

Algorithms for Numerical Analysis in High Dimensions

Gregory Beylkin, Martin Mohlenkamp

▶ **To cite this version:**

Gregory Beylkin, Martin Mohlenkamp. Algorithms for Numerical Analysis in High Dimensions. SIAM Journal on Scientific Computing, 2005, 26 (6), pp.2133-2159. 10.1137/040604959 . hal-02076682

HAL Id: hal-02076682
https://hal.science/hal-02076682

Submitted on 22 Mar 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ALGORITHMS FOR NUMERICAL ANALYSIS IN HIGH DIMENSIONS

GREGORY BEYLKIN† AND MARTIN J. MOHLENKAMP‡

Abstract. Nearly every numerical analysis algorithm has computational complexity that scales exponentially in the underlying physical dimension. The separated representation, introduced previously, allows many operations to be performed with scaling that is formally linear in the dimension. In this paper we further develop this representation by

(i) discussing the variety of mechanisms that allow it to be surprisingly efficient;

(ii) addressing the issue of conditioning;

(iii) presenting algorithms for solving linear systems within this framework; and

(iv) demonstrating methods for dealing with antisymmetric functions, as arise in the multiparticle Schrödinger equation in quantum mechanics.

Numerical examples are given.

Key words. curse of dimensionality, multidimensional function, multidimensional operator, algorithms in high dimensions, separation of variables, separated representation, alternating least squares, separation-rank reduction, separated solutions of linear systems, multiparticle Schrödinger equation, antisymmetric functions

AMS subject classifications. 65D15, 41A29, 41A63, 65Y20, 65Z05, 81-08

DOI. 10.1137/040604959

1. Introduction.

The computational complexity of most algorithms in dimension d grows exponentially in d. Even simply accessing a vector in dimension d requires M^d operations, where M is the number of entries in each direction. This effect has been dubbed the *curse of dimensionality* [1, p. 94], and it is the single greatest impediment to computing in higher dimensions. In [3] we introduced a strategy for performing numerical computations in high dimensions with greatly reduced cost, while maintaining the desired accuracy. In the present work, we extend and develop these techniques in a number of ways. In particular, we address the issue of conditioning, describe how to solve linear systems, and show how to deal with antisymmetric functions. We present numerical examples for each of these algorithms.

A number of problems in high-dimensional spaces have been addressed by the usual technique of separation of variables. Given an equation in d dimensions, one can try to approximate its solution f by a separable function as

$$f(x_1, \ldots, x_d) \approx \phi_1(x_1) \cdots \phi_d(x_d)\,. \tag{1.1}$$

This form allows one to approximate f with complexity growing only linearly in d and compute using only one-dimensional operations, thus avoiding the exponential dependence on d (see, e.g., [35]). However, if the best approximate solution of the

†Department of Applied Mathematics, University of Colorado at Boulder, 526 UCB, Boulder, CO 80309-0526 (beylkin@colorado.edu). The research of this author was supported by Department of Energy grant DE-FG02-03ER25583.

‡Department of Mathematics, Ohio University, 321 Morton Hall, Athens, OH 45701 (mjm@math.ohiou.edu). The research of this author was supported by National Science Foundation grants DMS-9902365 and DMS-215228.

form (1.1) is not good enough, then there is no way within this framework to improve the accuracy.

We use the natural extension of separation of variables

$$f(x_1, \ldots, x_d) = \sum_{l=1}^{r} s_l \phi_1^l(x_1) \cdots \phi_d^l(x_d) + \mathcal{O}(\epsilon)\,, \tag{1.2}$$

which we call a *separated representation*. We set an accuracy goal ϵ first, and then adapt $\{\phi_i^l(x_i)\}$, $\{s_l\}$, and r to achieve this goal with minimal separation rank r. The separated representation seems rather simple and familiar, but it actually has a surprisingly rich structure and is not well understood. It is not a projection onto a subspace, but rather a nonlinear method to track a function in a high-dimensional space while using a small number of parameters. In section 2 we develop the separated representation, extending the results in [3] and making connections with other results in the literature. We introduce the concept of the condition number of a separated representation, which measures the potential loss of significant digits due to cancellation errors. We provide analysis and examples to illustrate the structure of this representation, with particular emphasis on the variety of mechanisms that allow it to be surprisingly efficient. Note, however, that the theory is still far from complete.

Many linear algebra operations can be performed while keeping all objects in the form (1.2). We can then perform operations in d dimensions using combinations of one-dimensional operations, and so achieve computational complexity that formally scales *linearly* in d. Of course, the complexity also depends on the separation rank r. The optimal separation rank for a specific function or operator is a theoretical question, and is considered in section 2. The practical question is how to keep the separation rank close to optimal during the course of some numerical algorithm. As we shall see, the output of an operation, such as matrix-vector multiplication, is likely to have larger separation rank than necessary. If we do not control the separation rank, it will continue to grow with each operation. In section 3 we present an algorithm for reducing the separation rank back toward the optimal, and we also present a modified algorithm that avoids ill-conditioned representations. Although the modification required is very simple, it makes the overall algorithm significantly more robust.

In order to use the separated representation for numerical analysis applications, many algorithms and operations need to be translated into this framework. Basic linear algebra operations, such as matrix-vector multiplication, are straightforward and were described in [3], but other operations are not as simple. In section 4 we continue to expand the set of operations that can be performed within this framework by showing how to solve a linear system. Many standard methods (e.g., Gaussian elimination) do not make sense in the separated representation. We take two approaches to solving a linear system. First, we discuss how to use iterative methods designed for large sparse matrices, such as conjugate gradient. Second, we present an algorithm that formulates the system as a least-squares problem, combines it with the least-squares problem used to find a representation with low separation rank, and then solves this joint problem by methods similar to those in section 3. We also discuss how these two general strategies can be applied to problems other than solving a linear system.

One of our target applications is the representation and computation of wavefunctions of the multiparticle Schrödinger equation in quantum mechanics. These wavefunctions have the additional constraint that they must be antisymmetric under exchange of variables, a condition that seems to preclude having low separation rank. In section 5 we present the theory and algorithms for representing and computing

with such antisymmetric functions. We construct an antisymmetric separation-rank reduction algorithm, which uses a pseudonorm that is only nonzero for antisymmetric functions. This algorithm allows us to guide an iterative method, such as the power method, to converge to the desired antisymmetric solution.

We conclude the paper in section 6 by briefly describing further steps needed for the development of this methodology.

2. The separated representation. In this section we introduce the separated representation and discuss its properties. In order to emphasize the underlying physical dimension, we define operators and functions in d dimensions. To avoid confusion between, e.g., a "vector in two dimensions" and a "matrix," we clarify our notation and nomenclature. A *function f in dimension d* is a map $f : \mathbf{R}^d \to \mathbf{R}$ from d-dimensional Euclidean space to the real numbers. We write f as $f(x_1, \ldots, x_d)$, where $x_i \in \mathbf{R}$. A *vector $\mathbf{F}$ in dimension d* is a discrete representation of a function in dimension d on a rectangular domain. We write it as $\mathbf{F} = F(j_1, \ldots, j_d)$, where $j_i = 1, \ldots, M_i$. A linear *operator $\mathcal{A}$ in dimension d* is a linear map $\mathcal{A} : S \to S$, where S is a space of functions in dimension d. A *matrix $\mathbb{A}$ in dimension d* is a discrete representation of a linear operator in dimension d. We write $\mathbb{A} = A(j_1, j_1'; \ldots; j_d, j_d')$, where $j_i = 1, \ldots, M_i$ and $j_i' = 1, \ldots, M_i'$. For simplicity we assume $M_i' = M_i = M$ for all i.

DEFINITION 2.1 (separated representation of a vector). *For a given ϵ, we represent a vector $\mathbf{F} = F(j_1, j_2, \ldots, j_d)$ in dimension d as*

$$\sum_{l=1}^{r} s_l F_1^l(j_1) F_2^l(j_2) \cdots F_d^l(j_d) \equiv \sum_{l=1}^{r} s_l \mathbf{F}_1^l \otimes \mathbf{F}_2^l \otimes \cdots \otimes \mathbf{F}_d^l \,, \tag{2.1}$$

where s_l is a scalar, $s_1 \geq \cdots \geq s_r > 0$, and $\mathbf{F}_i^l$ are vectors in dimension 1 with entries $F_i^l(j_i)$ and unit norm. We require the error to be less than ϵ:

$$\left\| \mathbf{F} - \sum_{l=1}^{r} s_l \mathbf{F}_1^l \otimes \mathbf{F}_2^l \otimes \cdots \otimes \mathbf{F}_d^l \right\| \leq \epsilon \,. \tag{2.2}$$

We call the scalars s_l separation values *and the integer r the* separation rank.

The definition for a matrix is similar, with the matrices $\mathbb{A}_i^l = A_i^l(j_i, j_i')$ replacing the vectors $\mathbf{F}_i^l = F_i^l(j_i)$. For matrices it would be preferable to use the l^2 operator norm to measure the approximation error (2.2), but the operator norm may take too long to compute. Since we will also need to treat the matrices as vectors and compute inner products in section 3, we will use the Frobenius norm for matrices.

Classical loss of significance (loss of precision) occurs when we have numbers L and a with $L \gg a$ and attempt to compute a by subtracting $(L+a)-L$. Since $L \gg a$, when $(L + a)$ is formed in finite arithmetic, significant digits of a are lost. In the extreme case, $(L+a)$ is rounded to L, and a is lost completely. In the separated representation, similar loss of significance occurs when the summands in (2.1) become much larger than $\mathbf{F}$ itself. By our normalization convention, we have $\|s_l \mathbf{F}_1^l \otimes \mathbf{F}_2^l \otimes \cdots \otimes \mathbf{F}_d^l\| = s_l$. In the case when the summands are orthogonal, we have $\|\mathbf{F}\| = (\sum_{l=1}^{r} s_l^2)^{1/2}$, so the l^2 norm of the separation values provides a convenient way to measure and control the loss of precision.

DEFINITION 2.2 (condition number of a separated representation). *The condition number of* (2.1) *is the ratio*

$$\kappa = \frac{(\sum_{l=1}^{r} s_l^2)^{1/2}}{\|\mathbf{F}\|} \,. \tag{2.3}$$

In order to maintain significant digits when using the representation (2.1), we cannot allow κ to be too large. In particular, to achieve (2.2) numerically it suffices to have

$$\kappa\mu\|\mathbf{F}\| \leq \epsilon\,, \tag{2.4}$$

where μ is the machine roundoff (e.g., 10^{-16}).

The main point of the separated representation is that since we only operate on one-dimensional objects, the computational complexities are formally linear in d rather than exponential. The key, however, is to determine any hidden dependency of r on d. We demonstrated in [3], and discuss further in this paper, that in many physically significant problems the separation rank depends benignly on d, so that near-linear complexities can be achieved in practice. We show next how to do the basic operations of addition, inner product, and matrix-vector multiplication. Other basic operations such as scalar multiplication, trace, Frobenius norm, matrix-matrix multiplication, etc. follow a similar pattern. The following statements can be easily verified.

PROPOSITION 2.3 (basic linear algebra).

Vector representation cost: *The separated representation of a vector requires $\mathcal{O}(d \cdot r \cdot M)$ entries to store.*

Vector addition $\hat{\mathbf{F}} + \tilde{\mathbf{F}}$: *Vectors in the separated representation can be added by merging their representations as sums and resorting. There is no appreciable computational cost, but the new separation rank is nominally the sum of the two separation ranks: $r = \hat{r} + \tilde{r}$.*

Vector inner product $\langle \hat{\mathbf{F}}, \tilde{\mathbf{F}} \rangle$: *In the separated representation, we can compute the inner product of two vectors via*

$$\langle \hat{\mathbf{F}}, \tilde{\mathbf{F}} \rangle = \sum_{\hat{l}=1}^{\hat{r}} \sum_{\tilde{l}=1}^{\tilde{r}} \hat{s}_{\hat{l}} \tilde{s}_{\tilde{l}} \langle \hat{\mathbf{F}}_1^{\hat{l}}, \tilde{\mathbf{F}}_1^{\tilde{l}} \rangle \cdots \langle \hat{\mathbf{F}}_d^{\hat{l}}, \tilde{\mathbf{F}}_d^{\tilde{l}} \rangle \tag{2.5}$$

in $\mathcal{O}(d \cdot \hat{r} \cdot \tilde{r} \cdot M)$ operations.

Matrix-vector multiplication $\mathbb{A}\mathbf{F}$: *In the separated representation, multiplication can be done via*

$$\mathbf{G} = \mathbb{A}\mathbf{F} = \sum_{\hat{l}=1}^{r_{\mathbb{A}}} \sum_{\tilde{l}=1}^{r_{\mathbf{F}}} s_{\hat{l}}^{\mathbb{A}} s_{\tilde{l}}^{\mathbf{F}} (\mathbb{A}_1^{\hat{l}} \mathbf{F}_1^{\tilde{l}}) \otimes \cdots \otimes (\mathbb{A}_d^{\hat{l}} \mathbf{F}_d^{\tilde{l}})\,, \tag{2.6}$$

plus renormalizations and sorting. The computational cost is $\mathcal{O}(d \cdot r_{\mathbb{A}} \cdot r_{\mathbf{F}} \cdot M^2)$ and the resulting separation rank is nominally $r_{\mathbf{G}} = r_{\mathbb{A}} r_{\mathbf{F}}$.

