

Advances in the greedy optimization algorithm for nodes and collocation points using the method of fundamental solutions

A. Doyle^a, A. Nelson^a, M. Yuan^a, N.J. DeMoes^b, B.D. Wilkins^c, K.E. Grubaugh^{d,*},
T.V. Hromadka II^a

^a Department of Mathematics, United States Military Academy, Building 601, West Point, NY 10996, United States

^b College of Computer Science, Northeastern University, 360 Huntington Avenue, Boston, MA 02115, United States

^c Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, United States

^d Hromadka & Associates, 29809 Santa Margarita Parkway, Suite 102, Rancho Santa Margarita, CA 92688, United States

ARTICLE INFO

Keywords:

Method of fundamental solutions
Boundary element method
Boundary value problem
Complex variable boundary element method

ABSTRACT

The Method of Fundamental Solutions and Boundary Element Method commonly involve development of approximation functions featuring linear combinations of basis functions defined over the problem domain and boundary. These basis functions are selected with respect to the governing partial differential equation (PDE) being approximated, and are customarily fundamental solutions of the PDE operator. Points where singularities occur are known as source points and are traditionally uniformly distributed outside the problem domain. In this paper, basis functions with singularities at source points are examined with interest in optimizing the placement of the corresponding nodes to reduce computational error while also reducing the number of nodes involved. A motivation for such an optimization is the reduction in matrix solver requirements in solving large dense matrix systems. An algorithm is explored that has been shown to provide such an optimization capability with a 3D case study in steady state heat transport. It is shown that by including the nodal position coordinates as additional variables to be optimized, the resulting approximation function is improved in computational accuracy by increasing the number of degrees of freedom to be optimized.

1. Introduction

In this paper, a new procedure for determining nodal point locations is presented. The new methodology establishes the set of nodal point locations as another and powerful set of variables that can be optimized similar to the other modeling variables. This enables more precision in the modeling results and a significant reduction in the number of nodal points used one the model. This later attribute is of high importance when assessing large matrix solutions where computational success depends on the dimension of the matrix system under analysis. The three-dimensional Laplace equation with Dirichlet boundary conditions is investigated in a simply connected problem domain and simply closed boundary. The boundary value problem (BVP) investigated is a three-dimensional global steady state heat source located at the origin. Other potential applications of this method are in ideal fluid flow and electrostatics. The problem domain examined for example purposes is a well-known computational model, the Stanford Bunny, located entirely within the first octant and shown in Fig. 1. The three-dimensional ap-

proximation function under investigation is

$$\hat{u}(x, y, z) = \sum_{j=1}^n c_j \frac{1}{R_j(P_j)}, \quad (1)$$

where the c_j 's are real constants determined by collocation and the R_j 's are the usual non-zero radial distance measures between the nodal points P_j , now part of the optimization effort, and the evaluation point P located at (x, y, z) . Thus, $R_j = \sqrt{(x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2}$. This mathematical formulation is a distinction from the other schema, in that rather than nodal point positions being defined by the end user preferentially, they are now being defined by optimization to reduce computational error. This paper uses radial basis functions to assess the placement of node locations and collocation point locations. Other basis functions may be used in the approximation as long as they satisfy the governing PDE.

There are a handful of different types of point definitions used in computational modeling. Since the goal is to computationally solve a

* Corresponding author.

E-mail addresses: aidan.doyle@usma.edu (A. Doyle), anton.nelson@usma.edu (A. Nelson), matthew.yuan@usma.edu (M. Yuan), grubaugh.kameron@aol.com (K.E. Grubaugh).

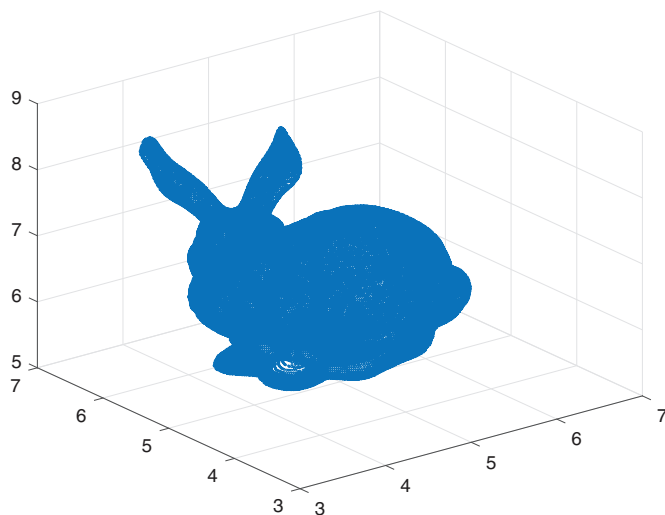


Fig. 1. Problem domain.

