

# SOD333 : Filtrage Bayésien

Bechir Trabelsi  
Mohamed Mkaouar  
Kais Cheikh

Octobre 2020

## Introduction

Un avion survole une zone dont le relief est connu : la hauteur  $h(r)$  du relief en chaque point de coordonnée horizontale  $r$  est connue, et enregistrée dans une carte numérique. La figure 1 montre un image 3D de hauteur de la région.

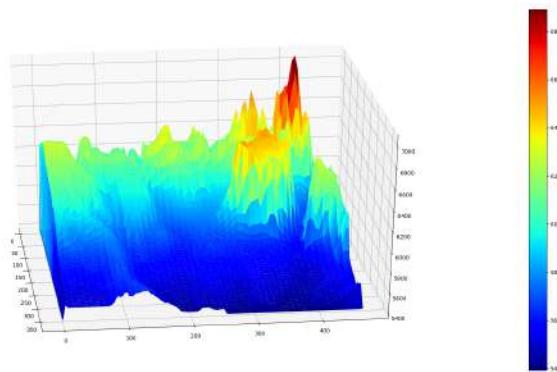


Figure 1: Relief 3D de la zone survolée

## Objectif

Dans ce TP, On désire approximer la position horizontale et la vitesse de l'avion étant donné l'ensemble de mesures bruitées (accélération horizontale, altitude) effectuées périodiquement le long d'une durée  $T$ .

## Modélisation du problème

Dans la suite, la position horizontale de l'avion est notée  $r$ , la position verticale, ou altitude, est notée  $z$ , et la vitesse horizontale est notée  $v$ . A l'instant 0, la position horizontale initiale de l'avion est  $r_0$ , son altitude initiale est  $z_0$  et sa vitesse horizontale initiale est  $v_0$ .

Le code ci dessous correspond à l'initialisation des variables du problème.

```

1 X1MIN,X1MAX = -1e4 , 1e4
2 X2MIN,X2MAX = -1e4 , 1e4
3 r0 = (-6000,2000)
4 v0 = (120,0)
5 sigma_r0 = 100
6 sigma_v0 = 10
7 sigma_INS = 7
8 sigma_ALT = 10
9 sigma_BAR = 20
10 Delta = 1
11 T =100

```

**Q1. Représentation pour chaque instant entre 0 et T de la position horizontale exacte  $r_k$  de l'avion.**

```

1 map = io.loadmat('mnt')['map']
2 N1 = map.shape[1]
3 N2 = map.shape[0]
4
5 plt.imshow(map,cmap='jet',extent=[X1MIN,X1MAX,X2MIN,X2MAX])
6
7
8 # Plotting the trajectory :
9 traj = io.loadmat('traj.mat')
10 rtrue = traj['rtrue']
11 vtrue = traj['vtrue']
12
13 plt.plot(rtrue[0,:],rtrue[1,:],'r-',label='Real')

```

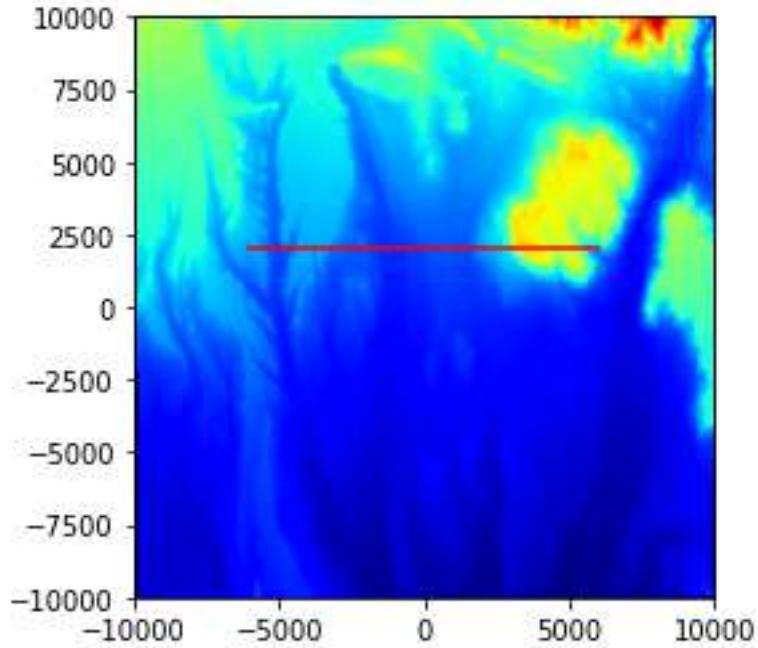


Figure 2: Modèle numérique de terrain et trajectoire réelle

**Q2. Représentation sur le même graphique pour chaque instant entre 0 et  $T$  de la position horizontale exacte  $r_k$  de l'avion, et son estimation inertielle  $r_k^{\text{INS}}$**

