Scientific Research Publishing

American Journal of Computational Mathematics, 2025, 15(2), 165-173
https://www.scirp.org/journal/ajcm
ISSN Online: 2161-1211
ISSN Print: 2161-1203

Two to Six Dimensional Numerical Solutions to the Poisson Eigenvalue Partial Differential Equation Using Generalized Multiquadrics

Edward J. Kansa[1], Pavel Holobvorodko[2]

[1]Convergent Solutions, Livermore, CA, USA
[2]Advanpix, LLC, Yokohama, Japan
Email: edwardjkansa@gmail.com, pavel@advanpix.com

How to cite this paper: Kansa, E.J. and Holobvorodko, P. (2025) Two to Six Dimensional Numerical Solutions to the Poisson Eigenvalue Partial Differential Equation Using Generalized Multiquadrics. *American Journal of Computational Mathematics*, **15**, 165-173.
https://doi.org/10.4236/ajcm.2025.152008

Received: February 14, 2025
Accepted: May 16, 2025
Published: May 19, 2025

Abstract

We solve numerically an eigenvalue elliptic partial differential equation (PDE) ranging from two to six dimensions using the generalized multiquadric (GMQ) radial basis functions (RBFs). Two discretization methods are employed. The first method is similar to the classic mesh-based discretization method requiring n centers per dimension or a total n^d points. The second method is based upon n randomly generated points in $\Re^d$ requiring far fewer data centers than the classic mesh method. Instead of having a crisp boundary, we form a "fuzzy" boundary. Using the analytic or numerical inverse interior and boundary operators, we find the local and global minima and maxima to cull discretization points. We also find that the GMQ-RBF "flatness" can be controlled by increasing the GMQ exponential, β. We perform a search to find the smallest root mean squared error (RMSE) by varying the exponent, the maximum, the minimum range of the GMQ shape parameter, and boundary influence, with the exponential having the most influence. Because the GMQ-RBFs are essentially nonlinear, it follows that the starting point of the simple search influences the end result. The optimal algorithm for high dimensional PDEs is still the subject of much research and may wait for the common place availability of massively parallel quantum computers for even higher dimensional PDEs and integral equations (IEs).

Keywords

Arbitrary Precision Arithmetic, Elliptic Partial Differential Equations, Multiquadric Radial Basis Functions, Solutions in Two-To-Six-Dimensional Space, Uniformly Meshed and Fuzzy Boundaries, Randomly Generated Points

1. Introduction

The more familiar practical applications of integral and partial differential equations occur in two- or three-dimensional context; however, higher dimensional equations may occur in financial, quantum molecular and nuclear applications, nuclear fusion, genetics, etc. Since analytical solutions do not exist or are so overwhelmingly complex, numerical solutions are required.

The standard approach of discretization of the in-dimensional hyperspace soon leads to the dilemma of the "curse of dimensionality", see [1]. For example, in a 3D problem, a discretization of 10 points per dimension requires 10^3 data centers whereas a problem in 10D with 10 points per dimension requires 10^{10} data center variables, and the correspond coefficient matrix will have 10^{20} elements.

A variety of algorithms have attempted to address the "curse of dimensionality". Among these are: operator splitting [2], Monte Carlo methods [3], and Quasi Monte Carlo methods [4], Domain Decomposition, see [5] [6] and Adaptive Sparse Grids see [7]. Papers [2] to [6] use the neural network (NN) procedure to execute a large number of simulations to train the NN to obtain optimal results.

Domain decomposition is most useful for elliptic PDEs; the solution is split into many approximately equally sized smaller problems on separate subdomains using pseudo boundary conditions. Using overlapping or non-overlapping conditions, the solutions are iterated until the error drops below the acceptance threshold. The variable transformation process can be as simple as a "rotation of a linear combination of independent variables to achieve a simpler system"; an example would be the combination of the mass, total energy, and principal momentum PDEs to produce the characteristic variables. Neural networks, see [8] [9], have been used to find approximate dimension reduction to problem of a large number of dimensions. The purpose of dimensionality reduction aims to provide better understanding of the data for both you and your model, since without dimensional reduction, many problems would be impossible to simulate.

