Journal of Computational Physics 348 (2017) 220–230

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Optimization via separated representations and the canonical tensor decomposition [☆]

Matthew J. Reynolds [a], Gregory Beylkin [b], Alireza Doostan [a,*]

[a] *Department of Aerospace Engineering Sciences, 429 UCB, University of Colorado, Boulder, CO 80309, United States*
[b] *Department of Applied Mathematics, UCB 526, University of Colorado, Boulder, CO 80309, United States*

ARTICLE INFO

Article history:
Received 3 September 2016
Received in revised form 5 April 2017
Accepted 6 July 2017
Available online 12 July 2017

Keywords:
Separated representations
Tensor decompositions
Canonical tensors
Global optimization
Quadratic convergence

ABSTRACT

We introduce a new, quadratically convergent algorithm for finding maximum absolute value entries of tensors represented in the canonical format. The computational complexity of the algorithm is linear in the dimension of the tensor. We show how to use this algorithm to find global maxima of non-convex multivariate functions in separated form. We demonstrate the performance of the new algorithms on several examples.

1. Introduction

Finding global extrema of a multivariate function is an ubiquitous task with many approaches developed to address this problem (see e.g. [1]). Unfortunately, no existing optimization method can guarantee that the results of optimization are true global extrema unless restrictive assumptions are placed on the function. Assumptions on smoothness of the function do not help since it is easy to construct an example of a function with numerous local extrema “hiding” the location of the true one. While convexity assumptions are helpful for finding global maxima, in practical applications there are many non-convex functions. For non-convex functions various randomized search strategies have been suggested and used but none can assure that the results are true global extrema (see, e.g., [2,1]).

We propose a new approach to the problem of finding global extrema of a multivariate function under the assumption that the function has certain structure, namely, a separated representation with a reasonably small separation rank. While this assumption limits the complexity of the function, i.e. number of independent degrees of freedom in its representation, there is no restriction on its convexity. Furthermore, a large number of functions that do not appear to have an obvious separated representation do in fact possess one, as was observed in [3–5]. In particular, separated representations have been used recently to address curse of dimensionality issues that occur when solving PDEs with random data in the context of uncertainty quantification (see, e.g., [6–13]).

We present a surprisingly simple algorithm that uses canonical tensor decompositions (CTDs) for the optimization process. A canonical decomposition of a tensor $\mathbf{U} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_d}$ is of the form,

[☆] This material is based upon work supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, under Award Number DE-SC0006402, and NSF grants DMS-1228359, CMMI-1454601, and DMS-1320919.

* Corresponding author.
E-mail address: alireza.doostan@colorado.edu (A. Doostan).

http://dx.doi.org/10.1016/j.jcp.2017.07.012
0021-9991/© 2017 Elsevier Inc. All rights reserved.

$$\mathbf{U} = U\left(i_1 \dots i_d\right) = \sum_{l=1}^{r} s_l u^l_{i_1} u^l_{i_2} \cdots u^l_{i_d},$$

where $u^l_{i_j}$, $j = 1, \dots, d$ are entries of vectors in $\mathbb{R}^{M_j}$ and r is the separation rank. Using the CTD format to find entries of a tensor with the maximum absolute value was first explored in [14] via an analogue of the matrix power method. Unfortunately, the algorithm in [14] has the same weakness as the matrix power method: its convergence can be very slow, thus limiting its applicability. Instead of using the power method, we introduce a quadratically convergent algorithm based on straightforward sequential squaring of tensor entries in order to find entries with the maximum absolute value.

For a tensor with even a moderate number of dimensions finding the maximum absolute value of the entries appears to be an intractable problem. A brute-force search of a d-directional tensor requires $\mathcal{O}\left(N^d\right)$ operations, an impractical computational cost. However, for tensors in CTD format the situation is different since the curse of dimensionality can be avoided.

Our approach to finding the entries with the maximum absolute value (abbreviated as "the maximum entry" where it does not cause confusion) is as follows: as in [14], we observe that replacing the maximum entries with 1's and the rest of the entries with zeros results in a tensor that has a low separation rank representation (at least in the case where the number of such extrema is reasonably small). To construct such a tensor, we simply square the entries via the Hadamard product, normalize the result and repeat these steps a sufficient number of times. The resulting algorithm is quadratically convergent as we raise the tensor entries to the power 2^n after n steps. Since the nominal separation rank of the tensor grows after each squaring, we use CTD rank reduction algorithms to keep the separation rank at a reasonable level. The accuracy threshold used in rank reduction algorithms limits the overall algorithm to finding the global maximum within the introduced accuracy limitation. However, in many practical applications, we can find the true extrema as this accuracy threshold is user-controlled. These operations are performed with a cost that depends linearly on the dimension d.

The paper is organized as follows. In Section 2 we briefly review separated representations of functions and demonstrate how they give rise to CTDs. In Section 3 we introduce the quadratically convergent algorithm for finding the maximum entry of a tensor in CTD format and discuss the selection of tensor norm for this algorithm. In Section 4 we provide a discussion on using the new algorithm to find the maximum absolute value of continuous functions represented in separated form. In Section 5 we test both the CTD and separated representation optimization algorithms on numerical examples, including a parameterized linear elliptic PDE. We provide our conclusions and a discussion of these algorithms in Section 6.

2. Background

2.1. Separated representation of functions and the canonical tensor decomposition

Separated representations of multivariate functions and operators for computing in higher dimensions were introduced in [3,4]. The separated representation of a function $u(x_1, x_2, \dots, x_d)$ is a natural extension of separation of variables as we seek an approximation

$$u(x_1, \dots, x_d) = \sum_{l=1}^{r} s_l u^{(l)}_1(x_1) \cdots u^{(l)}_d(x_d) + \mathcal{O}\left(\epsilon\right), \tag{1}$$

where $s_l > 0$ are referred to as s-values. In this approximation the functions $u^{(l)}_j(x_j)$, $j = 1, \dots, d$ are not fixed in advance but are optimized in order to achieve the accuracy goal with (ideally) a minimal *separation rank* r. In (1) we set $x_j \in \mathbb{R}$ while noting that in general the variables x_j may be complex-valued or low dimensional vectors. Importantly, a separated representation is not a projection onto a subspace, but rather a nonlinear method to track a function in a high-dimensional space using a small number of parameters. We note that the separation rank indicates just the nominal number of terms in the representation and is not necessarily minimal.

