Efficient Kriging via Fast Matrix-Vector Products

Nargess Memarsadeghi
NASA/GSFC, Code 587
Greenbelt, MD, 20771
Nargess.Memarsadeghi@nasa.gov

Vikas C. Raykar
Siemens Medical Solutions
Malvern, PA 19355
vikas.raykar@siemens.com

Ramani Duraiswami
University of Maryland
College Park, MD 20742
ramani@cs.umd.edu

David M. Mount
University of Maryland
College Park, MD 20742
mount@cs.umd.edu

Abstract—Interpolating scattered data points is a problem of wide ranging interest. Ordinary kriging is an optimal scattered data estimator, widely used in geosciences and remote sensing. A generalized version of this technique, called cokriging, can be used for image fusion of remotely sensed data. However, it is computationally very expensive for large data sets. We demonstrate the time efficiency and accuracy of approximating ordinary kriging through the use of fast matrix-vector products combined with iterative methods. We used methods based on the fast Multipole methods and nearest neighbor searching techniques for implementations of the fast matrix-vector products.

Keywords—geostatistics, image fusion, kriging, approximate algorithms, fast multipole methods, fast Gauss transform, nearest neighbors, iterative methods.

TABLE OF CONTENTS

1. INTRODUCTION

Scattered data interpolation is a problem of interest in numerous areas such as electronic imaging, smooth surface modeling, and computational geometry [1,2]. Our motivation arises from applications in geology and mining, which often involve large scattered data sets and a demand for high accuracy. For such cases, the method of choice is *ordinary kriging* [3]. This is because it is a best unbiased estimator [3–5]. Also in remote sensing, *image fusion* of multi-sensor data is often used to increase either the spectral or the spatial resolution of the images involved [6, 7]. A generalized version of the ordinary kriging, called *cokriging* [3, 5], can be used for image fusion of multi-sensor data [8, 9]. Unfortunately, ordinary kriging interpolant is computationally very expensive to compute exactly. For n scattered data points, computing the value of a single interpolant involves solving a dense linear system of order $n \times n$, which takes $O(n^3)$. This is infeasible for large n. Traditionally, kriging is solved approximately by local approaches that are based on considering only a relatively small number of points that lie close to the query point [3, 5]. There are many problems with this local approach, however. The first is that determining the proper neighborhood size is tricky, and is usually solved by *ad hoc* methods such as selecting a fixed number of nearest neighbors or all the points lying within a fixed radius. Such fixed neighborhood sizes may not work well for all query points, depending on local density of the point distribution [5]. Local methods also suffer from the problem that the resulting interpolant is not continuous. Meyer showed that while kriging produces smooth continuous surfaces, it has zero order continuity along its borders [10]. Thus, at interface boundaries where the neighborhood changes, the interpolant behaves discontinuously. Therefore, it is important to consider and solve the global system for each interpolant. However, solving such large dense systems for each query point is impractical.

Recently an approximation approach to kriging has been proposed based on a technique called *covariance tapering* [11, 12]. However, this approach is not efficient when covariance models have relatively large ranges. Also, finding a *valid* taper, as defined in [11], for different covariance functions is difficult and at times impossible (e.g. Gaussian covariance model).

In this paper, we address the shortcomings of the previous approaches through an alternative based on Fast Multipole Methods (FMM). The FMM was first introduced by Greengard and Rokhlin for fast multiplication of a structured matrix by a vector [13, 14]. If the Gaussian function is used for generating the matrix entries, the matrix-vector product is called the *Gauss transform*. We use efficient implementations of the Gauss transform based on the FMM idea (see [15,16]) in combination with the SYMMLQ iterative method [17] for solving large ordinary kriging systems. Billings *et al.* [18] had also suggested the use of iterative methods in combination with the FMM for solving such systems.

The remainder of this paper is organized as follows. Section

This work was done when Nargess Memarsadeghi and Vikas C. Raykar were graduate students in Computer Science Department of the University of Maryland at College Park.
The work of D. M. Mount has been supported in part by the National Science Foundation under grant CCR–0635099.

2 describes the ordinary kriging system and solving it via an iterative method. In Section 3 we introduce the matrix-vector products involved in solving such systems. We mention two existing efficient and approximate implementations of such products in Section 4. Section 5 describes our data sets. Our experiments and results are presented in Sections 6 and 7 respectively. Section 8 concludes this paper.

