Biharmonic Many Body Calculations for Fast Evaluation of Radial Basis Function Interpolants in Cluster Environments

George Roussos and B.J.C. Baxter

Imperial College, 180 Queen's Gate, London SW7 2BZ, U.K.

Abstract. This paper discusses the scalability properties of a novel algorithm for the rapid evaluation of radial basis function interpolants. The algorithm is associated with the problem of force calculation in many-body calculations. Contrary to previously developed fast summation schemes including treecodes and fast multipole methods, this algorithm has simple communication patterns which are achieved by exploiting the localisation and smoothness properties of radial basis functions. Thus, the algorithm is scalable even in low bandwidth environments like clusters of workstations and even for relatively small problem sizes.

1 Introduction

Recent results show that the radial basis function method produces high quality solutions to the multivariate scattered data interpolation problem, especially in higher dimensions. On the other hand, the method is associated with higher computational cost when compared against alternative methods, such as finite elements or multivariate spline interpolation. Indeed, solving directly the interpolation equations for N data points requires $\mathcal{O}(N^3)$ floating-point operations, while direct evaluation of the resulting interpolant at M locations requires $\mathcal{O}(MN)$ floating-point operations.

Several attempts aiming at the reduction of the computational complexity of both the calculation and the evaluation tasks have produced a number of novel algorithms. For example, rapid evaluation of radial basis function interpolants has been achieved using a variety of methods including subtabulation on a regular grid, treecodes, moment based methods and the fast multipole method. In [9] a method based on the Fast Gauss Transform and suitable quadrature rules has been developed. This method exploits fundamental smoothness and localisation properties of the radial basis function method to achieve high performance.

In this paper, we explore the performance of the method developed in [9] in the context of a cluster of workstations. Indeed, we discover that due to the regularity of its computational structure the method performs very well even in low bandwidth environments and small problem sizes.

2 Fast Evaluation of Radial Basis Function Interpolants

Radial Basis Function interpolation has the form

$$s(y) = \sum_{i=1}^{N} \lambda_i \varphi(\|y - x_i\|), \tag{1}$$

where λ_i are real coefficients, x_i points in $\mathbb{R}^d$ called *centres*, $\|\cdot\|$ the Euclidean norm and φ the *basis function.* The function $\varphi : \mathbb{R}^+ \to \mathbb{R}$ is unary and radially symmetric with respect to the norm, in the sense that it has the symmetries of the unit ball in $\mathbb{R}^d$. The coefficients λ_i are chosen so that the interpolation conditions are satisfied, that is

$$s(x_i) = f(x_i) = f_i, \quad i = 1, 2, \ldots, N, \tag{2}$$

where f_i is the value of the interpolated function f at location $x_i \in \mathbb{R}^d$.

Examples of useful choices of φ include the *Gauss kernel*, the *Euclidean distance*, the biharmonic *Hardy Multiquadric*

$$\varphi(r) = \sqrt{r^2 + c^2}, \tag{3}$$

and the *Inverse Multiquadric* which is exactly the inverse of (3).

Solving the interpolation problem (2) is equivalent to solving the system of linear equations $A\lambda = f$, where A is the *interpolation matrix* $A_{ij} = \varphi(\|x_i - x_j\|)$, λ is the vector of coefficients $(\lambda_1\ \lambda_2\ \ldots\ \lambda_N)^T$ and f is the vector of function values $(f_1\ f_2\ \ldots\ f_N)^T$. It is clear that the solution of the interpolation equations directly requires $\mathcal{O}(N^3)$ floating point operations, while the form of the interpolant (1) implies that evaluating s at M points $y_1, y_2, \ldots, y_M$, directly requires $\mathcal{O}(MN)$ operations.