In all these operations we see a common pattern, that linearity and separability allow us to use only one-dimensional operations, and so the computational cost is linear in d. In the addition and multiplication operations we see another common feature: the separation rank of the output is greater than the separation rank of the input. We will address this issue in section 3 by introducing an algorithm to reduce the separation rank.

Remark. It is useful to combine the separated representation with other techniques for reducing the computational complexity. Spatial or frequency partitioning can be used to obtain a sparse structure, generally at the cost of increasing the separation rank. The component vectors $\mathbf{F}_i^l$ could be sparse, or in the matrix case $\mathbb{A}_i^l$ could be banded, sparse, or it could have a more complicated structure such as partitioned SVD [2, 25, 22, 45, 24]. Such techniques will be essential for the efficient implementation of the separated representation for many problems, but they are not important for the purposes of this paper.

2.1. Analysis and examples. In dimension $d = 2$, the separated representation (2.1) of a vector $F(j_1, j_2)$ reduces to a form similar to the SVD (see, e.g., [17]) of $\mathbf{F}$ considered as a matrix in dimension one. In fact, we can construct an optimal representation by using an ordinary SVD algorithm, and then truncating. Since we do not have any orthogonality constraints on $\mathbf{F}_i^l$, the representation (2.1) is more general than the SVD, and we no longer have the uniqueness properties of the SVD. We call r the separation rank in analogy with the operator rank that the ordinary SVD produces. For the matrix $A(j_1, j_1'; j_2, j_2')$, we can again construct an optimal separated representation using an ordinary SVD *algorithm*, as was done in [44]. The separated representation does not correspond to the singular value *decomposition* of this matrix, however, because we use a different pairing of indices. Instead of separating the input coordinates (j_1', j_2') from the output coordinates (j_1, j_2), we separate the direction (j_1, j_1') from the direction (j_2, j_2'). Matrices that have full rank as operators may still have low separation rank. For example, the identity is trivially represented as $\mathcal{I}_1 \otimes \mathcal{I}_2$, with separation rank one.

When $d > 2$, Definition 2.1 differs substantially from the SVD and should be considered on its own. Although our initial intuition is based on the SVD, essentially *none* of its theoretical properties hold. In particular, while the SVD is defined without reference to ϵ, the separated representation requires ϵ for its very definition. While there are several algorithms to compute the SVD, there are none with proven convergence to compute a representation with optimal separation rank. For our purposes, we need to have small separation rank, but do not need it to be optimal (see section 3). We note that the case $\epsilon = 0$ appears to be a difficult problem (see, e.g., [36, p. 158]).

Representations very similar to (1.2) have been studied for more than thirty years for statistical applications, under the names of "canonical decomposition" and "parallel factors analysis." We refer the reader to [14, 27, 28, 13, 12] and the references therein. Since the problems considered for statistics have as input a dense d-dimensional data cube, their applications have been limited to small dimensions ($d \ll 10$, mostly $d = 3$). As it was intended for a different purpose, the numerical issues that we are considering here were not addressed.

Other representations have been proposed that are more restrictive than (1.2), but are easier to compute and/or have more easily studied properties. The *higher order SVD* of [14] (also compare to [36, p. 57]), for example, is a generalization of the SVD that replaces the singular values (corresponding to s_l) by a d-dimensional matrix, but as a consequence still has computational complexity that depends exponentially on d. Configuration interaction (CI) methods (see, e.g., [39]) use a representation that looks like (1.2), but with the additional constraint that all the vectors $\mathbf{F}_i^l$ are selected from a master orthonormal set, and, thus, if we pick any two of them, they will either be identical or orthogonal. This constraint greatly simplifies many calculations, but may require a much larger number of terms. For example, in our approach we can have a two-term representation

$$\begin{aligned} &\phi_1(x_1)\phi_2(x_2)\cdots\phi_d(x_d) \\ &+ c[\phi_1(x_1) + \phi_{d+1}(x_1)][\phi_2(x_2) + \phi_{d+2}(x_2)]\cdots[\phi_d(x_d) + \phi_{2d}(x_d)]\,, \end{aligned} \tag{2.7}$$

where $\{\phi_j\}_{j=1}^{2d}$ form an orthonormal set. To represent (2.7) while requiring all factors to come from a master orthogonal set would force us to multiply out the second term and thus obtain a representation with 2^d terms. Nonorthogonal CI methods (see, e.g., [37, 30]) appear to give results in between these two extremes.

2.1.1. Example: Sine of a sum. We next consider an elegant example that illustrates several phenomena we have observed in separated representations.

One early numerical test of the separated representation was to consider a sine wave in the diagonal direction, $\sin(x_1 + \cdots + x_d)$, and attempt to represent it in the separated form, using only real functions. We can use the usual trigonometric formulas for sums of angles to obtain a separated representation, but then we will have $r = 2^{d-1}$ terms. For example,

$$\begin{aligned}\sin(x_1 + x_2 + x_3) &= \sin(x_1)\cos(x_2)\cos(x_3) + \cos(x_1)\cos(x_2)\sin(x_3) \\ &\quad + \cos(x_1)\sin(x_2)\cos(x_3) - \sin(x_1)\sin(x_2)\sin(x_3).\end{aligned} \tag{2.8}$$

Our numerical algorithm, however, found a representation with only d terms, which led us to a new trigonometric identity in d dimensions.

LEMMA 2.4.

$$\sin\left(\sum_{j=1}^{d} x_j\right) = \sum_{j=1}^{d} \sin(x_j) \prod_{k=1,k\neq j}^{d} \frac{\sin(x_k + \alpha_k - \alpha_j)}{\sin(\alpha_k - \alpha_j)} \tag{2.9}$$

for all choices of $\{\alpha_j\}$ *such that* $\sin(\alpha_k - \alpha_j) \neq 0$ *for all* $j \neq k$.

A proof of this identity and some of its generalizations can be found in [32], along with a discussion of its relationship to the work of Calogero [10, 11] and Milne's identity [31, 7].

This example illustrates the following.

- The obvious analytic separated representation may be woefully inefficient, and thus we should be careful relying on our initial intuition.
- The separated representation has an *essential* nonuniqueness. Although uniqueness can occur under special circumstances, (2.9) demonstrates that it is not natural to demand it.
- There can be ill-conditioned representations (as, e.g., $\alpha_1 \to \alpha_2$), even when well-conditioned representations are readily available.

2.1.2. Example: Finite differences in an auxiliary parameter. We next consider an example of a construction for operators that are a sum of one-directional operators (i.e., $\sum_i^d \mathcal{A}_i$), which, like the Laplacian

$$\Delta = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \cdots + \frac{\partial^2}{\partial x_d^2}, \tag{2.10}$$

trivially have separation rank d.

Our numerical experiments uncovered a construction by which such operators can actually be represented with far fewer terms. This construction is based on approximating the limit

$$\sum_{i}^{d} \mathcal{A}_i = \lim_{h\to 0} \frac{1}{2h} \left(\bigotimes_{i=1}^{d} (\mathcal{I}_i + h\mathcal{A}_i) - \bigotimes_{i=1}^{d} (\mathcal{I}_i - h\mathcal{A}_i) \right), \tag{2.11}$$

which we recognize as a derivative. Under the assumption that $\mathcal{A}_i$ is bounded, for any given ϵ, we can choose h sufficiently small and obtain a separation-rank two representation. Unbounded operators like the Laplacian must first be modified or

restricted to make them bounded. The condition number (Definition 2.2) of (2.11), however, is $\kappa = \mathcal{O}(1/h)$, and such a representation may be unusable numerically. Accounting for conditioning, we obtain the following theorem.

THEOREM 2.5. *Let $\mathcal{A}_i$ be a fixed, bounded operator $\mathcal{A}$ acting in the direction i, $\epsilon > 0$ be the error bound, and $0 < \mu \ll 1$ be the machine unit roundoff. Assuming $\mu d\|\mathcal{A}\| < \epsilon$, we can represent $\sum_i^d \mathcal{A}_i$ to within ϵ in the operator norm with separation rank*

$$r = \mathcal{O}\left(\frac{\log(d\|\mathcal{A}\|/\epsilon)}{\log(1/\mu) - \log(d\|\mathcal{A}\|/\epsilon)}\right). \tag{2.12}$$

Proof. Consider the auxiliary operator-valued function of the real variable t,

$$\mathcal{G}(t) = \|\mathcal{A}\| \bigotimes_{i=1}^{d} \left(\mathbb{I}_i + t\frac{\mathcal{A}_i}{\|\mathcal{A}\|}\right), \tag{2.13}$$

and note that $\mathcal{G}'(0) = \sum_i^d \mathcal{A}_i$. Using an appropriate finite difference formula of order r, we approximate

$$\mathcal{G}'(0) \approx \sum_{j=1}^{r} \alpha_j \mathcal{G}(t_j) \equiv \|\mathcal{A}\| \sum_{j=1}^{r} \alpha_j \bigotimes_{i=1}^{d} \left(\mathbb{I}_i + t_j \frac{\mathcal{A}_i}{\|\mathcal{A}\|}\right), \tag{2.14}$$

thus providing a separation-rank r approximation. If we choose equispaced t_j with stepsize h, then the truncation error of this finite difference can be made proportional to $(h^r/r!)\mathcal{G}^{(r+1)}(\xi)$, where $|\xi| \le h$ (see, e.g., [23]). Pulling out the norm $\|\mathcal{A}\|$ as we did in (2.13) allows us to choose $h = \alpha/d$ for some $\alpha < 1$ and bound the truncation error by $d\|\mathcal{A}\|\alpha^r$. The error due to finite precision arithmetic and loss of significance is proportional to $\mu\|\mathcal{A}\|/h = \mu d\|\mathcal{A}\|/\alpha$. Adding these two errors and choosing $\alpha = (\mu d/r)^{1/(r+1)}$ yields the bound $d\|\mathcal{A}\|\mu^{r/(r+1)}$. Setting this equal to ϵ and solving for r, we obtain (2.12). ∎

The estimate (2.12) implies that, as long as $d\|\mathcal{A}\|/\epsilon \ll 1/\mu$, the separation rank is $\mathcal{O}(\log(d\|\mathcal{A}\|/\epsilon))$.

This example illustrates that

- low separation-rank can sometimes be achieved at the expense of (reasonably) increasing the condition number. For problems in high dimensions it is an excellent trade-off.

2.1.3. Example: Exponential expansions using quadratures. We next consider an example of a methodology for constructing separated representations, built upon the exponential. The exponential function converts sums into products via $e^{a_1+a_2+\cdots+a_d} = e^{a_1}e^{a_2}\cdots e^{a_d}$, valid as long as a_i commute. In this section a_i will be a function or operator in the direction i, such as x_i^2 or $\partial^2/\partial x_i^2$.

Suppose we wish to find a separated representation for the radial function $f(\|\mathbf{x}\|)$ supported on the ball of radius 1 in dimension d. Since physical forces often depend on the distance between interacting objects, this case is of practical interest (see, e.g., [19, 20]).

We first consider a one-dimensional function $f(y)$, and assume that we can approximate f on the interval $0 \le y \le 1$ by a sum of Gaussians, such that for some σ_l, τ_l, and r

$$\left| f(y) - \sum_{l=1}^{r} \sigma_l e^{\tau_l y^2} \right| < \epsilon |f(y)|. \tag{2.15}$$

By substituting $\sum_i^d x_i^2$ for y^2 and using the properties of exponentials, we obtain a separated representation for $f(\|\mathbf{x}\|)$ on the ball. In this case we obtain a pointwise relative error bound instead of (2.2). We thus have reduced the problem of finding a separated representation for a radial function in dimension d to the one-dimensional approximation problem (2.15). Usually the minimal r to satisfy (2.15) is not the optimal separation rank for $f(\|\mathbf{x}\|)$, but it does provide an excellent upper bound.

The approximation problem (2.15) is addressed in [5] by extending methods in [4]. For certain choices of $f(y)$ there is a systematic way to approximate it with small r, even if the function has a singularity at zero. For example, let us consider $f(y) = y^{-\alpha}$ for $\alpha > 0$.

LEMMA 2.6 (see [5]). *For any $\alpha > 0$, $0 < \delta < 1$, and $0 < \epsilon \leq \min\{\frac{1}{2}, \frac{8}{\alpha}\}$ there exist positive numbers τ_l and σ_l such that for all $\delta \leq y \leq 1$*

$$\left| \frac{1}{y^\alpha} - \sum_{l=1}^{r} \sigma_l e^{-\tau_l y^2} \right| \leq \frac{\epsilon}{y^\alpha}, \tag{2.16}$$

with

$$r = \log \epsilon^{-1} \left[c_0 + c_1 \log \epsilon^{-1} + c_2 \log \delta^{-1} \right], \tag{2.17}$$

where c_0, c_1, and c_2 are constants that depend only on α. For fixed power α and accuracy ϵ, we thus have $r = \mathcal{O}(\log \delta^{-1})$.