2. Ordinary Kriging via Iterative Methods

Kriging is an interpolation method named after Danie Krige, a South African mining engineer [5]. Kriging and its variants have been traditionally used in mining and geostatistics applications [3–5]. Kriging is also referred to as the *Gaussian process predictor* in the machine learning domain [19]. The most commonly used variant is called *ordinary kriging*, which is often referred to as a Best Linear Unbiased Estimator (BLUE) [3, 11]. It is considered to be *best* because it minimizes the variance of the estimation error. It is *linear* because estimates are weighted linear combination of available data, and is *unbiased* since it aims to have the mean error equal to zero [3]. Minimizing the variance of the estimation error forms the objective function of an optimization problem. Ensuring unbiasedness of the error imposes a constraint on this function. Formalizing this objective function with its constraint results in the following system [3, 5, 12].

$$\begin{pmatrix} C & L \\ L^T & 0 \end{pmatrix} \begin{pmatrix} w \\ \mu \end{pmatrix} = \begin{pmatrix} C_0 \\ 1 \end{pmatrix}, \tag{1}$$

where C is the matrix of points' pairwise covariances, L is a column vector of all ones and of size n, and w is the vector of weights $w_i, \ldots, w_n$. Therefore, the minimization problem for n points reduces to solving a linear system of size $(n+1)^2$, which is impractical for very large data sets via direct approaches. It is also important that matrix C be positive definite [3, 12]. Pairwise covariances are often modeled as a function of points' separation. These functions should result in a positive definite covariance matrix. Christakos [20] showed necessary and sufficient conditions for such permissible covariance functions. Two of these valid functions are the Gaussian and Spherical covariance functions [3, 5, 20]. In this paper, the Gaussian covariance function is used. For two points x_i and x_j, the Gaussian covariance function is defined as $C_{ij} = \exp\left(-3\|x_i - x_j\|^2/a^2\right)$, where a is the range of the covariance function.

For large data sets, it is impractical to solve the ordinary kriging system using direct approaches that take $O(n^3)$ time. Iterative methods generate a sequence of solutions which converge to the true solution in n iterations. In practice, however, we loop over $k \ll n$ iterations [21]. In particular, we used an iterative method called SYMMLQ which is appropriate for solving symmetric systems [17]. Note that the coefficient matrix in the ordinary kriging linear system while symmetric is *not* positive definite since it has a zero entry on its diagonal. Therefore, methods such as conjugate gradient are not applicable here [22]. The actual implementation of the SYMMLQ method requires one $O(n^2)$ *matrix-vector multiplication* per iteration. The storage is $O(n)$ since the matrix-vector multiplication can use elements computed on the fly without storing the matrix. Empirically the number of iterations required, k, is generally small compared to n leading to a computational cost of $O(kn^2)$.

3. Matrix-vector Multiplication

The $O(kn^2)$ quadratic complexity is still too high for large data sets. The core computational step in each SYMMLQ iteration involves the multiplication of a matrix $\mathbf{C}$ with some vector, say $\mathbf{q}$. For the Gaussian covariance model the entries of the matrix $\mathbf{C}$ are of the form $[\mathbf{C}]_{ij} = \exp\left(-3\|x_i - x_j\|^2/a^2\right)$. Hence, the j^{th} element of the matrix-vector product $\mathbf{Cq}$ can be written as $(\mathbf{Cq})_\mathbf{j} = \sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{q_i} \exp\left(-\mathbf{3}\|\mathbf{x_i} - \mathbf{x_j}\|^\mathbf{2}/\mathbf{a^2}\right)$–which is the weighted sum of n Gaussians each centered at x_i and evaluated at x_j.

*Discrete Gauss transform (*GT*)*

The sum of multivariate Gaussians is known as the *discrete Gauss transform* in scientific computing. In general, for each *target point* $\{y_j \in \mathbb{R}^d\}_{j=1}^m$ (which in our case are the same as the *source points* x_i) the discrete Gauss transform is defined as

$$G(y_j) = \sum_{i=1}^{n} q_i e^{-\frac{\|x_i - y_j\|^2}{h^2}}, \tag{2}$$

where h (in our case $h = a/\sqrt{3}$) is called the *bandwidth* of the Gaussian. Evaluating discrete Gauss transforms for m target points due to n different source locations arises in may applications. In this paper, the *Gauss transform* (GT) refers to this direct implementation, which takes $O(mn)$ time.

