Tentative de traitement d'un problème d'identification de paramètres dans un système différentiel en octave ou matlab ou python.

Il s'agit d'estimer les paramètres $\boldsymbol{m} = (c_{res}, c_s, r_i, r_0, r_f) \in \mathbb{R}^5$ qui apparaissent dans la matrice :

$$\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{m}) = \begin{pmatrix} -\frac{1}{c_{res}}(\frac{1}{r_i} + \frac{1}{r_f}) & \frac{1}{c_{res} r_i} \\ \frac{1}{c_s r_i} & -\frac{1}{c_s}(\frac{1}{r_i} + \frac{1}{r_0}) \end{pmatrix} \in \mathbb{R}^{2\times 2}$$

ainsi que dans une fonction du temps t, $\boldsymbol{G}(t) = \boldsymbol{G}(t, \boldsymbol{m}) = \begin{pmatrix} \frac{Q_{res}(t)}{c_{res}} + \frac{T_{ext}(t)}{c_{res} r_f} \\ \frac{Q_s(t)}{c_s} + \frac{T_{ext}(t)}{c_s r_0} \end{pmatrix} \in \mathbb{R}^2$, où $Q_{res}(t)$, $T_{ext}(t)$ et $Q_s(t)$ sont des données (mesurées), pour que la solution $t \in \mathbb{R} \to \boldsymbol{T}(t) = \begin{pmatrix} T_i(t) \\ T_s(t) \end{pmatrix} \in \mathbb{R}^2$ du système différentiel :

$$\frac{d\boldsymbol{T}}{dt}(t) = \boldsymbol{A}\boldsymbol{T}(t) + \boldsymbol{G}(t) \text{ pour } 0 \le t \le tmax,\ \boldsymbol{T}(0) = \begin{pmatrix} T_i(0) \\ T_s(0) \end{pmatrix} = \begin{pmatrix} T_i^{ini} \\ T_s^{ini} \end{pmatrix},$$

minimise la quantité ($\begin{pmatrix} T_i^{ini} \\ T_s^{ini} \end{pmatrix}$supposé connu) :

$$J = \frac{1}{2}\int_0^{tmax} (T_i(t) - T_{int}^d(t))^2 dt$$

$T_{int}^d(t)$ étant une fonction donnée (mesurée ; (on pourrait aussi minimiser $(T_i(t) - T_{int}^d(t))^2 + (T_s(t) - T_s^d(t))^2$ avec $T_s^d(t)$ donnée).

Pour cela il faut d'abord résoudre l'équation différentielle : on peut le faire numériquement (soi-même avec un schéma d'Euler implicite ou de Crank-Nicholson (ou trapèze) ou avec la fonction *lsode* d'octave (ou une autre par exemple *ode15s* qui est compatible matlab) ou encore explicitement puisque ici $\boldsymbol{A}$ est constante (indépendante du temps).

Il faut ensuite trouver un moyen de calculer le gradient de J par rapport à $\boldsymbol{m} = (c_{res}, c_s, r_i, r_0, r_f)$. Pour cela on peut résoudre autant d'équations différentielles qu'il y a de paramètres, (donc $4+$ quelque chose comme dimension de $\boldsymbol{m}$), une de ces équations étant par exemple (pour la dérivée par rapport à c_{res}) :

$$\frac{d\boldsymbol{z}^{res}}{dt}(t) = \boldsymbol{A}\boldsymbol{z}^{res}(t) + \underbrace{\begin{pmatrix} \frac{1}{c_{res}^2}(\frac{1}{r_i} + \frac{1}{r_f}) & -\frac{1}{c_{res}^2 r_i} \\ 0 & 0 \end{pmatrix}}_{\frac{\partial}{\partial c_{res}}\boldsymbol{A}} \boldsymbol{z}^{res}(t) + \underbrace{\begin{pmatrix} -\frac{1}{c_{res}}(\frac{Q_{res}(t)}{c_{res}} + \frac{T_{ext}(t)}{c_{res} r_f}) \\ 0 \end{pmatrix}}_{\frac{\partial}{\partial c_{res}}\boldsymbol{G}(t)} \text{ pour } 0 \le t \le tmax,$$

$$\boldsymbol{z}^{res}(0) = \begin{pmatrix} z_{int}^{res}(0) \\ z_s^{res}(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

alors la composante du gradient en c_{res} est :

$$\frac{\partial J}{\partial c_{res}} = \int_0^{tmax} (T_i(t) - T_{int}^d(t)) z_i^{res}(t) dt.$$

On va voir plus loin qu'on peut obtenir ce gradient en ne résolvant que 2 systèmes différentiels au lieu de $4+$ (quelque chose comme dimension de $\boldsymbol{m}$), mais après avoir déterminé ce gradient il faut un code d'optimisation :

bien sûr on peut faire une méthode de descente comme commencée mais cela sera lent ;

mais dans octave il y a un code (fonction *fminunc (sqp* est pour l'optimisation sous contraintes)) qui utilise des techniques assez raffinées (*bfgs*[1]etc ...) ainsi que dans python (scipy) ;

et bien sûr dans matlab on trouve une grande variété de possibilités dont *fminunc*[2] (*fminunc* d'octave semble voisin de celui de matlab mais ce n'est pas sûr).

