# Localisation robuste et dénombrement de valeurs propres

## Robust localization and counting of eigenvalues

L. B. NGUENANG * — E. KAMGNIA * — B. PHILIPPE **

* Département d'Informatique, Université de Yaoundé I, Yaoundé, Cameroun.
Emails: lnguenang@yahoo.fr, erkamgnia@yahoo.fr.

** INRIA Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes cedex, France.
Email: Bernard.Philippe@inria.fr

Ce travail a été réalisé dans l'équipe MOMAPPLI du LIRIMA.
This work was pursued within the team MOMAPPLI of the LIRIMA.
LIRIMA=Laboratoire International de Recherche en Informatique et Mathématiques en Afrique, http://lirima.org/

**RÉSUMÉ.** L'article se consacre à la localisation de valeurs propres pour une grande matrice creuse, a priori non symétrique, dans un domaine du plan complexe. Il combine deux notions déjà étudiées. La première précise l'effet de perturbations sur la matrice par la définition de $\varepsilon$-spectre ou pseudospectre. La deuxième consiste à dénombrer les valeurs propres entourées par une courbe a priori donnée dans le plan complexe. A partir de travaux antérieurs, on combine ici les deux approches avec l'objectif de mettre en commun les factorisations LU de la résolvante nécessaire aux deux approches et d'en diminuer le nombre. Les codes obtenus sont parallélisés.

**ABSTRACT.** This article deals with the localization of eigenvalues of a large sparse and not necessarily symmetric matrix in a domain of the complex plane. It combines two studies carried out earlier. The first work deals with the effect of applying small perturbations on a matrix, and referred to as $\varepsilon$-spectrum or pseudospectrum. The second study describes a procedure for counting the number of eigenvalues of a matrix in a region of the complex plain surrounded by a closed curve. The two methods are combined in order to share the LU factorization of the resolvent, that intervenes in the two methods, so as to reduce the cost. The codes obtained are parallelized.

**MOTS-CLÉS :** Pseudospectre, dénombrement de valeurs propres, déterminant, valeur singulière minimale, orbite,trace, client-serveur, Maître, Travailleurs, Accélération, Efficacité

**KEYWORDS :** Pseudospectrum,eigenvalue counting, determinant, minimum singular value, orbit, trace, Client-server,Master, Workers, speedup, efficiency

## 1. Sensibilité aux perturbations des valeurs propres d'une matrice

La localisation de valeurs propres pour une grande matrice creuse dans un domaine du plan complexe est une tâche délicate, spécialement lorsque la matrice n'est pas symétrique. En effet, pour une matrice symétrique $A$ et pour toute matrice symétrique de perturbation $\Delta$, on est assuré que pour toute valeur propre $\lambda$ de $A$, il existe une valeur propre $\tilde{\lambda}$ de $A+\Delta$ telle que

$$|\lambda - \tilde{\lambda}| \leq \|\Delta\|.$$

où la norme matricielle considérée est la norme $L_2$, comme elle l'est dans tout l'article. Par contre si la matrice $A$ n'est plus supposée symétrique, la variation des valeurs propres dépend de la proximité d'une matrice non diagonalisable dans le voisinage de $A$ puisque la perturbation d'un bloc de Jordan de dimension $k$ éclate les valeurs propres en fonction de la racine $k$-ième de la perturbation.

En effet si $A = XJX^{-1}$ est la forme de Jordan et $m$ la dimension du plus grand bloc de Jordan pour $\lambda \in \Lambda(A)$ spectre de $A$, alors pour $\Delta$ il existe $\tilde{\lambda} \in \Lambda(A+\Delta)$ tel que [10] pp.174 :

$$\frac{|\lambda - \tilde{\lambda}|}{\left(1 + |\lambda - \tilde{\lambda}| + \cdots + |\lambda - \tilde{\lambda}|^{m-1}\right)^{1/m}} \leq \left(\|X^{-1}\| \, \|X\| \, \|\Delta\|\right)^{1/m}.$$

Afin de prendre en compte l'effet de perturbations sur la matrice, la notion de $\varepsilon$-spectre ou pseudo-spectre d'une matrice $A \in \mathbb{C}^{n\times n}$ a été définie indépendamment mais simultanément par Godunov [3] et Trefethen [11].

Soit $A \in \mathbb{C}^{n\times n}$, et $\varepsilon \geq 0$. L' $\varepsilon-$spectre d'une matrice $A$ est défini par :

$$\begin{aligned}
\Lambda_\varepsilon(A) &= \{z \in \mathbb{C} : \exists \Delta \in \mathbb{C}^{n\times n} \; avec \; \|\Delta\| \leq \varepsilon \text{ et z est une valeur propre de } A+\Delta\} && [1] \\
&= \{z \in \mathbb{C} : \|(A - zI)^{-1}\| \geq \frac{1}{\varepsilon}\} && [2] \\
&= \{z \in \mathbb{C} : \sigma_{\min}(A - zI) \leq \varepsilon\}. && [3]
\end{aligned}$$

où $\sigma_{min}(A - zI)$ est la plus petite valeur singulière de $(A - zI)$. Construire un $\varepsilon$-spectre d'une matrice A consiste donc à déterminer une courbe de niveau $(\Gamma)$ de la fonction

$$z \in \Omega \subset \mathbb{C} \longrightarrow \sigma_{\min}(A - zI),$$

où $\Omega$ est la région en considération dans $\mathbb{C}$.

On peut aussi développer une approche alternative : étant donnée une courbe $(\Gamma)$ dans le plan complexe, il s'agit de compter le nombre de valeurs propres de $A$ entourées par $(\Gamma)$. Cela revient à évaluer l'intégrale

$$\frac{1}{2i\pi} \int_\Gamma \frac{d}{dz} \log\det(zI - A) dz.$$

Ce problème a été étudié dans [1], [2], [9] où plusieurs méthodes d'intégration étaient proposées. Plus récemment, dans [5] le contrôle du pas d'intégration a fait l'objet d'une étude fine. L'objectif est maintenant de combiner les deux approches pour la localisation et le dénombrement de valeurs propres d'une matrice A.

Il existe deux types d'approche pour la construction d'un pseudo-spectre : la méthode de maillage uniforme et les méthodes de suivi de contour. Dans cet article nous utiliserons une méthode de suivi de contour appelée méthode PAT.

## 2. La méthode de suivi de contour à l'aide de triangles : Méthode PAT

Cette méthode est présentée dans [6, 7]. On construit une séquence de triangles équilatéraux adjacents qui recouvrent le contour. On détermine un point du contour par bissection de chaque coté du triangle qui intersecte le contour (voir la figure 1)

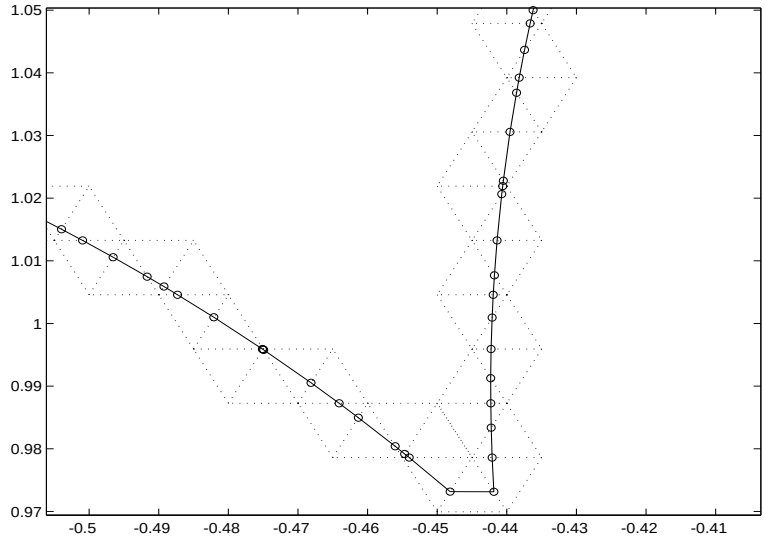


**Figure 1.** PAT *: Exemple de suivi de contour par des triangles équilatéraux.*

### 2.1. Définition du maillage

La définition d'un maillage adapté à la ligne de niveau repose sur la recherche de deux points ; l'un $z_i$, intérieur (bord compris) et l'autre $z_e$, extérieur au pseudo spectre $\Lambda_\varepsilon(A) = \{z \in \mathbb{C} | \sigma_{\min}(zI - A) \leq \varepsilon\}$ tels que $|z_e - z_i| = \tau$, où $\tau$ désigne la résolution du maillage.
Ces deux points sont dits $\varepsilon$-séparés et permettent de générer un maillage triangulaire $S(z_i, z_e)$ où $T_0 = \{S_{0,0}, S_{1,0}, S_{0,1}\}$ constitue le premier triangle. Un moyen pour trouver $z_i$ et $z_e$ consiste à partir d'une estimation $z_0$ d'une valeur propre proche d'un point de référence $z_{ref}$ et à construire la suite

$$z_{k+1} = z_0 + 2^k \tau e^{i\theta} \text{ pour un angle } \theta \text{ donné.}$$

