Guénon Marie et Favreau Jean-Dominique

VIM / Master SSTIM

Analyse d’image

Rapport de TD3

Table des matières

[Algorithme de Metropolis 2](#_Toc373747657)

[Calcul des énergies 2](#_Toc373747658)

[Mise à jour 3](#_Toc373747659)

[Tests 4](#_Toc373747660)

[4-connexité 4](#_Toc373747661)

[8-connexité 4](#_Toc373747662)

[Annexes 5](#_Toc373747663)

# Algorithme de Metropolis

Le but de ce TP est de classifier chaque pixel d’une image pour identifier des zones caractéristiques que l’on espère homogènes. Pour chaque pixel, on donne une classe à priori. Ensuite, pour chaque pixel on tire de manière aléatoire une nouvelle classe et on calcule l’énergie associée à cette nouvelle classe et l’énergie associée à l’ancienne classe. Le but est de minimiser l’énergie associée au pixel.  
Si l’énergie nouvelle est plus faible, on garde la nouvelle classe du pixel, sinon on la garde de manière aléatoire en étant de plus en plus restrictif au fur et à mesure du temps.

Nous allons donc commencer par nous intéresser au calcul de l’énergie, puis nous verrons le critère que nous utilisons pour décider de conserver la classe ou non et nous finirons par quelques tests.

## Calcul des énergies

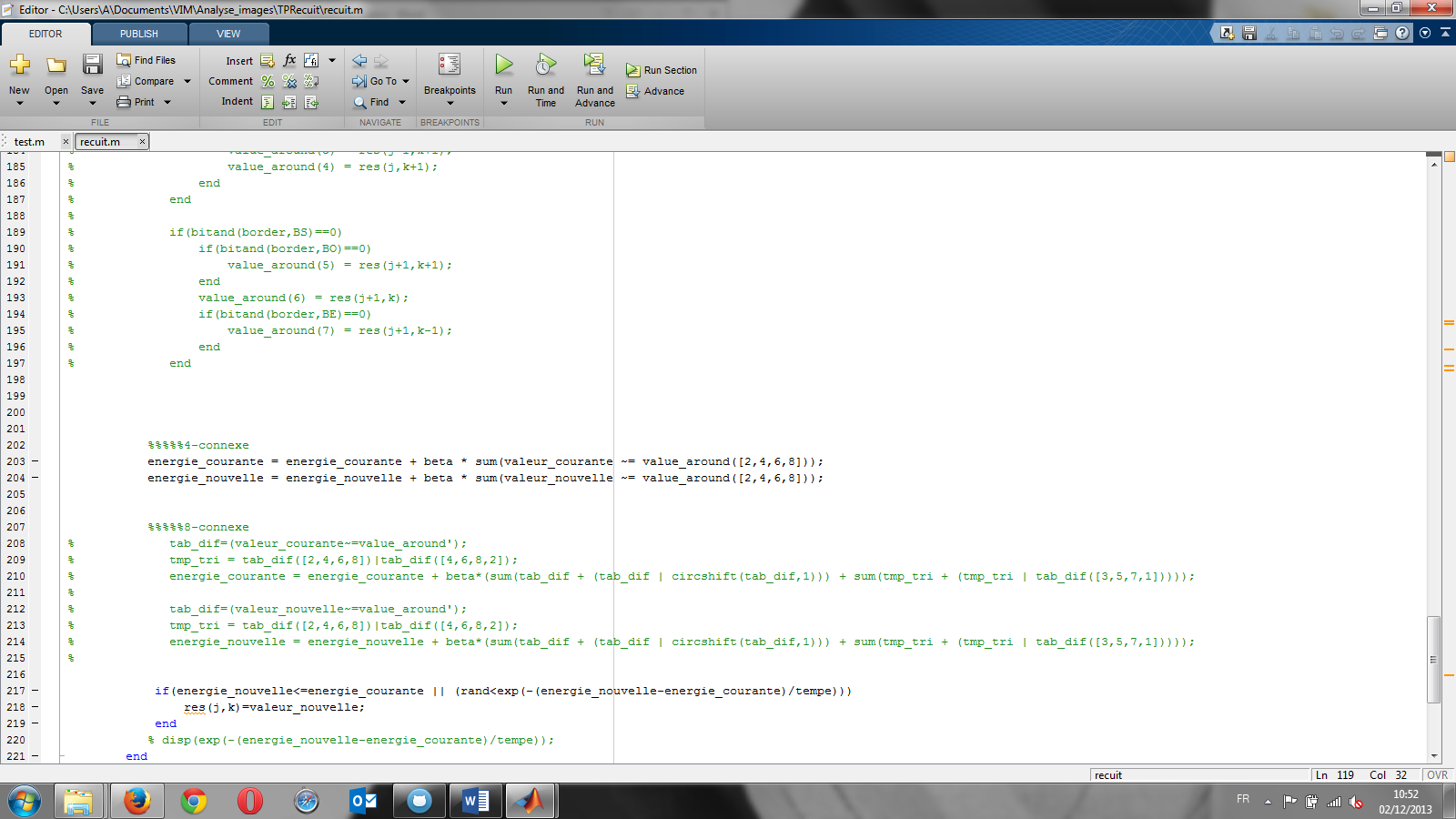
L’énergie correspondant à fidélité aux donné (vraisemblance) s’écrit :

