Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 2, 2026

GENERALIZATION OF THE NYSTRÖM METHOD FOR SOLVING HIGHER-DIMENSIONAL FREDHOLM INTEGRAL EQUATIONS AND ITS COMPUTATIONAL PERFORMANCE

Kainat Tahir*

*Comsats University Islamabad, Lahore Campus

*itskainatbajwa@gmail.com

DOI: https://doi.org/10.5281/zenodo.21467839

Keywords
the Nyström method, the multidimensional Fredholm integral equation, the tensor-product quadrature, the sparse grids, computational complexity, low rank approximations, and iterative solvers.

Article History
Received: 20 December 2025
Accepted: 04 February 2026
Published: 19 February 2026

Corresponding Author: *
Kainat Tahir

Abstract
This paper extends the Nyström method to the multidimensional domain of second kind Fredholm integral equations. The continuous integral operator over a domain $\Omega \subset \mathbb{R}^d$ is replaced by a multidimensional quadrature rule and collocation on the quadrature nodes yields a dense weighted-kernel system. Although low dimensional nodes can be constructed directly and accurately by Tensor-product quadrature, the number of nodes increases exponentially with d. The resulting matrix will be required to have quadratic size and must be solved using cubic time for a direct solution, creating a severe dimensional bottleneck. In order to overcome this shortcoming, the paper considers the problem of sparse-grid quadrature, dimension-adaptive rules, quasi-Monte Carlo sampling, matrix-free iterative solution and low-rank kernel compression. The error analysis method is developed to separate the quadrature consistency error and the amplification by the inverse discrete Fredholm operator, which are multidimensional. The computational study is addressed theoretically in terms of node numbers, memory usage, arithmetic complexity, convergence measures and through a reproducible benchmark design. The analysis indicates that tensor products are still competitive for smooth two-dimensional problems at moderate resolution; on the other hand, sparse grids and low-rank iterative methods become more and more relevant as the dimension and the accuracy requirements increase. The generalized framework is simple and easily implementable, and maintains the simplicity of the Nyström method while offering practical ways to control the curse of dimensionality.

1. Introduction
The multidimensional case of integral equations is found in potential theory, radiative transfer, elasticity, image reconstruction, transport problems and boundary-integral formulations. Second-kind forms are often more stable than first-kind forms (Atkinson, 1997; Jerri, 1985; Kress, 2014) since they are a mixture of the identity operator and the integral operator. The problem is for a bounded domain $\Omega \subset \mathbb{R}^d$.

$u(x) - \lambda \int_\Omega K(x,y)u(y)dy = f(x), \quad x \in \Omega \subset R^d. \quad (1)$

u is unknown, f is given, K is the kernel and λ is a scalar parameter. The classical Nyström method involves quadrature and collocation at quadrature points (Nyström, 1930; Atkinson, 1974). In any finite dimension, the choice and the number of quadrature nodes will be critical in deciding the

quality of the approximation. A full tensor grid has $N=n^d$ points in each of the coordinate directions. Thus, for a small rise in d the size of the matrix grows rapidly, as do memory requirements, evaluations of the kernels, times for factorization. In this paper, the goal is to develop the higher dimensional Nyström formulation, to compare quadrature constructions, to compute the scaling of the computations, and to identify efficient solution strategies. Special emphasis is placed on sparse grids, dimension-adaptive quadrature, matrix-free iteration and low rank kernel structure. The following method is able to capture the basic Nyström discretization and mitigate the effect of dimensional growth on the practical implementation.

2. Multidimensional Fredholm operator

Define T:

$$C(\Omega)\to C(\Omega) \quad (1)$$

by

$$(Tu)(x)=\int_\Omega K(x,y)u(y)dy. \quad (2)$$

Equation (1) is $(I-\lambda T)u=f$. T is a compact operator on the standard function spaces if K is continuous on $\Omega\times\Omega$ and a unique solution exists when $1/\lambda$ is not a characteristic value of T (Atkinson, 1997; Kress, 2014). Ω can be rectangular, simplicial, curved or broken into several parts. Quadrature shall take that geometry into account; a transformation away from a reference domain includes the Jacobian determinant.

$$\int_\Omega q(y)dy = \int_{\hat{\Omega}} q(F(\xi))\,|\det JF(\xi)|\,d\xi. \quad (3)$$

This mapped form allows the use of standard quadrature nodes in a reference cube or simplex for use in physical elements. In disconnected regions or regions with complex geometry, the global integral is computed by summing the element integrals.