Beatson and Newsam [2] first noted the intimate relation between the many body problem and the evaluation of the radial basis function interpolant. Furthermore, they exploited this observation to develop the mathematical framework required for the introduction of fast evaluation methods for a particular radial basis function in two dimensions. Indeed, in [2] the Laurent and Taylor expansions required by the Fast Multipole Method (FMM) in two dimensions were constructed. These results were used by Powell [8] to introduce a fast algorithm for the evaluation of radial basis function interpolants which is a higher order treecode. This method uses a decomposition of the set of interpolation centers similar to Appel [1] which results in observed computational complexity of $\mathcal{O}(N \log N)$ for fairly uniform distributions of centers. More recently, Beatson has implemented a full FMM based on the results of [2] with reported performance similar to that discussed in [4]. A variant of this method, whereby the coefficients of the multipole expansion are not calculated directly but approximated is discussed by Suter [10].

In particular, the relation between the Hardy Multiquadric and many body computations has received a physical justification. Indeed, Hardy [6] relates the solution of an interpolation problem to simulation of the Earth's geomagnetic field by a biharmonic potential. The biharmonic approach has the advantage over the use of a harmonic potential that the Earth is considered as a solid, rather than a hollow body.

Finally, it is worth pointing out that the force calculation step of the many body computation with the Plummer potential is exactly the evaluation of an Inverse Multiquadric interpolant, where $y_i = x_i$ are the locations of the point masses λ_i.

3 Algorithm Description

One of the features that makes radial basis function interpolation a useful technique is the fact that a unique interpolant is guaranteed under weak conditions on the location of the centres. Micchelli [7] specified these conditions by relating the interpolation matrix of several of the radial basis functions with almost positive (or negative) definite functions. A by-product of this proof is a way to represent certain radial basis functions as an integral of a Gaussian by a suitable measure. For example, we can prove that the Hardy Multiquadric can be rewritten in terms of Gaussian kernel sums in the following way

$$s(y) = c\sum_{i=1}^{N}\lambda_i + \frac{1}{\sqrt{2\pi}}\int_0^\infty \frac{e^{-sc^2}}{\sqrt{s}} \cdot \frac{c\sum_{i=1}^{N}\lambda_i - \sum_{i=1}^{N}\lambda_i e^{-s\|y-x_i\|^2}}{s}\,\mathrm{d}s, \qquad (4)$$

for centres $x_i, i = 1, 2, \ldots, N$ and evaluation point y in $\mathbb{R}^d$.

Formula (4) implies that two ingredients are required for the construction of a fast evaluation algorithm: the first ingredient is a rapid summation scheme for Gaussian kernels and the second a suitable quadrature rule for the approximation of the integral. The first ingredient is provided by the Fast Gauss Transform of Greengard and Strain [5] which will be discussed in following paragraphs. The second ingredient in this case is provided by Gauss-Laguerre quadrature. Of course, each radial basis function has a somewhat different integral representation and thus requires a suitable choice of quadrature rule (in [9] we have identified such rules for the most commonly used functions). It is also worth noting that the resulting algorithm works in any d-dimensional setting. This is in contrast to the FMM where significant differences exist between the two and the three dimensional cases. For briefness of exposition, in this paper we will consider only the three-dimensional case and will not provide the error estimates for the employed approximations.

By shifting the origin and re-scaling, we may assume that all the interpolation centers and all the evaluation points lie within the unit cube $B_0 = [0,1] \times [0,1] \times [0,1]$. This is a a convenient normalization and does not restrict the generality of the method.

The first element of the Fast Gauss Transform is the observation that we may express a Gaussian in $\mathbb{R}^3$ as the Hermite expansion

$$e^{-s\|y-x\|^2} = \sum_{\beta\geq 0} \frac{1}{\beta!} \left(\sqrt{s}(x-C)\right)^{\beta} h_{\beta}\left(\sqrt{s}(y-C)\right). \tag{5}$$

with h_β a Hermite function. Also, we may assume that the point y is contained in the box $B = \{y \in [0,1]^3 : \|y-C\|_\infty < r/\sqrt{2s}\}$ of side length $r\sqrt{2/s}$ for some $r < 1$ centered at C.