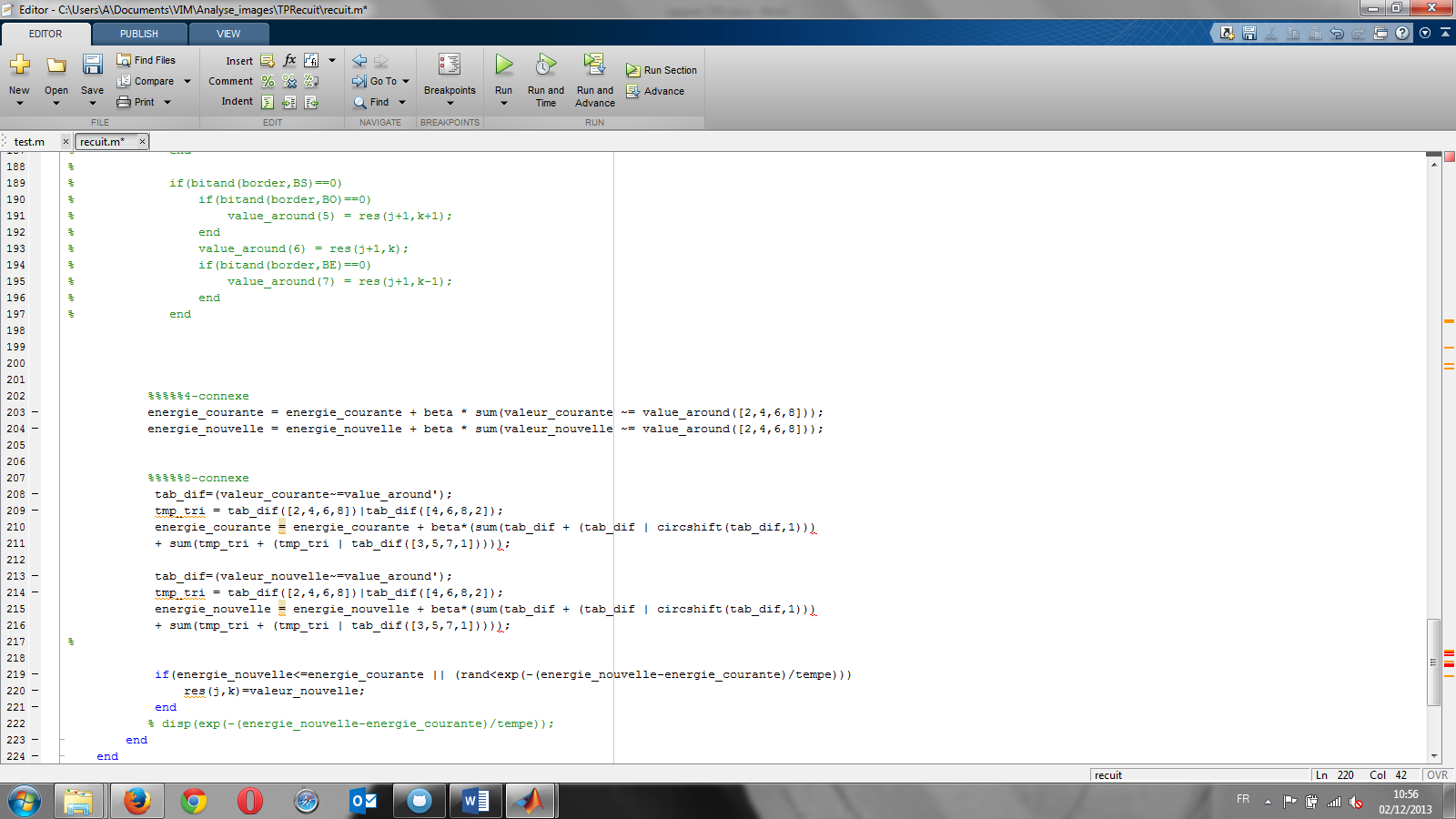
L’énergie du model à priori s’écrit :

