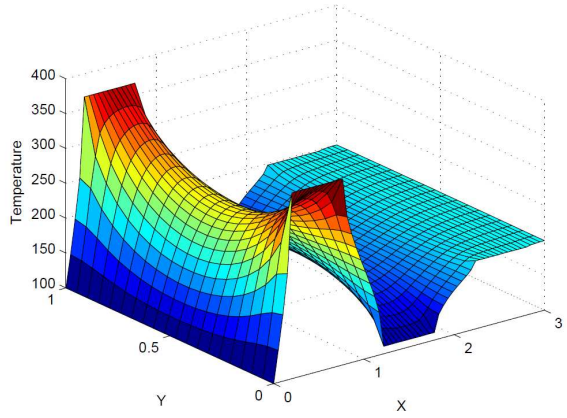

FIGURE 3 – Distribution optimale de la température dans les fours

La valeur de la fonction coût (4) pour la solution affiché dans la figure (3) est 0.0803. Cette solution donne une erreur par la norme L_2 sur le profil de sortie de 1.4×10^{-3}.

3 Calcul on-line

Dans cette section on utilise une méthode de calcul inverse différente pour pouvoir réaliser le calcul inverse et la reconfiguration en temps réel. La méthode des moindres carrés mobiles, bien qu'elle soit efficace, est lente et non adaptée au contrôle en temps réel. La méthode de Levenberg-Marquardt est fiable et conçue pour les moindres carrés non-linéaires [4]. Elle est tout de même facile à intégrer sur des plates-formes mobiles. Dans cette partie, pour la simplification des exemples, on utilisera un four à deux étages de chauffage seulement. Le système est illustré dans la figure (4).

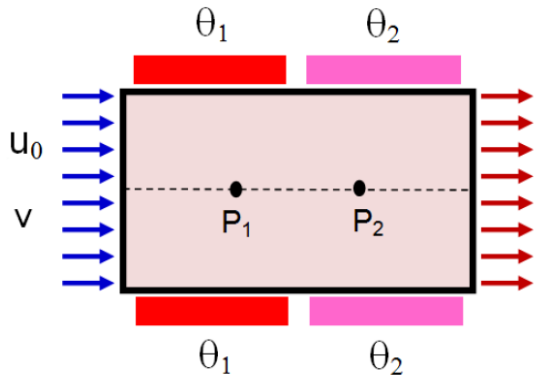

FIGURE 4 – Un modèle simplifié pour le contrôle en temps réel, P_1 et P_2 sont deux capteurs intégrés dans le système

Pour cette section on utilisera une fonction coût plus ou moins physique, qui est la différence entre l'intégrale de la température sur la ligne médiane et une constante β_1. On pourra interpréter β_1 comme la quatité de chaleur nécessaire à la résine pour réticuler dans le four. La limite supérieure des températures des étages de chauffage doit être inférieure à la température de dégradation de la résine. Ainsi la fonction coût s'écrit comme étant :

$$f(\vec{x}) = \frac{1}{2}\left(\int_0^{L_x} T(x, \frac{L_y}{2}, \theta_1, \theta_2)dx - \beta_1\right)^2 \tag{11}$$

La solution optimale est affiché sur la figure (5).

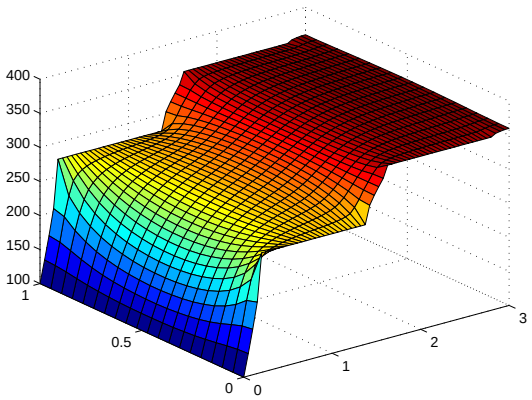

FIGURE 5 – La solution optimale pour la fonction coût (11)

3.1 Contrôle et calcul inverse

Dans cette section on suppose d'abord que le système fonctionne aux valeurs optimales θ_{1op} et θ_{2op}, données par l'optimisation d'une fonction coût préalablement définie. De ce fait, les capteurs P_1 et P_2 mesurent les températures T_{1op} et T_{2op}. On simule une panne en supposant que l'un des capteurs donne une valeur différente de la valeur optimale :

$$\begin{cases} T_{1\textbf{réel}} = T_{1op} \\ T_{2\textbf{réel}} = a \times T_{2op} \end{cases}$$

Le problème s'exprime donc par la minimisation d'une fonction coût définie par :

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=2}^{i=1}\left(U(P_i) - U(x_i, y_i, \theta_1, \theta_2)\right)^2 \tag{12}$$