4. Fast Approximate Matrix-vector Multiplication

Various fast approximation algorithms [15, 23] have been proposed to compute the discrete Gauss transform in $O(m+n)$ time. These algorithms compute the sum to any arbitrary ϵ precision. For any $\epsilon > 0$, we define $\widehat{G}$ to be an *ϵ-exact* approximation to G if the maximum absolute error relative to the total weight $Q = \sum_{i=1}^{n} |q_i|$ is upper bounded by ϵ, *i.e.*,

$$\max_{y_j} \left[\frac{|\widehat{G}(y_j) - G(y_j)|}{Q} \right] \leq \epsilon. \tag{3}$$

The constant in $O(m+n)$ depends on the desired accuracy ϵ, which however can be *arbitrary*. At machine precision there is no difference between the direct and the fast methods. The method relies on retaining only the first few terms of an infinite series expansion for the Gaussian function. These methods are inspired by the fast multipole methods (FMM), originally developed for the fast summation of the potential fields generated by a large number of sources, such as those arising in gravitational potential problems [13]. The fast Gauss transform (FGT) is a special case where FMM ideas were used for

the Gaussian potential [23]. The improved fast Gauss transform (IFGT) is a similar algorithm based on a single different factorization and data structure. It is suitable for higher dimensional problems and provides comparable performance in lower dimensions [15].

*Improved fast Gauss transform (*IFGT*)*

IFGT is an efficient algorithm for approximating the Gauss transform. The *fast Gauss transform*, first proposed by Greengard and Strain [23], is an ϵ-exact approximation algorithm for the Gauss transform. This algorithm reduces the Gauss transform's computational complexity from $O(mn)$ to $O(m+n)$. However, this algorithm's constant factor grows exponentially with dimension d. Later improvements, including the IFGT algorithm, reduced this constant factor to asymptotically polynomial order in terms of d. The IFGT algorithm was first introduced by Yang *et al.* [15]. Their implementation did not use a sufficiently tight error bound to be useful in practice. Also, they did not adaptively select the necessary parameters to achieve the desired error bound. Raykar *et al.* later presented an approach that overcame these shortcomings [16, 24]. We used the implementation due to Raykar *et al.* We briefly describe some of the key ideas in IFGT. Please see [16,24] for more details. For any point $x_* \in \mathbb{R}^d$ the Gauss Transform at y_j can be written as,

$$\begin{aligned} G(y_j) &= \sum_{i=1}^{n} q_i e^{-\frac{\|y_j - x_i\|^2}{h^2}} \\ &= \sum_{i=1}^{n} q_i e^{-\frac{\|(y_j - x_*) - (x_i - x_*)\|^2}{h^2}}, \\ &= \sum_{i=1}^{n} q_i e^{-\frac{\|x_i - x_*\|^2}{h^2}} e^{-\frac{\|y_j - x_*\|^2}{h^2}} e^{\frac{2(y_j - x_*)\cdot(x_i - x_*)}{h^2}}. \end{aligned} \tag{4}$$

In Equation (4) the first exponential inside the summation $e^{-\|x_i - x_*\|^2/h^2}$ depends only on the source coordinates x_i. The second exponential $e^{-\|y_j - x_*\|^2/h^2}$ depends only on the target coordinates y_j. However for the third exponential, $e^{2(y_j - x_*)\cdot(x_i - x_*)/h^2}$, the source and target are entangled. The crux of the algorithm is to separate this entanglement via Taylor series using multi-index notation. The p-term truncated Taylor series expansion for $e^{2(y_j - x_*)\cdot(x_i - x_*)/h^2}$ can be written as [16, 24],

$$\begin{aligned} & e^{2(y_j - x_*)\cdot(x_i - x_*)/h^2} \\ &= \sum_{|\alpha| \le p-1} \frac{2^\alpha}{\alpha!} \left(\frac{y_j - x_*}{h}\right)^\alpha \left(\frac{x_i - x_*}{h}\right)^\alpha + error_p. \end{aligned}$$