boundary value problem of a partial differential equation, then the geometry of the problem boundary is defined as part of the modeling description. Additionally, boundary conditions of a specified state variable (Dirichlet boundary conditions) or flux type boundary conditions are defined at various locations on the problem boundary. These locations where boundary conditions are specified are called “evaluation” points, or equivalent terms. The evaluation points selected to be targets for solving the computational equations are typically called “collocation” points, or an equivalent. Other types of points are utilized in some computational models, particularly for purposes of visualization of modeling outcomes. For example, computational fluid dynamic type models often use “probes” where computational outcomes are displayed for assessment and depiction of modeling results.

Typical node and collocation point placement is generally determined using one of two methodologies. The first inflates the problem boundary and then places nodes uniformly along the expanded boundary, as in [12]. The second is to place the nodes directly on the boundary uniformly. This paper examines the utility of using a “greedy” algorithm to place nodes and collocation points by comparing it against the two typical methodologies of placement. Nodal point and collocation point locations become additional variables within the approximation function to determine the optimal locations of each respectively. Since the approximation function is an entire function across the problem domain, the maximum error is evaluable and used as the metric for utility. This paper shows that the optimum solution determined using the greedy algorithm produces results with less error than traditional methods using the same number of basis functions.

2. Literature review

The method of fundamental solutions (MFS) [1] is a well-researched topic due to its wide range of applications in solving partial differential equations (PDEs) and the breadth of geometries suitable for the method. The MFS is a simple, mesh-free way to solve boundary value problems for PDEs. To use this method, a certain number of source points external to the problem geometry, and an equal number of corresponding collocation points on the problem boundary are selected. Then, a certain number of test points are chosen to construct an approximation [5]. The L_∞ error at the test points is dependent upon the calculated coefficients in the approximation function, which vary in relation to the placement of the source points and the collocation points. An abundance of previous research addresses the selection of source points and collocation points as well as algorithms to generate such distributions.

Alves (2009) attempts to create a space-efficient method of selecting source points and collocation points in his renowned paper on the selection of source points in the method of fundamental solutions. To accomplish this, Alves first considers the geometric center of the problem space and evenly distributes collocation points along the perimeter of the problem geometry. Alves then extends source points beyond the boundary in such a manner to be collinear with the collocation points and geometric center of the problem space. Many other algorithms apply a similar approach by first placing a circle or sphere, depending on the dimension of the problem, around the problem geometry and then evenly distributing source points and collocation points along it. These algorithms follow the assumption that the even distribution of points will allow discovery of the best combination of points. This family of algorithms attempts to achieve the best result given a limited number of nodes rather than minimize error. [2,9]

Chen et al. developed two other algorithms to optimize the placement of nodes in the MFS [10]. In the first, the algorithm searches for an optimal radius to inflate the problem boundary out to, and then places nodes on that optimal radial expansion. The second algorithm uses leave-one-out-cross-validation to compute the optimal boundary to place the node points. Chen’s Example 5 uses the same Stanford Bunny geometry used in this paper [10].

Novel to other research in the field [11], the algorithm presented in this paper exhaustively searches a grid to find the optimal placement for a certain number of nodes. The presented approach is not limited by the selection of the problem geometry and has no fixed geometry to the placement of nodes. The only condition is that collocation points remain on the boundary and source points are outside the domain.

Since the MFS is being used, the PDE must be in the form of Laplace’s equation, where the problem domain is simply connected and the boundary is simply closed. Thus, the problem becomes a Dirichlet problem, suitable for use of the MFS.

Our problem, which satisfies such conditions, is a Cartesian three-dimensional space which contains a single, ideal steady-state heat source point located at the origin, defined as (0, 0, 0). Steady state heat transport conditions are assumed throughout the space and the problem domain. Boundary conditions on the boundary, B , are defined by the global potential function $T(x, y, z, t = \infty) = \frac{U}{R}$ where U is a constant coefficient and R is the linear distance measured from the heat source point.