Any discretization of the univariate functions $u^{(l)}_j\left(x_j\right)$ in (1) with $u^{(l)}_{i_j} = u^{(l)}_j\left(x_{i_j}\right)$, $i_j = 1, \dots, M_j$ and $j = 1, \dots, d$, leads to a d-dimensional tensor $\mathbf{U} \in \mathbb{R}^{M_1 \times \cdots \times M_d}$, a canonical tensor decomposition (CTD) of separation rank r_u,

$$\mathbf{U} = U\left(i_1, \dots, i_d\right) = \sum_{l=1}^{r_u} s^u_l \prod_{j=1}^{d} u^{(l)}_{i_j}, \tag{2}$$

where the s-values s^u_l are chosen so that each vector $\mathbf{u}^{(l)}_j = \left\{u^{(l)}_{i_j}\right\}_{i_j=1}^{M_j}$ has unit Euclidean norm $\|\mathbf{u}^{(l)}_j\|_2 = 1$ for all j, l. We can also write the CTD of $\mathbf{U}$ using vectors $\mathbf{u}^{(l)}_j$ and the standard vector outer product $\circ$,

$$\mathbf{U} = \sum_{l=1}^{r_u} s^u_l \mathbf{u}^{(l)}_1 \circ \cdots \circ \mathbf{u}^{(l)}_d.$$

The CTD has become one of the key tools in the emerging field of numerical multilinear algebra (see, e.g., the reviews [15–17]). Given CTDs of two tensors **U** and **V** of separation ranks r_u and r_v, their inner product is defined as

$$\langle \mathbf{U}, \mathbf{V} \rangle = \sum_{l=1}^{r_u} \sum_{l'=1}^{r_v} s_l^u s_{l'}^v \prod_{j=1}^{d} \left\langle \mathbf{u}_j^{(l)}, \mathbf{v}_j^{(l')} \right\rangle, \tag{3}$$

where the inner product $\langle \cdot, \cdot \rangle$ operating on vectors is the standard vector dot product. The Frobenius norm is then defined as $\|\mathbf{U}\|_F = \sqrt{\langle \mathbf{U}, \mathbf{U} \rangle}$. Central to our optimization algorithm is the Hadamard, or point-wise, product of two tensors represented in CTD format. We define this product as,

$$\mathbf{U} * \mathbf{V} = (U * V)(i_1, \dots, i_d) = \sum_{l=1}^{r_u} \sum_{l'=1}^{r_v} s_l^u s_{l'}^v \prod_{j=1}^{d} \left(u_{i_j}^{(l)} \cdot v_{i_j}^{(l')} \right).$$

Common operations involving CTDs, e.g. sum or multiplication, lead to CTDs with separation ranks that may be larger than necessary for a user-specified accuracy. To keep computations with CTDs manageable, it is crucial to reduce the separation rank. The separation rank reduction operation for CTDs, here referred to as τ_ϵ, is defined as follows: given a tensor **U** in CTD format with separation rank r_u,

$$\mathbf{U} = U(i_1, \dots, i_d) = \sum_{l=1}^{r_u} s_l^u \prod_{j=1}^{d} u_{i_j}^{(l)},$$

and user-supplied error ϵ, find a representation

$$\mathbf{V} = V(i_1, \dots, i_d) = \sum_{l'=1}^{r_v} s_{l'}^v \prod_{j=1}^{d} v_{i_j}^{(l')},$$

with lower separation rank, $r_v < r_u$, such that $\|\mathbf{U} - \mathbf{V}\| < \epsilon \|\mathbf{U}\|$.

The workhorse algorithm used for τ_ϵ, i.e. the separation rank reduction problem, is Alternating Least Squares (ALS), which was introduced originally for data fitting as the PARAFAC (PARAllel FACtor) [18] and the CANDECOMP [19] models. ALS has been used extensively in data analysis of (mostly) three-way arrays (see e.g. the reviews [15–17] and references therein). For the experiments in this paper we use exclusively the randomized interpolative CTD tensor decomposition, CTD-ID [20], for τ_ϵ as a faster alternative to ALS.

2.2. Power method for finding the maximum entry of a tensor

The method for finding the maximum entry of a tensor in CTD format in [14] relies on a variation of the matrix power method adapted to tensors. To see how finding entries with maximum absolute value can be cast as an eigenvalue problem, we define the low rank tensor **Y** with 1s at the locations of such entries in **U**, and set $\lambda = \max_{i_1,\dots,i_d} |U(i_1, i_2, \dots, i_d)|$. The Hadamard product,

$$\mathbf{U} * \mathbf{Y} = \lambda \mathbf{Y}, \tag{4}$$

reveals the underlying eigenvalue problem. We describe the algorithm of [14] as Algorithm 1 and use it for performance comparisons.

Algorithm 1

To find the entries with maximum absolute value of a tensor $\mathbf{U} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_d}$ in CTD format, given a tolerance $\epsilon > 0$ for the reduction algorithm τ_ϵ, we define

$\mathbf{Y}_0 = \left(\prod_{i=1}^{d} \frac{1}{M_i} \right) (\mathbf{1}_1 \circ \mathbf{1}_2 \circ \cdots \circ \mathbf{1}_d)$, where $\mathbf{1}_i = (1, 1, \dots, 1)^T \in \mathbb{R}^{M_i}$,
and iterate (up to $k_{\max}$ times) as follows:

```
for $k = 1, 2, \dots, k_{\max}$ do
    1. $\mathbf{Q}_k = \mathbf{U} * \mathbf{Y}_{k-1}$, $\lambda_k = \langle \mathbf{Y}_{k-1}, \mathbf{Q}_k \rangle$, $\mathbf{Z}_k = \mathbf{Q}_k / \|\mathbf{Q}_k\|_F$
    2. $\mathbf{Y}_k = \tau_\epsilon(\mathbf{Z}_k)$
end for
```

The resulting tensor $\mathbf{Y}_k$ ideally has a low separation rank with zero entries except for a few non-zeros indicating the location of the extrema. In the case of a single extremum the tensor $\mathbf{Y}_k$ has separation rank 1.

