FULLY AND SPARSELY SUPPORTED RADIAL BASIS FUNCTIONS

E. J. KANSA[1] & P. HOLOBORODKO[2]
[1] Convergent Solutions, Livermore, USA.
[2] Advanpix LLC, Yokohama, Japan.

ABSTRACT
The central idea of this paper is that computer mathematics is not identical to ideal mathematics because computer numbers only have finite precision. All functions, especially the positive definite transcendental functions, are truncated and the expansion coefficients have finite precision and all branching operations require time to complete. Of all the known methods used to obtain numerical solutions to integral and partial differential equations, the global continuously differential radial basis functions (RBFs) that are implemented on computers closely resemble many aspects of ideal mathematics. The global RBFs have the attributes required to obtain very accurate numerical results for a variety of partial differential and integral equations with smooth solutions. Without the need for extremely fine discretization, the global RBFs have their spline properties and exponential convergence rates. The resulting system of full equations can be executed very rapidly on graphical processing units and field-programmable gate arrays because, with full systems, there is no branching and full systems solvers are very highly vectorized, optimizing the usage of very fast processors.
Keywords: branching, compactly supported radial basis functions, continuously differentiable, exponentially convergent, full equations, globally supported radial basis functions, multivariate, polynomials, sparse equations, splines, transcendental functions, wavelets.

1 INTRODUCTION

This paper attempts to clarify the differences between Platonic, ideal mathematics practiced in universities and the finite precision mathematics residing on electronic computers. First, in Platonic mathematics, rational and irrational numbers exist with infinite precision. Any piecewise continuous function, $F(x)$, can be represented perfectly as a linear combination of a convenient set of basis functions, $\varphi(x)$, as

$$F(x) = \sum \varphi_j(x)\, \alpha_j,$$

where $\varphi_j(x)$ is the basis function and α_j is its corresponding expansion coefficient. For transcendental functions, the expansion includes an infinite number of terms.

$$\varphi_g = \exp\left[-\left(r_{ij} / c_j\right)^2\right],$$

$$\varphi_{mq} = \left[1 + \left(r_{ij} / c_j\right)^2\right]^s.$$

where

$$r_{ij} = \left\{\sum\nolimits_k^n \left(x_i^k - y_j^k\right)^2\right\}^{1/2}$$

and $c_j > 0$ and $s \geq -1/2$.

2 MATHEMATICS IN ELECTRONIC COMPUTER WORLD

In contrast, no electronic computer has either infinite memory or infinite precision. Although beings existing in electronic computers are intended to mimic their counterparts in the Platonic world, this mimicking process is only a crude approximation and only partially

© 2020 WIT Press, www.witpress.com
ISSN: 2046-0546 (paper format), ISSN: 2046-0554 (online), http://www.witpress.com/journals
DOI: 10.2495/CMEM-V8-N3-208-219

interchangeable. All transcendental functions on electronic computers must be finite-length programs that act for a non-zero duration. Branching operations can be classified as either conditional or unconditional that require the expenditure of time for the targeted conditions to arise, sometimes stalling the logical procedural flow. All numbers represented in computers occupy a finite number of bits in the computer memory. Bits are the electronic memory spaces that form computer words [1], whether they be half, single, double, or quadruple precision words. In the ideal Platonic world, rational and irrational numbers have an infinite number of digits, such as the number π. However, the computer representation of π asymptotically approaches the ideal Platonic number π, with increasingly more digits. A computer cannot store an infinite number of digits.

The next class of important material beings are functions that relate an independent real, imaginary, or complex number to a dependent real, imaginary, or complex number by a given correspondence. Functions are specialized computer programs that occupy the circuit space and require a finite amount of computer processing time. Elementary operations such as add, subtract, multiply, or divide are stored and executed as machine language programs that enable the activities of more complex computer beings; these can be either simple polynomials or vectors, matrices, and their associated operations.

Splines are piecewise continuous functions for which all derivatives, up to order p, are continuous over the domain of definition. Some computer basis functions can range from piecewise constant, linear, quadratic, or triangles, trapezoids, and pyramids. Either a general approximating or interpolating function from polynomials or sines can be constructed from the expansion coefficients by either a least-squares procedure or solving a set of linear equations at specified data centers at which the unknown function is specified. Typically, low-order polynomials are used because of the notorious polynomial snaking that makes differentiation very unreliable. A more advanced class of functions are computer transcendental functions. Other types of basis functions are the multi-resolution wavelets that are either compactly supported wavelets polynomials or transcendental one-dimensional functions. This paper will concentrate on radial basis functions (RBFs) that can be (1) compactly supported polynomial, (2) globally supported polynomial functions, and (3) globally supported, conditionally positive definite transcendental basis functions (CD-RBFs).

