Using a Fast Multipole Method to Accelerate Spline Evaluations

FANG CHEN AND DAVID SUTER
Monash University, Australia

♦ ♦ ♦

In considering the problem of interpolating scattered data using spline methods, the authors present a general framework for using the multipole method to accelerate spline evaluations. Their article also illustrates the efficiency and accuracy of the fast multipole algorithm for the 2D vector spline.

♦

Scattered-data *interpolation* or *smoothing*—the problem of approximating a function from some scattered measurements—affects many science and engineering fields, including meteorology, geophysics, and biomedical science, as well as computer graphics and image processing. The interpolation problem involves defining a function that goes through (in the interpolating sense) or close to (in the smoothing sense) a given set of data measurements $\{y_i\}_1^N$. Each of the y_i corresponds to a data location x_i. The data y_i can be measurements with any dimension, that is, the measurement can be multiple-valued or vector at each point.

Spline-interpolation techniques provide a very useful tool for solving the scattered-data interpolation problem. The term *spline* can be employed for a variety of interpolation and approximation methods. Here we restrict ourselves to PDE (partial differential equation) splines[1] because of their abilities to model physical properties (such as deformation). Moreover, unlike other spline methods (such as tensor product splines and NURBS (*non*uniform *r*ational *B*-*s*plines), popular in some fields of computer-aided design), PDE splines do not prefer any specific direction. In spline methods, the spline function minimizes a smoothness-constraint functional. This functional is defined in terms of the derivatives of the interpolant and is usually related to a PDE (or an ODE—ordinary differential equation—for a univariate function).

The thin-plate spline, a well-known example of a PDE spline, is a flexible thin plate with minimum bending energy in terms of the second-order derivatives and is related to a biharmonic kernel. The thin-plate spline is a scalar function. Recently, interest also has developed in the study of vector splines, which have been used for the restitution of wind velocity fields in meteorology[2] and for other vector-field reconstructions. We have developed fast multipole method-based schemes to rapidly evaluate both scalar and vector splines.

In this article, we describe and illustrate a general fast method for efficiently evaluating splines in high dimensions. The method is based on a FMM for potential calculation.[3] The algorithm involves a tree-data structure and two hierarchical approximations: an upward multipole-expansion approximation and a downward local Taylor-series approximation. The CPU time of the vector kernel's direct calculation at N points with N data points increases at $O(N^2)$. Compared to the CPU time of direct calculation, which increases at a quadratic rate with the number of points, the fast algorithm achieves a higher evaluation speed at a linear rate.

