**Université Grenoble Alpes** **Polytech Grenoble**
**RICM5** **Année 2017–2018**

**Visualisation — TP n° 3**
**Interpolation de données éparses**

# 1 Introduction

## Objectif

Ce TP a pour but l'implémentation de deux algorithmes d'interpolation de données éparses, les algorithmes de Shepard et Multi Quadric décrits dans le polycopié de cours. Le but de ces algorithmes est d'obtenir une évaluation d'une fonction $F$ en tout point d'un domaine, à partir de certains valeurs connues (les $f_i$). Pour plus de précisions, référez-vous au polycopié de cours, page 42. Il est conseillé (mais non obligatoire) de réaliser ce TP en C/C++ ou Python afin d'utiliser une bibliothèque de résolution de systèmes linéaires pour l'implémentation du second algorithme.

## Données et résultats

Appliquez vos algorithmes sur des points dont les positions sont comprises entre 0 et 1. Ainsi pour tester vos algorithmes, utilisez une grille de n sur n points entre 0 et 1 représentant votre domaine. Dans ce domaine, les points sont définis par leur position x et y ainsi que par leur valeur, par exemple le point (0.2, 0.4, 5) indique que le point dont l'abscisse est 0.2 et l'ordonnée est 0.4 a pour valeur 5.

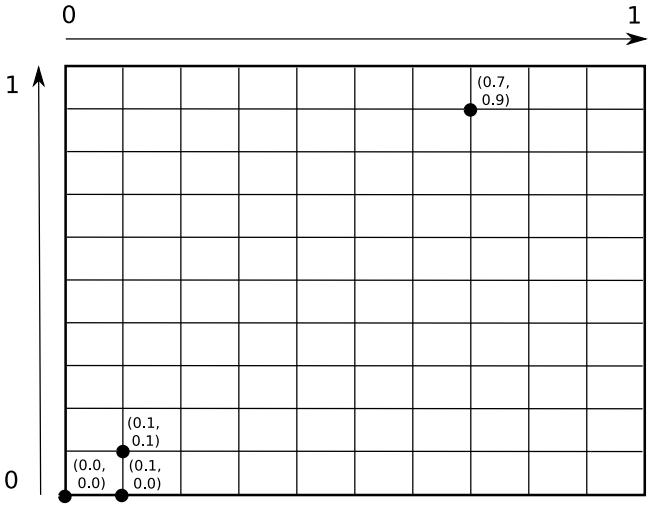


FIGURE 1 – Exemple de grille 10x10 points représentant le domaine.

Commencez par utiliser quelques points fixés (par exemple (0.2, 0.5, 2) et (0.4, 0.2, 5)) que vous définissez à la main pour tester vos algorithmes. Puis utilisez une fonction plus complexe (par exemple $sin(y/2) + cos(x/2)$) et générez des points aléatoirement dans le domaine (les $f_i$).

Pour visualiser le resultat, vous pouvez utiliser *gnuplot* avec la fonction *splot* qui permet d'afficher une surface en 3D. Le fichier décrivant la surface se compose d'un point (2 coordonnées et une valeur) par ligne, par exemple :

0.25 0.50 2.45

0.30 0.75 4.2

0.50 0.25 10.2

0.75 0.25 6.89

Pour obtenir une surface correspondant à votre fonction d'interpolation, définissez une grille échantillonée sur votre domaine (par exemple 1000 points par 1000). Appliquez ensuite vos algorithmes sur chacun des points de cette grille et redirigez le résultat vers la sortie standard pour l'afficher avec *gnuplot*. Par exemple :

*./interpolation > res.txt*

*gnuplot*

*splot "res.txt"*

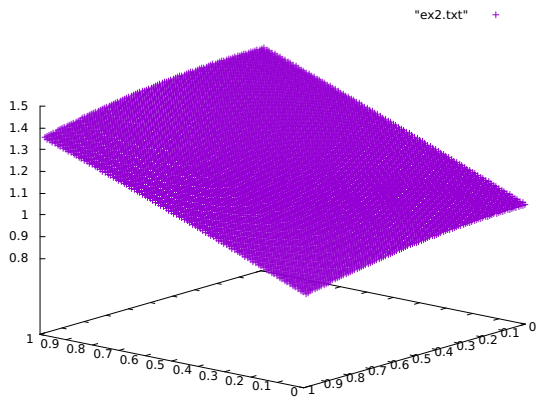


FIGURE 2 – Affichage de $sin(y/2) + cos(x/2)$ avec gnuplot

# 2 Méthode de Shepard

## Implémentation

Implémentez l'algorithme de Shepard (page 44 du polycopié de cours). Dans un premier temps, utilisez une valeur de 2 pour $\mu_i$ comme conseillé dans le polycopié.

## Influence de $\mu_i$

Testez le résultat de votre algorithme pour des valeurs différentes de $\mu_i$ et commentez l'influence de $\mu_i$ sur le résultat.

# 3 Méthode de Hardy

## Implémentation

Implémentez l'algorithme de Hardy, aussi appelé méthode des Multi-Quadric (page 56 du polycopié de cours). Cet algorithme nécessite de résoudre un système d'équations linéaires. Inutile de réimplémenter un solveur linéaire, vous devez utiliser une bibliothèque disponible.

Par exemple, *Eigen* (Documentation de Eigen) est facile à installer et utiliser en C/C++. Plusieurs classes de Eigen permettent de résoudre un système d'équations linéaires (lien). Vous pouvez par exemple utiliser la classe *LDLT* comme montré dans l'exemple situé sur la page d'Eigen citée précédemment. Il existe également une adaptation de *Eigen* pour Java : Jeigen, mais son utilisation est à vos risques et périls !

En Python, la fonction *numpy.lnalg.solver* (Lien vers la documentation) permet également de résoudre facilement des sytèmes d'équations linéaires.

### Influence de $R$

Testez et comparez les résultats de votre algorithme avec les trois suggestions du polycopié pour la valeur de $R$ (Hardy, Franke et Stead).

## 4 Rendu

Votre rendu sera composé d'un compte rendu et du code source de vos deux algorithmes. Le compte rendu devra contenir vos images de résulats commentées, calculées à partir de différentes fonctions de génération (par exemple $sin(y/2) + cos(x/2)$ mais également d'autres fonctions que vous aurez défini). Il contiendra également les images de résultats montrant l'impact de $\mu_i$ pour l'algorithme de Shepard et l'impact de $R$ pour l'algorithme de Hardy.