En exploitant les mesures de l'accélération horizontale  $a^{\text{INS}}$  et en utilisant la formule d'intégration, on peut calculer la trajectoire estimée de l'avion qui est représentée dans la figure obtenue grâce au code suivant :

```

1 # Reading the INS values :
2 a_INS = io.loadmat('ins.mat')['a_INS']
3
4 nmax = np.shape(a_INS)[1]
5 r_INS = np.zeros(rtrue.shape)
6 v_INS = np.zeros(vtrue.shape)
7 r_INS[:,0] = r0
8 v_INS[:,0] = v0
9 for k in range(1,nmax):
10     r_INS[:,k] = r_INS[:,k-1]+Delta*v_INS[:,k-1]
11     v_INS[:,k] = v_INS[:,k-1]+Delta*a_INS[:,k-1]
12
13 plt.plot(r_INS[0,:],r_INS[1,:], 'm-', label='INS')

```

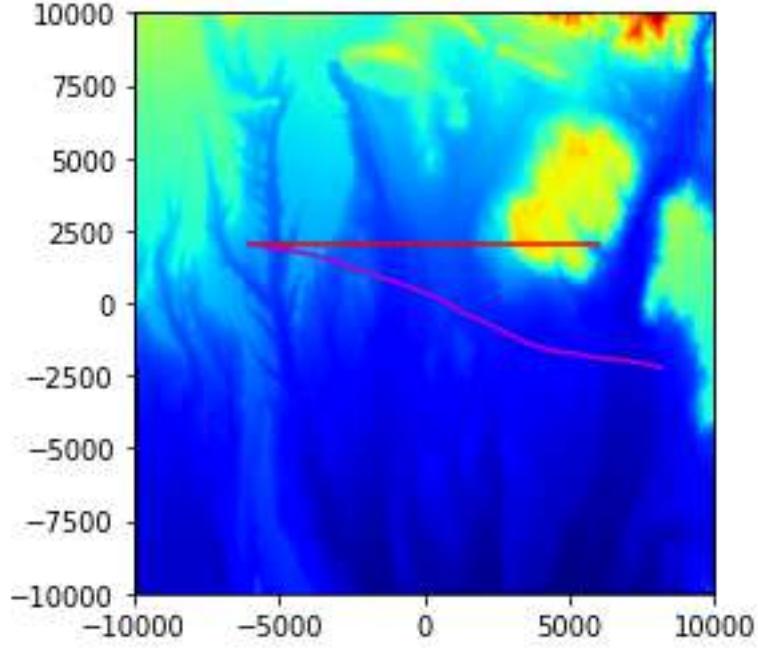


Figure 3: Modèle numérique du terrain, la trajectoire réelle et son estimation inertielles

Il y a une grande déviation relativement à l'échelle du plot (environ 2500). La trajectoire estimée dévie de plus en plus de la trajectoire réelle.

### Q3. Démonstration que le modèle d'état correspondant est donnée par :

$$\begin{pmatrix} \delta r_k \\ \delta v_k \end{pmatrix} = \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} \delta r_{k-1} \\ \delta v_{k-1} \end{pmatrix} - \Delta \begin{pmatrix} 0 \\ w_k^{\text{INS}} \end{pmatrix} \quad \text{avec} \quad \begin{pmatrix} \delta r_1 \\ \delta v_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

où la suite  $w_k^{\text{INS}}$  est un bruit blanc gaussien centré de matrice de covariance  $\sigma_{\text{INS}}^2 I_2$ .

On a

$$\begin{pmatrix} r_k^{\text{ISS}} \\ v_k^{\text{INS}} \end{pmatrix} = \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} r_{k-1}^{\text{INS}} \\ v_{k-1}^{\text{INS}} \end{pmatrix} + \Delta \begin{pmatrix} 0 \\ a_k^{\text{INS}} \end{pmatrix}$$

et comme:

$$\begin{pmatrix} r_k \\ v_k \end{pmatrix} = \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} r_{k-1} \\ v_{k-1} \end{pmatrix} + \Delta \begin{pmatrix} 0 \\ a_k^{\text{INS}} + w_k^{\text{INS}} \end{pmatrix}$$

