Efficient kriging for real-time spatio-temporal interpolation

P.228 20th Conference on Probability and Statistics in the Atmospheric Sciences

Balaji Vasan Srinivasan, Ramani Duraiswami*, Raghu Murtugudde†
University of Maryland, College Park, MD

Abstract

Atmospheric data is often recorded at scattered station locations. While the data is generally available over a long period of time it cannot be used directly for extracting coherent patterns and mechanistic correlations. The only recourse is to spatially and temporally interpolate the data both to organize the station recording to a regular grid and to query the data for predictions at a particular location or time of interest. Spatio-temporal interpolation approaches require the evaluation of weights at each point of interest. A widely used interpolation approach is *kriging*. However, kriging has a computational cost that scales as the cube of the number of data points N, resulting in cubic time complexity for each point of interest, which leads to a time complexity of O(N^4) for interpolation at O(N) points. In this work, we formulate the kriging problem, to first reduce the computational cost to O(N^3). We use an iterative solver (Saad, 2003), and further accelerate the solver using fast summation algorithms like GPUML (Srinivasan and Duraiswami, 2009) or FIGTREE (Morariu et al., 2008). We illustrate the speedup on synthetic data and compare the performance with other standard kriging approaches to demonstrate substantial improvement in the performance of our approach. We then apply the developed approach on ocean color data from the Chesapeake Bay and present some quantitative analysis of the kriged results.

1 Introduction

Kriging (Isaaks and Srivastava, 1989) is a group of geostatistical techniques to interpolate the value of a random field (e.g., the elevation, z, of the landscape as a function of the geographic location) at an unobserved location from observations of its value at nearby locations. It belongs to a family of linear least squares estimation algorithms that are used in several geostatistical applications. It has its origin in mining application, where it was used to estimate the changes in ore grade within the mine (Krige, 1951). Kriging has since been applied in several scientific disciplines including atmospheric science, environmental monitoring and soil management.

Kriging can be linear or non-linear. Simple kriging, ordinary kriging and universal kriging are linear variants. Indicator kriging, log-normal kriging and disjunctive kriging are non-linear variants, that were developed to account for models where the best predictor is not linear. Moyeed and Papritz (2002) show that the performance of linear and non-linear kriging are comparable except in the use of skewed data where non-linear kriging performs better.

With improved sensors and ease of data collection, the amount of data available to krige has increased by several fold. One drawback of kriging is the computational cost for large data sizes. One approach to allow kriging of large data is to use local neighborhood kriging where only the closest observations are used for each prediction. Although computationally attractive, the methods require a local neighborhood for each location where the prediction is made, and predicting on a fine grid is still computationally demanding. Another disadvantage is the discontinuity in prediction along the peripheries of the local regions.

Another strategy for acceleration is to approximate the covariance matrix to result in a sparser kriging system. Furrer et al. (2006) use tapering to sparsify the covariance matrix in simple kriging and thus reduce the complexity of the least squares. Memarsadeghi and Mount (2007) extend a similar idea to ordinary kriging. Kammann and Wand (2003) use a low rank approximation to the covariance matrix to reduce the space and time complexity. Sakata et al. (2004) use the Sherman-Morrison-Woodbury formula on the sparsified covariance matrix with spatial sorting. The performance of all these approaches depends on the underlying data, and reduces dramatically whenever the recording stations are located close to each other.

Alternatively, fast algorithms to solve the exact kriging problem have also been proposed. Hartman and Hssjer (2008) build a Gaussian Markov random field to represent the station and the kriging points of interest to accelerate the kriging solution. Memarsadeghi et al. (2008) use fast summation algorithms (Yang et al., 2004; Raykar and Duraiswami, 2007) to krige in linear time (O(N)). However, the speedup in these approaches are dependent on the distribution of the station and kriged locations. In this pa-

*Balaji Vasan Srinivasan and Ramani Duraiswami are with the Perceptual Interfaces and Reality Laboratory, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, Email: [balajiv,ramani]@umiacs.umd.edu

†Raghu Murtugudde is from the Earth System Science Interdisciplinary Center, University of Maryland, College Park, MD 20742, Email: ragu@essic.umd.edu

per, we propose to use a graphical processing unit (GPU) based fast algorithm to solve the exact kriging system. We first formulate the kriging problem to solve it in O(kN^2) using iterative solvers (Saad, 2003) and then parallelize the computations on a GPU to achieve "near" real-time performance. Unlike other approaches discussed before, our acceleration is independent of station location and distribution and does not rely on covariance approximation.

The paper is organized as follows: In Section 2, we introduce the kriging problem and formulate the mean and covariances for an ordinary kriging system. In Section 3, we propose our modification to the kriging formulation to solve it in O(kN^2). In Section 4, we discuss the acceleration of our formulation on a graphical processor. In Section 5, we discuss our experiments on various synthetic, spatial and spatio-temporal datasets to illustrate the performance of our kriging algorithm.

