Computer Physics Communications 261 (2021) 107663

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

# Practical algorithms for multivariate rational approximation

Anthony P. Austin [a], Mohan Krishnamoorthy [b,*], Sven Leyffer [b], Stephen Mrenna [c], Juliane Müller [d], Holger Schulz [e]

[a] Naval Postgraduate School, Monterey, CA 93943, United States of America
[b] Argonne National Laboratory, Lemont, IL, 60439, United States of America
[c] Fermi National Accelerator Laboratory, Batavia, IL, 60510, United States of America
[d] Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, United States of America
[e] University of Cincinnati, Cincinnati, OH 45219, United States of America

ARTICLE INFO

*Article history:*
Received 6 August 2019
Received in revised form 4 September 2020
Accepted 6 October 2020
Available online 22 October 2020

*Keywords:*
Discrete least-squares
Multivariate rational approximation
Semi-infinite optimization
Surrogate modeling

ABSTRACT

We present two approaches for computing rational approximations to multivariate functions, motivated by their effectiveness as surrogate models for high-energy physics (HEP) applications. Our first approach builds on the Stieltjes process to efficiently and robustly compute the coefficients of the rational approximation. Our second approach is based on an optimization formulation that allows us to include structural constraints on the rational approximation (in particular, constraints demanding the absence of singularities), resulting in a semi-infinite optimization problem that we solve using an outer approximation approach. We present results for synthetic and real-life HEP data, and we compare the approximation quality of our approaches with that of traditional polynomial approximations.

## 1. Introduction

Optimization problems arising in complex science and engineering applications often involve simulations that are computationally expensive to evaluate (several minutes to hours or more per evaluation). The simulations are usually nonlinear and black box; we have no analytic description of the function $f(\cdot)$ that maps the parameter inputs $x \in D \subset \mathbb{R}^n$ to simulation outputs. The computational expense limits the number of evaluations we can do during the optimization. A widely used approach to mitigate this difficulty is to use a fast-to-evaluate *surrogate model*, $s(x)$, as a proxy for the simulation [1]: $f(x) = s(x) + e(x)$, where $e(x)$ denotes the difference between the true function and the surrogate model. We fit a surrogate model based on a set of pre-evaluated parameter–function value pairs and use it during the optimization search, thus reducing the number of queries to the expensive simulation. Types of surrogate models include Gaussian process models [2], radial basis functions [3], multivariate adaptive regression splines [4], and polynomial regression models [5].

Polynomial models have several advantages, such as a simple representation and being easy to build and use; however, they have poor extrapolation behavior and are severely limited in their ability to cope with singularities. These drawbacks can reduce their effectiveness at representing elements of the physics in many applications. Because of these drawbacks, one turns to models based on rational functions (quotients of polynomials), whose ability to capture singularities naturally via their poles can make them considerably more powerful than polynomials [6,7]. Unfortunately, rational approximations can be numerically fragile to compute and are prone to having spurious singularities. Moreover, how to select the appropriate combination of numerator and denominator degree is not always clear.

In this article, we investigate the utility of rational approximations as surrogate models. We propose two methods for computing multivariate rational approximations $r(x) = p(x)/q(x)$. The first approach is based on the univariate methods of [8,9] and provides a robust and efficient way to compute the coefficients of $p(x)$ and $q(x)$. Although it tries to reduce the propensity for the resulting $r(x)$ to contain unwanted singularities by using ideas from linear algebra to minimize the degree of $q(x)$, it does not *guarantee* that $r(x)$ will be pole free in the parameter domain.

The second approach uses a constrained optimization formulation that includes structural constraints on $r(x)$ to enforce the absence of poles in $D$. These constraints are motivated from applications that arise, for example, in high-energy physics (HEP) simulations. Although it is computationally more expensive than the first approach, the second approach allows us to guarantee that the computed approximation is free of poles for box-shaped parameter domains, which can be crucial in the context of computing surrogate models for use in optimization. In particular, the guaranteed absence of poles ensures that subsequent optimization problems involving our rational approximations are well-defined.

[*] Corresponding author.
*E-mail address:* mkrishnamoorthy@anl.gov (M. Krishnamoorthy).

https://doi.org/10.1016/j.cpc.2020.107663
0010-4655/Published by Elsevier B.V.

### 1.1. Previous work on rational approximation

The literature on rational approximation is too vast to cover comprehensively here; we refer the reader to standard texts such as [10, Ch. V], [11, Ch. 5], and [12, Ch. 23–27] for the history and basic concepts. In this article, we are concerned with multivariate models. Multivariate rational approximation has been studied extensively by Cuyt and co-authors [13,14]; in particular, Cuyt and Yang have recently developed practical error bounds [15].