Puisque  $\delta r_k = r_k - r_k^{INS}$  et  $\delta v_k = v_k - v_k^{INS}$ , le nouvel modèle d'état s'écrit:

$$\begin{aligned} \begin{pmatrix} \delta r_k \\ \delta v_k \end{pmatrix} &= \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} r_{k-1} \\ v_{k-1} \end{pmatrix} - \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} r_{k-1}^{INS} \\ v_{k-1}^{INS} \end{pmatrix} + \Delta \begin{pmatrix} 0 \\ a_k \end{pmatrix} - \Delta \begin{pmatrix} 0 \\ a_k^{INS} \end{pmatrix} \\ &= \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} \delta_{k-1}^{INS} \\ \delta_{k-1}^{INS} \end{pmatrix} + \Delta \begin{pmatrix} 0 \\ -w_k^{INS} \end{pmatrix} \\ &= \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} \delta_{k-1}^{INS} \\ \delta_{k-1}^{INS} \end{pmatrix} - \Delta \begin{pmatrix} 0 \\ w_k^{INS} \end{pmatrix} \end{aligned}$$

**Q4. Lecture à chaque instant  $t_k$ , sur la carte de la hauteur  $h(r_k)$  du relief au point de coordonnée horizontale  $r_k$  situé à la verticale de l'avion et représentation graphique dans un plan vertical du profil du relief survolé.**

Le tableau suivant décrit la position horizontale de l'avion à chaque instant,

Instant $t$	$t_1$	$t_2$	$t_3$	...	$t_{100}$	$t_{101}$
Hauteur $h(r_k)$	5871	5869	5868	...	5899.99	5942.01

et étant donné la carte numérique, on représente sur la figure 4 le relief survolé sur un plan vertical

```

1 rtrue_map = np.zeros(rtrue.shape[1])
2 N = len(rtrue_map)
3 N2, N1 = np.shape(map)
4 for k in range(N):
5     i= int((X2MAX - rtrue[1,k]) * N2 /(X2MAX-X2MIN) )
6     j= int((rtrue[0,k] - X1MAX) * N1 /(X1MAX-X1MIN) )
7     rtrue_map[k] = map[i,j]
8
9 plt.plot(range(N),rtrue_map, 'b-', label='profil_reel')
10 plt.show()

```

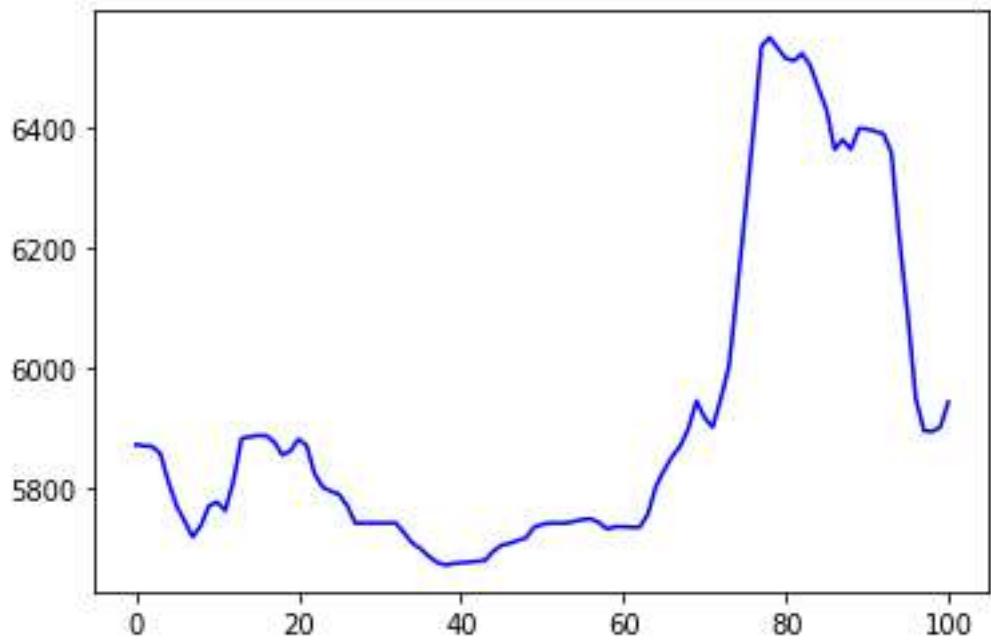


Figure 4: Profil réel du terrain survolé

**Q5. Représentation sur le même graphique pour chaque instant entre 0 et  $T$  du profil exact du relief survolé et les mesures bruitées de la hauteur du relief survolé.**

