

Manuel de Référence
Fascicule R4.03 : Analyse de sensibilité
Document : R4.03.06

Algorithme de recalage

Résumé :

Dans ce document est présenté l'algorithme de recalage de `MACR_RECAL`. Il s'agit d'un algorithme de Levenberg-Marquardt avec bornes.

On décrit dans un premier temps la méthode générale avant d'en préciser certains éléments. Sont détaillés le calcul de la fonctionnelle, de la matrice Jacobienne, la détermination du paramètre initial de régularisation ainsi que son évolution, la gestion des bornes et le critère de convergence.

Table des matières

1	Introduction	3
2	Algorithme de Levenberg-Marquardt.....	4
2.1	Position du problème.....	4
2.2	Résolution.....	4
3	Mise en œuvre pratique.....	7
3.1	Définition de la fonctionnelle	7
3.2	Expression de la matrice Jacobienne	8
3.3	Régularisation du système linéaire	9
3.3.1	Valeur initiale de λ	9
3.3.2	Evolution de la valeur de λ	9
3.4	Limitations du domaine d'évolution des paramètres.....	9
3.5	Adimensionnement.....	10
3.6	Critère de convergence	11
4	Algorithme global.....	12
5	Bibliographie	12

1 Introduction

Avant d'aborder à proprement parler la problématique du recalage, il est utile de rappeler quelques éléments sur l'identification de paramètres. Supposons que l'on désire identifier n paramètres à partir d'un essai mécanique donné. Dans le cadre de cette identification, on définit les grandeurs :

- \mathbf{c} , le vecteur des n paramètres à identifier, appartenant à \mathbf{O} , convexe fermé de \mathbb{R}^n .
- \mathbf{d} , le vecteur des grandeurs calculées lors d'une simulation de l'essai en utilisant les paramètres \mathbf{c} , par opposition à \mathbf{d}^{exp} , le vecteur des grandeurs mesurées lors d'un essai expérimental. Tous deux appartiennent à l'espace \mathbf{L} des grandeurs observables. La simulation de l'essai expérimental, paramétrée par le vecteur \mathbf{c} , peut être réalisée par différentes méthodes : différences finies, éléments finis, éléments de frontière, C'est ce que nous appellerons le problème direct.

Le but de l'identification est de déterminer le jeu de paramètres \mathbf{c} réduisant la différence entre grandeurs mesurées et expérimentales (en espérant fortement que la réduction de cet écart soit suffisante pour obtenir le jeu de paramètres souhaité ...). On introduit donc une fonctionnelle coût notée \mathbf{J} dépendant de \mathbf{c} et mesurant la distance entre \mathbf{d} et \mathbf{d}^{exp} .

$$\mathbf{J}(\mathbf{c}) = \|\mathbf{d} - \mathbf{d}^{\text{exp}}\| \quad \text{éq 1-1}$$

où $\|\cdot\|$ désigne une norme sur \mathbf{L} .

L'identification s'exprime donc sous la forme du problème de minimisation suivant :

$$\text{Déterminer } \mathbf{c}^* \in \mathbf{O} \text{ tel que } \mathbf{J}(\mathbf{c}^*) = \min_{\mathbf{c} \in \mathbf{O}} \mathbf{J}(\mathbf{c})$$

Enfin, on définit le recalage comme la minimisation d'un type de fonctionnelles particulier dites « moindres carrés » qui s'expriment sous la forme :

$$\mathbf{J}(\mathbf{c}) = \sum_{n=1}^N j_n^2(\mathbf{c}) \quad \text{éq 1-2}$$

Il est communément admis que l'algorithme de minimisation le plus efficace pour ce type de fonctionnelles est l'algorithme de Levenberg-Marquardt. C'est ce dernier qui est implanté dans la commande `MACR_RECAL` du *Code_Aster* et que nous présentons dans la suite.

