Compression d'image et triangulation

Nicolas Ehrhardt et David Hoffmann

8 juin 2012

1 Introduction

La compression d'image est dominée par des algorithmes basés sur les transformées de Fourier et plus récemment sur la théorie des ondelettes. L'un des désavantages souvent critiqué de la compression par stockage des coefficients de Fourier est son caractère non-localisant. La théorie des ondelettes se propose de résoudre ce problème.

Si la théorie de Fourier est valable pour tout type de signal elle ne prend pas assez en compte le caractère ordonné d'une image. En effet, les objets dans l'image ont une structure particulière : des bords et des surfaces homogènes notamment. Pour mieux rendre compte de cette structure interne, un nouveau type de compression a été introduit, il consiste à stocker un nombre de points fixés de l'image avec leur couleur sous forme de matrice creuse, ainsi qu'une triangulation associée.

1.1 Décompression d'une triangulation

On obtient alors l'image décodée de manière suivante. Pour tout point de l'image, chercher le triangle de la triangulation qui le contient, obtenir alors ses coordonnées barycentriques x_1, x_2, x_3 associées aux trois sommets du triangle. On note dans le même ordre c_1, c_2, c_3 les couleurs des trois sommets. La couleur du point en question est alors $\sum x_i c_i$.

Il est assez simple de voir que l'image reconstruite sera alors composée de triangles dans lesquels la couleur sera un dégradé entre les sommets. Ainsi, on s'attend à avoir une image lissée en quelque sorte et non pas "bruitée" dans le cas d'une compression type Fourier de mauvaise qualité.

1.2 Erreur

Plusieurs types d'erreurs peuvent être utilisés pour comparer les qualités des images. Considérons notre image comme une surface, pour chaque pixel, sa couleur (coordonnée z) est fonction de son abscisse et de son ordonnée dans dans l'image (x, y). On notera $c(x, y) = I(x, y)$ l'image originale, et $c_d(x, y) = I_d(x, y)$ l'image décompressée.

Erreur L^p

Comme indiqué, il suffit de calculer la quantité suivante pour obtenir l'erreur L^p :

$$(\sum_x \sum_y |c(x, y) - c_d(x, y)|^p)^{\frac{1}{p}}$$

Ceci étant généralisable à L^∞.

Erreur L^p en projection

Pour cette erreur, il s'agit de projeter tout point de l'image I sur l'image I_d. Dans notre cas, puisque l'image décompressée est une union de triangles on considère la distance au plan associé au triangle.

$$(\sum_x \sum_y |\pi_d c(x, y) - c_d(x, y)|^p)^{\frac{1}{p}}$$

Cette erreur est contestable nous en ferons la démonstration.

2 Algorithme de compression adaptatif

Si s'intéresser à une triangulation a des avantages sur le plan visuel, il en a aussi sur le plan théorique. On a vu en introduction que l'on stockait les points sous forme de matrice creuse pour gagner en place, mais cela ne nous affranchi pas du stockage de la triangulation, sous forme d'une matrice $n \times 3$. Hormis si l'on travaille avec une triangulation de Delaunay... Évidemment, cette triangulation étant unique, il est inutile de la stocker, elle peut très bien être reconstruite juste avant la décompression en $O(Nln(N))$. L'algorithme suivant utilise les bonnes propriétés d'une triangulation de Delaunay pour réduire l'espace de stockage nécessaire.

2.1 Principe de l'algorithme

Initialisation : On construit un maillage qui inclut tous les points de l'image (N). Chaque point ayant comme erreur associée la couleur maximale de ses quatres voisins moins la couleur minimale de ses quatre voisins.

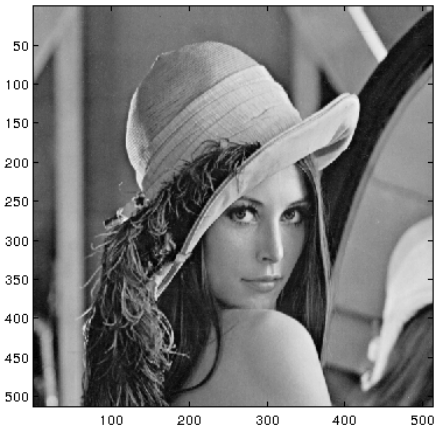
Étape i : On dispose de $N - i$ points dans notre maillage. On choisit le point auquel est associé la plus petite erreur et on l'enlève du maillage. On recalcule la triangulation de delaunay pour ses points voisins. Et on met à jour leur erreur associée.