For centres $x_1, x_2, \ldots, , x_N \in \mathbb{R}^3$ inside box B we can precompute the moments

$$A_\beta = \frac{1}{\beta!} \sum_{i=1}^{N} \lambda_i \left(\sqrt{s}(x_i - C)\right)^{\beta}, \tag{6}$$

which we can then use to evaluate the Gaussian sum at a point y by

$$\sum_{i=1}^{N} \lambda_i \exp\left(-s\|y-x_i\|^2\right) = \sum_{\beta\geq 0} A_\beta h_\beta \left(\sqrt{s}(y-C)\right). \tag{7}$$

Thus, it is possible to approximate the Gaussian (7) in terms of the moments (6). The second element of the FGT is the decomposition of the computational space B_0 into subboxes B of side length $r\sqrt{2/s}$ parallel to the axes, for some fixed parameter r. Each centre is assigned to the subbox B that contains it and contributes only to the p^3 moments of subbox B. At the end of the precomputation step, the p^3 moments for each of the subboxes B have been computed. The precomputation requires $\mathcal{O}(p^3 N)$ operations.

For the estimation of the FGT at a particular evaluation point y contained in subbox D, we need to consider the influence of only some of the nearest neighbour boxes of D. Indeed, due to the exponential decay of the Gauss kernel, its effect on subboxes away from its centre may be insignificant within certain accuracy. For example, taking into account only the $(2l+1)^3$ nearest neighbours to D, introduces error bounded by $Qe^{-2r^2l^2}$. Hence, for $r = 1/2$ and $l = 6$ relative accuracy of 10^{-7} is obtained. We will call the set of $(2l+1)^3$ nearest neighbours the *interaction list* of box D. Thus, in order to estimate the Gaussian sum on the left side of (7) at point y, we have to accumulate the p^3 moments for each of the boxes B in the interaction list of D. Evaluation at a single point requires $\mathcal{O}((2l+1)^3 p^3)$ operations. Overall, the computational complexity of the FGT is $\mathcal{O}(p^3 N + p^3(2l+1)^3 M)$.

We now turn our attention to the second ingredient of our method, that is the calculation of the integral form (4). We can approximate s using a q-term generalised Gauss-Laguerre quadrature rule

$$s(y) = c\sum_{i=1}^{N} + \frac{1}{\sqrt{2\pi}} \sum_{k=1}^{q} w_k f(t_k), \tag{8}$$

where $f(t) = (c\sum_{i=1}^{N} \lambda_i + \sum_{i=1}^{N} \lambda_i e^{-t\|y-x_i\|^2})/t$. Thus, rather than evaluating directly the sum of Inverse Multiquadrics at overall cost of $\mathcal{O}(MN)$ operations,

```
1: choose $q$, $p$ and $r$to guarantee the required precision
2: compute the weights $w_k$ and nodes $t_k$ of the quadrature
3: for each quadrature node $t_k$ do
4:   subdivide $B_0$ into boxes of side at most $\sqrt{2/t_k}$
5: end for
{start first stage: precompute moments}
6: for each centre $x_j$ do
7:   for each quadrature node $t_k$ do
8:     find the box $C$ which contains $x_j$
9:     for $\beta < p$ do
10:      compute the contribution of $x_j$ to the moments $A_\beta$
         of box $C$ using (6) on page 4
11:    end for
12:  end for
13: end for
{start second stage: evaluate moments}
14: for each evaluation point $y_i$ do
15:   for each quadrature node $t_k$ do
16:     find the box $B$ that contains $y_j$
17:     for each of the $(2l+1)^3$ nearest neighbours of $B$ do
18:       accumulate the series (7) on page 4 truncated after $p^3$
          terms to obtain an approximation to the Gaussian with parameter $t_k$
19:     end for
20:     accumulate the contribution of the $k$-th point of the quadrature rule (8)
21:   end for
22: end for
```

Algorithm 1: Fast Summation of Hardy Multiquadrics.

we may evaluate q sums of Gaussians (one for each quadrature node t_k) via the Fast Gauss Transform in $\mathcal{O}\big(q(N+M)\big)$ operations. Recall that the decrease in the computational complexity of the latter task is due to the decoupling of the precomputation of the moments of the points x_i and the estimation of the interpolant at points y_j through the already computed moments.