3. Generalized Nyström discretization

Assume that $(y_j,w_j)_{j=1}^N$ is a multidimensional quadrature rule on Ω:

$$\int_\Omega q(y)dy \approx \sum_{j=1}^N w_j q(y_j). \quad (4)$$

The continuous extension of Nyström is

$$\tilde{u}_N(x)=f(x)+\lambda\sum_{j=1}^N w_j K(x,y_j)\tilde{u}_j. \quad (5)$$

At the point $x = y_i$, the value of collocation is equal to

$$\sum_{j=1}^N[\delta_{ij}-\lambda w_j K(y_i,y_j)]\tilde{u}_j=f(y_i), \quad i=1,\dots,N. \quad (6)$$

With $K_N=[K(y_i,y_j)]$, $W=\mathrm{diag}(w_1,\dots,w_N)$, $A=I-\lambda K_N W$, and $f_N=(f(y_1),\dots,f(y_N))^T$, the system is

$$A\tilde{u}=f_N. \quad (7)$$

The formula is the same as the 1D formula, except that nodes, weights and kernel evaluations are multidimensional. Each of the weights scales a column of K_N in the y_j column of integration points. From Equation (5) the approximate solution at an arbitrary x can be obtained.

4. Multidimensional quadrature strategies

4.1 Tensor-product quadrature

In coordinate k, pick a rule with one dimension, n_k nodes, $\Omega=\prod_{k=1}^d[a_k,b_k]$. Both multi-index nodes and weights are

$$y_i=(y_{i1}^{(1)},\dots,y_{id}^{(d)}), \quad w_i=\prod_{k=1}^d w_{ik}^{(k)}. \quad (8)$$

$$N=\prod_{k=1}^d n_k; \quad \text{if } n_k=n, \text{then } N=n^d. \quad (9)$$

Constructing tensor Gauss–Legendre rules is simple and they are very accurate when the integrand is smooth (Davis & Rabinowitz, 1984; Gautschi, 2004; Golub & Welsch, 1969). The drawback of their limitation is the number of nodes is exponential. As a result, Tensor products are more suitable when d=2 or d=3, n is still small, domains are still separable, and the kernels are of high precision.

4.2 Sparse-grid quadrature

The sparse grid method of Smolyak (1963), Bungartz & Griebel (2004) and Zenger (1991) are sparse grids based on tensor products of one dimensional difference rules and discard a lot of high level tensor combinations that are of minor significance for functions with bounded mixed derivatives. A level-L sparse operator can be represented in a schematic diagram as:

$$A_{L,d} = \sum_{|i|_1\le L+d-1}(\Delta i_1\otimes\cdots\otimes\Delta i_d). \quad (10)$$

Sparse-grid node counts grow as roughly $O(2^L L^{d-1})$ as opposed to $O(2^{Ld})$ in a similar full tensor construction, for a fixed d. Sparse grids work very well when a mixture of derivatives are smooth. Discontinuities, strongly anisotropic features, and high dimension can be disadvantages for their use; in this case it is better to use dimension adaptive index selection (Gerstner & Griebel, 2003; Novak & Ritter, 1999).

4.3 Quasi-Monte Carlo and unstructured domains

Randomized quasi-Monte Carlo methods can also practically achieve faster convergence for sufficiently regular integrands (Dick et al., 2013); Monte Carlo integration has an error of order $O(N^{-1'2})$ which does not depend heavily on d. These techniques do not require the growth of tensors and may prove helpful in higher dimensions. But they do not converge in a classical way and their stochastic error analysis makes a deterministic Nyström analysis difficult. In the case of simplices and curved elements, the natural rules of cubature or mapped elements are frequently more convenient than tensor rules based on cubes.