All classes of RBFs can operate in any dimensional space because the independent variable is either the Euclidean pair or the geodesic pair separation. All classes of RBFs can be used for not only interpolations and approximations, but also for integral equations (IEs) and partial differential equations (PDEs) [1–2]. However, the procedure to find the unknown set of expansion coefficients, $\{\alpha\}$, can yield round-off errors if the machine precision is not sufficient for the solution procedure. This problem occurs when the condition number of a system of equations having a matrix $\mathbf{A}$ has a condition number, κ, which is greater than the inverse of the machine epsilon, ε_m. During the numerical solution of a system of linear or nonlinear equations, unstable numerical results may occur due to the accumulation of rounding errors [3].

No electronic computer has either infinite memory or infinite precision. Although beings existing in electronic computers are intended to mimic their counterparts in the Platonic world, this mimicking process is only a crude approximation.

Branching redirects the computational flow to another direction [4]. Branching can be either conditional or unconditional. In the computer world, a branch instruction can change the program counter of a central processing unit (CPU) that stores the memory address of the next instruction to be executed. Therefore, a branch can cause the CPU to begin fetching its instructions from a different sequence of the memory, thereby changing the control flow.

When a branch is not taken, the CPU's program counter is unchanged; the flow of control is unchanged. The CPU does not know ahead of time which path will be taken. Whenever a branch is encountered, the CPU must stall until the decision has been resolved, and discards everything in the pipeline that is behind the branch instruction. This stalling lowers performance that is often ignored.

3 SOME COMPUTER SCIENCE TOOLS

Modern computers employ a variety of techniques to accomplish tasks. One common method, especially on higher-end processors with floating point units, is to combine a rational fraction approximation (an approximation to an infinite series) with range reduction and a table lookup, and then use the polynomial to compute the correction. Devices that lack hardware multipliers often use an algorithm called CORDIC and related techniques that use only addition, subtraction, bit-shift, and table lookup. These methods are commonly implemented in hardware floating-point units for enhanced performance. For very high-precision calculations, when series expansion convergence becomes too slow, trigonometric functions can be approximated by the arithmetic-geometric mean that approximates the trigonometric function by the complex elliptic integral.

Several scientific applications that involve transcendental functions require a very high degree of arithmetic precision. The hardware implementation of a parameterizable floating-point library for computing transcendental functions employ both the CORDIC algorithm and Taylor series expansions. Unfortunately, many RBF users are unaware of the computer science aspects of computer mathematics [4]. The weak form of the PDEs requires integration with a test function. Integration has the benefit of smoothing any numerical noise caused by differentiation. When utilizing Gauss's divergence theorem, volume integration is transformed into the difference of two surface integrals. The weak form may involve the blend of analytic and numerical integration methods with CD-RBFs going beyond one dimension; this complication should be considered beforehand. In the computer world, the expansion coefficients are found from the discretization of the integral equation, the partial differential equation, or integral partial differential equations; see Ref. [5] for examples and details.

Some basis functions can range from piecewise constant, linear, quadratic functions, or triangles, trapezoids, and pyramids. Typically, low-order polynomials are used because of the notorious polynomial snaking that makes differentiation very unreliable. It was documented in [2, 5] that for the numerical solutions of PDEs, the strong-form numerical solution is sufficient since the partial derivatives can be calculated with a high degree of accuracy.

However, the procedure to find the unknown set of expansion coefficients can yield round-off errors if the machine precision is not sufficient for the solution procedure; for example, when solving a system of linear or nonlinear equations, unstable numerical results may occur due to the accumulation of rounding errors [3]. Modern computers use a variety of techniques. One common method, especially on higher-end processors with floating point units, is to combine a rational fraction approximation (an approximation to an infinite series) with range reduction and a table lookup, and then use the polynomial to compute the correction. Devices that lack hardware multipliers often use an algorithm called CORDIC and related techniques that use only addition, subtraction, bit shift, and table lookup. These methods are commonly implemented in hardware floating-point units for enhanced performance. For very high-precision calculations, when series expansion convergence becomes too slow, trigonometric functions can be approximated by the arithmetic–geometric mean that approximates the trigonometric function by the complex elliptic integral.

4 METHODS FOR SOLVING EQUATION SYSTEMS

Direct methods compute the solution to a problem in a finite number of steps. These methods would give the precise answer if they were performed in infinite precision arithmetic. Examples include Gaussian elimination, the QR factorization method for solving systems of linear equations, and the simplex method of linear programming. In practice, finite precision is used, and the result is an approximation of the true solution (assuming stability). In contrast to direct methods, iterative methods are not expected to terminate in a finite number of steps. Starting from an initial guess, iterative methods form successive approximations that converge to the exact solution only in the limit.