A partir de mesures de l'altitude et de la hauteur de l'avion au-dessus du terrain, on représente ci-dessous le profil exact du relief et les mesures bruitées de la hauteur de celui-ci entre 0 et  $T$ .

```

1 h_ALT = io.loadmat('alt.mat')['h_ALT'][0]
2 plt.plot(range(N),h_ALT,'r+',label='mesures altimetriques')
3 plt.legend()
4 plt.show()
```

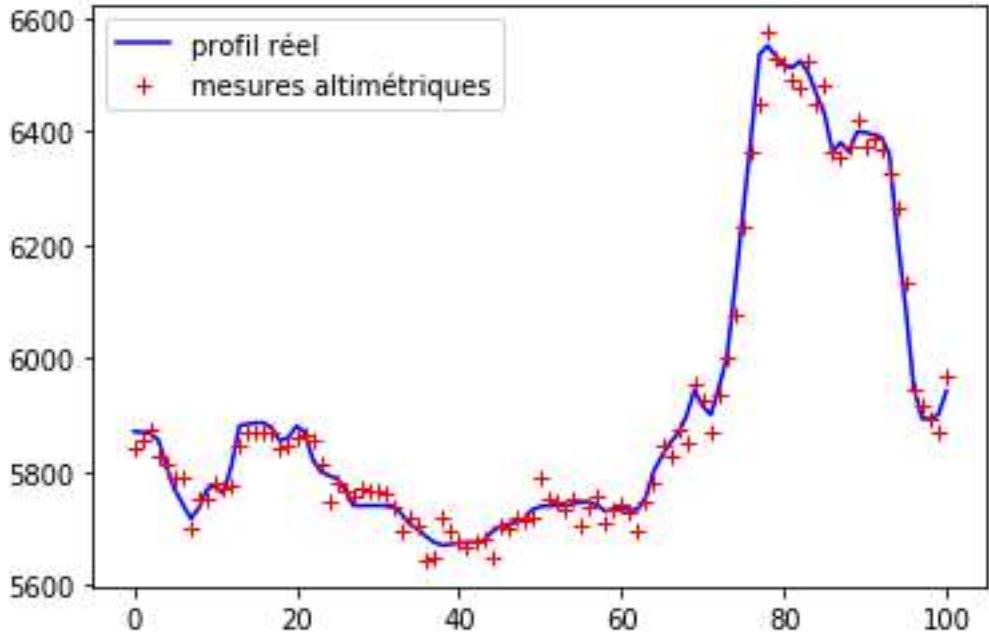


Figure 5: Profil réel du terrain survolé et mesures altimétriques

#### **Q6. Démonstration :**

- La mesure bruité  $h_k^{\text{ALT}}$  fournie par le radio-altimètre et par le baro-altimètre peut être reliée à l'erreur d'estimation inertieille  $\delta r_k$  sur la position horizontale par

$$h_k^{\text{ALT}} = h \left( r_k^{\text{INS}} + \delta r_k \right) + w_k^{\text{BAR}} - w_k^{\text{ALT}}$$

- L'expression de la fonction de vraisemblance.

On a

$$h_k^{\text{ALT}} = z_k^{\text{BAR}} - d_k^{\text{ALT}} = h(r_k) + w_k^{\text{BAR}} - w_k^{\text{ALT}}$$

En remplaçant,  $r_k$  par son expression  $r_k = r_k^{\text{INS}} + \delta r_k$  on trouve l'équation d'observation :

$$h_k^{\text{ALT}} = h \left( r_k^{\text{INS}} + \delta r_k \right) + w_k^{\text{BAR}} - w_k^{\text{ALT}}$$

Comme  $w_k^{\text{BAR}}$  et  $w_k^{\text{ALT}}$  sont des bruits blancs gaussiens de variances respectives  $\sigma_{\text{BAR}}^2$  et  $\sigma_{\text{ALT}}^2$ , la différence  $w_k^{\text{BAR}} - w_k^{\text{ALT}}$  est une variable aléatoire gaussienne centrée de variance  $\sigma^2 = \sigma_{\text{BAR}}^2 + \sigma_{\text{ALT}}^2$ . Ainsi si on note  $q$  la distribution de la loi gaussienne centrée de variance  $\sigma$  la fonction de vraisemblance s'écrit:

$$g(x) = q \left( h_k^{\text{ALT}} - h(r_k + x) \right)$$

## Q7. Filtrage particulaire SIR

La loi du vecteur initial  $(\delta r_0, \delta v_0)$  étant donnée (condition initiale incertaine), on dispose de  $N$  particules  $(\xi_1^0, \dots, \xi_N^0)$  où chaque élément  $\xi_i^0$  est un vecteur de  $\mathbb{R}^4$  normalement distribué de moyenne nulle et de matrice de covariance

$$\Sigma_0 = \begin{pmatrix} \sigma_{r_0}^2 I_2 & 0 \\ 0 & \sigma_{v_0}^2 I_2 \end{pmatrix}$$