Table 1. Quadrature options for higher-dimensional Nyström discretization.

Quadrature strategy	Typical node behavior	Strength	Limitation
Full tensor product	$N=n^d$	High accuracy and simple construction for smooth low-dimensional problems	Exponential growth with dimension
Smolyak sparse grid	Approximately $O(2^L L^{d-1})$	Reduces nodes for bounded mixed regularity	Complex indexing; less effective for nonsmooth data
Dimension-adaptive sparse grid	Adds influential multi-indices	Responds to anisotropy	Requires reliable contribution indicators
Quasi-Monte Carlo	User-selected N; weak dimension dependence	Useful when d is moderately large	Convergence depends on effective dimension and regularity
Elementwise cubature	Sum of local element nodes	Handles complex geometry	Mesh generation and interface bookkeeping

5. Computational implementation

5.1 Assembly algorithm

1. Define Ω, its geometric mapping or element decomposition, λ, K(x,y), and f(x).
2. Generate multidimensional quadrature nodes y_j and weights w_j using a tensor, sparse-grid, quasi-Monte Carlo, or elementwise rule.
3. Evaluate fN and assemble kernel entries $K_{ij}=K(y_i,y_j)$.
4. Scale column j by w_j and form $A_{ij}=\delta_{ij}-\lambda w_j K_{ij}$.
5. Estimate conditioning and solve $A\tilde{u}=fN$ using a direct or iterative method.
6. Evaluate the multidimensional Nyström extension on an independent validation set.
7. Refine the quadrature level and compare errors, residuals, timing, memory, and observed convergence.

5.2 Matrix-free and low-rank solution

A dense direct method stores $O(N^2)$ values and requires $O(N^3)$ operations. Matrix-free Krylov methods avoid storing A when the product KN(Wv) can be evaluated on demand (Saad, 2003). Without acceleration, one matrix-vector product still costs $O(N^2)$, but it reduces memory and may be effective when convergence requires relatively few iterations. Preconditioning is important when λ is near a characteristic value or the kernel produces clustered eigenvalues.

Many smooth or asymptotically smooth kernels admit low-rank block approximations. Hierarchical matrices, adaptive cross approximation, and related compression methods can reduce storage and matrix-vector work toward O(N log N) or O(Nr), where r is an effective numerical rank (Bebendorf, 2008; Hackbusch, 2015). Related Nyström sampling methods also demonstrate the computational value of low-rank kernel structure (Gittens & Mahoney, 2016; Williams & Seeger, 2001). Separable kernels provide the clearest case:

$$K(x,y) \approx \sum_{r=1}^{R} a_r(x) b_r(y). \quad (11)$$

Then KNWv can be evaluated in O(RN) work instead of $O(N^2)$. Low-rank approximation and

sparse-grid quadrature address different bottlenecks: sparse grids reduce N, while compression reduces the cost associated with the remaining N nodes.

6. Computational performance analysis

For n points in every coordinate and $N=n^d$ total nodes, the main dense costs are

$$Kernel\ evaluations: O(N^2) = O(n^{2d}), \quad (12)$$

$$Memory: O(N^2) = O(n^{2d}), \quad (13)$$

$$Direct\ factorization: O(N^3) = O(n^{3d}). \quad (14)$$

These expressions quantify the curse of dimensionality. With n=10, the node counts are 100, 1,000, 10,000, and 100,000 for d=2,3,4,5. The corresponding dense matrices contain 10^4, 10^6, 10^8, and 10^{10} entries. Assuming eight-byte real values, matrix storage is approximately 0.08 MB, 8 MB, 0.8 GB, and 80 GB before solver workspace. Thus, a direct tensor-product approach can become impractical between four and five dimensions even at a modest per-coordinate resolution.

Table 2. Dimensional scaling for a full tensor grid with ten points per coordinate.