3. A quadratically convergent algorithm for finding maximum entries of tensors

In order to construct $\mathbf{Y}$ from (4), we simply square all entries of the tensor, normalize the result, and repeat these steps a number of times. Fortunately, the CTD format allows us to square tensor entries using the Hadamard product. These operations increase the separation rank, so the squaring step is (usually) followed by the application of a rank reduction algorithm such as ALS, CTD-ID, or their combination.

Algorithm 2

To find the entries with maximum absolute value of a tensor $\mathbf{U} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_d}$ in CTD format, given a reduction tolerance $\epsilon > 0$ for the reduction algorithm τ_ϵ, we define

```
$\mathbf{Y}_0 = \mathbf{U}/\left\|\mathbf{U}\right\|_F$, and iterate (up to $k_{\max}$ times) as follows:

for $k = 1, 2, \dots, k_{\max}$ do
    1. $\mathbf{Q}_k = \mathbf{Y}_{k-1} * \mathbf{Y}_{k-1}$
    2. $\mathbf{Y}_k = \tau_\epsilon\left(\mathbf{Q}_k\right)$, $\mathbf{Y}_k = \mathbf{Y}_k/\left\|\mathbf{Y}_k\right\|_F$
end for
```

Our main point is the new algorithm has a much faster convergence rate. We now compare convergence rates of the algorithm in [14] and Algorithm 2.

Lemma 3.1. *Consider Algorithms 1 and 2 without using the reduction algorithm (i.e. use $\epsilon = 0$ in τ_ϵ). Then the convergence rate of Algorithm 1 is linear and that of Algorithm 2 is quadratic.*

Proof. Let us identify the largest (by absolute value) entries of a tensor $\mathbf{F}$ as $a = \left|F\left(j_1, j_2, \dots, j_d\right)\right|$ and the next largest entries as b so that the ratio $b/a < 1$. Entries of the tensor in Algorithm 1 decay to zero at the rate $x_m = (b/a)^m$, where m is the number of iterations. Computing the limit

$$\lim_{m\to\infty} \frac{x_{m+1}}{x_m} = \frac{b}{a}$$

confirms the linear rate. In contrast entries of the tensor in Algorithm 2 decay rapidly at the rate $x_m = (b/a)^{2^m}$. Computing the limit

$$\lim_{m\to\infty} \frac{x_{m+1}}{x_m^2} = 1$$

confirms the quadratic rate. □

As a result, the total number of iterations for entries to reach an arbitrarily small finite ε is logarithmic,

$$m \le \log_2\left(\frac{\log\left(\varepsilon\right)}{\log\left(b\right) - \log\left(a\right)}\right).$$

Since the ratio of entries is expected to be reasonably small for most entries, the tensor will rapidly become sparse after only a small number of iterations when using the reduction algorithm τ_ϵ. Since we choose ϵ for the reduction algorithm, we can assume (at least theoretically) that reduction does change the location of the maxima. In practical applications the feasibility of using such an ϵ depends on the original ratio b/a. If the ratio b/a is close to 1 and it is not practical to choose ϵ small enough, then the computed location of the global maximum may be perturbed. However in such cases the algorithm will find values that are close the global maximum and differ by the amount affected by ϵ. In this sense all current algorithms for finding global maximum for non-convex functions have the same problem. However, in our case control over ϵ can help to quantify the difference between computed and actual global maximum. In many practical engineering design problems involving non-convex optimization such an estimation can be useful.

3.1. Termination conditions for Algorithm 2

We have explored three options to terminate the iteration in Algorithm 2. First, the simplest, is to fix the number of iterations in advance provided some knowledge of the gap between the entry with the maximum absolute value and the rest of the entries is available. The second option is to define $\lambda_k = \left\langle \mathbf{Y}_k, \mathbf{U}\right\rangle$ and terminate the iteration once the rate of its decrease becomes small, i.e., $\left(\lambda_k - \lambda_{k-1}\right)/\lambda_{k-1} < \delta$, where $\delta > 0$ is a user-supplied parameter. The third option is to observe the separation rank of $\mathbf{Y}_k$ and terminate the iteration when it becomes smaller than a fixed, user-selected value. In Section 5 we indicate which termination condition is used in our numerical examples.

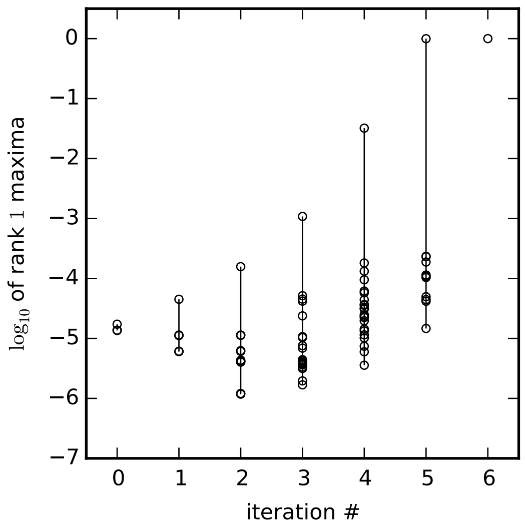

Fig. 1. Comparing the maximum absolute values of entries from rank one terms of a CTD. After 6 iterations Algorithm 2 converges to a rank 1 tensor.