Our rational approximations are least-squares models, which can be formulated in a nonlinear or linearized fashion as we describe below. Our first approach is an extension of the algorithms for the linearized problem presented in [8,9,16]. The nonlinear problem is an example of a *separable* nonlinear least-squares problem, and algorithms for it often exploit this structure. Examples include the Gauss–Newton algorithm developed by Golub and Pereyra [17] and the full-Newton algorithm of Borges [18]. The so-called "AAA" algorithm of [19] is a particularly interesting recently developed alternative to traditional methods for rational least-squares problems; however, as it currently only works in the univariate context, we do not explore it further here. Of course, least-squares approximations are not the only type of rational model. Recent work of interest on rational models other than simple least-squares approximations includes the rational minimax approximation algorithm of [20].

One of the appeals of a least-squares approach to rational approximation is that it is naturally robust to noise in the data being fit. Cuyt, Salazar Celis, and co-authors [21–23] have developed an alternative approach to rational approximation in the face of noisy or uncertain data that takes as its input a set of uncertainty intervals for each datum and then solves a linear or quadratic programming problem to find a rational function that passes through all uncertainty intervals. Our method is both different from this approach and better-suited to our application. In our context, it is not clear how to construct the uncertainty intervals this alternative method requires. Moreover, it expends additional effort (solving an optimization problem) compared with our approach based on linear algebra without delivering the pole-free guarantee that our semi-infinite optimization provides.

Our semi-infinite optimization approach is a multivariate form of constrained-denominator rational approximation, which has been studied extensively in the univariate context; see, e.g., [24, 25]. In [25], the authors prove the existence of best (uniform) rational approximations with a lower bound on the denominator and bounded denominator coefficients. The authors also present a restricted denominator differential correction algorithm and shows that it converges under certain conditions. In [24], the author considers both lower and upper bounds on the denominator. He shows these bounds to be equivalent to convex constraints on the denominator coefficients, hence demonstrating the existence of a best solution. Our formulation of the constrained-denominator problem using semi-infinite optimization offers a different perspective than this earlier work. In addition to being practical, it may yield theoretical insight as well, though we do not explore this in detail here. For instance, by invoking semi-infinite optimality conditions (see, e.g., [26]), we could, in principle, obtain necessary conditions for optimal pole-free rational approximations. This would partially answer an open question from [24] concerning characterizations of best constrained rational approximations.

### 1.2. High-energy physics motivation

Our work is motivated by simulations for studying complex physical phenomena, especially in high-energy physics. Simulations are often used to guide real-world experiments in order to find "interesting" physics or to verify that models derived from physical understanding are in agreement with experiments [27]. However, these simulations (as well as physics experiments) are generally resource intensive (computationally or otherwise) [28]. A single simulation may require many hours of compute time on a modern supercomputer, thus limiting the number of simulation runs that can realistically be done.

This severely limits applications that require extensive parameter space exploration. Our aim is to replace the costly simulations with rational approximations that are much cheaper to evaluate. In particular, we want to construct and numerically optimize an objective function over a space of model parameters that is defined as the mismatch between experimental data and simulation predictions.

### 1.3. Outline of the paper

In Section 2, we establish our notation and describe the types of models that we will generate. In Section 3, we devise a method for constructing rational models based on linear algebra. This approach is flexible and easy to implement, but it has the drawback that singularities may be present. Although singularities are acceptable in some contexts, we generally have to prevent singularities in particular regions of the parameter space because they may cause an unbounded objective function in our optimization procedure, which is not acceptable. In Section 4, we describe a separate approach based on semi-infinite optimization that allows us to achieve this goal. In Section 5, we present some numerical results, and in Section 6 we describe our high-energy physics application and show the superior performance of our pole-free rational approximations over polynomial approximations and rational approximations with poles. In Section 7, we summarize our key findings and discuss potential avenues for further research.

## 2. Notation and setup

We denote by $n$ the number of parameters in our model, and our generic variables are $x_1, \ldots, x_n$. By the *degree* of an $n$-variate monomial $x_1^{i_1} \cdots x_n^{i_n}$, we mean its *total degree*, in other words, the sum $i_1 + \cdots + i_n$, as distinguished from its *maximal degree* $\max(i_1, \ldots, i_n)$. The degree of an $n$-variate polynomial is the maximum of the degrees of its constituent monomials. We write $\mathcal{P}_d^n$ for the space of all $n$-variate polynomials of degree at most $d$; this is a real vector space of dimension $\alpha(d) = \binom{n+d}{d}$.

