PETROLEUM 2015
GEOSTATISTICS

Fr B01

Efficient Neighborhoods for Kriging with Numerous Data

M. Vigsnes* (Norwegian Computing Center), P. Abrahamsen (Norwegian Computing Center), V.L. Hauge (Norwegian Computing Center) & O. Kolbjornsen (Norwegian Computing Center)

SUMMARY

Kriging is a data interpolation method that can be used to populate regular grids from data scattered in space, and requires the solution of a linear equation system the size of the number of data. When the data is numerous the speed of the calculation is slow. In this paper we propose to divide the regular grid into rectangular sub-segments and let all the grid cells in each sub-segment share a common data neighborhood. The advantage of this approach is that the number of data in the neighborhoods can be small compared to the complete dataset and it is possible to reuse some of the computations for all grid cells in each sub-segment. We show that the precision can be controlled through selection of neighbourhood size, and that the speed of the calculations can be optimized through selection of sub-segment size. We show that this is an efficient method for kriging when number of data is huge, giving a significant speed-up even for high data densities and precisions.

Introduction

Kriging is a data interpolation method that can be used to populate regular grids from data scattered in space. Kriging requires the solution of a linear equation system the size of the number of data. For numerous data we encounter two problems: The speed of calculations is slow and numerical instabilities may occur for really large datasets. For this reason many implementations approximate the kriging system by using subsets of data around each grid node (Chilès and Delfiner, 1999), (Emery, 2009). In this paper we propose to divide the regular grid into rectangular sub-segments and let all the grid nodes in each sub-segment share a common data neighborhood. The advantage of this approach is that the number of data in the neighborhoods can be small compared to the complete dataset *and* it is possible to reuse some of the computations for all grid nodes in each sub-segment. We will show that this is efficient and that we can control the precision of this approximation by choosing the size of the data neighborhood.

Kriging

Consider a regular grid $\mathfrak{L}_\mathfrak{D} \subset \mathfrak{D}$ where $\mathfrak{D}$ is a hyperrectangle (orthotope) in $\mathbb{R}^d$. Assume that the grid $\mathfrak{L}_\mathfrak{D}$ covers $\mathfrak{D}$ and contains N grid nodes. The objective is to predict a random field $z(x)$ at each of the N grid nodes in $\mathfrak{L}_\mathfrak{D}$ given n observations using kriging.

By organizing the observations of $z(x)$ in a n-dimensional vector $\mathbf{z} = \big[z(\mathbf{x}_1), z(\mathbf{x}_2), \ldots, z(\mathbf{x}_n)\big]$; $\mathbf{x}_i \in \mathfrak{D}$, the (simple) kriging equation reads

$$z^*(\mathbf{x}) = m + \mathbf{k}'(\mathbf{x})\,\mathbf{K}^{-1}\,(\mathbf{z} - \mathbf{m}) \qquad \forall \quad \mathbf{x} \in \mathfrak{L}_\mathfrak{D}, \tag{1}$$

where $z^*(\mathbf{x})$ is the predicted value at $\mathbf{x}$, m is the known mean, $\mathbf{m}$ is a vector containing m, $\mathbf{k}(\mathbf{x}) = \mathrm{Cov}\big\{z(\mathbf{x}), \mathbf{z}\big\}$, and $\mathbf{K} = \mathrm{Var}\big\{\mathbf{z}\big\}$ is the kriging matrix.

Obtaining $z^*(\mathbf{x})$ in (1) is essentially done in four steps. The first step is to establish $\mathbf{K}$, which is an $\mathscr{O}(n^2)$ process. Secondly, calculating the Cholesky factorization is an $\mathscr{O}(n^3)$ process. The third step is to solve the equation system using the Cholesky factorization. This is done by calculating the location independent (dual kriging) weights

$$\mathbf{w} = \mathbf{K}^{-1}\,(\mathbf{z} - \mathbf{m}). \tag{2}$$

Solving for the weights is an $\mathscr{O}(n^2)$ process. The fourth step is inserting the weights in (1):

$$z^*(\mathbf{x}) = m + \mathbf{k}'(\mathbf{x}) \cdot \mathbf{w} \tag{3}$$

This is an $\mathscr{O}(nN)$ process that is dominated by calculating the n covariances at every grid node.

Data neighborhoods

A common approach to reduce the number of data, n, is to use a moving data neighborhood. This means that the subset of the data that is closest to a grid node $\mathbf{x} \in \mathfrak{L}_\mathfrak{D}$ is chosen when predicting $z(\mathbf{x})$. The number of data in the neighborhoods is usually chosen quite small (< 100). The downside is that all four steps in solving (1) must be calculated for every grid node so the CPU time can become long for large grids.