1070-9924/98/$10.00 © 1998 IEEE

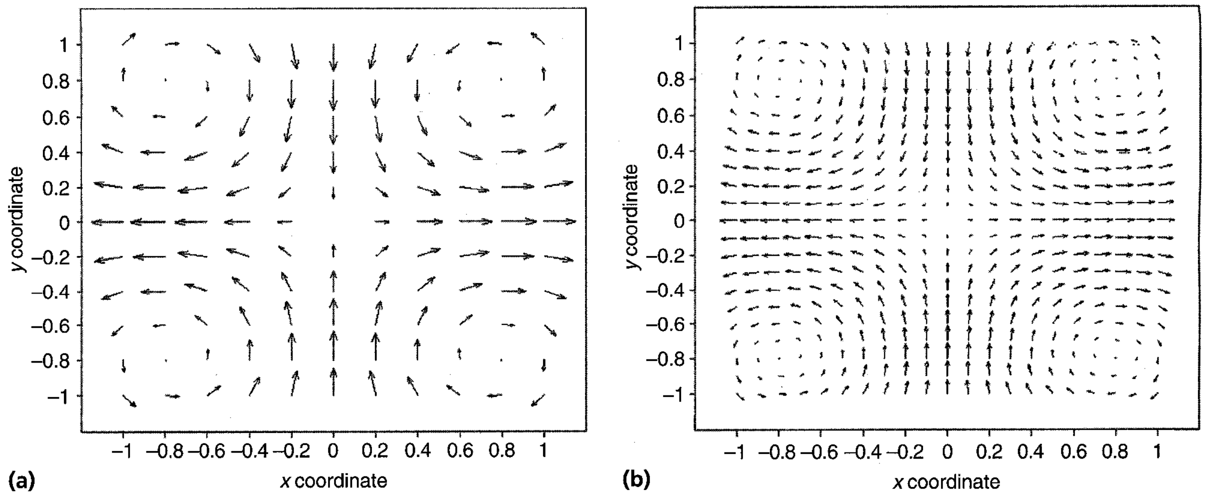


Figure 1. An example of the vector spline interpolant shows (a) the sample data and (b) the reconstruction based on the vector spline defined by Equations 1 and 2 with $\alpha = 1, \beta = 0$.

The spline solution

The splines to which we and others have applied FMM for evaluation are ones whose kernels are members of or closely related to the polyharmonic family.[4–8] Although some of the methods[4] appear to depend on specific kernel properties (for example, a biharmonic kernel), the technique only requires that the shifted summation of the kernel function has an expression that can be truncated (with error rapidly decreasing with distance) at a certain point. We do not know precisely what class of kernels this technique cannot use, but it cannot be applied to completely general kernels (for example, general oscillatory kernels would be beyond the scope of this technique[9]).

In general, we can express the spline solution as a linear combination of the shifted versions of a kernel function, plus a polynomial term. That is, the spline is of the form

$$f(x) = \sum_{i=1}^{N} c_i \Phi(|x - x_i|) + p_{m-1}(x) \quad , \tag{1}$$

where $\Phi(x)$ is the kernel function determined by the smoothness functional or the associated PDE. The kernel Φ is a radial function $r^2 \log r$ for a thin-plate spline (where $r = |x - x_i|$) and has a matrix form in the case of vector splines. For example, a 2D second-order vector-spline kernel has the form of

$$\Phi(r) = \begin{pmatrix} \frac{1}{\alpha}\partial_{xx} + \frac{1}{\beta}\partial_{yy} & \left(\frac{1}{\alpha} - \frac{1}{\beta}\right)\partial_{xy} \\ \left(\frac{1}{\alpha} - \frac{1}{\beta}\right)\partial_{xy} & \frac{1}{\alpha}\partial_{yy} + \frac{1}{\beta}\partial_{xx} \end{pmatrix} K(r) \quad , \tag{2}$$

where α and β are model parameters, and $K(r) = r^4 \log r$ is a triharmonic scalar kernel that satisfies $\Delta^3 K = \delta$. The other term, $p_{m-1}(x) \in \Pi_{m-1}$, in Equation 1 is a polynomial with at most degree $m - 1$, where m is the lowest order of the derivative appearing in the smoothness functional. $c_i (i = 1, \ldots, N)$ are coefficients that satisfy the constraints $\sum_{i=1}^{N} c_i q_{m-1}(x_i) = 0$ for any $q_{m-1}(x) \in \Pi_{m-1}$. In this spline, α and β control the amount of variation in the reconstructed field's divergence and curl. (We discussed the details for this 2D vector spline and a 3D vector analogy in earlier works.[5,6])

To determine the spline coefficients c_i, we must solve the following linear system:

$$\begin{cases} f(x_i) = y_i \\ \sum_{i=1}^{N} c_i q_{m-1}(x_i) = 0 \end{cases} . \tag{3}$$

Hence, we obtain an explicit spline interpolant. Figure 1 gives an example of the vector spline interpolant.

In this article, we are only interested in the evaluation process with the given c_i—in evaluating, for example, the spline interpolant (Equation 1) at x for given c_i and x_i. (We are not concerned here with the polynomial p_{m-1}, because the key to fast evaluation of f is the fast evaluation of the shifted kernels' summation.) Equation 1 shows that every data point x_i, ($i = 1, \ldots, N$) contributes to the evaluation at the point x. Hence, evaluating the spline at a given point requires $O(N)$ operations. The complete spline evaluation at M points requires $O(MN)$ operations. This causes the evaluation process to be very computationally expensive when both the data points and the evaluation points are large in number.