The construction uses the integral representation

$$\frac{1}{y^\alpha} = \frac{2}{\Gamma(\alpha/2)} \int_{-\infty}^{\infty} e^{-y^2 e^{2t} + \alpha t} \, dt, \tag{2.18}$$

as described in [19, 20]. For a given accuracy, the rapid decay of the integrand restricts integration to a finite interval. Using the trapezoidal rule for an appropriately selected interval and stepsize yields a separated representation (see [19, 20]). The resulting representation is then optimized further using the results in [5].

This approach shows that

- there is a general, semianalytic method (based on representations with exponentials) to compute separated representations of radial functions.

These representations for radial functions can be used to construct representations for functions of operators. In particular, we can substitute an operator of the form $\sum_i^d \mathcal{A}_i$, such as the Laplacian, for y^2 in (2.15), and obtain a separated representation for $f((\sum_i^d \mathcal{A}_i)^{1/2})$, valid on a portion of its spectrum. Using Lemma 2.6 with $\alpha = 2$ and scaling to an appropriate interval, we can obtain a separated representation for the inverse Laplacian.

LEMMA 2.7. *There exists a separated representation for Δ^{-1}, valid on the annulus $(\sum_{i=1}^d \xi_i^2)^{1/2} \in [\delta D, D]$ in Fourier space, with separation rank* (2.17).

By applying the Fourier transform, we can obtain the corresponding Green's function

$$\frac{1}{(d-2)\Omega_d} \frac{1}{\|x\|^{d-2}} \leftrightarrow \sum_{l=1}^{r} \sigma_l \bigotimes_{i=1}^{d} \frac{1}{\sqrt{4\pi\tau_l}} \exp(-x_i^2/4\tau_l), \tag{2.19}$$

where Ω_d is the surface area of the unit sphere in dimension d.

Notice that the bound in Lemma 2.7 is independent of d. For periodic problems it is more natural to construct a representation on a cube rather than a ball. In dimension d the cube inscribed in the unit ball has side length $2d^{-1/2}$. If we compensate

for this effect to maintain a cube of fixed side length, then the separation rank grows as $\mathcal{O}(\log(d))$.

Robert Harrison (personal communication, 2003) pointed out to us an example that produces a separated representation for a multiparticle Green's function without going through a Fourier-space construction. Following [26], we consider the multiparticle operator $\mathcal{I} - \sum_{i=1}^{d} \Delta_i$, where Δ_i is the Laplacian in dimension three. The Green's function for this operator is

$$G(\mathbf{x}) = \frac{1}{(2\pi)^{3d/2}} \frac{K_{3d/2-1}(\|\mathbf{x}\|)}{\|\mathbf{x}\|^{3d/2-1}}, \tag{2.20}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, $x_i \in \mathbb{R}^3$, and K is the modified Bessel function. Using its integral representation [18, equation (8.432.6)]

$$K_\nu(y) = \frac{1}{2} \left(\frac{y}{2}\right)^\nu \int_0^\infty t^{-\nu-1} e^{-t-y^2/4t} \, dt, \tag{2.21}$$

we have

$$G(\mathbf{x}) = \pi^{3d/2} \int_0^\infty t^{-3d/2} e^{-t-\|\mathbf{x}\|^2/4t} \, dt, \tag{2.22}$$

and changing variables $t = e^{-2s}$, we obtain

$$G(\mathbf{x}) = 2\pi^{3d/2} \int_{-\infty}^\infty e^{-e^{-2s}+(3d-2)s} e^{-\|\mathbf{x}\|^2 e^{2s}/4} \, ds. \tag{2.23}$$

The integral (2.23) is similar to that in (2.18) and, using a similar approach to that in Lemma 2.6, we obtain a separated representation for the Green's function $G(\mathbf{x})$ with any desired accuracy.

These representations illustrate that

- there is a practical way to compute separated representations of Green's functions, and in important cases low separation rank can be achieved.

This set of examples also used another property of separated representations that we should point out explicitly:

- separation rank is invariant under separable changes of coordinates, for example the Fourier transform.

2.1.4. Classes of functions. Let us turn our consideration from specific, physically meaningful operators to general functions in high dimensions. Traditional approaches to characterizing a wide class of low-complexity functions (using smoothness and decay of derivatives) have not been productive so far. For example, translating the results in [41] into our context shows that for functions in the class W_2^k, which is characterized using partial derivatives of order k, there is a separated representation with separation rank r that has error

$$\epsilon = \mathcal{O}(r^{-kd/(d-1)}). \tag{2.24}$$

However, a careful analysis of the proof of Theorem 4.1 in [41] shows that the "constant" in the $\mathcal{O}(\cdot)$ is at least $(d!)^{2k}$, and that the inductive argument can only run if $r \geq d!$. Thus this result, while technically correct, does not provide a useful class of functions with low separation rank.

The class of *complete asymptotically smooth* functions is considered in [16, 43, 42]. Theorem 1 in [43] says essentially that error $\epsilon = \mathcal{O}(2^{-k})$ can be achieved with $r = \mathcal{O}(k^{d-1})$, with constants that depend on d. The separation rank is still exponential in d. We also note that $r = M^{d-1}$ is always achievable by applying the ordinary SVD recursively, so this result is really giving information about the "resolution" in each direction M rather than r.

The *sparse grid* approach, introduced in [46] and reviewed in [9], considers function classes that allow hierarchically arranged efficient grids in high dimensions. Each grid point corresponds to a separable basis function, so this construction yields a separated representation with separation rank equal to the number of grid points. For certain classes of functions, the sparse grid approach can reduce the number of grid points from the naive $\mathcal{O}(M^d)$ to $\mathcal{O}(C^d M(\log M)^{d-1})$. This reduction improves computational times, but the complexity still grows exponentially with d.

Currently we do not have any characterization of functions with low separation rank. The examples above, however, show that there are surprising mechanisms that allow low separation rank, and thus lack of a complete theory should not delay use of this representation. Two of the constructions described were discovered through numerical experimentation and we expect there are several other mechanisms that we have not yet encountered. In section 5 we discuss the case of antisymmetric functions, where the nominal separation rank is very large, but a weak formulation allows us to use a low separation-rank representation and retain the ability to compute with the true function. We believe that a weak approach such as this may be the key to a characterization of functions with low separation rank.

3. Reducing the separation rank. As discussed above, linear algebra operations such as addition and multiplication produce a vector in the separated representation, but with larger separation rank. In order to prevent uncontrolled growth of the separation rank, we need to find another separated representation for the same vector, but with reduced separation rank. In this section we discuss our algorithms for reducing the separation rank, while maintaining the prescribed accuracy. We assume that $\mathbf{G}$ is given to us in the separated representation

$$\mathbf{G} = \sum_{l=1}^{r_{\mathbf{G}}} s_l^{\mathbf{G}} \mathbf{G}_1^l \otimes \mathbf{G}_2^l \otimes \cdots \otimes \mathbf{G}_d^l, \tag{3.1}$$

but with larger separation rank $r_{\mathbf{G}}$ than necessary for its representation. We note that to reduce the separation rank of a matrix $\mathbb{A}$, we simply treat the component matrices $\mathbb{A}_i^l$ as vectors using the Frobenius inner product and norm.

Although it would be useful to obtain a representation with the optimal separation rank, after years of study of similar objects [27, 28, 14, 29, 12], there is no known algorithm that guarantees optimality. However, since we are only interested in the total computational cost, we do not need the optimal solution. In fact, a nearly optimal representation will have similar computational cost, and is much easier to construct. Even when $d = 2$, and a solution with optimal separation rank can be obtained through the singular value decomposition, we find it much more efficient overall to use a faster algorithm that produces a suboptimal solution (this algorithm has been used in [6]). In higher dimensions a suboptimal representation is actually *preferred* when it has better conditioning (see Definition 2.2). Using the algorithm described below, we uncovered the trigonometric identity (2.9) and the derivative formulation (2.14), and produced the numerical results in section 3.3.

We start with $\mathbf{F}$, an initial approximation of $\mathbf{G}$ in (3.1),

$$\mathbf{F} = \sum_{\tilde{l}=1}^{r_{\mathbf{F}}} s_{\tilde{l}}^{\mathbf{F}} \mathbf{F}_1^{\tilde{l}} \otimes \mathbf{F}_2^{\tilde{l}} \otimes \cdots \otimes \mathbf{F}_d^{\tilde{l}}. \tag{3.2}$$

If we have no other information, then we start with a random vector with $r_{\mathbf{F}} = 1$. If we are performing an iteration such as the power method, then we use the previous iterate as our starting guess. We then call the core algorithm described in section 3.1, and it improves the approximation $\mathbf{F}$ without changing $r_{\mathbf{F}}$. We exit the entire separation-rank reduction successfully if $\|\mathbf{F} - \mathbf{G}\| < \epsilon$. If not, we either call the core routine again and repeat the process, or decide that $r_{\mathbf{F}}$ is not sufficient. It is easy to prove that $\|\mathbf{F} - \mathbf{G}\|$ can never increase, so in general it decreases at each step. If the relative decrease becomes too small, then we assume $r_{\mathbf{F}}$ is not sufficient, increase $r_{\mathbf{F}}$ by one by appending another (random) term to $\mathbf{F}$, and repeat the process again. The threshold at which we increase $r_{\mathbf{F}}$ can strongly affect the speed of the algorithm; currently we select it via experimentation. In outline form this algorithm is as follows:

- Loop while $\|\mathbf{F} - \mathbf{G}\| > \epsilon$:
  - Call the algorithm in section 3.1 once.
  - If the relative change in $\|\mathbf{F} - \mathbf{G}\|$ is too small, then increase $r_{\mathbf{F}}$.

3.1. Alternating least squares. For a given separation rank $r_{\mathbf{F}}$, the best separated representation is that which minimizes the error $\|\mathbf{F}-\mathbf{G}\|$ subject to the condition number constraint (2.4). We first describe an algorithm that ignores the constraint on κ (Definition 2.2), and then modify it in section 3.2 to take κ into account. The minimization of $\|\mathbf{F} - \mathbf{G}\|$ is a nonlinear least-squares problem. To make this problem tractable, we exploit its multilinearity and use an alternating least-squares approach. The alternating least-squares algorithm (without the condition number constraint) is used in statistics for a similar purpose (e.g., [21, 28, 29, 8, 15, 40]). In our case the input is a vector in the separated representation, rather than a dense data vector in dimension d, and, thus, we can consider much larger values of M and d.

The alternating least-squares algorithm takes the initial approximation $\mathbf{F}$ in (3.2) and then iteratively refines it. We refine only in one direction k at a time, and then "alternate" (loop through) the directions $k = 1, \ldots, d$. To refine in the direction k, fix the vectors in the other directions $\{\mathbf{F}_i^{\tilde{l}}\}_{i \neq k}$, and then solve for new $\mathbf{F}_k^{\tilde{l}}$ (and $s_{\tilde{l}}^{\mathbf{F}}$) to minimize $\|\mathbf{F} - \mathbf{G}\|$. It is easy to show that $\|\mathbf{F} - \mathbf{G}\|$ never increases, and so usually decreases at each step. In outline form this algorithm is

- Loop $k = 1, \ldots, d$:
  - Fix $\{\mathbf{F}_i^{\tilde{l}}\}_{i \neq k}$ and solve for $\mathbf{F}_k^{\tilde{l}}$ (and $s_{\tilde{l}}^{\mathbf{F}}$) to minimize $\|\mathbf{F} - \mathbf{G}\|$.

The point of this alternating approach is that each refinement is a *linear* least-squares problem, which can be solved with standard linear algebra techniques (see, e.g., [17]). By taking the gradient of $\|\mathbf{F} - \mathbf{G}\|$ with respect to the vector entries in direction k and setting it equal to zero, one obtains a linear system (the normal equations) to solve for the updated vector entries. In our case, the system naturally divides into separate systems for each coordinate, with the same matrix. For fixed k, we form the matrix $\mathbb{B}$ with entries

$$B(\hat{l}, \tilde{l}) = \prod_{i \neq k} \langle \mathbf{F}_i^{\tilde{l}}, \mathbf{F}_i^{\hat{l}} \rangle. \tag{3.3}$$

Then for a fixed coordinate j_k, form the vector $\mathbf{b}_{j_k}$ with entries

$$b_{j_k}(\hat{l}) = \sum_{l=1}^{r_{\mathbf{G}}} s_l^{\mathbf{G}} G_k^l(j_k) \prod_{i \neq k} \langle \mathbf{G}_i^l, \mathbf{F}_i^{\hat{l}} \rangle. \tag{3.4}$$

The normal equations for the direction k and coordinate j_k become

$$\mathbb{B}\mathbf{c}_{j_k} = \mathbf{b}_{j_k}, \tag{3.5}$$

which we solve for $\mathbf{c}_{j_k} = c_{j_k}(\tilde{l})$ as a vector in $\tilde{l}$. After computing $c_{j_k}(\tilde{l})$ for all j_k, we let $s_{\tilde{l}}^{\mathbf{F}} = \|c_{j_k}(\tilde{l})\|$ and $F_k^{\tilde{l}}(j_k) = c_{j_k}(\tilde{l})/s_{\tilde{l}}^{\mathbf{F}}$, where the norm is taken with respect to the coordinate j_k.