Étant donnés un point intérieur $z_i$ et un point extérieur $z_e$, le réseau uniforme $\mathscr{S}$ de triangles équilatéraux

$$S(z_i, z_e) = \{S_{kl} = z_i + k\,(z_e - z_i) + l\,(z_e - z_i)e^{i\frac{\pi}{3}},\ (k,l) \in \mathbb{Z}^2\}$$

définit un maillage uniforme de points vérifiant : $|S_{k,l+1} - S_{k,l}| = |S_{k+1,l} - S_{k,l}| = \tau$, où $\tau = |z_i - z_e|$, pour tout $(k,l) \in \mathbb{Z}^2$. Cette grille permet de définir deux classes de triangles $T_{kl} = \{S_{k,l}, S_{k+1,l}, S_{k,l+1}\}$ et $\tilde{T}_{kl} = \{S_{k,l}, S_{k+1,l}, S_{k+1,l-1}\}$ qui sont liées par les rotations $R(S_{k,l}, \frac{\pi}{3})$ et $R'(S_{k,l}, -\frac{\pi}{3})$ (voir la figure 2).

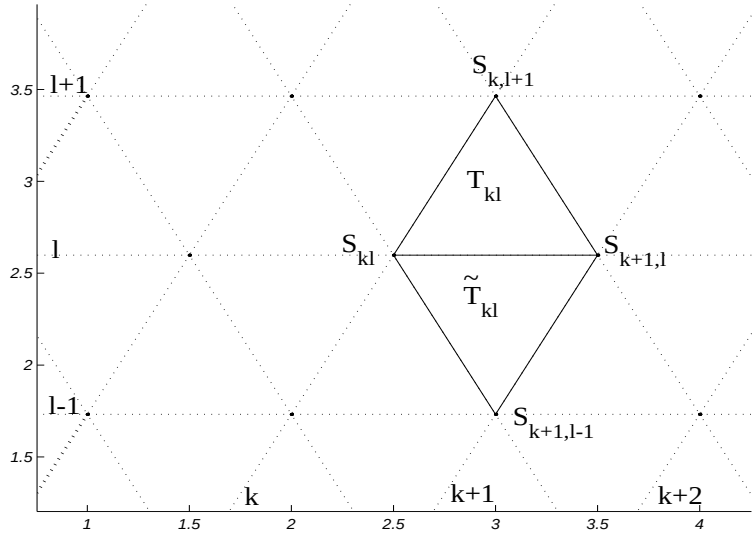


**Figure 2.** PAT *: Le réseau des triangles équilatéraux.*

## 2.2. Construction d'une orbite

Soit $\Omega(z_i, z_e)$ l'ensemble des triangles équilatéraux défini à partir d'un point intérieur $z_i$ et d'un point extérieur $z_e$ ; soit $\mathscr{O}_L$ la partie de $\Omega(z_i, z_e)$ constituée des triangles qui ont au moins deux sommets $\varepsilon$-séparés ; $\mathscr{O}_L$ est un ensemble fini. Pour chaque triangle $T \in \mathscr{O}_L$, il n'y a qu'un sommet qui soit seul de sa catégorie(point intérieur ou point extérieur). Ce point est appelé le *pivot de* $T$ et est noté $p(T)$. On définit alors la bijection $F$ de l'ensemble $\mathscr{O}_L$ dans lui-même par

$$\begin{cases} F(T) = \mathscr{R}(p(T), \theta)\ T, \\ \text{où } \theta = \frac{\pi}{3} \text{ si } p(T) \text{ est intérieur, ou } \theta = -\frac{\pi}{3}, \text{ sinon} \\ \text{ et où } \mathscr{R}(z, \theta) \text{ est la rotation de centre } z \in \mathbb{C} \text{ et d'angle } \theta. \end{cases}$$

La transformation F est illustrée par la figure 3.

En partant d'un triangle quelconque $T_0 \in \mathscr{O}_L$, on peut construire la $F-$orbite de $T_0$ :

$$O(T_0) = \{T_k \equiv F^k(T_0), k \in \mathbb{Z}\} \subset \mathscr{O}_L.$$

Puisque tous les triangles construits intersectent la ligne de niveau, l'ensemble est situé dans une partie bornée du plan complexe. Il s'en suit qu'il existe un entier $n$ tel que $F^n(T) = T$ et par conséquent la suite $(T_k)_{k\in\mathbb{Z}}$, est périodique ; comme nous utilisons des coordonnées entières pour identifier les points $S_{kl}$, le test $F^n(T) = T$ pourra être implémenté de manière sûre. Comme la somme des angles des rotations construisant une orbite doit être nulle à $2\pi$ près, le nombre de triangles distincts d'une orbite est toujours pair.

## 2.3. L'algorithme PAT

L'algorithme est décrit dans l'algorithme 1 et la méthode PAT est définie en trois étapes :

**ETAPE I : construction du triangle initial.** Choisir $z_0$ point intérieur. Cela est réalisé en calculant une valeur propre de la matrice $A$ dans le voisinage d'une valeur de référence $z_{ref}$ fournie par l'utilsateur. A partir des paramètres $\tau > 0$ et $\theta$, on construit

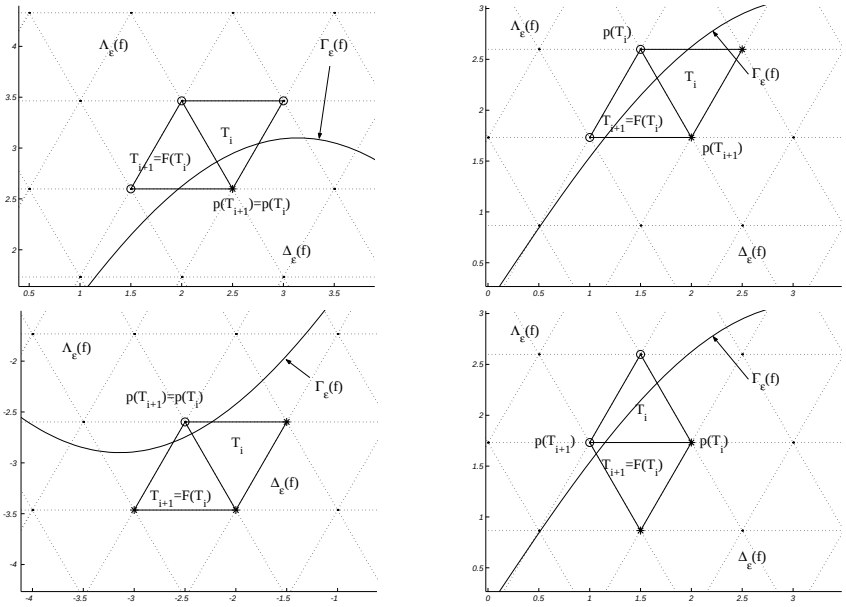


**Figure 3.** PAT *: La transformation F.*

**Algorithm 1** PAT : suivi d'une ligne de niveau $\partial\Lambda_\varepsilon(A)$ à l'aide de triangles.

**Require:** $A \in \mathbb{C}^{n\times n}$, $\varepsilon > 0$, $\tau > 0$ la taille des mailles du réseau, $\theta$ l'inclinaison du réseau, $z_0$ un point intérieur ($\sigma_{\min}(A - z_0 I) \leq \varepsilon$).