The quadrature nodes t_k are the zeros of the generalised Laguerre polynomial $L_q^{(-1/2)}(t)$ and the weights may be computed by a well-known formula. Overall, the fast evaluation of Inverse Multiquadric interpolants may be performed by Algorithm 1. Overall, the fast evaluation algorithm requires $\mathcal{O}(qp^3N + qp^3(2l+1)^3M)$ operations.

The Gaussian quadrature nodes and the corresponding weights may be computed using one of a number of standard methods, for example using Gautschi's ORTHOPOL package [3]. From these, the weights and nodes for quadrature when the Multiquadric or Inverse Multiquadric constant c is not the unit are calculated by $t_k^* = t_k/c^2$ and $w_k^* = w_k/c$.

4 Parallelism

In a clusted based environment Algorithm 1 has to compete against treecodes and Fast Multipole Methods. Due to the regular communication/computation patterns exhibited by Algorithm 1, we believe that it is significantly more scalable even in relatively small problem sizes. Indeed, in this section we discuss the observed performance of the algorithm in practise and discuss the implications of this result.

In treecodes parallelism may be exploited in the tree building, moment accumulation and moment evaluation stages. Cell-level synchronisation is required in the tree building phase, when different processors try to simultaneously modify the same part of the tree. For moment precomputation, a processor calculating the moments of a specific cell needs to wait until the moments for all its children have been computed. This requires the use of cell level mutexes to avoid dependency conflicts. On the other hand, the moment evaluation stage requires only communication between processors: the evaluation of the moments at a certain point requires information of nearby centre locations and coefficients, along with moment information for well separated cells. This information may be replicated and there is no need for write backs.

Orthogonal Recursive Bisection (ORB) [11] is currently the most successful algorithm in achieving good data locality while preserving load balance. The aim of the method is to provide data locality by explicitly partitioning the computational space and assigning the parts to the available workstations. The algorithm subdivides recursively the computational domain in two parts with equal computational cost. In this context, the computational cost of a particular region is defined to be the total number of interactions between each point in this region either with a centre or a cell. The assignment of domains to processors is done using the following rule: Initially all processors are associated with the entire domain. At each ORB step, the processors are split in two groups and assigned to one of the two subspaces. This process builds an ORB tree[1] which is separate from the cell tree used in the sequential algorithm.

On the other hand, the algorithm discussed in this paper provides for a clear approach to achieving both load balancing and data locality. Indeed, the algorithm has two distinct stages: first the precomputation of the moments and second the evaluation of the quadrature at a point. It is sufficient to split the set of centres (for the first stage) and the set of evaluation points (second stage) into subgroups of approximately equal size with arbitrarily selected members. Each group may be stored locally to avoid communication overheads. At the end of the first step a broadcast of each nodes moments is required and the accumulation those computed at the other nodes. This is a relatively small amount of data, independent of the number of centres and evaluation points.

An implementation of the above approach to exploit parallelism using a message passing paradigm is straightforward and particularly effective given a static

[1] This is only one of several data structures introduced specifically for the distributed treecode and are not used by the sequential algorithm.

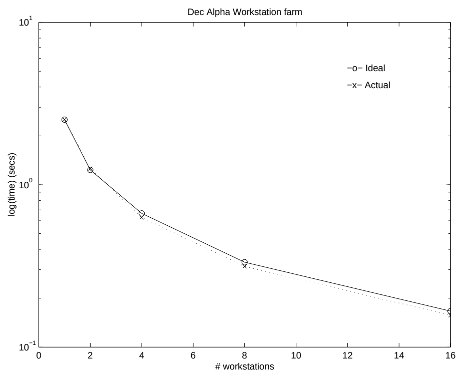

Fig. 1. Performance on the DEC Alpha cluster.

computational environment. On the other hand, it is often desirable to employ unused processor cycles in idle or under-used machines to complete a large scale computation. This algorithm offers the opportunity to do so, using its potential for adaptive parallelism. The problem decomposition approach we favour is based on Piranha-type parallelism implemented using MPI 2.0 dynamic processes. Our approach requires that for a given problem, client machines offering a compute service request a part of the computation and return the result. Thus each client consuming a part of the problem, the property which gives the name to the method.