For fixed k and j_k, it requires $r_{\mathbf{F}}^2 \cdot d \cdot M$ operations to compute $\mathbb{B}$, $r_{\mathbf{F}} r_{\mathbf{G}} \cdot d \cdot M$ operations to compute $\mathbf{b}_{j_k}$, and $r_{\mathbf{F}}^3$ to solve the system. Since $\mathbb{B}$ and the inner products in $\mathbf{b}_{j_k}$ are independent of j_k, the computation for another value of j_k has incremental cost $r_{\mathbf{G}} r_{\mathbf{F}} + r_{\mathbf{F}}^2$. Similarly, many of the computations involved in $\mathbb{B}$ and $\mathbf{b}_{j_k}$ are the same for different k. Thus, one full alternating least-squares iteration costs $\mathcal{O}(d \cdot r_{\mathbf{F}}(r_{\mathbf{F}}^2 + r_{\mathbf{G}} \cdot M))$. Because this algorithm uses inner products, which can only be computed to within roundoff error μ, the best accuracy obtainable is $\epsilon = \sqrt{\mu}$.

3.2. Controlling the condition number. Several of the most efficient mechanisms for producing low separation-rank representations exhibit ill-conditioning (see sections 2.1.1 and 2.1.2), and, thus, we need to control the condition number κ. Instead of just trying to minimize $\|\mathbf{F} - \mathbf{G}\|$, we add a penalty based on κ and minimize $\|\mathbf{F} - \mathbf{G}\|^2 + \alpha(\kappa\|\mathbf{F}\|)^2 = \|\mathbf{F} - \mathbf{G}\|^2 + \alpha \sum_{\tilde{l}}^{r_{\mathbf{F}}} (s_{\tilde{l}}^{\mathbf{F}})^2$, where α is a small parameter. This strategy is sometimes called "regularization" in other contexts. The modification needed in the alternating least-squares algorithm is especially simple. Since by definition $(s_{\tilde{l}}^{\mathbf{F}})^2 = \sum_j c_{j_k}^2(\tilde{l})$, we have $\sum_{\tilde{l}}^{r_{\mathbf{F}}} (s_{\tilde{l}}^{\mathbf{F}})^2 = \sum_{j_k} \sum_{\tilde{l}}^{r_{\mathbf{F}}} c_{j_k}^2(\tilde{l})$, and so we are still solving a linear least-squares problem. The only modification we need to make in our algorithm is to add $\alpha\mathbb{I}$ to $\mathbb{B}$ in (3.3).

In contrast to typical regularizations, α is fixed given the roundoff error μ. We wish to choose α to balance the terms so that the error due to loss of significance due to ill-conditioning is weighted the same as the error in the approximation. The error due to loss of significant digits is bounded by $\kappa\mu\|\mathbf{F}\|$, which indicates that we should choose $\alpha = \mu^2$. However, α that small would be truncated when $\mathbb{B} + \alpha\mathbb{I}$ is constructed, so we are forced to choose α just slightly larger than the roundoff error μ. This choice limits the best accuracy obtainable overall to $\sqrt{\mu}$, which is consistent with the best accuracy in the ordinary alternating least-squares algorithm.

3.3. Numerical example: Sine of a sum. In [3] we reported the basic performance of the separation-rank reduction on random vectors. Since random vectors in high dimensions are nearly orthogonal and, thus, do not easily exhibit ill-conditioning, we instead consider a particular example chosen for its ill-conditioning. We consider sine of the sum of variables, as in section 2.1.1. Suppose we have a separated representation with insufficient separation rank, as it would be the case if the $j = d$ term in the sum (2.9) is missing. This term then is the error in the representation, so the iteration will try to minimize it. The only way to minimize it is by maximizing its denominator, which occurs when $\sin(\alpha_k - \alpha_d) = \pm 1$ for all k. This condition forces $\sin(\alpha_1 - \alpha_2) \to 0$, which makes the $j = 1$ term in the sum increase without bound and, thus, sends the condition number κ to infinity.

Without the modification in section 3.2, we indeed see that as we iterate using alternating least squares, κ increases until the representation is untenable numerically. With the modification, the representation stabilizes at a poor, but tenable, representation, which can then be "grown" into a good approximation. For example, when $d = 10$ and $r = 9$, and we attempt to force an ill-conditioned representation by performing 1000 iterations, a typical result is approximation error $\epsilon = 0.055$ and $\kappa = 1.3 \cdot 10^5$. When we then allow $r = 10$, we achieve an approximation with $\epsilon = 1.11 \cdot 10^{-4}$ and $\kappa = 1.9 \cdot 10^4$. (Choosing α_j equally spaced in the identity leads to $\kappa \approx 7$.) By allowing $r = 11$, which is more than needed in the identity, we achieve $\epsilon = 1.58 \cdot 10^{-7}$ and $\kappa = 1.3 \cdot 10^2$. Our conclusion from many experiments of this type is that with the modification in section 3.2, the alternating least-squares algorithm is robust enough to use routinely, and provides close-to-optimal separation rank.

4. Solving a linear system. In this section we discuss how to solve the linear system $\mathbb{A}\mathbf{F} = \mathbf{G}$ for $\mathbf{F}$, where all objects are in the separated representation. One of the standard methods for solving a linear system is Gaussian elimination (LU factorization). In the separated representation, however, we do not act on individual entries in a d-dimensional matrix, so it is not clear if there is a generalization of this approach.

The situation is better with iterative algorithms. The first approach is to apply one of the iterative methods designed for large sparse systems. We also use this opportunity to describe how to combine other iterative algorithms with the separated representation. The second approach is to formulate the system as a least-squares problem, combine it with the least-squares problem used to find a representation with low separation rank, and then solve this joint problem by methods similar to those in section 3. This approach incorporates a separation-rank constraint into the formulation of the problem, and can serve as a model for how to approach other problems. We give a numerical example to illustrate this algorithm.

4.1. Iterative algorithms. Under the assumption that $\|\mathbb{I}-\mathbb{A}\| < 1$, one method for solving a system is the iteration

$$\mathbf{F}_{k+1} = (\mathbb{I} - \mathbb{A})\mathbf{F}_k + \mathbf{G}. \tag{4.1}$$

It requires only matrix-vector multiplication and vector addition, both of which can be performed using the algorithms in Proposition 2.3 with computational cost linear in d. Since these basic linear algebra operations increase the separation rank, we apply the separation-rank reduction algorithm in section 3 to $\mathbf{F}_{k+1}$ before using it in the next iteration.

This example illustrates our basic computational paradigm for iterative algorithms: replace the linear algebra operations in some standard algorithm by those in Proposition 2.3, and then insert the separation-rank reduction algorithm in section 3 between each step. The steepest descent and conjugate gradient methods (see, e.g., [17]), for example, require only matrix-vector multiplication, vector addition, and vector inner products, and so fit into this model. One can construct other iterations, such as

$$\mathbf{F}_{k+1} = (I - c\mathbb{A}^*\mathbb{A})\mathbf{F}_k + c\mathbb{A}^*\mathbf{G}, \tag{4.2}$$

where c is chosen to make $c\|\mathbb{A}^*\mathbb{A}\| < 1$.

There are several other important iterative algorithms that can be extended to use the separated representation by the same approach, for example,

- the power method ($\mathbf{F}_{k+1} = \mathbb{A}\mathbf{F}_k$) to determine the largest eigenvalue (in absolute value) and the corresponding eigenvector for a matrix $\mathbb{A}$,
- Schulz iteration ($\mathbb{B}_{k+1} = 2\mathbb{B}_k - \mathbb{B}_k\mathbb{A}\mathbb{B}_k$) to construct $\mathbb{A}^{-1}$ [38],
- sign iteration ($\mathbb{A}_{k+1} = (3\mathbb{A}_k - \mathbb{A}_k^3)/2$) to construct sign($\mathbb{A}$) (see, e.g., [33, 2]).

Building on these, we can then perform

- the inverse power method ($\mathbb{A}\mathbf{F}_{k+1} = \mathbf{F}_k$) for computing the smallest eigenvalue (in absolute value) and the corresponding eigenvector for a matrix $\mathbb{A}$,
- scaling and squaring ($(\exp(\mathbb{A}/2^n))^{2^n}$) to construct the matrix exponential $\exp(\mathbb{A})$.

Additional algorithms to compute, e.g., the fractional power of a function, f^α for $f > 0$ and $0 < \alpha < 1$, are under development.

It is essential of course that all the matrices and vectors involved have low separation rank. Our heuristics for deciding if these algorithms are appropriate is the following:

1. The initial matrix/vector must have low separation rank.
2. The final matrix/vector must have low separation rank (a priori or a posteriori).
3. The iteration should move us from the initial to the final state without creating excessive separation rank in the intermediate matrices/vectors. This condition is relatively simple to achieve in self-correcting algorithms like Schulz iteration since we can use low accuracy initially and then gradually improve it as the iteration progresses. This issue is more difficult in algorithms that are not self-correcting.

We note that in general, as the complexity estimates demonstrate, the cost of these types of algorithms is dominated by the separation-rank reduction step.

4.2. Finding a low separation-rank solution to a linear system. The problem of solving $\mathbb{A}\mathbf{F} = \mathbf{G}$ can be cast as that of minimizing the error $\|\mathbb{A}\mathbf{F} - \mathbf{G}\|$. We then add a constraint on the separation rank of $\mathbf{F}$, and, thus, simultaneously solve the linear system and find a reduced separation-rank representation for the solution. This approach of reformulating a problem as a least-squares minimization and then constraining to low separation rank has wide applicability, and will also be used in section 5.3.

We use an algorithm very similar to that described in section 3, with the same issues of convergence and conditioning. Note that if $\mathbb{A} = \mathbb{I}$, then it is in fact the same problem. We describe next the modifications needed in the core alternating least-squares algorithm in section 3.1. Given a matrix

$$\mathbb{A} = \sum_{l=1}^{r_{\mathbb{A}}} s_l^{\mathbb{A}} \mathbb{A}_1^l \otimes \mathbb{A}_2^l \otimes \cdots \otimes \mathbb{A}_d^l \tag{4.3}$$

and a right-hand-side vector

$$\mathbf{G} = \sum_{l=1}^{r_{\mathbf{G}}} s_l^{\mathbf{G}} \mathbf{G}_1^l \otimes \mathbf{G}_2^l \otimes \cdots \otimes \mathbf{G}_d^l, \tag{4.4}$$

we iteratively refine an approximate solution

$$\mathbf{F} = \sum_{l=1}^{r_{\mathbf{F}}} s_l^{\mathbf{F}} \mathbf{F}_1^l \otimes \mathbf{F}_2^l \otimes \cdots \otimes \mathbf{F}_d^l. \tag{4.5}$$

Fixing a direction k, we now pose the linear least-squares problem to refine in that direction. In contrast to the algorithm in section 3.1, the coordinates do not split into separate problems, so instead of (3.3) we form the matrix $\mathbb{B}$ with entries

$$B((\hat{j},\hat{l}),(j,l)) = \sum_{l_1}^{r_{\mathbb{A}}} \sum_{l_2}^{r_{\mathbb{A}}} s_{l_1}^{\mathbb{A}} s_{l_2}^{\mathbb{A}} \left(\sum_{j'}^{M} A_k^{l_1}(j',\hat{j}) A_k^{l_2}(j',j) \right) \prod_{i \neq k} \langle \mathbb{A}_i^{l_1} \mathbf{F}_i^l, \mathbb{A}_i^{l_2} \mathbf{F}_i^{\hat{l}} \rangle. \tag{4.6}$$

We then form the vector $\mathbf{b}$ with entries

$$b((\hat{j},\hat{l})) = \sum_{l_1}^{r_{\mathbb{A}}} s_{l_1}^{\mathbb{A}} \sum_{l_3}^{r_{\mathbf{G}}} s_{l_3}^{\mathbf{G}} \left(\sum_{j'}^{M} A_k^{l_1}(j',\hat{j}) g_k^{l_3}(j') \right) \prod_{i \neq k} \langle \mathbf{G}_i^{l_3}, \mathbb{A}_i^{l_1} \mathbf{F}_i^{\hat{l}} \rangle. \tag{4.7}$$

The normal equations for the direction k become

$$\mathbb{B}\mathbf{c} = \mathbf{b}, \tag{4.8}$$

which we solve for $\mathbf{c} = c((j,l))$. Once we find $\mathbf{c}$, we compute $s_l^{\mathbf{F}} = \|c(\cdot,l)\|$ and set $F_k^l(j) = c(j,l)/s_l^{\mathbf{F}}$.