3.2. *Selection of tensor norm*

Working with tensors, we have a rather limited selection of norms that are computable. The usual norm chosen for work with tensors is the Frobenius norm which may not be appropriate in all situations where the goal is to find the maximum absolute value entries. Unfortunately, the Frobenius norm is only weakly sensitive to changes of individual entries. The alternative s-norm (see [20, Section 4]), i.e., the largest s-value of the rank one separated approximation to the tensor in CTD format, is better in some situations. In particular, it allows the user to lower the tolerance to below single precision (within a double precision environment). The improved control over truncation accuracy makes this norm more sensitive to individual entries of the tensor. We note that while it is straightforward to compute the s-value of a rank one separated approximation via a convergent iteration, theoretically it is possible that this computed number is not the largest s-value. However, we have not encountered such a situation using the s-norm in practical problems.

3.3. *Numerical demonstration of convergence*

We construct an experiment using random tensors, with low separation rank ($r = 4$ in what follows), in dimension $d = 6$ with $M = 32$ samples in each direction. The input CTD is formed by adding a rank three CTD and a rank one CTD together. The rank three CTD is formed using random factors ($\mathbf{u}_j^{(l)}$ in (2)) whose samples were drawn from a uniform distribution on $[.9, 1]$ (the s-values were set to 1). The rank one CTD has all zeros except for a spike added in at a random location. The size of this spike is chosen to make the maximum entry have a magnitude of 3.5. For the experiments in this section we set the reduction tolerance of τ_ϵ in Algorithm 2 to $\epsilon = 10^{-6}$ and use the Frobenius norm.

We observe that as the iteration proceeds, the maximum entry of one of the CTD terms eventually becomes dominant relative to the rest. The iteration continues until the rest of the terms are small enough to be eliminated by the separation rank reduction algorithm τ_ϵ. This is illustrated in Fig. 1 where we display the maximum entry from all rank one terms in the CTD for each iteration. Before the first iteration (iteration 0 in Fig. 1), the gaps between the maxima of the rank one terms are not large. As we iterate, the maximum entry of one of the rank one terms rapidly separates itself from the others and, by iteration 6, the reduction algorithm removes all terms except one. This remaining term has an entry 1 corresponding to the location of the maximum entry of the original tensor and all other entries are 0.

Another interesting case occurs when a tensor has multiple maximum entry candidates, either close to or exactly the same in magnitude. To explore this case, we construct an experiment similar to the previous one, namely, find the maximum absolute value of a CTD of dimension $d = 6$ and $M = 32$ (more precisely, $M_j = 32$) samples in each direction. This CTD is constructed with the same rank 3 random CTD as above, but this time we added in two spikes at random locations so that there are two maximal spikes of absolute value 3.5. In cases such as this, Algorithm 2 can be run for a predetermined number of iterations, after which the factors are examined and a subset of candidates can be identified and explicitly verified. For example, in Fig. 2, we observe that by iteration 6 there are only two candidates for the location of the maximum. As a warning we note that allowing the code to continue to run many more iterations will leave only one term due to the finite reduction tolerance ϵ of τ_ϵ in Algorithm 2. Indeed, in Fig. 2 we observe the maximum entries of the two rank one terms begin to split around iteration number 26, and by iteration 37 only one term remains. This example illustrates a potential danger of insisting on a single term termination condition.

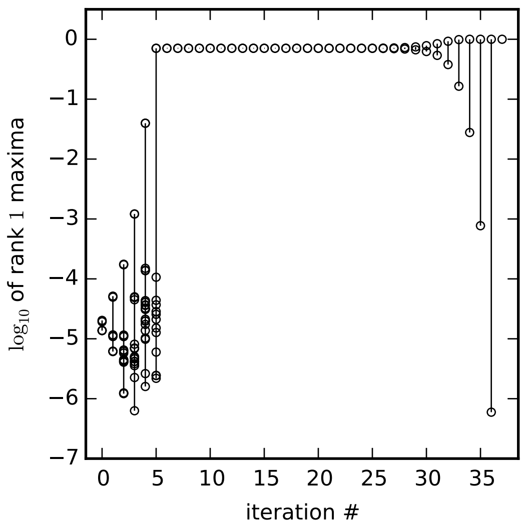

Fig. 2. In this example there are two maxima of the same size. The values at these locations converge to $1/\sqrt{2}$ and the rest of the entries converge to 0. If allowed to run long enough, the value at one of the locations becomes dominant due to the truncation error of the rank reduction algorithm. The other location is then eliminated by the algorithm and only one term remains.

4. A discussion on CTD-based global optimization of multivariate functions

We describe an approach using Algorithm 2 for the global optimization of functions that admit low rank separated representations. We consider two types of problems: first, where the objective function is already in a separated form, ready to yield a CTD appropriate for use with Algorithm 2. The second, more general problem, is to construct a separated representation given data or an analytic expression of the function, from which a CTD can then be built.

In the first problem the objective function is already represented in separated form as in (1), and an interpolation scheme is associated with each direction. Such problems have been subject to growing interest due to the use of CTDs for solving operator equations, e.g., deterministic or stochastic PDE/ODE systems (see, for example, [3,4,7,13,6,21]), where the solution is directly approximated in a low rank separated format. The assumption here is that such a representation satisfies certain convergence criteria, for instance, related to the residual. The practitioner will then be interested in finding the global extremum of this approximate solution as a proxy to that of the true solution.

In the second problem the goal is to optimize a multivariate function where no separated representation is readily available. What typically is available is a data set of function values (scattered or on a grid). In such cases our recourse is to first construct a separated representation of the underlying multivariate objective function and then re-sample it in order to obtain a CTD of the form (2). The construction of a separated representation given a set of function values can be thought of as a regression problem (see [5,9]). The algorithms in these papers use function samples to build a separated representation of form (1). Following [5], an interpolation scheme is set up in each direction and the ALS algorithm is used to reduce the problem to regression in a single variable. The univariate interpolation scheme may use a variety of possible bases, as well as nonlinear approximation techniques. Statistical tools, such as cross-validation, may be used to enhance the stability of the regression problem and avoid over-fitting of the separated representation to the data [9]. Once a separated representation is constructed, the CTD for finding the global maxima is then obtained to satisfy the local interpolation requirement stated below. The resulting CTD is then used as input into Algorithm 2 to produce an approximation of the global maxima. An example of this approach is included in Section 5.3 below.