The high computational cost is a major disadvantage of the spline methods and precludes their broad application to problems with large data sets. Obviously, for a problem involving large data in multidimensions, such as data derived from modern imaging methods—for ex-

Related Work

Leslie Greengard and Vladimir Rokhlin first proposed the fast multipole method for 2D potential calculation.[1] This method has been popular for solving the various problems of calculating *N*-body interactions. Various attempts have been made to improve the FMM algorithm. For instance, Christopher R. Anderson[2] presented an alternative method of constructing the coefficients of the multipole expansions by using a Poisson formula. Fast Fourier transform (FFT) techniques have been applied to accelerate the local Taylor approximation stage, because this stage dominates the overall runtime of the FMM.[3] (An exhaustive survey of this extensive literature is beyond of the scope of this article.)

In fact, FMM application is not restricted to *N*-body simulations: it provides a general approach for rapid calculation of a finite summation. John P. Boyd[4] has discovered the similarity between the multipole method and the FFT for some problems. More closely related to the work summarized here, Rick Beatson and Garry Newsam[5] and David Suter[6] have used FMM to fast-evaluate a 2D thin-plate spline kernel, as well as other radial basis functions. Suter's method, close to Anderson's for the harmonic potential kernel,[2] employed a Poisson formula for a biharmonic thin-plate spline kernel to calculate the multipole expansion coefficients by an integral along a closed contour surrounding the centers of the kernel. Our work further extends the use of FMM applications by using them for the rapid evaluation of the DIV-CURL vector splines in 2D and 3D.[8,9]

The FMM has also been used to solve integral equations, such as Dirichlet and Neumann boundary problems,[10] and to construct conformal mapping[11] in partial differential equation theory. Other related works have contributed to speeding up the FMM by incorporating various techniques into the multipole method—for instance, preconditioned multipole iteration,[12] grid-based representation,[13] and parallel implementation.[14]

References

1. L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations," *J. Computational Physics*, Vol. 73, 1987, pp. 325–348.
2. C.R. Anderson, "An Implementation of the Fast Multipole Method without Multipoles," *SIAM J. Scientific Statistical Computing*, Vol. 13, No. 4, July 1992, pp. 923–947.
3. W.D. Elliott and J.A. Board, "Fast Fourier Transform Accelerated Fast Multipole Algorithm," *SIAM J. Scientific Statistical Computing*, Vol. 17, No. 2, Mar. 1996, pp. 398–415.
4. J.P. Boyd, "Multipole Expansions and Pseudospectral Cardinal Functions: A New Generalization of the Fast Fourier Transform," *J. Computational Physics*, Vol. 103, 1992, pp. 184–186.
5. R.K. Beatson and G.N. Newsam, "Fast Evaluation of Radial Basis Functions: 1," *Computational Mathematics with Applications*, Vol. 24, No. 12, Dec. 1992, pp. 7–20.
6. D. Suter, "Fast Evaluation of Splines Using Poisson Formula," *J. Applied Science and Computations*, Vol. 1, No. 1, June 1994, pp. 70–87.
7. R.K. Beatson and W. A. Light, "Fast Evaluation of Radial Basis Functions: Methods for 2-Dimensional Polyharmonic Splines," *IMA J. Numerical Analysis*, Vol. 17, 1997, pp. 343–372.
8. F. Chen and D. Suter, "Fast Evaluation of Vector Splines in Two Dimensions," *Proc. 15th IMACS World Congress on Scientific Computation, Modeling, and Applied Mathematics*, Vol. 1, Aug. 1997, pp. 469–474.
9. F. Chen and D. Suter, "Fast Evaluation of Vector Splines in Three Dimensions," to be published in Computing.
10. V. Rokhlin, "Rapid Solution of Integral Equations of Classical Potential Theory," J. Computational Physics, Vol. 60, 1985, pp. 187–207.
11. S.T. O'Donnell and V. Rokhlin, "A Fast Algorithm for the Numerical Evaluation of Conformal Mapping," SIAM J. Scientific Statistical Computing, Vol. 10, No. 3, 1989, pp. 475–487.
12. K. Nabors et al., "Preconditioned, Adaptive, Multipole-Accelerated Iterative Methods for Three-Dimensional First-Kind Integral Equations of Potential Theory," SIAM J. Scientific Statistical Computing, Vol. 15, No. 3, 1994, pp. 713–735.
13. C.L. Berman, "Grid-Multipole Calculations," SIAM J. Scientific and Statistical Computing, Vol. 16, No. 5, 1995, pp. 1082–1091.
14. F. Zhao and S.L. Johnson, "The Parallel Multipole Method on the Connection Machine," SIAM J. Scientific Statistical Computing, Vol. 12, No. 6, 1991, pp. 1420–1437.