On leur associe des poids selon la fonction de vraisemblance  $g$  énoncé dans la question précédente évalué en chaque particule. Ensuite, pour chaque pas temporel  $t_k$ , on effectue une redistribution multinomiale selon les poids calculés, et on met à jour les coordonnées des particules selon l'équation d'état. A la fin de chaque itération les poids doivent être re-évalués (on réapplique la vraisemblance). L'algorithme peut s'écrire donc comme suit:

---

### Algorithm 1 Filtre particulaire SIR

---

- Initialiser  $(\xi_1^0, \dots, \xi_N^0)$  où  $\xi_i^0 \in \mathbb{R}^4$ , normalement distribué de moyenne nulle et de matrice de covariance  $\Sigma_0$ .
- Initialiser les poids des particules  $w_i^0 = g(\xi_i^0) / \sum_{j=1}^N g(\xi_j^0)$

**Pour**  $k \leq n_{\max}$  **Faire :**

- Effectuer la redistribution multinomiale  $\longrightarrow (\hat{\xi}_1^{k-1}, \dots, \hat{\xi}_N^{k-1})$  - Mutation des particules selon l'équation d'état

$$\xi_i^k = \begin{pmatrix} I_2 & \Delta I_2 \\ 0 & I_2 \end{pmatrix} \hat{\xi}_i^{k-1} - \Delta \begin{pmatrix} 0 \\ w_k^{\text{INS}} \end{pmatrix}$$

- Correction des poids  $w_i^k = g(\xi_i^k) / \sum_{j=1}^N g(\xi_j^k)$

**fin Pour**

---

On a implémenté cet algorithme pour différentes valeurs de  $N$  et on a suivi l'évolution du nuage des particules et leur moyenne pondérée au cours du temps.

Les figures suivants ont été effectués pour  $N = 5000$ . Les points noirs représentent les particules et l'étoile représente leur somme pondérée  $\xi_k^* = \sum_{i=1}^N \xi_i^k w_i^k$ . La figure 11 représente la région explorée par les particules au cours de l'algorithme et la figure 10 représente le relief réel, les valeurs altéritmiques observées et finalement les valeurs de la somme pondérée des particules.

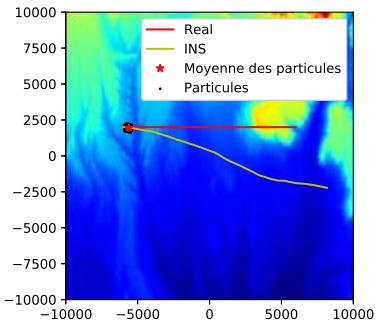


Figure 6: SIR à l'instant  $k = 3$

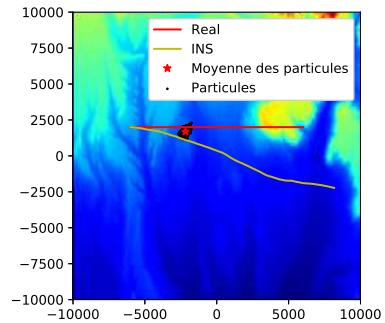


Figure 7: SIR à l'instant  $k = 33$

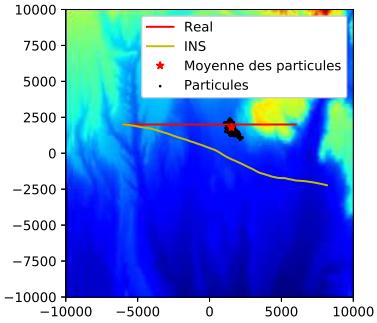


Figure 8: SIR à l'instant  $k = 63$

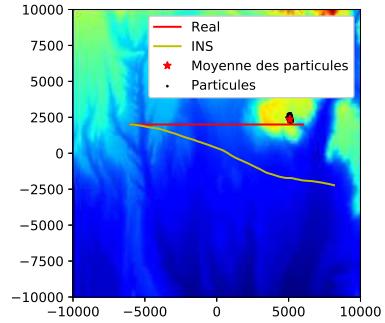


Figure 9: SIR à l'instant  $k = 93$

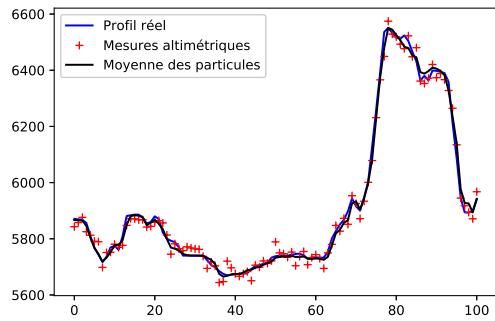


Figure 10: Le relief de la moyenne pondérée de toutes les particules.