Let $x^{(0)}, \ldots, x^{(K-1)}$ be $K$ points in $\mathbb{R}^n$, and let $f_0, \ldots, f_{K-1}$ be the corresponding real data values. Our aim is to find an $n$-variate rational function $r(x) = p(x)/q(x)$ with $p \in \mathcal{P}_M^n$ and $q \in \mathcal{P}_N^n$ such that $r(x^{(k)}) \approx f_k$ for each $k$. One natural approach is to choose $p$ and $q$ to solve the discrete least-squares problem

$$\underset{p,q}{\text{minimize}} \quad \sum_{k=0}^{K-1} \left( \frac{p(x^{(k)})}{q(x^{(k)})} - f_k \right)^2 \quad \text{subject to} \quad p \in \mathcal{P}_M^n, q \in \mathcal{P}_N^n, \tag{1}$$

but the nonlinearity in $q$ makes this problem challenging. It is usually easier to work with the linearized problem

$$\underset{p,q}{\text{minimize}} \quad \sum_{k=0}^{K-1} \left(p\left(x^{(k)}\right) - f_k q\left(x^{(k)}\right)\right)^2 \quad \text{subject to} \quad p \in \mathcal{P}_M^n, \quad q \in \mathcal{P}_N^n, \tag{2}$$

which is the formulation we will use in the following. Note that the solutions to Eqs. (1) and (2) do not generally coincide.

As written, Eqs. (1) and (2) are incompletely specified. The objective in Eq. (1) depends only on the ratio of $p$ to $q$, and additional normalization conditions must be imposed to pin down the solution. Likewise, a normalization condition is needed in Eq. (2) to exclude the trivial solution $p = q = 0$. We will address these issues in detail in later sections.

## 3. Multivariate rational models via linear algebra

Our first approach to constructing rational models is based on ideas from linear algebra following [8,9]; see also [29] and [12, Ch. 26]. We extend the method of these references to the multivariate case. One such extension has been proposed in [16]; our method may be viewed as a generalization of that extension to handle situations in which the data used to construct the model come from arbitrary sample points in the parameter space instead of from a tensor product grid.

### 3.1. Basic algorithm

The basic idea is as follows. Let $L = \max(M, N)$. Given a basis $\varphi_0, \ldots, \varphi_{\alpha(L)-1}$ for $\mathcal{P}_L^n$, consider the Vandermonde-like matrices $V_M \in \mathbb{R}^{K\times\alpha(M)}$ and $V_N \in \mathbb{R}^{K\times\alpha(N)}$ whose $(k, j)$ entries are $\varphi_j(x^{(k)})$. Express $p$ and $q$ as

$$p(x) = \sum_{j=0}^{\alpha(M)-1} a_j\varphi_j(x), \qquad q(x) = \sum_{j=0}^{\alpha(N)-1} b_j\varphi_j(x),$$

and gather the coefficients $a_j$, $b_j$ into vectors $a \in \mathbb{R}^{\alpha(M)}$ and $b \in \mathbb{R}^{\alpha(N)}$, respectively. Let $F = \operatorname{diag}(f_0, \ldots, f_{K-1})$. Then, since the $k$th entries of $V_M a$ and $V_N b$ are $p(x^{(k)})$ and $q(x^{(k)})$, respectively, the linearized problem Eq. (2) may be rewritten as the following linear least-squares problem to find coefficients, $a$, $b$, that

$$\underset{a,b}{\text{minimize}} \quad \|V_M a - FV_N b\|_2^2. \tag{3}$$

Just as Eq. (2) has the trivial solution $p = q = 0$, Eq. (3) has the trivial solution $a = 0$, $b = 0$. To forbid this solution, we impose the normalization condition $\|b\|_2 = 1$. If the choice of $b$ needed to solve Eq. (3) is known, then the corresponding choice of $a$ is given by $a = Zb$, where $Z = V_M^+ FV_N$ and $V_M^+$ is the Moore–Penrose pseudoinverse of $V_M$. Substituting this relationship into Eq. (3), we are left with the problem to find the coefficients, $b$, of the denominator that

$$\underset{b}{\text{minimize}} \quad \|(V_M Z - FV_N)b\|_2^2 \quad \text{subject to} \quad \|b\|_2 = 1, \tag{4}$$

and this may be solved by taking $b$ to be the right singular vector corresponding to the smallest singular value of $W = V_M Z - FV_N$.