Dimension d	Nodes $N=10^d$	Dense entries N^2	Approx. matrix storage	Direct-solve implication
2	100	10,000	0.08 MB	Small dense solve
3	1,000	1,000,000	8 MB	Moderate dense solve
4	10,000	100,000,000	0.8 GB	Large memory and factorization cost
5	100,000	10,000,000,000	80 GB	Direct dense method generally impractical

Performance should be judged against the achieved accuracy, not only node count, because a sparse grid can show up with fewer nodes but still pay with larger constants or a sort of reduced accuracy for nonsmooth integrands. Also a low-rank solver might cut execution time, yet it can smuggle in compression error, so the "fast" part can be a bit misleading. In general, the recommended measures are pretty broad: total nodes, kernel evaluations, assembly time, solution time, peak memory, iteration count, residual, condition estimate, and independent set solution error.

7. Error and convergence analysis

Let TN denote the quadrature operator. Subtraction of the continuous and discrete equations yields

$$(I - \lambda TN)(u - \tilde{u}N) = \lambda(T - TN)u. \quad (15)$$

When the inverse exists,

$$\|u - \tilde{u}N\|\infty \leq |\lambda|\|(I - \lambda TN)^{-1}\|\infty \|(T - TN)u\|\infty. \quad (16)$$

The first factor after $|\lambda|$ measures stability, while the last factor is the multidimensional quadrature consistency error. High quadrature accuracy alone is insufficient when the discrete resolvent is large. The algebraic residual must also be separated from discretization error:

$$Rrel = \|fN - A\tilde{u}\|\infty / max(1, \|fN\|\infty). \quad (17)$$

For a known exact solution, evaluate

$$Etest = maxx \in G|u(x) - \tilde{u}N(x)|, \quad (18)$$

where G is an independent multidimensional validation set. Tensor quadrature error depends on coordinate-wise regularity. Sparse-grid convergence depends strongly on bounded mixed derivatives, and quasi-Monte Carlo accuracy depends on variation and effective dimension (Bungartz & Griebel, 2004; Dick et al., 2013). Therefore, a performance claim must identify the assumed regularity class and the quadrature family.

8. Reproducible numerical test design

A controlled computational study can use manufactured solutions. Choose Ω, K, λ, and an exact u, then calculate

$$f(x) = u(x) - \lambda \int \Omega K(x,y)u(y)dy. \quad (19)$$

Recommended tests include a separable smooth kernel, a nonseparable analytic kernel, an anisotropic kernel, and a localized feature. In the

separable scenario, you're basically checking low-rank acceleration; the nonseparable one gives a more general baseline. Then anisotropic measures dimension-adaptive refinement ,and the localized case is meant to examine elementwise, or adaptive cubature, depending on what you're doing. Experiments should run in d=2,3,4 and where it is practical, go beyond that to higher dimensions.

Each run should state the quadrature level, the exact node count, the domain mapping, arithmetic precision, matrix construction strategy, the solver ,the tolerance, hardware, and whether kernel values were cached. Tensor methods and sparse-grid methods should be compared at similar error levels. Use the same discrete system for both direct and iterative solvers, and report compression tolerances separately from quadrature tolerances and linear-solver tolerances.

9. Discussion

The higher-dimensional Nyström method is kind of like a plain extension of the one-dimensional scheme, but in practice the computational behavior changes a lot. The integral turns into a multidimensional cubature problem, and the dense matrix "size" is basically the cubature node count. Tensor products are rather transparent, highly accurate, and not too hard to verify, so they work as a solid baseline for two- and three-dimensional cases. But their exponential scaling blocks them from being a universal remedy.

Sparse grids cut down the number of nodes when the integrand has enough mixed smoothness. The dimension-adaptive versions, they often help a lot when only some coordinate interactions matter, sort of selectively. Quasi-Monte Carlo methods can be a fallback when the dimension is higher, and when the deterministic tensor structure is just not a good fit. Also at the algebra level, you really need iterative solvers and kernel compression, because once dense factorization is the main cost, things get expensive fast. The best implementation usually mixes a node-reduction approach with a matrix-acceleration strategy, rather than leaning on only one of them .