In [10], the authors develop a new method of scaling up physics-informed neural networks (PINNs) to solve arbitrary high-dimensional PDEs; the Stochastic Dimension Gradient Descent (SDGD), decomposes a gradient of PDEs into pieces corresponding to different dimensions. Dimensionality reduction can be achieved by computing the eigenvectors of the covariance matrix of the independent coordinates. Operator splitting can have an effect if and only if processes are sufficiently separable or weakly coupled, see [11].

In this paper, we will examine the traditional numerical approach of discretizing the domain, Ω, consisting of d-dimensions. If ethe ζth dimension of each is discretized with m_k^ζ points, then the total number of discretization points N is:

$$N = \Pi_\zeta^D m_k^\zeta \tag{1}$$

Assume $U(\boldsymbol{x})$ is a function of point, $\boldsymbol{x} \in \Re^d$. Also, assume that $U(x)$ can be expanded in terms of N basis functions such that

$$U(\boldsymbol{x}) = \sum_{j=1}^N \phi_j(x)\alpha_j \tag{2}$$

where ϕ_j is a basis function $\boldsymbol{x} \in \Re^d$ and α_j is the that is found by solving the interpolation problem:

$$\Phi \alpha = \boldsymbol{U} . \tag{3}$$

where $U(x_i)$ is an $N \times 1$ column vector of the dependent variable at N locations, and Φ is an $N \times N$ matrix of the form:

$$\begin{pmatrix} \phi_{1,1} & \cdots & \phi_{1,N} \\ \vdots & & \vdots \\ \phi_{N,1} & \cdots & \phi_{N,N} \end{pmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} = \begin{pmatrix} U_1 \\ \vdots \\ U_N \end{pmatrix}, \tag{4}$$

where $\phi_{i,j}$ represents the ith basis function evaluated at the point, $\boldsymbol{x}_j$.

Consider the general integral or partial differential equation of the form:

$$\mathcal{L}U(\boldsymbol{x},t) = f(\boldsymbol{x},t) \quad \text{over } \Omega \backslash \partial\Omega \text{ with } d_i \text{ scretization points,} \tag{5}$$

$$\mathcal{B}U(\boldsymbol{x},t) = g(\boldsymbol{x},t) \quad \text{on } \partial\Omega, \text{ with } n_b \text{ boundary discretization points,} \tag{6}$$

The subscripts i and b refer to the interior and boundary domains, respectively.

We use the generalized multiquadric (GMQ) radial basis function:

$$\phi(x) = \left[1 + r_{ij}^2 / c_j^2\right]^{\beta} \tag{7}$$

where

$$r_{ij}^2 = \left(\boldsymbol{x}_i - y_j\right)^2 \tag{8}$$

where r_{ij}^2 is the squared Euclidean distance or an appropriate geodesic, and c_j^2 is the dilutional shape parameter, and β is non-integer exponent, and $\beta \geq -1/2$.

Assume the inverse operators, $\mathcal{L}^{-1}$ and $\mathcal{B}^{-1}$, exist. Then, the solution, $U(x,t)$, over the entire domain, Ω, is:

$$U(x,t) = U_I(x,t) + U_B(x,t). \tag{9}$$

If Dirichlet boundary conditions are enforced, then $\mathcal{B}$ on the boundary is the identity operator. Then we can locate the maxima, minima and inflection points on the boundary and discretize these locations. If the discretized points are stored as a matrix of n_B rows and n_d columns, then we may search the matrix for the local and global maxima and minima. Similarly, over the interior, we can locate the maxima, minima, and inflection points to discretize these locations. In addition, if the local interpolation is a low order polynomial, some points may be culled from the discretization set, similar to the sparse grid method for either meshed or randomly ordered discretization.

The test problem studied in this paper is the eigenvalue problem over a hypercube of 2 to 6 dimensions.

$$\Sigma_{\zeta}\left(\partial_{\zeta}^2\right) U(x) = \lambda^2 U(x) \quad \text{over } \Omega \backslash \partial\Omega, \tag{10}$$

$$U(x,t) = \cos\left(\Sigma_i x_i\right) \quad \text{on } \partial\Omega. \tag{11}$$

We shall obtain the solutions by traditional uniform meshing and the proposed method of finding the maxima, minima, and saddle points and discretized these regions with randomly generated d-dimensional points.