The truncation number p is chosen based on the prescribed error ϵ. Ignoring error terms for now $G(y_j)$ can be approximated as,

$$\widehat{G}(y_j) = \sum_{i=1}^{n} q_i e^{-\|x_i - x_*\|^2/h^2} e^{-\|y_j - x_*\|^2/h^2} \left[\sum_{|\alpha| \le p-1} \frac{2^\alpha}{\alpha!} \left(\frac{y_j - x_*}{h}\right)^\alpha \left(\frac{x_i - x_*}{h}\right)^\alpha \right]. \tag{5}$$

Rearranging the terms Equation (5) can be written as

$$\begin{aligned} \widehat{G}(y_j) &= \sum_{|\alpha| \le p-1} \left[\frac{2^\alpha}{\alpha!} \sum_{i=1}^{n} q_i e^{-\|x_i - x_*\|^2/h^2} \left(\frac{x_i - x_*}{h}\right)^\alpha \right] \\ & \quad e^{-\|y_j - x_*\|^2/h^2} \left(\frac{y_j - x_*}{h}\right)^\alpha \\ &= \sum_{|\alpha| \le p-1} C_\alpha e^{-\|y_j - x_*\|^2/h^2} \left(\frac{y_j - x_*}{h}\right)^\alpha, \end{aligned}$$

where,

$$C_\alpha = \frac{2^\alpha}{\alpha!} \sum_{i=1}^{n} q_i e^{-\|x_i - x_*\|^2/h^2} \left(\frac{x_i - x_*}{h}\right)^\alpha.$$

The coefficients C_α can be evaluated separately in $O(n)$. Evaluation of $\widehat{G}(y_j)$ at m points is $O(m)$. Hence the computational cost has reduced from the quadratic $O(nm)$ to the linear $O(n+m)$. We have omitted the constants in the computational cost. A detailed analysis of the cost can be seen in [16, 24].

Thus far, we have used the Taylor series expansion about a certain point x_*. However if we use the same x_* for all the points we typically would require very high truncation number since the Taylor series is valid only in a small open ball around x_*. The IFGT algorithm uses a data adaptive space partitioning scheme–the farthest point clustering algorithm [25]–to divide the n sources into K spherical clusters– and then build a Taylor series at the center of each cluster.

The final algorithm has four stages. The first stage involves determining parameters of the algorithm based on specified error bounds, bandwidth, and data distribution. Second, the d-dimensional space is subdivided using a k-center clustering [25]. Next, a truncated representation of the Gaussian inside each cluster is built using a set of decaying basis functions. Finally, the influence of all the data in a neighborhood using coefficients at cluster centers are collected and the approximate GT is evaluated. Please see [16, 24] for details.

GT ***with nearest neighbors (***GTANN***)***

GTANN is also an efficient algorithm for calculating matrix-vector products. This method was implemented by Raykar [26], where it is referred to as the FigTree method. This method is most effective when the Gaussian models being used have small ranges, while IFGT gives good speed-ups when dealing with large range values.

First, based on the desired error bound, a search radius is calculated. Then, for each target point, source points within that radius are considered. Since the Gaussian function dissipates very rapidly, nearby points have the greatest influence. These

source points are calculated via fixed-radius nearest neighbor search routines of the ANN library [27]. Finally, for each target point, their nearest neighbor source points are calculated in matrix-vector product calculations, involving the covariance matrix C in the ordinary kriging system.

5. Data sets

We generated three sets of sparse data sets. For the first set, the number of sampled points varied from 1000 up to 5000, while their covariance model had a small range value of 12. For the second set, we varied the number of samples in the same manner, except that the points' covariance model had a larger range equal to 100. Finally, we sampled 5000 points from dense grids, where points' covariance model had ranges equal to $a = 12, 24, 100, 250$, and 500. For each input data set we use 200 query points which are drawn from the same dense grid but are not present in the sampled data set. One hundred of these query points were sampled uniformly from the original grids. The other 100 query points were sampled from the same Gaussian distributions that were used in the generation of a small percentage of the sparse data.

6. Experiments

We used the SYMMLQ iterative method as our linear solver. We set the desired solutions' relative error, or the convergence criteria for the SYMMLQ method, to $\epsilon_2 = 10^{-3}$. Thus, if SYMMLQ is implemented exactly, we expect the relative error to be less than 10^{-3}. The exact error is likely to be higher than that. We developed three versions of the ordinary kriging interpolator, each of which calculates the matrix-vector products involved in the SYMMLQ method differently.