Critère d'arrêt : On peut se donner un nombre de points à enlever, ou demander à s'arrêter lorsque l'erreur atteint un seuil.

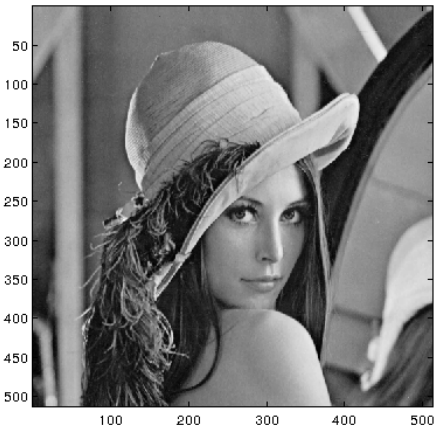
Calcul de l'erreur associée : L'erreur associée à un point est l'erreur entre l'image décompressée à partir du maillage auquel on aurait enlevé ce point et l'image initiale. Celle-ci dépend du choix de la méthode avec laquelle nous calculerons l'erreur.

2.2 Résultats

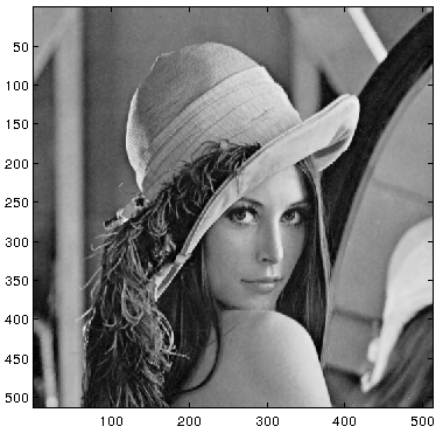
Prenons pour commencer une image avec un certain nombre de bords et de la texture. Comme d'habitude, notre amie Lena fera parfaitement l'affaire. Il faudra regarder de plus près les cheveux et le haut du chapeau pour voir apparaître les premières erreurs.