The fundamental solution of the governing Laplace equation used to create the approximation function is of the form $1/r$. Therefore, the basis functions are of the form C_j/R_j . Linear combinations of such basis functions also satisfy the Laplace equation. Therefore, the approximation function used in this paper will be of the form

$$\sum_{j=1}^n \frac{C_j}{R_j}$$

where n is the number of source/collocation points and R_j is the distance to each source point. The C_j values are determined by satisfying the boundary condition at collocation points specified on B .

The MFS is of importance in these computational efforts as it melds with the Boundary Element Methods (BEM), including the Complex Variable Boundary Element Method (CVBEM) for computationally solving boundary value problems of partial differential equations. Consequently, a focus on the MFS as conducted in this paper is similarly addressing the BEM and CVBEM modeling approaches.

The CVBEM first appeared in the *Journal of Numerical Methods in Engineering* in 1984 and has since become a modeling method used in many disciplines, such as the geosciences, ideal fluid flow, heat transfer, and other engineering disciplines [7,8]. Throughout the last thirty-five years, the CVBEM has continued to be the subject of research, leading to advancements in the uses and efficiency of the method. Of special interest among these advancements is the development of a multi-dimensional approach for this computational method, enabling the application to case studies on three dimensional objects [6].

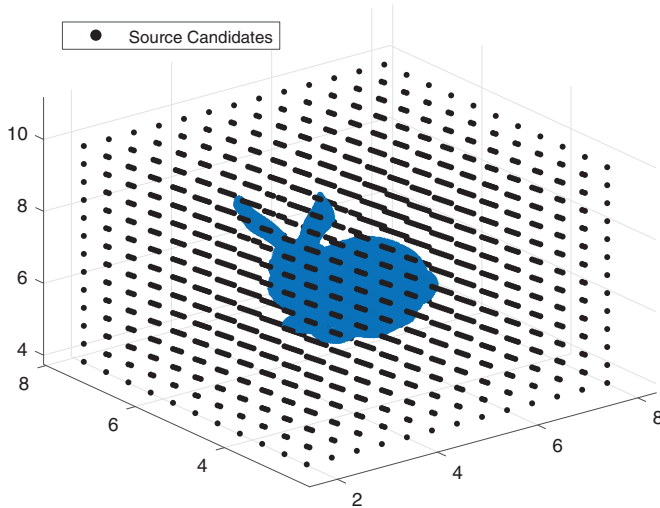


Fig. 2. Problem domain surrounded by pool of candidate node locations.

The CVBEM offers an analytic approach to solving BVPs. This method is unique because the result is a continuous approximation function throughout the entire problem domain without the use of interpolation [3]. The function contains both imaginary and real parts that satisfy the Laplace equation, representing streamlines and the potential function, respectively. The CVBEM approximation function will represent the potential at a location, z . The general equation for this function is

$$\hat{w}(z) = \sum_{j=1}^n c_j g_j(z)$$

where c_j is the j th complex coefficient, $g_j(z)$ is the j th function in the set of complex basis functions, and n is the number of functions used in the BVP.

The algorithm described in this paper has the advantage of being independent of geometries. To demonstrate this, we selected a complex geometry - the Stanford bunny. We used a simplified 34,834 node version of the well-known bunny centered at (2,2,2) to create a more computationally intense and realistic model, depicted in Fig. 1.

3. Computational candidate nodal points and collocation points

Unlike traditional methods where the node locations, collocation point locations, and pairs are pre-selected, the optimization method requires the evaluation of potential node and collocation point locations for accuracy. In order to incorporate node and collocation positions as degrees of freedom in the model, candidate pools for both are generated. We created a subspace in the first octant containing the problem domain (the Stanford Bunny), but offset from the origin and axes. In order to accomplish this, we translated the bunny data set off of the origin and constructed a cube with a user-specified distance between its faces and the extreme x,y and z coordinates of the bunny, shown in Fig. 1.

Using a grid of user-specified resolution, we discretized the cube subspace to form a pool of candidate nodes. Then, we removed any points contained in the subspace that were on or inside the convex hull of the bunny from the pool of candidate nodes using the In hull package [4]. Fig. 2 provides an illustration of the pool of candidate node positions.