2 Algorithme de Levenberg-Marquardt

2.1 Position du problème

Il existe plusieurs familles d'algorithmes de minimisation [bib1]. Pour les problèmes relativement réguliers, les plus utilisées sont les méthodes de descente. Leur principe est de générer de manière itérative une suite $(\mathbf{c}^k)_{k \in \mathbb{N}}$ définie par :

$$(\mathbf{c}^{k+1}) = \mathbf{c}^k + \alpha^k \mathbf{g}^k \quad \text{éq 2.1-1}$$

telle que, pour $f(x) = \mathbf{J}(\mathbf{c}^k + x \mathbf{g}^k)$, $x \in \mathbb{R}_+$

- $f(x)$ est décroissante au voisinage de 0^+
- $f(\alpha^k) = \min_{x>0} f(x)$

\mathbf{g}^k est la direction de descente au pas k . C'est la méthode de détermination de \mathbf{g}^k qui conditionne la nature donc l'efficacité de l'algorithme utilisé, sachant que ces techniques sont principalement basées sur des approximations de \mathbf{J} à l'ordre 1 ou à l'ordre 2. Pour l'algorithme de Levenberg-Marquardt, on manipule une approximation à l'ordre 2 de la fonctionnelle.

2.2 Résolution

Dans le cadre du recalage, on manipule des fonctionnelles coût moindre carré du type :

$$\mathbf{J}(\mathbf{c}) = \sum_{n=1}^N j_n^2(\mathbf{c}) \quad \text{éq 2.2-1}$$

où par exemple $j_n(\mathbf{c}) = (F_n^{calc}(\mathbf{c}) - F_n^{exp})$, avec des notations évidentes.

La particularité de ces fonctionnelles coût réside dans le fait que l'on connaît la forme de leurs dérivées premières et secondes :

$$(\nabla_{\mathbf{c}} \mathbf{J}(\mathbf{c}))_i = 2 \sum_{n=1}^N j_n(\mathbf{c}) \frac{\partial j_n}{\partial c_i} \quad \text{éq 2.2-2}$$

$$(\mathbf{H}(\mathbf{c}))_{ij} = 2 \sum_{n=1}^N \left(\frac{\partial j_n}{\partial c_i} \cdot \frac{\partial j_n}{\partial c_j} + j_n(\mathbf{c}) \frac{\partial^2 j_n}{\partial c_i \partial c_j} \right) \quad \text{éq 2.2-3}$$

Alors, en supposant que le deuxième terme de l'équation précédente est négligeable devant le premier (ce qui est vrai quand les j_k sont linéaires en \mathbf{c} : ce terme est nul), on peut réécrire :

$$(\mathbf{H}(\mathbf{c}))_{ij} \approx 2 \sum_{n=1}^N \frac{\partial j_n}{\partial c_i} \cdot \frac{\partial j_n}{\partial c_j} \quad \text{éq 2.2-4}$$

Il est intéressant à ce niveau d'introduire la matrice de sensibilité ou matrice Jacobienne définie par :

$$\mathbf{A} = \begin{bmatrix} \frac{\partial j_1}{\partial c_1} & \frac{\partial j_1}{\partial c_2} & \dots & \frac{\partial j_1}{\partial c_n} \\ \frac{\partial j_2}{\partial c_1} & \frac{\partial j_2}{\partial c_2} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \frac{\partial j_N}{\partial c_1} & \frac{\partial j_N}{\partial c_2} & \dots & \frac{\partial j_N}{\partial c_n} \end{bmatrix} \quad \text{éq 2.2-5}$$

On peut ainsi exprimer le gradient et le Hessien par :

$$\nabla_{\mathbf{c}} \mathbf{J}(\mathbf{c}^k) = 2 \mathbf{A}^T \mathbf{j} \quad \text{éq 2.2-6}$$

$$\mathbf{H}(\mathbf{c}^k) \approx 2 \mathbf{A}^T \mathbf{A} \quad \text{éq 2.2-7}$$

avec $\mathbf{j} = [j_1, \dots, j_N]^T$.