Instead of looking at grid node specific data neighborhoods we suggest to divide $\mathfrak{D}$ into sub-segments where each sub-segment share a common data neighborhood. This has the advantage that the location independent weights, (2), can be reused for all the grid nodes inside the sub-segment.

For simplicity we propose to use equally sized sub-segments $\mathfrak{D}_i$ that are hyperrectangles in $\mathbb{R}^d$. We assume that the side lengths of the hyperrectangle $\mathfrak{D}$ are large compared to the correlation ranges so that we can ignore boundary effects. Then, the only relevant length scales are the correlation ranges,

$\{R_1, R_2, \ldots, R_d\}$. The side lengths of the sub-segments and the common data neighborhoods are therefore chosen proportional to the correlation ranges in each direction. This is illustrated in Fig. 1 for a two dimensional situation. The size of the sub-segment is $SR_1 \times SR_2$ and the size of the common data neighborhood is $(2P+S)R_1 \times (2P+S)R_2$ where S and P are dimensionless proportionality constants. The constant P tells us how large, in terms of correlation ranges, the data neighborhood extends the sub-segment.

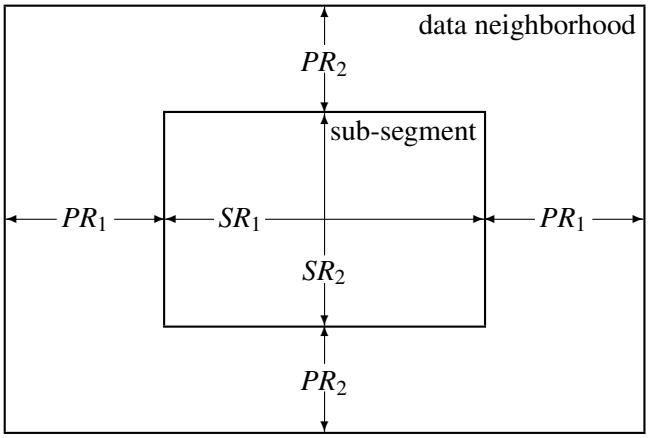


Figure 1 *Illustration of a sub-segment $\mathfrak{D}_i$ and its corresponding data neighborhood in $\mathbb{R}^2$.*

CPU time

Consider the data density, $\rho_n = nV_R/V_{\mathfrak{D}}$, and grid node density, $\rho_N = NV_R/V_{\mathfrak{D}}$, where $V_{\mathfrak{D}}$ is the volume of $\mathfrak{D}$ and V_R is the volume defined by the product of the correlation ranges: $V_R = R_1 R_2 \cdots R_d$. The CPU time per grid node can be approximated by

$$\frac{\text{CPU}}{N} \approx T_{\mathbf{K}} \frac{\rho_n^2 (2P+S)^{2d}}{\rho_N S^d} + T_{\text{Chol}} \frac{\rho_n^3 (2P+S)^{3d}}{\rho_N S^d} + T_{\text{Solve}} \frac{\rho_n^2 (2P+S)^{2d}}{\rho_N S^d} + T_{z^*} \rho_n (2P+S)^d, \tag{4}$$

where the T's are time constants that can be estimated, one for each of the four steps in solving (1). They depend on hardware and implementation.

Choosing the size of common data neighborhoods

The size of the common data neighborhood is determined by P. The choice of P is a compromise between acceptable error and acceptable CPU time. Let us consider the maximum absolute error (MAE):

$$\text{MAE} = \max_{x \in \mathfrak{L}_{\mathfrak{D}}} \left| z^*(\mathbf{x}) - z^*_{\text{cdn}}(\mathbf{x}) \right| / \sigma, \tag{5}$$

where $z^*(\mathbf{x})$ is the kriging predictor obtained using all data, $z^*_{\text{cdn}}(\mathbf{x})$ is the kriging predictor obtained using common data neighborhoods and $\sigma^2 = \text{Var}\{z(\mathbf{x})\}$. So the calculated MAE is relative to the standard error of the random field, $z(\mathbf{x})$.

The relation between P and the MAE can be obtained from simulation experiments. Datasets with 2000 observations are generated on a 1000×1000 grid at the locations shown in Fig. 2. The data are drawn at random from a Gaussian distribution with zero expectation, variance one and correlations determined by the exponential variogram. Data densities of 5, 45 and 80 are obtained using correlation ranges 50, 150 and 200 respectively. A total of 100 samples are drawn for each range, and kriging is performed for various values of P. Empirical relations between P and the MAE are shown in Fig. 3. We see an approximate log-linear relation between the MAE and P, and observe that the MAE increases with smaller data density, ρ_n.