A convergence test, often involving the residual, is specified in order to decide when a sufficiently accurate solution has been found. Even by using an infinite precision arithmetic, these methods would not reach the exact solution within a finite number of steps. Examples include Newton's method, the bisection method, and the Jacobi iteration. In computational matrix algebra, iterative methods are generally needed for large problems. With iterative methods, one needs to decide if the current approximate solution is sufficiently convergent to terminate the iterative procedure [6].

Iterative methods, such as the GMRES and the conjugate gradient method, are more commonly used instead of direct methods. For iterative methods, the number of steps needed to obtain the exact solution is so large that an approximation is acceptable. Furthermore, the original continuous problem must be replaced by a discrete problem whose solution is known to be an approximation.

5 COMPUTING AND TRUNCATION ERRORS

Computing errors may be either rounding errors, truncation errors, or stability errors. As discussed previously, computer words are finite-sized beings that occupy a finite amount of computer memory. Truncation errors are committed whenever a mathematical procedure is approximated, and the approximate solution differs from the exact solution. Similarly, discretization induces a discretization error because the solution of the discretized problem does not coincide with the solution of the continuous problem. A process is called ill-conditioned if a very small error in the input parameter results in a huge error in the output parameter.

Addition, multiplication, exponentiation, and division of positive numbers are all well-conditioned operations, whereas subtraction is ill-conditioned. No matter how carefully these operations are performed, there is a rounding error that depends upon the number of digits being used. Once any error is generated, it will propagate through the calculation. Any approximate numerical algorithm will create a truncation error because either the exact algorithm was unknown or not utilized. A good example of a computational instability error is the negative diffusion that causes the numerical solution to accumulate successively larger errors in time that the computer code will eventually crash.

Integration was developed in academic mathematics by constructing a large sequence of trapezoids under a curve, $f(x)$, of width, Δx, and then letting both the number, N, of trapezoids go to infinity while simultaneously letting the trapezoidal width, Δx, go to zero. The integral of $f(x)$ is the limiting sum of an infinite number of trapezoids in the nonphysical Platonic world. However, on an electronic computer, this procedure is not possible because of both memory and execution speed restrictions. The process of calculating the integral of function exactly requires one to find the sum of infinite sum of trapezoids. But numerically, one can only find the sum of a finite number of trapezoids. The computer world approximation of the mathematical procedure can only approach asymptotically the exact solution. Similarly, to differentiate a function, the differential element approaches zero, but numerically one can only choose a non-zero value of the differential element.

Both the original problem and the algorithm used to solve that problem can be either well-conditioned or ill-conditioned. An algorithm that solves a theoretically well-conditioned, well-posed problem may be either numerically stable or unstable. One art of numerical analysis is to find a stable algorithm for solving a well-posed mathematical problem. An example worthwhile studying is the 2 × 2 matrix, **H**, in [4],

$$\mathrm{H} = \begin{bmatrix} 1+\delta & 1 \\ 1 & 1 \end{bmatrix}.$$

H can become numerically singular if $\mathrm{H}_{11} \to 1$. If the condition number is $6/\delta > 1/\varepsilon_m$, where ε_m is the machine epsilon, the matrix is numerically singular.

6 OPTIMIZING DATA CENTER LOCATIONS

The discretization poses three problems: (1) Is the discretization sufficiently representative in $\Re^n$ to capture the local and global extremes of the desired unknown solution? (2) Can the total number of data centers be minimized to control the magnitude of the condition number? (3) Can the curse of dimensionality be mitigated? Let **L** be either the partial differential or integral operator over the interior domain and let **B** be the appropriate well-posed boundary operators on the boundaries. The appropriate set of governing equations are:

$$\mathbf{L}U(\mathbf{x},t) = f(\mathrm{x},t) \text{ over } \Omega\backslash\partial\Omega,$$
$$\mathbf{B}U(\mathbf{x},t) = g(\mathbf{x},t) \text{ on } \partial\Omega.$$

If both the interior and boundary forcing functions are set to zero, the exact solution for the Laplace equation would be a quadric equation; for a hyperbolic equation, the exact solution would be a wave equation with constant characteristics. These nonzero forcing functions force local changes in the solution, often producing extrema and saddle points near or at the region upon which they interact. Such local regions require sufficient spatial resolution.

For CD-RBFs, the accuracy improves if some data centers that are used to calculate the IE, PDE, or IPDE can extend slightly beyond the boundary [7]. Increasing the exponent, s in the MQ RBF, causes the MQ basis function to become flatter near the data center, $\mathbf{y}_j$. In the same study, two sets of shape parameters, {cI}: interior shape parameter set, $\{\mathrm{c_I}\}$, and a boundary shape parameter set, $\{\mathrm{c_B}\}$, are used with improved errors [8].