Ecrivons alors le développement limité à l'ordre 2 de \mathbf{J} :

$$\mathbf{J}(\mathbf{c}) \approx \mathbf{J}(\mathbf{c}^k) + (\mathbf{c} - \mathbf{c}^k)^T \cdot \nabla_{\mathbf{c}} \mathbf{J}(\mathbf{c}^k) + \frac{1}{2} (\mathbf{c} - \mathbf{c}^k)^T \mathbf{H}(\mathbf{c}^k) (\mathbf{c} - \mathbf{c}^k) \quad \text{éq 2.2-8}$$

Soit $\mathbf{g}^k = \mathbf{c} - \mathbf{c}^k$, le pas de descente au point \mathbf{c}^k , il doit vérifier la condition de stationnarité de l'approximation quadratique :

$$\nabla_{\mathbf{c}} \mathbf{J}(\mathbf{c}^k) + \mathbf{H}(\mathbf{c}^k) \mathbf{g}^k = 0 \quad \text{éq 2.2-9}$$

D'après l'expression du gradient et du Hessien de \mathbf{J} , on peut écrire :

$$(\mathbf{A}^T \mathbf{A}) \mathbf{g}^k = -\mathbf{A}^T \mathbf{j} \quad \text{éq 2.2-10}$$

La résolution de cette équation mène à un algorithme connu sous le nom de Gauss-Newton, très efficace mais qui présente néanmoins quelques inconvénients :

- $(\mathbf{A}^T \mathbf{A})$ peut être quasiment singulière et causer la non-existence de solution.
- On n'a aucun contrôle sur \mathbf{g}^k , qui peut être trop grand et donc sortir les paramètres de l'espace admissible.

Pour pallier ces inconvénients, on préfère utiliser l'algorithme de Levenberg-Marquardt qui propose une régularisation de l'algorithme de Gauss-Newton :

$$(\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}) \mathbf{g}^k = -\mathbf{A}^T \mathbf{j} \quad \text{éq 2.2-11}$$

où λ est un scalaire et \mathbf{I} la matrice identité.

On remarque que si $\lambda = 0$, on retrouve la direction donnée par Gauss-Newton et si $\lambda \rightarrow +\infty$, on retrouve la direction donnée par l'opposée du gradient de \mathbf{J} i.e. la plus grande pente.

L'algorithme de Levenberg-Marquardt consiste donc, en partant d'une valeur de λ « assez élevée », de la diminuer d'un facteur 10 par exemple, à chaque décroissance de \mathbf{J} . On passe ainsi graduellement d'un algorithme de plus grande pente à l'algorithme de Gauss-Newton. On peut donc présenter cette procédure sous la forme :

- Choix d'un point de départ \mathbf{c}^0 et d'une valeur initiale de λ
- A l'itération k , résoudre

$$(\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}) \mathbf{g}^k = -\mathbf{A}^T \mathbf{j}$$

$$\mathbf{c}^{k+1} = \mathbf{c}^k + \mathbf{g}^k$$
- Si $\mathbf{J}(\mathbf{c}^{k+1}) < \mathbf{J}(\mathbf{c}^k)$, alors $\lambda = \lambda / 10$ sinon $\lambda = \lambda * 10$
- Test de convergence

Remarque :

Nous avons considéré ci-dessus l'algorithme de Levenberg-Marquardt sous l'angle de la régularisation de l'algorithme de Gauss-Newton. Il est possible de donner un éclairage différent à cette algorithme en le considérant comme un algorithme de région de confiance [bib2]. En effet, on peut montrer aisément que le système [éq 2.2-11] est la condition de stationnarité du problème de minimisation :

$$\text{Déterminer } \mathbf{g}^k \text{ tel que } \mathbf{g}^k = \text{ArgMin} \left(\mathbf{g}^{kT} \cdot \mathbf{A}^T \mathbf{j} + \frac{1}{2} \mathbf{g}^{kT} \mathbf{A}^T \mathbf{A} \mathbf{g}^k \right) \text{ soumis à } \|\mathbf{g}^k\| \leq D^k$$

$$\text{Où } D^k = -(\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{j} \text{ et } \lambda \geq 0.$$

Il s'agit là d'une implantation très simple de l'algorithme de Levenberg-Marquardt au sein de laquelle différentes questions ne sont pas abordées :

- Comment définir la fonctionnelle \mathbf{J} quand on dispose de plusieurs essais ?
- Comment choisir la valeur initiale de λ ?
- Comment faire évoluer λ de manière plus fine ?
- Comment définir le domaine d'évolutions de chaque paramètre ?