2 Linear kriging

Linear kriging is divided into simple kriging (known mean), ordinary kriging (unknown but constant mean) and universal kriging (the mean is an unknown linear combination of known functions), depending on the mean value specification. We shall restrict the discussion here to ordinary kriging, however the approach that we propose in Section 3 and the accelerations in Section 4 are generic and apply to other categories as well.

Ordinary kriging is widely used because it is statistically the *best linear unbiased estimator (B.L.U.E)*. Ordinary kriging is linear because its estimates are linear combination of of the available data. It is unbiased because it attempts to keep the mean residual to be zero. Finally, it is called *best* because it tries to minimize the residual variance.

2.1 B.L.U.E Formulation

Let the data be sampled at N locations $(x_1, x_2, \ldots, x_N)$, and the corresponding values be $v_1, v_2, \ldots, v_N$. The value $\hat{v_j}$ at an unknown location $\hat{x_j}$ is estimated as a weighted linear combination of v's, given by,

$$\tilde{v_j} = \sum_{i=1}^{N} w_i v_i. \tag{1}$$

Here, $\tilde{v_j}$ is the estimate, and let $\hat{v_j}$ be the actual value (unknown) at $\hat{x_j}$. To find the weights, the values (v_i and $\hat{v_j}$) are assumed to be stationary random functions,

$$E[v_i] = E[\hat{v_j}] = E(v). \tag{2}$$

For unbiased estimates,

$$\begin{aligned} E[\hat{v_j} - \tilde{v_j}] &= 0 \\ E[\hat{v_j}] - E[\tilde{v_j}] &= 0 \\ E[v] - E\left[\sum_{i=1}^{N} w_i v_i\right] &= 0 \\ E[v] - \sum_{i=1}^{N} w_i E[v] &= 0 \\ \sum_{i=1}^{N} w_i &= 1 \end{aligned} \tag{3}$$

Let the residue be r_j;

$$r_j = \tilde{v_j} - \hat{v_j}. \tag{4}$$

Therefore, the residual variance is given by,

$$Var(r_j) = Cov\{\tilde{v_j}\tilde{v_j}\} - 2Cov\{\tilde{v_j}\hat{v_j}\} + Cov\{\hat{v_j}\hat{v_j}\} \tag{5}$$

The first term can further be simplified as follow,

$$\begin{aligned} Cov\{\tilde{v_j}\tilde{v_j}\} &= Var\{\tilde{v_j}\} = Var\{\sum_{i=1}^{N} w_i v_i\} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{N} w_i \cdot w_k \cdot Cov\{v_i v_k\} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{N} w_i w_k \hat{C}_{ik} \end{aligned} \tag{6}$$

The second term can be written as,

$$\begin{aligned} Cov\{\tilde{v_j}\hat{v_j}\} &= Cov\{\left(\sum_{i=i}^{N} w_i v_i\right)\hat{v_j}\} \\ &= \sum_{i=1}^{N} w_i \cdot Cov\{v_i\hat{v_j}\} = \sum_{i=1}^{N} w_i \hat{C}_{i0} \end{aligned}$$

Finally, assuming that the random variables have the same variance σ_v^2, the third term can be expressed as

$$Cov\{\hat{v_j}\hat{v_j}\} = \sigma_v^2 \tag{7}$$

Substituting from Eqs. (6-7) in Eq. (5),

$$Var(r_j) = \sum_{i=1}^{N}\sum_{k=1}^{N} w_i w_k \hat{C}_{ik} + \sum_{i=1}^{N} w_i \hat{C}_{i0} + \sigma_v^2. \tag{8}$$

For ordinary kriging, it is required to find w by minimizing $Var(r_j)$ with respect to w subject to the constraint Eq. 3. This can be written as the minimization of the penalized cost function,

$$\begin{aligned} J(w) = \sum_{i=1}^{N}\sum_{k=1}^{N} w_i w_k \hat{C}_{ik} &+ \sum_{i=1}^{N} w_i \hat{C}_{i0} \\ +\sigma_v^2 &+ 2\lambda\left(\sum_{i=1}^{N} w_i - 1\right), \end{aligned} \tag{9}$$

with 2λ the Lagrange multiplier. Taking derivatives of J with respect to w and λ,

$$\frac{\partial J}{\partial w_i} = 2\sum_{k=1}^{N} w_k \hat{C}_{i,k} - 2\hat{C}_{i0} + 2\lambda \tag{10}$$

$$\frac{\partial J}{\partial \lambda} = \sum_{i=1}^{N} w_i - 1 \tag{11}$$

Setting this to zero, we get the following system to solve, to obtain the weights w and λ,

$$\begin{pmatrix} \hat{C}_{11} & \dots & \hat{C}_{1n} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \hat{C}_{n1} & \dots & \hat{C}_{nn} & 1 \\ 1 & \dots & 1 & 0 \end{pmatrix} \times \begin{pmatrix} w_1 \\ \vdots \\ w_N \\ \lambda \end{pmatrix} = \begin{pmatrix} \hat{C}_{10} \\ \vdots \\ \hat{C}_{N0} \\ 1 \end{pmatrix}$$

$$\Rightarrow \hat{\mathbf{C}}\mathbf{w} = \hat{\mathbf{c}}_* \tag{12}$$

In order to obtain the kriged output at M locations Eq. 12 needs to be solved at each location.