ample, satellite data or magnetic resonance imaging data—direct calculation of a spline is inefficient or prohibitive.

A general description of FMM

The FMM provides a general approach for rapid calculation of a finite summation of a suitable kernel function K with individual sources $\{x_n\}^N{}_1$ and weights $\{a_n\}^N{}_1$:

$$\sum_{n=1}^{N} a_n K(x - x_n) \, .$$

A suitable kernel function means a kernel function whose pairwise interaction can be expressed in a multipole-like series expansion as long as the two points are separated by an appropriate distance, which is referred to as a *far field* in FMM literature. The essential idea of the fast multipole algorithm is to combine a cluster of far-field data *sources*, each of them corresponding to a multipole expansion, into a single source with a combined multipole expansion. (The analogy hinted at here is with electric-field sources; one of the first applications of multipole methods was in the calculation of electric field and potential.[3]) For more information, see the "Related Work" sidebar.

We can construct the FMM hierarchical tree-like data structure by repeatedly subdividing the original domain into four equal-size subdomains, thus implementing the algorithm based on this data tree. For every evaluation point at the bottom of the data tree, the contributions coming from all data sources are separated into two parts: one from the near field and the other from the far field. The FMM computes the near-field contributions using a direct method, and divides the far-field sources into a number of clusters, associating each with a combined multipole expansion. These multipole expansions propagate upward through the tree—effectively creating large clusters—and then shift, under a convergence condition, into the local Taylor-series expansions. Finally, the local Taylor expansions propagate downward to the bottom for approximating all far-field contributions. The process takes the following four stages.

Stage 1: data-tree construction.

Assume that the whole data (both data sources and evaluation points) are within a regular n-dimensional region, named a parent *panel* or root (level 0). (Here we use *panel* as general description: it will refer to a square in 2D space, or a cube in 3D space.) Repeatedly subdividing each parent panel into 2^n children panels forms an n-dimensional tree-data structure. Suppose this procedure terminates at level l—then $(2^n)^l$ panels are obtained at the finest level. For each panel P in this tree, we define a few notations used in the FMM:

- *near field of* P: a set of panels that are at the same level as P and share a common corner, edge, or face with P.
- *far field of* P: a set of panels that are at the same level as P and are not in the near field of P.
- *well-separated neighbors of* P: a set of panels that are in the far field of P and whose parents are in the near field of the parent of P.

In general, a panel in the n-dimensional tree has at most $3^n - 1$ near-field elements and at most $6^n - 3^n$ well-separated neighbors. In particular, there are at most eight near-field squares and 27 well-separated neighbors in 2D (see Figure 2a).

Stage 2: upward approximation.

Beginning from the finest level, for each panel that contains data sources, a combined multipole expansion at the center of this panel is constructed to approximate the whole contributions from those sources (see Figure 2b). Then, all of the children multipole expansions shift to their parent to form the upward multipole approximation (see Figure 2c).

Stage 3: downward approximation.

Beginning from the coarsest level, for each panel that contains the evaluation points, the FMM constructs a local Taylor series at the center of this panel from two sources. One is obtained by translating its well-separated neighbor multipole expansions, which cover contributions from all well-separated neighbors, and the other comes from the shifted parent local Taylor series, which cover contributions from all far-field sources, excluding well-separated neighbors (see Figure 2d).

Stage 4: evaluation.

At the finest level, for each evaluation point within the panel P, we obtain the evaluation result by adding together the truncated local Taylor-series approximation for all far-field interactions and the direct calculation with data sources within P and the near field of P.