**Ensure:** Un triangle initial $T_0 = (z_i, z_e, z') \in \mathscr{O}_L$, son orbite $O(T_0)$, une ligne polygonale de $N$ sommets ($N = \text{card}(O(T_0))$) inscrite dans le bord $\partial\Lambda_\varepsilon(A)$ du pseudo-spectre $\Lambda_\varepsilon(A)$.

$h = \tau e^{i\theta}$ ; $Z = z_0 + h$ ; $I_t = 0$ ;{*Recherche d'un point extérieur*}
**while** $\sigma_{\min}(A - z_0 I) \leq \varepsilon$,
  $I_t = I_t + 1$ ; $h = 2h$ ; $Z = z_0 + h$ ;
**end while**
$z_i = z_0$ ; $z_e = Z$ ; {*ETAPE I : Construction du triangle initial* }
**do** $k = 1 : I_t$,
  $z = (z_e + z_i)/2$ ;
  **if** $\sigma_{\min}(A - zI) \leq \varepsilon$, **then**
    $z_i = z$ ;
  **else**
    $z_e = z$ ;
  **end if**
**end**
$z' = z_i + \tau e^{i(\theta+\pi/3)})$ ; $T_0 = (z_i, z_e, z')$ ;
$T_1 = F(T_0)$ ; $i = 1$ ; $\mathscr{I} = \{[z_i, z_e],\ [z', p(T_1)]\}$ ; {*ETAPE II : Construction d'une $F-$orbite*}
**while** $T_i \neq T_0$,
  $T_{i+1} = F(T_i)$ ; $i = i + 1$ ;
  Choisir le nouveau côté de $T_i$ qui intersecte $\partial\Lambda_\varepsilon(A)$ et l'insérer dans la liste $\mathscr{I}$ ;
**end while**
{*ETAPE III : Construction de la ligne polygonale inscrite dans $\partial\Lambda_\varepsilon(A)$* }
**doall** $I \in \mathscr{I}$,
  Extraire $z_I \in \partial\Lambda_\varepsilon(A) \cap I$ par bissections itérées ;
**end**

alors un point $Z$ du réseau, extérieur à $\Lambda_\varepsilon(A)$ sur la droite $z = z_0 + k\tau e^{i\theta}$, point qui existe car $\lim_{|z|\to\infty} \sigma_{\min}(zI - A) = \infty$. Pour cela, on recherche pour $k = 1, 2, \cdots$, le premier point $z_k = z_0 + 2^k \tau e^{i\theta}$ qui soit extérieur. Puisque ce point est sommet dans le réseau $\mathscr{S}$, par bissection successive de l'intervalle $[z_{k-1}, z_k]$, on en déduit, par ajustement, un intervalle de longueur de côté $\tau$ qui intersecte le bord $\partial\Lambda_\varepsilon(A)$ de $\Lambda_\varepsilon(A)$ et on construit un triangle équilatéral sur ce segment.

**ETAPE II : construction de l'orbite.** La fonction $F$ est successivement appliquée jusqu'à la fermeture de l'orbite. Le test d'égalité de triangles est sûr puisqu'il est fondé sur un test d'entiers.

**ETAPE III : construction de points de la ligne de niveau.** Extraction par bissection. Cette étape peut être omise si on ne recherche qu'une ligne polygonale contenant le pseudo-spectre (voir section 4).

Le nombre minimum de triangles dans une orbite est de 6. Ceci correspond à la situation où le pivot est le même pour tous les triangles de l'orbite et par conséquent les angles de toutes les rotations sont égaux à $\pi/3$. Une telle orbite correspond souvent à une valeur de $\tau$ qui est grande par rapport à la longueur de $\partial\Lambda_\varepsilon(A)$.

**Proposition 1** *Supposons que l'algorithme 1 détermine une orbite $O(T_0)$ qui contienne $N$ triangles et que $N > 6$. Soit $\ell$ la longueur de la courbe $\partial\Lambda_\varepsilon(A)$ et $\eta$ la précision pour la détermination des points du contour. La complexité du calcul est définie par le nombre $\mathscr{N}_\varepsilon$ de points où $s(z) = \sigma_{\min}(A - zI)$ est calculé. Ce nombre est tel que*

$$\mathscr{N}_\varepsilon \leq (N+1)\lceil \log \frac{\tau}{\eta} \rceil \text{ et } N = O\left(\frac{\ell}{\tau}\right).$$

Pour calculer la valeur singulière minimale de $B = zI - A$, on choisit la méthode de Lanczos pour calculer la valeur propre maximale de la matrice augmentée

$$\begin{pmatrix} 0 & B^{-1} \\ B^{-*} & 0 \end{pmatrix}$$

grâce à une factorisation LU de la matrice $B$.

Quand la matrice $A$ et le point $z_0$ sont réels avec une inclinaison $\theta$ nulle du réseau, alors l'orbite admet comme axe de symétrie l'axe des réels. Dans ce cas, on peut arrêter la détermination de l'orbite dès que le nouvel intervalle intégrant la liste $\mathscr{I}$ est réel. Ce cas réduit de moitié le travail nécessaire à la construction de l'orbite et de la ligne polygonale recherchée.

## 3. La méthode EIGENCNT

Soit une matrice $A \in \mathbb{C}^{n\times n}$ et une ligne polygonale $\Gamma$ ne contenant pas de valeur propre de $A$. Alors le nombre de valeurs propres de $A$ qui appartiennent au domaine de frontière $\Gamma$ est donné par la formule de Cauchy

$$N_\Gamma = \frac{1}{2i\pi} \int_\Gamma \frac{f'(z)}{f(z)} dz$$

où $f(z) = \det(zI - A)$ est le polynôme caractéristique de $A$. Pour calculer effectivement cette intégrale par morceaux, on considère une discrétisation $\Gamma = \bigcup_{i=0}^{N-1} [z_i, z_{i+1}]$ de la ligne

polygonale où $[z_i, z_{i+1}]$ dénote un segment de la ligne de niveau avec pour extrémité $z_i$ et $z_{i+1}$. Soit $z$ et $z+h$ deux points de $(\Gamma)$. Puisque

$$\begin{aligned}(z+h)I-A &= (zI-A)+hI \\ &= (zI-A)(I+hR(z)),\end{aligned}$$

où $R(z)=(zI-A)^{-1}$, on peut écrire

$$f(z+h) = f(z)\,\det(I+hR(z)). \qquad [4]$$

En posant $\Phi_z(h)=\det(I+hR(z))$, on a

$$\begin{aligned}\int_z^{z+h}\frac{f^{'}(z)}{f(z)}dz &= \log(f(z+h))-\log(f(z)) \\ &= \log\left(\frac{f(z+h)}{f(z)}\right) \\ &= \log(\Phi_z(h)) \\ &= \log|\Phi_z(h)|+i\arg(\Phi_z(h)).\end{aligned}$$

Le déterminant est obtenu à partir de la factorisation LU de la matrice $(zI-A)$ et seule la partie imaginaire est utile. Une méthode robuste et parallèle de calcul de déterminant de grandes matrices creuses est proposée dans [4].

## 3.1. Contrôle du pas d'intégration

Afin d'intégrer de manière sûre, (c'est-à-dire de ne pas perdre de valeurs propres pendant l'intégration), il est nécessaire d'introduire un contrôle de pas. Dans [5], on montre que la condition

$$|\mathrm{Arg}(\Phi_z(s))| < \pi, \quad \forall s \in [0,h], \qquad [5]$$

où Arg représente l'argument principal, est appelée **Condition (A)** et qu'elle est nécessaire et suffisante ; elle est équivalente à :

$$\Phi_z(s) \notin (-\infty, 0], \quad \forall s \in [0,h].$$

Une condition plus restrictive que la **Condition (A)** mais suffisante est définie par :

$$|\Phi_z(s)-1| < 1, \quad \forall s \in [0,h]; \qquad [6]$$

elle sera appelée **Condition (B)**. Si elle n'est satisfaite qu'en $z+h$, elle se réduit à :

$$|\Phi_z(h)-1| < 1, \qquad [7]$$

et sera appelée **Condition (B')** ; cette condition est celle utilisée dans [1]. S'il est clair que **Condition (B)** implique **Condition (A)** il n'en est pas le cas pour **Condition (B')**. Comme il serait difficile de vérifier la **Condition (B)**, on utilise une approximation linéaire donnée par :

$$\Phi_z(u) \approx 1 + u\,\Phi_z'(0)$$

d'où la **Condition (C)** suivante :

$$|\delta|\,|\Phi_z'(0)| < 1. \qquad [8]$$

où

$$\Phi'_z(0) = \left[\frac{d}{du}\left(\det(I + u\,R(z))\right)\right]_{u=0},$$

avec $R(z) = (A - zI)^{-1}$. Lorsque $\Phi'_z(0) \approx 0$, l'approximation linéaire n'est pas valable et

$$\Phi_z(\delta) \not\approx 1 + \delta\,\Phi'_z(0).$$

Pour garantir un bon pas d'intégration, on impose que la **Condition (C)** soit satisfaite aux deux extrémités de l'intervalle :

$$|\delta| \leq \min\left(|\Phi'_z(0)|, |\Phi'_{z+\delta}(0)|\right).$$

Les trois conditions sont logiquement reliées comme suit :