$$\mathbf{v}_* = \hat{\mathbf{v}}^{\mathbf{T}}\hat{\mathbf{C}}^{-\mathbf{1}}\hat{\mathbf{C}}_*, \tag{13}$$

where,

$$\mathbf{v}_* = \begin{pmatrix} \tilde{v}_{\tilde{1}} \\ \vdots \\ \tilde{v}_{\tilde{M}} \end{pmatrix}, \hat{\mathbf{v}} = \begin{pmatrix} \tilde{v}_{\tilde{1}} \\ \vdots \\ \tilde{v}_{\tilde{M}} \\ 0 \end{pmatrix}$$

$$\hat{\mathbf{C}}_* = \begin{pmatrix} \hat{C}_{\tilde{1}1} & \dots & \hat{C}_{\tilde{1}N} \\ \vdots & \ddots & \vdots \\ \hat{C}_{\tilde{M}1} & \dots & \hat{C}_{\tilde{M}N} \\ 1 & \dots & 1 \end{pmatrix}$$

2.2 Covariance functions

There are two ways of specifying the covariances C_{ij}, either by a standard function or by evaluating it empirically at each location. The latter approach is not suitable when there are large number of stations, or when it is required to krige at a non-station location. A functional form of covariance is preferred in these cases. The covariance function is generally chosen to reflect prior information. In the absence of such knowledge, the Gaussian function is the most widely used covariance (Isaaks and Srivastava, 1989),

$$k_{ij} = s \exp\left(-\frac{\|x_i - x_j\|^2}{h^2}\right). \tag{14}$$

Another advantage with a functional representation is that it is possible to krige by computing the matrix C on-the-fly thus saving a lot of space requirements.

2.3 Computation complexity

Evaluation of a single set of weights w for a given location requires the solution of the system in Eq. (12) and has a computational complexity of O(N^3), N being the number of samples. Further, to get the weights at M locations, the complexity increases to O(MN^3). For $M \approx N$, the assymptotic complexity is O(N^4), this is undesirable for large N.

3 Proposed approach

Without loss of generality, the system in Eq. (13) can be transposed,

$$\mathbf{v}_* = \hat{\mathbf{C}}_*^{\mathbf{T}}\hat{\mathbf{C}}^{-\mathbf{1}}\hat{\mathbf{v}}. \tag{15}$$

The covariance functions C_{ij} are assumed to be symmetric in our discussions, therefore $\hat{\mathbf{C}}^{\mathbf{T}} = \hat{\mathbf{C}}$; hence there is no difference in transposing. However, this strategy applies to asymmetric covariance matrices as well. Now, the kriging comprises of solving a linear system followed by a covariance matrix-vector product; thus resulting in a complexity of O($N^3 + N^2$). This is an order reduction from the original formulation. Note that, in this approach, the weights are not evaluated explicitly, and so the storage is also avoided. Writing the two steps of the new formulation,

$$\text{Solve for } \mathbf{y}, \hat{\mathbf{C}}\mathbf{y} = \hat{\mathbf{v}} \tag{16}$$

$$\mathbf{v}_* = \hat{\mathbf{C}}_*^{\mathbf{T}}\mathbf{y}$$

Such a formulation makes kriging similar to the training and prediction in Gaussian process regression (Rasmussen and Williams, 2005), a Bayesian machine learning approach (Bishop, 2006). The Gaussian process formulation allows the definition of a variance (V_j) (Rasmussen and Williams, 2005) at the j^{th} kriged location is given by,

$$V_{\tilde{j}} = \hat{C}_{\tilde{j}\tilde{j}} - \hat{V}_{\tilde{j}} \tag{17}$$

where, $\hat{V}_{\tilde{j}}$ is given by,

$$\hat{V}_{\tilde{j}} = \begin{pmatrix} \hat{C}_{\tilde{j}1} \\ \vdots \\ \hat{C}_{\tilde{j}N} \end{pmatrix}^T \begin{pmatrix} \hat{C}_{11} & \dots & \hat{C}_{1N} \\ \vdots & \ddots & \vdots \\ \hat{C}_{N1} & \dots & \hat{C}_{NN} \end{pmatrix}^{-1} \begin{pmatrix} \hat{C}_{\tilde{j}1} \\ \vdots \\ \hat{C}_{\tilde{j}N} \end{pmatrix} \tag{18}$$

For simple kriging, iterative solution techniques like conjugate gradient (CG) and GMRES (Saad, 2003) can be used. For ordinary kriging, the linear system in Eq. (16) results in a "saddle-point" problem, and SymmLQ (Paige and Saunders, 1975) performs very well for these problems and is used here.