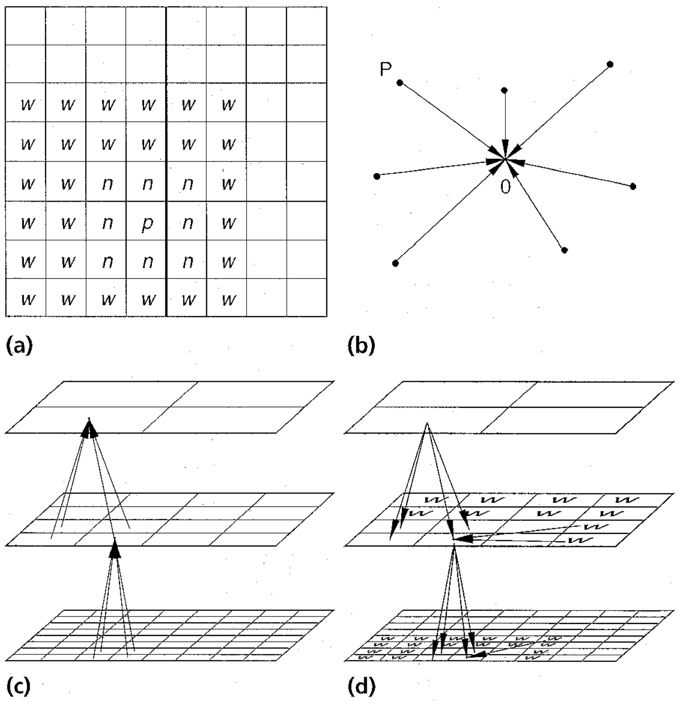
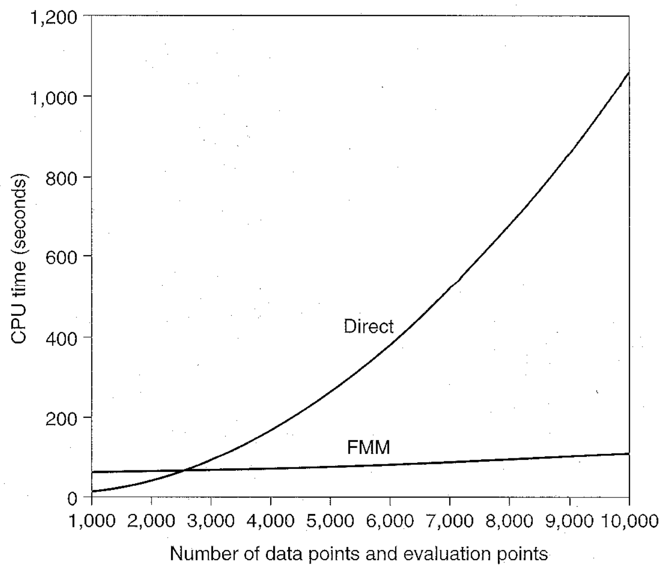


Figure 2. Data-tree construction and hierarchical approximations: (a) the nearest neighbors (denoted by *n*) and the well-separated neighbors (denoted by *w*) of *P*; (b) a cluster of data sources within *P* combined into a single source at the center of *P*; (c) upward multipole expansion approximations build by shifting all existing children's multipole expansions to their parent; (d) downward local Taylor-series expansions approximations build by adding together two types of local Taylor expansions—one is translated from the well-separated field (denoted by an arrow from *w*) and the other is shifted from its parent.



Numerical results

Here we show the numerical results of the FMM for the evaluation of 2D vector splines (Equation 1).[5] Such vector-spline kernels are defined in terms of the second-order derivatives of the related scalar kernel (see Equation 2). Moreover, these vector kernels are related to parameters α and β, which depend on the modeled data. The issue of selecting α and β in practical applications[2] is beyond of the scope of this article. Here we only emphasize that the proposed FMM is independent of the value of α and β, except for the special case where $\alpha = \beta$.