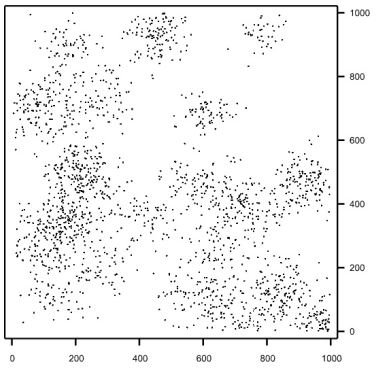


Figure 2 *Random data locations (2000) used in the simulation experiments.*

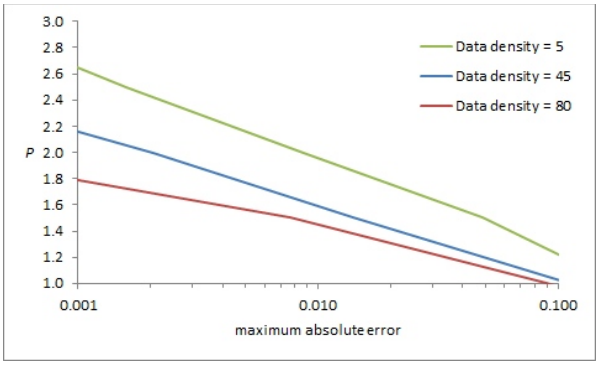


Figure 3 *Common data neighborhood size in terms of P versus MAE for different data densities, ρ_n.*

Fig. 4 shows the predicted random field and the corresponding error from one simulation experiment for various P values. We observe that the MAE decreases in the order of one decade when increasing P by 0.5. The edges of the sub-segments are to some extent visible in the predicted random field for $P = 0.5$, but not for higher P values.

Minimizing the CPU time by selecting the sub-segment size

For a given P, the sub-segment size S that minimizes the CPU time can be calculated from (4). This is in principal possible to do analytically but in this case the minimum is unique and more easily found by a binary search.

The CPU time in seconds versus sub-segment size S is shown in Fig. 5. The figure shows results for the P value that corresponds to a MAE of 0.01 for data density $\rho_n = 45$. Step 4 is only dependent on the common data neighborhood size; hence the time related to this operation is increasing with increasing S as P and ρ_n are fixed. However, for the operations related to the kriging matrix $\mathbf{K}$, i.e. Step 1-3, the time decreases with increasing S. For a small S, where $P \gg S$, the overlap dominates the CPU time for these steps. We observe that the minimum is found as a compromise between Step 1-3 and Step 4.

As a comparison, for a moving neighbourhood, the sub-segment is the size of a single grid cell. For a two dimensional case, assuming that $R_1 = R_2$ and that the grid cells are square, this corresponds to $S = \sqrt{1/\rho_N}$. For a data density of $\rho_n = 45$, this gives $S = 0.0067$. In order to achieve a MAE of 0.01 (P = 1.7) the CPU time is approximately 3.7 hours for this S.

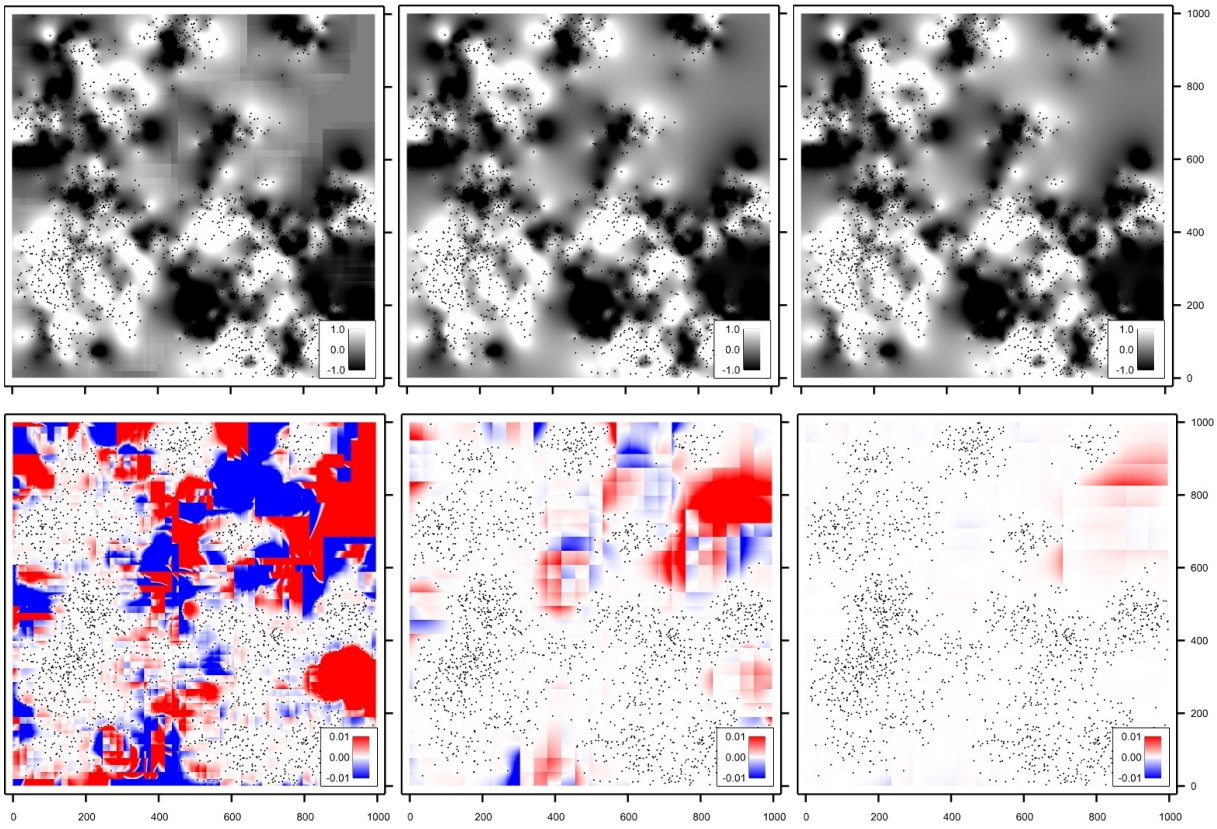
The speed-up for kriging in sub-segments compared to kriging using all data is given in Fig. 6 for MAE of 0.1 and 0.01. The gain is substantial for small data densities, i.e. up to 40 times more efficient. However, we observe that also for a higher data density, $\rho_n = 80$, the calculation is 2 and 4 times more efficient for MAE of 0.01 and 0.1 respectively.