3.1 Possible accelerations

The key computation in each iteration of SymmLQ/CG/GMRES is the covariance matrix-vector product. By using fast single processor algorithms (Morariu et al., 2008) or parallelization, the cost of this matrix-vector product in each iteration can be significantly reduced.

Single processor accelerations use approximation algorithms to compute the matrix-vector product in linear time

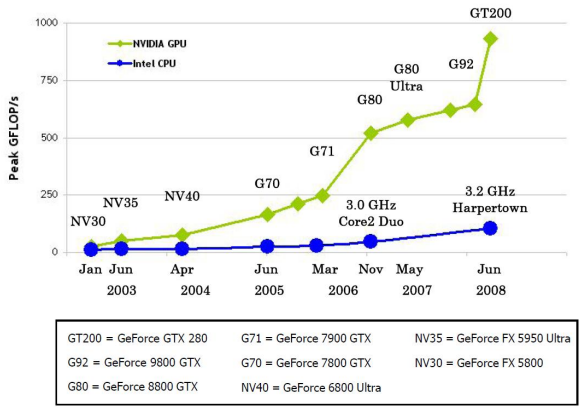


Figure 1: *Growth in the CPU and GPU speeds over the last* 6 *years on benchmarks (Image from NVIDIA (2008))*

O(N), within some known error bound. However, the performance is dependent on the distribution of the station locations. Maximum speedups are obtained when the data is recorded at relatively uniform sampling.

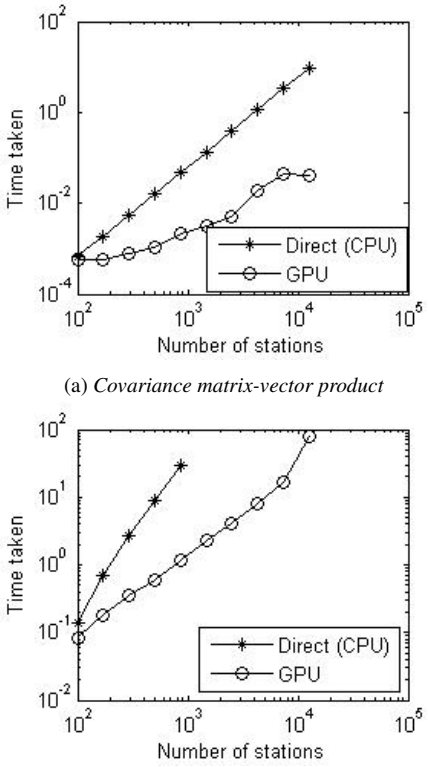
Alternatively, the matrix-vector product can be divided across multiple cores. No approximation is involved here and this has a complexity of O(N^2). An advantage of this approach is that it is independent of the locations.

Strategy: Both these approaches have their own advantages/disadvantages. A parallel implementation is independent of the station locations and hence is useful when the data are recorded at sparse locations. If it is known apriori that the station locations are equally space (to some degree), single-processor based accelerations are more beneficial. In this paper, we use a parallel approach on a graphical processor to accelerate the matrix vector product. For well-gridded data, our approach will be marginally slower compared to a linear approximation like FIGTREE (Morariu et al., 2008).

4 Graphical processors

Computer chip-makers are no longer able to easily improve the speed of processors, with the result that computer architectures of the future will have more cores, rather than more capable faster cores. This era of multi-core computing requires that algorithms be adapted to the data parallel architecture. If algorithms can be so devised the benefit is a significant improvement in performance. A particularly capable set of data parallel processors are the graphical processors, which have evolved into highly capable compute coprocessors. A graphical processing unit (GPU) is a highly parallel, multi-threaded, multi-core processor with tremendous computational horsepower.

In 2008, while the fastest Intel CPU could achieve only $\sim$ 50 Gflops speed theoretically, GPUs could achieve $\sim$ 950 Gflops on actual benchmarks (NVIDIA, 2008). Fig. 1 shows the relative growth in the speeds of NVIDIA GPUs and Intel CPUs as of 2008 (similar numbers are reported for AMD/ATI CPUs and GPUs). The recently announced FERMI architecture significantly improves these benchmarks. Moreover, GPUs power utilization per flop is an order of magnitude better. GPUs are particularly well-suited for data parallel computation and are designed as a single-program-multiple-data (SPMD) architecture with very high arithmetic intensity (ratio of arithmetic operation to memory operations). However, the GPU does not have the functionalities of a CPU like task-scheduling. Therefore, it can efficiently be used to assist the CPU in its operation rather than replace it.