Gauss Transform (GT): Computes the matrix-vector product exactly.
Improved Fast Gauss Transform (IFGT): Approximates the matrix-vector product to the ϵ precision via the IFGT method mentioned in Section 4.
Gauss transform with nearest neighbors (GTANN): Approximates the matrix-vector product using the GTANN method mentioned in Section 4.

All experiments were run on a Sun Fire V20z running Red Hat Enterprise release 3, using the g++ compiler version 3.2.3. Our software is implemented in C++ and uses the Geostatistical Template Library (GsTL) [28] and Approximate Nearest Neighbor library (ANN) [27]. GsTL is used for building and solving the ordinary kriging systems, and ANN is used for finding nearest neighbors when using the GTANN approach. In all cases, for the IFGT and GTANN approaches we required the approximate matrix-vector products to be evaluated within $\epsilon = 10^{-4}$ accuracy. All experiments' results are averaged over five runs. We designed three experiments to study the effect of covariance ranges and number of data points on performances of our different ordinary kriging versions.

Experiment 1: We varied number of scattered data points from 1000 up to 5000, with a small fixed Gaussian covariance model of range $a = 12$.
Experiment 2: We varied number of sampled points. This time, points had a larger range equal to $a = 100$ for their covariance model.
Experiment 3: Finally, we examined the effect of different covariance ranges equal to $a = 12, 25, 100, 250$, and 500 on a fixed data set of size 5000.

7. Results

In this section we compare both the quality of results and the speed-ups achieved for solving the ordinary kriging system via iterative methods combined with fast approximate matrix-vector multiplication techniques.

Figure 1 presents results of the first experiment mentioned. When utilizing approximate methods IFGT and GTANN the exact residuals are comparable to those obtained from GT. The IFGT approach gave speed-ups ranging from 1.3–7.6, while GTANN resulted in speed-ups ranging roughly from 50–150. This is mainly due to the fact that the covariance function's range is rather small. Since there are only a limited number of source points influencing each target point, collecting and calculating the influence of all source points for each target point (the IFGT approach) is excessive. The GTANN approach works well for such cases by considering only the nearest source points to each target point. In both cases, the speed-ups grow with number of data points. Algorithms perform similarly for points from the Gaussian distribution to those from uniform distribution.

Figure 2 shows results of the second experiment. While the GTANN approach did not result in significant speed-ups, the IFGT gave constant factor speed-ups ranging roughly from 1.5 to 7.5, as we increased the number of data points. The IFGT approach results in larger residuals for small data sets, and when solving the ordinary kriging system for query points from the uniform distribution. In particular, for $n = 1000$ and $n = 2000$, the performance of IFGT is not acceptable with respect to the exact residuals calculated. This poor overall result for these cases is because the SYMMLQ method did not meet its convergence criteria and reached maximum number of iterations. Increasing the required accuracy for the IFGT algorithm, by changing the $\epsilon = 10^{-4}$ to 10^{-6} resolved this issue and gave residuals comparable to those obtained from the GT method. As the number of points increases, the quality of results approaches those of the exact methods. For the query points from the Gaussian distribution, the quality of results are comparable to the exact method, when using the IFGT approach. The GTANN approach also results in comparable residuals to the exact methods in all cases.

Figure 3 presents results of the last set of experiments. In all cases, the quality of results is comparable to those obtained from exact methods. The IFGT approach resulted in speed-ups of 7–15 in all cases. The GTANN approach is best when

used for covariance functions with small range values of 12, and 25. While the GTANN approach is slower than the exact methods for range values larger than 100, it results in speed-up factors of 151–153, and 47–49 for range values 12 and 25 respectively. Thus, the GTANN approach is efficient for small range values, and so is the IFGT approach for large ranges.