Conclusion

Kriging in sub-segments is an efficient method for kriging when number of data is huge. The parametrization of the problem allows us to compromise between minimizing the CPU time and increasing the precision. The speed-up of the calculations is significant, even for high data densities and precisions.

References

Chilès, J.P. and Delfiner, P. (1999) Geostatistics; Modeling Spatial Uncertainty. Wiley, New York.
Emery, X. (2009) The kriging update equations and their application to the selection of neighboring data. *Computers & Geosciences*, **13**:269–280.



$P = 0.5, S = 0.12$
$MAE = 0.34, CPU = 5s$

$P = 1.0, S = 0.25$
$MAE = 0.040, CPU = 17s$

$P = 1.5, S = 0.39$
$MAE = 0.0088, CPU = 34s$

Figure 4 *The top row shows the predicted random fields, $z(\mathbf{x})$, for $P = 0.5$, 1.0 and 1.5. The bottom row shows the corresponding error, $z^*(\mathbf{x}) - z^*_{cdn}(\mathbf{x})$, compared to kriging using all data. The results are from one simulation experiment for data density $\rho_n = 45$. The average numbers of data in the common data neighborhoods are 53, 198 and 399 respectively.*

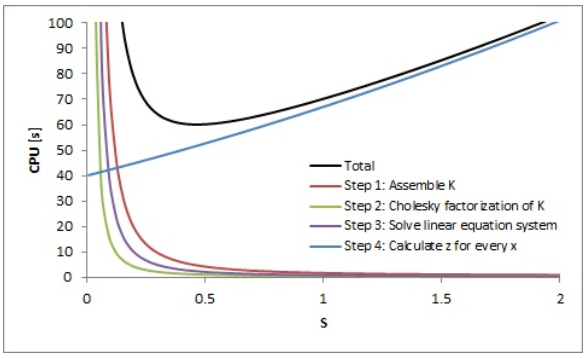


Figure 5 *CPU time for the operations included in (4) for increasing S. The selected $P = 1.7$ gives $MAE = 0.01$, for data density $\rho_n = 45$.*

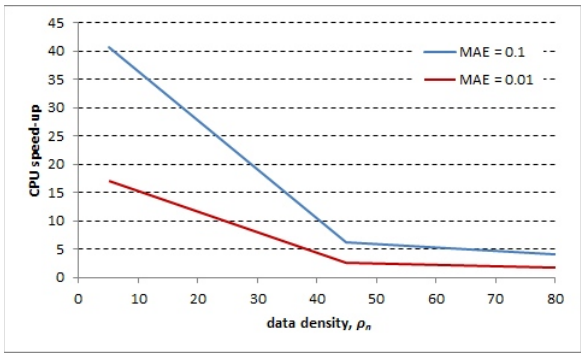


Figure 6 *CPU speed-up relative to kriging using all data for $MAE = 0.1$ and 0.01, for increasing data density ρ_n.*