In such a case, the vector kernel degenerates to the thin-plate spline kernel. Therefore, evaluating the vector spline, when $\alpha = \beta$, is equivalent to evaluating two independent thin-plate splines.[7] The corresponding FMM for the thin-plate spline is certainly more efficient than that of the vector case (principally because there are no off-diagonal terms in Equation 2). When $\alpha = \beta$, because the computational cost for the same truncation index p is less than for $\alpha \neq \beta$, we can increase the truncation index and gain more accuracy for the same total computational cost. That is, we can trade time saved for increased accuracy, and in this sense, the efficiency of the $\alpha = \beta$ solution is greater. However, the issue should not be whether the case $\alpha = \beta$ is more efficient for a given physical problem, but whether the case $\alpha = \beta$ is appropriate in terms of the problem's underlying physics.

The numerical results we show are all based on the general case of $\alpha \neq \beta$. We implemented the algorithm in a C program on a Silicon Graphics Indy (R4600 processor) to illustrate the algorithm's efficiency and accuracy.

Timing

Figure 3 shows the CPU time for a five-level FMM and the CPU time for direct calculation. The data points and evaluation points are generated randomly and distributed uniformly

Figure 3. The CPU time for a five-level fast multipole scheme and for direct calculation of a 2D vector spline. The number of truncations for both multipole and Taylor expansions is set to 10. The line with "FMM" represents the CPU time for a five-level multipole algorithm, which increases linearly with the number of points *N*. The line with "Direct" represents the CPU time for direct calculation, which has a order of $O(N^2)$.

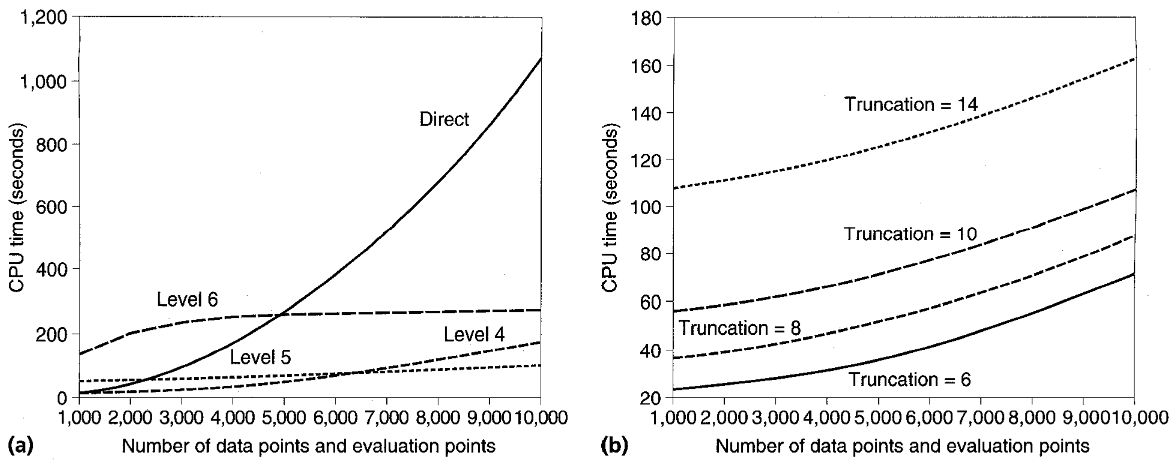


Figure 4. The CPU time of the fast algorithm is influenced by the parameters *l* and *p*; (a) the CPU time changes with different levels, where level *l* means a $2^l \times 2^l$ grid of squares, and truncation index $p = 10$; (b) the CPU time changes with different truncation indexes in a five-level FMM scheme. The runtime increases rapidly when the truncation index increases.

over $[0, 1] \times [0, 1]$. The series truncation is set to 10 for both the multipole and the local Taylor expansions. As shown in Figure 3, the computation time using the FMM has a linear relation with N, while the timing using direct method increases with a rate of $O(N^2)$.

In the FMM, the refinement level l influences the runtime of the fast multipole algorithm. More specifically, increasing the refinement level produces more panels, which results in higher computational costs for the implementation. On the other hand, increasing the number of data points N requires a finer subdivision of the data tree to avoid too much near-field direct calculations. Therefore, the choice of the refinement level mainly depends on the number of data points. Figure 4a depicts the CPU time of the FMM changing with the different levels (fixed $p = 10$), where level l means a $2^l \times 2^l$ grid of panels. As Figure 4a shows, when the number of points is larger than 6,500, a level-5 scheme performs much more efficiently than a level-4 scheme. In general, for a nearly uniform data system, the refinement level is $l \approx \log_{2^n} N$, so that most of the panels at this level contain at least one data source (where n is the dimensions of the space in which the problem will be solved).