Nous allons préciser ces différents points dans la suite.

3 Mise en œuvre pratique

3.1 Définition de la fonctionnelle

Lors d'un recalage, l'utilisateur dispose souvent de plusieurs mesures différentes ; il s'agit de grandeurs physiques discrètes, éventuellement de natures différentes, mesurées au cours d'un ou plusieurs essais. Ce sont des fonctions d'un paramètre donné noté t (temps, abscisse, ...) que l'on peut donc représenter par :

$$F_1^{\text{exp}} = \begin{bmatrix} t_1 & f_1^{\text{exp}}(t_1) \\ \vdots & \vdots \\ t_N & f_1^{\text{exp}}(t_N) \end{bmatrix} \quad F_2^{\text{exp}} = \begin{bmatrix} t_1 & f_2^{\text{exp}}(t_1) \\ \vdots & \vdots \\ t_M & f_2^{\text{exp}}(t_M) \end{bmatrix} \quad F_L^{\text{exp}} = \begin{bmatrix} t_1 & f_L^{\text{exp}}(t_1) \\ \vdots & \vdots \\ t_P & f_L^{\text{exp}}(t_P) \end{bmatrix} \quad \text{éq 3.1-1}$$

Chacune de ces mesures expérimentales dispose de son pendant

$$F_1^{\text{calc}}(\mathbf{c}^k) = \begin{bmatrix} \tilde{t}_1 & f_1^{\text{calc}}(\mathbf{c}^k, \tilde{t}_1) \\ \vdots & \vdots \\ \tilde{t}_I & f_1^{\text{calc}}(\mathbf{c}^k, \tilde{t}_I) \end{bmatrix} \quad F_2^{\text{calc}}(\mathbf{c}^k) = \begin{bmatrix} \tilde{t}_1 & f_2^{\text{calc}}(\mathbf{c}^k, \tilde{t}_1) \\ \vdots & \vdots \\ \tilde{t}_J & f_2^{\text{calc}}(\mathbf{c}^k, \tilde{t}_J) \end{bmatrix} \quad F_L^{\text{calc}}(\mathbf{c}^k) = \begin{bmatrix} \tilde{t}_1 & f_L^{\text{calc}}(\mathbf{c}^k, \tilde{t}_1) \\ \vdots & \vdots \\ \tilde{t}_K & f_L^{\text{calc}}(\mathbf{c}^k, \tilde{t}_K) \end{bmatrix} \quad \text{éq 3.1-2}$$

calculé pour un jeu de paramètre \mathbf{c}^k donné. Remarquons que les grandeurs calculées ne sont pas forcément en même nombre que les grandeurs mesurées ni évaluées pour la même valeur du paramètre t . On peut alors définir la fonctionnelle moindres carrés à minimiser par :

$$\mathbf{J}(\mathbf{c}^k) = \frac{\sum_{i=1}^N \left(\frac{f_1^{\text{exp}}(t_i) - f_1^{\text{calc}}(\mathbf{c}^k, t_i)}{f_1^{\text{exp}}(t_i)} \right)^2 + \sum_{i=1}^M \left(\frac{f_2^{\text{exp}}(t_i) - f_2^{\text{calc}}(\mathbf{c}^k, t_i)}{f_2^{\text{exp}}(t_i)} \right)^2 + \dots + \sum_{i=1}^P \left(\frac{f_L^{\text{exp}}(t_i) - f_L^{\text{calc}}(\mathbf{c}^k, t_i)}{f_L^{\text{exp}}(t_i)} \right)^2}{\mathbf{J}(\mathbf{c}^0)} \quad \text{éq 3.1-3}$$

Il est important de remarquer que :