The sums $\sum_{j'}^{M} A_k^{l_1}(j',\hat{j}) A_k^{l_2}(j',j)$ and $\sum_{j'}^{M} A_k^{l_1}(j',\hat{j}) g_k^{l_3}(j')$ are computed for all values at initialization, for total cost $\mathcal{O}(dr_{\mathbb{A}}^2 M^3 + dr_{\mathbb{A}} r_{\mathbf{G}} M^2)$. Computing $\mathbb{A}_i^{l_1}\mathbf{F}_i^{\hat{l}}$ costs $\mathcal{O}(dr_{\mathbb{A}} r_{\mathbf{F}} M^2)$. The inner products $\langle \mathbb{A}_i^{l_1}\mathbf{F}_i^l, \mathbb{A}_i^{l_2}\mathbf{F}_i^{\hat{l}} \rangle$ and $\langle \mathbf{G}_i^{l_3}, \mathbb{A}_i^{l_1}\mathbf{F}_i^{\hat{l}} \rangle$ can be computed for all i with cost $\mathcal{O}(dr_{\mathbb{A}}^2 r_{\mathbf{F}}^2 M + dr_{\mathbb{A}} r_{\mathbf{F}} r_{\mathbf{G}} M)$. Each time we increment k, we only need to update $\mathbb{A}\mathbf{F}$ and the inner products for one value of i, and so the cost for one loop in k has the same order as the cost for a single k. Computing the products $\prod_{i \neq k}$ takes $\mathcal{O}(dr_{\mathbb{A}}^2 r_{\mathbf{F}}^2 + dr_{\mathbb{A}} r_{\mathbf{F}} r_{\mathbf{G}})$. These can also be updated as k is incremented to preserve the overall complexity, although in practice this cost is not dominant, so we recompute them to avoid potential division by zero. The final assembly of $\mathbb{B}$ and $\mathbf{b}$ costs $\mathcal{O}(r_{\mathbb{A}}^2 r_{\mathbf{F}}^2 M^2 + r_{\mathbb{A}} r_{\mathbf{F}} r_{\mathbf{G}} M)$ and solving the normal equations costs $\mathcal{O}(r_{\mathbf{F}}^3 M^3)$. These last operations are not reused for different k, so their contributions to the overall complexity include a factor of d. The dominant complexity will depend on the relative sizes of the different parameters, but under the assumption that the separation ranks are small, the behavior is $\mathcal{O}(dM^3)$, as compared to $\mathcal{O}(dM)$ for the core alternating least-squares algorithm in section 3.1.

When the separation rank $r_{\mathbf{F}}$ appears to be insufficient, we add a new random term to $\mathbf{F}$, as we did in section 3. We find, however, that when the linear system is poorly conditioned, this random vector can overwhelm the approximate solution that we already have. To prevent this from happening, we precondition the new term by running the alternating least-squares iteration on it, while keeping those terms that we already have fixed. Once this appears to have converged, we include this new term in the main iteration.

4.3. Numerical example: Low separation-rank solutions of linear systems. We performed three sets of tests on the algorithm in section 4.2. Since they involved random vectors, and the alternating least-squares is initialized with random

vectors, the performance varied from run to run. We report a typical result for each test. All tests were performed on a laptop with a 1.7 GHz processor, had $d = 20$ and $M = 30$, and requested approximation within $\epsilon = 10^{-6}$. For simplicity of explanation, we used very low separation ranks, but as long as we avoid the separable case ($r_{\mathbf{F}_0} = 1$ or $r_{\mathbb{A}} = 1$; see below), these tests illustrate the general behavior.

The purpose of the first test was to check the correctness of the algorithm for random inputs. We generated a random $\mathbf{F}_0$ with $r_{\mathbf{F}_0} = 2$ and a random $\mathbb{A}$ with $r_{\mathbb{A}} = 6$, and formed $\mathbf{G} = \mathbb{A}\mathbf{F}_0$. Then we used the above algorithm to construct a solution $\mathbf{F}$ of the linear system to approximate $\mathbf{F}_0$. The algorithm performed 50 iterations at $r_{\mathbf{F}} = 1$ and then "decided" that it was stuck. It then increased $r_{\mathbf{F}}$ to 2, performed 18 iterations and achieved the requested error bound with $\|\mathbb{A}\mathbf{F} - \mathbf{G}\|/\|\mathbf{G}\| = 9.96 \cdot 10^{-7}$. We did not attempt to measure or control the condition number of the linear system, but did measure the error $\|\mathbf{F} - \mathbf{F}_0\|/\|\mathbf{F}_0\| = 1.08 \cdot 10^{-6}$. There was considerable variation between runs, but separation rank $r_{\mathbf{F}} = 2$ or 3 is achieved routinely in less than 30 seconds.

The purpose of the second test was to check the algorithm for a physically significant operator. The second test is identical to the first except that we chose $\mathbb{A}$ to be a discretization of the Laplacian Δ. The second derivatives were represented with a 9-point centered finite difference with order 8. We used (2.10) with $r_{\mathbb{A}} = d = 20$ for its separated representation, rather than the more efficient form from Theorem 2.5, in order to avoid possible confounding of effects. We also disabled the modification in section 3.2 for controlling the condition number. We estimated $\|\mathbb{A}\| \approx 1.17 \cdot 10^5$. The algorithm performed only 5 iterations at $r_{\mathbf{F}} = 1$ before "deciding" that it was stuck, with error $\|\mathbb{A}\mathbf{F} - \mathbf{G}\|/\|\mathbf{G}\| = 0.231$, and increasing the separation rank to 2. As noted at the end of section 4.2, we sometimes find it necessary to precondition the newly added term, and so have incorporated an initial fitting into our standard algorithm. In this example, the desired error bound was achieved after 4 iterations in this initial fitting, and then one iteration of the main algorithm achieved $\|\mathbb{A}\mathbf{F} - \mathbf{G}\|/\|\mathbf{G}\| = 3.94 \cdot 10^{-8}$. The total time used was less than one minute.

We then measured the error $\|\mathbf{F} - \mathbf{F}_0\|/\|\mathbf{F}_0\| = 2.99 \cdot 10^{-8}$. This linear system is singular, so we would expect a much worse result. It appears that there are two effects causing this favorable behavior. The first is that the constraint on the separation rank of $\mathbf{F}$ discourages the addition of extra terms, such as a constant term that is in the nullspace of Δ. In order to confirm this effect, we attempted to suppress it by starting with $r_{\mathbf{F}} = 3$, but the error did not become larger. The second effect is that a random vector is expected to have a small ($\sim M^{-d/2} \approx 10^{-15}$) projection on the constant vector (or any other fixed vector). Since we initialized with random vectors, we started with virtually nothing in the nullspace of $\mathbb{A}$, and it had no incentive to grow, especially considering the small number of iterations. In fact, the measured projections of $\mathbf{F}_0$ and our initial and final $\mathbf{F}$ onto the constant vector were smaller than the machine roundoff μ. We attempted to increase the projection on the constant vector by choosing $d = 2$ and $M = 10$. Then, when we set $r_{\mathbf{F}} = 3$, the error on some runs was as large as 0.1.

The purpose of the third test was to estimate the separation rank of the inverse Laplacian. If the right-hand side has separation rank one, then the separation rank of the solution cannot exceed that of the inverse operator. We kept $\mathbb{A}$ as a discretization of Δ, constructed $\mathbf{G}$ randomly with separation rank $r_{\mathbf{G}} = 1$, and attempted to find $\mathbf{F}$. We note that an explicit construction for $\mathbb{A}^{-1}$ is available in section 2.1.3 and yields an independent estimate of the separation rank of $\mathbb{A}^{-1}$. We limited the algorithm to

TABLE 1
Achieved errors and cumulative time used (in seconds) for solving a linear system involving the Laplacian in dimension 20.

$r_{\mathbf{F}}$	$\|\mathbb{A}\mathbf{F} - \mathbf{G}\|/\|\mathbf{G}\|$	Time
1	$2.5 \cdot 10^{-2}$	13
3	$3.8 \cdot 10^{-3}$	84
5	$6.8 \cdot 10^{-4}$	213
9	$8.0 \cdot 10^{-5}$	789
13	$8.4 \cdot 10^{-6}$	2048
19	$9.5 \cdot 10^{-7}$	6121

two iterations at each value of $r_{\mathbf{F}}$, and present the achieved error and time used for selected $r_{\mathbf{F}}$ in Table 1.

5. Antisymmetry. Motivated by the goal of computing the wavefunctions of the multiparticle Schrödinger equation, in this section we describe how to deal with functions that satisfy the antisymmetry constraint. This constraint is that the function must be odd under the exchange of any pair of variables (i.e., $f(x_1, x_2) = -f(x_2, x_1)$). We first sketch in section 5.1 the electronic N-particle Schrödinger equation in quantum mechanics. Since electrons are fermions, we are interested in the antisymmetric solutions, which we call wavefunctions. Although our motivation is the Schrödinger equation, the technique presented in this section demonstrates how a weak formulation can be used to represent functions that otherwise would have excessively large separation rank.

The straightforward construction of an antisymmetric function (via Slater determinants) yields separation rank $\mathcal{O}(N!)$. We avoid this problem by instead representing a "proto-wavefunction," that is, a function whose antisymmetrization yields the wavefunction itself. A representation of a proto-wavefunction allows us to do operations involving the wavefunction, such as computing its inner product with arbitrary functions, without constructing the wavefunction explicitly. The tools that make such an approach possible are well known in physics (e.g., Löwdin rules), and we review them in section 5.2.

The key to our approach is how we guide the algorithm for solving the electronic multiparticle Schrödinger equation to an antisymmetric solution. We work within the simplest solution algorithm, the power method applied to the shifted Hamiltonian. We modify the power method to find the largest eigenvalue that has an antisymmetric eigenfunction, rather than the largest eigenvalue overall. Applying the projector onto antisymmetric functions after each iteration would accomplish this goal, but this straightforward approach has computational complexity $\mathcal{O}(N!)$. We present in section 5.3 a separation-rank reduction algorithm that uses the pseudonorm induced by the projection onto antisymmetric functions. It has the effect of removing those eigenvalues that do not have an antisymmetric eigenfunction, and so allows the power method to find the correct eigenvalue and the wavefunction. In section 5.4 we present a numerical example of computing the wavefunction with our methods.

5.1. The ground-state multiparticle Schrödinger problem. The time-independent multiparticle Schrödinger equation describes the steady state of an interacting system of nonrelativistic particles. We will work within the Born–Oppenheimer approximation, so the nuclei are fixed and their effect is given by a potential. For each of the N electrons in the system there are three spatial variables $r = (x, y, z)$, and thus $3N$ total. The Schrödinger equation does not account for the spin of particles,

so each electron has an additional discrete spin variable σ taking the values $\{-\frac{1}{2}, \frac{1}{2}\}$. We denote the combined variables (r, σ) by γ. Without changing our basic formalism, we will consider the combined variable γ to be a single direction when using the separated representation.

The Hamiltonian operator for the multiparticle Schrödinger equation is the sum of three terms, $\mathcal{H} = \mathcal{T} + \mathcal{V} + \mathcal{W}$. The kinetic energy term $\mathcal{T} = -\frac{1}{2}\nabla^2$ is defined by

$$-2\mathcal{T} = (\Delta_1 \otimes \mathcal{I}_2 \otimes \cdots \otimes \mathcal{I}_N) + \cdots + (\mathcal{I}_1 \otimes \cdots \otimes \Delta_N), \tag{5.1}$$

where the three-dimensional Laplacian

$$\Delta_i = \frac{\partial^2}{\partial x_i^2} + \frac{\partial^2}{\partial y_i^2} + \frac{\partial^2}{\partial z_i^2} \tag{5.2}$$

corresponds to electron i. The nuclear potential portion $\mathcal{V}$ is given by

$$\mathcal{V} = (\mathcal{V}_1 \otimes \mathcal{I}_2 \otimes \cdots \otimes \mathcal{I}_N) + \cdots + (\mathcal{I}_1 \otimes \cdots \otimes \mathcal{V}_N), \tag{5.3}$$

where $\mathcal{V}_i$ is the operator that multiplies by the function $v(r_i)$, which includes nuclear potential terms as well as any external potentials. The electron-electron interaction portion $\mathcal{W}$ of the Hamiltonian is defined by

$$\mathcal{W} = \sum_{i=1}^{N-1} \sum_{m=i+1}^{N} \mathcal{W}_{im}\,, \tag{5.4}$$

where $\mathcal{W}_{im}$ is multiplication by the electron-electron interaction (Coulomb) potential $w(r_i - r_m) = c/|r_i - r_m|$.

The antisymmetric eigenfunctions of $\mathcal{H}$ represent electronic states of the system and we refer to them as "wavefunctions." The bound-state wavefunctions have negative eigenvalues. The ground-state multiparticle Schrödinger problem is to compute the wavefunction with the most negative eigenvalue. We note that the most negative eigenvalue has an eigenspace with no antisymmetric eigenfunctions, and thus is not associated with a wavefunction.

We need to discretize the operator $\mathcal{H}$ to form its matrix representation $\mathbb{H}$. For the purposes of this paper the particular choice of discretization is not important, although it is very important for actual implementations. We let M denote the number of degrees of freedom used for each electron, and, as we will see later, the antisymmetry constraint will force $N \le M$. We also need to put $\mathbb{H}$ in the separated representation, which we do using techniques based on Theorem 2.5. A partial analysis of this construction, without the antisymmetry condition, appeared in [3], and a more detailed analysis is in progress.

To construct the wavefunction ψ we will use the power method. The power method repeatedly applies a given matrix $\mathbb{A}$ to a test vector $\mathbf{F}_0$, using the iteration

$$\begin{aligned} \mathbf{G}_m &= \mathbb{A}\mathbf{F}_m, \\ \mathbf{F}_{m+1} &= \mathbf{G}_m/\|\mathbf{G}_m\|, \quad m = 0, 1, \ldots. \end{aligned} \tag{5.5}$$

If $\mathbb{A}$ is diagonalizable and has a distinct eigenvalue largest in magnitude, then for arbitrary $\mathbf{F}_0$, the iterates $\mathbf{F}_m$ will converge to the corresponding eigenvector (up to a sign). To use $\mathbb{H}$ within the power method, we choose $c \approx \|\mathbb{H}\|$ and set $\mathbb{A} = c\mathbb{I} - \mathbb{H}$, so that the eigenvalue that we want is positive and largest in magnitude.