(a) *Covariance matrix-vector product*

(b) *SymmLQ*

Figure 2: *Speedup obtained with GPUML (Srinivasan and Duraiswami, 2009) for matrix-vector product (Fig. 2a) and kriging (Fig. 2b)*

The covariance matrix vector product can also be viewed as a weighted summation of covariance functions, and GPU-accelerated summations are available as an open source, GPUML (Srinivasan and Duraiswami, 2009) and we used this to accelerate the kriging. NVIDIA GPUs are equipped with an easy programming model called *Compute Unified Device Architecture (CUDA)* (NVIDIA, 2008) and were used here.

4.1 Speedup

For the Gaussian covariance function, the speedup obtained with GPUML is shown in Fig. 2a. By using GPUML over each iteration, the speedup is further increased over that for SymmLQ (ordinary kriging) in Fig. 2b.

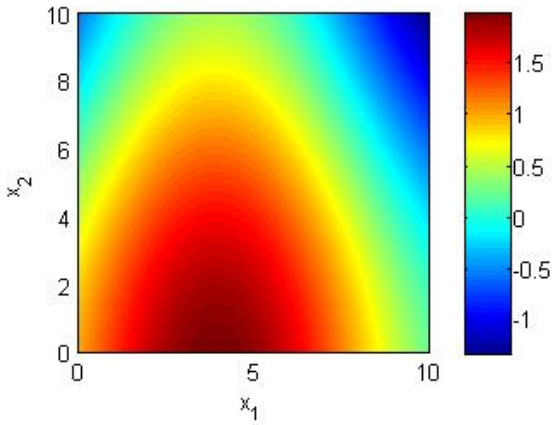


Figure 3: *Synthetic surface generated and sampled to test the performance of our kriging*

5 Experiments

All our experiments were performed on a Intel Core2 quad core 3GHz processor with 4GB RAM. We used the NVIDIA GTX280 graphical processor with 240 cores arranged as 30 multi-processors and a common 1GB global memory.

5.1 Experiment1: Performance analysis

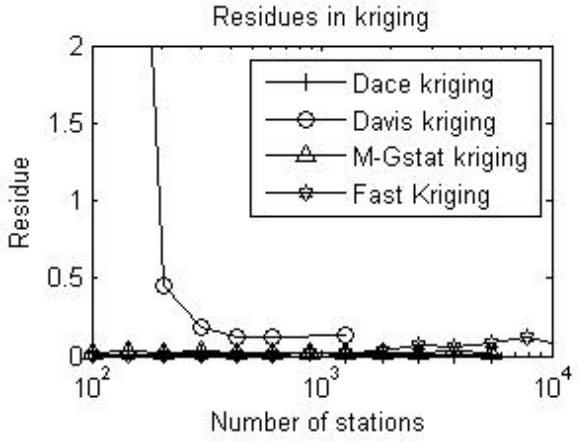
In this experiment, we tested our kriging approach on a $2-$dimensional synthetic data in Sakata et al. (2004). The data was sampled randomly on a $2-$d grid, and value at each sampled location was generated using the relation,

$$f(x_1, x_2) = \sin(0.4x_1) + \cos(0.2x_2 + 0.2), \quad (19)$$
$$0.0 \leq x_1, x_2 \leq 10.0.$$

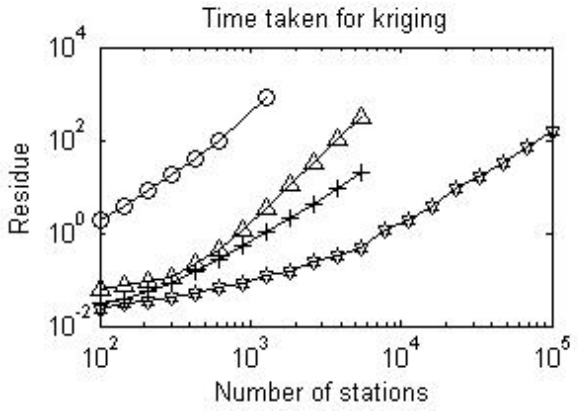
The surface represented by such a function is given in Fig. 3. Although the surface is not so complex, we used different sampling size to test the speed of various kriging approaches across "number of stations". We compared our approach against several open-source kriging packages: Dace kriging (Lophaven et al., 2002), Davis kriging (Lafleur, 1998) (algorithm based on Davis (1990)) and mGstat kriging (Hansen, 2004). All these packages are highly optimized for best kriging performance. The comparison of the residues and time taken by various kriging approaches is shown in Fig. 4. It can be seen that the proposed approach has the best time performance for comparable residues.