If $K = \alpha(M) + \alpha(N) - 1$, then the number of data points matches the number of degrees of freedom in $p$ and $q$, less 1 for the normalization condition. In this case, we expect that the objective in Eq. (4) can be driven to zero, yielding a linearized rational interpolant to the data, sometimes called a *multipoint Padé approximation*. We write $W = (V_M V_M^+ - I)FV_N$, where $I$ is the identity matrix. Since $V_M V_M^+ - I$ is ($-1$ times) the orthogonal projector onto $\operatorname{Ran}(V_M)^\perp$, it has rank at most $K - \alpha(M) = \alpha(N) - 1$. Since $W$ is of size $K \times \alpha(N)$, this implies that $W$ is rank deficient – it has at least one zero singular value – so $b$ can indeed be chosen to satisfy Eq. (4) with an objective value of zero, as expected.

### 3.2. Discrete multivariate orthogonal polynomials

While in principle one can use any basis $\varphi_0, \ldots, \varphi_{\alpha(L)-1}$ for $\mathcal{P}_L^n$, some bases are better suited to numerical computation than are others. In particular, it is important that the basis be chosen so that the Vandermonde-like matrices $V_M$ and $V_N$ are well conditioned. We would ideally choose the basis so that $V_M$ and $V_N$ have orthonormal columns; in addition to ensuring that operations involving these matrices are robust to rounding error, this would make working with the pseudoinverse of $V_M$ trivial, because we would have $V_M^+ = V_M^*$, where $V_M^*$ is the Hermitian adjoint of $V_M$. We can accomplish this by choosing $\varphi_0, \ldots, \varphi_{\alpha(L)-1}$ so that they are orthogonal with respect to the discrete inner product[1]

$$\langle h, g\rangle = \sum_{k=0}^{K-1} h(x^{(k)})g(x^{(k)}) \tag{5}$$

on $\mathcal{P}_L^n$ associated with the sample points $x^{(k)}$. The orthogonality condition $\langle\varphi_i, \varphi_j\rangle = \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta, is precisely the statement that $V_M$ and $V_N$ have orthonormal columns.

One way to construct such a basis is via a multivariate version of the familiar *Stieltjes process* [30] from the theory of (univariate) orthogonal polynomials. Discussions may be found elsewhere in the literature – see, for example, [31,32] – but to keep this paper self-contained, we describe the process in the form in which we use it here.

The Stieltjes process may be viewed as a variant of the Gram–Schmidt process that orthogonalizes the columns of a Vandermonde matrix without performing the numerically unsavory operation of evaluating high-order monomials, that is, without explicitly forming the matrix itself. In a single variable, it works as follows. We begin by assigning $\varphi_0(x) = 1/\langle 1, 1\rangle$. Then, having constructed $\varphi_0, \ldots, \varphi_{j-1}$, we construct $\varphi_j$ by orthogonalizing $x\varphi_{j-1}(x)$ against $\varphi_0, \ldots, \varphi_{j-1}$,

$$\hat{\varphi}_j(x) = x\varphi_{j-1}(x) - \sum_{i=0}^{j-1} \langle x\varphi_{j-1}, \varphi_i\rangle \varphi_i(x), \tag{6}$$

and normalizing,

$$\varphi_j(x) = \frac{\hat{\varphi}_j(x)}{\sqrt{\langle\hat{\varphi}_j, \hat{\varphi}_j\rangle}}. \tag{7}$$

Since the operation of multiplication by $x$ is self-adjoint (i.e., $\langle x\varphi, \psi\rangle = \langle\varphi, x\psi\rangle$ for all $\varphi$, $\psi$), the orthogonality condition can be used to show that only the $i = j-1$ and $i = j-2$ terms in the sum for $\hat{\varphi}_j$ are nonzero, leading to a three-term recurrence relation for $\varphi_j$. This recurrence can be used to evaluate polynomials that are expressed as linear combinations of the $\varphi_j$ at arbitrary points.

The multivariate case works similarly. The key difference is that since there is no canonical ordering of the monomials in several variables – no agreed-upon order in which to list the columns of a multivariate Vandermonde matrix – we must first select one and then develop a version of the Stieltjes process tailored to that ordering. The ordering we use is as follows. We say that $x_1^{i_1}\cdots x_n^{i_n} < x_1^{j_1}\cdots x_n^{j_n}$ if $i_1 + \cdots + i_n < j_1 + \cdots + j_n$ or if $i_1 + \cdots + i_n = j_1 + \cdots + j_n$ and $i_k > j_k$, where $k$ is the smallest index such that $i_k \neq j_k$. For instance, in $n = 3$ variables $x_1 = x$, $x_2 = y$, and $x_3 = z$, this ordering lists the monomials of degree 3 or less in the following sequence:

$$1, x, y, z, x^2, xy, xz, y^2, yz, z^2, x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3.$$