8. CONCLUSIONS

We integrated efficient implementations of the Gauss Transform for solving ordinary kriging systems. We examined the effect of number of points and the covariance functions' ranges on the running time for solving the system and the quality of results. The IFGT is more effective as number of data points increases. Our experiments using the IFGT approach for solving the ordinary kriging system demonstrated speed-up factors ranging from 7–15 when using 5000 points. Based on our tests on varying number of data points, we expect even higher speed-up factors compared to the exact method when using larger data sets. In almost all cases, the quality of IFGT results are comparable to the exact methods. The GTANN approach outperformed the IFGT method for small covariance range values, resulting in speed-up factors as high as 153 and 49 respectively. The GTANN approach is slower than the exact methods for large ranges (100 and over in our experiments), and thus is not recommended for such cases. The quality of results for GTANN was comparable to the exact methods in all cases. Please see [26, 29] for details of methods used in this work.

Future work involves efficient kriging via fast approximate matrix-vector products for other covariance functions, where a factorization exists. We also plan on using preconditioners [21] for our iterative solver, so that the approximate versions converge faster, and we obtain even higher speed-ups.

REFERENCES

[1] I. Amidror, "Scattered data interpolation methods for electronic imaging systems: A survey," *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 157–176, April 2002.

[2] P. Alfeld, "Scattered data interpolation in three or more variables," *Mathematical methods in computer aided geometric design*, pp. 1–33, 1989.

[3] E. H. Isaaks and R. M. Srivastava, *An Introduction to Applied Geostatistics*. New York, Oxford: Oxford University Press, 1989.

[4] A. G. Journel and C. J. Huijbregts, *Mining Geostatistics*. New York: Academic Press Inc, 1978.

[5] P. Goovaerts, *Geostatistics for Natural Resources Evaluation*. New York, Oxford: Oxford University Press, 1997.

[6] D. L. Hall, *Mathematical techniques in multisensor data fusion*. Norwood: Artech House Inc, 1992.

[7] Y. Zhang, "Understanding image fusion," *Photogrammetric Engineering and Remote Sensing*, pp. 657–661, June 2004.

[8] N. Memarsadeghi, J. L. Moigne, D. M. Mount, and J. Morisette, "A new approach to image fusion based on cokriging," in *the Eighth International Conference on Information Fusion*, vol. 1, July 2005, pp. 622–629.

[9] N. Memarsadeghi, J. L. Moigne, and D. M. Mount, "Image fusion using cokriging," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'06)*, July 2006, pp. 2518 – 2521.

[10] T. H. Meyer, "The discontinuous nature of kriging interpolation for digital terrain modeling," *Cartography and Geographic Information Science,*, vol. 31, no. 4, pp. 209–216, 2004.

[11] R. Furrer, M. G. Genton, and D. Nychka, "Covariance tapering for interpolation of large spatial datasets," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 502–523, September 2006.

[12] N. Memarsadeghi and D. M. Mount, "Efficient implementation of an optimal interpolator for large spatial data sets," in *Proceedings of the International Conference on Computational Science (ICCS'07)*, ser. Lecture Notes in Computer Science, vol. 4488. Springer–Verlag, May 2007, pp. 503–510.

[13] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulation," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348, 1987.

[14] L. Greengard, "The rapid evaluation of potential fields in particle systems," Ph.D. dissertation, Yale, NYU, 1987.

[15] C. Yang, R. Duraiswami, and L. Davis, "Efficient kernel machines using the improved fast Gauss transform." in *Advances in Neural Information Processing Systems*, L. K. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005, pp. 1561–1568.

[16] V. C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov, "Fast computation of sums of Gaussians in high dimensions," Department of Computer Science, University of Maryland, College Park, MD, 20742, Tech. Rep., 2005, CS-TR-4767.

[17] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 12, no. 4, pp. 617–629, September 1975.

[18] S. D. Billings, R. K. Beatson, and G. N. Newsam, "Interpolation of geophysical data using continuous global surfaces," *Geophysics*, vol. 67, no. 6, pp. 1810–1822, November-December 2002.

[19] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[20] G. Christakos, "On the problem of permissible covariance and variogram models," *Water Resources Research*, vol. 20, no. 2, pp. 251–265, February 1984.