If the objective function is given analytically, it is preferable to use analytic techniques to approximate the original function via a separated representation. In such cases, sampling the objective function becomes an important issue when finding global maxima of a function via Algorithm 2. The key sampling requirement is to ensure that the objective function can be interpolated at an arbitrary point from its sampled values up to a desired accuracy. This implies that sampling rate must depend on the local behavior of the function, e.g., the sampling rate is greater near singularities. For example, if a function has algebraic singularities, an efficient method to interpolate is to use wavelet bases. In such a case we emphasize the importance of using interpolating scaling functions, i.e., scaling functions whose coefficients are function values or are simply related to those. Since the global maxima are typically searched for in finite domains, the multiresolution basis should also work well in an interval (or a box in higher dimensions). This leads us to suggest the use of Lagrange interpolating polynomials within an adaptive spatial subdivision framework. We note that rescaled Lagrange interpolating polynomials form an orthonormal basis on an interval and the spatial subdivision is available as multiwavelet bases (see e.g., [22,23]).

We highlight that finding the maximum entries of a CTD yields only an approximation of the maxima and their locations for the function. In particular, as in all surrogate-based optimization techniques, the accuracy of the maxima and their locations identified from CTD cannot be generally guaranteed in the second problem type described above. This is because the

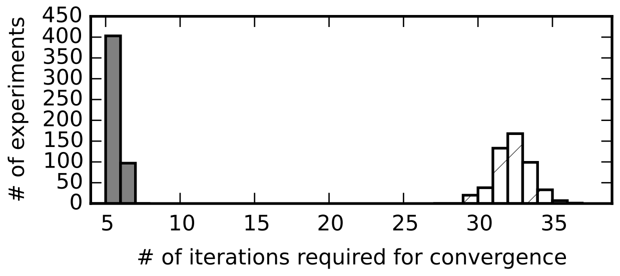

Fig. 3. Histograms illustrating the number of iterations required for convergence to a rank one CTD by the power method optimization (hatched pattern) and the squaring method optimization (solid gray).

construction of surrogates, including CTDs, from function values generally relies on minimizing some norm of the mismatch between the surrogate and the function at sample points, without any consideration or guarantee on the value and location of the surrogate extrema. Due to the interpolation requirements stated above the constructed CTDs are high-quality approximations so that, if needed, a local optimization algorithm can be used as a last step for finding the true global maxima. However, if the location of the true maxima is outside the local search domain, no improvement can be achieved. Such approaches are among widely-used methods to improve the quality of surrogate-based optimization with relatively smaller number of original function evaluations, see, e.g., [24–28]. Since in this case we use local optimization, it is preferable to oversample the CTD representation of a function, i.e. to use large M_j's in (2). This will not result in a prohibitive cost as the rank reduction algorithms, e.g. ALS, that operate on CTDs scale linearly with respect to the number of samples in each direction [4]. However, the total number of samples needed to satisfy the local interpolation requirement may be large and, thus, additional steps to accelerate the reduction algorithm τ_ϵ in Algorithm 2 may be required. One such technique consists of using the *QR* factorization as explained in [20, Section 2.4].

An example of applying analytic techniques to put a function in separated form and subsequently using Algorithm 2 to optimize the separated function is provided in Section 5.2.

5. Numerical examples

5.1. Comparison with power method algorithm from Section 2.2

To test Algorithm 2, we construct tensors from random factors and find the entries with the largest absolute value. For these examples we select dimension $d = 8$ and set $M = 32$ samples in each direction. The test CTDs are constructed by adding rank four and rank one CTDs together. The rank four CTDs are composed of random factors (vectors $\mathbf{u}_j^{(l)}$ in (2)) whose entries are drawn from the uniform distribution on $[.9, 1]$ and the corresponding s-values are set to 1. The rank one CTDs contain all zeros except for a magnitude 4 spike added in at a random location. The CTDs in these examples have a unique maximum entry, and we chose to terminate the algorithm when the output CTD reaches separation rank one. We run tests for both Algorithm 2 and Algorithm 1, using for the rank reduction operation τ_ϵ the CTD-ID algorithm with accuracy threshold $\epsilon = 10^{-6}$ in the s-norm.

A histogram displaying the required number of iterations to converge for 500 individual tests is shown in Fig. 3. While both algorithms find the correct maximum location in all tests, Algorithm 2 outperforms Algorithm 1 in terms of the number of iterations required. While Algorithm 2 requires much fewer iterations than the power method for the same problem, a lingering concern may be that the squaring process in Algorithm 2 requires the reduction of CTDs with much larger separation ranks. In Fig. 4 we show histograms of computation times. We observe that despite reducing larger separation rank CTDs, the computation times using Algorithm 2 are significantly smaller. These computation times are for our MATLAB code running on individual cores of a Intel Xeon 2.40 GHz processor.

5.2. Optimization example

As an example of applying analytic techniques to put a function in separated form for the purpose of optimization, we consider Ackley's test function [29], commonly used for testing global optimization algorithms,

$$u(\mathbf{x}) = a\, e^{-b\left(\frac{1}{d}\sum_{i=1}^{d} x_i^2\right)^{1/2}} + e^{\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)}, \tag{5}$$

where d is the number of dimensions and a, b, and c are parameters. For our tests we set $d = 10$, $a = 20$, $b = 0.2$, and $c = 2\pi$. We choose this test function for two reasons: first, it has many local maxima, thus a local method without a good initial guess may not find the correct maximum. Second, one of the terms is not in separated form, hence we must construct a separated approximation of the function first.