Matrix-vector multiplication produces a vector with separation rank $r_{\mathbb{A}} r_{\mathbf{F}}$. As we iterate within the power method, the separation rank of $\mathbf{F}_m$, if unattended, would grow rapidly. To avoid this, we apply the separation-rank reduction after each iteration. The power method, however, does not take into account the antisymmetry constraint on the wavefunction. We do not seek the largest eigenvalue of $\mathbb{A}$, but rather the largest eigenvalue that has an antisymmetric eigenfunction. In the following sections we describe how to incorporate this antisymmetry constraint.

5.2. The antisymmetrizer and the Slater determinant. Given a function of N variables, one can compute its antisymmetric projection using a linear operator, called the *antisymmetrizer* and defined by (see, e.g., [34])

$$\mathcal{A} = \frac{1}{N!} \sum_{p \in S_N} (-1)^p \mathcal{P}, \tag{5.6}$$

where S_N is the permutation group on N elements. For the element $p \in S_N$, the operator $\mathcal{P}$ acts on a function by permuting its variables, as $\mathcal{P}\psi(\gamma_1, \gamma_2, \dots) = \psi(\gamma_{p(1)}, \gamma_{p(2)}, \dots)$. The sign $(-1)^p$ is -1 if p is an odd permutation and 1 if it is even. If $\mathcal{A}$ is applied to a separable function, then the result can be expressed as a Slater determinant:

$$\mathcal{A} \prod_{j=1}^{N} \phi_j(\gamma_j) = \frac{1}{N!} \begin{vmatrix} \phi_1(\gamma_1) & \phi_1(\gamma_2) & \cdots & \phi_1(\gamma_N) \\ \phi_2(\gamma_1) & \phi_2(\gamma_2) & \cdots & \phi_2(\gamma_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\gamma_1) & \phi_N(\gamma_2) & \cdots & \phi_N(\gamma_N) \end{vmatrix}. \tag{5.7}$$

If the functions $\{\phi_j\}$ are linearly dependent, then the determinant will evaluate to zero. We note that in the discrete case, where ϕ_j are replaced by vectors of length M, we thus must have $N \leq M$.

Since the number of elements in the permutation group S_N is $|S_N| = N!$, one cannot in practice apply the operator $\mathcal{A}$ in (5.6) or expand the determinant in (5.7). Instead one works with a "proto-wavefunction" that would generate the wavefunction upon the application of $\mathcal{A}$. Thus one represents (5.7) using only the product $\prod_{j=1}^{N} \phi_j(\gamma_j)$. Since $\mathcal{H}$ is purely symmetric, it commutes with the antisymmetrizer, and we have $\mathcal{H}\mathcal{A} \prod_{j=1}^{N} \phi_j(\gamma_j) = \mathcal{A}\mathcal{H} \prod_{j=1}^{N} \phi_j(\gamma_j)$. We therefore can defer the application of $\mathcal{A}$ and act on the proto-wavefunction instead. A sum of determinants can similarly be generated from a sum of separable functions.

This representation is sufficient because we really do not need the wavefunction itself, and the operations that we do need to perform can be applied without explicitly antisymmetrizing. For example, we can compute the inner product of antisymmetrized products by computing the determinant of inner products using the Löwdin rules (e.g., [34]), namely,

$$\left\langle \mathcal{A} \prod_{j=1}^{N} \phi_j(\gamma_j), \mathcal{A} \prod_{j=1}^{N} \tilde{\phi}_j(\gamma_j) \right\rangle = \frac{1}{N!} \begin{vmatrix} \langle \phi_1, \tilde{\phi}_1 \rangle & \langle \phi_1, \tilde{\phi}_2 \rangle & \cdots & \langle \phi_1, \tilde{\phi}_N \rangle \\ \langle \phi_2, \tilde{\phi}_1 \rangle & \langle \phi_2, \tilde{\phi}_2 \rangle & \cdots & \langle \phi_2, \tilde{\phi}_N \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \phi_N, \tilde{\phi}_1 \rangle & \langle \phi_N, \tilde{\phi}_2 \rangle & \cdots & \langle \phi_N, \tilde{\phi}_N \rangle \end{vmatrix}. \tag{5.8}$$

This formula reduces the problem to that of computing a determinant of numbers, which can be accomplished in at most $\mathcal{O}(N^3)$ operations by, e.g., diagonalization. (Asymptotically, we expect spatial locality to lead to $\mathcal{O}(N)$ complexity.)

5.3. Antisymmetric separation-rank reduction. If we applied the power method to $\mathcal{A}\mathbb{A}$ instead of $\mathbb{A}$, it would produce the wavefunction. As noted above, applying $\mathcal{A}$ has complexity $\mathcal{O}(N!)$, and produces a vector with separation rank $\mathcal{O}(N!)$. In this section we show how to apply the power method to $\mathcal{A}\mathbb{A}$, while never actually applying $\mathcal{A}$. The key is to incorporate $\mathcal{A}$ into the separation-rank reduction algorithm, and then use (5.8) to evaluate its effect.

The separation-rank reduction algorithm in section 3 is, at heart, the minimization of $\|\mathbf{F}-\mathbf{G}\|$ with $\mathbf{G}$ fixed and a constraint on the separation rank of $\mathbf{F}$. The algorithm in section 4.3 for solving a linear system is, at heart, the minimization of $\|\mathbb{A}\mathbf{F}-\mathbf{G}\|$. We now use the same principle to construct an antisymmetric separation-rank reduction algorithm that minimizes $\|\mathcal{A}(\mathbf{F}-\mathbf{G})\|$. We accomplish this without applying $\mathcal{A}$ directly by using the pseudonorm $\|\cdot\|_{\mathcal{A}} = \|\mathcal{A}(\cdot)\|$ for the approximation error bound (2.2) and the power method normalization (5.5).

Let us drop the index m in the power method and consider the problem of reducing the separation rank of $\mathbf{G} = \mathbf{G}_m/\|\mathbf{G}_m\|$ to obtain $\mathbf{F} = \mathbf{F}_{m+1}$. We begin with a fixed vector $\mathbf{G}$ and an approximation $\mathbf{F}$, and will again fix a direction k and refine in that direction, as in section 4. For simplicity we describe the $k = 1$ case. A straightforward calculation, which we omit, produces the normal equations for this linear least-squares problem. We form the matrix $\mathbb{B}$ with entries

$$B((\hat{j},\hat{l}),(j,l)) = \begin{vmatrix} \delta_{j\hat{j}} & F_2^{\hat{l}}(j) & \cdots & F_N^{\hat{l}}(j) \\ F_2^{l}(\hat{j}) & \langle \mathbf{F}_2^{l}, \mathbf{F}_2^{\hat{l}} \rangle & \cdots & \langle \mathbf{F}_2^{l}, \mathbf{F}_N^{\hat{l}} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ F_N^{l}(\hat{j}) & \langle \mathbf{F}_N^{l}, \mathbf{F}_2^{\hat{l}} \rangle & \cdots & \langle \mathbf{F}_N^{l}, \mathbf{F}_N^{\hat{l}} \rangle \end{vmatrix}, \tag{5.9}$$

where $\delta_{j\hat{j}}$ is 1 if $j = \hat{j}$ and 0 otherwise. We form the vector $\mathbf{b}$ with entries

$$b((\hat{j},\hat{l})) = \sum_{l'}^{r_{\mathbf{G}}} s_{l'}^{\mathbf{G}} \begin{vmatrix} G_1^{l'}(\hat{j}) & \langle \mathbf{G}_1^{l'}, \mathbf{F}_2^{\hat{l}} \rangle & \cdots & \langle \mathbf{G}_1^{l'}, \mathbf{F}_N^{\hat{l}} \rangle \\ G_2^{l'}(\hat{j}) & \langle \mathbf{G}_2^{l'}, \mathbf{F}_2^{\hat{l}} \rangle & \cdots & \langle \mathbf{G}_2^{l'}, \mathbf{F}_N^{\hat{l}} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ G_N^{l'}(\hat{j}) & \langle \mathbf{G}_N^{l'}, \mathbf{F}_2^{\hat{l}} \rangle & \cdots & \langle \mathbf{G}_N^{l'}, \mathbf{F}_N^{\hat{l}} \rangle \end{vmatrix}. \tag{5.10}$$

The normal equations for the direction $k = 1$ become

$$\mathbb{B}\mathbf{c} = \mathbf{b}, \tag{5.11}$$

which we solve for $\mathbf{c} = c((j,l))$. We then let $s_l^{\mathbf{F}} = \|c(\cdot,l)\|$ and $F_1^l(j) = c(j,l)/s_l^{\mathbf{F}}$.

In ordinary least-squares problems we are guaranteed that the normal equations will have a unique solution, under the condition that the vectors with which we approximate are linearly independent. In the present case, however, we are using the pseudonorm $\|\cdot\|_{\mathcal{A}}$, which has a nontrivial nullspace, so we will not have a unique solution. In particular, we can add to $\mathbf{F}_1^l$ any linear combination of $\{\mathbf{F}_i^l\}_{i=2}^N$ and still have a solution, since this added term is zero under $\|\cdot\|_{\mathcal{A}}$. We require that $\mathbf{F}_1^l$ be orthogonal to $\{\mathbf{F}_i^l\}_{i=2}^N$, so that we get a unique, minimal-norm solution, and so suppress the nonantisymmetric part that is invisible to the pseudonorm $\|\cdot\|_{\mathcal{A}}$. To accomplish this, we first construct a unitary matrix $\mathbb{U}_l$ that rotates the subspace spanned by $\{\mathbf{F}_i^l\}_{i=2}^N$ into the first $N-1$ coordinate directions. The construction is essentially the same as that used in Schur factorization or Householder QR factorization, and we refer the reader to [17] for details. Then, to each $M \times M$ block $B((\cdot,\hat{l}),(\cdot,l))$, we apply $\mathbb{U}_{\hat{l}}$

on the left and $\mathbb{U}_l$ on the right, and then remove the first $N-1$ columns and rows, which now represent the nullspace. Similarly, we apply $\mathbb{U}_{\hat{l}}$ to the subvector $b((\cdot,\hat{l}))$ and remove its first $N-1$ entries. We then solve (5.11) in this new form, insert $N-1$ entries with value zero in each subvector, and then rotate the solution back to obtain $\mathbf{c}$.

It requires $\mathcal{O}(r_{\mathbf{F}}^2 N^2 M)$ operations to compute the inner products in $\mathbb{B}$, $\mathcal{O}(r_{\mathbf{F}}^2 N^3 M^2)$ operations to compute the determinants in $\mathbb{B}$, and similarly $\mathcal{O}(r_{\mathbf{G}} r_{\mathbf{F}} N^3 M)$ to compute $\mathbf{b}$. Solving (5.11) then takes $\mathcal{O}(r_{\mathbf{F}}^3 M^3)$ operations. The inner products can be updated and reused for different k, but the other operations cannot. One full alternating least squares iteration costs $\mathcal{O}(r_{\mathbf{F}} NM(r_{\mathbf{G}} N^3 + r_{\mathbf{F}}^2 M^2 + r_{\mathbf{F}} N^3 M))$.

5.3.1. An accelerated algorithm. In our setup we have $M \geq N$. Assuming that M is proportional to N, the dominant term in the computational complexity is $\mathcal{O}(r_{\mathbf{F}}^2 N^4 M^2) = \mathcal{O}(N^6)$, which comes from the computation of all the determinants in $\mathbb{B}$. In this section we describe a linear algebra technique that reduces this complexity by a factor of N^2. Our algorithm is based on the following proposition.

PROPOSITION 5.1 (determinant of a rank-2 perturbation).

$$|\mathbb{I} + \mathbf{u}_1\mathbf{v}_1^* + \mathbf{u}_2\mathbf{v}_2^*| = 1 + \mathbf{v}_1^*\mathbf{u}_1 + \mathbf{v}_2^*\mathbf{u}_2 + \mathbf{v}_1^*\mathbf{u}_1\mathbf{v}_2^*\mathbf{u}_2 - \mathbf{v}_1^*\mathbf{u}_2\mathbf{v}_2^*\mathbf{u}_1. \tag{5.12}$$

To show this, observe that any eigenvector that is not in the span of $\{\mathbf{u}_1, \mathbf{u}_2\}$ has eigenvalue 1, and so does not affect the determinant. We can then consider an arbitrary vector $\alpha\mathbf{u}_1 + \beta\mathbf{u}_2$, and solve algebraically the eigenvalue problem in this two-dimensional subspace. Multiplying these two eigenvalues gives us (5.12).

