Performance du SVD pour débruiter les spectres RMN et Raman

Guillaume Laurent[1], William Woelffel[2], Virgile Barret-Vivin[1]
Emmanuelle Gouillart[2], Christian Bonhomme[1]

[1]Sorbonne Universités, UPMC Univ Paris 06, CNRS, Collège de France, Laboratoire de Chimie de la Matière Condensée de Paris (LCMCP), 11, Place Marcelin Berthelot, 75231 Paris Cedex 05, France
[2]Saint-Gobain, CNRS, Surface du Verre et Interfaces, UMR 125, 39 quai Lucien Lefranc – BP 135, 93303 Aubervilliers Cedex, France

guillaume.laurent@upmc.fr

Abstract *(SVD performance to denoise NMR and Raman spectra)*

Singular Value Decomposition (SVD) is a mathematical tool that can be used to remove noise from spectra. In this article, we used it on Nuclear Magnetic Resonance and Raman spectra. The results proved that this technique can be very efficient, either on a single 1D spectrum or on an array of spectra and that it can be easily generalised. We compared execution time on a few processors and graphic cards with Java, Matlab and Python. Impressive differences were seen, probably due to the used optimisations. Execution time is now short enough to apply SVD on continuous experiments.

1. Introduction

Différentes techniques spectroscopiques sont couramment utilisées dans les laboratoires pour caractériser les échantillons. Parmi celles-ci, certaines sont particulièrement sensibles tandis que d'autres sont précises mais peu sensibles. C'est le cas notamment de la Résonance Magnétique Nucléaire (RMN) et du Raman. En effet, ces deux techniques sont des sondes locales particulièrement puissantes (1,2) mais elles souffrent malheureusement d'une mauvaise sensibilité. Tandis qu'en RMN seul un noyau sur 10^5 est visible dans les conditions habituelles (3), en Raman seul un photon sur 10^6 est observable (4).

Depuis leur découverte, ces techniques ont bénéficié de nombreux développements technologiques, ce qui a permis d'augmenter leur sensibilité. On peut citer par exemple l'augmentation du champ magnétique (5) et la rotation à l'angle magique (MAS) (6) en RMN, ou l'effet d'exaltation de surface (SERS) (7) et l'utilisation de sources non-linéaires (8) en Raman.

Malgré ces améliorations, les spectres RMN et Raman obtenus sont souvent très bruités. Cela est particulièrement vrai lorsqu'on étudie des matériaux à l'état solide, qui sont plus souvent amorphes que cristallins, comme par exemple les verres. La distribution d'environnement chimique entraîne un élargissement des pics et une perte de sensibilité. Ainsi, il est souvent nécessaire d'acquérir un spectre pendant de longues heures, voire pendant une semaine dans le cas d'échantillons compliqués, afin d'avoir un signal exploitable et un bruit limité. De telles durées entraînent des contraintes importantes, que ce soit en

termes d'occupation des appareils, de stabilité de l'échantillon, ou encore de stabilité de l'équipement.

Dans d'autres cas, les chercheurs s'intéressent à cartographier leur échantillon ou à obtenir une cinétique d'évolution, ce qui conduit à acquérir un nombre important de spectres en un temps restreint. Des traitements statistiques sont alors pertinents pour analyser les résultats (9).

Que ce soit pour un spectre unique ou pour une famille de spectres, le débruitage apparaît comme une solution intéressante pour améliorer leur qualité et pour réduire la durée des expériences nécessaires. Pour ce faire, une technique mathématique de débruitage des spectres, fondée sur la décomposition en valeurs singulières (SVD) et l'approximation de rang réduit, a été mise au point par Cadzow en 1988 (10). Elle a récemment été implémentée sous Java par Man *et al* (11), en utilisant la technologie CUDA disponible sur les cartes graphiques Nvidia, ce qui permet un gain notable en temps de traitement. Dans ces conditions, il devient tout à fait envisageable de retirer le bruit d'un spectre en quelques secondes.

La différence de temps de calcul entre le processeur et la carte graphique nous a intrigués et nous avons cherché à vérifier si une telle différence était également présente sous Matlab et Python, deux logiciels fortement utilisés par la communauté scientifique.

Au cours de cette étude, après avoir introduit le fonctionnement du SVD, nous chercherons à vérifier l'applicabilité du débruitage à un spectre RMN et à une famille de spectres Raman. Enfin, nous comparerons les temps de calcul obtenus avec les trois logiciels, sur des processeurs et des cartes graphiques datant de 2008 et de 2013.