Cette fonction est minimisé dans l'espace des paramètres I en utilisant la méthode de Levenberg-Marquardt. Ainsi, on définit la fonction coût comme étant une somme de sensibilités :

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=2}^{i=1}\left(r_i^2\right)^2 \tag{13}$$

On définit aussi le Jacobien comme étant la matrice des dérivées des différentes sensibilitées par rapport aux paramètres d'optimisation :

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & \cdots & \frac{\partial r_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial r_m}{\partial x_1} & \frac{\partial r_m}{\partial x_2} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix} \tag{14}$$

On écrit de plus le gradient et l'Hessienne respectivement sous les formes suivantes [4] :

$$\nabla f(\mathbf{x}) = J(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \tag{15}$$

$$\nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) + \sum_{j=1}^{m} \mathbf{r}_j(\mathbf{x}) \nabla^2 \mathbf{r}_j(\mathbf{x}) \tag{16}$$

On note que les sensibilités r_i ainsi que leurs dérivées sont obtenues de façon direct par la PGD comme étant :

$$\frac{\partial r_j}{\partial \theta_1} = \sum_{i=1}^{i=N}\left(\gamma_{ji} \cdot \frac{\partial \Theta_{1i}(\theta_1)}{\partial \theta_1} \cdot \Theta_{2i}(\theta_2)\right) \frac{\partial r_j}{\partial \theta_2} = \sum_{i=1}^{i=N}\left(\gamma_{ji} \cdot \frac{\partial \Theta_{2i}(\theta_2)}{\partial \theta_2} \cdot \Theta_{1i}(\theta_1)\right) \tag{17}$$

avec γ_i le remplacement des coordonnées du capteur P_j dans les fonctions $F_i(x,y)$. Le problème itératif de minimisation est donc la convergence de l'équation (18) :

$$x^{k+1} = x^k - \left[J\left(\mathbf{x}^\mathbf{k}\right)^T J\left(\mathbf{x}^\mathbf{k}\right) + \lambda I\right]^{-1} \times J\left(\mathbf{x}^\mathbf{k}\right)^T \mathbf{r}\left(\mathbf{x}^\mathbf{k}\right) \tag{18}$$

On note que cet algorithme est différent de l'algorithme de Newton par le fait de remplacer le terme englobant la dérivée seconde : $\sum_{j=1}^{m} \mathbf{r}_j(\mathbf{x}) \nabla^2 \mathbf{r}_j(\mathbf{x})$ par $\lambda\bar{\bar{I}}$, λ étant un paramètre qui évolue dans l'algorithme [4]. D'autre part, afin d'accélérer le contrôle on-line, toutes les dérivées des sensibilités, ainsi que le Jacobien sont stockés en mémoire de façon générique. Ceci est possible grâce à la représentation séparée de la solution donnée par la PGD comme montre l'équation (17). La figure (6) montre le champ de température de la simulation d'une panne dans le deuxième étage d'échauffement. La solution est obtenue après 5 itérations et avec 1.5 milliseconde seulement sur un simple ordinateur portable, en utilisant MATLAB.

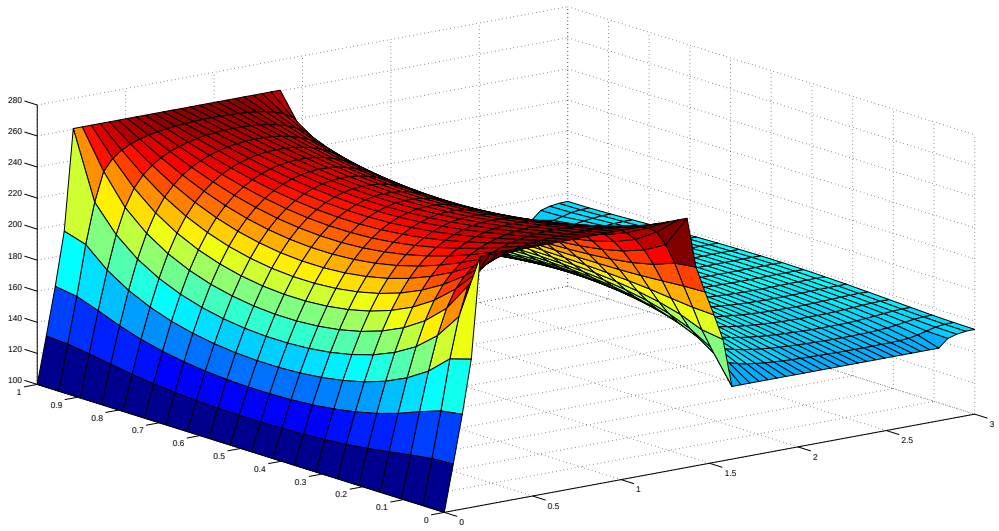

FIGURE 6 – La solution donnée par le calcul inverse, en considérant $a = 0.4$

3.2 Reconfiguration du système

Dans cette section on se base sur les données obtenues du calcul inverse pour reconfigurer le système à l'optimum. Pour cela, on reprend la même fonction coût de l'équation (11). On minimise en supposant que le second étage d'échauffement est défectueux. Comme régime d'urgence, le four fonctionnera en utilisant la température $\theta_{i\,\mathbf{réel}}$ détectée par le calcul inverse décrit dans la partie (3.1). La fonction coût se réduit à :