Mais voyons la technique de l'état adjoint. On va en faire deux présentations. La première, assez "physicienne", consiste à écrire d'abord l'équation différentielle de la variation $\boldsymbol{\delta T} = \begin{pmatrix} \delta T_{int} \\ \delta T_{sur} \end{pmatrix}$ de $\boldsymbol{T}$ qui provient de variations $\boldsymbol{\delta A}$, $\boldsymbol{\delta m}$ de $\boldsymbol{A}$ et $\boldsymbol{m}$ (comme $\boldsymbol{z}^{res}$) :

1. Ces techniques sont des astuces qui en gros "singent" la méthode de Newton appliquée au calcul du zéro du gradient de J, en faisant des approximations "évolutives" du Hessien de J (dérivées seconde de J par rapport aux paramètres $(c_{res}, c_s, r_i, r_0, r_f)$.

2. Cela a été pour moi l'occasion de chercher à nouveau des bibliothèques fortran contenant des sous-programmes d'optimisation. Dans les bibliothèques (que j'ai gardée) qui viennent des deux livres qui ont conduit probablement à la conception de matlab par C.B. Moler vers 1977 à savoir G.E. Forsythe, M.A. Malcolm, C.B. Moler, 1977, Computer methods for mathematical computations : Prentice-Hall et D. Kahaner, C.B. Moler, S. Nash, 1989, Numerical methods and software : Prentice-Hall, on ne trouve pas de codes satisfaisants. Dans la bibliothèque SLATEC que j'ai gardée également et qui est l'ancêtre de la bibliothèque GSL (gnu scientic library) il y a des outils trop éloignés de ce qu'il faut. Dans netlib qui contient des tas de contributions on doit trouver ce qu'il faut. Dans Harwell il y a de bons outils gratuits un peu anciens (sous licence particulière) qu'on doit retrouver dans la bibliothèque payante NAG. Dans la GSL écrites en C et qui est à l'origine de ce qu'on trouve dans Octave et Python semble-il on commence à trouver des routines convenables. Mais il me semble que matlab est encore plus riche.

$$\frac{d\boldsymbol{\delta T}}{dt}(t) = \boldsymbol{A}\boldsymbol{\delta T}(t) + \boldsymbol{(\delta A)T}(t) + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{m}}(t,\boldsymbol{m})\boldsymbol{\delta m} \text{ pour } 0 \le t \le tmax,\ \boldsymbol{\delta T}(0) = \begin{pmatrix} \delta T_{int}(0) \\ \delta T_s(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

On introduit alors l'état adjoint qui est la solution $\boldsymbol{T_*}(t) = \begin{pmatrix} T_{i^*}(t) \\ T_{s*}(t) \end{pmatrix}$ de l'équation différentielle "rétrograde" qui "rétropropage" les résidus, calculable après $\boldsymbol{T}(t)$ ($\boldsymbol{A}^T$ étant la matrice transposée de $\boldsymbol{A}$) :

$$\frac{d\boldsymbol{T_*}}{dt}(t) = -\boldsymbol{A}^T\boldsymbol{T_*}(t) - \begin{pmatrix} T_i(t) - T_{int}^d(t) \\ 0 \end{pmatrix} \text{ pour } tmax \ge t \ge 0,\ \boldsymbol{T_*}(tmax) = \begin{pmatrix} T_{int*}(tmax) \\ T_{s*}(tmax) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Alors je dis que la variation δJ de J qui provient de variations $\boldsymbol{\delta A}$, $\boldsymbol{\delta m}$ de $\boldsymbol{A}$ et $\boldsymbol{m}$ est (où on note $\boldsymbol{u^T v}$ le produit scalaire dans $\mathbb{R}^2$ de deux vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$) :

$$\delta J = \int_0^{tmax} \boldsymbol{T_*}(t)^T((\boldsymbol{\delta A})\boldsymbol{T}(t) + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{m}}(t,\boldsymbol{m})\boldsymbol{\delta m})dt,$$

d'où l'expression de la dérivée partielle de J par rapport à un paramètre ξ qui peut être un des paramètres $c_{res}, c_s, r_i, r_0, r_f$:

$$\frac{\partial J}{\partial \xi} = \int_0^{tmax} \boldsymbol{T_*}(t)^T(\underbrace{\frac{\partial \boldsymbol{A}}{\partial \xi}}_{\text{matrice } 2\times 2} \boldsymbol{T}(t) + \underbrace{\frac{\partial \boldsymbol{G}}{\partial \xi}(t,\boldsymbol{m})}_{\text{vecteur de } \mathbb{R}^2})dt.$$