2. Matériels et méthodes

2.1 Spectroscopies

L'échantillon et les conditions expérimentales présentées ici permettent d'obtenir rapidement un spectre modèle, le but étant de pouvoir utiliser le même protocole sur des échantillons beaucoup moins favorables.

2.1.1 RMN en phase solide

Synthèse de particules sphériques de TEOS/ MTEOS (50/50 molaire) par aérosol

Tous les produits chimiques cités ultérieurement ont été utilisés tels quels, sans purification préalable. Une solution a été préparée en mélangeant 10,18 g (57,1 mmol) de méthyltriéthoxysilane (MTEOS, 98 % Alfa Aesar, M=178,30 g.mol-1) avec 11,89 g (57,1 mmol) d'orthosilicate de tétraéthyle (TEOS > 99 %, Aldrich ; M=208.33 g.mol-1) et 29 mL d'eau osmosée milliQ ainsi que 50 mg d'une solution d'acide chlorhydrique à 37 % en masse (HCl, VWR ; M=36,46 g.mol-1).

Juste après avoir été agitée à 500 tours/min pendant au moins une heure afin d'obtenir un sol pré-hydrolysé, la solution a été atomisée et séchée sur un mini Spray Dryer B-290 (BUCHI) équipé d'un atomiseur (diamètre de la buse 0,7 mm) et d'une pompe péristaltique. Les températures en entrée et sortie de spray étaient fixées respectivement à 220°C et aux alentours de 95-120°C.

Caractérisation

L'échantillon a été analysé sur un spectromètre Bruker Avance III à 300 MHz pour le ^{1}H et 60 MHz pour le ^{29}Si. Une sonde Bruker double canal large bande ^{1}H-X à rotation à l'angle magique (MAS) a été utilisée avec un rotor de 4 mm de diamètre tournant à 14 kHz.

Le spectre a été obtenu en 2048 passages en utilisant la polarisation croisée (CP) avec un délai de relaxation de 1 s, un temps de contact ^{1}H-^{29}Si de 5 ms, suivi d'une séquence de 24 échos CPMG de 8 ms (12) synchronisés sur la rotation. La durée d'acquisition étant longue, un découplage ^{1}H de faible puissance à 7 kHz a été appliqué, correspondant à la condition $\nu_{rot}/2$ (13) pour ne pas endommager le matériel.

2.1.2 Raman

2000 spectres de 2000 points chacun ont été simulés sous python par combinaison linéaire de 4 composantes localisées à 450, 510, 750 et 900 cm^{-1}. Un bruit gaussien y a ensuite été ajouté.

2.2 Outils mathématiques et informatiques

2.2.1 Décomposition en Valeurs Singulières (SVD)

La SVD est une technique mathématique permettant de décomposer une matrice H quelconque en trois matrices dont le produit équivaut à H (Figure 1). La première, U est carrée et correspond au nombres de lignes de H ; la deuxième S est diagonale et a la même taille que H ; La troisième V^T est carrée également et correspond au nombre de colonnes de H. V^T est la transposée de la matrice V. Au sein de S, les valeurs sont classées par ordre décroissant le long de la diagonale, les valeurs les plus grandes correspondant aux signaux les plus intenses, les valeurs les plus faibles correspondant au bruit.

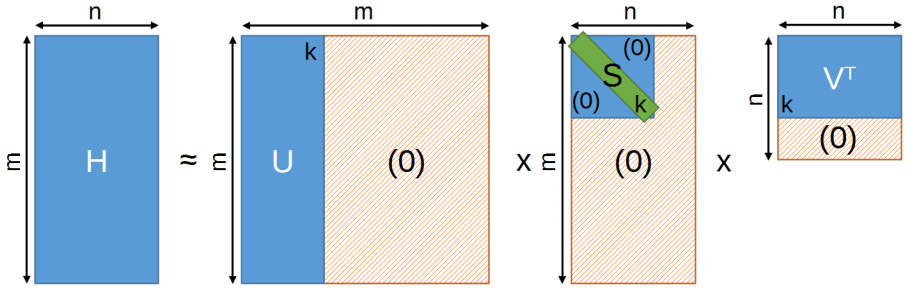

Figure 1: Décomposition en valeurs singulière d'une matrice et approximation de rang réduit

