Mini_ker, adjoint et sensibilités, un résumé*

Juin 2005

Avertissement *Ce texte prend appui sur les articles fondateurs de Cacuci, en omettant ses aspects de rigueur mathématique, et le lecteur est prié de se rapporter aux articles originaux pour ce qui concerne les ensembles supports et la possibilité de définir l'opérateur adjoint. On a aussi simplifié les notations en perdant la distinction entre les divers produits scalaires nécessaires, car il est facile d'en remonter ls signification suivant les opérandes. On a donc indistinctement utilisé la notation "bra-ket" de Dirac. Par rapport à Cacuci, on a étendu son vecteur de paramètres aux transferts du TEF, et on a introduit un vecteur explicite de commande pour rendre les applications plus souples.*

1 Le problème et sa résolution par l'adjoint

On a un modèle écrit sous TEF, dépendant d'une loi de commande $h(t)$, et une trajectoire $\mathcal{T}$, ayant - une fois - initialisé une loi de commande. On a un critère à maximiser :

$$J = \psi(\eta(T), h(T)) + \int_0^T l(\eta(\tau), h(\tau)\, d\tau \tag{1}$$

A une trajectoire proche, provoquée par exemple par une modification $u(t)$ de h, correspond une variation de la fonctionnelle J :

$$dJ = \psi_x(\eta(T), h(T))\, x(T) + \int_0^T \{l_x(\eta(\tau), h(\tau))\, x(\tau) + l_u(\eta(\tau), h(\tau))\, u(\tau)\}\, d\tau \tag{2}$$

on a noté x la variation $\Delta\eta$ de η et on n'a pas précisé les produits scalaires implicites. Ce qu'on veut, c'est éliminer x de cette variation, de manière à ce qu'on obtienne $< d_u J \mid$, le gradient qui permet d'attaquer les algorithmes divers de descente pour résoudre notre problème de commande optimale.

Ce problème est traité classiquement par maximisation de J sous la contrainte des équations du SLTC, avec l'introduction d'un vecteur de Lagrange λ, et du Hamiltonien du système. Nous suivons ici l'approche de Cacucci, d'abord sans introduire explicitement les transferts, pour ne pas alourdir la présentation. Dans la section 2, on étendra au TEF.

Le système d'équations du SLTC est :

$$\partial_t x(t) \;=\; A(t)\, x(t) + B(t)\, u(t) \tag{3}$$

*Al1, Décembre 2003 ; révisions Juillet 2004 et Janvier 2005 ; section adjoint-propagateur et quadruple comparaison, de Juin 2005

On choisit un produit scalaire - qui en général comprendra une intégration temporelle - et on introduit une variable adjointe $v(t)$, vecteur ligne, par laquelle on multiplie scalairement (3) en isolant la partie commande à droite de l'égalité :

$$< v(t) \mid \partial_t x(t) > - < v(t) \mid A(t) \mid x(t) > \;=\; < v(t) \mid B(t) \mid u(t) > \qquad (4)$$

Précision mathématique[1] *:* le système (4) est équivalent à (3) lorsqu'il est pris $\forall v(t)$ (Lax-Milgram). Ainsi, pour toute fonction $v(t)$, l'égalité (4) est vérifiée. On est donc en mesure de définir un nouveau problème dont $v(t)$ est solution : ce système restera indépendant de $x(t)$

$$- < \partial_t v(t) \mid x(t) > - < v(t) \mid A(t) \mid x(t) > \;=\; < l_x(\eta(t), h(t)) \mid x(t) > \qquad (5)$$

cette fois ci pour tout x. Ce problème définit le système adjoint à (3) pour la fonction de coût J.

En effectuant une intégration par partie dans le premier produit scalaire[2], on obtient :

$$\begin{aligned} & < v(t) \mid \partial_t x(t) > \;-\; < v(t) \mid A(t) \mid x(t) > \\ = & < l_x(\eta(t) \mid h(t)), x(t) > \;-\; < v(T) \mid x(T) > + < v(0) \mid x(0) > \end{aligned} \qquad (6)$$

pour tout v, ce qui permet d'identifier le terme de droite de (5) avec celui de (6). De cette façon, on élimine le terme dépendant de $x(t)$ dans la variation dJ par une expression dépendant de $v(t)$ obtenue indépendamment de $x(t)$. Il suffit de résoudre le système adjoint suivant :