- si une mesure calculée f_j^{calc} n'est pas définie à un instant t_i , alors on interpole linéairement sa valeur
- si une mesure expérimentale f_j^{exp} est nulle, on ne divise pas la quantité $f_j^{\text{exp}}(t_i) - f_j^{\text{calc}}(\mathbf{c}^k, t_i)$ et elle est présente telle quelle dans l'expression de la fonctionnelle
- la fonctionnelle \mathbf{J} est normalisée de manière à valoir 1. au début des itérations de recalage

3.2 Expression de la matrice Jacobienne

Pour le calcul de la matrice Jacobienne, on définit le vecteur \mathbf{j} des erreurs par :

$$\mathbf{j} = \begin{bmatrix} \frac{f_1^{\text{exp}}(t_1) - f_1^{\text{calc}}(\mathbf{c}^k, t_1)}{f_1^{\text{exp}}(t_1)} \\ \vdots \\ \frac{f_1^{\text{exp}}(t_N) - f_1^{\text{calc}}(\mathbf{c}^k, t_N)}{f_1^{\text{exp}}(t_N)} \\ \frac{f_2^{\text{exp}}(t_1) - f_2^{\text{calc}}(\mathbf{c}^k, t_1)}{f_2^{\text{exp}}(t_1)} \\ \vdots \\ \frac{f_P^{\text{exp}}(t_L) - f_P^{\text{calc}}(\mathbf{c}^k, t_L)}{f_P^{\text{exp}}(t_L)} \end{bmatrix} \quad \text{éq 3.2-1}$$

Soit :

$$j_i^K = \frac{f_K^{\text{exp}}(t_i) - f_K^{\text{calc}}(\mathbf{c}^k, t_i)}{f_K^{\text{exp}}(t_i)} \quad \text{éq 3.2-2}$$

On retrouve alors l'expression de la matrice Jacobienne de [éq 2.2-5] :

$$\mathbf{A} = \begin{bmatrix} \frac{\partial j_1^1}{\partial c_1} & \frac{\partial j_1^1}{\partial c_2} & \dots & \frac{\partial j_1^1}{\partial c_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial j_N^1}{\partial c_1} & \frac{\partial j_N^1}{\partial c_2} & \dots & \frac{\partial j_N^1}{\partial c_n} \\ \frac{\partial j_1^2}{\partial c_1} & \frac{\partial j_1^2}{\partial c_2} & \dots & \dots \\ \frac{\partial j_1^2}{\partial c_1} & \frac{\partial j_1^2}{\partial c_2} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \frac{\partial j_L^P}{\partial c_1} & \frac{\partial j_L^P}{\partial c_2} & \dots & \frac{\partial j_L^P}{\partial c_n} \end{bmatrix} \quad \text{éq 3.2-3}$$

Où les termes sont calculés par différences finies directes :

$$\frac{\partial j_1^1}{\partial c_i}(c_1, \dots, c_i, \dots, c_n) \approx \frac{j_1^1(c_1, \dots, c_i + \alpha c_i, \dots, c_n) - j_1^1(c_1, \dots, c_i, \dots, c_n)}{\alpha c_i} \quad \text{éq 3.2-4}$$

3.3 Régularisation du système linéaire

Nous abordons ici le problème de la détermination et de l'évolution du paramètre de régularisation λ . On définit pour ce faire :

- $\lambda_{\max} = \text{Max}$ (Valeurs propres de $\mathbf{A}^T \mathbf{A}$), $\lambda_{\min} = \text{Min}$ (Valeurs propres de $\mathbf{A}^T \mathbf{A}$),
 $\text{cond} = \lambda_{\max} / \lambda_{\min}$ si $\lambda_{\min} \neq 0$
- $\mathbf{Q}(\mathbf{c}) = \mathbf{J}(\mathbf{c}^k) + (\mathbf{c} - \mathbf{c}^k)^T \cdot \mathbf{A}^T \mathbf{j} + \frac{1}{2} (\mathbf{c} - \mathbf{c}^k)^T \cdot (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}) (\mathbf{c} - \mathbf{c}^k)$