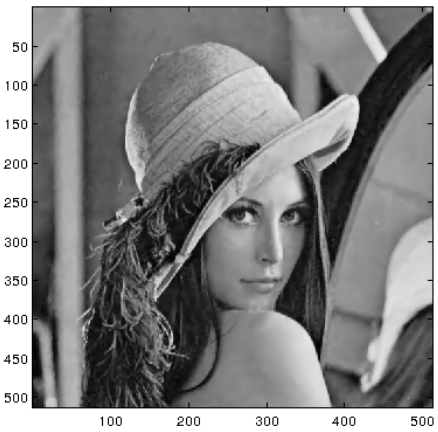
(a) Image originale



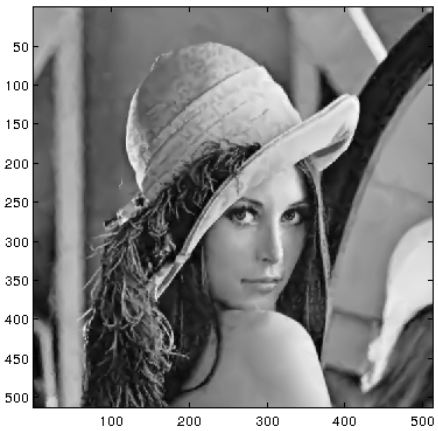
(b) 102144 Noeuds ; snr : 3.4489



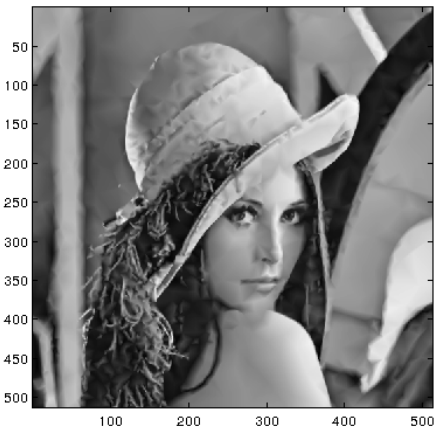
(c) 42144 Noeuds ; snr : 9.9287



(d) 12144 Noeuds ; snr : 25.9734

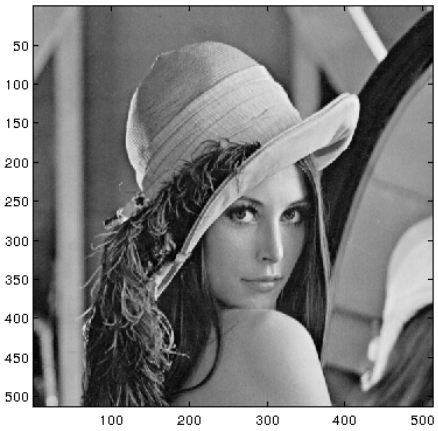


(e) 7144 Noeuds ; snr : 39.5056

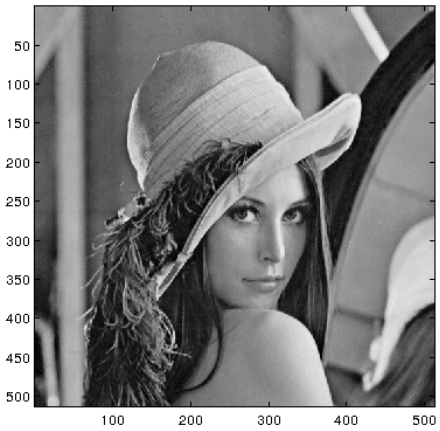


(f) 4144 Noeuds ; snr : 63.0030

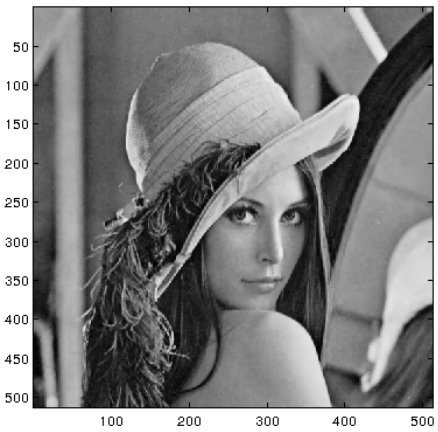
FIGURE 1 – Résultat pour l'algorithme de compression adaptatif, Erreur L^2