$$- < \partial_t v(t) \mid \;=\; < v(t) \mid A(t) + < l_x(\eta(t), h(t)) \mid \qquad (7)$$

les termes de borne de l'intégration par parties sont choisis en fonction de la forme de dJ obtenue par remplacement dans l'intégrande du terme dépendant de x :

$$\begin{aligned} dJ \;=\; & < \psi_x(\eta(T), h(T)) \mid x(T) > - < v(T) \mid x(T) > \\ & + \int_0^T \{ < v(\tau) \mid B(\tau) \mid u(\tau) > + < l_h(\eta(\tau), h(\tau)) \mid u(\tau) > \} d\tau \\ & + < v(0) \mid x(0) > \end{aligned} \qquad (8)$$

ainsi, en prenant comme conditions terminales $v(T) = \psi_x(T)$, on élimine la dépendance en $x(T)$. $v(0)$ est déterminé par la résolution de l'adjoint, et $x(0)$ est connu.

In fine, on obtient le gradient voulu, qui ne dépend que de la correction $u(t)$ à appliquer à la loi de commande, soit :

$$dJ = < v(0) \mid x(0) > + \int_0^T \{ < v(\tau) \mid B(\tau) \mid u(\tau) > + < l_h[\eta(\tau), h(\tau)] \mid u(\tau) > \} \, d\tau \qquad (9)$$

d'où on déduit le gradient recherché :

$$\nabla_u J(\tau) = < v(\tau) \mid B(\tau) + < l_h[\eta(\tau), h(\tau)] \mid \qquad (10)$$

[1] cette partie a été corrigée par stepH.

[2] c'est là que l'on restreind la généralité de la vision de Cacuci qui envisage des produits scalaires plus généraux.

Pour ceux qui n'auraient pas pris le temps de tout suivre, cette forme est en effet pertinente en ce que les variables adjointes répercutent tout l'effet du changement de la commande sur le segment de trajectoire postérieur à t (de t à T), c'est le fondement du théorème du maximum de Pontryagine.

Le théorème du **maximum de Pontryagine** dit en effet que pour la trajectoire optimale, pour tout t, le Hamiltonien du système est maximum en la commande h pour η et v déterminés par le système aller et retour[1] :

$$H(\eta, v, h) = < v \mid g(\eta, h) > + l(\eta, h) \tag{11}$$

à chaque instant, ce qui peut être utile pour déterminer $u(t)$, par exemple avec une méthode de « programmation dynamique ». Globalement donc, avec l'adjoint, on a simplifié un problème de recherche d'extremum. Au départ, on travaille dans un espace de dimension celle du vecteur commande fois le nombre de pas d'intégration (*npas*), puisque la variation de $u(t)$ à l'instant t se répercute sur les instants ultérieurs. Après passage de l'adjoint, on travaille indépendamment dans *npas* espaces, car alors, en supposant que l'on soit proche de l'optimum, on peut optimiser $u(t)$ dans l'intervalle $[t, t + \delta t]$ sans remettre en question l'optimalité de $u(t)$ en dehors de cet intervalle.

2 Extension au TEF

Avec des signes relativement non conformes aux divers formulaires éventuellement entre eux contradictoires, le SLTC à la TEF peut s'écrire[2] :

$$\begin{bmatrix} \partial_t \cdot - A & -B \\ -C^\dagger & 1 - D \end{bmatrix} \quad \begin{bmatrix} \Delta\eta \\ \Delta\varphi \end{bmatrix} \quad = \quad \begin{bmatrix} B^u u^\eta \\ D^u u^\varphi \end{bmatrix} \tag{12}$$

Les matrices prenant leur définition la plus simple des Jacobiennes du TEF, c'est-à-dire que si :

$$\begin{cases} \partial_t \eta(t) = g(\eta(t), \varphi(t), h^\eta(t)) \\ \varphi(t) = f(\eta(t), \varphi(t), h^\varphi(t)) \end{cases}$$

on a défini :

$$\begin{aligned} A = \partial_\eta g \quad &; \quad B = \partial_\varphi g; \\ C^\dagger = \partial_\eta f \quad &; \quad D = \partial_\varphi f; \\ B^h = \partial_h g \quad &; \quad D^h = \partial_h f \end{aligned} \tag{13}$$