This order is related to the popular "grevlex" order [33, Sec. 2.2] and has two features that make it convenient. One is that the monomials are ordered by degree. The other is that it yields a

[1] This will be an inner product only if the $x^{(k)}$ constitute a set of linear independence for $\mathcal{P}_L$. We assume throughout this article that this is true.

simple inductive process for listing the monomials in sequence. This is most easily described by example. To construct the three-variable sequence above, we begin with the constant monomial 1. We then multiply 1 by each of the variables in order to obtain the three linear monomials $x$, $y$, and $z$. To produce the quadratic monomials, we first multiply each of the linear monomials by $x$, retaining the order, to produce $x^2$, $xy$, and $xz$. We then multiply by $y$ the linear monomials that do not contain $x$, giving $y^2$ and $yz$. Finally, we multiply the linear monomials that contain neither $x$ nor $y$ – in other words, $z$ – by $z$, giving $z^2$. The cubic monomials are constructed similarly. We multiply all of the quadratic monomials by $x$ to obtain $x^3$ through $xz^2$. Then, we multiply by $y$ the quadratic monomials that do not contain $x$, giving $y^3$ through $yz^2$. Finally, we multiply by $z$ the lone quadratic monomial that contains neither $x$ nor $y$ to produce $z^3$.

The multivariate Stieltjes process that we use is a straightforward outgrowth of this construction. We associate each orthogonal polynomial $\varphi_j$ with its corresponding term in the monomial sequence, beginning with the association $\varphi_0 \leftrightarrow 1$. Having constructed $\varphi_0, \ldots, \varphi_j$, we construct $\varphi_{j+1}$ by multiplying the appropriate previously constructed polynomial by the appropriate variable and orthogonalizing. For instance, taking the three-variable case as an example once more, to produce $\varphi_{12}$, which is associated with the monomial $x^2z$, we multiply $\varphi_6$, which is associated with the monomial $xz$, by $x$ and then orthogonalize the result against $\varphi_0, \ldots, \varphi_{11}$.

This process can be easily adapted to compute the Vandermonde-like matrix $V_L$ corresponding to $\varphi_0, \ldots, \varphi_{\alpha(L)-1}$, which is what we really want, rather than the polynomials themselves. Such a version of the process is given in Algorithm 3.1. In implementing this procedure in finite-precision arithmetic, all of the standard caveats about the numerical stability of the Gram–Schmidt processes apply. In particular, some form of reorthogonalization is mandatory to ensure that the columns of the computed $V_L$ are orthogonal to working precision. In our implementation, we use the standard technique of performing the orthogonalization twice, which is usually sufficient [34,35], [36, §6.9].

Like the univariate Stieltjes process, the multivariate process produces a recurrence relation that can be used to evaluate polynomials expressed in the generated orthogonal basis at arbitrary points. Unlike the univariate recurrence, the multivariate recurrence cannot be reduced to three terms, but it may possess other structure depending on the monomial ordering that is used. The recurrence generated by Algorithm 3.1 is presented in Algorithm 3.2.

**Algorithm 3.1:** Multivariate Stieltjes Process for Vandermonde-like Matrix

```
Input : Points $x^{(0)}, \ldots, x^{(K-1)} \in \mathbb{R}^n$ that are a set of linear
        independence for $\mathcal{P}_L^n$.
Output: Vandermonde-like matrix $V_L$ corresponding to a
        basis $\varphi_0, \ldots, \varphi_{\alpha(L)-1}$ for $\mathcal{P}_L^n$, orthonormal with
        respect to Eq. (5), and coefficients $r_{i,j}$ for use in the
        recurrence of Algorithm 3.2.
$i \leftarrow 1$
for $j = 1$ to $n + 1$ do
    $i_j \leftarrow 0$        /* $i_j$ marks start of sequence last
    multiplied by $x_j$. */
$v_0 = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T / \sqrt{K}$        /* Begin with constant
polynomial. */
for $d = 1$ to $L$ do
    for $j = 1$ to $n$ do
        $i^* \leftarrow i$
        for $k = i_j$ to $i_{n+1}$ do
            $\hat{v}_i \leftarrow \mathrm{diag}(x^{(0,j)}, \ldots, x^{(K-1,j)}) v_k$    /* Multiply by
            $x_j$. */
            for $\ell = 0$ to $i - 1$ do        /* Orthogonalize
            (Gram-Schmidt) Eq. (6). */
                $r_{\ell,i} \leftarrow v_\ell^* \hat{v}_i$
                $\hat{v}_i \leftarrow \hat{v}_i - r_{\ell,i} v_\ell$
            $r_{i,i} \leftarrow \sqrt{\hat{v}_i^* \hat{v}_i}$        /* Normalize Eq. (7) */
            $v_i \leftarrow \hat{v}_i / r_{i,i}$
            $i \leftarrow i + 1$
        $i_j \leftarrow i^*$    /* Update bookkeeping information.
        */
    $i_{n+1} \leftarrow i - 1$
$V_L \leftarrow \begin{bmatrix} v_0 & \cdots & v_{\alpha(L)-1} \end{bmatrix}$
```