7 CONTROL OF ROUNDING ERRORS

Rounding is a process by which there is a loss of significant digits in floating-point operations. Relevant to the rounding procedure is the machine epsilon, ε_m, which is defined as the maximum relative error. The goal is to control and correct the rounding errors during numerical computations. A summary of the procedures and algorithms to control the accumulation of round-off errors is found in Refs. [9–13]. Historically, CPU architectures were designed without any intention to use arbitrary-precision computations because of very small market demands. Most computer applications such as games and social networking do not require extended precision. Consequently, software methods must be employed to obtain extended precision that is limited to the miniscule scientific computing market.

Higham [14] has praised the Advapix Multi-precision Computing Toolbox (AMCT) [4], as the best multi-precision software package to date. AMCT employs the best computer science practices for a large suite of multi-precision applications. If implemented properly, extended precision computations can be very fast.

The traditional method to enhance the convergence of finite difference, finite element, or finite volume methods as well as compactly supported RBF methods is to refine greatly the

discretization to make the average point separation, <h>, as small as possible, thereby increasing the total number of discretization points, *N*. This approach works well as long as there is sufficient computer memory. However, as higher dimensional problems are attempted, the curse of dimensionality dominates. Then the option of increasing the average value of the shape parameter, <c>, is very attractive, except that increasing the average of the shape parameter may make the condition number so large as to render any calculations meaningless.

Some authors treat the double-precision condition number, of $O(10^{16})$, as an impenetrable barrier similar to the vacuum speed of light. Some authors [15–18] used extended precision that enables the computer arithmetic operations and associated mathematics to resemble asymptotically the idealized Platonic mathematics of academia. The notable exception was Ref. [19] that used MATLAB's miserably slow variable precision arithmetic (VPA) in the time-dependent time integration of 4D Burgers' equations. The simulations were halted after 30 days of continuous computations, and VPA was abandoned. Often ignored by academic mathematicians is the role of computer science in computer arithmetic, and its role in computational mathematics.

8 EXAMPLES OF AMCT PERFORMANCE

The calculation of eigenvectors and eigenvalues, and especially the eigenvectors of a general matrix **A**, may give erroneous results even though the matrix **A** is well conditioned [20]. The eigenvalues of the Grcar matrix were computed by MATLAB in double precision and by the AMCT in extended precision. In ideal Platonic mathematics, the eigenvalues of the Grcar matrix and its transpose should be identical. In Fig. 1, the eigenvalues of the Grcar matrix are

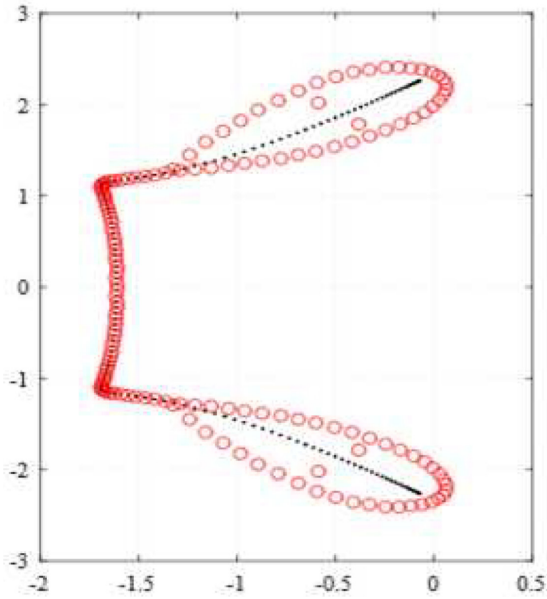

Figure 1: Eigenvalues of the Grcar matrix and its transpose calculated in double precision. The condition number of the Grcar matrix and its transpose is 3.61. The dots are the eigenvalues of the Grcar matrix, and the open circles are the eigenvalues of the transpose matrix.

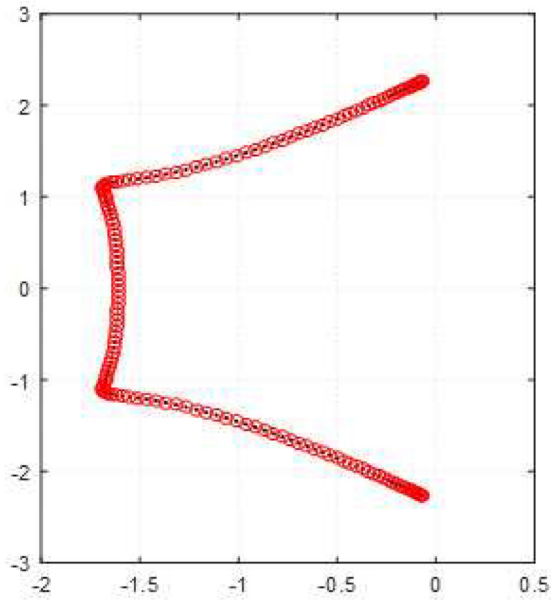

Figure 2: The plots of the eigenvalues calculated in quadruple precision. Both the eigenvalues of the Grcar matrix (dots) and its transpose are now calculated in quadruple precision. The condition number of the Grcar matrix and its transpose are both 3.61.

displayed in dots, and the eigenvalues of the transposed matrix in open circles. In theory the dots and open circles should coincide, especially since the condition number, κ, of the Grcar matrix is small, $\kappa(\mathbf{A}_G) = 3.61$.