Another parameter involved in FMM is the truncation index p of the series expansions. This truncation index also influences the FMM efficiency and can be chosen according to the required accuracy. Figure 4b shows that the runtime of the fast algorithm varies with the truncation index (fixed $l = 5$). Obviously, the calculation will take longer if the series expansion is calculated using more terms. However, the trade-off between the runtime and the accuracy should be considered when choosing the truncation index, because the series expansion with fewer terms will result in greater error (see Table 1). In practical FMM implementations, to achieve a given precision ϵ, the truncation index can be chosen as $p \geq |\log_2(\epsilon)|$.[6]

When implementing FMM, we spend much time constructing the local Taylor-series expansion due to the relatively large number of well-separated neighbors (particularly, there are 27 well-separated neighbors for 2D FMM and 189

Table 1: The largest absolute error and relative error of FMM with respect to different truncation indexes for a five-level scheme, where the number of data points and evaluation points are both equal to 4,000. The numerical results show that the maximum errors decrease quickly as the truncation index increases.

Truncation *p*	$\varepsilon^{f_1}_{max}$	$\varepsilon^{f_1}_{re-max}$	$\varepsilon^{f_2}_{max}$	$\varepsilon^{f_2}_{re-max}$
6	0.104688	0.000687582	0.135803	0.00330963
8	0.00866754	0.000266301	0.0116244	0.000253477
10	0.000930466	3.87372e–05	0.00125236	4.21111e–05
14	1.7925e–05	2.72763e–07	1.51653e–05	3.27185e–07

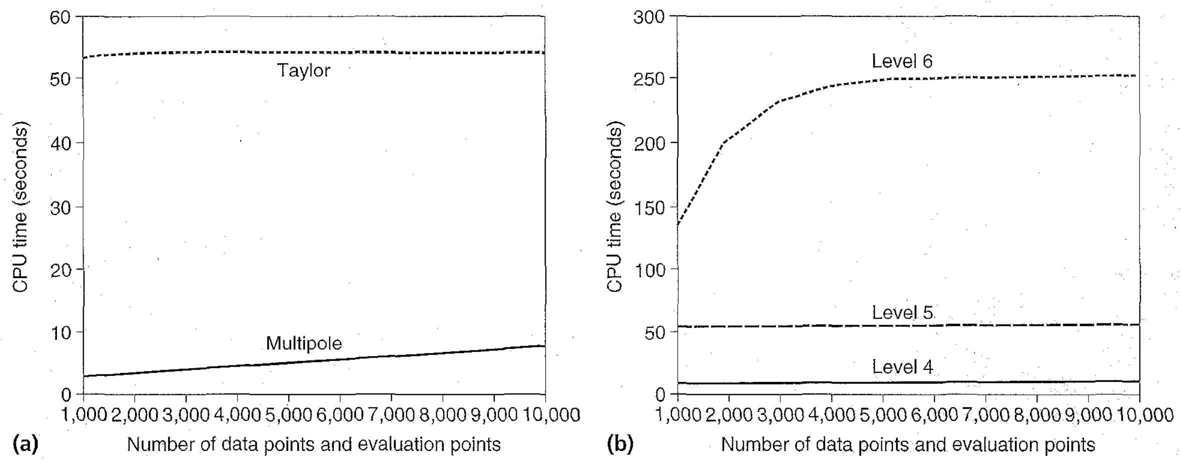


Figure 5. The CPU time spent constructing local Taylor series: (a) CPU time spent on the Taylor approximation and on the multipole approximation for a five-level scheme shows that the Taylor approximation process is time-dominated; (b) CPU time on building Taylor series increases sharply as the refinement level becomes large, but with a fixed level the time spent on building the Taylor series depends less on the number of evaluation points.

well-separated neighbors for 3D FMM). Figure 5a shows that the local Taylor-series approximation dominates the runtime of the FMM algorithm. In addition, the CPU time spent on the local Taylor-series approximation increases sharply as the refinement levels l increase (see Figure 5b); however, it then depends less on the number of points N. The last property implies that when the number of points increases, we can employ a higher-level scheme to reduce the amount of the direct calculations between the nearest neighbours.