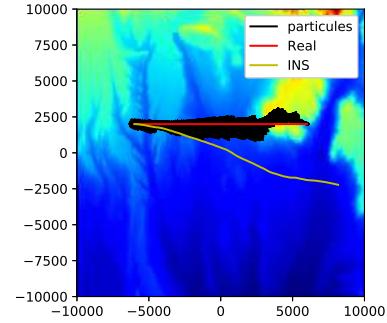


Figure 11: trajectoire de toutes les particules.

## **Q8. Filtrage particulaire SIS et Filtrage particulaire avec redistribution adaptative**

### **Filtrage particulaire SIS**

On a implémenté cet algorithme pour différentes valeurs de  $N$  et on a suivi l'évolution du nuage des particules et leur moyenne pondérée au cours du temps.

La figure 17 représente la région explorée par les particules au cours de l'algorithme et la figure 16 représente le relief réel, les valeurs altéritmiques observés et finalement les valeurs de la somme pondérées des particules.

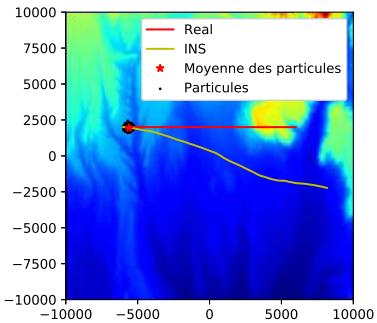


Figure 12: SIS à l'instant  $k = 3$

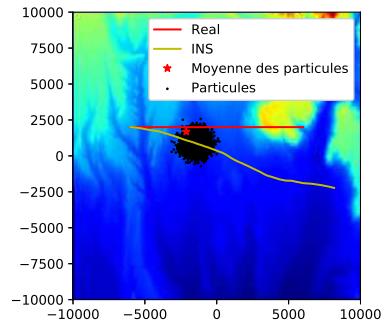


Figure 13: SIS à l'instant  $k = 33$

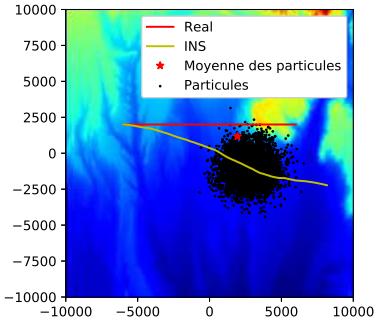


Figure 14: SIS à l'instant  $k = 63$

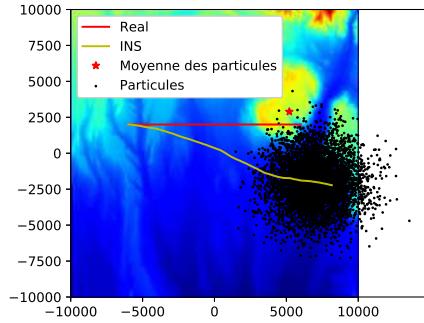


Figure 15: SIS à l'instant  $k = 93$

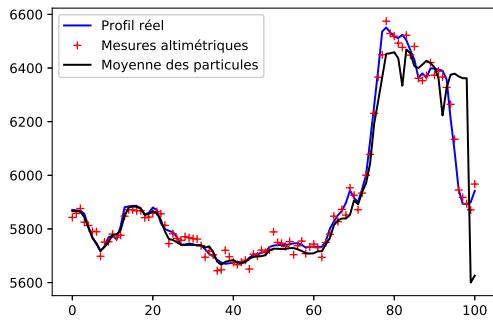


Figure 16: Le relief de la moyenne pondérée de toutes les particules

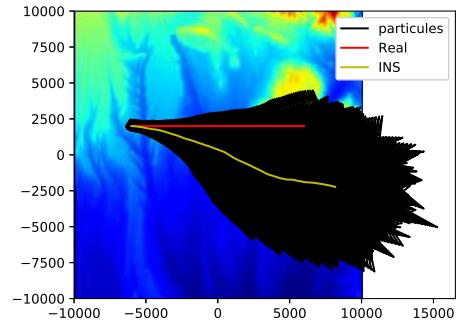


Figure 17: trajectoire de toutes les particules

### Filtrage particulaire avec redistribution adaptative

Pour mettre en évidence l'effet de la redistribution, nous avons mis en oeuvre un filtre avec redistribution adaptative, c'est-à-dire la redistribution ne s'effectue que lorsque