The second example is the Noschese-Pasquini-Reichel (NPR) matrix, which is a tridiagonal Toeplitz matrix. In ideal Platonic mathematics, the eigenvalues and eigenvectors of this tridiagonal Toeplitz matrix must give identical eigenvectors and eigenvalues, especially since $\kappa(\mathbf{A}_{NPR}) = 1.81$. When both the Gcrar and the tridiagonal Toeplitz matrices are calculated with 50 digits of accuracy, then the eigenvalues of both matrices and their transposes coincide (see Fig. 2). Single and double precision are not reliably capable of resembling ideal Platonic mathematics.

9 LUH'S RBF SHAPE PARAMETER THEORY

Luh [21–23] made a significant achievement by developing a theory to find the optimal CD-RBF shape parameters by greatly expanding the original work of Madych and Nelson [24]. Luh considered factors such as the type CD-RBF spatial dimension and fill distance to construct the MN (Madych–Nelson) curve. For example, Luh constructs an MN(c) curve for the generalized MQ-RBF and searches for the minimum region of MN curve where c is optimal. Luh used Mathematica's extended precision features to calculate the interpolating functions, avoiding ill-conditioning caused by the theoretically optimal choice of the shape parameter, for a variety of interpolation cases obtaining interpolation errors of less than 10^{-147}. With a strong foundational theory for choosing the average shape parameter by Luh's theory, important multi-dimensional PDE and IE problems can be simulated in extended precision on massively parallel computers. The guesswork previously employed in choosing the optimal shape parameters are now obsolete.

10 DOMAIN DECOMPOSITION FOR RBF SYSTEMS

The dominant method to solve very large systems of equations on parallel computers is the domain decomposition method (DDM). The DDM can be iterative or non-iterative, either of which has been successfully implemented on both the sparse and full matrices arising from both sparsely and globally supported RBF-based applications. For convenience, the following analysis will be limited to full systems arising from CD-RBFs. Considering global systems, the DDM splits a huge boundary value problem of rank N into K smaller boundary value problems, each of which are of rank N_K, where $N_K << N$. The condition number, κ, of a matrix depends upon its rank, distribution of data centers, and the distribution of large shape parameters. The most prevalent approach for DDM is the iterative Picard scheme. Within a Picard iteration loop, the unknown function and its normal and tangential derivatives on either side of the domain split are iterated until all of the continuity discrepancies tend toward zero. The full system requires $O(N^3)$ operations for a solution, then, on each sub-domain, $O(N_K{}^3)$ operations are required using the optimally vectorized matrix-vector operations. By operating in parallel, there is a huge reduction in the total computational time. A sample of the literature in which DDM and CD-RBFs are applied can be found in Refs. [25–32].

Alternatively, Refs. [33, 34] used a non-iterative domain decomposition approach that can be characterized as a block Gaussian elimination method. The direct block Gaussian elimination method is a theoretically exact since no iteration methods are used. However, the Schur complement blocks may become ill-conditioned due to rounding errors, and extended arithmetic precision is recommended.

11 LOCAL RBF METHODS

Because global RBFs can become rapidly very ill-conditioned whenever the average shape parameter and the rank become very large, various alternatives to the direct CD-PDE approach have been investigated. Currently, most RBF applications are limited to three dimensions or less, making the curse of dimensionality unimportant. Because of the reluctance to utilize extended precision software packages and rather use single- or double-precision computers, alternatives to direct RBF methods were developed. Local RBF alternatives to direct CD-RBFs such as RBF-FD, RBF-QR, RBF-RA and Hermite-based RBFs are among the approaches that were developed [35–40]. Instead of spectral convergence rates, the convergence rates are fourth to sixth order. If the PDEs and IEs are primarily two- and three-dimensional, then the slower convergence is hardly an issue. There are several approaches that reduce the condition number issue using CD-RBFs by transforming them into more benign basis functions, albeit with slower convergence rates. The radial basis function-rational approximation (RBF-RA) of vector-valued analytic functions has all components of the vector and share the same singularities. The RBF-RA is more accurate, robust, and easier to implement. In contrast to the stable RBF-QR and RBF-GA algorithms that are based on finding a better conditioned base in the same RBF-space, the RBF-RA can be used with any type of smooth radial kernel. The RBF-GA seeks to find a numerically well-conditioned basis function set in the same function space that is spanned by the ill-conditioned nearly flat original Gaussian RBFs. By exploiting some of the properties of the incomplete gamma function, the change of basis can be achieved without requiring any infinite expansions or their approximations.