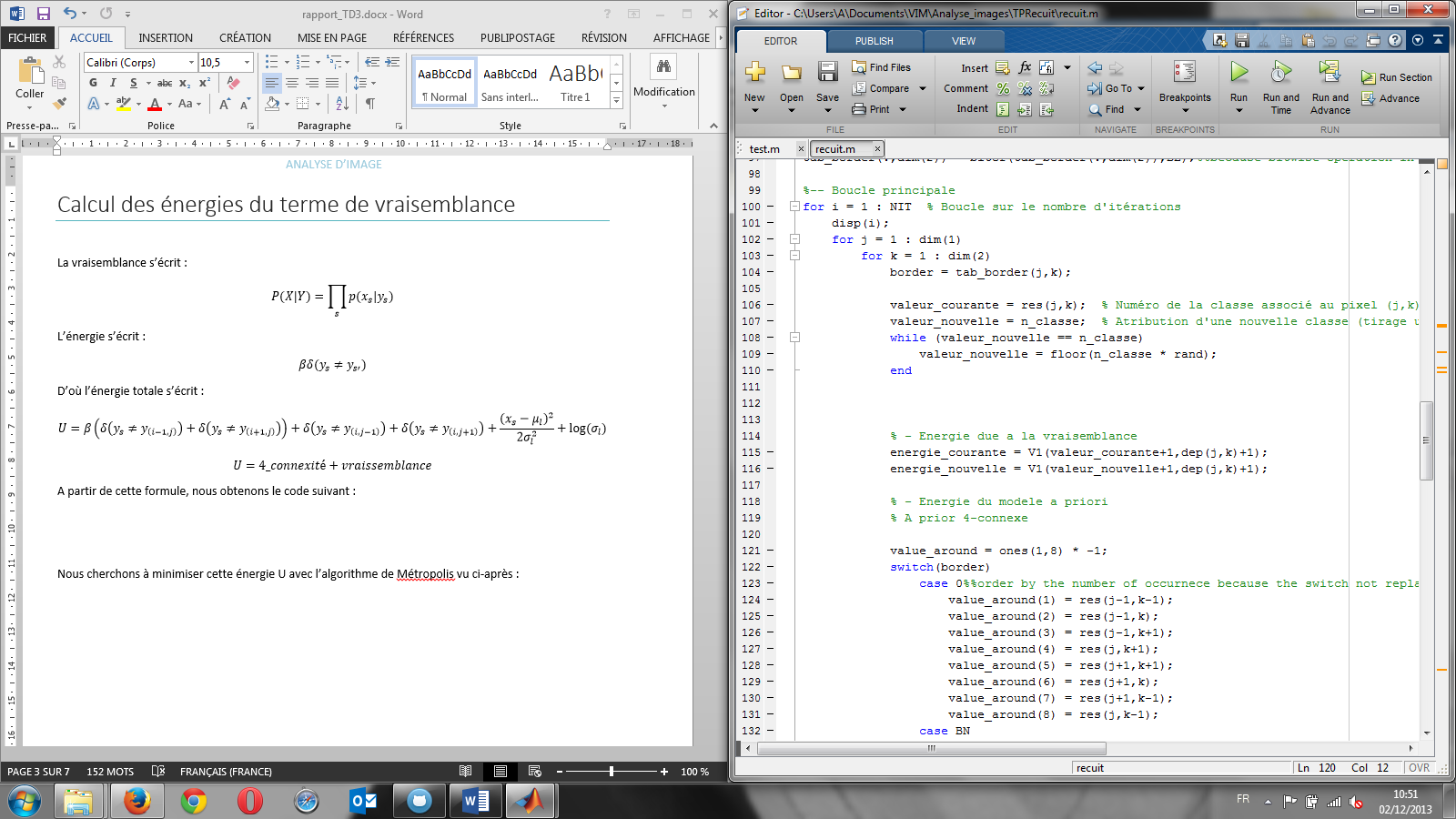
D’où l’énergie totale (en utilisant la 4-connexité) s’écrit :