Le système direct sous TEF correspondant est ainsi :

$$\partial_t \delta\eta(\tau) = g_0(t) + A(t)\,\delta\eta(\tau) + B(t)\,\delta\varphi(\tau) + B^h(t)\,\delta h^\eta(\tau) \tag{14}$$

$$\delta\varphi(\tau) = C^\dagger(t)\,\delta\eta(\tau) + D(t)\,\delta\varphi(\tau) + D^h(t)\delta h^\varphi(\tau) \tag{15}$$

[1] le cas où l'instant terminal T est aussi à optimiser revient à ajouter la condition $H(T) = 0$, cf annexe

[2] je ne trouve pas de raison pour ne pas se restreindre à B^u et D^u diagonales, car on a introduit ce vecteur pour simplifier l'utilisation de Mini_ker pour la commande optimale. Les cas plus complexes peuvent être traités par l'intermédiaires de commandes-transferts.

avec $t \leq \tau < t + \delta t$, $\delta t, \delta t$ étant le pas de temps local de l'intégration numérique. et sa forme matricielle intégrée pour la trajectoire est :

$$\begin{bmatrix} I - \frac{\delta t}{2} A & -\frac{\delta t}{2} B \\ -C^{\dagger} & 1 - D \end{bmatrix} \quad \begin{bmatrix} \delta\eta \\ \delta\varphi \end{bmatrix} \quad = \quad \begin{bmatrix} \delta t g_0 + B^u u^{\eta}(t + \frac{\delta t}{2}) \\ D^u u^{\varphi}(t + \frac{\delta t}{2}) \end{bmatrix} \tag{16}$$

C'est la forme que j'ai utilisée pour le Mini_ker [1], avec stockage des Jacobiennes, parce qu'en pratique on en a de multiples usages (calcul de la matrice d'avance d'état, analyse en Borel,...).

En ce qui concerne le SLTC et son adjoint, on y a étendu l'application de la commande aux transferts par souci de généralité, que l'on invoque aussi dans la fonction de coût :

$$J = \psi[\eta(T), \varphi(T), h(T)] + \int_0^T l[\eta(\tau), \varphi(T), h(\tau)]\, d\tau \tag{17}$$

où la fonction vectorielle h englobe les commandes sur les deux types de variables du TEF. Le modèle adjoint (à 12) s'écrit donc de la manière suivante :

$$\begin{bmatrix} -\partial_t \cdot - A^{\dagger} & -C \\ -B^{\dagger} & 1 - D^{\dagger} \end{bmatrix} \quad \begin{bmatrix} v \\ w \end{bmatrix} \quad = \quad \begin{bmatrix} l_x \\ l_y \end{bmatrix} \tag{18}$$

avec comme conditions initiales : $v(T) = \psi_x(T) \quad \{w(T)$ est déterminée par $v(T)\}$. Ce qui fait que le système adjoint est par définition linéarisé.

Le gradient d'intérêt se décompose en deux parties $\nabla_{ux} J$, $\nabla_{uy} J$:

$$\begin{aligned} dJ \;=\; & < v(0) \mid x(0) > \\ & + \int_0^T \{< v(\tau) \mid B^u(\tau) \mid u^x(\tau) > + < l_{ux}(\eta(\tau), h^x(\tau)) \mid u^x(\tau) >\}\, d\tau \\ & + \int_0^T \{< w(\tau) \mid D^u(\tau) \mid u^y(\tau) > + < l_{uy}(\eta(\tau), h^y(\tau)) \mid u^y(\tau) >\}\, d\tau \end{aligned} \tag{19}$$

avec des notations évidentes. Et ainsi finalement, on obtient :

$$\begin{aligned} \nabla_{ux} J(\tau, T) &= < v(\tau) \mid B^u(\tau) \quad + < l_{ux}(\eta(\tau), h^x(\tau)) \mid \\ \nabla_{uy} J(\tau, T) &= < w(\tau) \mid D^u(\tau) \quad + < l_{uy}(\eta(\tau), h^y(\tau)) \mid \end{aligned} \tag{20}$$

Le Hamiltonien correspondant peut être défini en ne conservant que les termes dépendants de h :

$$\begin{aligned} H(\eta, \varphi, v, w, h^x, h^y) \;=\; & < v \mid g(\eta, \varphi, h^x) > \\ + \;\; < w \mid f(\eta, \varphi, h^y) > \;+\; & l(\eta, \varphi, h) \end{aligned} \tag{21}$$

à chaque instant.

[1] contrairement à la version originelle dans laquelle on avait une distribution des signes et des δt un peu aléatoire.