3.3.1 Valeur initiale de λ

Connaissant les grandeurs ci-dessus, on définit l'algorithme suivant :

- Si $\lambda_{\min} = 0$, alors $\lambda = 1.E-3 \lambda_{\max}$
- Sinon
 - Si $\text{cond} < 1.E5$, alors $\lambda = 1.E-16 \lambda_{\max}$
 - Sinon $\lambda = \text{Abs}(1.E5 \lambda_{\min} - \lambda_{\max}) / 10001$

Remarque :

Dans le dernier cas, la valeur attribuée à λ a pour effet de ramener le conditionnement de $\mathbf{A}^T \mathbf{A}$ à 1.E5

3.3.2 Evolution de la valeur de λ

Pour faire évoluer λ , on définit le ratio $R^k = \frac{\mathbf{J}(\mathbf{c}^k) - \mathbf{J}(\mathbf{c}^{k+1})}{\mathbf{Q}(\mathbf{c}^k) - \mathbf{Q}(\mathbf{c}^{k+1})}$, qui permet d'évaluer la validité de

l'approximation quadratique de \mathbf{J} : plus il est proche de 1, plus cette approximation est valable. On en déduit l'algorithme suivant [bib2] :

- Si $R^k < 0.25$, alors $\lambda = \lambda * 10$
- Si $R^k > 0.75$, alors $\lambda = \lambda / 15$

3.4 Limitations du domaine d'évolution des paramètres

Pour des raisons diverses telles que pour garantir la validité physique des paramètres (module d'Young strictement positif, coefficient de Poisson compris entre 0 et 0.5, ...), il est nécessaire de limiter leur domaine d'évolution. On impose donc que \mathbf{c} reste dans un domaine admissible \mathbf{O} , convexe fermé de \mathbb{R}^n . Ceci impose donc des contraintes sur les paramètres :

$$\mathbf{c}_{\text{sup}} > \mathbf{c}^k + \mathbf{g}^k > \mathbf{c}_{\text{inf}}$$

Après dualisation de ces conditions par introduction des multiplicateurs de Lagrange μ_{\inf} et μ_{\sup} , on résout le système :

Trouver \mathbf{g}^k μ_{\inf} μ_{\sup} tels que

$$\begin{cases} (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}) \mathbf{g}^k + \mu_{\inf} + \mu_{\sup} = -\mathbf{A}^T \mathbf{j} \\ \mathbf{c}^k + \mathbf{g}^k > \mathbf{c}_{\inf} \\ \mu_{\inf} > 0 \\ (\mathbf{c}^k + \mathbf{g}^k - \mathbf{c}_{\inf})_i (\mu_{\inf})_i = 0 \quad \forall i = [1, n] \\ \mathbf{c}^k + \mathbf{g}^k < \mathbf{c}_{\sup} \\ \mu_{\sup} < 0 \\ (\mathbf{c}^k + \mathbf{g}^k - \mathbf{c}_{\sup})_i (\mu_{\sup})_i = 0 \quad \forall i = [1, n] \end{cases}$$

Cette résolution s'effectue à l'aide d'un algorithme de contraintes actives. Pour toute précision sur cet algorithme, se reporter à [bib3] ou [bib4].

3.5 Adimensionnement

On est souvent amenés à identifier des paramètres de natures physiques diverses. Les ordres de grandeur de ces paramètres peuvent être extrêmement différents. Ceci peut engendrer de très fortes différences dans les ordres de grandeur des composantes du gradient et du Hessien de la fonctionnelle coût et compromettre la résolution.

Pour pallier cette difficulté, il est impératif d'adimensionner les inconnues avant de débiter la résolution. Voici une procédure simple et efficace.