### 3.3. Spurious poles and degree reduction

Rational approximations are powerful because of their ability to capture singularities in the function being approximated with singularities of their own; however, if the approximations are computed naively, one often finds that they possess singularities that bear little resemblance to those of the function under consideration. This can happen even when approximating well-behaved functions of a single variable, where the unwanted singularities in the approximation are known as *spurious poles* or *Froissart doublets*. This is a serious problem: in our context, an unwanted singularity in the surrogate model leads to an unbounded objective for our optimization procedure, which can be problematic.

Unwanted singularities can be broadly classified into two types: those that arise in the mathematics and those that arise from noise and numerical artifacts. It seems that little can be done about the former; sometimes the solution to the least-squares problem Eq. (4) really does have a singularity in an undesirable location that does not clearly correspond to a singularity of the function being approximated. Unwanted singularities of the latter type usually emerge when the approximation has more degrees of freedom than are necessary to fit the given data. One advantage of the construction just described is that it affords a natural way to handle this situation. This technique was first described in [8] for univariate approximation; we extend this idea to the multivariate case.

Our construction calls for computing $b$ in Eq. (4) as the right singular vector of $W = V_M Z - F V_N$ corresponding to the smallest singular value. If this singular value is nearly zero, then our rational approximation will fit the data nearly exactly. If $W$ has many singular values that are nearly zero, then there are many possible choices for $b$ – and thus many possible rational approximations – that will have this property. The key idea is this: *If there are many approximations that will work, one should use the approximation with the lowest-degree denominator.* In one dimension, reducing the degree of the denominator by 1 reduces the number of poles of the approximation by 1. If the approximation is already fitting the data well, it is highly likely that the pole that will be eliminated is a spurious one.

Multivariate rational approximations are more complicated than univariate ones: their singularities may not be isolated, and even if we eliminate unnecessary degrees of freedom, they will still, in general, have uncountably many singular points. As such, it is too much to hope that the degree-reduction approach to eliminating unwanted singularities will work as well as it does in the univariate case, especially with noisy input data. We will see this in some of our later experiments. Nevertheless, it can still be highly effective.

**Algorithm 3.2:** Recurrence for Evaluating Multivariate Orthogonal Polynomial Series

```
Input  : Evaluation point $x \in \mathbb{R}^n$, expansion coefficients $c_0$,
         ..., $c_{\alpha(L)-1}$, recurrence coefficients $r_{i,j}$ from
         Algorithm 3.1.
Output: $s = c_0\varphi_0(x) + \cdots + c_{\alpha(L)-1}\varphi_{\alpha(L)-1}(x)$, where the
         $\varphi_i \in \mathcal{P}_L^n$ are the orthogonal polynomials associated
         with the Vandermonde matrix constructed by
         Algorithm 3.1.
$i \leftarrow 1$
for $j = 1$ to $n + 1$ do
    $i_j \leftarrow 0$          /* $i_j$ marks start of sequence last
    multiplied by $x_j$. */
$y_0 \leftarrow 1/\sqrt{\langle 1, 1\rangle}$   /* Begin with $y_0 = \varphi_0(x)$ (constant).
*/
for $d = 1$ to $L$ do
    for $j = 1$ to $n$ do
        $i^* \leftarrow i$
        for $k = i_j$ to $i_{n+1}$ do /* Recurrence for $y_i = \varphi_i(x)$.
        */
            $\hat{y}_i \leftarrow x_j y_k$
            for $\ell = 0$ to $i - 1$ do
                $\hat{y}_i \leftarrow \hat{y}_i - r_{\ell,i} y_\ell$
            $y_i \leftarrow \hat{y}_i / r_{i,i}$
            $i \leftarrow i + 1$
        $i_j \leftarrow i^*$      /* Update bookkeeping information.
        */
    $i_{n+1} \leftarrow i - 1$
$s \leftarrow c_0 y_0 + \cdots + c_{\alpha(L)-1} y_{\alpha(L)-1}$                /* Evaluate
$s = c_0\varphi_0(x) + \cdots + c_{\alpha(L)-1}\varphi_{\alpha(L)-1}(x)$. */
```