A d-dimensional point is represented by

$$\boldsymbol{x} = \left(x^1, x^2, \cdots, x^d\right), \tag{12}$$

where the superscript denotes the dimensionality of the point, and the subscript denotes the index of the point location. We shall assume that the computational domain is a dimensional hyper-cube and whose boundary surfaces are (d-1)-dimensional hypercubes. Each hyper-cube has 2^d vertex points, 3d lines, and 2d surfaces.

For simplicity, assume the hyper-cube is discretized by m points along each dimension. Then a uniformly distributed set of m points per dimension will have N points, where

$$N = m^d . \tag{13}$$

The problem to be solved numerically is a d-dimensional Poisson eigenvalue problem of the form:

$$\nabla^2 U\left(\boldsymbol{x}\right) = -\lambda^2 U\left(\boldsymbol{x}\right) \quad \text{over } \Omega\backslash\partial\Omega, \tag{14}$$

$$U\left(\boldsymbol{x}\right) = \cos\left(x^1 + \cdots + x^d\right) \quad \text{on } \partial\Omega. \tag{15}$$

The exact solution is:

$$U^{exact}\left(\boldsymbol{x}\right) = \cos\left(x^1 + \cdots + x^d\right), \tag{16}$$

where $-\lambda^2 = -d$.

The forcing function for the interior problem, $\Omega\backslash\partial\Omega$, is simply ∇^2 whose inverse is $\int \mathrm{d}x \int \mathrm{d}x'$. Given that

$$f\left(\boldsymbol{x}\right) = \cos\left(x^1 + \cdots + x^d\right), \tag{17}$$

$$\text{then} \quad F\left(x\right) - d * \cos\left(x^1 + \cdots + x^d\right). \tag{18}$$

Each side of the d-dimensional hypercube ranges from $-\pi/2$ to $\pi/2$. Since the random number d-dimensional generator will yield numbers from $-\pi/2$ to 1, each coordinate must be linearly scaled to the interval $[-\pi/2, \pi/2]$. In addition, from Equations (17) and (18), the two to six dimensional summations range from $[-\pi/2, \pi/2]$ to $[-6\pi/2, 6\pi/2]$, respectively. Rather sparse sampling between the extrema is needed, thereby allowing for the culling of superfluous data centers. Because of the random nature of the process, we will not predict the size of the set of equations.

We can determine the local and global maxima and minima by examining $F(\boldsymbol{x})$, the analytically or numerically integrated forcing function. For discretization points away from the extrema, we can delete those points between extrema if the variations are small in magnitude.

The analytic or numerical integration of the forcing function over both the interior and boundary and domain will yield information about the locations of the global and local extrema. Having found the extrema locations, those points associated with smooth variations can be deleted, thus reducing the condition number and execution time. However, the deletion of points is only convenient for ran-

domly generated points.

Two discretization schemes will be used to solve numerically the Poisson eigen value problem:

1) The classic mesh-based discretization scheme, and

2) The randomly generated discretization scheme with data center culling.

2. Classic Mesh Discretization Results

The classical discretization method is to the familiar classic finite difference, elements or volumes methods. for 5 and 6 dimensional hypercubes the memory capacity of most available computers in the set of tables below, we present the results of a search for the lowest RMSE. Note that the generalized MQ RBF, see [11] [12]. is per se nonlinear, and nonlinear problems, can have multiple solutions depending upon the starting set of parameters.

In this set of calculations, the boundary surfaces are each populated my m randomly distributed points, and the interior domain is populated by randomly distributed points.

The computer used for the calculations is the following: CPU: AMD EPYC 7V13 64-core, 2.45GHz/3.7GHz, RAM: 256GB, GPU: 4 x RTX 4090.