En sélectionnant seulement les k premiers points de la matrice diagonale, on ne conserve que les points correspondant aux signaux d'intérêt, au détriment du bruit. Au contraire, en supprimant les premiers points de la matrice diagonale, il est possible de supprimer un artefact ou un signal de solvant. La matrice d'intérêt est ensuite reconstruite à partir des trois matrices tronquées, ce que l'on nomme l'approximation de rang réduit. Ce procédé est également appelé méthode de Cadzow (10). Même s'il est beaucoup utilisé dans le domaine de la compression d'image (14), il reste peu utilisé dans le domaine de la spectroscopie (15,16). A noter toutefois que la SVD est une étape clé dans de nombreux traitements statistiques, tels que l'analyse en composante principale (PCA) (17).

2.2.2 CUDA et CULA

Depuis quelques années, la puissance des cartes graphiques a considérablement augmenté (18). Même si leur fonction principale est de traiter et d'afficher des images, notamment dans les jeux vidéos, Nvidia a développé à partir de 2007 la technologie CUDA (Compute Unified Device Architecture) qui permet d'effectuer des calculs scientifiques avec des fonctions facilement manipulables (19). Celles-ci se sont perfectionnées en fonction de l'architecture des cartes graphiques. Depuis CUDA 7, les cartes avec une capacité de calcul inférieure à 2 ne sont plus supportées. C'est le cas de la 8400 GS et de la GTX 260. Il faut donc utiliser un driver de la série 340.

CULA est une bibliothèque basée sur CUDA (20) qui implémente la fonction SVD sur la carte graphique. Elle est disponible en deux versions : une version payante ayant des fonctionnalités complètes et une version gratuite ne pouvant pas manipuler les nombres en double précision. Cela n'est guère dommageable vu que les cartes graphiques de la famille GeForce ont une faible capacité de calcul en double précision, contrairement aux cartes graphiques professionnelles des familles Tesla et Quadro.

2.2.3 Processeurs (CPU) et cartes graphiques (GPU) utilisés

Dans cette étude, nous disposions deux CPU à quatre cœurs, caractérisés avec CPU-Z.

- Intel Core 2 Quad Q8200 à 2330 MHz avec 4 Go de mémoire DDR2 à 400 MHz, datant de 2008, sous Linux Fedora 21 64 bits
- Intel Core i5 4670K overclocké à 4200 MHz avec 8 Go de mémoire DDR3 à 1000 MHz, datant de 2013, sous Windows 7 64 bits

Nous disposions également de trois GPU caractérisés avec GPU-Z.

- Nvidia GeForce 8400 GS à 567 MHz avec 8 cœurs et 512 Mo de mémoire à 400 MHz, datant de 2008, sous Linux Fedora 21 64 bits
- Nvidia Geforce GTX 260 à 576 MHz avec 216 cœurs et 896 Mo de mémoire à 1000 MHz, datant de 2008, sous Linux Fedora 21 64 bits
- Nvidia GeForce GTX 660 à 1085 MHz avec 960 cœurs et 2048 Mo de mémoire à 1500 MHz, datant de 2012, sous Windows 7 64 bits

2.2.4 Logiciels et protocole

L'application SVD sous Java est disponible en deux versions : CPU et GPU (21). Les versions de 2012 ont été utilisées. Pour Matlab (The MathWorks, Inc., Natick, Massachusetts, United States), c'est la version R2014a avec la Parallel Computing Toolbox disposant de la fonction gpuArray qui était disponible. Enfin, en ce qui concerne Python, c'est la version 2.7.10 qui a été installée avec le superpack numpy sous Windows et le paquet rpm numpy sous Linux ; pycuda 2015.1.3 (22) et scikit-cuda 0.5.0 ont été ajoutés. Il est important de noter que par défaut sous Windows, le pilote graphique se met en erreur lors du calcul avec le GPU. Il est donc nécessaire de modifier la clé de registre TdrLevel à 0 (23).

Afin de s'assurer de la reproductibilité des résultats, les tests ont été effectués systématiquement plusieurs fois, avec une variation de l'ordre de 0,1 s pour les calculs courts et de 1s pour les plus longs. Une attention particulière a été portée sur l'absence de tâches de fond utilisant le CPU ou le GPU, telles que les mises à jour du système ou le module flash pour le navigateur internet.