In order to further emphasize the quality of our kriging, we performed spatial interpolation on the "Stanford bunny" point cloud data[1] shown in Fig. 5a using the ordinary kriging approach. The point cloud contains 35947 points along the surface of the bunny. In order to krige this data, the cloud was extended to size 104502 by adding points along the normal inside and outside and assigning values $0, -1$ and 1 respectively. The surface was kriged at $8,000,000$ points (regular spatial grid $200 \times 200 \times 200$)

[1] `http://graphics.stanford.edu/data/3Dscanrep/`



(a) *Residues*



(b) *Time taken*

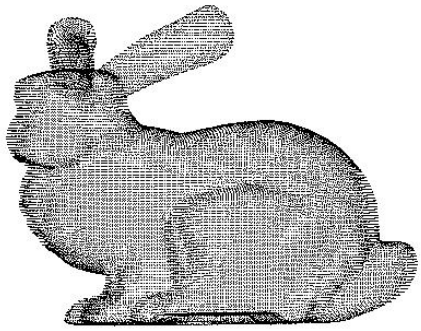
Figure 4: *Performance comparison across different kriging approaches for the synthetic data in (Sakata et al., 2004)*

from which the isosurface was found using standard routines (Turk and O'Brien, 2002). Our GPU implementation took only 7 minutes to krige at the 8-million points while a direct implementation would have taken ~ 2 days on a state-of-the-art PC. The quality of the resulting interpolation can be seen from Fig. 5b.

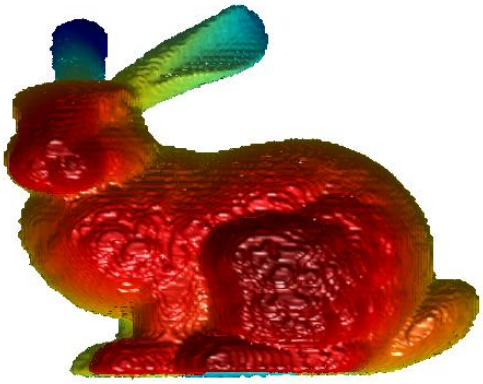
5.2 Experiment2: Chesapeake Bay Data

We applied our kriging to the ocean color data from the Chesapeake bay. The ocean color is an indicator of the concentration of chlorophyll in the water. The data are composite maps of SeaWiFS chlorophyll measured at a 2km resolution in the Chesapeake bay region recorded every 7-days between 1998 and 2002 by NASA. The region across which the data was recorded is shown in Fig. 6. There were some days for which no recording available.

We used the Gaussian covariance matrix (Eq. 14) and set the length scales (h in Eq. 14) to 0.06 degrees along the latitude and longitude scales and 15 days along the temporal scale. It was observed that on an average our approach took $70 - 80$s to krige the data for a single day (a direct implementation would have taken ~ 8 hours for the same. The kriging results for the 15^{th} day of each



(a) *Point cloud*



(b) *Kriged surface*

Figure 5: *Spatial interpolation of the "Stanford" bunny point cloud data. Our approach took only* 7 *minutes, a direct implementation would have taken* ~ 2 *days.*

month in the year is shown in Fig.7. It can be seen that the chlorophyll concentration is low during the cooler periods of the year due to the relative inactivity of the alga, whereas it increases by several folds during the warmer periods (April-August) because of the blooms.

Fig. 8 shows the result of a variability analysis on the kriged ocean color data over the entire $5-$year period. It can be observed that ocean color has a large variability in regions very close to the land region. This is in line with what is expected because of the well-known fact that human activity (in the land regions) contribute to a large extent to the chlorophyll concentration and hence a larger variation; thus further validating our kriging.

6 Conclusion

In this paper, we have proposed a new formulation for the kriging problem, that can be solved using iterative solvers like conjugate gradient, SymmLQ. We further accelerate each iterations on a GPU by using fast covariance matrix vector products. The resulting kriging is illustrated to be faster than many available tools and further validated on

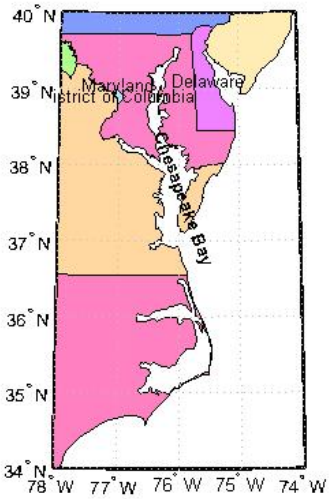


Figure 6: *Region across which ocean color data were available (indicated in white color)*

synthetic and recorded data.