From a data set of 34,834 points that composed the surface of the bunny, we utilized every 300th point to form a pool of candidate collocation points on the boundary. Fig. 3 depicts the candidate collocation points.

After creating the candidate locations for possible node and collocation selection, implementation of the algorithm evaluates the utility of candidate node pairing by measuring the summation of the computa-

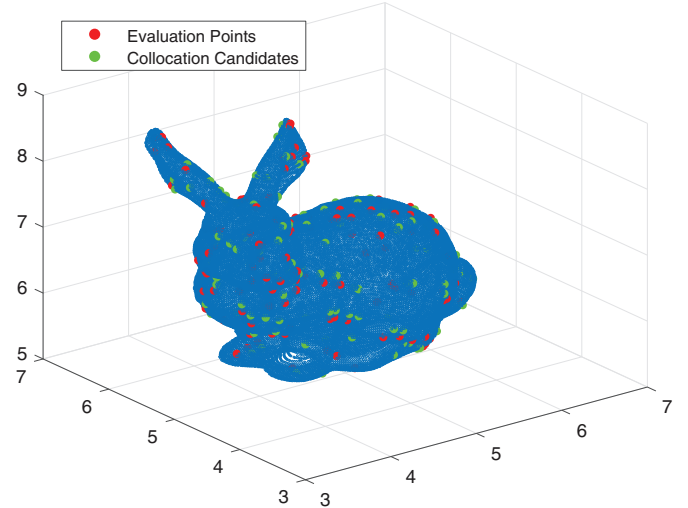


Fig. 3. Problem domain with evaluation points and collocation candidates depicted.

tional error at the evaluation points. Fig. 3 depicts the evaluation points on the problem boundary.

4. Description of computational algorithm

The computational algorithm described minimizes the error by selecting node-collocation point pairs with the least L_∞ error at the evaluation points.

Given a problem geometry, domain, and defined boundary conditions, the algorithm generates a candidate pool for possible nodal and collocation points, as described in the above section. The number of node-collocation point pairs in the model is also input.

The algorithm for error minimization consists of three steps:

- Step 1. Approximation function generation. After selecting one source point outside the problem domain and one collocation point on the problem boundary, the error of the approximation function is set at zero at the point on the boundary. Then, the coefficients to the approximation function are calculated.
 - Step 2. Test point error assessment. The algorithm then calculates the approximation function value and boundary function values at all of the evaluation points. Since the evaluation points are located on the problem boundary, the boundary condition functions are guaranteed to be defined. Then, the error is calculated by taking the absolute value of the difference between the two values.
 - Step 3: Collocation and source pair selection. After individual error is calculated for each evaluation point, the max error at each evaluation point is recorded. This method was used because it directly reflects the magnitude of local error at each test point, and, by minimizing this value, local maximum error is also guaranteed to decrease. After calculating the max error for each collocation-source point pair, the algorithm selects the pair with the lowest value and adds it to the approximation function. These points are eliminated from their respective candidate pools and included in any further approximations with more nodes.
- Steps 1–3 are repeated for each node used in an approximation. For each additional node calculated, the approximation function will have a term added to it.

For each iteration of steps 1–3, the coefficients of the approximation function are determined using *linsolve()*, which runs in $O(n^3)$ time. The L_∞ error is calculated for each evaluation point, which takes $O(C)$ time, with C being the number of evaluation points used. Finding the L_∞ norm also takes $O(C)$ time. Therefore, each node-collocation point pair has a

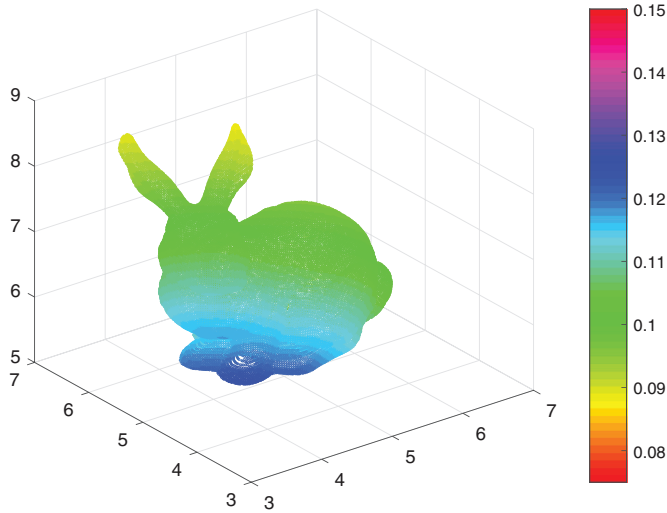


Fig. 4. True values of $\frac{1}{R}$ on the Stanford bunny.