Error estimation

The error of an FMM algorithm strongly depends on the truncation index of the series expansions. The error bounds decrease as the truncation index increases. Table 1 shows the maximum absolute error and maximum relative error with a different number of truncations for each component f_1 and f_2.

We have shown how to use the FMM to accelerate the evaluations of splines related to the vector splines. Our approach has proven to be very efficient and generally applicable as no regular grid is necessary, and it is able to encompass whole families of PDE splines (including scalar and vector). The fast spline technique should greatly extend the spline application domains and will provide a more powerful and effective tool for scientists and engineers.

We have used the PDE splines for several tasks, including reconstructing cardiac motion from measurements taken from MRI image sequences, historical-film restoration ("unwarping" the distortion present in very old film material), and medical-image restoration. Other work includes further accelerating spline calculation by investigating such methods as wavelet decomposition, multiprocessor implementation, and domain-decomposition methods. We are also investigating spline applications in new areas—applications that become feasible when the techniques we have discussed here reduce the computational time. ◆

References

1. G. Wahba, *Spline Models for Observational Data*, Soc. Industrial and Applied Mathematics, Philadelphia, Pa., 1990.
2. L. Amodei and M.N. Benbourhim, "A Vector Spline Approximation with Application to Meteorology," *Curves and Surfaces*, P.J. Laurent, A. Le Mehaute, and L.L. Schumaker, eds., Academic Press, Boston, 1991, pp. 5–10.
3. L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations," *J. Computational Physics*, Vol. 73, 1987, pp. 325–348.
4. D. Suter, "Fast Evaluation of Splines Using Poisson Formula," *J. Applied Science and Computations*, Vol.

1, No. 1, 1994, pp. 70–87.
5. F. Chen and D. Suter, "Fast Evaluation of Vector Splines in Two Dimensions," *Proc. 15th IMACS World Congress Scientific Computation, Modeling, and Applied Mathematics*, Vol. 1, Aug. 1997, pp. 469–474.
6. F. Chen and D. Suter, "Fast Evaluation of Vector Splines in Three Dimensions," to be published in *Computing*.
7. R.K. Beatson and G.N. Newsam, "Fast Evaluation of Radial Basis Functions: 1," *Computational Mathematics with Applications*, Vol. 24, No. 12, Dec. 1992, pp. 7–20.
8. R.K. Beatson and W.A. Light, "Fast Evaluation of Radial Basis Functions: Methods for 2-Dimensional Polyharmonic Splines," *IMA J. Numerical Analysis*, Vol. 17, 1997, pp. 343–372.
9. A. Brandt, "Multilevel Computations of Integral Transforms and Particle Interactions with Oscillatory Kernels," *Computer Physics Communications*, Vol. 65, 1991, pp. 24–38.

Fang Chen is a research officer for the Centre for Magnetic Resonance at the University of Queensland. Her research interests include spline approximation, fast computation, cardiac motion analysis, and biomedical image computing. The research presented in the article was completed during her PhD study at the Dept. of Electrical and Computer Systems Engineering at Monash University. Contact her at the Centre for Magnetic Resonance, Univ. of Queensland, St. Lucia Campus, Australia, Qld 4072; fiona@cmr.uq.edu.au.

David Suter is a senior lecturer in the Dept. of Electrical and Computer Systems Engineering, Monash University. His research interests include the fast approximation of spline functions, the applications of spline functions, computer vision, historial film restoration, and biomedical imaging. He obtained a BSc in applied mathematics and physics from the Flinder University of South Australia and his PhD in computer vision from La Trobe University. Contact him at the Dept. of Electrical and Computer Systems Eng., Monash Univ., Clayton Campus, Australia, Vic. 3168; d.suter@eng.monash.edu.au.