We initialize the optimization search for all considered dimensions by starting with the 2D search first and setting the GMQ exponent, $\beta = 0.5$, the minimal value of the shape parameter, $c_{\min} = 1.0$, the maximum value of the shape parameter, $c_{\max} = 1.0$, and the multiplier of the boundary forcing functions, $s = 1.0_k$. The optimal value for the 2D results were the initial values for the 3D results, and so on until the 5D results were the initial values for the 6D optimization.

These parameters are optimized by performing a simple by performing a simple search on the variations of the parameters that yield the smallest root mean squared errors (RMSE). Rather than using an optimization -based library routine, we used three "do" loops per search parameter. We allowed a parameter to increase by 1.03, stay unchanged, and decrease 1/1.03, each time calculating the RMSE, while saving the optimized set of parameters. The entire search was performed in an overall search 100 times. This approach was performed for each dimension.

Based upon the results of papers [12] [13], we allowed the GMQ exponential, β, to become quite large allowing the GMQ basis function to become very flat near the data center, $\boldsymbol{y}_j$, and rise very rapidly near the evaluation center, $\boldsymbol{x}_i$. In addition, the pre-wavelet dilation parameters, $c_{\max}^2$ and $c_{\min}^2$ also are used to control the flatness of the GMQ basis function near x_i and y_j. The parameter, s_k, is used to increase the boundary shape parameter. The distributed GMQ shape parameter is assumed to have an exponentially varying distribution of the shape parameters, form, see [14]. In addition. Buhmann, see [15], showed C^∞ are pre-wavelets having problem dependent length scales. We use a simple form of the shape parameter distribution given by:

$$c_j^2 = c_{\min}^2 \exp\left(\psi\left(j-1\right)/\left(N-1\right)\right) \tag{19}$$

where

$$\psi = \log\left(c_{\max}^2 / c_{\min}^2\right) \tag{20}$$

To obtain the highest performance of the GMQ, it will be necessary to use arbitrary precision arithmetic to avoid ill-condoning or loss of numerical precision. Since computer chip manufacturers realize scientific computing is only a miniscule portion of the market, a software alternative is required, and the best available is ***ADVANPIX***, see [16].

The tables one to five present the results of our searches of the mesh-based discretization. Note, for the five and six dimensional cases, our super computer memory restrictions prevented. Further mesh refinement. The procedure used in found in reference [17] (**Tables 1-5**).

Table 1. Two dimensional results.

EMSE	Digits	β	s_k	$c_{\max}^2$	$c_{\min}^2$	m_d
0.2884	200	28.794	1.093	3.91	1.176e−2	3
0.3438	200	26.664	0.988	2.782	9.552e−3	4

Table 2. Three dimensional results.

RMSE	Digits	β	s_k	$c_{\max}^2$	$c_{\min}^2$	m_d
7.5577e−5	220	45.224	1.41	2.472	8.4972e−3	3
2.460e−3	220	46.35	1.495	2.010	6.9092−3	4
0.481465	220	43.427	1.304	6.365	−7.115e−3	8

Table 3. Four dimensional results.

RMSE	Digits	β	s_k	$c_{\max}^2$	$c_{\min}^2$	m_d
0.465	200	35.12	1.75	1.3541	3,46e−2	4
0.767	200	25.736	1.21	2.284	4.72e−3	5

Table 4. Five dimensional results.

RMSE	Digits	β	s_k	$c_{\max}^2$	$c_{\min}^2$	m_d
0102989	220	37.6868	1.540	1.8384	0.006708	3
0.01076	220	30.6318	1.018	2.54616	8.071e−3	4
0.481465	220	28.471	0.8041	6.3656	7.1154e−3	8

Table 5. Six dimensional results.

RMSE	digits	β	s_k	$c_{\max}^2$	$c_{\min}^2$	m_d
3.604e−5	400	41.343	1.495	2.136	6.908e−3	3
2.127e−5	500	49.837	1.898	2.010	9.219e−3	4

Because of the exponential growth of data centers in 4, 5, and 6 dimensions, our

"super computers" and the need for increasingly more precision, we are not able to extend the size of available data centers. For this reason, we chose to explore randomly generated scattered data centers.