run-time of $O(n^3)$. The process is repeated for each pair. Assuming there are a candidates each in the nodal and collocations point pool, there are $\binom{a}{2} = \frac{a(a-1)}{2}$ possible pairs to examine. Overall, this means that the run-time for each iteration is $O(n^3 a^2)$.

For an n -node model, there will be n iterations of steps 1–3. Though each iteration alters the approximation function and candidate pools, the impact on the theoretical run-time is minimal. Therefore, the overall run-time of the algorithm is $O(n^3 a^2)$. When the problem geometry is fixed, the collocation and nodal point pools are fixed and a becomes a constant. In this case, the run-time is $O(n^3)$.

5. Example problem: heat source at the origin

To demonstrate the algorithm, we examined the following heat source problem. A source emits heat uniformly and can be modeled by

$$u(x, y, z) = \frac{h}{\sqrt{(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2}} \quad (2)$$

where h is the constant heat source strength defined at source location (x_j, y_j, z_j) . For this problem, the heat source has strength $h = 1$ and is

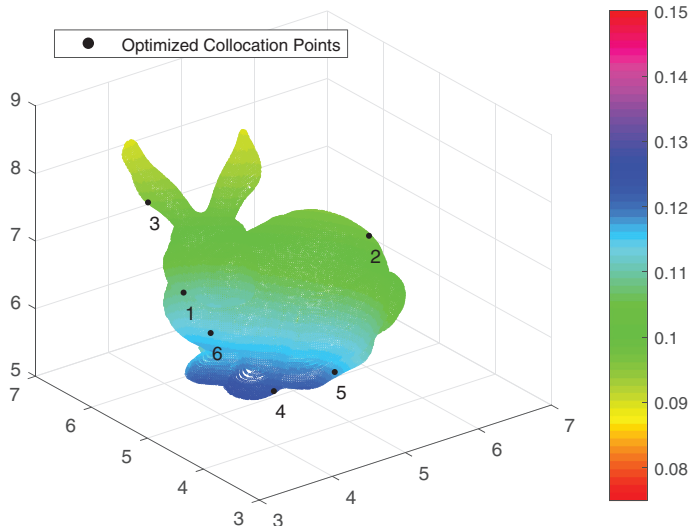


Table 1
Optimal nodes and collocation points for approximation with $n = \{1, 2, \dots, 6\}$.

n	Node	Collocation Point	L_∞ Error
1	(1.606, 2.324, 4.159)	(3.221, 4.633, 7.237)	0.01449
2	(7.606, 7.324, 7.660)	(5.629, 4.460, 7.311)	0.00538
3	(7.106, 6.824, 7.659)	(3.520, 5.658, 7.997)	0.00334
4	(2.606, 3.824, 5.159)	(4.085, 4.148, 5.698)	0.00222
5	(3.106, 3.324, 6.160)	(4.814, 4.010, 5.788)	0.00185
6	(3.106, 4.324, 5.660)	(4.028, 5.201, 6.095)	0.00161

located at the origin (0, 0, 0). Thus, the potential function is to be approximated by

$$u(x, y, z) = \frac{1}{\sqrt{x^2 + y^2 + z^2}}. \quad (3)$$

The example problem domain used to test the algorithm is the Stanford Bunny model, shifted and scaled into the first octant. The selected problem domain consisted of 34,834 points, yielding 117 evenly spaced points chosen as candidate collocation points and a different 117 points used as evaluation points. The pool of candidate source points was generated by finding the minimum and maximum x , y , and z coordinates on the boundary of the bunny and then drawing a box of a set distance away from the most extreme points and filling that box with a fine mesh. The resulting pool contained 1786 candidate sources. Fig. 4 displays the true value of the potential on the surface of the bunny.