The procedure we recommend is summarized in Algorithm 3.3. The algorithm attempts to reduce the denominator degree by 1, checking to see whether this is possible by examining the smallest singular value of the $W$ matrix associated with the reduced degree. If this singular value is smaller than a chosen threshold $\eta$, it deems the reduction successful and then tries to reduce the degree by 1 again. It continues until the smallest singular value of $W$ is too large for the reduction to be considered viable. It then repeats the process to reduce the degree of the numerator by considering the problem of fitting an approximation to the reciprocal data.

By considering the nullity of $W$, we can reduce the degree in steps greater than 1: if $W$ has many singular values that lie below the threshold, we could eliminate many degrees of freedom simultaneously. Nevertheless, we have found that the stated approach is more robust, especially in the presence of noise. For this procedure to succeed, the approximation must be expressed in a well-behaved basis such as the discrete orthogonal polynomial basis described in the preceding section. With a badly behaved basis, the singular values of $W$ may not decay as rapidly, resulting in opportunities for degree reduction (and thus for singularity reduction) being missed.

How should we choose the threshold $\eta$? With noiseless input data, a singular value of $W$ will be negligible if its size relative to the largest singular value is on the order of the rounding error incurred during the computation. In this case, an appropriate choice for $\eta$ is a small power of 10 times the machine epsilon; for double-precision arithmetic, values such as $\eta = 10^{-12}$ or $\eta = 10^{-14}$ work well. If the input data are noisy, the threshold should be increased so that any singular value below the noise level is regarded as negligible. For instance, if all but the 6 leading digits of the data are noisy, then setting $\eta = 10^{-5}$ (a factor of 10 larger than the relative noise level of $10^{-6}$) may be appropriate.

**Algorithm 3.3:** Degree Reduction

```
Input  : Vandermonde-like matrix $V_L$ computed with
         Algorithm 3.1, diagonal matrix $F$ of sample values,
         maximum numerator and denominator degrees $M$
         and $N$, threshold $\eta$.
Output: Reduced degrees $M$ and $N$.
/* Reduce the denominator degree. */
while true do
    $Z \leftarrow V_{M-1}^* F V_N$
    $\sigma_{\min}, \sigma_{\max} \leftarrow$ smallest, largest singular values of $Z$
    if $\sigma_{\min} < \eta\sigma_{\max}$ then
        $M \leftarrow M - 1$
    else
        break
/* Reduce the numerator degree. */
while true do
    $Z \leftarrow V_{N-1}^* F^{-1} V_M$
    $\sigma_{\min}, \sigma_{\max} \leftarrow$ smallest, largest singular values of $Z$
    if $\sigma_{\min} < \eta\sigma_{\max}$ then
        $N \leftarrow N - 1$
    else
        break
```

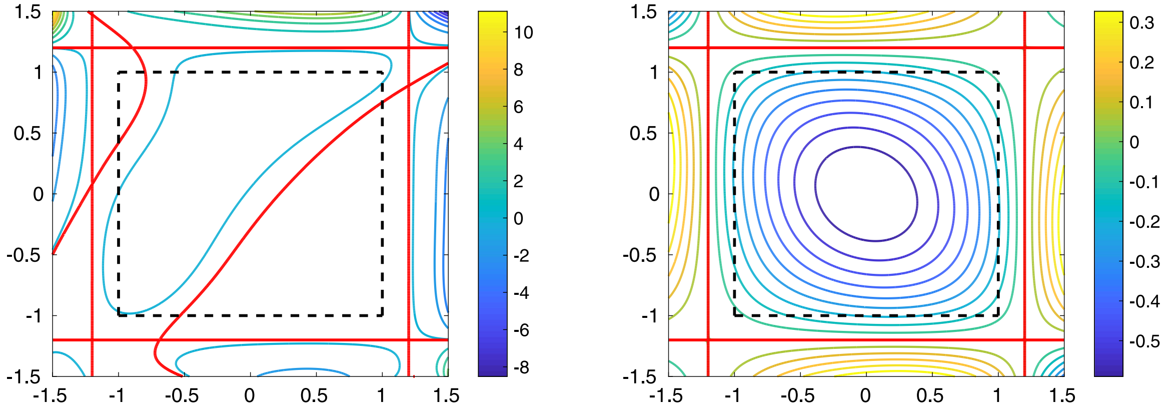
To illustrate the effectiveness of this general procedure, we consider the problem of computing a rational approximation to the bivariate function $f(x, y) = \exp(xy)/\big((x^2 - 1.44)(y^2 - 1.44)\big)$. We sample this function in 1000 uniformly randomly distributed points in $[-1, 1] \times [-1, 1]$ and attempt to fit a rational approximation with numerator and denominator degrees $M, N = 20$. Fig. 1 displays a contour plot of the denominator of the computed approximation. Without degree reduction, we obtain the picture in Fig. 1a. In addition to the singularity curves at $x = \pm 1.2$ and $y = \pm 1.2$ that reflect the true singularities of $f$, the approximation possesses a pair of spurious singularity curves that wind their way through the middle of the square. Applying Algorithm 3.3 with $\eta = 10^{-12}$ reduces the numerator degree to $M = 12$ and the denominator degree to $N = 9$ and produces a rational approximation with a denominator that generates the contour plot of Fig. 1b. The spurious singularity curves have disappeared.