12 SPARCE AND FULL EQUATION SOLVERS

The performance of direct solvers for sparse matrices [10, 41, 42] depends entirely on the locations of non-zero elements. Bad locations of zeros can require full $O(N^3)$ complexity. To

alleviate this complexity problem, all sparse system algorithms use matrix pre-ordering to minimize the fill-in; this pre-ordering must be considered in calculating the total computational time. The optimal pre-ordering algorithm is known only for symmetric-positive definite matrices. But, in the case of unsymmetrical matrices, optimal pre-ordering is an NP-complete task. All existing direct sparse solvers use heuristic ideas to perform the pre-ordering. The complexity of direct sparse solvers for sparse matrices can vary wildly from $O(N^3)$ to $O(N^{3/2})$.

Direct sparse solvers have a poor capability for parallel execution, whereas dense solvers are extremely suitable for parallel computers. It is interesting to note in Ref. [38] that in the process of comparing RBF-QA and direct solvers with extended precision, their Figure 12 demonstrates that direct solvers + extended precision + CD-RBFs are more efficient, even though RBF-QA produces a sparse system.

Theoretically, iterative methods are the only way to beat the complexity of direct solvers since iterative solvers are supposed to converge within $O(N^2)$ operations. However, iterative solvers suffer from slow convergence and require good pre-conditioners. Constructing good pre-conditioners is a separate problem that needs to be solved. It may be possible that with careful and non-trivial tuning of all parts of sparse solvers (direct or iterative), sparse solvers may be faster than solving asmall dense matrix in extended precision on a single CPU. However, everything depends on the problem being studied and the computational environment for comparison.

Both GPUs (graphic processing units) and FPGAs (field-programmable gate arrays) possess highly parallel structures that make them more efficient than general-purpose CPUs for algorithms where the processing of large blocks of data are operated upon in parallel. However, GPUs and FPGAs have very limited functionality for branching operations that are abundant in sparse solvers, making it very difficult for porting and running sparse solvers in parallel.

In contrast to the various sparse solvers, dense solvers enjoy massive parallelism because there is no branching and all manipulations can be formulated as efficient BLAS 3/GEM operations that are well optimized and are designed for massive parallelism on all imaginable platforms. With the proper implementation, the direct solver CD-RBF algorithm can be unbeatable in speed and accuracy. To verify this assertion, such codes must be implemented as a specialized low-level C/C++ code for RBFs that are heavily optimized for CPUs, GPUs, or FPGAs. The extreme option is the creation of a specialized processor for RBF based solvers on GPUs and FPGAs whose implementation will be even faster than double precision computations.

13 DISCUSSION

A much-neglected area of modeling inquiry is the simulation of higher dimensional partial differential and integral equations relating to plasma fusion, molecular quantum mechanics, and cellular metabolism. A survey of popular techniques for higher dimensional IE, PDE, and I-PDE problems was presented in Ref. [43]. Unfortunately, Monte Carlo techniques converge extremely slowly as $N^{-1/2}$, and quasi-Monte Carlo methods suffer from biasing. Tensor product finite difference, finite element, and finite volume methods in higher dimensions have limited applicability and operator splitting does not yield meaningful results when processes are coupled dimensionally.

For both local and global RBF matrices, the condition number grows with β increasing number of elements and the shape parameter; however, the condition number increases at a faster rate with global systems. Luh [21–23] developed a shape parameter theory for CD-RBFs

that is implemented with the extended precision feature of MATHEMATICA with extremely accurate results having max and RMS errors, $O(10^{-147})$, against problems with analytic solutions.

Yet, there is a very strong reluctance to execute problems on computers with extended precision, citing the poor performance of sloppily written old extended precision packages that ignored modern developments. Many researchers still cling on to their beliefs that only single- and double-precision computers are only worth considering.

Another strongly held belief is that RBFs that produce sparse systems of equations are superior because of faster execution because the system of equations is dominated by zero entries. The new GPUs and FPGAs are ideal for full equation systems such as those arising from *s* since there is no branching involved that is opposite of that with sparse systems. If the problem being solved is a two- or three-dimensional one, it is possible that the local RBF method would execute faster even using direct sparse solvers. However, if the problem being solved is in four dimensions or higher, the combination of extended precision using shape parameter refinement, non-iterative domain decomposition methods, full-matrix solvers based upon block Gaussian elimination methods with full vectorization on dedicated GPUs and FPGAs have a very high probability to outperform sparse single- or double-precision solvers. There is now evidence in Figure 12 of the paper of Wright and Fornberg [38], that direct solvers + extended precision + CD-RBFs can outperform sparse RBF-QA, contrary to intuition.