Soit $\mathbf{c}^0 = {}^T [c_1^0, c_2^0, \dots, c_n^0]$, le vecteur initial des grandeurs à reconstruire. On définit la matrice d'adimensionnement \mathbf{D} :

$$\mathbf{D} = \begin{bmatrix} c_1^0 & & & \\ & c_2^0 & & \\ & & & c_{n-1}^0 \\ & & & & c_n^0 \end{bmatrix} \quad \text{éq 3.5-1}$$

Alors, si les c_i^0 sont tous non nuls, on peut définir les inconnues adimensionnelles par :

$$\tilde{\mathbf{c}}^0 = \mathbf{D}^{-1} \cdot \mathbf{c}^0 \quad \text{éq 3.5-2}$$

De même, on introduit une fonctionnelle coût adimensionnelle :

$$\tilde{\mathbf{J}}(\tilde{\mathbf{c}}) = \mathbf{J}(\mathbf{D} \cdot \tilde{\mathbf{c}}) = \mathbf{J}(\mathbf{c}) \quad \text{éq 3.5-3}$$

Ainsi que son gradient :

$$\nabla_{\tilde{\mathbf{c}}} \tilde{\mathbf{J}}(\tilde{\mathbf{c}}) = \frac{\partial \tilde{\mathbf{J}}(\tilde{\mathbf{c}})}{\partial \tilde{\mathbf{c}}} = \frac{\partial \mathbf{J}(\mathbf{D}.\tilde{\mathbf{c}})}{\partial \tilde{\mathbf{c}}} = \frac{\partial \mathbf{J}(\mathbf{c})}{\partial \mathbf{c}} \cdot \frac{\partial \mathbf{c}}{\partial \tilde{\mathbf{c}}} = \mathbf{D}.\nabla_{\mathbf{c}} \mathbf{J}(\mathbf{c}) \quad \text{éq 3.5-4}$$

Et sa matrice Jacobienne :

$$\tilde{A}_{ij} = A_{ij} * c_j^0 \quad \text{éq 3.5-5}$$

D'un point de vue algorithmique, le calcul de la matrice Jacobienne se fait classiquement avec la fonctionnelle \mathbf{J} , puis elle est adimensionnée ainsi que les paramètres courants \mathbf{c} , avant d'être transmis à l'algorithme de minimisation. A la sortie de ce dernier, les paramètres $\tilde{\mathbf{c}}$ sont redimensionnés pour permettre le calcul de la fonctionnelle \mathbf{J} .

3.6 Critère de convergence

Le critère de convergence utilisé dans MACR_RECAL consiste à tester la décroissance du gradient de la fonctionnelle. On rappelle que l'utilisation de ce critère se justifie naturellement par le fait que l'objectif de l'algorithme de recalage est d'annuler ce gradient.

$$\frac{\|\nabla_{\mathbf{c}} \mathbf{J}(\mathbf{c}^k)\|}{\|\nabla_{\mathbf{c}} \mathbf{J}(\mathbf{c}^0)\|} < \text{Prec.} \quad \text{éq 3.6-1}$$

Où Prec est par défaut pris égal à 1.E-3.

4 Algorithme global

De manière à expliciter l'enchaînement des diverses opérations décrites ci-dessus, on présente formellement l'algorithme du recalage :

- Initialisations
 - $k = 0$
 - calcul de \mathbf{A} , adimensionnement de \mathbf{A}
 - Calcul de λ initial
 - **Itérations globales**
 - Adimensionnement de \mathbf{c}^k
 - Résolution de l'équation de Levenberg-Marquardt
 - Imposition du respect des bornes
 - Redimensionnement de \mathbf{c}^{k+1}
 - Calcul de $\mathbf{J}(\mathbf{c}^{k+1})$
 - Actualisation de λ
 - Calcul de \mathbf{A} , adimensionnalisation de \mathbf{A}
 - Test de convergence
 - $k = k + 1$
- Fin

5 Bibliographie

- [1] J.C. CULIOLI : « Introduction à l'optimisation », Ellipses, 1994
- [2] M. S. BAZARAA, H.D. SHERALI & C.M. SHETTY : « Nonlinear Programming, Theory and Algorithms », Wiley & Sons, 1979
- [3] « Contact par des conditions cinématiques », Document *Code_Aster* [R5.03.50]
- [4] K. KUNISCH & F. RENDL ; « An infeasible active set method for quadratic programming with simple bounds », SIAM Journal on Optimization, Volume 14, Number 1, 2003