

"Zoom" en 3D : une première toolbox Matlab

Emilie Koenig

31 mars 2010

Une première toolbox Matlab a été créée en généralisant au 3D les fonctions utilisées pour le zoom 2D.

Je présenterai dans un premier temps les différentes fonctions de test utilisées pour générer les résultats présentés ci-après. Je parlerai ensuite des limitations actuelles de la toolbox, en partie liées à l'utilisation sous Matlab. Enfin, je décrirai avec plus de précision les fonctions utilisées pour la réalisation d'un zoom 3D sous Matlab.

Les images présentées par la suite sont des exemples de réalisations. La visualisation a été réalisée soit avec Matlab, soit avec un outil de visualisation réalisé par des élèves de l'ISIMA. Cet outil de visualisation construit le champ 3D en créant des isosurfaces successives transparentes par la méthode des Marching Cubes. Donc, contrairement à la visualisation réalisée avec Matlab qui ne montre qu'une seule isosurface, nous pouvons en visualiser un nombre prescrit avec le programme appelé "champ".

1 Les fonctions de test

Deux fichiers de tests ont été créés. Ces fichiers de tests permettent de réaliser l'augmentation d'un champ 3D initial cubique ou non. Ce champ initial est soit un champ 3D synthétique connu (créé à partir d'une cascade de Poisson composée 3D), soit un champ 3D donné tel qu'un nuage (créé par une simulation ou réel).

1.1 Champ initial : cascade de Poisson composée 3D

Le premier fichier, "test_zoom_3D.m", permet de réaliser l'augmentation de résolution d'un champ 3D créé par une cascade de Poisson composée 3D (voir partie 3.1). Pour plus de facilités, ce champ est cubique.

Les figures 1 et 2 représentent les mêmes champs 3D. Les figures (a) représentent un champ CPC 3D initial, $32 \times 32 \times 32$. Les figures (b) et (c) sont des versions augmentées d'un facteur $x2$ et $x2^2$ du champ initial. Nous pouvons parfaitement y voir l'évolution des isosurfaces qui deviennent de plus en plus "granuleuses" dans la visualisation Matlab et de plus en plus "nuageuse" avec l'autre outil de visualisation.

Les figures 3 et 4 représentent la même évolution que les figures 1 et 2, mais partant cette fois-ci d'un objet 3D déterministe. Ici, les figures (a) illustrent le champ initial (toujours $32 \times 32 \times 32$) qui est une "boule exponentielle". Les détails ajoutés lors des augmentations successives ont les mêmes caractéristiques que ceux ajoutés pour créer les figures 1 et 2. Nous pouvons donc observer soit une boule qui devient gouttelettes (figure 3), soit un nuage de plus en plus morcelé (figure 4).

1.2 Champ initial : simulation d'un cirrus

Le premier fichier, "test_zoom_3D_cirrus.m", permet de réaliser l'augmentation de résolution d'un cirrus 3D synthétique. Ce champ initial est issu de données créées par F. Szczap (pour plus de renseignements sur ces données, voir le document "Plan_travail_1.doc"). Seule une partie du champ entier est traitée. Ce champ initial n'est pas nécessairement cubique. Les données et documents concernant cette modélisation de cirrus 3D se trouvent dans le dossier "modele_cirrus".

Les figures 5 et 6 représentent l'augmentation successive de la résolution d'un champ initial issu de la simulation d'un cirrus 3D.