Acknowledgement

This work is a part of Chesapeake Bay Forecast System and we gratefully acknowledge NOAA for funding the project. We also thank Drs. Nail Gumerov, Clarissa Anderson and Jim Beauchamp, for providing us the data that we used in this paper and for the valuable feedbacks that helped us improve our kriging tool.

References

Bishop, C., 2006: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Davis, J., 1990: *Statistics and Data Analysis in Geology*. John Wiley & Sons, Inc., New York, NY, USA.

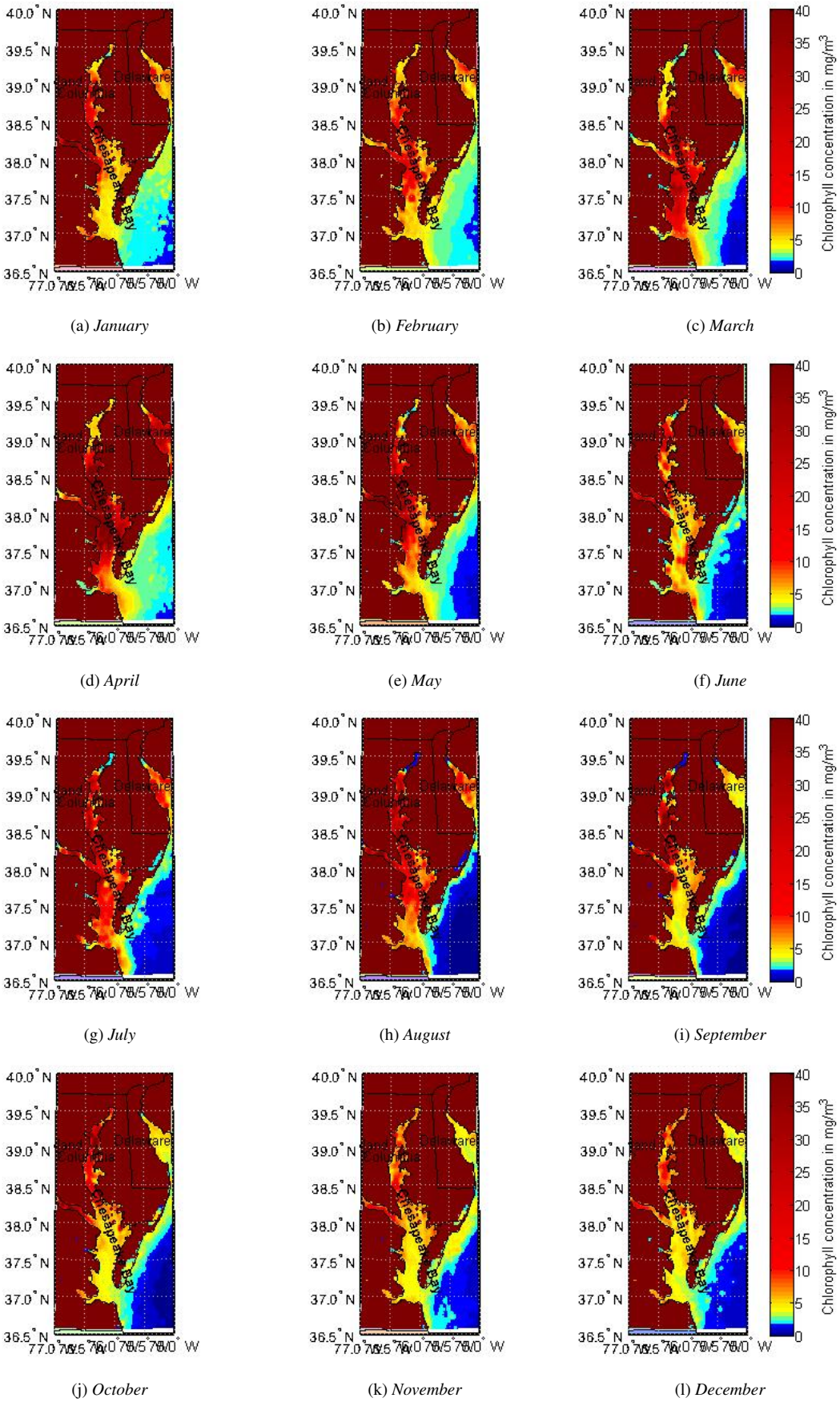
Furrer, R., M. Genton, and D. Nychka, 2006: Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, **15**, 502–523.

Hansen, T., 2004: mgstat: A geostatistical matlab toolbox. Online web resource.
URL `http://mgstat.sourceforge.net/`

Hartman, L. and O. Hssjer, 2008: Fast kriging of large data sets with Gaussian markov random fields. *Computational Statistics & Data Analysis*, **52**, 2331 – 2349.

Isaaks, E. and R. Srivastava, 1989: *Applied Geostatistics*. Oxford University Press, 542 pp.