Geometry also gets a say in the performance. Hyperrectangles make it easier because you can use direct tensor rules and sparse-grid rules, yet curved or irregular domains usually force mappings or element decompositions. Jacobian factors have to be built into the weights, and the validation points should span the physical domain, not only the reference geometry. These small but important choices are what keep the generalized method accurate, and also reproducible, across setups.

10. Conclusion

The Nyström method has been generalized to second-kind Fredholm integral equations, on regions in $\mathbb{R}^d$. Multidimensional quadrature gives you a collection of weighted nodes, collocation sets up a rather dense kernel system and then the Nyström extension helps recompose the solution over essentially the whole domain. The straightforward tensor-product formulation is fairly accurate and practical when the dimension is low but it runs into exponential node growth, quadratic memory demand, and also cubic cost if you solve the system directly. Sparse grids, dimension-adaptive quadrature, quasi-Monte Carlo rules, matrix-free Krylov solvers, and low-rank kernel compression are all complementary techniques that can reduce the overall computational burden. For error analysis you really have to split things into quadrature consistency, stability amplification, algebraic residuals, and compression error, otherwise it gets murky. A dependable performance evaluation should report accuracy, node count, kernel evaluations, time, memory, conditioning, and how the solver behaves in practice. Overall, these choices make the generalized Nyström method a workable framework for multidimensional integral equations, while also pinning down the regimes where dimensional growth is still the dominant limitation.

REFERENCES

Atkinson, K. E. (1974). Iterative variants of the Nyström method for the numerical solution of integral equations. Numerische Mathematik, 22, 17–31.

Atkinson, K. E. (1997). The numerical solution of integral equations of the second kind. Cambridge University Press.

Bebendorf, M. (2008). Hierarchical matrices: A means to efficiently solve elliptic boundary value problems. Springer.

Bungartz, H.-J., & Griebel, M. (2004). Sparse grids. Acta Numerica, 13, 147–269.

Davis, P. J., & Rabinowitz, P. (1984). Methods of numerical integration (2nd ed.). Academic Press.

Dick, J., Kuo, F. Y., & Sloan, I. H. (2013). High-dimensional integration: The quasi-Monte Carlo way. Acta Numerica, 22, 133–288.

Gautschi, W. (2004). Orthogonal polynomials: Computation and approximation. Oxford University Press.

Gerstner, T., & Griebel, M. (2003). Dimension-adaptive tensor-product quadrature. Computing, 71, 65–87.

Gittens, A., & Mahoney, M. W. (2016). Revisiting the Nyström method for improved large-scale machine learning. Journal of Machine Learning Research, 17(117), 1–65.

Golub, G. H., & Van Loan, C. F. (2013). Matrix computations (4th ed.). Johns Hopkins University Press.

Golub, G. H., & Welsch, J. H. (1969). Calculation of Gauss quadrature rules. Mathematics of Computation, 23(106), 221–230.

Hackbusch, W. (2015). Hierarchical matrices: Algorithms and analysis. Springer.

Jerri, A. J. (1985). Introduction to integral equations with applications. Marcel Dekker.

Kress, R. (2014). Linear integral equations (3rd ed.). Springer.

Novak, E., & Ritter, K. (1999). High dimensional integration of smooth functions over cubes. Numerische Mathematik, 75, 79–97.

Nyström, E. J. (1930). Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. Acta Mathematica, 54, 185–204.

Saad, Y. (2003). Iterative methods for sparse linear systems (2nd ed.). SIAM.

Smolyak, S. A. (1963). Quadrature and interpolation formulas for tensor products of certain classes of functions. Soviet Mathematics Doklady, 4, 240–243.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), Advances in neural information processing systems 13 (pp. 682–688). MIT Press.

Zenger, C. (1991). Sparse grids. In W. Hackbusch (Ed.), Parallel algorithms for partial differential equations (pp. 241–251). Vieweg.