On a une formule analogue pour la 8-connexité.

A partir de ces formules, nous obtenons le code suivant :







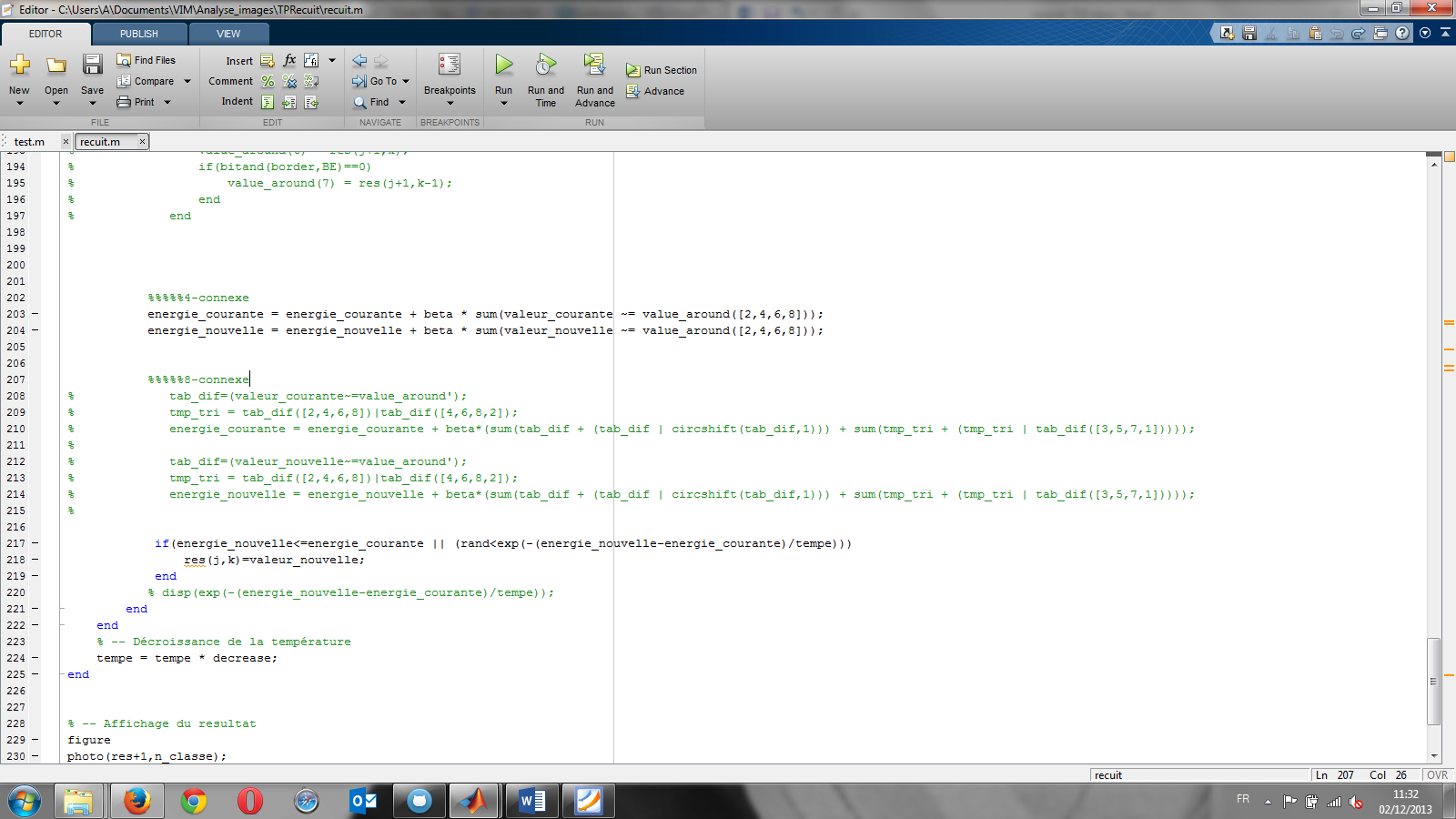
Les conditions aux limites sont gérées au préalable, avec le tableau value\_around :