2.1 Intégration du système adjoint

Le système (18) s'intègre comme pour le direct de t à $t+\delta t$ pour donner δv, avec la source l_x constante, comme pour les matrices, ce qui donne :

$$\begin{bmatrix} -I - \frac{\delta t}{2} A^\dagger & -\frac{\delta t}{2} C \\ -B^\dagger & 1 - D^\dagger \end{bmatrix} \quad \begin{bmatrix} \delta v \\ \delta w \end{bmatrix} \quad = \quad \begin{bmatrix} \delta t (l_x + A^\dagger v + C w) \\ l_y \end{bmatrix} \tag{22}$$

[1] Le plus simple consiste à prendre δt négatif pour obtenir l'incrément δv vers l'arrière. On résout ainsi ce système simplement par le même appel à la routine **ker** du Mini_ker en inversant les arguments B et $C^\dagger$ et avec toutes les matrices transposées ; de même, le vecteur **gamma**[2] doit être remplacé. Ici, le remplaçant de g_0 est sous forme linéaire $= A^\dagger v + Cw$, et on a un terme source l_x. On a besoin, comme pour le direct, de l'équivalent de l'équation des transferts qui n'est autre que la deuxième ligne de (18) :

$$(1 - D^\dagger) \mid w >= B^\dagger \mid v > + l_y \tag{23}$$

3 ... et Mini_ker

Mini_ker constitue une maquette logicielle destinée à développer et éprouver les méthodes exploitant le SLTC rendues accessibles en simulation par l'utilisation du TEF. La refondation du code originel établi par stb est basé sur la concision : ne pas avoir à écrire chaque formule utile et dérivable plus d'une seule fois. D'où la programmation suivante, qui se veut simple et de bon goût : entre deux mises à jour de la commande, on fait systématiquement un aller-retour dans Mini_ker.

En avant, $u(t)$ est donnée (loi ou fichier). On calcule la trajectoire de manière classique, en ajoutant comme diagnostic un calcul de J, pour vérifier l'approche à l'optimum[3]. On stocke η, φ, h. Par rapport à une trajectoire standard, on a donc des *f_set* supplémentaires[4], pour la densité de coût $l(t)$, et $\Psi(T)$.

Au retour, à partir de T, initialiser v, remplacer le calcul aller par restauration de η, φ, h et déterminer l'incrément temporel (qui a pu varier en cours de simulation aller). Même calcul des matrices[5], vecteurs donnés par (22). Calcul de w par résolution de (23). Appel de ker (le même qu'à l'aller) mais avec δt négatif, calcul et stockage des deux vecteurs $d_u J(t)$ donnés par les formules (20) et du Hamiltonien (cf fichiers générés —.data).

Phase hors Mini_ker- il reste à corriger la commande $u(t)$: à vos souhaits ! Il est clair que chacun aura à se faire la main sur des exos différents. Il y a le cas important des coûts quadratiques, qu'il faudra développer. Voir quels algorithmes de descente (par gradients conjugués ou quasi lorsqu'on peut développer la fonction de coût au deuxième ordre seulement), comment aborder les problèmes avec contrainte sur la commande etc. Une piste intéressante par son aspect généraliste pourrait

[1] Le cas du 0 de la 2ième ligne et du lien avec le pb du dfdpi a été résolu et amène à la définition du gain non-$\mathcal{L}$, cf papier Propagateurs etc.

[2] pardon pour les lecteurs étrangers au TEF, Γ est le vecteur connu à droite de l'équation de cellule du système (16).

[3] cet aller permet aussi éventuellement de calculer des sensibilités en avant.

[4] cf manuels Mini_ker.

[5] en restaurant $\varphi(t)$, on n'a plus besoin d'itérer sur la D-loop.

combiner l'adjoint et la programmation dynamique, cette dernière chose que nous avions pratiquée avec Harold Vasselin pour optimiser la gestion de stocks de froid et de chaud et ainsi climatiser à moindre frais un immeuble de bureaux au Québec (cf sa thèse). Bref, y a de quoi alimenter quelques Zircons.

Juillet 2004 : la version couplée à Minuit a été délivrée, cf exo "pousse" et le manuel du Mini_ker. Et en Juin 2005, on a aussi Lorenz 63, et les essais de stepH avec NEDyM. La conclusion de nombreux efforts de stepH avec Minuit n'est guère encouragente, et il s'avère que le pilotage de Minuit lors de son balayage d'un espace paramétrique accidenté (le modèle NEDyM comporte par construction de nombreuses poches de chaos) est difficile.