**(A)** Condition nécessaire et suffisante

$\Uparrow$

**(B)** Condition suffisante

$\approx$

**(C)** Approximation de la condition suffisante

Pour que la **Condition C** soit satisfaite à chaque morceau, il sera nécessaire d'ajouter des points intermédiaires supplémentaires. Plus précisément : si **Condition (C)** est violée, on insère $M$ points uniformément répartis entre $z$ and $z+h$ où

$$M = \min\left(\left\lceil |h|\ |\Phi'_z(0)| \right\rceil, M_{\max}\right), \qquad [9]$$

et $M_{\max}$ étant un paramètre pré-défini. Lorsque la **Condition (B')** est violée et la **Condition (C)** satisfaite, on insère le point $z + h/2$.

## 3.2. Calcul de la dérivée

Pour le calcul de dérivée, à partir d'un simple calcul polynomial, on peut montrer que :

$$\Phi'_z(0) = \left[\frac{d}{du}\left(\det(I + u\,R(z))\right)\right]_{u=0} \qquad [10]$$

$$= \operatorname{trace}(R(z)). \qquad [11]$$

La factorisation $P(A - zI)Q = LU$ est déjà faite pour le calcul du $\det(I + \delta\,R(z))$. Cependant le coût du calcul de la $\operatorname{trace}(R(z))$ est très élevé puisqu'il nécessite de résoudre $n$ systèmes linéaires ; en effet, puisque $\operatorname{trace}(R(z)) = \sum_{i=1}^{n} R(z)_{ii}$ où

$$R(z)_{ii} = e_i^T (zI - A)^{-1} e_i, \qquad [12]$$

il faut résoudre les systèmes linéaires suivants :

$$(zI - A)x_i = e_i, \text{ pour } i = 1, \cdots n. \qquad [13]$$

Nous proposons deux méthodes pour estimer la dérivée à moindre coût : par taux d'accroissement et par estimation stochastique.

### 3.2.1. Estimation de la dérivée par taux d'accroissement

Dans [5] on envisage d'approcher la dérivée par taux d'accroissement. Nous choisissons ici

$$\left[\frac{d}{d\delta}\left(\det(I+\delta\, R(z))\right)\right]_{\delta=0} \approx \frac{\Phi_z(s)-\Phi_z(0)}{s},$$

où $s = O(\sqrt{\varepsilon}|z|)$ ; on exploitera la structure creuse de la matrice lors de sa factorisation LU. Étant donnés $z$ et $z+h$, la dérivée de $\Phi_z$ au point 0 est estimée par :

$$\Phi_z'(0) \approx \frac{\Phi_z(s)-1}{s},$$

où $s = \alpha h$ avec $\alpha = \min(10^{-6}\mu/|h|, 1)$, and $\mu = \max_{z\in\Gamma}|z|$. Ainsi, une deuxième factorisation LU est nécessaire pour le calcul de $\Phi_z(s)$.

### 3.2.2. Estimation de la dérivée par méthode stochastique

Cette approche consiste à trouver un estimateur de la trace en diminuant le nombre de systèmes linéaires à résoudre. On choisit de façon aléatoire, $N$ indices entre 1 et $n$, la moyenne arithmétique de la trace obtenue avec les $N$ indices est supposée approcher $\frac{trace(R(z))}{n}$, ce qui nous amène à écrire :

$$\sum_{i=1}^{n} R(z)_{ii} \simeq \frac{n}{N} \sum_{i=1..N; j_i \in 1..N} R(z)_{j_i j_i}. \qquad [14]$$

**Dimensionnement de l'échantillon**
Pour estimer le nombre de systèmes linéaires à résoudre afin d'avoir une approximation de la trace de la résolvante avec au moins un chiffre significatif, nous choisissons un point $z_k$ de la ligne de niveau et nous approchons la trace pour différentes valeurs de N. Pour chaque valeur de N, nous effectuons 10 tirages et nous calculons la moyenne des erreurs, l'erreur maximale est également mentionnée pour illustrer le pire des cas.
La figure 4 présente les résultats obtenus pour la matrice ADD32 de taille $n = 4960$ et la matrice CRY10000 de taille $n = 10000$. Sur cette figure, nous pouvons remarquer qu'avec $N = 400$ nous avons une bonne approximation de la trace pour la matrice ADD32 de taille $n = 4960$ alors que pour la matrice CRY10000 de taille $n = 10000$, il faut résoudre $N = 1000$ pour être sûr de toujours avoir une bonne approximation de la trace de la résolvante.

## 3.3. L'algorithme EIGENCNT

La méthode EIGENCNT est décrite dans l'algorithme 2. A partir d'une liste de points $Z$, l'algorithme inclut de nouveaux points selon que la **Condition (C)** ou la **Condition (B')** est satisfaite ou pas. La complexité de l'algorithme dépend du nombre de déterminants à calculer et du nombre de systèmes linéaires à résoudre. Pour chaque point $z \in Z$, on calcule $\det(zI-A)$ et $\Phi_z'(0)$ qui nécessite soit un nouveau calcul de déterminant lorsque l'on estime la dérivée par le taux d'accroissement soit la résolution de $N$ systèmes linaires avec $N \ll n$, lorsque l'on utilise la méthode stochastique.
Lorsque l'estimation de la dérivée se fait avec le taux d'accroissement, si en sortie, la liste $Z$ contient $N_z$ points, alors la complexité de l'algorithme peut s'exprimer sous la forme :

$$\mathscr{C}_{eig} = 2\mathscr{C}_{LU} N_z,$$

où $\mathscr{C}_{LU}$ est le nombre d'opérations pour la factorisation LU complexe de $(zI-A)$.
Lorsque l'estimation de la dérivée est stochastique, pour chaque point $z \in Z$, on calcule

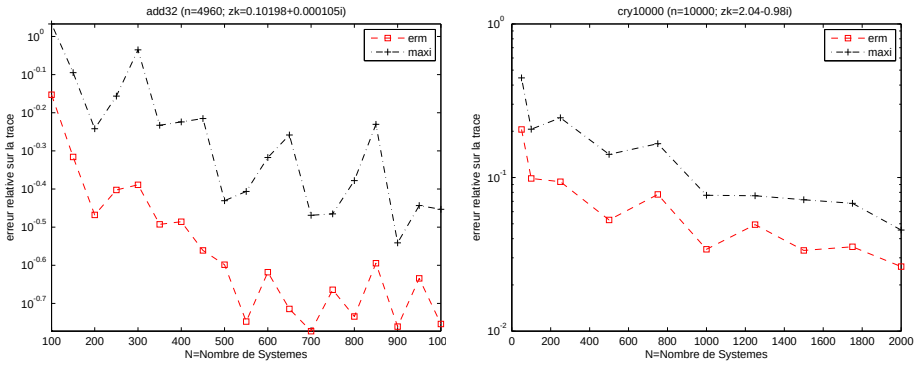


**Figure 4.** *Estimation du nombre de systèmes nécessaires pour approximer la trace.*

$\det(zI-A)$, avec la même factorisation $LU$ nous résolvons $N$ systèmes linéaires pour estimer $\Phi_z'(0)$ ; avec $N_z$ points en sortie, la complexité de l'algorithme peut alors s'exprimer sous la forme :

$$\mathscr{C}_{eig} = \mathscr{C}_{LU} N_z + \mathscr{C}_{solve} N,$$

où $\mathscr{C}_{solve}$ est le nombre d'opérations pour la résolution du système $(zI-A)x_i = e_i$ (c'est à dire une descente-remontée des systèmes triangulaires), pour un indice $i$ donné.