Kammann, E. E. and M. P. Wand, 2003: Geoadditive models. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **52**, 1–18.



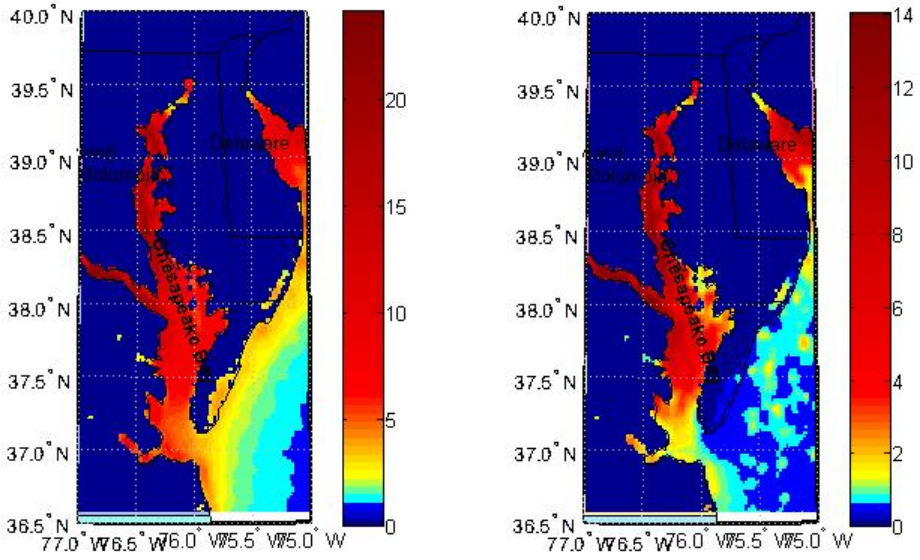
(a) *January* (b) *February* (c) *March*

(d) *April* (e) *May* (f) *June*

(g) *July* (h) *August* (i) *September*

(j) *October* (k) *November* (l) *December*

Figure 7: *Kriging result using the* $7-$*day Chesapeake bay ocean color data on th* 15^{th} *day of each month in* 2002



(a) *Mean* (b) *Standard deviation*

Figure 8: *Variability analysis: Mean and standard deviation of the kriged over the* 5 *year period from* 1998 *to* 2002.

Krige, D., 1951: A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, **52**, 119–139.

Lafleur, C., 1998: Matlab kriging toolbox. Online web resource. URL `http://www2.imm.dtu.dk/~hbn/dace/`

Lophaven, S., H. Nielsen, and J. Sndergaard, 2002: Dace, a Matlab kriging toolbox. Online web resource. URL `http://www2.imm.dtu.dk/~hbn/dace/`

Memarsadeghi, N. and D. Mount, 2007: Efficient implementation of an optimal interpolator for large spatial data sets. *Proceedings of the 7th international conference on Computational Science, Part II*, Springer-Verlag, Berlin, Heidelberg, 503–510.

Memarsadeghi, N., V. C. Raykar, R. Duraiswami, and D. M. Mount, 2008: Efficient kriging via fast matrix-vector products. *Aerospace Conference, 2008 IEEE*, 1–7.

Morariu, V., B. Srinivasan, V. Raykar, R. Duraiswami, and L. Davis, 2008: Automatic online tuning for fast Gaussian summation. *Advances in Neural Information Processing Systems*. URL `http://sourceforge.net/projects/figtree/`

Moyeed, R. and A. Papritz, 2002: An empirical comparison of kriging methods for nonlinear spatial point prediction. *Mathematical Geology*, **34**, 365–386.

NVIDIA, 2008: *NVIDIA CUDA Programming Guide 2.0*.

Paige, C. C. and M. A. Saunders, 1975: Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, **12**, 617–629.

Rasmussen, C. and C. Williams, 2005: *Gaussian Processes for Machine Learning*. The MIT Press.

Raykar, V. and R. Duraiswami, 2007: The improved fast Gauss transform with applications to machine learning. *Large Scale Kernel Machines*, 175–201.

Saad, Y., 2003: *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics.

Sakata, S., F. Ashida, and M. Zako, 2004: An efficient algorithm for kriging approximation and optimization with large-scale sampling data. *Computer Methods in Applied Mechanics and Engineering*, **193**, 385 – 404.

Srinivasan, B. and R. Duraiswami, 2009: Scaling kernel machine learning algorithm via the use of GPUs. *GPU Technology Conference*, NVIDIA Research Summit. URL `http://www.umiacs.umd.edu/~balajiv/GPUML.htm`

Turk, G. and J. O'Brien, 2002: Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.*, **21**, 855–873.

Yang, C., C. Duraiswami, and L. Davis, 2004: Efficient kernel machines using the improved fast gauss transform. *Advances in Neural Information Processing Systems*.