4 Mini_ker et LLSQ

[1] La difficulté pratique d'utiliser Minuit dans des cas complexes et l'intérêt de la formule de Newton (cf annexe) nous amènent à proposer d'introduire dans Mini_ker le calcul des sensibilités des variables à une liste de paramètres à déterminer. On a déjà montré comment il était possible de calculer ces sensibilités par rapport à un paramètre choisi parmi les variables Fortran des modèles. On propose ainsi une extension de ces calculs à la liste des **free-parameter :— ;** , qui seraient inclus dans la version 1.02.

[1] Linear Least SQuare fit, programme mettant en œuvre la méthode de Householder de Paris Sud Informatique.

5 la Collab

[1]Il existe déjà une certaine pratique dans la secte, on pourrait en profiter pour faire un point sur les exos / difficultés / particularités rencontrées.

Sur ce qui a été fait dans ZOOM : avec un modèle explicite des sensibilités (en avant, donc, c'est-à-dire en doublant les équations du modèle), on gère le calcul d'un écart quadratique entre modèle / mesures dans ZSTEER, et on utilise la méthode de Newton (via OLLSQ) pour déterminer la mise à jour des paramètres (on relance donc ZOOM dans l'itération de l'assimilation des données). Olivier Fudym a ainsi fitté un modèle à deux paramètres par la méthode flash. S'agissait d'un modèle de séchage de grains (ou de bois ?). JLJ (et STB ?) on dû faire un fit plus pédagogique avec une source sinusoïdale. J'ignore ce qui a été fait avec Minuit.

C'est StepH qui veut joindre son modèle de climat-carbone avec celui de Vincent Gitz qui travaille en «inter-temporel», c'est-à-dire que le modèle détermine un coefficient de propension des ménages à l'investissement au fil du temps censé maximiser leur consommation intégrée sur un horizon fixé. Il s'agit donc non pas d'un fit paramétrique, mais de la recherche d'une commande optimale, d'où ce texte. L'adjoint de Mini_ker a été validé par comparaison avec les sensibilités automatiques à la perturbation d'une condition initiale, et les deux calculs précédents permettent bien de retrouver la matrice de propagation du système déterminée indépendamment (cf papier SLTC). Ceci valide donc la partie matricielle, les termes sources provenant de la commande et de l'intégrande de la fonction de coût ont été validé avec la recherche d'optimum d'une loi de commande pour le cas d'un exercice menant à une solution analytique (exo "pousse"). On obtient une précision de l'ordre du % seulement, et ce problème numérique est bien connu des fans du fit : d'où la version 1.01 de Mini_ker en double précision, particulièrement utile pour ce cas d'application.

[1]**retour au texte plus ancien**

ANNEXE : Le cas du FIT

Rappelons la méthode de Newton[1] déjà largement utilisée avec ZOOM (cf Fudym, et JLJ et al.) pour référence au problème de l'utilisation de l'adjoint dans le même contexte. On dispose d'une série de mesures sur une séquence temporelle, rassemblées dans le vecteur μ. Par ailleurs, et aux mêmes instants pour simplifier, un modèle fournit les mêmes observables M. Alors, au minimum de l'écart quadratique[2]

$$Q^2 =< (M-\mu)^2 > \tag{24}$$

une variation $d\varpi$ des paramètres est liée à la variation du modèle par :

$$\mid d\varpi >= -\left[\frac{\partial M}{\partial \varpi}^{\dagger}\frac{\partial M}{\partial \varpi}\right]^{-1}\frac{\partial M}{\partial \varpi}^{\dagger} \mid M-\mu > \tag{25}$$

On va établir cette formule car cela nous indiquera le lien avec l'adjoint.

$$Q^2(\varpi+d\varpi) =< (M-\mu)^2 > +2< (M-\mu) \mid \frac{\partial M}{\partial \varpi}d\varpi > + < d\varpi\frac{\partial M}{\partial \varpi}^{\dagger}\frac{\partial M}{\partial \varpi}d\varpi > \tag{26}$$

par le fait que la matrice du terme du second degré en $d\varpi$ est symétrique définie positive, le minimum est atteint pour $d\varpi$ donné par 25.