# 4. La combinaison des deux méthodes et leur parallélisation

## 4.1. Combinaison des méthodes PAT et EIGENCNT

On combine maintenant les deux procédures PAT et EIGENCNT : deux approches sont possibles.

### 4.1.1. Première approche

L'algorithme PAT détermine une liste de points Z de la courbe de niveau $(\Gamma_\varepsilon)$. A chaque point $z \in Z$, $\det(A-zI) = \mathrm{sign}(P)\mathrm{sign}(Q)\det(U)$,$\sigma(z) = \sigma_{\min}(A-zI)$ sont calculés, où $P(A-zI)Q = LU$ est la factorisation LU. Le calcule de $\sigma_{\min}(A-zI)$ se fait par Lanczos sur

$$M_z = \begin{pmatrix} 0 & U^{-1}L^{-1} \\ L^{-*}U^{-*} & 0 \end{pmatrix},$$

puisque $\sigma_{\min}(A-zI) = \sigma_{\min}(LU)$. On fait appel à EIGENCNT en lui envoyant la liste des points.

### 4.1.2. Seconde approche

L'algorithme PAT construit plutôt une liste de triangles qui recouvrent la courbe $\Gamma$, sans interpolation pour obtenir des points la courbe $\Gamma$. On peut se servir de l'algorithme 3. A partir des noeuds des triangles de la liste, on déduit deux listes de points : $Z_{int}$ de points intérieurs et $Z_{ext}$ de points extérieurs. On fait appel à EIGENCNT en lui envoyant la liste $Z_{ext}$. Nous utilisons ici la seconde approche.

### 4.1.3. Méthode combinée

Les grandes étapes du calcul sont :

**Algorithm 2** EIGENCNT : Algorithme de dénombrement de valeurs propres dans un domaine du plan entouré par $(\Gamma)$.

**Require:** ENTREES : $Z$={noeuds de $(\Gamma)$}, $M_{\text{pts}}$ = nombre maximum de points autorisés,
$M_{\max}$ = nombre maximum de points à insérer simultanément,
SORTIES : *neg* = nombre de valeurs propres entourées par $(\Gamma)$ ;

Statut($Z$)=-1 ;
**while** Statut($Z$)$\neq 0$ et longueur($Z$) $< M_{pts}$,
  **do** $z \in Z$ tel que Statut($z$)==-1,
    calcul $det(zI-A)$ et $\Phi'_z(0)$ ;
    Statut($z$) = 1 ;
  **end**
  **do** $z \in Z$ tel que Statut($z$)=1,
    **if Condition (C)** non satisfaite au point $z$, **then**
      Générer $M$ points $\tilde{Z}$ comme dans (9) ;
      $Z$=$Z \cup \tilde{Z}$ ; Statut($\tilde{Z}$)=-1 ;
    **else if Condition (B')** non satisfaite au point $z+h$ ; **then**
      $Z$=$Z \cup \{z+h/2\}$ ; Statut($z+h/2$)=-1 ;
    **else**
      Statut($z$)=0 ;
    **end if**
  **end**
  **if** pas de nouveaux points à insérer dans $Z$ ; **then**
    **do** $z \in Z$,
      **if Condition (C)** est non satisfaite anormalement ; **then**
        $Z$=$Z \cup \{z-h/2\}$ ; Statut($z-h/2$)=-1 ;
      **end if**
    **end**
  **end if**
**end while**
Integral = $\sum_{z \in Z} \text{Arg}(\Phi_z(0))$ ; *neg* = round(Integral/ $2\pi$) ;

**Algorithm 3** TRIANGLESET : Algorithme de construction de liste de triangles recouvrant $\Gamma$.

**Require:** ENTREES : $\varepsilon$, $\tau$ et $\tilde{z}_0$ tel que $s(\tilde{z}_0) \leq \varepsilon$
**Ensure:** construire $T_0 \in T_L$ triangle initial ;

**do** $k \geq 0$,
  $T_{k+1} = F(T_k)$ ;
  **if** $T_{k+1} = T_0$, **then**
    $L = k$ ;
    break ;
  **end if**
**end**

– Choisir la résolution $\tau > 0$ et l'inclinaison $\theta$ du maillage triangulaire et définir un point de référence $z_{ref} \in \mathbb{C}$ autour duquel on recherche des valeurs propres. Calculer une valeur propre $z_0$ voisine de $z_{ref}$ par itération inverse ou autre méthode plus rapide.

– Par PAT, construire une liste de triangles à partir de $z_0$ et $\tau e^{i\theta}$ (étapes I et II de PAT). En déduire la ligne polygonale $Z_{ext}$ des sommets extérieurs de la liste des triangles de l'orbite. Pour tout sommet $z$, les nombres $\sigma_{\min}(zI-A)$ et $\det(zI-A)$ sont donc calculés. Ces deux quantités complexes sont obtenues à partir de la même factorisation LU de la matrice $(zI-A)$.

– A l'aide de EIGENCNT, calculer le nombre de valeurs propres entourées par la ligne polygonale trouvée précédemment. Cette étape rajoutera des points sur les segments de la ligne polygonale si la longueur est trop grande pour une intégration sûre. Chaque nouveau point entraîne une ou deux factorisations LU de la résolvante.

### 4.1.4. Résultats Numériques

Les tests sont conduits sur des matrices issues soit de Matrix Market [14] soit de la collection de Tim Davis[15]. Les résultats sont rassemblés dans le tableau 1. Pour chaque exécution en plus de la matrice, on doit spécifier un point de référence $zref$, une taille du maillage $\tau$ , un niveau de la ligne de niveau à détecter $\eta$ et le nombre $N_s$ de systèmes linéaires à résoudre pour calculer la dérivée $\Phi'_z(0)$ en un point $z$. Le programme nous retourne le nombre $nvp$ de valeurs propres entourées par la ligne polygonale formée par les points extérieurs de chaque triangle de la chaîne des triangles construite, le nombre $nT$ de triangles constituant la chaîne et le nombre $nintv$ d'intervalles obtenu à la fin de l'intégration. Les tests numériques ont été menés sur une machine équipée de deux processeurs,

| Name | $n$ | Zref | $\tau$ | $\eta$ | nvp | Ns | nT | nintv |
|---|---|---|---|---|---|---|---|---|
| GRCAR100 | 100 | 1.7+1.1i | $10^{-1}$ | $10^{-6}$ | 100 | 10 | 344 | 2109 |
| FS-680-1 | 680 | $7x10^{9}$ | $10^{5}$ | $10^{5}$ | 6 | 10 | 646 | 2763 |
| ADD20 | 2395 | 0.1+0.1i | $10^{-5}$ | $10^{-4}$ | 43 | 100 | 496 | 251 |
| CRY2500 | 2500 | 1.5+1.7i | $10^{-1}$ | $10^{-3}$ | 569 | 100 | 170 | 13199 |
| UTM3060 | 3060 | -1.5+1.2i | $10^{-1}$ | $10^{-3}$ | 4 | 100 | 6 | 3083 |
| MHD4800B | 4800 | i | $10^{-4}$ | $10^{-6}$ | 1230 | 100 | 1370 | 29630 |
| ADD32 | 4960 | 0.1+0.1i | $10^{-5}$ | $10^{-4}$ | 89 | 100 | 408 | 1240 |
| UTM5940 | 5940 | 1.5+1.2i | $10^{-2}$ | $10^{-6}$ | 1 | 100 | 6 | 926 |
| CRY10000 | 10000 | 1.5+1.7i | $10^{-1}$ | $10^{-3}$ | 652 | 1000 | 312 | 31552 |
| BCSSTK17 | 10974 | 1+i | $10^{-1}$ | $10^{-6}$ | 518 | 1000 | 6 | 5656 |
| BCSSTM25 | 15439 | 0.1+0.1i | $10^{-3}$ | $10^{-2}$ | 6 | 1000 | 150 | 706 |
| BCSSTM25 | 15439 | 0.1+0.1i | $10^{-3}$ | $10^{-1}$ | 30 | 1000 | 1950 | 3477 |
| E40R5000 | 17281 | 1+2i | $10^{-1}$ | $10^{-6}$ | 2 | 1000 | 6 | 2090 |
| AF23560 | 23560 | i | $10^{-1}$ | $10^{-6}$ | 1 | 1000 | 6 | 731 |
| BCSSTK31 | 35588 | 0.1+0.1i | $10^{-1}$ | $10^{-6}$ | 2318 | 1000 | 6 | 15606 |
| FINAN512 | 74752 | 2+25i | $10^{-1}$ | $10^{-3}$ | 611 | 1000 | 238 | 26884 |

**Tableau 1.** *Nombre de valeurs propres localisées pour différentes matrices en différentes régions*

chacun ayant 6 cœurs et 24 GB de mémoire ; chaque cœur a un cache L1 de 32 KB et un cache L2 de 256 KB ; les 6 cœurs d'un processeur partagent un cache L3 de 12 MB ; chaque cœur est un Intel(R) Xeon(R) de 3.47 GHZ de vitesse d'horloge ; la machine est équipée d'un système d'exploitation Red had 4.6.2-1 ; du compilateur gcc 4.6.2 ; le code est écrit en C et compilé avec pour flag -g -w -O3 ; le solveur UMFPACK est utilisé pour la factorisation LU de matrice creuses et la résolution des systèmes linéaires.