Fixing $\hat{l}$ and l in $\mathbb{B}$, we obtain an $M \times M$ block, in which each entry requires the computation of a determinant at cost N^3, for total cost M^2N^3. The determinants required, however, are quite similar. By reusing some of the computations we reduce the cost to compute this block to $\mathcal{O}(M^2N)$. The block is of the form

$$A(i,j) = \begin{vmatrix} \delta_{ji} & \mathbf{w}_j^* \\ \mathbf{x}_i & \mathbb{E} \end{vmatrix}, \tag{5.13}$$

where $\mathbf{w}_j$ and $\mathbf{x}_i$ are column vectors of length $N-1$, $\mathbb{E}$ is a fixed $(N-1)\times(N-1)$ matrix, and $(\cdot)^*$ indicates transpose. We perform Gaussian elimination (LU decomposition) with full pivoting to $\mathbb{E}$, at cost $\mathcal{O}(N^3)$. If this completes without detecting a (numerically) zero pivot, then the LU decomposition tells us the determinant $|\mathbb{E}|$ and gives us the ability to apply the inverse $\mathbb{E}^{-1}$. We then write

$$A(i,j) = \begin{vmatrix} 1 & 0 \\ 0 & \mathbb{E} \end{vmatrix} \left| \mathbb{I} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & \mathbf{w}_j^* \end{bmatrix} + \begin{bmatrix} \delta_{ji} - 1 \\ \mathbb{E}^{-1}\mathbf{x}_i \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right|. \tag{5.14}$$

Then for M values of i we compute $\mathbb{E}^{-1}\mathbf{x}_i$ at cost $\mathcal{O}(N^2)$, for net cost MN^2. Finally, for M^2 values of (i,j) we compute the rightmost determinant in (5.14) at cost $\mathcal{O}(N)$ using Proposition 5.1.

If the Gaussian elimination detects a zero pivot on the last $(N-1)$ step, then we modify the factorization in (5.14) so that the augmentation of $\mathbb{E}$ in its first term is nonsingular. In (5.14) we chose to augment $\mathbb{E}$ above and on the left by zero vectors because this provides the simplest expression, but any other fixed vectors will do. In particular, by putting an extra 1 in the proper column of the first row, and -1 in the proper row of the first column, we can effectively introduce a 1 in the position where

the zero pivot occurred, and thus remove it. The second term in (5.14) is modified to compensate for these changes, but then the same general procedure is followed.

If a zero pivot is detected before the last step, then we can conclude that $A(i,j) = 0$ for all (i,j) and not compute them. One way to see this is to note that the rows of $\mathbb{E}$ span a subspace of dimension at most $N-3$. Augmenting with one extra coordinate via $\mathbf{x}_i$ can increase the span to dimension $N-2$, and including the vector $[\delta_{ji}\ \mathbf{w}_j^*]$ can increase the span to dimension $N-1$, but this still leaves a singular matrix.

5.4. Numerical example: Schrödinger wavefunction. In this section we illustrate the effect of the antisymmetric separation-rank reduction by computing the wavefunction for an academic model of the multiparticle Schrödinger equation with one-dimensional particles and simplified potentials. Our model is certainly not realistic, but the results do show some similarities to phenomena observed in physics. We provide tools to interpret the results, and show that the wavefunctions that we compute are consistent with the intuition developed by CI methods (see, e.g., [39]).

For our example we will let $\gamma = x$ be a one-dimensional, periodic, spinless variable. We choose the nuclear potential $v(x) = c_v \cos(2\pi x)$ and the electron interaction potential $w(x_i - x_m) = c_w \cos(2\pi(x_i - x_m))$. Our Hamiltonian is thus

$$\mathcal{H} = \mathcal{T} + c_v \sum_{i=1}^{N} \cos(2\pi x_i) + c_w \sum_{i=1}^{N-1} \sum_{m=i+1}^{N} \cos(2\pi(x_i - x_m)). \tag{5.15}$$

We discretize by sampling the variable x at M equispaced points to form the vector $\mathbf{x}$. The second derivatives $\partial^2/\partial x^2$ in $\mathcal{T}$ are represented with a 9-point centered finite difference with order 8, which we denote $\mathbb{D}^2$. We choose the shift $c \approx \|\mathbb{H}\|/2$ and set $\mathbb{A} = c\mathbb{I} - \mathbb{H}$. We combine $c\mathbb{I} - \mathbb{T} - \mathbb{V}$ into $\sum_i^N [(c/N)\mathbb{I}_i - \mathbb{D}_i^2 - c_v \cos(2\pi \mathbf{x}_i)]$, and then represent it using the construction in (2.14) with stepsize h_v and a p_v-point finite difference in the auxiliary parameter. The electron interaction is first separated using a trigonometric identity as $\cos(2\pi(\mathbf{x}_i - \mathbf{x}_m)) = \cos(2\pi\mathbf{x}_i)\cos(2\pi\mathbf{x}_m) - \sin(2\pi\mathbf{x}_i)\sin(2\pi\mathbf{x}_m)$, and then each term is represented using a second derivative version of (2.14) with stepsize h_w and a p_w-point finite difference. We use the parameters $N = 5$, $M = 30$, $c_v = 100$, $c_w = 5$, $c = 14000$, $h_v = 0.1$, $p_v = 6$, $h_w = 0.1$, and $p_w = 5$, which were chosen not for realism, but simply to provide an elegant example. With these parameters, we have a separation rank $r = 16$ approximation for $\mathbb{A}$ with relative error less than 10^{-7}. As discussed in the previous sections, given a working precision of 10^{-16}, an allowance for the conditioning of the representation, and the need to compute square roots in order to compute norms, this is the smallest error that we are able to measure.

We first construct a separable approximation $\mathbf{F}_0$ to the wavefunction by running the power method algorithm while forcing the separation-rank reduction algorithm to yield the best approximation with separation rank one. For separable functions, one does not need the full antisymmetric separation-rank reduction, so we instead orthogonalize the vectors $\mathbf{F}_i^1$ after the ordinary reduction. By always orthogonalizing in the order of increasing i, the vectors naturally order themselves from low to high "energy" (frequency). We conjecture that this process produces the Hartree–Fock solution, but we have not studied this issue in detail. In this example, we compute $\mathbf{F}_0$ using 10000 iterations.

We then perform the main method with $\mathbf{F}_0$ as our starting guess, using $\epsilon = 10^{-4}$ and 1000 iterations. It took about a half an hour to run on a laptop with a 1.7 GHz processor and 640 MB of memory, using double precision with machine roundoff μ about 10^{-16}. It produced an approximation for the wavefunction with separation rank two and condition number $\kappa = 0.999898082$ (by its definition (2.3), κ may be

TABLE 2

Separation rank, achieved approximation, and eigenvalue estimates for the separable ($\mathbf{F}_0$) and main ($\mathbf{F}$) approximations to the wavefunction.

	r	ϵ	$\|\mathcal{A}\mathbb{H}\mathbf{F}\|$	$\langle \mathcal{A}\mathbb{H}\mathbf{F}, \mathcal{A}\mathbf{F}\rangle$
$\mathbf{F}_0$	1	$3.4 \cdot 10^{-3}$	322.6727395	322.3859013
$\mathbf{F}$	2	10^{-4}	321.8852595	321.8844158

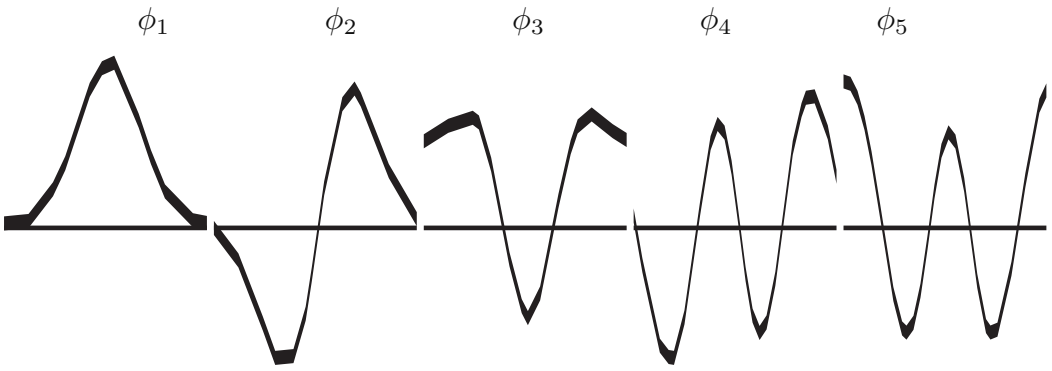


FIG. 1. *The computed separable approximation $\mathbf{F}_0$ to the wavefunction.*

smaller than 1 by ϵ). We monitor the convergence of the algorithm by comparing two eigenvalue estimates for $\mathbb{H}$, namely $\|\mathcal{A}\mathbb{H}\mathbf{F}\|$ and $\langle \mathcal{A}\mathbb{H}\mathbf{F}, \mathcal{A}\mathbf{F}\rangle$, which agree only when $\mathbf{F}$ is indeed an eigenvector. These estimates are presented in Table 2, along with the corresponding estimates for $\mathbf{F}_0$. The relative discrepancy in eigenvalue estimate is $2.62 \cdot 10^{-6}$, which is smaller than the best that we expect to achieve, given that we chose $\epsilon = 10^{-4}$. Using this criteria, we conclude that the algorithm has converged and we did indeed approximate an eigenfunction of $\mathbb{H}$. Note that the eigenvalue estimates are smaller than those for $\mathbf{F}_0$, as expected, and that the eigenvalue need not be negative since our potentials are not strictly negative.

5.4.1. Analysis of the wavefunction. We now perform an "autopsy" on our wavefunction and compare it qualitatively with the intuition based on CI methods. First consider the separable approximation $\mathbf{F}_0$. Its factors can be identified as the orbitals $\{\phi_j\}_j^N$. Figure 1 shows a graphical representation of $\mathbf{F}_0$ and illustrates these orbitals.

We use these orbitals to help us visualize the wavefunction that the main iteration produces. In CI methods the factors $\mathbf{F}_i^l$ are all taken from the set of eigenfunctions $\{\phi_j\}$ of the single-electron Hamiltonian. By simple permutations we can put each separable term in the "maximum coincidence" ordering, where as many factors as possible are lined up with the orbitals $\{\phi_j\}_j^N$. Factors that do not agree can then be interpreted as excited states. Since the Slater determinant form (5.7) is unchanged by unitary transformations, our antisymmetric separation-rank reduction algorithm ignores them as well, and so generally produces proto-wavefunctions that are not suitable for visualization. There is not an obvious definition for the maximum-coincidence order in our case, so to each term of the output of the antisymmetric separation-rank reduction we apply the following algorithm, which is a unitary transformation with a bias toward aligning lower i.

1. Find the i such that $|\langle \mathbf{F}_i^l, \phi_1\rangle|$ is maximized, and permute it to position 1.
2. Transform $\mathbf{F}_1^l \to \mathbf{F}_1^l + \sum_{i=2}^N \langle \mathbf{F}_i^l, \phi_1\rangle \mathbf{F}_i^l$ and $\mathbf{F}_i^l \to \mathbf{F}_i^l - \langle \mathbf{F}_i^l, \phi_1\rangle \mathbf{F}_1^l$ for $i > 1$, and then normalize them.

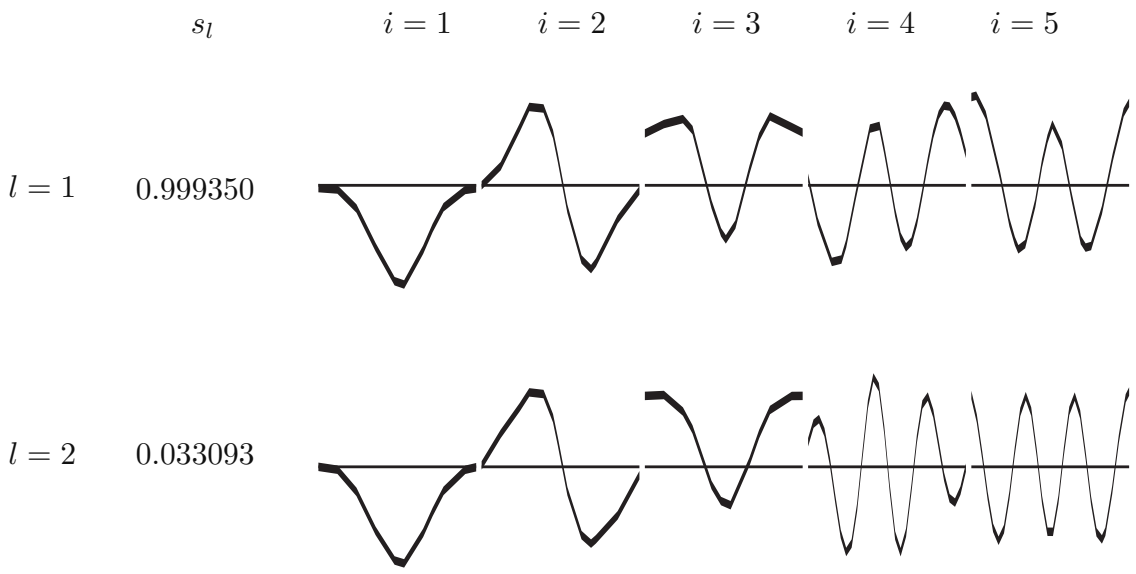


FIG. 2. *The computed wavefunction, represented by* $\sum_{l=1}^{2} s_l \prod_{i=1}^{5} \mathbf{F}_i^l$. *Each subgraph shows one vector* $\mathbf{F}_i^l$.