REFERENCES

[1] Kansa, E.J., Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics I: surface approximations and partial derivative estimates. *Computers & Mathematics with Applications*, **19**(**8–9**), pp. 127–145, 1990. https://doi.org/10.1016/0898-1221(90)90270-t

[2] Kansa, E.J., Multiquadrics – A scattered data approximation scheme with applications to computational fluid dynamics II: solutions to parabolic, hyperbolic, and elliptic partial differential equations. *Computers & Mathematics with Applications*, **19**(**8–9**), pp. 147–161, 1990. https://doi.org/10.1016/0898-1221(90)90271-k

[3] Kansa, E.J. & Holoborodko, P., On the ill-conditioned nature of RBF strong collocation. *Engineering Analysis with Boundary Elements*, **78**, pp. 26–30, 2017. https://doi.org/10.1016/j.enganabound.2017.02.006

[4] http://www.exforsys.com/tutorials/c-language/decision-making-and-branching-in-c.html/

[5] Chen, W., et al., *Recent Advances in Radial Basis Function Collocation Methods*, Springer Briefs in Applied Sciences and Technology: Berlin, 2014.

[6] Coffer, H.N. & Stadtherr, M.A., Reliability of iterative linear equation solvers in chemical process simulation. *Computers & Chemical Engineering*, **20**, pp. 1123–1132, 1996. https://doi.org/10.1016/0098-1354(95)00074-7

[7] Fedoseyev, A., et al., Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary. *Computers & Mathematics with Applications*, **43**(**3–5**), pp. 439–45, 2002.

[8] Wertz, J., et al., The role of the multiquadric shape parameters in solving elliptic partial differential equations. *Computers & Mathematics with Applications*, **51**(**8**), pp. 1335–1348, 2006. https://doi.org/10.1016/j.camwa.2006.04.009

[9] Rump, S.M., Verification methods: rigorous results using floating point arithmetic. *Acta Numerica*, **19**, pp. 287–449, 2010. https://doi.org/10.1017/s096249291000005x

[10] Higham, N.J., *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM: Philadelphia, 2002.
[11] Brent, R. & Zimmermann, P., *Modern Computer Arithmetic*, Cambridge Monographs on Computational and Applied Mathematics, 2010.
[12] Hida, Y.L., et al., Quad-Double Arithmetic: Algorithms, Implementation, and Application, Lawrence.Berkeley National Laboratory Technical Report LBNL-46996, 2000.
[13] Li, X.S., et al., Design, implementation and testing of extended and mixed precision BLAS. *ACM Transactions on Mathematical Software*, **1**, pp. 152–205, 2002. https://doi.org/10.1145/567806.567808
[14] Higham, N.J., Multi-precision world, *SIAM News*, pp. 2–3, October 2017.
[15] Huang, C.S., et al., Error estimate, optimal shape factor, and high precision computation of multiquadric collocation method. *Engineering Analysis with Boundary Elements*, **31(1)**, pp. 614–623, 2007. https://doi.org/10.1016/j.enganabound.2006.11.011
[16] Huang, C.S., et al., On the increasingly flat radial basis function and optimal shape parameter for the solution of elliptic PDEs. *Engineering Analysis with Boundary Elements*, **34**, pp. 802–809, 2010. https://doi.org/10.1016/j.enganabound.2010.03.002
[17] Cheng, A.H.D., Multiquadric and its shape parameter - a numerical investigation of error estimate, condition number, and round-off error by arbitrary precision computation. *Engineering Analysis with Boundary Elements*, **36**, pp. 220–239, 2012. https://doi.org/10.1016/j.enganabound.2011.07.008
[18] Sarra, S.A., Radial basis function approximation methods with extended precision floating point arithmetic. *Engineering Analysis with Boundary Elements*, **35**, pp. 68–76, 2011. https://doi.org/10.1016/j.enganabound.2010.05.011
[19] Kansa, E.J., & Geiser, J., Numerical solution to time-dependent 4D inviscid Burgers equations. *Engineering Analysis with Boundary Elements*, **37**, pp. 637–645, 2013. https://doi.org/10.1016/j.enganabound.2013.01.003
[20] https://www.advanpix.com/2011/10/12/multiprecision-computationeigenvalues-eigenvector/
[21] Luh, L.T., The choice of the shape parameter – A friendly approach. *Engineering Analysis with Boundary Elements*, **98**, pp. 103–109, 2019. https://doi.org/10.1016/j.enganabound.2018.10.011
[22] Luh, L.T., The mystery of the shape parameter III. *Applied and Computational Harmonic Analysis*, **40**, pp. 186–199, 2016. https://doi.org/10.1016/j.acha.2015.05.001
[23] Luh, L.T., Solving Poisson's equations by the MN curve approach. Preprint, arXiv:1905.07992 [math.NA], 2019.
[24] Madych, W.R. & Nelson, S.A., Multivariate interpolation and conditionally positive definite functions, II, *Mathematics of Computation*, **54**, pp. 211–230, 1990. https://doi.org/10.1090/s0025-5718-1990-0993931-7
[25] Ling, L. & Kansa, E.J., Preconditioning for radial basis functions with domain decomposition methods. *Mathematical and Computer Modelling*, **40**, pp. 1413–1427, 2004. https://doi.org/10.1016/j.mcm.2005.01.002
[26] Koczka, G., et al., An iterative domain decomposition method for solving wave propagation problems. *Electromagnetics*, **34**(**3–4**), pp. 210–221, 2014. https://doi.org/10.1080/02726343.2014.877751
[27] Hon, Y.C. & Wu, Z., Additive Schwarz domain decomposition with radial basis approximation. *International Journal of Applied Mathematics*, **4**, pp. 81–98, 2002.