## 4.2. Parallélisation

Le problème de suivi de ligne de niveaux offre par sa nature même une possibilité de parallélisation. Il suffit de démarer la construction de l'orbite dans six directions ; ce qui constitue six tâches de calcul indépendantes. On n'attend pas la fin de la construction de l'orbite pour commencer l'intégration ; dès que l'on a deux points extérieurs, ce qui constitue un segment de la ligne de niveau, l'on peut immédiatement procéder à la génération de nouveaux points en testant les conditions. Ces nouveaux points donnent lieu à des tâches de calcul indépendantes : calculs de déterminants et de dérivées. Ainsi, un modèle de calcul de type *client-serveur* est considéré : un processus serveur, dit *Maître*, est utilisé pour générer un ensemble de tâches de calcul et plusieurs processus clients, dit *Travailleurs* sont utilisés pour effectuer les calculs. Lorsqu'un *Travailleur* termine sa tâche, il rend le résultat au *Maître* qui lui communique une nouvelle tâche de calcul. Cet ordonnancement dynamique des tâches optimise l'équilibrage des charges entre les processeurs.

### 4.2.1. Modèle architecturale

L'architecture de mise en œuvre de notre modèle est présentée dans la figure 5 ; on y distingue deux principales tâches de calculs : la tâche $compute\_chain()$ qui, étant donné un point intérieur $z_i$, une résolution du maillage $\tau$ et une inclinaison $\theta$, progresse dans la direction $\theta$ à la recherche de $z_e$ tels que $\sigma_{min}(z_e I - A)$, puis construit un premier triangle et une orbite située dans la direction $\theta$. La deuxième tâche est $compute\_det(z)$ et consiste au calcul du déterminant et à l'estimation de la dérivée au point $z$.

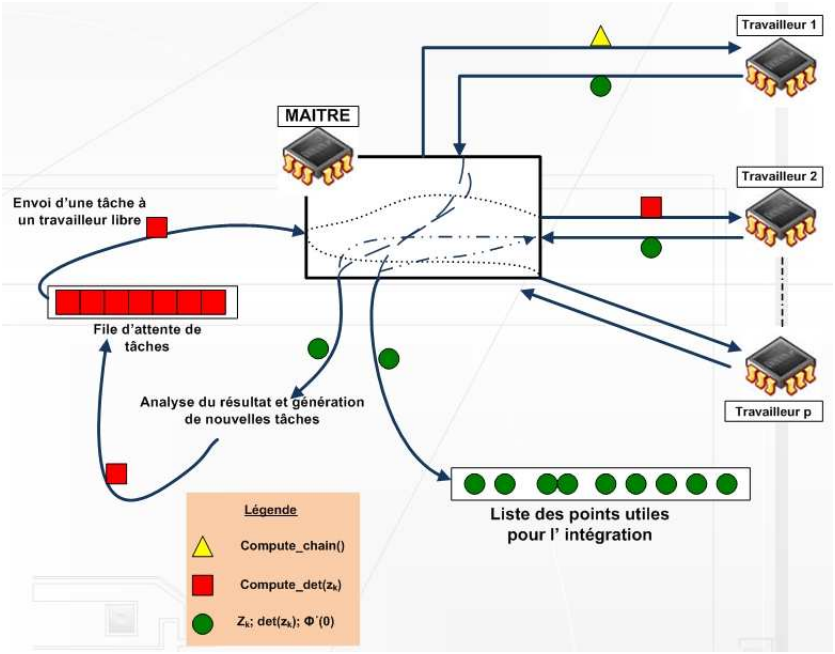


**Figure 5.** *:Modèle de calcul parallèle*

### 4.2.2. Le Maître

Pour maximiser l'efficacité, le processus *Maître* doit générer le maximum possible de tâches pour occuper les *Travailleurs*.
Au démarrage, le *Maître* génère six tâches $compute\_chain()$. Ces tâches sont envoyées aux *Travailleurs* et le *Maître* passe dans un état d'attente de messages.
A la réception d'un message, le *Maître* analyse les données reçues, si elles proviennent d'une tâche $compute\_chain()$, le *Maître* teste les conditions, génère de nouveaux points $z_k$ puis pour chaque point $z_k$ généré, crée une tâche $compute\_det(z_k)$. Ces tâches sont rangées dans une file d'attente puis servies aux *Travailleurs* au fur et à mesure qu'ils se libèrent. Si les données proviennent d'une tâche $compute\_det(z_k)$, le *Maître* les rangent et attribue une autre tâche au *Travailleur*.

### 4.2.3. Le Travailleur

Le *Travailleur* est un processus simple qui répond aux requêtes du processus *Maître*. Il offre aux processus *Maître* les services suivants :
$compute\_chain()$ : étant donné $\tau$ une taille du maillage, $\eta$ un niveau de la ligne à détecter et $d$, une direction, construire l'orbite dans la direction $d$ en retournant chaque point extérieur au *Maître*. Cette tâche nécessite pour chaque sommet du triangle, le calcul d'un $\sigma_{min}$. Lorsqu'il s'agit d'un point extérieur $z_k$ (dont à retourner au Maître), il faut en plus du calcul de $\sigma_{min}$, calculer $\Phi'_{z_k}(0)$ en résolvant $N$ systèmes linéaires lorsque l'on utilise l'estimation stochastique pour la dérivée ; cette résolution utilise la même factorisation LU effectué pour le calcul de $\sigma_{min}$.
$compute\_det(z_k)$ : étant donné un point $z_k$, calculer $\det(z_kI - A)$ et $\Phi'_{z_k}(0)$. Cette tâche nécessite une factorisation LU pour le calcul du déterminant de $(z_kI - A)$. Lorsque l'estimation de la dérivée est stochastique, il faut résoudre $N$ systèmes linéaires pour le calcul de $\Phi'_{z_k}(0)$ alors que lorsque l'estimation de la dérivée utilise le taux d'accroissement, il faut effectuer une nouvelle factorisation LU en un point voisin de $z_k$
Chacun de ces services nécessite un accès à la matrice. Pour cela cette matrice est diffusée aux différents *Travailleurs* au démarrage de l'application. Dans le cas des matrices de très grande taille, un processeur physique risque de manquer de ressources mémoire pour assurer le service. En plus la complexité de ces services augmente considérablement car le calcul de valeurs singulières, de déterminants et de dérivées devient très long. Un second niveau de parallélisme peut être introduit si les nœuds de calcul utilisés ont plusieurs cœurs, ceci en exploitant soit la méthode décrite dans [4] lorsque la dérivée est approchée par taux d'accroissement soit la résolution parallèle des systèmes linéaires.