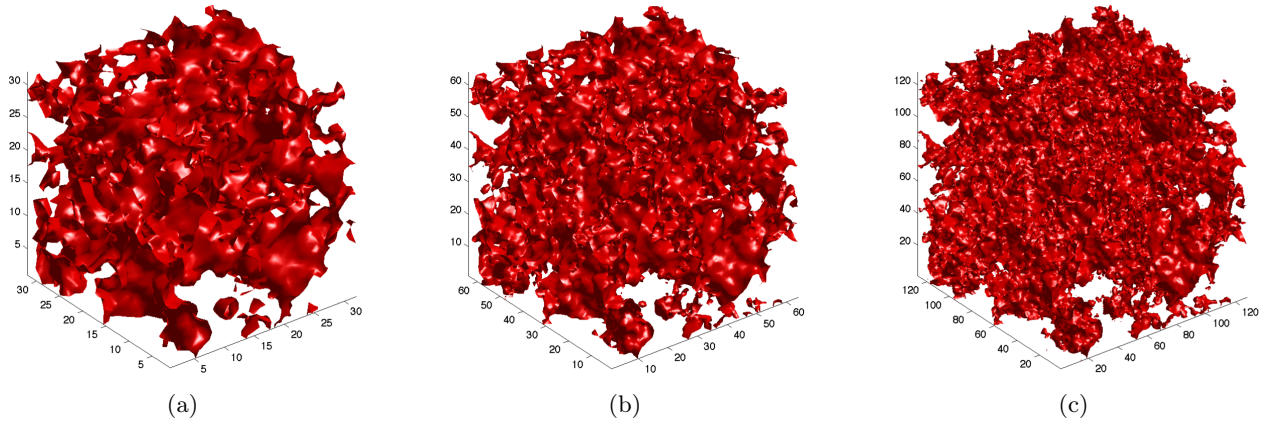


FIG. 1 – (a) Champ 3D $32 \times 32 \times 32$ initial ; (b) et (c) augmentations successives de la résolution d'un facteur 2 à chaque étape. Taille finale du champ 3D (c) : $128 \times 128 \times 128$. Visualisation Matlab

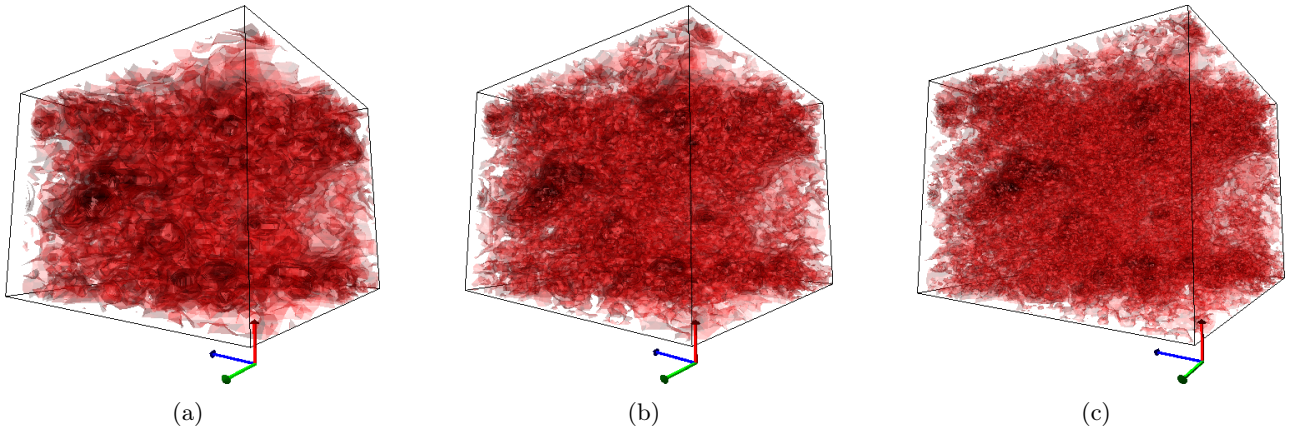


FIG. 2 – (a) Champ 3D $32 \times 32 \times 32$ initial ; (b) et (c) augmentations successives de la résolution d'un facteur 2 à chaque étape. Taille finale du champ 3D (c) : $128 \times 128 \times 128$. Visualisation à l'aide des Marching Cubes

2 Les limitations actuelles

Actuellement, les limites principales viennent du fait que cette augmentation de résolution est réalisée sous Matlab. En particulier, la création de détails par cascades de Poisson composées 3D ne peut pas dépasser un cube de taille $256 \times 256 \times 256$. Ceci est dû à la façon dont Matlab gère la mise en mémoire d'un champ 3D. Donc au maximum, nous pouvons réaliser une augmentation de résolution telle que le champ final ne dépasse pas une taille de 256^3 . La solution est alors de re-développer notre méthode en C ou C++.

3 Les fonctions de la toolbox

3.1 Création d'un champ par cascades de Poisson composées

La fonction CPC 3D utilisée ici est une fonction codée en C et interfacée avec Matlab. Les codes sources se trouvent dans le dossier "CPC_3D". Exécuter le fichier make depuis Matlab permet de créer l'exécutable "cpc3d_zoom" utilisé par la suite.