3. Randomly Generated Points in d-Dimensions

In the following tables, we abandoned the mesh-based method that has been used historically in favor of a hybrid point enveloping and random d-dimensional "fuzzy" data scheme. There are 2^d points enveloping the d-dimensional hypercube; the number of such enveloping points is quite small compared to the number of randomly generated points.

With the mesh-based hypercube, the boundary us very crisp and is only one point in thickness. With the randomly generated d-dimensional points, the boundary is "fuzzy" in nature and has an arbitrary thickness that is "user-defined". So, for the distance, δ, from the surfaces generated by these points to the interior of randomly generated d-dimensional points will be considered the "fuzzy" boundary, $\partial\Omega$ containing a total of n_b points. The region contained within the "fuzzy" boundary is the interior, $\Omega\backslash\partial\Omega$ containing ni points, such that $N = n_b + n_i$.

The points $x \in \Re^d$ is checked to determine whether it belongs in a bin near an extremum or a zero. Approximately 15% of those points outside of the designated bins are deleted thereby reducing the total number of points and beneficially the condition number. Because random numbers of points were used, no nice patterns were discernible (Table 6).

Table 6. Summary of the randomly generated and culled points RMSE using parameters from the mesh-based results.

RMSE	Dim	Pts	β	s_k	$c^2_{\min}$	$c^2_{\max}$
0.2477	2	91	21.41	16.96	3.960	3.9632
7.96e−3	3	183	37.686	1.018	7.33e−3	1.8947
3.95e−3	4	534	30.64	1.018	4.87e03	2.546
1.48e−3	5	389	29.57	1.016	7,83e−3	2.273
8.97e−4	6	574	21.54	1.003	3.826e−2	1.739

4. Conclusion

Although we constructed a simple d-dimensional set of eigenvalue Poisson partial differential equations to solve, some interesting conclusions were uncovered: 1) Arbitrary precision arithmetic is very necessary because the use of large MQ exponents, β, and the range of variable shape parameters, may produce very ill-conditioned equation systems. 2) Meshed based methods have an exponential growth, n^d, where n is the number of points per dimension and d is the number of dimensions, whereas the scattered data approach requires n locations and only n^2 matrix elements reducing the strain on computer memory requirements as the dimen-

sionality grows. 3) The information gained from the inverse of the interior subdomain and boundary subdomain problems may help in populating regions near the extrema and zeros and avoid over-populating noncritical regions. 4) Variable MQ shape parameters were first implemented in [18] [19] based upon the observation these parameters were dependent upon the curvature of the solution. Recently in [20], a formal mathematical study showed this observation was true and formulated the shape parameter distribution on more theoretical foundation. 5) Neural network methods combined with the proposed algorithm presented here would be very useful in finding numerical solutions to a broader class of important elliptic, parabolic, hyperbolic partial differential equations as well as integral equations. 6) It was our intent not to present a rigorous analysis of the roles of the MQ exponent, shape parameter characteristic "knobs" and boundary shape parameter multipliers in reducing the errors between the known analytic and numerical solutions, but rather the randomly generated data centers in d-dimensions and increasing the population of points near the extrema and zeros of the known integrated forcing functions over the entire domain. This work is intended to interest more research to maximize the potential of this presented approach.

Acknowledgements

The authors want to thank Prof. Leevan Ling of Hong Kong SAR and Prof. Andreas Karageorghis of the University of Cyprus for their very helpful comments.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Bellman, R.E. and Rand Corporation (1957) Dynamic Programming. Princeton University Press.

[2] Naito, R. and Yamada, T. (2023) Deep High-Order Splitting Method for Semilinear Degenerate PDEs and Application to High-Dimensional Nonlinear Pricing Models. *Digital Finance*, **6**, 693-725. https://doi.org/10.1007/s42521-023-00091-z

[3] Yu, W. and Mascagni, M. (2023) Monte Carlo Methods for Partial Differential Equations with Applications to Electronic Design Automation. Department of Computer Science, Florida State University. https://doi.org/10.1007/978-981-19-3250-2.