$$N_{eff} \leq cN, N_{eff} = 1 / \sum_{j=1}^N (w_i^{k-1})^2$$

où  $N_{eff}$  est la taille effective de l'échantillon,  $N$  est le nombre de particules et  $c$  est le seuil de redistribution tel que si  $c = 0$  alors il s'agit du filtre SIS, sinon c'est le filtre SIR qui intervient. De plus,  $N_{eff}$  est toujours plus petite que  $N$ , ce qui nous a mené à varier  $c$  que dans l'intervalle  $[0, 1]$ .

Nous représentons le nuages des particules à quatre instants différents entre 0 et  $T$  avec  $N = 1000$ .

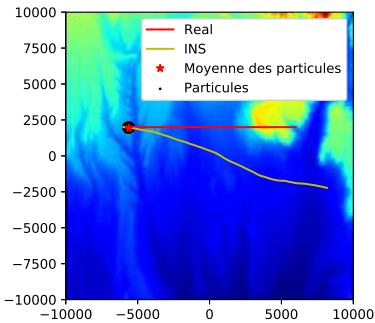


Figure 18: adaptive à l'instant  $k = 3$

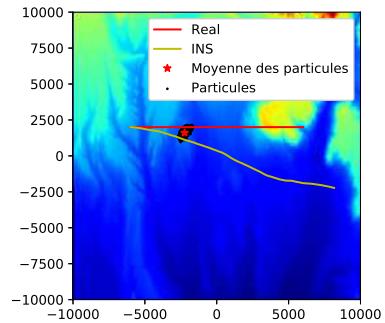


Figure 19: adaptive à l'instant  $k = 33$

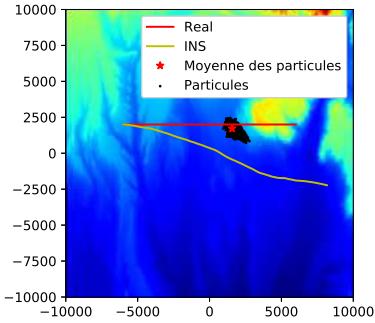


Figure 20: adaptive à l'instant  $k = 3$

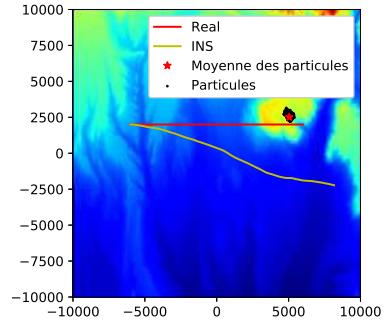


Figure 21: adaptive à l'instant  $k = 3$

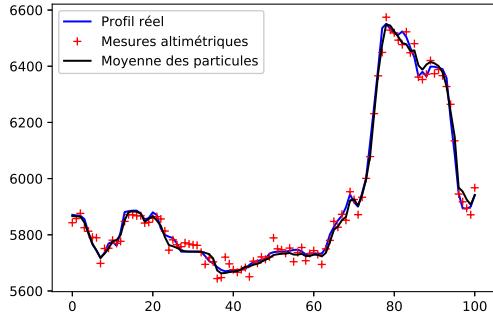


Figure 22: Le relief de la moyenne pondérée de toutes les particules

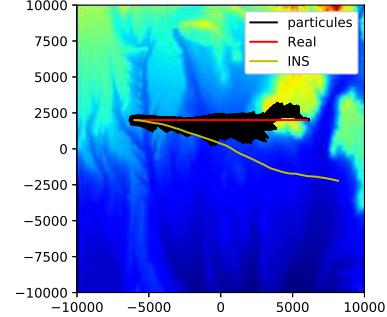


Figure 23: trajectoire de toutes les particules

Afin d'observer l'impacte du nombre  $N$  des particules et le coefficient de sensibilité  $c$  des algorithmes.  
On calcul le MSE :

$$MSE = \frac{1}{T} \sum_{k=1}^T \|r_k - \bar{r}\|_2^2$$

avec  $r_k$  la position réel à l'instant  $k$  et  $\bar{r}$  la position obtenue par la somme pondéré des particules.