3 Résultats et discussion

3.1 Débruitage

3.1.1 Spectre RMN

La difficulté pour appliquer la SVD à un spectre RMN est qu'il n'est pas directement sous la forme d'une matrice mais plutôt sous celle d'un vecteur unidimensionnel : ce n'est qu'une série de points complexes. Il est donc nécessaire de transformer le spectre en une matrice de Toeplitz ou de Hankel. La première est définie par sa première ligne et sa première colonne, avec toutes les valeurs identiques le long des diagonales. La deuxième est définie par sa première ligne et sa dernière colonne, ce qui donne des valeurs identiques le long des anti-diagonales. Pour reconstruire le spectre après débruitage, il suffit de faire la moyenne de chaque diagonale ou anti-diagonale, selon le type de matrice.

Le rapport entre la hauteur et la largeur de la matrice peut être varié en coupant le spectre à l'endroit désiré et en mettant la première partie horizontalement et la deuxième partie verticalement. Des résultats préliminaires ont montré que le meilleur débruitage était obtenu pour une matrice carrée.

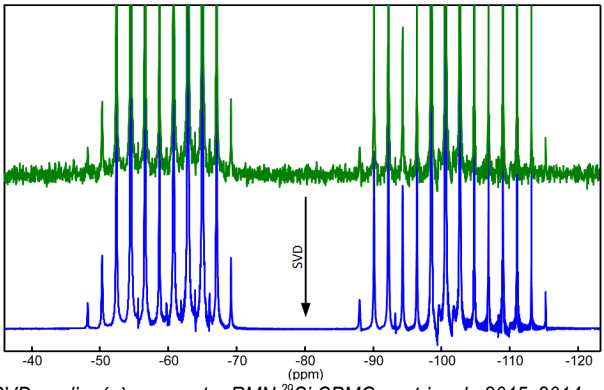

Figure 2: SVD appliqué à un spectre RMN ^{29}Si CPMG, matrice de 2015x2014 points, k=47

Dans un premier temps, le programme Java pour le GPU a été utilisé (Figure 2). Des résultats visuellement similaires peuvent être obtenus en utilisant Matlab ou Python. L'avantage du programme Java est qu'il peut directement importer et exporter les spectres RMN, contrairement à Matlab ou Python qui nécessitent un module complémentaire. Le débruitage est particulièrement efficace puisque la ligne de base devient un simple trait. Les faibles pics à -48 et -88 ppm sont bien visibles, malgré un signal sur bruit de 2-3 De nouveaux petits signaux sont mis en évidence vers -65 ppm. De même, des ondulations serrées sont visibles vers -110 ppm. Il s'agit en fait d'un artefact lié à la Transformée de Fourier du signal tronqué.

3.1.2 Série de spectres Raman

Dans un deuxième temps, la SVD a été appliquée à une série de 2000 spectres Raman avec des intensités variables (Figure 3). Dans ce cas, il n'a pas été nécessaire de transformer

le signal en matrice puisqu'il s'agissait directement d'une série de spectres. Toutefois, le nombre de spectres a été ajusté pour avoir autant de spectres que de points mesurés, et donc une matrice carrée. On notera le faible rapport signal sur bruit initial.

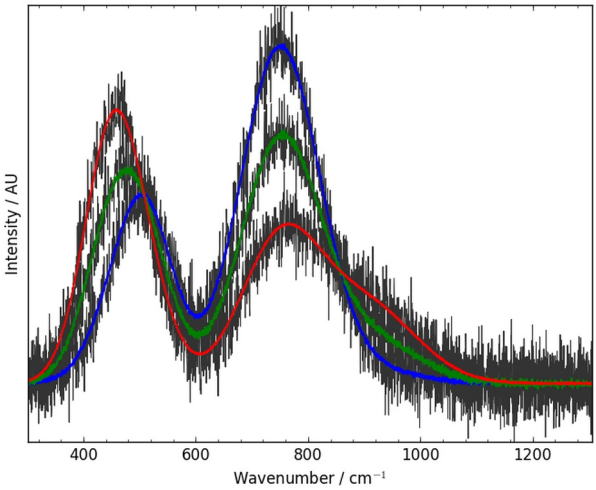

Figure 3: SVD appliqué à une série de spectres Raman, matrice de 2000x2000 points, k=2

Là encore, le débruitage est efficace et les différentes composantes sont bien respectées. Tandis que les spectres rouges et bleu présentent un bruit quasi-inexistant, il semble toujours faiblement présent sur le spectre vert. Cette différence est peut-être liée au fait que les matrices de Toeplitz ou de Hankel sont semi-circulaires, ce qui n'est pas le cas d'une série de spectres. En effet, dans une matrice semi-circulaire, à chaque ligne le signal et le bruit sont décalés d'un cran, mais ils restent identiques, ce qui permet probablement un gain en efficacité de débruitage. Dans une série de spectres, chaque ligne est totalement différente, même si les mêmes composantes sont présentes avec une intensité variable.

