TP1 : plans remplissant l'espace et krigeage (3h)

Victor Picheny, Rodolphe Le Riche

13 décembre 2017, 13h30- 16h45

Les objectifs de ce TP sont :
— construire des plans remplissant l'espace (de différentes familles) à l'aide des méthodes vues en cours
— analyser et comparer ces plans à l'aide d'outils graphiques et de critères type maximin, etc.
— construire un plan d'expériences pour le cas test "réservoir magmatique"
— construire un modèle de krigeage à partir des données issues du cas test

NB : les séances suivantes (notamment : TP optimisation globale) utiliseront le modèle de krigeage généré.

1 Construction de plans

1.1 Tessellations centroïdales de Voronoï

Créez un script R permettant de générer une tesselation centroïdale de Voronoï (normalisée sur $[0,1]^d$) pour un nombre quelconque de points et une dimension quelconque. On implémentera l'algorithme de MacQueen ; le nombre d'itérations sera défini comme un paramètre d'entrée du script.

```
generateCVT <- function(npts, dim, nite)
```

Le plan d'expériences s'écrira comme une matrice de *npts* lignes et *dim* colonnes.

1.2 Hypercubes latins

Créez un script R permettant de générer un hypercube latin aléatoire (normalisé) pour un nombre quelconque de points et une dimension quelconque. Pour générer des permutations : `sample.int`.

```
generateLHS <- function(npts, dim)
```

1.3 Critères

Créez un script R permettant de calculer le critère maximin d'un plan. Le script fournira également en sortie la matrice des interdistances (taille $npts \times npts$). Attention à éviter les calculs redondants (la matrice est symétrique !). On peut éventuellement utiliser la fonction `outer`.

```
evalMinDist <- function(X)
{ (.....)
return( list(minDist = d, allDist = D) ) }
```

1.4 Hypercubes latins optimisés

Utilisez le script précédent pour construire des hypercubes latins optimisés. Selon votre vitesse d'avancement, vous coderez au choix :
— une recherche aléatoire pure : on génère un certain nombre de plans et on garde le meilleur ;

— un algorithme d'échange simple : on génère un plan, puis on propose des échanges aléatoires qu'on accepte s'ils améliorent le critère ;
— un algorithme d'échange avec recuit simulé.

Dans les 2 derniers cas, les performances peuvent être améliorées en choisissant pour l'échange un point critique pour le critère.

2 Analyse

Générez et comparez des plans CVT, LHS quelconque et LHS optimisé pour les configurations suivantes :
— dimension 2, 10 points ;
— dimension 5, 70 points ;
— dimension 10, 150 points.

Observez en particulier les différences sur :
— la répartition sur les marginales de dimension 1 (histogrammes)
— la répartition sur les marginales de dimension 2 (fonction `pairs`)
— la valeur du critère *maximin*, éventuellement l'histogramme des interdistances.

Peut-on déterminer facilement les "meilleurs" plans ? Essayez éventuellement d'autres critères / visualisations afin de faciliter votre choix (soyez créatifs !). Vous pouvez également utiliser le script `discrepancy.R` fourni.

Préparation du cas test "réservoir magmatique" Sauvegardez le "meilleur" plan de taille 70×5 que vous ayez généré selon la méthode de votre choix. **Ce plan est nécessaire pour la suite des TPs**.

3 Prise en main du cas test "réservoir magmatique"

Se référer au document "*Test case for the UP4 2017-18 : Identification of a volcano reservoir from surface displacements*". Lire la section 2 et suivre les instructions de cette section pour prendre en main le code.

Pour finir, générez les observations correspondantes à votre plan d'expériences à l'aide de la fonction `compute_wls`.

4 Modèle de krigeage

4.1 Réflexions préliminaires

Avant de vous lancer dans la création d'un modèle de krigeage proprement dit, prenez le temps de vous poser des questions telles que :
— Une représentation graphique des données est-elle possible ?
— Un pré-traitement des données est-il nécessaire (données aberrantes, changement d'échelle, ...) ?
— Peut-on s'aider de la physique du problème ?
— Comment allez-vous tester les modèles ?
— ...

4.2 Construction des modèles

A l'aide du package `DiceKriging`, construisez et comparez différents modèles en essayant différents noyaux et différentes tendances à partir du plan d'expériences généré précédemment. Choisissez le modèle qui vous semble le mieux adapté. Ce choix sera à justifier dans le rapport.

Donnez une analyse critique de votre modèle : vous semble-t-il que la qualité du modèle est suffisante pour apprendre des informations sur la fonction objectif ?

Commandes R

Le package `DiceKriging` vous permettra de gagner un temps précieux. La construction de modèles se fait alors à l'aide de la fonction `km`, et les prédictions à l'aide de la fonction `predict`. De plus, ce package possède déjà des outils de diagnostic des modèles (`plot(model), summary(model)`) et il permet d'estimer les paramètres des noyaux par maximum de vraisemblance ou par validation croisée. Vous pouvez également utiliser les outils de visualisation du package `DiceView` (`sectionview`).

Outre les diagnostics visuels, un indicateur classique est le Q^2 (part de variance expliquée par le modèle), qui se calcule (en validation croisée avec `DiceKriging`) ainsi :

```
resLOO <- leaveOneOut.km(model, type="UK", trend.reestim=FALSE)
Q2 <- 1 - sum((resLOO$mean - model@y)^2) / sum( (model@y - mean(model@y))^2)
```

Bonus : si le temps le permet

Programmez un enrichissement adaptatif du plan d'expériences. Par exemple, en ajoutant séquentiellement le point qui maximise la variance de prédiction du modèle. Vous pourrez chercher le nouveau point avec la fonction `cmaes.R` du TP d'optimisation globale.