The true maximum of (5) occurs at $\mathbf{x} = \mathbf{0}$ with a value of $u(\mathbf{0}) = a + e$. The first term in (5) is radially symmetric which we approximate with a linear combination of Gaussians using the approximation of e^{-xy} from [30, Section 2.2],

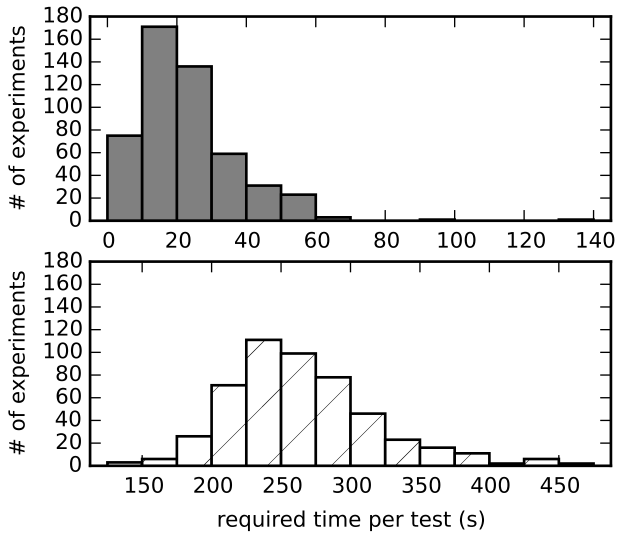

Fig. 4. Histograms illustrating the computation times, in seconds, required for convergence to a rank one CTD by the power method optimization (hatched pattern) and the squaring method optimization (solid gray).

$$G_e(x) = \frac{hb}{2\sqrt{\pi d}} \sum_{j=0}^{R} \exp\left(-\frac{b^2}{4d} e^{s_j} - x^2 e^{s_j} + \frac{1}{2} s_j\right), \tag{6}$$

where $s_j = s_{start} + jh$, and h, s_{start}, and R are parameters chosen such that given $\epsilon, \delta > 0$,

$$\left| G_e(x) - e^{-\frac{b}{\sqrt{d}}x} \right| \leq \epsilon_G \tag{7}$$

for $0 < \delta < x < \infty$. We arrive at the approximation of the first term in (5),

$$\left| e^{-b\left(\frac{1}{d}\sum_{i=1}^{d} x_i^2\right)^{1/2}} - \sum_{j=0}^{R} w_j \prod_{i=1}^{d} \exp\left(-x_i^2 e^{s_j}\right) \right| \leq \epsilon_G \tag{8}$$

where $w_j = \frac{hb}{2\sqrt{\pi d}} \exp\left(-\frac{b^2}{4d} e^{s_j} + \frac{1}{2} s_j\right)$.

To correctly sample (5), we first consider two terms of the function separately, the radially-symmetric exponential term, and the exponential cosine term which is already in separated form, $\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) = \prod_{i=1}^{d} \exp\left(\frac{1}{d}\cos(cx_i)\right)$. These terms require different sampling strategies. The sampling rate of the radially-symmetric exponential term near the origin should be significantly higher than away from it. On the other hand, uniform sampling is sufficient for the exponential cosine term. We briefly discuss the combined sampling of these terms to satisfy both requirements.

To sample the radially-symmetric exponential term in (5), we use the expansion by Gaussians (6) as a guide. We choose 10 equally-spaced points centered at zero and sample the Gaussian e^{-x^2}. For each Gaussian in (6), we scale these points by the factor $e^{s_j/2}$ in order to consistently sample each Gaussian independent of its scale. Since (6) includes a large number of rapidly decaying Gaussians, too many of the resulting samples will concentrate near the origin. Therefore, we select a subset as follows: first, we keep all samples from the sharpest Gaussian, x_{0l}, $l = 1, \ldots, 10$. We then take the samples from second sharpest Gaussian and keep only those outside the range the previous samples, i.e., we keep x_{1m} such that $|x_{1m}| > \max_l |x_{0l}|$, $m = 1, \ldots, 10$. Repeating this process for all terms $j = 0, 1, \ldots, R$ in expansion (6) yields samples for the radially-symmetric exponential term in (5).

We oversample the exponential cosine term using 16 samples per oscillation. Notice that this sampling strategy is not sufficient for the radially-symmetric exponential term in (5) (and vice-versa). To combine these samples, we first find the radius where the sampling rate of the radially-symmetric exponential term in (5) drops below the sampling rate chosen to sample the exponential cosine term. We then remove all samples whose coordinates are greater in absolute value than the radius, and replace them with samples at the chosen rate of 16 samples per oscillation. The resulting set of samples consists of $M = 1080$ points for each direction $j = 1, \ldots, 10$.

Using expansion (6) leads to a 77 term CTD representation for the chosen accuracy $\epsilon_G = 10^{-8}$ and parameter $\delta = 3 \cdot 10^{-6}$ in (7). An initial reduction of this CTD using the CTD-ID algorithm with accuracy threshold $\epsilon = 10^{-6}$ in the Frobenius norm produces a CTD with separation rank 11 representing the function $u(\mathbf{x})$. We use this CTD as an input into Algorithm 2, yielding a rank one solution after 24 iterations. We set the reduction tolerance in Algorithm 2 for this experiment to

$\epsilon = 10^{-6}$ and find the maximum entry of the tensor located at a distance 2.13×10^{-3} away from the function's true maximum location, the origin. Using the output as the initial guess in a local, gradient-free, optimization algorithm (in our case a compass search, see e.g. [31]) yields the maximum value with relative error of 1.03×10^{-7} located at a distance 1.84×10^{-6} away from the origin.

5.3. PDE optimization example

As an example of constructing a separated representation from function evaluations for the purpose of optimization, we consider the problem of finding the maximum of the solution $u(\mathbf{x}, \mathbf{z})$ to the parameterized linear elliptic PDE

$$\begin{aligned} -\nabla \cdot (a(\mathbf{x}, \mathbf{z})\nabla u(\mathbf{x}, \mathbf{z})) &= 1, \quad \mathbf{x} \in \mathcal{D}, \\ u(\mathbf{x}, \mathbf{z}) &= 0, \quad \mathbf{x} \in \partial\mathcal{D}, \end{aligned} \tag{9}$$

defined on the unit square $\mathcal{D} = (0, 1) \times (0, 1)$ with boundary $\partial\mathcal{D}$. This problem represents a generic model for diffusive processes and is widely used to study the effect of parameter uncertainty in applications such as subsurface flows, heat conduction, and chemical transport.