Ce tableau contient la valeur de tous les voisins de x, s’ils existent, sinon il contient -1. Et on déplie de la manière suivante :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c |  |  |  |  |  |  |  |  |  |  |  |
| h | x | d |  | 🡺 |  | a | b | c | d | e | f | g | h |
| g | f | e |  |  |  |  |  |  |  |  |  |  |  |

## Mise à jour

Comme nous l’avons déjà vu, si l’énergie nouvelle est plus faible, on garde la nouvelle classe du pixel, sinon on la garde de manière aléatoire en étant de plus en plus restrictif au fur et à mesure du temps.

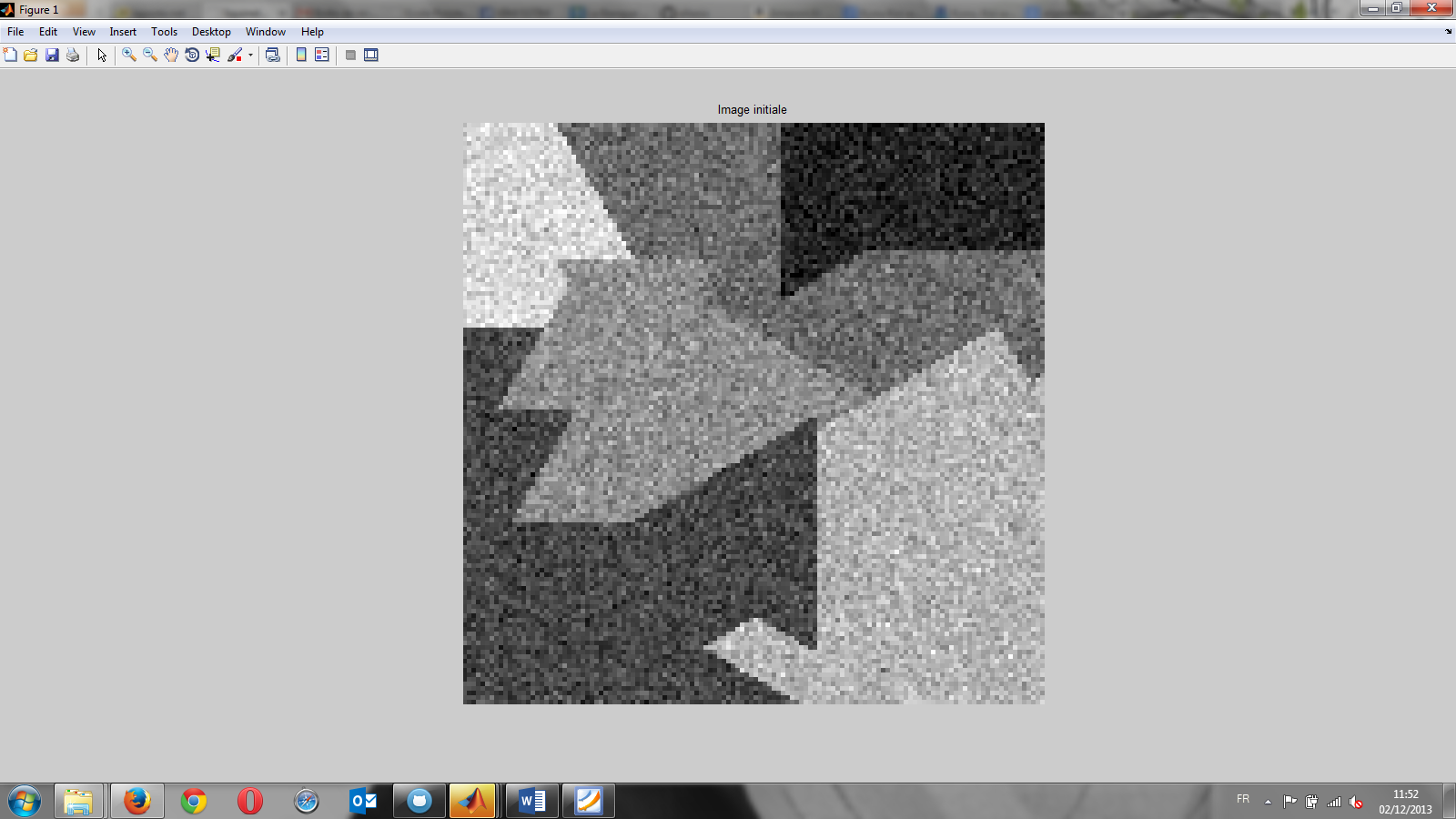


La première partie de la condition ci-dessus correspond à ce qu’on aurait dans le cas d’un algorithme variationnel (type descente de gradient…). La deuxième partie correspond à la partie stochastique de l’algorithme et est là pour permettre de visiter des "endroits" que l’on ne visiterait jamais avec un algorithme variationnel classique, ce qui nous permet d’atteindre le minimum global et pas seulement un minimum local.

# Tests

## 4-connexité

Image initiale :



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *β* | *tempe* | *decrease* | *NIT* | résultat | commentaire |
| 1.0 | 1.0 | 0.99 | 150 |  | On voit que les zones sont plutôt bien séparées, mais qu’il reste du bruit à l’intérieur de celles-ci. |
| 5.0 | 2.0 | 0.99 | 250 |  | Il y a moins de bruits ici mais les erreurs sont produites sur des zones plus larges et généralement sur les contours. Cela est dû au fait qu’on donne moins de poids à la vraissemblance. |
| 2.0 | 0.1 | 0.99 | 150 |  | Nous pouvons voir ici qu’il y a beaucoup plus de bruit. Ceci est due au fait que la température initiale est faible est nous acceptons moins de variations et nous convergeons donc plus vers des minimaux locaux. |
| 2.0 | 1.0 | 0.9 | 150 |  | Le résultat ici est similaire au cas précédent mais pour des raisons différentes. Le decrease est plus faible, et nous faisons donc diminuer la température plus rapidement, ce qui nous limite les sauts aléatoires plus vite. |
| 2.0 | 1.0 | 0.99 | 150 |  | Nous avons ici un résultat plutôt correct, même si nous pouvons constater quelques irrégularités sur les contours. |

beta = Paramètre du modèle de Potts, force des interactions entre pixels (fidélité ?)