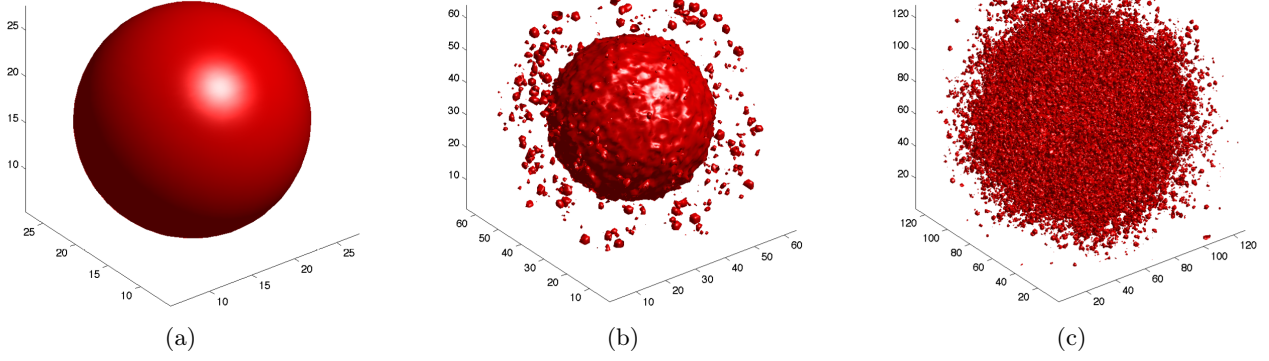


FIG. 3 – (a) Champ 3D $32 \times 32 \times 32$ initial : "boule exponentielle"; (b) et (c) augmentations successives de la résolution d'un facteur 2 à chaque étape. Taille finale du champ 3D (c) : $128 \times 128 \times 128$. Visualisation Matlab

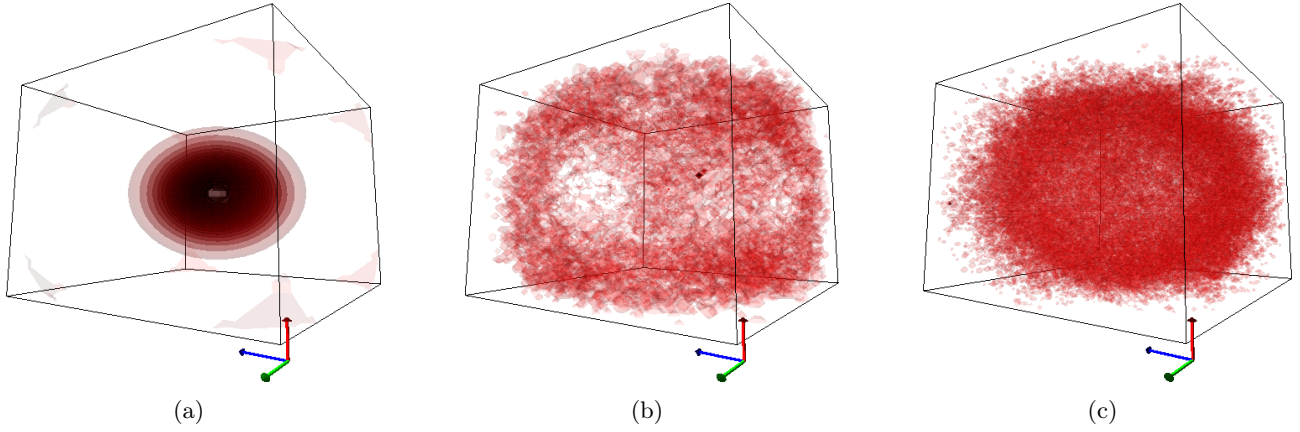


FIG. 4 – (a) Champ 3D $32 \times 32 \times 32$ initial : "boule exponentielle"; (b) et (c) augmentations successives de la résolution d'un facteur 2 à chaque étape. Taille finale du champ 3D (c) : $128 \times 128 \times 128$. Visualisation à l'aide des Marching Cubes

3.2 Integration et derivation fractionnaires 3D

Pour ces deux fonctions, j'ai simplement remplacé la transformée de Fourier 2D originelle par la transformée de Fourier généralisée proposée par Matlab. Ces deux fonctions sont codées dans les fichiers "integfrac3D.m" et "derivfrac3D.m".