The diffusion coefficient $a(\mathbf{x}, \mathbf{z})$ is parametric and following [32] is modeled by

$$a(\mathbf{x}, \mathbf{z}) = a_0 + \sigma_a \sum_{k=1}^{d} \sqrt{\zeta_k}\varphi_k(\mathbf{x})z_k, \tag{10}$$

where $\mathbf{z} = (z_1, \ldots, z_d)$ and the parameters z_k are distributed over the interval $[-1, 1]$, and we choose $a_0 = 0.1$, $\sigma_a = 0.01$, and $d = 20$. In (10), $\{\zeta_k\}_{k=1}^{d}$ are the d largest eigenvalues associated with $\{\varphi_k\}_{k=1}^{d}$, the $L_2(\mathcal{D})$-orthonormalized eigenfunctions of the kernel $C_{aa}(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|_1/l_c)$, where l_c is set to $l_c = 2/3$. Given the choices of parameters a_0, σ_a, d, and l_c, the model in (10) leads to strictly positive realizations of $a(\mathbf{x}, \mathbf{z})$.

The objective in this example is to approximate $u(\mathbf{x}, \mathbf{z})$ at $\mathbf{x}^* = (0.4839, 0.4839)$ as a function of $\mathbf{z}$ and find the maximum value of $u(\mathbf{x}^*, \mathbf{z})$ to study the worst-case scenario response of the system in the presence of parameter variations. In the following experiments we take two different approaches to optimizing $u(\mathbf{x}^*, \mathbf{z})$: The first approach is to construct a separated representation of $u(\mathbf{x}^*, \mathbf{z})$, evaluate it on a tensor grid yielding a CTD, and find the maximum via Algorithm 2. For comparison, we also use a genetic algorithm, a representative approach to finding the maximum of non-convex functions. Both methods require evaluation of $u(\mathbf{x}^*, \mathbf{z})$ (at a fixed $\mathbf{z} \in [-1, 1]^{20}$) by solving (9). In this example we use a finite element solver with a triangular mesh of uniform size $h = 1/32$.

In our approach, we construct a separated representation by first computing 600 realizations of $u(\mathbf{x}^*, \mathbf{z})$ via solution of (9). Using the resulting data points, we estimate $u(\mathbf{x}^*, \mathbf{z})$ in separated form using the method from [5,9]. The resulting function has separation rank $r = 5$, where the factors $u_j^{(l)}(z_j)$ in (1) are represented in a Legendre basis of polynomials of degree at most three. We then validate the separated representation by computing 3400 additional samples of $u(\mathbf{x}^*, \mathbf{z})$; the relative RMS validation error is 3.256×10^{-4}, which is within the accuracy of the finite element solver.

Using the separated representation, we construct a CTD with 32 samples equispaced in $[-1, 1]$ for each direction. We then use Algorithm 2 (with an ALS tolerance of $\epsilon = 10^{-3}$ to keep the separation rank relatively low) to find the maximum entry of the CTD. As a baseline for comparing the output from Algorithm 2, we use the genetic algorithm from the MATLAB Optimization Toolbox with the recommended settings.

To compare the performance of Algorithm 2 with the genetic algorithm, we conducted 500 tests of each approach since both algorithms have random components. Algorithm 2 uses random rank updates during the ALS reduction procedure, while the genetic algorithm uses different initial populations for each test. The largest maximum found by Algorithm 2 in the 500 tests was 8.744×10^{-1}, with mean 8.731×10^{-1} and standard deviation 5.7×10^{-4}. For comparison, the largest maxima found by the genetic algorithm in these tests was 8.750×10^{-1}, with mean 8.747×10^{-1} and standard deviation 3.6×10^{-4}. Taking into account that the accuracy of solving (9) and the RMS validation error of the separated representation are both of the same order as the computed standard deviations, the results obtained from the two algorithms are consistent.

On the other hand, an important difference between the two methods is the number of function evaluations required to find the maximum, in our case solving (9) once for each function evaluation. While in the 500 experiments described above the genetic algorithm required on average 4.419×10^4 solutions of (9) to obtain the maximum, the separated representation of $u(\mathbf{x}^*, \mathbf{z})$ was constructed using only 600 solutions of (9). Algorithm 2 required on average of 34 iterations to compute the maximum, whereas the largest number of iterations required was 43.

It is well known that genetic algorithms require a large number of function evaluations. In practical applications computing a large number of solutions of PDEs may be impractical. In such applications it is quite common to construct so-called surrogate models to reduce computational costs (see, e.g. [33]). For example, one may use a separated representation as such a surrogate, and then optimize via genetic algorithm. However, even with this modification, the advantage of Algorithm 2 over genetic algorithms (and other global search methods) is linear scaling of its complexity with the dimension d.

6. Discussion and conclusions

Entries of a tensor in CTD format with the largest absolute value can be reliably identified by the new quadratically convergent algorithm, Algorithm 2. This algorithm, in turn, can be used for global optimization of multivariate functions that admit separated representations. Unlike algorithms based on random search strategies (see e.g. [1]), our approach allows the user to estimate the potential error of the result. The computational cost of Algorithm 2 is dominated by the cost of reducing the separation ranks of intermediate CTDs, not the dimension of the optimization problem. If ALS is used to reduce the separation rank and d is the dimension, r is the maximum separation rank (after reductions) during the iteration, and M is the maximum number of components in each direction, then the computational cost of each rank reduction step can be estimated as $\mathcal{O}\left(r^4 \cdot M \cdot d \cdot N_{iter}\right)$, where N_{iter} is the number of iterations required by the ALS algorithm to converge. The computational cost of the CTD-ID algorithm is estimated as $\mathcal{O}\left(r^3 \cdot M \cdot d\right)$ and, if it is used instead of ALS, the reduction step is faster by a factor of $\mathcal{O}\left(r \cdot N_{iter}\right)$ [20]. We note that, while linear in dimension, Algorithm 2 may require significant computational resources due to the cubic (or quartic for ALS) dependence on the separation rank r.