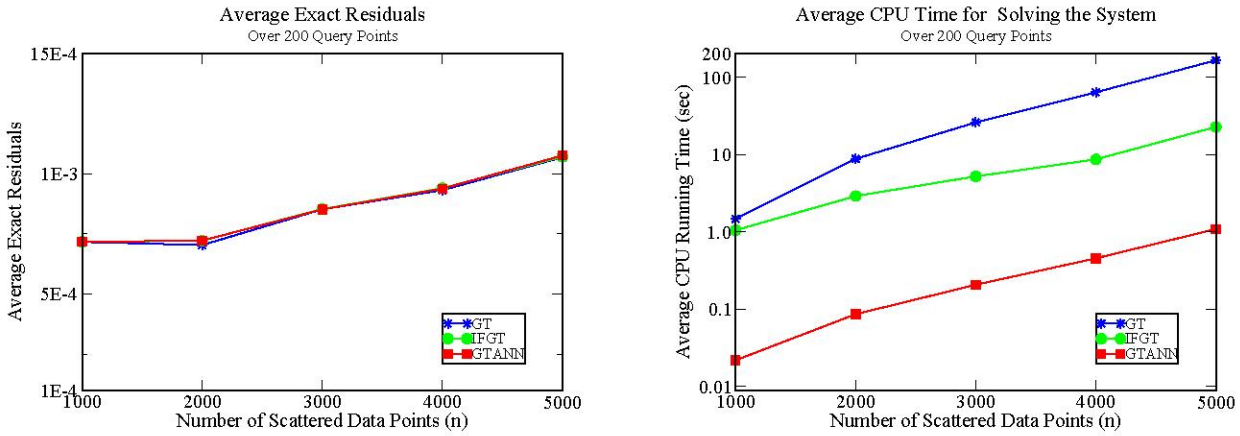


Figure 1. Experiment 1, Left: Average absolute errors. Right: Average CPU times

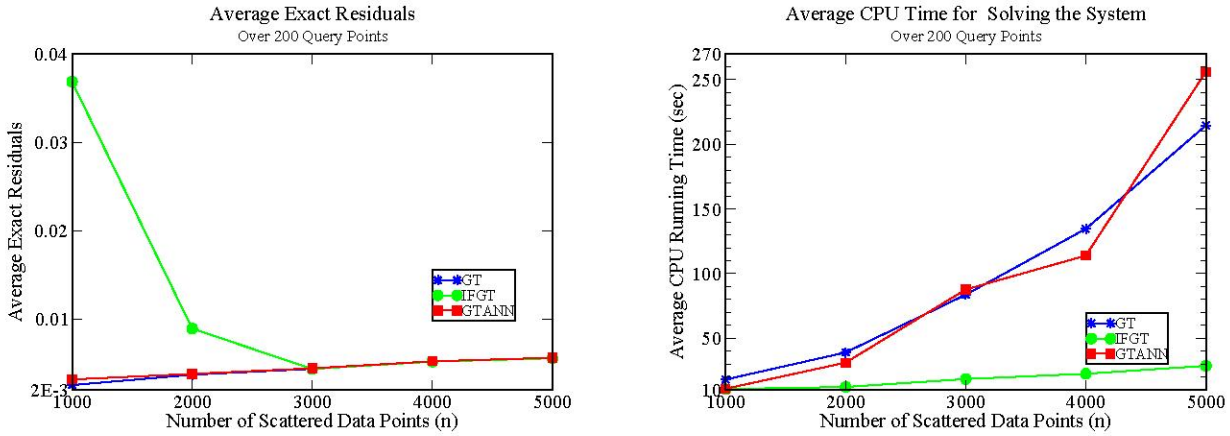


Figure 2. Experiment 2, Left: Average absolute errors. Right: Average CPU times

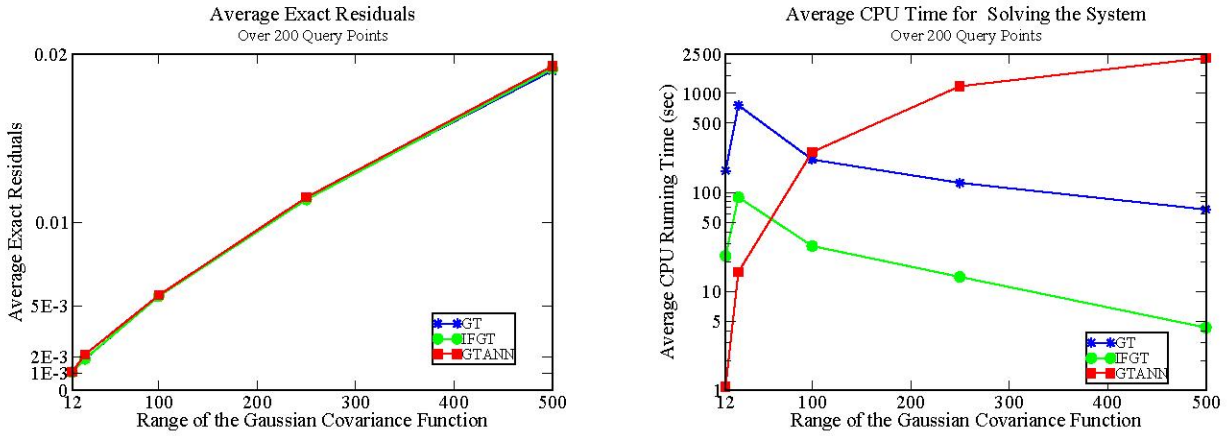


Figure 3. Experiment 3, Left: Average absolute errors. Right: Average CPU times

[21] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[22] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*. McGraw-Hill Companies, 1996.

[23] L. Greengard and J. Strain, "The fast Gauss transform," *SIAM Journal of Scientific and Statistical Computing,*, vol. 12, no. 1, pp. 79–94, 1991.

[24] V. C. Raykar and R. Duraiswami, *Large Scale Kernel Machines*. MIT Press, 2007, ch. The Improved Fast Gauss Transform with applications to machine learning.

[25] T. Feder and D. H. Greene, "Optimal algorithms for approximate clustering," in *Proc. 20th Annual ACM Symposium on Theory of Computing*, 1988, pp. 434–444.

[26] V. C. Raykar, "Scalable machine learning for massive datasets: Fast summation algorithms," Ph.D. dissertation, University of Maryland, College Park, MD, 20742, March 2007.

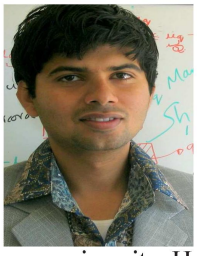
[27] D. M. Mount and S. Arya, "ANN: A library for approximate nearest neighbor searching," http://www.cs.umd.edu/~mount/ANN/, May 2005.

[28] N. Remy, "GsTL: The Geostatistical Template Library in C++," Master's thesis, Department of Petroleum Engineering of Stanford University, March 2001.

[29] N. Memarsadeghi, "Efficient algorithms for clustering and interpolation of large spatial data sets," Ph.D. dissertation, University of Maryland, College Park, MD, 20742, April 2007.

BIOGRAPHY