The step-wise greedy algorithm minimized computational error. The maximum error decreased exponentially with each additional collocation-source pair, depicted in Fig. 5. The algorithm chose points that are not constrained to a sphere or expanded bunny geometry around the problem domain, which are the typical methods used to generate source point locations. MATLAB's `linsolve()` command did not encounter any singular or badly-conditioned matrices while in this particular problem.

With the node locations in Table 1 and the coefficients in Table 2, the approximation function can be described completely. For $n = 6$, the approximation function of f is:

$$\begin{aligned} f(x, y, z) &= \frac{1}{\sqrt{x^2 + y^2 + z^2}} \approx \hat{f}(x, y, z) \\ &= \frac{0.5069}{\sqrt{(x - 1.606)^2 + (y - 2.324)^2 + (z - 4.160)^2}} \end{aligned}$$

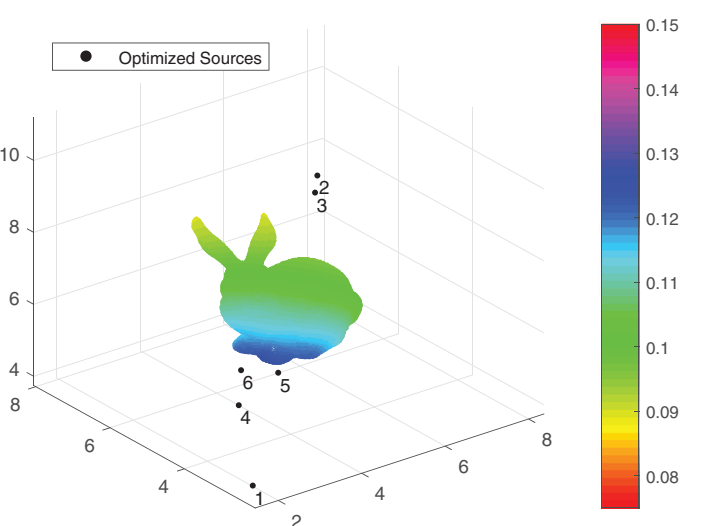


Fig. 5. (Left) Approximation of $\frac{1}{R}$ using 6 nodes. Collocation points are shown in black (Right) Source point locations chosen by the algorithm. Numbers indicate the iteration a source was selected.