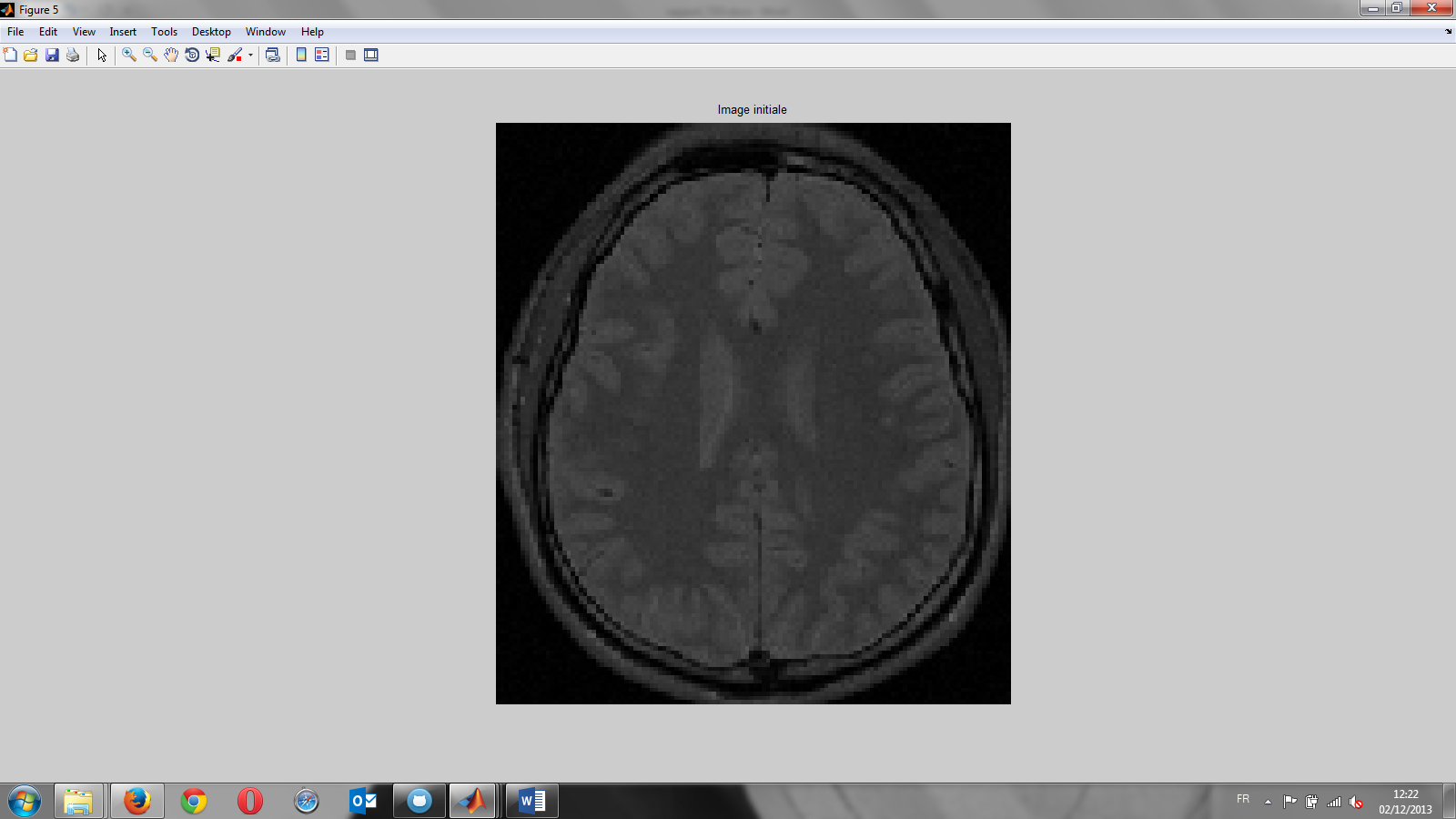
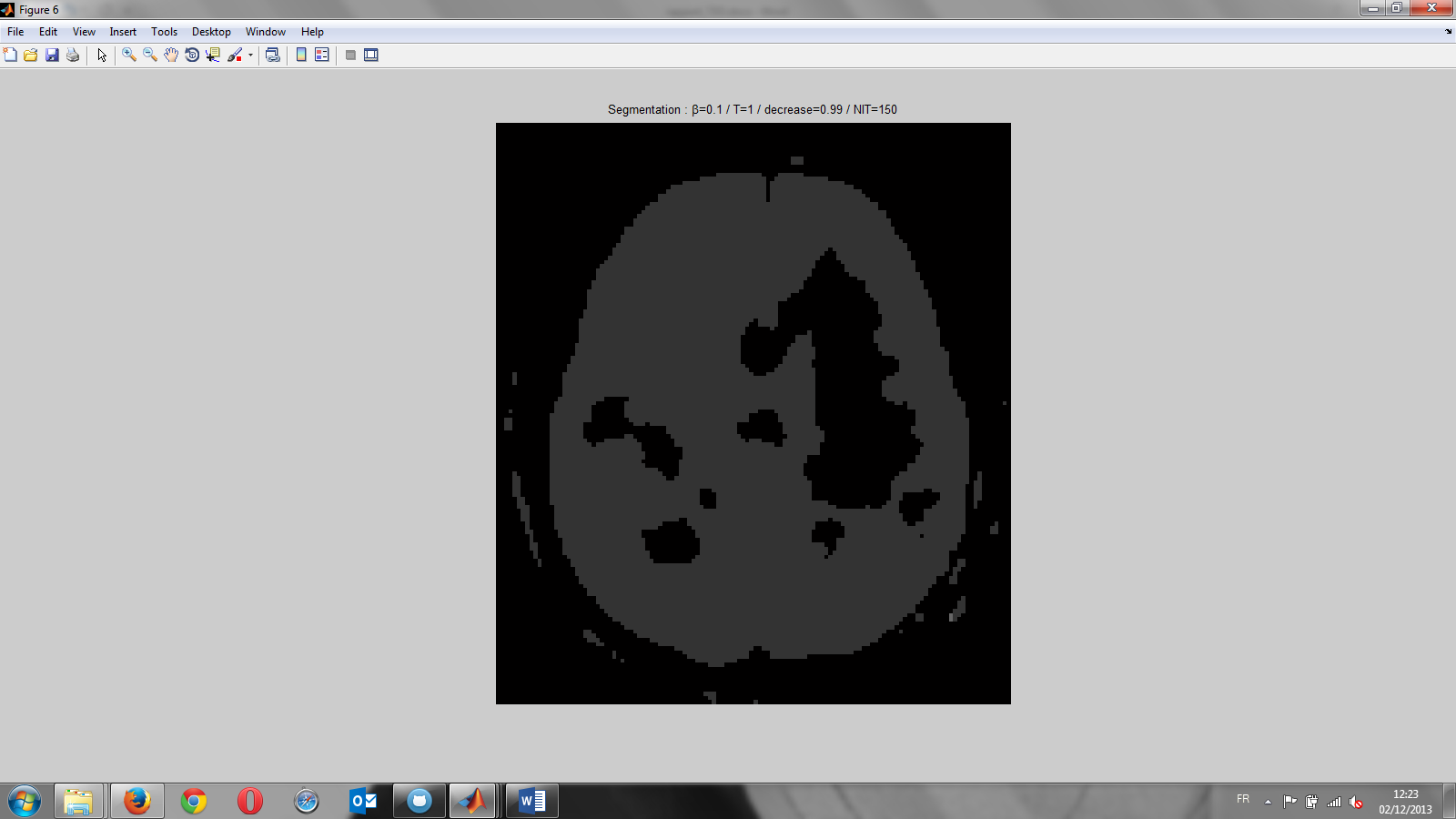
Tempe = température initiale

Decrease = Coefficient de décroissance de la température (erreur qu’on se permet)

NIT = nombre d’itérations

## 8-connexité

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *β* | *tempe* | *decrease* | *NIT* | résultat | commentaire |
| 2.0 | 1.0 | 0.99 | 150 |  | Le β est démultiplié par tous les clics de la 8-convexité et donne donc trop peu d’importance à la valeur initiale du pixel.  On remarque par ailleurs, que les problèmes sont surtout situés sur les angles et les contours. |
| 0.5 | 1.0 | 0.99 | 150 |  | Nous avons ici diminué β et donc donné plus d’importance à la fidélité des données. Et nous pouvons remarquer que nous avons de bons résultats malgré une petite erreur au milieu à gauche. |
| 0.1 | 1.0 | 0.99 | 150 |  | Nous avons continué à diminuer β. Et nous pouvons constater ici que nous n’accordons maintenant plus assez d’importance au model à priori. |

# Annexes