Instructions pour le T.P. du master M2 de Physique - Option Problèmes inverses - Décembre 2012

Déconvolution d'image

Filtre de Wiener approché

1. Lire l'image de Saturne par le HST (saturn.fits) et la réponse impulsionnelle (saturn_psf.fits).
2. Afficher ces deux images.
3. Dans un premier temps, nous allons utiliser une version simplifiée du filtre de Wiener :

$$\hat{x}_\nu = \frac{\hat{h}^*_\nu \ \hat{y}_\nu}{|\hat{h}_\nu|^2 + \alpha \ |\hat{h}_0|^2}$$

   Expliquer ce que représentent les différentes variables de la formule ci-dessus. Quel est le rôle du terme $\alpha \ |\hat{h}_0|^2$? Donner le pseudo-code (en Yorick) correspondant à ce filtrage.
4. Discuter des modalité d'application sur l'exemple de l'image de Saturne. Coder une fonction effectuant ce filtrage et régler la valeur de α de façon *optimale*.

Approche inverse par méthode itérative

Nous allons résoudre le problème de la déconvolution en minimisant la fonction de coût :

$$f_{\text{post}}(\boldsymbol{x}) = f_{\text{data}}(\boldsymbol{x}) + f_{\text{prior}}(\boldsymbol{x})$$

avec

$$\begin{aligned} f_{\text{data}}(\boldsymbol{x}) &= (\mathbf{H}\cdot\boldsymbol{x}-\boldsymbol{y})^{\mathrm{T}}\cdot\mathbf{W}\cdot(\mathbf{H}\cdot\boldsymbol{x}-\boldsymbol{y}) \\ &= \sum_i w_i \left[(h*x)_i - y_i\right]^2 \end{aligned}$$

où **H** est l'opérateur de convolution par *h* et

$$\begin{aligned} f_{\text{prior}}(\boldsymbol{x}) &= \mu \ \|\mathbf{D}\cdot\boldsymbol{x}\|^2 \\ &= \mu \sum_{i,j}\left[x_{i+1,j}-x_{i,j}\right]^2 + \mu \sum_{i,j}\left[x_{i,j+1}-x_{i,j}\right]^2 \end{aligned}$$

avec **D** un opérateur linéaire (de type *différences finies*).

1. En utilisant les expressions algébriques, montrer que le minimum de la fonction de coût est la solution d'un système linéaire d'équations de la forme :

$$\mathbf{A}\cdot\boldsymbol{x} = \boldsymbol{b}$$

   Donner les expressions de **A** et de ***b***.
2. Écrire un code Yorick qui calcule **A**.***x*** étant donné ***x***. Il faut utiliser la FFT pour la convolution ; pour appliquer l'opérateur $\mathbf{D}^{\mathrm{T}}.\mathbf{D}$, on propose la fonction suivante :

```
func DtD(x) {
  r = array(double, dimsof(x)) ; // pour stocker le resultat
  dx = x(dif,); // differences finies pour la 1ere dimension
  r(2:0,) += dx;
  r(1:-1,) -= dx;
  dx = x(,dif); // differences finies pour la 2eme dimension
  r(,2:0) += dx;
  r(,1:-1) -= dx;
  return r;
}
```

3. La méthode des gradients conjugués linéaires est un algorithme simple et efficace (rapide et peu gourmand en mémoire) pour résoudre de façon itérative un système linéaire d'équations de la forme : $\mathbf{A}\cdot\boldsymbol{x} = \boldsymbol{b}$ où le « *vecteur* » ***x*** contient les inconnues et la matrice **A** est positive définie positive. Est-ce les conditions de notre problème ? Les ingrédients de la méthode sont :
   - un opérateur réalisant l'opération ***x*** → **A**.***x*** ;
   - une estimation initiale de ***x*** ;
   - le « *vecteur* » ***b*** ;

   La fonction **cg** dans **tp-utils.i** permet d'appliquer les gradients conjugués.