## 4.3. Résultats Numériques

Nous avons implémenté notre algorithme sur la grille de calcul `igrida` du centre de recherche INRIA de Rennes et particulièrement sur le cluster `lambda` qui est équipé de 11 nœuds de calculs. Chaque nœud possède 2 x 6 CPU de modèle `Westmere-EP` et de référence `Intel(R) Xeon(R) CPU E5645 @ 2.40GHz`, ainsi que 48GB de RAM, les nœuds sont reliés entre eux grâce à un réseau `infiniband` de `1GB/s`. Les nœuds de calculs sont équipés du système `Debian` version 6.0.8, de `openmpi 1.6`. Le code est écrit en C et compilé avec pour flag `-g -w -O3` ; le solveur UMFPACK [12, 13] est utilisé pour la factorisation LU de matrices creuses et la résolution des systèmes linéaires. Les calculs sont effectués en arithmétique complexe double précision et la bibliothèque MPI[16, 17] assure la connexion entre le *Maître* et les *Travailleurs*.
L'algorithme parallèle est exécuté sur plusieurs matrices ; dans un premier exemple nous illustrons sur la figure 6 la construction de la chaîne de triangles en démarrant dans 6 directions ; dans l'exemple 2, nous illustrons le fait que les processus lancés dans ces 6 directions peuvent découvrir plusieurs composantes du pseudo spectre ; ceci est illustré par la figure 7. Les figures 8 et 9 montrent l'accélération et l'efficacité de l'algorithme

parallèle sur trois matrices.

### 4.3.1. Premier cas test du programme parallèle

Le premier cas test concerne la matrice BCSSTM25 qui est issue d'un problème généralisé de valeurs propres. Cette matrice est tirée de la collection de Harwell-Boeing de Matrix Market[14] sa taille est $n = 15439$, sa norme 2 vaut $7 \times 10^8$ et son conditionnement $6.1 \times 10^9$, pour un point de référence $z_{ref} = 0.1 + 0.1i$ une taille de maillage $\tau = 10^{-3}$ et un niveau de ligne $\eta = 10^{-2}$. La figure 6 présente une orbite de 150 triangles, la ligne polygonale obtenue en considérant les points extérieurs des triangles contient 6 valeurs propres et le nombre d'intervalles à la fin de l'intégration est 494.

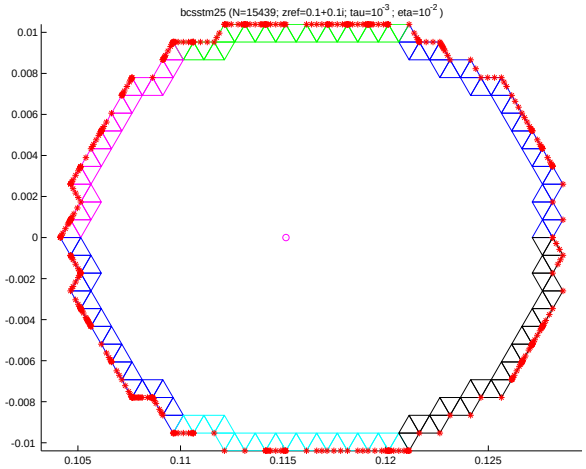


**Figure 6.** *: Ligne de niveau construite pour la matrice* BCSSTM25 *(les étoiles rouge représentent les points nécessaires à l'intégration)*

### 4.3.2. Deuxième cas test du programme parallèle

Nous illustrons dans ce cas test la découverte de plusieurs composantes du pseudo-spectre lorsque l'on démarre la construction dans 6 directions la matrice considérée est FS-680-1, une matrice issue de l'étude d'un problème de radiations chimiques elle est également tirée de la collection Harwell-Boeing da Matrix Market[14], sa taille est $n = 680$, sa norme 2 vaut $7.2 \times 10^{13}$ et son conditionnement $2.1 \times 10^4$, pour un point de référence $z_{ref} = 7 \times 10^9$ une taille de maillage $\tau = 10^5$ et un niveau de ligne $\eta = 10^6$.
Deux composantes du pseudo spectre sont construites :

– une première orbite de 646 triangles dont l'intégration sur les points externes donne 6 valeurs propres pour un nombre total de points insérés égale à 2677 ;

– une deuxième de 1250 triangles dont l'intégration sur les points extérieurs donne 488 valeurs propres avec un nombre de points égale à 12893.

La figure 7 présente ces résultats.

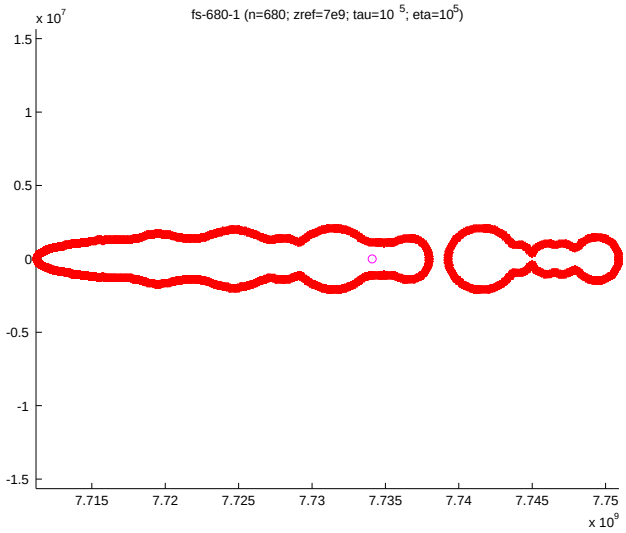


**Figure 7.** *: Ligne de niveau construite pour la matrice* FS-680-1

### 4.3.3. Autres cas test

Pour illustrer le gain obtenu en parallélisant l'algorithme combiné, nous avons effectué des tests avec des matrices dont les caractéristiques sont décrites dans le tableau 2. Les

| Matrice | Origine | Type | n | nZ |
|---|---|---|---|---|
| E40R5000 | modélisation d'écoulement | réelle non symétrique | 17281 | 553956 |
| CHEVRON1 | modélisation sismique | complexe non hermitienne | 37365 | 330633 |
| CIRCUIT4 | modélisation électrique | réelle non symétrique | 80209 | 307604 |

**Tableau 2.** *Caractéristiques de quelques matrices tests.*

zones d'intérêt ($z_ref$), la résolution du maillage ($\tau$), le niveau de la ligne ($\eta$), le nombre de triangles, le nombre de sommets de la ligne polygonale et le nombre de valeurs propres encerclées sont présentés dans le tableau 3. Le tableau 4 présente les temps de calcul, l'accélération et l'efficacité pour les matrices test du tableau 2. De ces données, il ressort que, au dessus de 7 processeurs (84 cœurs), on commence à perdre en accélération et efficacité ; ceci est dû non seulement aux tranches incomplètes mais également au temps de communication qui augmente avec le nombre de processeurs(cœurs).

Les courbes de la figure 8 et 9 illustrent l'accélération et l'efficacité obtenue pour les matrices du tableau 2 en fonction du nombre de processeurs employés de 1 à 10. Chaque processeur a 12 cœurs et les calculs sont effectués sur des cœurs.