Preuve. On peut écrire :

$$\begin{aligned}
\delta J &= \int_0^{tmax} (T_i(t) - T_{int}^d(t))\delta T_{int}(t)dt = \int_0^{tmax} \begin{pmatrix} T_i(t) - T_{int}^d(t) \\ 0 \end{pmatrix}^T \boldsymbol{\delta T}(t)dt \\
&= \int_0^{tmax} (-\frac{d\boldsymbol{T_*}}{dt}(t) - \boldsymbol{A}^T\boldsymbol{T_*}(t))^T\boldsymbol{\delta T}(t)dt \\
&= \underbrace{-\boldsymbol{T_*}(t)^T\boldsymbol{\delta T}(t)|_{t=0}^{t=tmax}}_{0} + \int_0^{tmax} (\boldsymbol{T_*}(t)^T\frac{d\boldsymbol{\delta T}}{dt}(t) - \underbrace{(\boldsymbol{A}^T\boldsymbol{T_*}(t))^T\boldsymbol{\delta T}(t)}_{\boldsymbol{T_*}(t)^T\boldsymbol{A}\boldsymbol{\delta T}(t)}dt \\
&= \int_0^{tmax} \boldsymbol{T_*}(t)^T(\frac{d\boldsymbol{\delta T}}{dt}(t) - \boldsymbol{A}\boldsymbol{\delta T}(t))dt = \int_0^{tmax} \boldsymbol{T_*}(t)^T((\boldsymbol{\delta A})\boldsymbol{T}(t) + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{m}}(t,\boldsymbol{m})\boldsymbol{\delta m})dt,
\end{aligned}$$

d'où le résultat.

On peut aussi procéder de manière un peu plus "mathématique" en rappelant d'abord que pour minimiser une fonction $\boldsymbol{x} \to f(\boldsymbol{x})$ sous la contrainte $\boldsymbol{c}(\boldsymbol{x}) = 0$ ($f(\boldsymbol{x})$ est un scalaire mais $\boldsymbol{x}$ et $\boldsymbol{c}$ peuvent être de dimensions quelconques et pas forcément les mêmes), on considère le lagrangien (méthode des multiplicateurs de Lagrange) $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T\boldsymbol{c}(\boldsymbol{x})$ ($\boldsymbol{\lambda}$ est le vecteur des multiplicateurs de Lagrange) et on démontre que l'optimum est à trouver parmi les solutions $(\boldsymbol{x}, \boldsymbol{\lambda})$ de $\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{\lambda}) = 0$, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = 0$ (conditions nécessaires d'optimum : autant d'équations que d'inconnues). Dans le cas de contraintes inégalités $\boldsymbol{h}(\boldsymbol{x}) \ge 0$ c'est un peu différent mais les conditions sont connues (conditions de Karush, Kuhn, Tucker).

Ici il y a un continuum de contraintes $\frac{d\boldsymbol{T}}{dt}(t) = \boldsymbol{A}\boldsymbol{T}(t) + \boldsymbol{G}(t,\boldsymbol{m})$ pour $0 \le t \le tmax$, $\boldsymbol{T}(0) = \begin{pmatrix} T_{int}(0) \\ T_s(0) \end{pmatrix} = \begin{pmatrix} T_{int}^{ini} \\ T_s^{ini} \end{pmatrix}$ et comme Lagrangien il est "naturel" de considérer [3] la quantité $\mathcal{L}(\boldsymbol{T}, \boldsymbol{T_*}) = J(\boldsymbol{T}(\boldsymbol{m})) - \int_0^{tmax} \boldsymbol{T_*}(t)^T(\frac{d\boldsymbol{T}}{dt}(t,\boldsymbol{m}) - \boldsymbol{A}(\boldsymbol{m})\boldsymbol{T}(t,\boldsymbol{m}) - \boldsymbol{G}(t,\boldsymbol{m}))dt$ (ainsi $\boldsymbol{T_*(t)}$ est un multiplicateur de Lagrange) et plutôt que d'essayer d'écrire les conditions nécessaires d'optimalité on remarque que $\delta\mathcal{L} = \delta J$ si le continuum de contraintes est satisfait. Alors on exprime $\delta\mathcal{L}$ lorsque $\boldsymbol{A}$ et $\boldsymbol{m}$ varient de $\boldsymbol{\delta A}$ et $\boldsymbol{\delta m}$ ce qui provoque une variation $\boldsymbol{\delta T}$ de $\boldsymbol{T}$ et on s'arrange pour supprimer le terme en $\boldsymbol{\delta T}$ par un choix judicieux de $\boldsymbol{T_*}$. On obtient ainsi le gradient de J en fonction des paramètres puisque $\mathcal{L} = J$. Le mieux est de compléter soi-même le calcul :