3.2 Temps de calcul

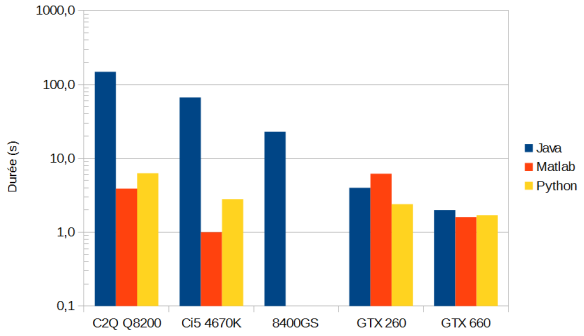

Figure 4: Temps de calcul du SVD sur CPU et GPU, en échelle logarithmique, pour une matrice de taille 1025x1024

Lors de l'utilisation des applications Java CPU et GPU, le temps de calcul passe de 149 s pour un processeur de quatre cœurs (Q8200) à 23 s pour une carte graphique d'entrée de gamme (8400 GS), soit un gain d'un facteur 6,5 (Figure 4). En fait l'application sur le CPU est purement monotâche, tandis que l'application sur le GPU utilise les quatre cœurs du CPU, au moins une partie du temps, pour distribuer le travail au GPU. L'application GPU semble donc plus optimisée.

Pour vérifier cette hypothèse, nous avons testé le même jeu de données sous Matlab. Ce logiciel est en effet connu pour sa performance en termes de calculs. Le résultat est surprenant puisque le temps nécessaire sur le même processeur n'est plus que de 4 s, soit un gain d'un facteur 37. Le calcul est ici multitâche. Avec un processeur de 2013, la durée descend même à 1s. En utilisant Python, le temps de calcul est légèrement plus grand qu'avec Matlab, mais il reste dans le même ordre de grandeur.

En ce qui concerne les cartes graphiques, les résultats les moins bons pour la GTX 260 sont obtenus par Matlab avec 6 s contre 4 s pour Java et 2 s pour Python. Matlab semble effectuer une grande partie du calcul non pas sur le GPU mais sur le CPU, ce qui le pénalise. Avec la GTX 660, les résultats sont similaires, de l'ordre d'une seconde. En effet, la mesure du temps sous Java est très imprécise puisqu'elle nécessite un chronomètre externe.

Au vu de ces résultats, il semble intéressant de pousser plus avant la comparaison entre les matériels et les logiciels, en s'interrogeant sur l'influence des optimisations utilisées par le programme, telles que le calcul multitâche ou encore des fonctionnalités SSE. On peut également se poser la question des bibliothèques utilisées, la bibliothèque Intel MKL étant particulièrement rapide.

5 Conclusion

Dans cet article, nous avons pu vérifier l'applicabilité de la technique de débruitage SVD à différentes spectroscopies, à la fois sous la forme de spectres unidimensionnels comme en RMN ou sous la forme de séries de spectres comme en Raman. Cette technique est généralisable à l'ensemble des spectroscopies et devrait permettre dans les années à venir d'atteindre des sensibilités exceptionnelles.

Jusqu'à présent, le facteur limitant était le temps de calcul nécessaire. Toutefois, il est maintenant possible de traiter des matrices de 1000x1000 en continu, soit en utilisant un algorithme optimisé, soit en utilisant les cartes graphiques. L'influence des optimisations du programme paraît une piste de réflexion intéressante.