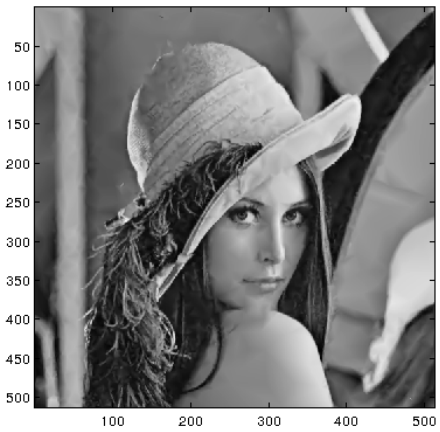
(a) Image originale



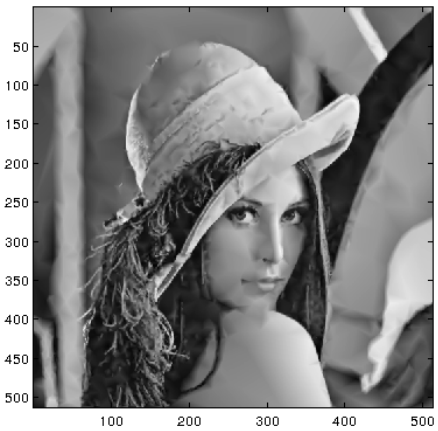
(b) 102144 Noeuds ; snr : 4.0225



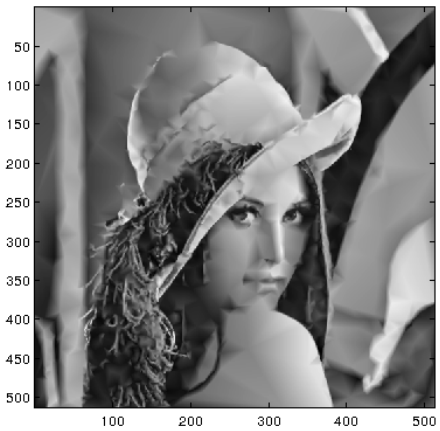
(c) 42144 Noeuds ; snr : 11.2483



(d) 12144 Noeuds ; snr : 32.0820

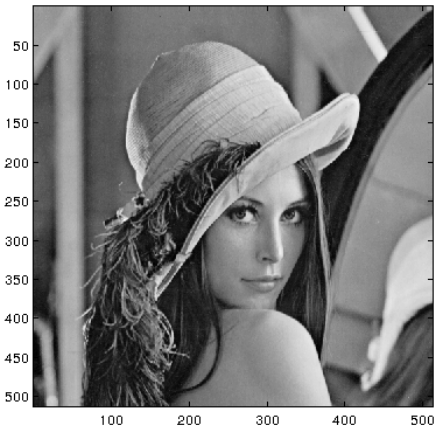


(e) 7144 Noeuds ; snr : 56.2650

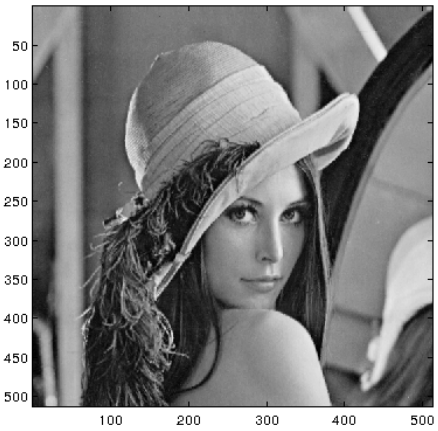


(f) 4144 Noeuds ; snr : 102.4836

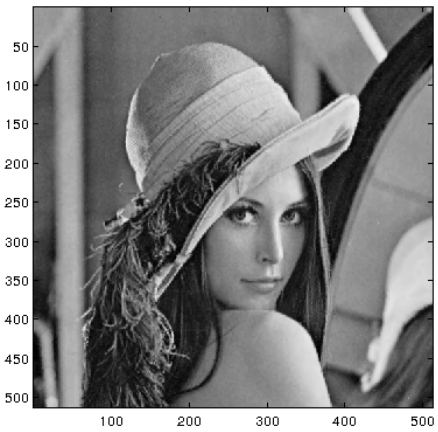
FIGURE 2 – Résultat pour l'algorithme de compression adaptatif, Erreur L^∞