[4] Kuo, F.Y. and Nuyens, D. (2016) Application of Quasi-Monte Carlo Methods to Elliptic PDEs with Random Diffusion Coefficients: A Survey of Analysis and Implementation. *Foundations of Computational Mathematics*, **16**, 1631-1696. https://doi.org/10.1007/s10208-016-9329-5

[5] Kim, H.H. and Yang, H.J. (2024) Domain Decomposition Algorithms for Neural Network Approximation of Partial Differential Equations. In: Dostál, Z., *et al.*, Eds., *Domain Decomposition Methods in Science and Engineering XXVII*, Springer, 27-37. https://doi.org/10.1007/978-3-031-50769-4_3

[6] Hu, Q., Basir, S. and Senocak, I. (2025) Non-Overlapping, Schwarz-Type Domain Decomposition Method for Physics and Equality Constrained Artificial Neural Networks. *Computer Methods in Applied Mechanics and Engineering*, **436**, Article ID:

117706. https://doi.org/10.1016/j.cma.2024.117706

[7] Huang, J., Guo, W. and Cheng, Y. (2023) Adaptive Sparse Grid Discontinuous Galerkin Method: Review and Software Implementation. *Communications on Applied Mathematics and Computation*, **6**, 501-532. https://doi.org/10.1007/s42967-023-00268-8

[8] Chen, W., Hwang, H. and Tsai, T. (2012) Maxima-Finding Algorithms for Multidimensional Samples: A Two-Phase Approach. *Computational Geometry*, **45**, 33-53. https://doi.org/10.1016/j.comgeo.2011.08.001

[9] Hu, Z., Shukla, K., Karniaakis, G.E. and Kawaguchi, K. (2023) Tackling the Curse of Dimensionality with Physics-Informed Neural Networks. *Neural Networks*, **176**, Article ID: 106368.

[10] Cohen, S.N., Jiang, D. and Sirignano, J. (2023) Curse of Dimensionality with Physics-Informed Neural Networks. *Journal of Machine Learning Research*, **24**, 1-49.

[11] Sarra, S.A. and Kansa, E.J. (2009) Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations. Computational Mechanics, Tech Science Press.

[12] Kansa, E.J. (2023) A Numerical Method for Solving Ill-Conditioned Equation Systems Arising from Radial Basis Functions. *American Journal of Computational Mathematics*, **13**, 356-370. https://doi.org/10.4236/ajcm.2023.132019

[13] Larsson, E. and Fornberg, B. (2005) Theoretical and Computational Aspects of Multivariate Interpolation with Increasingly Flat Radial Basis Functions. *Computers & Mathematics with Applications*, **49**, 103-130. https://doi.org/10.1016/j.camwa.2005.01.010

[14] Wertz, J., Kansa, E.J. and Ling, L. (2006) The Role of the Multiquadric Shape Parameters in Solving Elliptic Partial Differential Equations. *Computers & Mathematics with Applications*, **51**, 1335-1348. https://doi.org/10.1016/j.camwa.2006.04.009

[15] Buhmann, M.D. (2003) Radial Basis Functions. Cambridge University Press. https://doi.org/10.1017/cbo9780511543241

[16] Multiprecision Computing Toolbox for MATLAB.

[17] Kansa, E.J. and Holoborodko, P. (2020) Fully and Sparsely Supported Radial Basis Functions. *International Journal of Computational Methods and Experimental Measurements*, **8**, 208-219. https://doi.org/10.2495/cmem-v8-n3-208-219

[18] Kansa, E.J. (1990) 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**, 147-161. https://doi.org/10.1016/0898-1221(90)90271-k

[19] Kansa, E.J. and Carlson, R.E. (1992) Improved Accuracy of Multiquadric Interpolation Using Variable Shape Parameters. *Computers & Mathematics with Applications*, **24**, 99-120. https://doi.org/10.1016/0898-1221(92)90174-g

[20] Heidari, M., Mohammadi, M. and De Marchi, S. (2023) Curvature Based Characterization of Radial Basis Functions: Application to Interpolation. *Mathematical Modelling and Analysis*, **28**, 415-433. https://doi.org/10.3846/mma.2023.16897