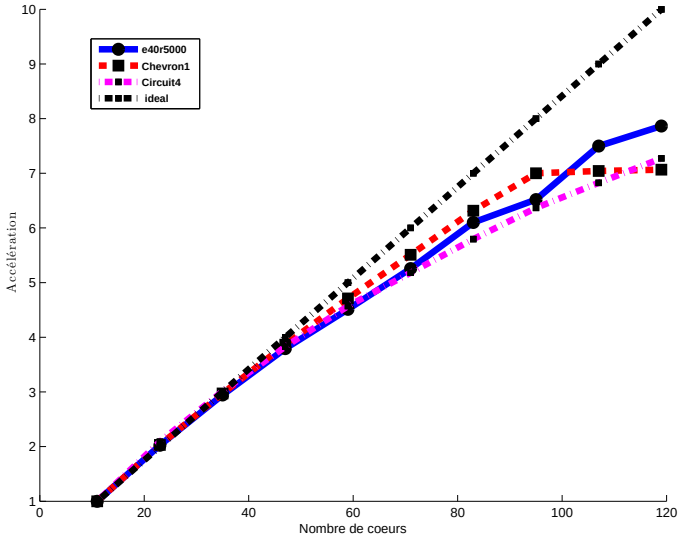


**Figure 8.** *Accélération première série de tests par rapport à 120 coeurs*

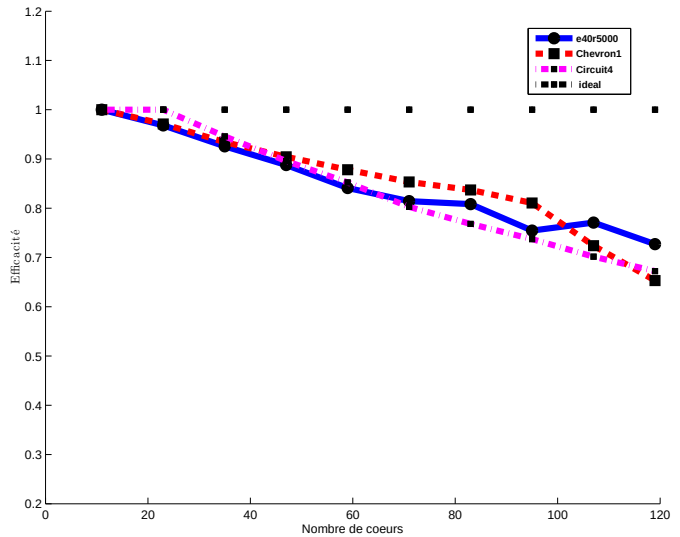


**Figure 9.** *Efficacité première série de tests*

| Matrice | $z_{ref}$ | $\tau$ | $\eta$ | Lignes | Triangles | Sommets | Valeurs P |
|---|---|---|---|---|---|---|---|
| e40r5000 | $1.7+1.1i$ | 0.1 | $10^{-6}$ | 1 | 6 | 1516 | 2 |
| Chevron1 | $-0.039$ | $10^{-4}$ | $10^{-4}$ | 1 | 48 | 1377 | 2 |
| Circuit4 | 0 | $10^{-4}$ | $10^{-4}$ | 2 | 122(278) | 11243(78463) | 128(11507) |

**Tableau 3.** *Nombre de valeurs propres par zone d'intérêt.*

## 5. Conclusion

Dans cet article, nous avons développé un algorithme parallèle en combinant deux approches : PAT qui construit une orbite de triangles équilatéraux en suivant une ligne de niveau $\eta$ et la méthode EIGENCNT, qui compte le nombre de valeurs propres entourées par une ligne polygonale a priori donnée dans le plan complexe. Cet algorithme réduit le nombre de factorisations LU en exploitant les factorisations LU obtenues lors du calcul de $\sigma_{min}$ pour calculer la dérivée. L'utilisation d'un modèle *Client-Serveur* pour l'implémentation de cet algorithme permet d'optimiser l'équilibrage de charges entre les différents processeurs d'un réseau de machines, ainsi que de réduire la complexité des données échangées. Les communications sont réduites à un échange de messages courts, constitués de quelques nombres complexes entre le *Maître* et les *Travailleurs* permettant d'utiliser des machines géographiquement éloignées.
Les bonnes performances de cet algorithme nous permettent de dénombrer les valeurs propres de grandes matrices dans une région du plan complexe en construisant des lignes polygonales faisant intervenir un nombre de factorisation LU très important pour plus de valeurs propres à dénombrer.
Malgré le fait que cet algorithme permette une localisation robuste de valeurs propres de matrice dans le plan complexe, un deuxième niveau de parallélisme doit être considéré pour paralléliser le calcul de déterminant et/ou d'une dérivée et même de $\sigma_{min}$. Cela serait possible en considérant des matrices bloc-diagonales comme dans [4].

## 6. Bibliographie

[1] O. Bertrand et B. Philippe. Counting the eigenvalues surrounded by a closed curve. *Siberian Journal of Industrial Mathematics*, 4 :73–94, 2001.

[2] D. Bindel, Bounds and error estimates for nonlinear eigenvalue problems. Berkeley Applied Mathematical Seminar, October 2008. D. Bindel, Bounds and error estimates for nonlinear eigenvalue problems. Berkeley Applied Mathematical Seminar, October 2008.

[3] S. Godunov. Spectral portraits of matrices and criteria of spectrum dichotomy. In L. Atanassova and J. Hezberger, editors, *Third Int'l. IMACS-CAMM Symposium on Computer Arithmetic and Enclosure Methods*, Amsterdam, 1992.

[4] E. Kamgnia et L. B. Nguenang. Some efficient methods for computing the determinant of large matrices. *ARIMA Journal*, Special issue CARI'12, 17 :73-92, 2014.

[5] E. Kamgnia et B. Philippe. Counting eigenvalues in domains of the complex field. *ETNA*, 40 :1–16, 2013.

[6] D. Mezher et B. Philippe. Parallel computation of pseudospectra of large sparse matrices. *Parallel Comput.*, 28(2) :199–221, 2002.

[7] D. Mezher et B. Philippe. PAT : A reliable path following algorithm. *Numer. Algorithms*, 29 :131–152, 2002.

[8] E. Polizzi et A. Sameh. A parallel hybrid banded systems solver : the SPIKE algorithm. *Parallel Comput.*, 32 :177–194, 2006.

| | E40R5000 | | | CHEVRON1 | | | CIRCUIT4 | | |
|---|---|---|---|---|---|---|---|---|---|
| Proc | Temps | Acc. | Eff. | Temps | Acc. | Eff. | Temps | Acc. | Eff. |
| 1(12) | 1475.86 | 1.0 | 1.0 | 4831.73 | 1.0 | 1.0 | 56095.59 | 1.0 | 1.0 |
| 2(24) | 729.11 | 2.02 | 0.96 | 2379.44 | 2.03 | 0.97 | 26828.33 | 2.09 | 1.0 |
| 3(36) | 501.09 | 2.94 | 0.92 | 1625.50 | 2.97 | 0.93 | 18636.98 | 3.00 | 0.94 |
| 4(48) | 389.13 | 3.79 | 0.88 | 1250.23 | 3.86 | 0.90 | 14655.26 | 3.82 | 0.89 |
| 5(60) | 327.19 | 4.51 | 0.84 | 1025.75 | 4.71 | 0.87 | 12269.34 | 4.57 | 0.85 |
| 6(72) | 280.76 | 5.25 | 0.81 | 877.10 | 5.50 | 0.85 | 10827.51 | 5.18 | 0.80 |
| 7(84) | 241.97 | 6.09 | 0.80 | 765.03 | 6.31 | 0.83 | 9678.83 | 5.79 | 0.76 |
| 8(96) | 226.44 | 6.51 | 0.75 | 690.19 | 7.00 | 0.81 | 8810.29 | 6.36 | 0.73 |
| 9(108) | 196.84 | 7.49 | 0.77 | 686.09 | 7.04 | 0.72 | 8219.26 | 6.82 | 0.70 |
| 10(120) | 187.64 | 7.86 | 0.72 | 683.78 | 7.06 | 0.65 | 7713.35 | 7.27 | 0.67 |

**Tableau 4.** *Temps(en s), accélération et efficacité pour e40r5000, Chevron1 et Circuit4.*

[9] Y. Maeda, Y. Futamura, AND T. Sakurai, Stochastic estimation method of eigenvalue density for nonlinear eigenvalue problem on the complex plane, JSIAM Letters, 3 :61-64, 2011.

[10] G. W. Stewart and Ji-guang Sun. Matrix Perturbation Theory Academic Press Inc. page 174, 1990.

[11] L. Trefethen. Pseudospectra of Matrices. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis*, pages 234–266. Longman, 1992.

[12] UMFPACK ; Unsymmetric Multifrontal Sparse LU Factorization Package ; *http ://www.cise.ufl.edu/research/sparse/umfpack/*

[13] T. A. Davis Algorithm 832 : UMFPACK, an unsymmetric-pattern multifrontal method *ACM Transactions on Mathematical Software*, 30 :2, 196 – 199, 2004.

[14], Matrix Market, *Service of the Mathematical and Computational Sciences Division / Information Technology Laboratory / National Institute of Standards and Technology*, http ://math.nist.gov/MatrixMarket/.

[15], T. A. Davis and Y. Hu The University of Florida Sparse Matrix Collection, *ACM Transactions on Mathematical Software*, Vol 38, Issue 1, 2011, pp 1 :1 - 1 :25, http ://www.cise.ufl.edu/research/sparse/matrices.

[16], Naoyuki Shida and Shinji Sumimoto and Atsuya Uno MPI Library and Low-Level Communication on the K computer, *FUJITSU Scientific & Technical Journal* ., Vol 48, 2012, pp 3 :324 - 3 :330,

[17], MPI Forum MPI : A Message-Passing Interface Standard, *Message Passing Interface Forum*, http ://www.mpi-forum.org/