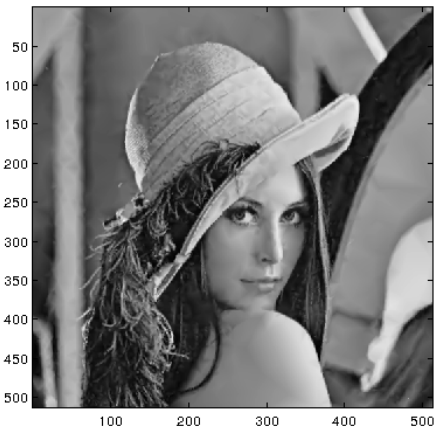
(a) Image originale



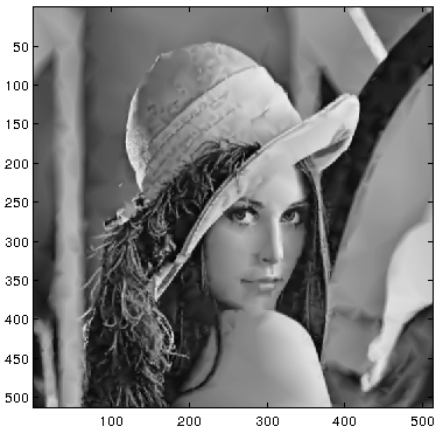
(b) 102144 Noeuds ; snr : 12.1013



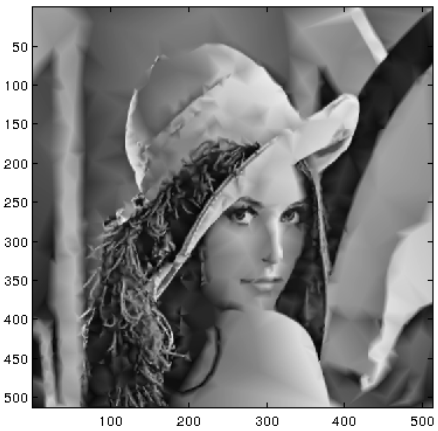
(c) 42144 Noeuds ; snr : 23.6226



(d) 12144 Noeuds ; snr : 36.7843



(e) 7144 Noeuds ; snr : 57.8413



(f) 4144 Noeuds ; snr : 105.7261

FIGURE 3 – Résultat pour l'algorithme de compression adaptatif, Erreur L^∞

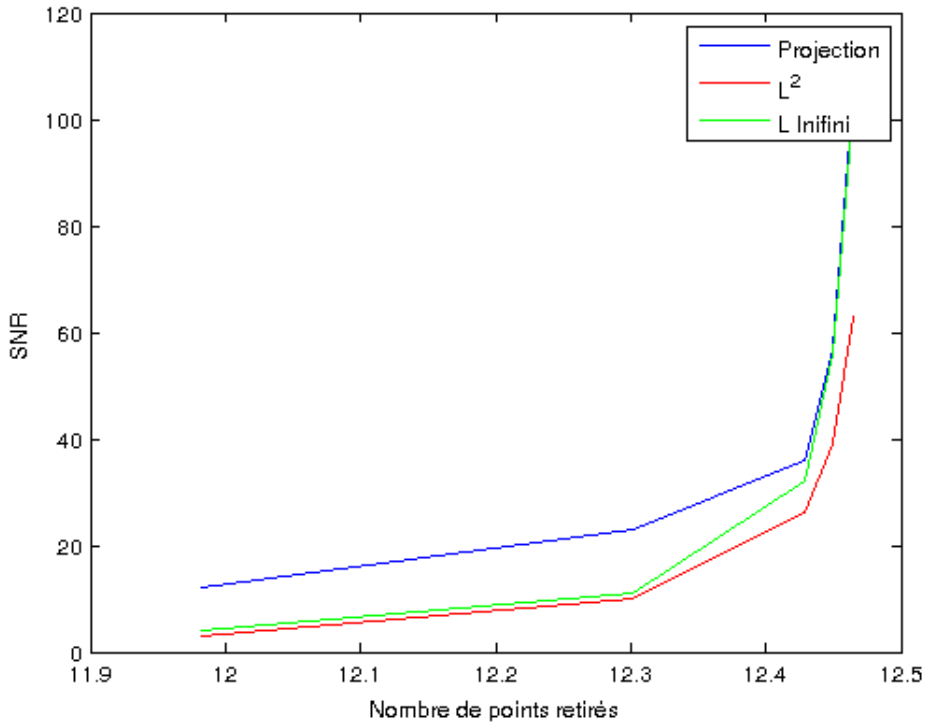


FIGURE 4 – Qualité des images compressées selon le choix du calcul de l'erreur

3 Algorithme des bissections

3.1 Principe

Si certains algorithmes partent d'une image complète mais lourde pour l'alléger en enlevant des points là où ils sont le moins utiles, d'autres partent de rien du tout et construisent pas à pas l'image en ajoutant les points. C'est le cas de l'algorithme des bissections.

Dans cet algorithme, on commence avec peu de triangles, mais de grande taille. On les coupera au fur et à mesure, aux meilleurs endroits possible.

Il s'agit de subdiviser l'image par des triangles de manière récursive. Chaque triangle en contenant deux autres triangles de même aire. Ceci à l'avantage de nous fournir une structure de stockage en arbre binaire fort agréable.

Par contre, la triangulation qui en résulte ne sera pas du tout une triangulation de Delaunay. Pire encore, ce sera même une triangulation non-conforme, c'est-à-dire que les arêtes peuvent avoir des points en plein milieu (cf. figure suivante). Ceci permet de rendre compte de manière plus simple des bords abruptes d'une image : là où l'algorithme précédent a besoin d'encadrer un bord par deux cotés d'un triangle pour effectuer un dégradé "abrupte", cette méthode n'en nécessite qu'un.

3.2 Description de l'algorithme

Initialisation : On part d'un maillage minimal avec les 4 coins et 2 triangles, comme sur la figure :

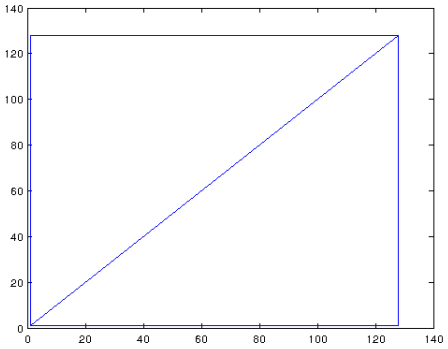


FIGURE 5 – Triangulation initiale

Étape i : On commence par chercher parmi tous les triangles quel est celui qui porte la plus grande erreur d'approximation, pour le couper en deux. Une fois ce triangle choisi, on doit déterminer quelle est la meilleure manière de le couper. En pratique, on s'autorise 3 coupes possibles, qui correspondent aux 3 médianes du triangle choisi :

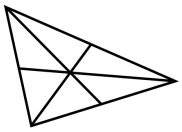


FIGURE 6 – Médianes d'un triangle

On détermine parmi ces 3 médianes laquelle est la coupe la plus avantageuse. Pour cela, dans chacune des trois possibilités, on calcule la somme des erreurs des deux triangles qui résulteraient de la coupe. On choisit bien sûr les deux triangles dont la somme des erreurs est la plus petite.