3.3 Interpolation 3D par splines

Pour le moment, cette fonction "interp_3D" se base sur la fonction "mexinterp" d'interpolation 1D et 2D créée par Unser et Tevenaz. Dans un premier temps, le champ 3D est découpé en plans successifs qui sont chacun interpolés par la version 2D de la fonction. Ensuite, le champ résultant est interpolé le long de toutes ses colonnes. Ceci permet de créer un champ 3D deux fois plus grand que le champ d'origine.

3.4 Ajout des détails et procédure de zoom

Le zoom est réalisé par la fonction "zoom_x2_3D". L'ajout des détails est réalisé de la même façon que dans la version 2D de la procédure de zoom. La principale différence est que ces détails sont générés avec la fonction "cpc3d_zoom" décrite précédemment.

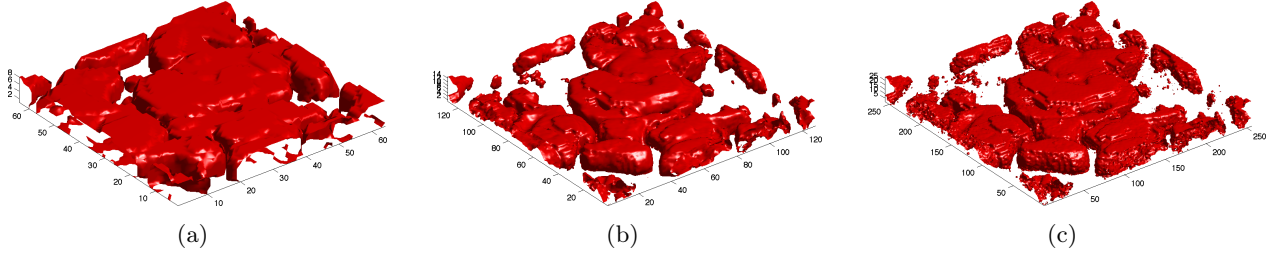


FIG. 5 – (a) Champ 3D $64 \times 64 \times 8$ initial : modélisation d'un cirrus; (b) et (c) augmentations successives de la résolution d'un facteur 2 à chaque étape. Taille finale du champ 3D (c) : $256 \times 256 \times 32$. Visualisation Matlab

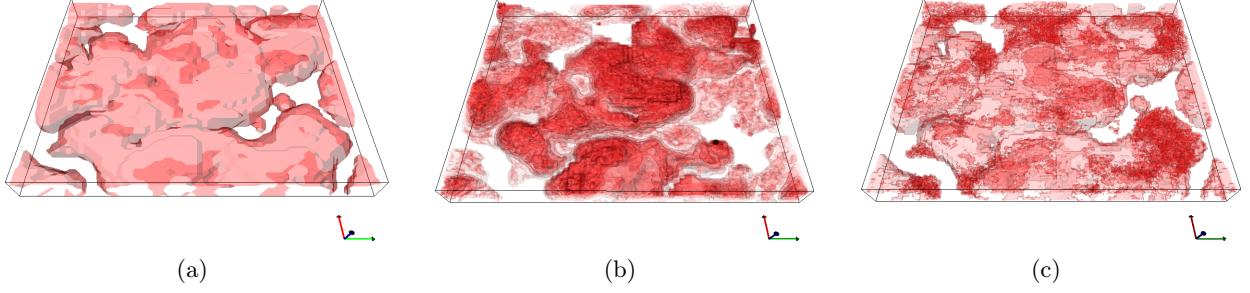


FIG. 6 – (a) Champ 3D $64 \times 64 \times 8$ initial : modélisation d'un cirrus; (b) et (c) augmentations successives de la résolution d'un facteur 2 à chaque étape. Taille finale du champ 3D (c) : $256 \times 256 \times 32$. Visualisation à l'aide des Marching Cubes

3.5 Conservation

L'étape de conservation est réalisée, comme dans la version 2D, en appliquant une carte de conservation au champ 3D augmenté. Un zoom arrière (une décimation) du champ 3D augmenté est obtenu, grâce à la fonction "zoom_arriere_fois2_3D" en réalisant une convolution généralisée du champ par une boîte 2D. Ce champ est ensuite comparé au champ d'origine. La différence des deux, créant la carte, est ensuite interpolé avec la fonction "interpol_image_fois2_3D" qui utilise la fonction d'interpolation généralisée proposée par Matlab.