Nargess Memarsadeghi is a computer engineer at the Information Systems Division (ISD) of the Applied Engineering and Technology Directorate (AETD) at NASA Goddard Space Flight Center (GSFC) since July 2001. She received her Ph.D. from the University of Maryland at College Park in Computer Science in May 2007. She is interested in design and development of efficient algorithms for large data sets with applications in image processing, remote sensing, and optics via computational geometry and scientific computation techniques.



Vikas C. Raykar received the B.E. degree in electronics and communication engineering from the National Institute of Technology, Trichy, India, in 2001, the M.S. degree in electrical engineering from the University of Maryland, College Park, in 2003, and the Ph.D. degree in computer science in 2007 from the same university. He currently works as a scientist in Siemens Medical Solutions, USA. His current research interests include developing scalable algorithms for machine learning.

Ramani Duraiswami is an associate professor in the department of computer science and UMIACS at the University of Maryland, College Park. He directs the Perceptual Interfaces and Reality Lab., and has broad research interests in scientific computing, computational acoustics and audio, computer vision and machine learning.

David Mount is a professor in the Department of Computer Science at the University of Maryland with a joint appointment in the University's Institute for Advanced Computer Studies (UMIACS). He received his Ph.D. from Purdue University in Computer Science in 1983, and since then has been at the University of Maryland. He has written over 100 research publications on algorithms for geometric problems, particularly problems with applications in image processing, pattern recognition, information retrieval, and computer graphics. He currently serves as an associate editor for the journal ACM Trans. on Mathematical Software and served on the editorial board of Pattern Recognition from 1999 to 2006. He served as the program committee co-chair for the 19th ACM Symposium on Computational Geometry in 2003 and the Fourth Workshop on Algorithm Engineering and Experiments in 2002.