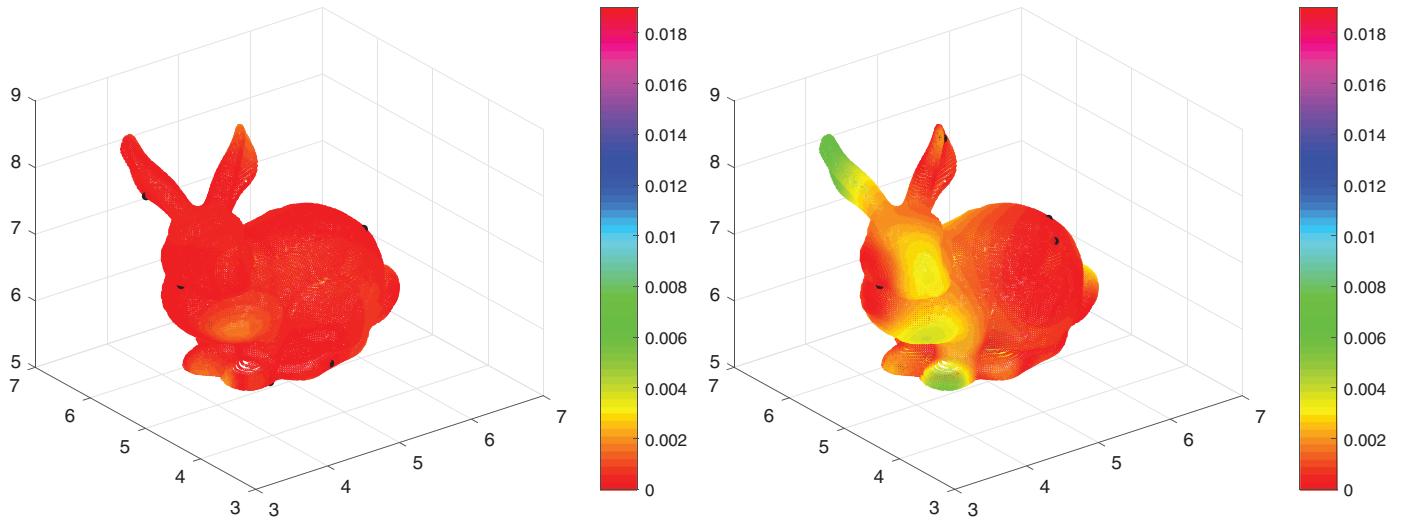


Fig. 6. (Left) L_∞ error of greedy approximation on the problem boundary. Since the solution satisfies the Laplace equation, the max error occurs on the boundary (Right) L_∞ error of approximation when points are placed on a sphere around the problem domain.

Table 2

Coefficients of corresponding node with $n = \{1, 2, \dots, 6\}$. Note that $C_i(n = j) \neq C_i(n = j + 1)$. Corresponding nodes in Table 1.

n	C_1	C_2	C_3	C_4	C_5	C_6
1	0.4547					
2	0.3652	0.1106				
3	0.3112	0.5840	-0.3528			
4	0.4028	0.5026	-0.3047	-0.0392		
5	0.4646	0.3946	-0.2332	-0.0458	-0.0118	
6	0.5069	0.3610	-0.2120	-0.0759	-0.0151	0.0107

$$\begin{aligned}
 & + \frac{0.3610}{\sqrt{(x-7.606)^2 + (y-7.324)^2 + (z-7.660)^2}} \\
 & + \frac{-0.2120}{\sqrt{(x-7.106)^2 + (y-6.824)^2 + (z-7.660)^2}} \\
 & + \frac{-0.0759}{\sqrt{(x-2.606)^2 + (y-3.824)^2 + (z-5.160)^2}} \\
 & + \frac{-0.0151}{\sqrt{(x-3.106)^2 + (y-3.324)^2 + (z-6.160)^2}} \\
 & + \frac{0.0107}{\sqrt{(x-3.106)^2 + (y-4.324)^2 + (z-5.660)^2}}
 \end{aligned} \quad (4)$$

6. Discussion of computational results

Previous research in MFS has typically placed the nodes in a fixed geometry around the problem domain. Usually the geometry is either an n -sphere or an expansion of the problem domain. This method outperforms the greedy method in terms of computational time ($O(n)$ compared to $O(n^3 a^2)$) but under-performs in terms of error reduction. To demonstrate this, we evaluated the same problem geometry and background function using a fixed sphere geometry around the bunny with $n = 6$. The surface error on the bunny using either method can be seen in Fig. 6. It is worth noting that since the background function and the approximation function are both potential functions, the maximum error is also a potential function.

The boundary L_∞ error is optimized in both methods when solving for the coefficients; however, only the greedy method spends the extra computational effort to optimize the locations of the nodes and collocation points.

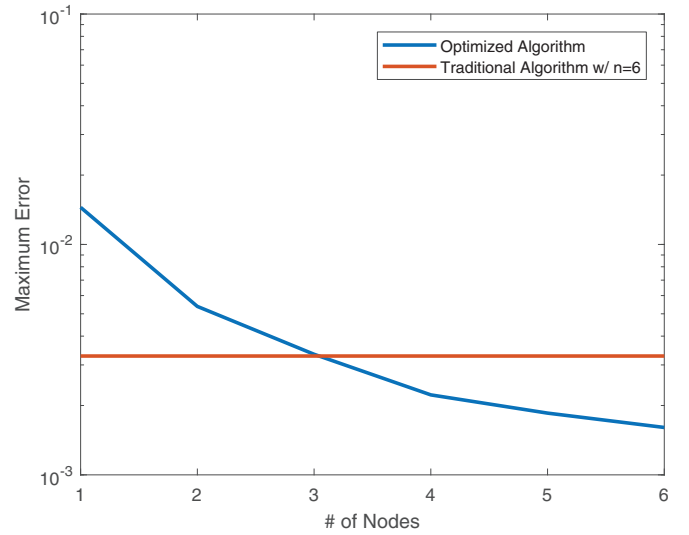


Fig. 7. Comparison of traditional algorithm with optimized algorithm.

tion points. This computational effort increases run-time for $n = 6$ from 2s to 80s (Dell Latitude, i3 core) when the source and collocation pools are 1786 points and 117 points respectively for the greedy method.