La figure 24 représente l'évolution du MSE pour le SIR et le SIS en fonction des nombres des particules. On remarque que le SIR possède une performance meilleur que le SIS.

On a simulé l'algorithme adaptative pour différents valeurs du paramètre  $c$  avec  $N = 1000$ . La figure 25 représente la MSE pour des différents valeurs de  $c$ . La valeur minimale du MSE est 27.66 atteinte pour  $c = 0.444$ .

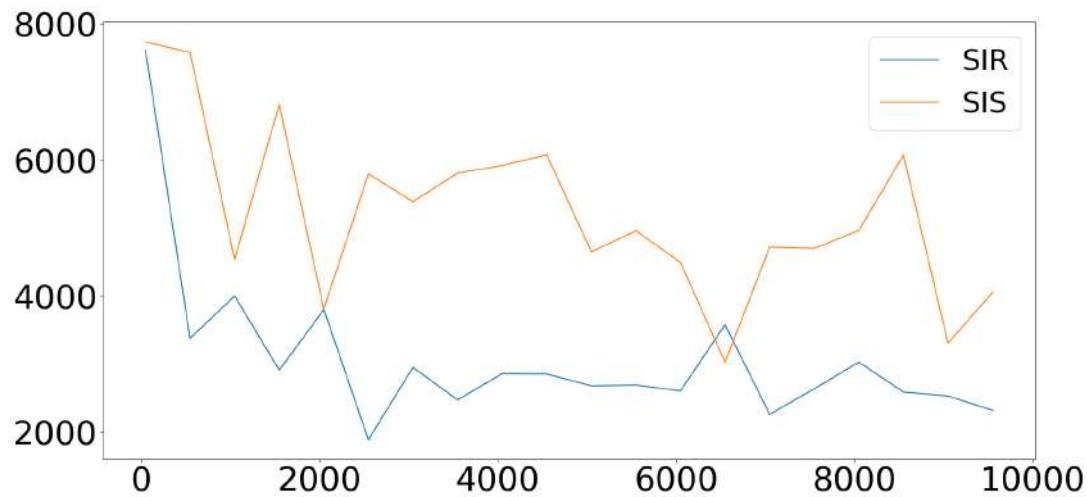


Figure 24: MSE; SIR vs SIS en fonction du N.

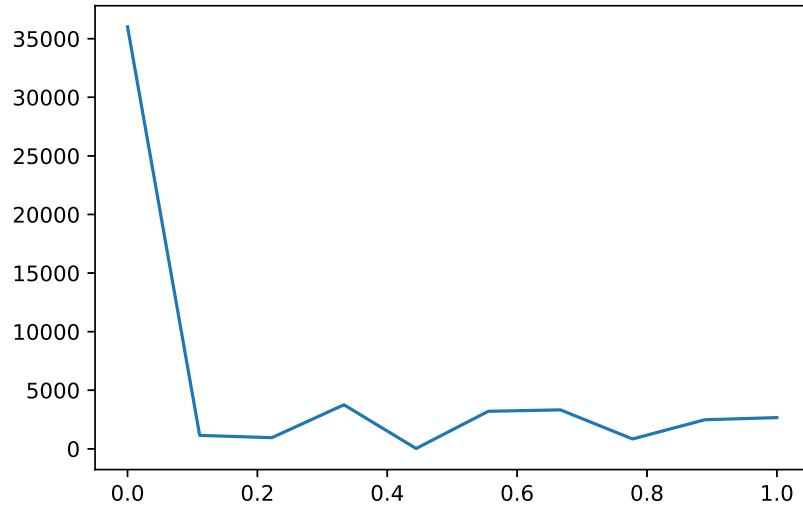


Figure 25: MSE on fonction du c.

## Conclusion

L'avantage du SIR est de suivre le vrai relief mais au cours du temps plusieurs particules ne sont plus choisi.  
L'avantage du SIS est d'exploiter au maximum l'espace des particules, mais on ne suit pas forcément le relief.

L'algorithme adaptatif est une combinaison des deux algorithmes précédents mais qui est plus performant. En effet, au cours de l'exécution, on explore l'espace à l'aide du l'algorithme SIS. S'ils s'étalement beaucoup, on utilise l'algorithme SIR pour s'approcher encore du de la cible. Ensuite si le nombre effectif des particules choisi diminue considérablement, on réutilise l'algorithme SIS pour augmenter de nouveaux le nombre des particules effectifs.

En conclusion, l'algorithme adaptatif présente une meilleur estimation que le SIR et le SIS, il suffit de trouver le bon paramètre  $c$ , dans ce cas  $c = 0.4$ .