En effet, avec les termes du développement au second ordre de Q^2 :

$$Q^2(\varpi+d\varpi) = Q^2 + < \nabla_{\varpi}Q^2 \mid d\varpi > + \frac{1}{2} < d\varpi \mid \frac{\partial^2 Q^2}{\partial \varpi^2} \mid d\varpi > \tag{27}$$

on a explicitement :

$$\begin{aligned} \nabla_{\varpi}Q^2 &= 2 < M-\mu \mid \frac{\partial M}{\partial \varpi} \\ \frac{\partial^2 Q^2}{\partial \varpi^2} &= 2\frac{\partial M}{\partial \varpi}^{\dagger}\frac{\partial M}{\partial \varpi} \end{aligned} \tag{28}$$

d'où $d\varpi$ au minimum.

Utilisation de Minuit[3]

Ce que fournit l'adjoint, c'est le premier de ces termes ($\nabla_{\varpi}Q^2$). On constate ainsi dans ce cas le travail qu'il reste à faire pour la recherche de l'extremum, avec pour ce cas utilisation d'une méthode de gradients conjugués.

O.Fudym a pour une raison que j'ai oubliée, fait appel à MINUIT[4]. Comme le modèle est appelé à partir de MINUIT, Olivier a du faire des acrobaties avec des shells pour relancer ZOOM avec les paramètres récursivement modifiés.

[1] que certains dénomment méthode de Gauss

[2] on a supposé encore pour simplifier que les mesures étaient indépendantes et de même incertitude.

[3] ©CERN.

[4] logiciel super-sophistiqué du CERN, méthode à métrique variable, premières descentes par symplexe puis gradients conjugués.

La solution boite noire par utilisation de Minuit[1] transforme le programme **Principal** de Mini_ker en subroutine pilotée par **DSDQ**, le programme gérant la recherche d'optimum, avec une subroutine à l'interface **FMINI**. Un manuel de Minuit est disponible en ligne au CERN[2].

Utilisation de la formule de Newton[3]

Lors du traitement de ce problème de détermination paramétrique avec ZOOM, on a jusqu'à présent construit explicitement le modèle de $\frac{\partial M}{\partial \varpi}$ dans chaque processeur de cellule et transfert, stocké tous les éléments le long de la trajectoire et appliqué la formule de Newton 25 par appel à la routine OLLSQ[4]. Avec un appel à LLSQ après ZSTEER en fin de trajectoire, on détermine les nouveaux paramètres et on boucle au début de ZINIT.

La version 1.02 de Mini_ker va permettre d'utiliser la formule de Newton beaucoup plus directement. On prévoit en effet que la **free-parameters'** liste provoque le calcul de la matrice $\frac{\partial M}{\partial \varpi}$ à chaque pas de temps du calcul.

Lors d'une rencontre avec un instant (i) de mesure fournissant $ket\mu_i$, signalé par **ZObs True**, on incrémente le calcul de $d\varpi$ (variable **ddpi**, fichier de sortie **ddpi.data**) par la formule (25) que l'on décline sur chaque instant de mesure (i) avec l'hypothèse que la matrice d'erreur de mesure σ_i est indépendante d'une prise de mesure à l'autre :

$$\boxed{\mid d\varpi > = \sum_i dd\varpi_i = -\sum_i \left[\frac{\partial M}{\partial \varpi}^{\dagger} \sigma_i^{-1} \frac{\partial M}{\partial \varpi}\right]_i^{-1} \frac{\partial M}{\partial \varpi}^{\dagger} \sigma_i^{-1} \mid M_i - \mu_i >} \qquad (29)$$

On peut éventuellement prévoir de ne commencer ces calculs que lorsque la dimension du vecteur des mesures dépasse suffisammment le nombre de paramètres. Dans la méthode de Householder, on minimise en pratique la norme euclidienne de $\frac{\partial M}{\partial \varpi}\sigma_i^{-\frac{1}{2}} + \mid M_i - \mu_i >$ (cf 26). Les cas les plus habituellement traités possèdent une simple erreur de mesure $\sigma^{\frac{1}{2}}$ diagonale.

---

[1] option **MinuiK** ou **Monitor.**

[2] voir aussi manuels Mini_ker sur la page web de **ZOOM**

[3] version 1.02

[4] qui fournit également la matrice de covariance des paramètres. Cette routine est issue d'un agorithme de Golub et programmée par l'équipe du PSI (Paris Sud Informatique de l'université d'Orsay) Jean Rauel et Germain Bonne pour l'UNIVAC – LLSQ et MLLSQ. Le nom de OLLSQ correspond à la version jyg, avec la convention "raw-wise" adoptée pour cause de compatibilité avec la CERLIB et ZOOM.