Finally, we note that the applicability of our approach to global optimization extends beyond the use of separated representations of multivariate functions or tensors in CTD format. In fact, any structural representation of a multivariate function that allows rapid computation of the Hadamard product of corresponding tensors and has an associated algorithm for reducing the complexity of the representation should work in a manner similar to Algorithm 2.

References

[1] R. Horst, P.M. Pardalos, Handbook of Global Optimization, Vol. 2, Springer Science & Business Media, 2013.
[2] J. Spall, Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, vol. 65, John Wiley & Sons, 2005.
[3] G. Beylkin, M.J. Mohlenkamp, Numerical operator calculus in higher dimensions, Proc. Natl. Acad. Sci. USA 99 (16) (2002) 10246–10251.
[4] G. Beylkin, M.J. Mohlenkamp, Algorithms for numerical analysis in high dimensions, SIAM J. Sci. Comput. 26 (6) (2005) 2133–2159.
[5] G. Beylkin, J. Garcke, M.J. Mohlenkamp, Multivariate regression and machine learning with sums of separable functions, SIAM J. Sci. Comput. 31 (3) (2009) 1840–1857, http://dx.doi.org/10.1137/070710524.
[6] F. Chinesta, P. Ladeveze, E. Cueto, A short review on model order reduction based on proper generalized decomposition, Arch. Comput. Methods Eng. 18 (4) (2011) 395–404.
[7] A. Doostan, G. Iaccarino, A least-squares approximation of partial differential equations with high-dimensional random inputs, J. Comput. Phys. 228 (12) (2009) 4332–4345.
[8] A. Doostan, G. Iaccarino, N. Etemadi, A Least-Squares Approximation of High-Dimensional Uncertain Systems, Annual Research Brief, Center for Turbulence Research, Tech. Rep., Stanford University, 2007.
[9] A. Doostan, A. Validi, G. Iaccarino, Non-intrusive low-rank separated approximation of high-dimensional stochastic models, Comput. Methods Appl. Mech. Eng. 263 (2013) 42–55.
[10] M. Hadigol, A. Doostan, H. Matthies, R. Niekamp, Partitioned treatment of uncertainty in coupled domain problems: a separated representation approach, Comput. Methods Appl. Mech. Eng. 274 (2014) 103–124.
[11] A. Nouy, A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations, Comput. Methods Appl. Mech. Eng. 196 (37–40) (2007) 4521–4537.
[12] A. Nouy, Generalized spectral decomposition method for solving stochastic finite element equations: invariant subspace problem and dedicated algorithms, Comput. Methods Appl. Mech. Eng. 197 (2008) 4718–4736.
[13] A. Nouy, Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems, Arch. Comput. Methods Eng. 17 (2010) 403–434.
[14] M. Espig, W. Hackbusch, A. Litvinenko, H. Matthies, E. Zander, Efficient analysis of high dimensional data in tensor formats, in: Sparse Grids and Applications, Springer, 2013, pp. 31–56.
[15] R. Bro, PARAFAC. Tutorial and applications, Chemom. Intell. Lab. Syst. 38 (1997) 149–171.
[16] G. Tomasi, R. Bro, A comparison of algorithms for fitting the PARAFAC model, Comput. Stat. Data Anal. 50 (7) (2006) 1700–1734.
[17] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500, http://dx.doi.org/10.1137/07070111X.
[18] R.A. Harshman, Foundations of the Parafac Procedure: Model and Conditions for an "Explanatory" Multi-Mode Factor Analysis, Working Papers in Phonetics, vol. 16, UCLA, 1970.
[19] J.D. Carroll, J.J. Chang, Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart–Young decomposition, Psychometrika 35 (1970) 283–320.
[20] D.J. Biagioni, D. Beylkin, G. Beylkin, Randomized interpolative decomposition of separated representations, J. Comput. Phys. 281 (2015) 116–134.
[21] H. Cho, D. Venturi, G. Karniadakis, Numerical methods for high-dimensional probability density function equations, J. Comput. Phys. 305 (2016) 817–837.
[22] B. Alpert, A class of bases in L^2 for the sparse representation of integral operators, SIAM J. Math. Anal. 24 (1) (1993) 246–262.
[23] B. Alpert, G. Beylkin, D. Gines, L. Vozovoi, Adaptive solution of partial differential equations in multiwavelet bases, J. Comput. Phys. 182 (1) (2002) 149–190.
[24] J.W. Bandler, Q.S. Cheng, S.A. Dakroury, A.S. Mohamed, M.H. Bakr, K. Madsen, J. Sondergaard, Space mapping: the state of the art, IEEE Trans. Microw. Theory Tech. 52 (1) (2004) 337–361.
[25] N.V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P.K. Tucker, Surrogate-based analysis and optimization, Prog. Aerosp. Sci. 41 (1) (2005) 1–28.
[26] Z. Zhou, Y.S. Ong, P.B. Nair, A.J. Keane, K.Y. Lum, Combining global and local surrogate models to accelerate evolutionary optimization, IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. 37 (1) (2007) 66–76.
[27] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evol. Comput. 14 (3) (2010) 329–355.
[28] Y. Jin, Surrogate-assisted evolutionary computation: recent advances and future challenges, Swarm Evol. Comput. 1 (2) (2011) 61–70.
[29] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.
[30] G. Beylkin, L. Monzón, Approximation of functions by exponential sums revisited, Appl. Comput. Harmon. Anal. 28 (2) (2010) 131–149.

[31] T.G. Kolda, R. Lewis, V. Torczon, Optimization by direct search: new perspectives on some classical and modern methods, SIAM Rev. 45 (3) (2003) 385–482.
[32] M. Reynolds, A. Doostan, G. Beylkin, Randomized alternating least squares for canonical tensor decompositions: Application to a PDE with random data, SIAM J. Sci. Comput. 38 (5) (2017) A2634–A2664.
[33] Y. Ong, P. Nair, A. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling, AIAA J. 41 (4) (2003) 687–696.