$$\begin{aligned}
\delta\mathcal{L} = \delta J &= \int_0^{tmax} \begin{pmatrix} T_{int}(t) - T_s(t) \\ 0 \end{pmatrix}^T \boldsymbol{\delta T}(t)dt - \int_0^{tmax} \boldsymbol{T_*}(t)^T(\frac{d\boldsymbol{\delta T}}{dt}(t) - \boldsymbol{A}\boldsymbol{\delta T}(t) - \boldsymbol{(\delta A)T}(t) - \frac{\partial \boldsymbol{G}(t,\boldsymbol{m})}{\partial \boldsymbol{m}}\boldsymbol{\delta m})dt \\
&= -\boldsymbol{T_*}(t)\boldsymbol{\delta T}(t)|_{t=0}^{t=tmax} \\
&+ \int_0^{tmax} ((\begin{pmatrix} T_{int}(t) - T_s(t) \\ 0 \end{pmatrix}^T \boldsymbol{\delta T}(t) + \frac{d\boldsymbol{T_*}}{dt}(t)^T\boldsymbol{\delta T}(t) + \boldsymbol{T_*}(t)^T(\boldsymbol{A}\boldsymbol{\delta T}(t) + \boldsymbol{(\delta A)T}(t) + \frac{\partial \boldsymbol{G}(t,\boldsymbol{m})}{\partial \boldsymbol{m}}\boldsymbol{\delta m})dt \\
&= ...
\end{aligned}$$

3. On fait abstraction de la condition initiale pour simplifier, mais on pourrait ajouter des multiplicateurs adéquats.

et en complétant le calcul on aboutit à l'équation différentielle satisfaite par l'état adjoint. Ainsi l'équation satisfaite par l'état adjoint ne "sort pas du chapeau" (en automatique, dans les problèmes de commande optimale on trouve des concepts analogues que je n'ai jamais manipulés mais qui m'intéressent).

Il faut compléter cette étude par plusieurs points :

d'abord on optimise une quantité dont l'ordre de grandeur doit rester raisonnable par exemple $J = \frac{1}{tmax}\int_0^{tmax}(T_i(t) - T_{int}^d(t))dt$ et on fournit à l'optimiseur J le gradient de J par rapport à des paramètres "raisonnables" par exemple en les normalisant de sorte que leurs ordres de grandeur soient 1 ;

ensuite il faut pouvoir contrôler la précision du gradient par rapport à ces paramètres normalisés en comparant (expérimentalement) l'évaluation du gradient avec $\frac{J(paramètre+petite\,variation)-J(paramètre)}{petite\,variation}$ (avoir 3 à 4 chiffres significatifs de commun est quasiment indispensable avec J et des paramètres de valeurs "raisonnables") ;

ensuite il n'y a aucune raison pour que le gradient obtenu par discrétisation de l'état adjoint soit égal au gradient du problème discrétisé, donc ayant choisi une discrétisation de l'équation différentielle $\frac{d\boldsymbol{T}}{dt}(t) = \boldsymbol{A}(\boldsymbol{m})\boldsymbol{T}(t) + \boldsymbol{G}(t, \boldsymbol{m})$ par exemple (Crank-Nicholson) $\frac{\boldsymbol{T}^{n+1}-\boldsymbol{T}^n}{\Delta t} = \boldsymbol{A}(\boldsymbol{m})(\frac{\boldsymbol{T}^{n+1}+\boldsymbol{T}^n}{\Delta t}) + \frac{1}{2}(\boldsymbol{G}(t^n, \boldsymbol{m}) + \boldsymbol{G}(t^{n+1}, \boldsymbol{m}))$ il faut refaire les calculs en discret pour obtenir l'état adjoint du problème discret ;

personnellement j'aimerais bien d'abord piger ces modèles R3C2 et autres [4] et manipuler un peu en octave, matlab et python (il y a de quoi fabriquer de beaux exos).

4. J'ai récupéré sur internet (le fichier s'appelle AHTTv500.pdf) un cours du MIT, le livre : J.H. Lienhard IV, J.H. Lienhard, 2019, A Heat Transfer Texbook (fifth edition), Phlogiston Press, Cambridge Massachusetts qui en parle un peu (voir pages 63 et 196).