The testbed for the algorithm implementation consists of two clusters of workstations. The first is assembled from commodity components and employs ten Intel based personal workstations running Linux 2.2, a UNIX-like operating system. The nodes are connected over a standard Ethernet network (10Mbits/sec), organised in a one dimensional torus topology. The second cluster consists of sixteen high end Digital Unix 4.0 workstations connected over fast Ethernet and organised in a star topology.

We have implemented the static version of the algorithm on the Alpha workstation farm. Synchronisation was implemented with allreduce operations in a SPMD model. Even on relatively small problems ($N = M = 32,000$) the method scales very well. In this case, the I/O is local at the filesystem of each workstation and data are distributed and collected using standard operating system services rather than a distributed filesystem. The actual performance of the method is shown in Figure 1. Note that for a problem of similar size the best scalable implementation of a treecode achieves approximately 65% efficiency [12, 11]. On the other hand, the fast evaluation method examined here achieves in excess of 94% efficiency. A treecode has achieved similar speedup only on a shared memory multiprocessor.

The dynamic parallelism variant of the method also offers competitive performance. In test runs between one and twenty workstations participated to the computation. For a problem of size $N = M = 10^8$ it was possible to reduce the computation time by a factor of eight on the low bandwidth cluster.

5 Conclusions

In this paper we have discussed the scalability properties of a rapid evaluation method for Radial Basis Function. In contrast to hierarchical methods, this algorithm exhibits a regular computation and communication structure due to the localisation and smoothness characteristics of the radial basis function method. This regularity results in predictable patterns which can be exploited to provide for a scalable implementation even on low bandwidth clusters of workstations. In particular, due to the intimate relation between the many body problem using the Plummer potential and the evaluation of the Inverse Multiquadric, we anticipate that a hybrid method can be devised which will benefit from the scalability properties of Algorithm 1 for the computation of far filed interactions in many body calculations.

References

1. A.W. Appel (1985) "An Efficient Program for Many-body Simulation", *SIAM J. Sci. Stat. Comp.*, Vol. 6, No. 1, pp. 85-103.
2. R.K. Beatson and G.N. Newsam (1992) "Fast Evaluation of Radial Basis Functions: Part I", *Comp. Math. Applic.*, Vol. 24, No. 12, pp. 7-19.
3. W. Gautschi (1994) "Algorithm 726: ORTHOPOL – A Package of Routines for Generating Orthogonal Polynomials and Gauss-type Quadrature Rules", *ACM Trans. Math. Soft.*, Vol. 20, No. 1, pp. 21-62.
4. L. Greengard (1987) *The Rapid Evaluation of Potential Fields in Particle Systems*, The MIT Press.
5. L. Greengard and J. Strain (1991) *The Fast Gauss Transform*, SIAM J. Sci. Stat. Comput, Vol. 12(1), pp 79 - 94.
6. R.L. Hardy (1997) "The Mathematical Physics of a Biharmonic Approach to Disturbing Potential based on Multiquadric Summation", *IMACS Conference on Radial Basis Functions*, May 27-29, Pacific Grove, CA.
7. C.A. Micchelli (1986) "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Functions", *Constr. Approx.*, Vol. 2, pp. 11-22.
8. M.J.D. Powell (1993) "Truncated Laurent Expansions for the Fast Evaluation of Thin-plate Splines", *Num. Alg.*, Vol. 5, No. 2, pp. 99-120.
9. George Roussos (1999) "Computation with Radial Basis Functions", *Ph.D. Thesis, Imperial College of Science, Technology and Medicine*, London, UK.
10. D. Suter (1993) "Multipole Methods for Visual Reconstruction", *SPIE Geometric Methods in Computer Vision II*, Vol. 2031, pp. 16-26.
11. M.S. Warren and J.K. Salmon (1992) "Astrophysical N-body Simulations using Hierarchical Tree Data Structures", *Proceedings of Supercomputing 92*, ACM Press, pp. 570-572.
12. M.S. Warren and J.K. Salmon (1993) "A Parallel Hashed Oct-tree N-body Algorithm", *Proceedings of Supercomputing 93*, ACM Press, pp. 12-21.