With the same source and collocation pools as above, the error as n increases from 1 to 6 can be seen in Fig. 7. The greedy algorithm has the same error at $n = 3$ as the traditional algorithm at $n = 6$.

Reducing the required number of nodes at the cost of increased run-time is attractive for certain kinds of problems. For real-world applications, this means reducing the amount of sensors (nodes) required to get a reading within a selected error tolerance. Computationally, matrix solvers for poorly conditioned problems are limited by the size of the matrix. Having fewer nodes, and thus a smaller matrix of coefficients, allows for better approximation of these poorly conditioned problems.

Finally, with the traditional fixed geometry, error stopping criteria is programmatically difficult. If the error is too high for a certain n , the fixed geometry must be changed to reduce the error. Stopping criteria for the greedy algorithm do not require any changing of the geometry; new sources can be added continuously until the error is below tolerance.

7. Topics for future research

The examined algorithm tests all candidate locations and results in a long run-time for problems with large candidate pools. Future research may focus on implementing a methodology that excludes candidate locations based on the maximum error associated with the candidate's one node approximation error. Exploration into using the gradient between potentials as rejection criteria could be informative as well, as could using the gradients of the approximation function to develop vector trajectories and associated streamlines. Additionally, follow-on lines of research pertaining directly to MFS node and collocation optimization could involve strategically designing both sets of points' candidate pools to further reduce run-time. Additionally, the resolution, or size, of the grids that compose the node and collocation point candidate pools have a significant effect on computation time for the model. Further investigation into the density of both pools, along with the distribution of nodes is warranted. It is worth noting that randomly scattering nodes with a uniform distribution may be a more computationally-efficient method of generating the candidate node pool. The increase in computational efficiency caused by optimizing node and collocation point positions allows for applying the CVBEM to a 3-dimensional problem. Another useful path for the optimization of C_j and P_j would be to employ the Hilbert space of the L_2 error.

Declaration of Competing Interest

None.

Acknowledgements

The authors would like to thank the faculty of the United States Military Academy for their technical support and valuable feedback in the

production of this manuscript, and to Dr. Hromadka for his continued support and personal funding on this research topic.

References

- [1] Alves CJ. On the choice of source points in the method of fundamental solutions. *Engineering Analysis with Boundary Elements* 2009;33(12):1348–61. doi:10.1016/j.enganabound.2009.05.007. Special Issue on the Method of Fundamental Solutions in honour of Professor Michael Golberg.
- [2] Dean T, Hromadka T, Kastner T, Phillips T. Modeling potential flow using Laurent series expansions and boundary elements. *Numer Methods Partial Differ Equ* 2010;28(2):573–86. doi:10.1002/num.20636.
- [3] DeMoes N, Bann G, Wilkins B, Hromadka T, Boucher R. 35 Years of advancements with the complex variables boundary element method. *Int J Comput Methods Exper Meas* 2019;7(1):1–13.
- [4] D'Errico J.. In hull. Mathworks File Exchange; 2012.
- [5] HON YC, LI M. A discrepancy principle for the source points location in using the mfs for solving the bhcp. *Int J Comput Methods* 2009;6(2):181–97. doi:10.1142/S0219876209001759.
- [6] Hromadka T. A multi-Dimensional complex variable boundary element method. WIT Press; 2002.
- [7] Hromadka T, Guymon G. A complex variable boundary element method: development. *Int J Numer Methods Eng* 1984;20:25–37.
- [8] Hromadka T, Whitley R. Approximate three-dimensional steady state potential flow problems using two-dimensional complex polynomials. *Eng Anal Bound Elem* 2005;29:190–4.
- [9] Lee S. The use of equivalent source method in computational acoustics. *J Comput Acoust* 2017;25(1):1–19.
- [10] S Chen C, Karageorghis A, Li Y. On choosing the location of the sources in the mfs. *Numer Algorithms* 2015;72.
- [11] Schaback R. Adaptive numerical solution of mfs systems. *Method Fundam Sol-A Meshless Method* 2007:1–27.
- [12] Young D, Chen K, Chen J, Kao J. A modified method of fundamental solutions with source on the boundary for solving laplace equations with circular and arbitrary domains. *Comput Model Eng Sci* 2007;19(3):197.