Enfin, on effectue la coupe, autrement dit :

- On ajoute un point (le sommet de la médiane qu'on doit créer)
- On enlève de la triangulation le triangle sur lequel on travaillait
- On ajoute à la place les deux triangles issus de la coupe

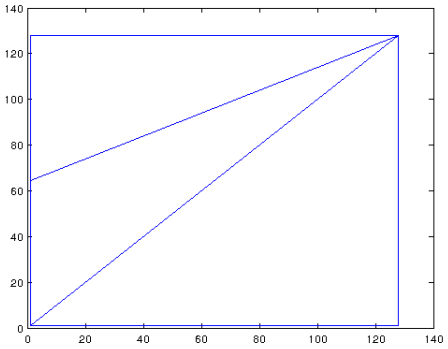


FIGURE 7 – L'un des triangles a été coupé selon sa médiane pour en former deux nouveaux

Critère d'arrêt : On se donne un nombre de points à ajouter.

3.3 Résultats

Pour pouvoir comparer le rendu avec ce qui s'est fait à l'algorithme précédent, on reprend notre chère Lena, que l'on bissecte sans vergogne. Malgré tout, on a beau augmenter le nombre de points, le résultat n'est jamais aussi probant que dans la compression adaptative. On remarque en particulier que l'erreur ne tend pas vers 0, ce qui est fâcheux.

//////////////// HUM...

Les résultats de cet algorithme n'ont pas été aussi satisfaisants que dans la compression de Delaunay. En effet, il arrive assez vite dans un minimum local dont il n'arrive pas à sortir. En effet, la manière dont on choisit la coupe dans l'algorithme des bissections va causer le mécanisme suivant :

- On arrive à un triangle presque plat, qui contient beaucoup de points alignés sur son bord ainsi qu'un unique autre point (le 3^e sommet).
- Lorsque ce triangle finit par être celui qui a la plus grande erreur cumulée, on le choisit pour le couper.
- Malheureusement, il se peut que la coupe qui sépare les points alignés augmente l'erreur au lieu de la diminuer, donc on prendra systématiquement l'une des autres coupes, qui ne les séparera pas du tout. Et on est ramené à la situation initiale !

Dans cette situation, on voit donc que le triangle fautif sera coupé à tous les coups, et le reste de l'image restera tel quel. On observe d'ailleurs ces structures en éventail lorsqu'on regarde la triangulation obtenue :

////// Figure de la triangulation

4 Comparaison des deux algorithmes

4.1 Stockage

On cherche tout d'abord quel espace mémoire sera nécessaire au stockage d'un maillage de n points.

Compression adaptative Dans l'algorithme de compression adaptative, il suffit de stocker les points et leur couleur, puisqu'on peut retrouver la triangulation de Delaunay facilement. D'où la simple constatation : pour stocker n points, il faut donc stocker $3n$ entiers.

Bissections En revanche, pour l'algorithme des bissections, il faut aussi stocker la triangulation (une liste de triangles). Or, à chaque itération de l'algorithme, on rajoute un point et un triangle. Comme on commence avec 4 points et 2 triangles, on obtient que pour stocker n points, il faudra aussi stocker $n-2$ triangles. On devra donc stocker $3*(n-2)$ points pour les triangles en plus des $3n$ entiers pour les points eux-mêmes. Au total, il faut donc stocker $3n+3(n-2) \sim 6n$ entiers.

Comparaison A nombre de points égal, on prend donc 2 fois plus de place avec la méthode des bissections. Pour pouvoir comparer équitablement les deux méthodes, on doit considérer des images avec des nombres de points différents : 2 fois plus de points pour la méthode adaptative que pour la méthode avec bissections.

On note toutefois qu'il pourrait exister des méthodes de stockage plus efficaces pour les bissections avec des arbres binaires.

4.2 Erreur

A taille de stockage équivalente, on peut légitimement comparer l'erreur et voir lequel est le plus efficace selon l'erreur L^p :

4.3 Exemples