Bibliographie

1. Efremov EV, Ariese F, Gooijer C. Achievements in resonance Raman spectroscopy: Review of a technique with a distinct analytical chemistry potential. Anal Chim Acta. 2008 Jan 14;606(2):119–34.
2. Bonhomme C, Gervais C, Laurencin D. Recent NMR developments applied to organic–inorganic materials. Prog Nucl Magn Reson Spectrosc. 2014 Feb;77:1–48.
3. Levitt MH. Spin Dynamics: Basics of Nuclear Magnetic Resonance. Second edition. John Wiley & Sons Ltd; 2008. 744 p.
4. Gautam R, Samuel A, Sil S, Chaturvedi D, Dutta A, Ariese F, et al. Raman and mid-infrared

spectroscopic imaging: applications and advancements. Curr Sci. 2015 Feb 10;108(3):341–56.

5. Bruker Corporation. 23.5 Tesla Standard-Bore, Persistent Superconducting Magnet: The World's First 1 Gigahertz NMR Spectrometer [Internet]. [cited 2015 Nov 8]. Available from: https://www.bruker.com/products/mr/nmr/magnets/magnets/avance-1000/overview.html
6. Andrew ER, Newing RA. The Narrowing of Nuclear Magnetic Resonance Spectra by Molecular Rotation in Solids. Proc Phys Soc. 1958 Dec 1;72(6):959–72.
7. Fleischmann M, Hendra PJ, McQuillan AJ. Raman spectra of pyridine adsorbed at a silver electrode. Chem Phys Lett. 1974 May 15;26(2):163–6.
8. Kano H, Segawa H, Leproux P, Couderc V. Linear and nonlinear Raman microspectroscopy: History, instrumentation, and applications. Opt Rev. 2014 Nov 27;21(6):752–61.
9. Gillis N, Plemmons RJ. Sparse nonnegative matrix underapproximation and its application to hyperspectral image analysis. Linear Algebra Its Appl. 2013 May 15;438(10):3991–4007.
10. Cadzow JA. Signal enhancement-a composite property mapping algorithm. IEEE Trans Acoust Speech Signal Process. 1988 Jan;36(1):49–62.
11. Man PP, Bonhomme C, Babonneau F. Denoising NMR time-domain signal by singular-value decomposition accelerated by graphics processing units. Solid State Nucl Magn Reson. 2014 Jul;61-62:28–34.
12. Malfait WJ, Halter WE. Increased 29Si NMR sensitivity in glasses with a Carr–Purcell–Meiboom–Gill echotrain. J Non-Cryst Solids. 2008 Sep 15;354(34):4107–14.
13. Weingarth M, Tekely P, Bodenhausen G. Efficient heteronuclear decoupling by quenching rotary resonance in solid-state NMR. Chem Phys Lett. 2008 Dec 4;466(4–6):247–51.
14. Aharon M, Elad M, Bruckstein A. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. IEEE Trans Signal Process. 2006 Nov;54(11):4311–22.
15. Brissac C, Malliavin TE, Delsuc MA. Use of the Cadzow procedure in 2D NMR for the reduction of t1 noise. J Biomol NMR. 1995 Dec 1;6(4):361–5.
16. Palacký J, Mojzeš P, Bok J. SVD-based method for intensity normalization, background correction and solvent subtraction in Raman spectroscopy exploiting the properties of water stretching vibrations. J Raman Spectrosc. 2011 Jul 1;42(7):1528–39.
17. Shlens J. A Tutorial on Principal Component Analysis [Internet]. 2014 [cited 2015 May 6]. Available from: http://arxiv.org/abs/1404.1100
18. Richer J-M. Cuda - Introduction et Historique [Internet]. [cited 2015 Nov 8]. Available from: http://www.info.univ-angers.fr/~richer/cuda_crs1.php
19. Nickolls J, Buck I, Garland M, Skadron K. Scalable Parallel Programming with CUDA. Queue. 2008 Mar;6(2):40–53.
20. Humphrey JR, Price DK, Spagnoli KE, Paolini AL, Kelmelis EJ. CULA: hybrid GPU accelerated linear algebra routines. In 2010 [cited 2015 Nov 8]. p. 770502–770502 – 7. Available from: http://dx.doi.org/10.1117/12.850538
21. Man PP. GPU SVD Java application for FID denoising with SVD [Internet]. 2014 [cited 2015 Nov 8]. Available from: http://www.pascal-man.com/navigation/faq-java-browser/SVD-Java-application-GPU.shtml
22. Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. Parallel Comput. 2012 Mar;38(3):157–74.
23. TDR Registry Keys (Windows Drivers) [Internet]. [cited 2015 Nov 8]. Available from: https://msdn.microsoft.com/en-us/Library/Windows/Hardware/ff569918(v=vs.85).aspx