$$f(\overline{x}) = \frac{1}{2}\left(\int_0^{L_x} T(x, \frac{L_y}{2}, \theta_1, \theta_{2\,\mathbf{réel}}) - \beta_1\right)^2 \tag{19}$$

Dans ce cas, on optimise aussi en utilisant la méthode de Levenberg-Marquardt par la même méthodologie décrite dans la section (3.1). L'optimisation est achevée au bout de 4 itérations seulement. Le temps nécessaire pour calculer l'optimum de la reconfiguration du système est inférieur à une milliseconde sur un ordinateur portable utilisant MATLAB. La solution est affichée sur la figure (7).

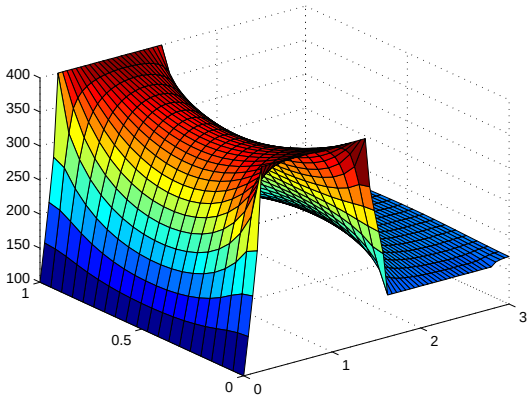

FIGURE 7 – La solution donnée par la minimisation de (19)

On déduit donc que tous les calculs réalisées on-line nécessitent moins que trois millisecondes. Sur ce, cette méthode a été portée sur une plate-forme mobile, pour démontrer l'efficacité et la simplicité de

l'algorithme utilisé.

4 Conclusion

Dans ce travail on illustre les possibilités qu'offrent la PGD dans le cadre du contrôle en temps réel. Pour cela il faut réaliser un calcul puissant off-line, englobant toutes les sources de variabilités du système comme coordonnées supplémentaires du problème. De plus, on réalise l'optimisation du modèle off-line. Cette optimisation est réalisée par deux méthodes différentes, on illustre ici la méthode des moindres carrées mobiles. La méthode de Levenberg-Marquardt est illustrée plus tard dans le contrôle. Enfin, on couple la résolution « générique » off-line à une technique de contrôle et reconfiguration on-line, réalisée en place sur des plate-formes mobiles.

Cette méthode constitue un changement de paradigme dans le domaine de la mécanique numérique. Bien que l'exemple montré est loin d'être dessisif, la méthode est robuste et on cherchera à exploiter les possibilités offertes, aussi bien que les limites.

Références

[1] A. Jacob. *Globalisation Of The Pultrusion Industry*, Reinforced Plastics 50(5), 38-41, 2006.

[2] C. Ghnatios, F. Chinesta, E. Cueto, A. Leygue, A. Poitou, P. Breitkopf, P. Villon. *Parametric thermal modelling of pultrusion processes*, Composite Part A, *submitted*.

[3] C. Ghnatios, F. Masson, A. Huerta, E. Cueto, F. Chinesta. *Proper Generalized Decomposition Based Dynamic Data-Driven of Thermal Processes*, Computational Methods in Applied Mechanics and Engineering, *submitted*.

[4] D. Marquardt. *An algorithm for least-squares estimation of nonlinear parameters*, SIAM Journal on Applied Mathematics 11, 431-441, 1963.

[5] F. Chinesta, A. Ammar, E. Cueto. *Proper generalized decomposition of multiscale models*, International Journal of Numerical Methods in Engineering 83, 1114-1132, 2010.

[6] F. Chinesta, A. Ammar, E. Cueto. *Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models*, Archives of Computational Methods in Engineering 17, 327-350, 2009.

[7] J. Li, S.C. Joshi, Y.C. Lam. *Curing optimization for pultruded composite sections*, Composites Science and Technology 62(3), 457-467,2002.

[8] P. Breitkopf, A. Racineux, P. Villon. *An introduction to moving least squares meshfree methods*, Meshfree Computational Mechanics 11, 825-867, 2002.

[9] P. Breitkopf, H. Naceur, A. Racineux, P. Villon. *Moving least squares response surface approximation formulation and metal forming applications*, Computers And Structures 83, 1411-1428, 2005.

[10] P. Ladeveze. *Nonlinear computational structural mechanics*, Springer NY, 1998.