[28] Li, J. & Hon, Y.C., Domain decomposition for radial basis meshless methods. *Numerical Methods for PDEs*, **20**, pp. 450–462, 2004. https://doi.org/10.1002/num.10096
[29] Ingber, M.A., et al., A mesh free approach using radial basis functions and parallel domain decomposition for solving three dimensional diffusion equations. *International Journal for Numerical Methods in Engineering*, **60**, pp. 2183–2201, 2004. https://doi.org/10.1002/nme.1043
[30] Duan, Y., et al., Coupling projection domain decomposition method and Kansa's method in electrostatic problems. *Computer Physics Communications*, **180**, pp. 200–214, 2009. https://doi.org/10.1016/j.cpc.2008.09.009
[31] Herrera, I. & Rosas Medina, A., The derived-vector space framework and four general purposes massively parallel DDM algorithms. *Engineering Analysis with Boundary Elements*, **137**(**3**), pp. 646–657, 2013. https://doi.org/10.1016/j.enganabound.2012.12.003
[32] Hernandez-Rosales, A. & Power, H., Non-overlapping domain decomposition algorithm for the Hermite radial basis function meshless collocation approach: applications to convection diffusion problems. *Journal of Algorithms & Computional Technology*, **1**, pp. 127–159, 2007. https://doi.org/10.1260/174830107780122685
[33] Kansa, E.J. & Hon, Y.C., Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations. *Computers & Mathematics with Applications*, **39**, pp. 123–137, 2000.
[34] https://en.wikipedia.org/wiki/Schur complement method.
[35] Fornberg, B. & Flyer, N., *A Primer on Radial Basis Functions with Applications to the Geosciences*, SIAM: Philadelphia, PA, 2015.
[36] Gonzalez-Rodriguez, P., et al., Laurent series based RBF-FD method to avoid ill conditioning. *Engineering Analysis with Boundary Elements*, **52**, pp. 24–31, 2015. https://doi.org/10.1016/j.enganabound.2014.10.018
[37] Ilati, M. & Dehghan, M., The use of Radial Basis Functions (RBFs) collocation and RBF-QR methods for solving the coupled nonlinear sine-Gordon equations. *Engineering Analysis with Boundary Elements*, **52**, pp. 99–109, 2015. https://doi.org/10.1016/j.enganabound.2014.11.023
[38] Wright, G.B. & Fornberg, B., Stable computations with flat radial basis functions using vector-valued rational approximations. *Journal of Computational Physics*, **331**, pp. 137–156, 2017. https://doi.org/10.1016/j.jcp.2016.11.030
[39] Rashidinia, J., et al., A stable method for the evaluation of Gaussian radial basis function solutions of interpolation and collocation problems. *Computers & Mathematics with Applications*, **72**, pp. 178–193, 2016. https://doi.org/10.1016/j.camwa.2016.04.048
[40] Fasshauer, G.E. & Zhang, J.G., On choosing "optimal" shape parameters for RBF approximation. *Numerical Algorithms*, **45**, pp. 345–368, 2007. https://doi.org/10.1007/s11075-007-9072-8
[41] Davis, T.A., *Direct Methods for Sparse Linear Systems*, SIAM: Philadelphia, 2006.
[42] Saad, Y., *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM: Philadelphia, 2003.
[43] Sarra, S.A. & Kansa, E.J., *Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations*, Advances in Computational Mechanics, **2**, ISSN:1940-5820, 2009.