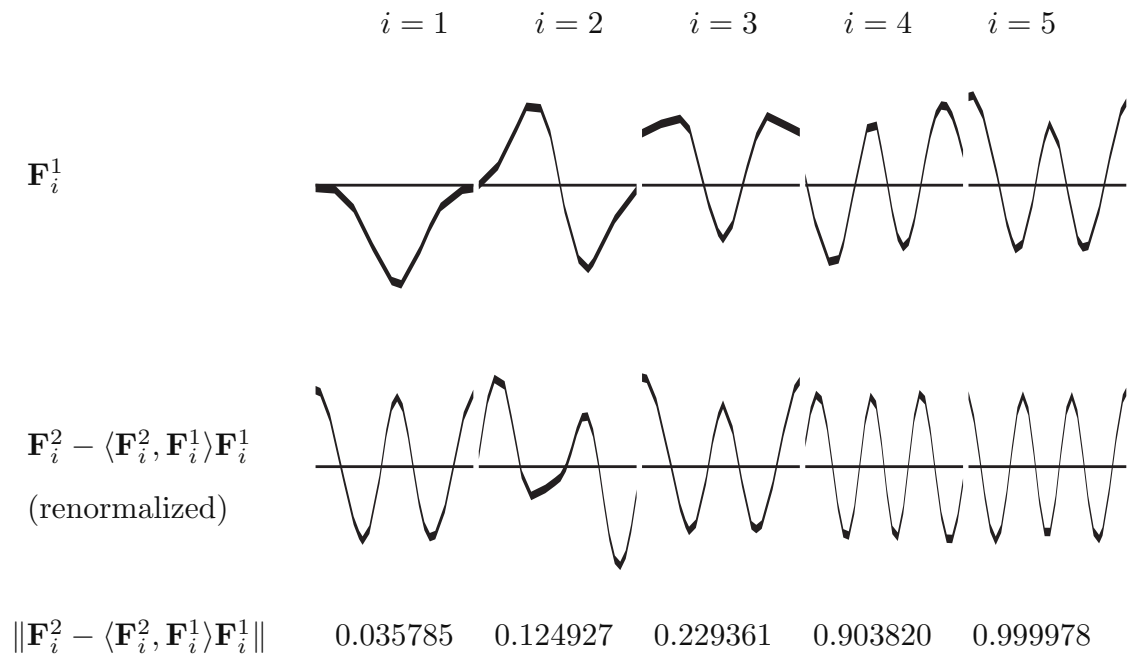


FIG. 3. *The basis spanned by the wavefunction in Figure* 2 *for each electron, and the magnitude in the second direction.*

3. Fix position 1, exclude ϕ_1, and do the algorithm recursively on the remaining positions.

The resulting approximate wavefunction is shown in Figure 2. The second term appears to have the first three electrons in their ground states, and the last two excited into higher states.

We next compute the component of $\mathbf{F}_i^2$ that is orthogonal to $\mathbf{F}_i^1$ for each i. By renormalizing, we obtain a basis for the space spanned by $\mathbf{F}_i^1$ and $\mathbf{F}_i^2$, shown in Figure 3. We can see that electrons one and three have a component in the electron

TABLE 3

The amount of the ground state orbitals present in the two terms of the wavefunction in Figure 2, and their net excitations.

Ground state orbital	Wavefunction term	
	$l = 1$	$l = 2$
1	0.999999	0.999359
2	0.999992	0.992361
3	0.999998	0.973370
4	0.999861	0.455506
5	0.999968	0.233443
Excitation level	0.018945	1.344271

five ground state, and that electrons four and five have components in what looks like orbitals six and seven. Electron two has a component in what appears to be a mixture of orbitals four and one. We also give the strength of these components, namely the norm of the projection of $\mathbf{F}_i^2$ orthogonal to $\mathbf{F}_i^1$. We can see that electrons one and two are essentially unexcited, electron three has a significant component in the electron five ground state, electron four is a nearly even mixture, and electron five is almost completely excited.

The results in Figure 3 are useful for developing our intuition and comparing with CI, but they depend on the maximum-coincidence order that we used, which is somewhat arbitrary. To get more meaningful data, we use $\mathbf{F}_0$ to get a numerical measure of the amount of the ground state orbitals present in each term in the wavefunction. We compute $(\sum_{i=1}^{N} \langle \mathbf{F}_i^l, \phi_j \rangle^2)^{1/2}$ for each l and j. These quantities are invariant under unitary transformations on the proto-wavefunction, but the amounts computed for different j may not be simultaneously realizable by any unitary transformation. In CI methods they evaluate to either 0 or 1. We also compute the "net excitation" as the amount of norm unaccounted for by the ground state orbitals. (See [37] for a discussion of measuring excitation level.) The results for our example are given in Table 3. We see that the first term is essentially unexcited. The second term shows a decrease as we move to higher electrons, but still has a significant component in the ground state of the fifth electron. The fractional excitation level suggests that we are in a case analogous to (2.7), where CI would require significantly more terms. Nonorthogonal CI methods (see, e.g., [37, 30]) would fall in between.

6. Future work. Our current efforts are focused in three directions.

First, we are working out "technical" details to allow these techniques to be used routinely in dimensions two and three. Separated representations have been used in two-dimensional problems of wave propagation [6], and will soon be extended to three dimensions. Separated representations have been used in quantum chemistry within (three-dimensional) one-particle theories [19, 20]. A complete transition to separated representations will require the ability to compute the square or cubic roots of a function, a multiresolution structure that is efficient for potentials and Green's functions with singularities, and the resolution of several other issues. Such work is under way and will be reported elsewhere.

Second, we are developing algorithms to compute the wavefunctions of the multiparticle Schrödinger operator. The computation of the antisymmetric wavefunction demonstrated in this paper is a step in this direction. The next major issue is size-extensivity and its impact on the separation rank of the wavefunction. The separated representation in its current form is not size-extensive, so we are pursuing a hierarchical version that may be able to achieve an approximate, algorithmic size-extensivity.

Finally, we are working to resolve the critical question as to what extent separated representations can represent functions and operators in general.

Acknowledgments. We would like to thank Dr. Robert Harrison (ORNL and University of Tennessee) and Dr. Lucas Monzón (University of Colorado) for many useful conversations about this project.

REFERENCES

[1] R. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, NJ, 1961.
[2] G. Beylkin, N. Coult, and M. J. Mohlenkamp, *Fast spectral projection algorithms for density-matrix computations*, J. Comput. Phys., 152 (1999), pp. 32–54.
[3] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Natl. Acad. Sci. USA, 99 (2002), pp. 10246–10251.
[4] G. Beylkin and L. Monzón, *On generalized Gaussian quadratures for exponentials and their applications*, Appl. Comput. Harmon. Anal., 12 (2002), pp. 332–373.
[5] G. Beylkin and L. Monzón, *On approximation of functions by exponential sums*, Appl. Comput. Harmon. Anal., to appear.
[6] G. Beylkin and K. Sandberg, *Wave propagation using bases for bandlimited functions*, Wave Motion, 41 (2005), pp. 263–291.
[7] G. Bhatnagar, *A short proof of an identity of Sylvester*, Int. J. Math. Math. Sci., 22 (1999), pp. 431–435.
[8] R. Bro, *Parafac. tutorial & applications*, Chemom. Intell. Lab. Syst., 38 (1997), pp. 149–171.
[9] H.-J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer. (2004), pp. 147–269.
[10] F. Calogero, *Remarkable matrices and trigonometric identities.* II, Commun. Appl. Anal., 3 (1999), pp. 267–270.
[11] F. Calogero, *Classical Many-Body Problems Amenable to Exact Treatments*, Lecture Notes in Phys., 66, Springer, Berlin, 2001.
[12] J.-F. Cardoso, *Higher-order contrasts for independent component analysis*, Neural Comp., 11 (1999), pp. 157–192.
[13] J.-F. Cardoso and A. Souloumiac, *Blind beamforming for non-Gaussian signals*, IEE Proceedings F, 140 (1993), pp. 362–370.
[14] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
[15] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-*1 *and rank-*$(R_1, R_2, \ldots, R_N)$ *approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.
[16] J. M. Ford and E. E. Tyrtyshnikov, *Combining Kronecker product approximation with discrete wavelet transforms to solve dense, function-related linear systems*, SIAM J. Sci. Comput., 25 (2003), pp. 961–981.
[17] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[18] I. Gradshteyn and I. M. Ryzhik, eds., *Table of Integrals, Series, and Products*, Academic Press, New York, 1965.
[19] R. Harrison, G. Fann, T. Yanai, and G. Beylkin, *Multiresolution quantum chemistry in multiwavelet bases*, in Computational Science—ICCS 2003, P. M. A. Sloot et al., eds., Lecture Notes in Comput. Sci. 2660, Springer, Berlin, 2003, pp. 103–110.
[20] R. Harrison, G. Fann, T. Yanai, Z. Gan, and G. Beylkin, *Multiresolution quantum chemistry: Basic theory and initial applications*, J. Chem. Phys., 121 (2004), pp. 11587–11598.
[21] R. A. Harshman, *Foundations of the PARAFAC procedure: Model and Conditions for an "Explanatory" Multimodal Factor Analysis*, Working Papers in Phonetics 16, UCLA, 1970, 1–84; also available online from http://publish.uwo.ca/∼harshman/wpppfac0.pdf.
[22] T. Hrycak and V. Rokhlin, *An improved fast multipole algorithm for potential fields*, SIAM J. Sci. Comput., 19 (1998), pp. 1804–1826.
[23] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.
[24] R. Jakob-Chien and B. K. Alpert, *A fast spherical filter with uniform resolution*, J. Comput. Phys., 136 (1997), pp. 580–584.
[25] P. Jones, J. Ma, and V. Rokhlin, *A fast direct algorithm for the solution of the Laplace equation on regions with fractal boundaries*, J. Comput. Phys., 113 (1994), pp. 35–51.

[26] M. H. Kalos, *Monte Carlo calculations of the ground state of three- and four-body nuclei*, Phys. Rev. (2), 128 (1962), pp. 1791–1795.
[27] T. G. Kolda, *Orthogonal tensor decompositions*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 243–255.
[28] P. M. Kroonenberg and J. de Leeuw, *Principal component analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 45 (1980), pp. 69–97.
[29] S. E. Leurgans, R. A. Moyeed, and B. W. Silverman, *Canonical correlation analysis when the data are curves*, J. Roy. Statist. Soc. Ser. B, 55 (1993), pp. 725–740.
[30] A. Lüchow and R. Fink, *On the systematic improvement of fixed-node diffusion quantum Monte Carlo energies using natural orbital CI guide functions*, J. Chem. Phys., 113 (2000), pp. 8457–8463.
[31] S. C. Milne, *A q-analog of the Gauss summation theorem for hypergeometric series in* U(n), Adv. in Math., 72 (1988), pp. 59–131.
[32] M. J. Mohlenkamp and L. Monzón, *Trigonometric identities and sums of separable functions*, The Mathematical Intelligencer, to appear.
[33] A. H. R. Palser and D. E. Manolopoulos, *Canonical purification of the density matrix in electronic-structure theory*, Phys. Rev. B, 58 (1998), pp. 12704–12711.
[34] R. Pauncz, *The Symmetric Group in Quantum Chemistry*, CRC Press, Boca Raton, FL, 1995.
[35] V. Pereyra and G. Scherer, *Efficient computer manipulation of tensor products with applications to multidimensional approximation*, Math. Comp., 27 (1973), pp. 595–605.
[36] T. M. Rassias and J. Šimša, *Finite Sums Decompositions in Mathematical Analysis*, Pure Appl. Math. (N.Y.), John Wiley & Sons Ltd., Chichester, UK, 1995.
[37] S. P. Rudin, *Configuration Interaction with Non-orthogonal Slater Determinants Applied to the Hubbard Model, Atoms, and Small Molecules*, Ph.D. thesis, The Ohio State University, Columbus, OH, 1997.
[38] G. Schulz, *Iterative Berechnung der reziproken Matrix*, Z. Angew. Math. Mech., 13 (1933), pp. 57–59.
[39] C. D. Sherrill and H. F. Schaefer III, *The configuration interaction method: Advances in highly correlated approaches*, Adv. in Quantum Chem., 127 (1999), pp. 143–269.
[40] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis: Applications in the Chemical Sciences*, John Wiley & Sons, New York, 2004.
[41] V. N. Temlyakov, *Estimates of best bilinear approximations of periodic functions*, Proc. Steklov Inst. Math., (1989), pp. 275–293.
[42] E. Tyrtyshnikov, *Kronecker-product approximations for some function-related matrices*, Linear Algebra Appl., 379 (2004), pp. 423–437.
[43] E. E. Tyrtyshnikov, *Tensor approximations of matrices generated by asymptotically smooth functions*, Mat. Sb., 194 (2003), pp. 147–160.
[44] C. F. Van Loan and N. Pitsianis, *Approximation with Kronecker products*, in Proceedings of the Linear Algebra for Large Scale and Real-Time Applications (Leuven, 1992), NATO Adv. Sci. Inst. Ser. E Appl. Sci. 232, Kluwer, Dordrecht, the Netherlands, 1993, pp. 293–314.
[45] N. Yarvin and V. Rokhlin, *A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics*, J. Comput. Phys., 147 (1998), pp. 594–609.
[46] C. Zenger, *Sparse grids*, in Parallel Algorithms for Partial Differential Equations (Kiel, 1990), Notes Numer. Fluid Mech. 31, Vieweg, Braunschweig, 1991, pp. 241–251.