## 4. Multidimensional rational approximation with constraints

The algorithm just described is simple and powerful; however, even with degree reduction, it does not guarantee that the computed approximation is free of singularities in the domain of interest. In this section, we add constraints to the rational approximation problem Eq. (2) that enforce this requirement. We show that these constraints lead to a semi-infinite optimization problem (see, e.g., [26,37,38]), which we solve using an outer approximation approach due to Polyak [39]. We are motivated by a class of structural constraints that arise in HEP data analysis, for which it is known that the underlying function has no poles in a certain domain $D$ (but may have them outside of $D$), and we exploit this information by enforcing the same condition for our rational approximation.

Formally, we can write the constraint that "$r(x)$ has no poles in $D$" as the condition that

$$q(x) \neq 0, \quad \forall\, x \in D.$$



(a) Without degree reduction.

(b) With degree reduction.

**Fig. 1.** Contour plots of the denominators of the rational approximations computed in the example of Section 3.3. Red lines denote zero-level curves (and hence curves of singularities present in the approximation). The dashed black line outlines the unit square $[-1, 1] \times [-1, 1]$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

However, this condition is not a convenient constraint to add to Eq. (2) because it describes an open set. Instead, we use the equivalent condition that "$q(x)$ does not change sign in $D$", which can be written without loss of generality as

$$q(x) \geq \tau > 0, \qquad \forall x \in D, \tag{8}$$

where $\tau > 0$ is an arbitrary positive constant. (We use $\tau = 1$ in our experiments.) In most cases, $D$ will be a simple set such as bounded hyper-rectangle $D = \prod_{i=1}^{n}[L_i, U_i]$. We then formulate the multivariate constrained rational approximation problem as the following constrained least-squares problem:

$$\underset{p,q}{\text{minimize}} \sum_{k=0}^{K-1} \left(p(x^{(k)}) - f_k q(x^{(k)})\right)^2 \quad \text{subject to } q(x) \geq \tau,$$
$$\forall x \in D \quad \text{and} \quad p \in \mathcal{P}_M^n, q \in \mathcal{P}_N^n. \tag{9}$$

This is a linear least-squares problem in the coefficients of the polynomials $p(x)$ and $q(x)$ with a linear semi-infinite constraint; see, for example, the monographs and surveys by [26,37,38].

If $q(x) = b^T x + b_0$ and $D$ is affine, then we can use linear programming duality to replace the semi-infinite constraint by a set of equivalent finite-dimensional affine constraints; see, for example, [38]. In general, however, this transformation does not exist unless we also assume that $q(x)$ is convex, which would add a semi-definite constraint in the quadratic case and more complex conic constraints in general. Hence, we will instead consider an outer approximation approach to solving Eq. (9).

### 4.1. A practical algorithm for general rational approximation

For general denominators, we apply a method due to Polyak. The algorithm maintains a finite set $U$ of points $x^{(k)} \in D$ at which the semi-infinite constraint is enforced. It then alternates between solving the finite-dimensional relaxation of Eq. (9), which at iteration $l$ is given by

$$\underset{p,q}{\text{minimize}} \sum_{k=0}^{K-1} \left(p(x^{(k)}) - f_k q(x^{(k)})\right)^2 \quad \text{subject to } q(x^{(k)}) \geq \tau,$$
$$\forall k = 0, \ldots, K-1+l, \tag{10}$$

and an optimization problem to check Eq. (8). We let the solution of this problem be $p_l(x)$, $q_l(x)$, and then solve the following minimization problem to global optimality to check whether Eq. (8) holds:

$$\underset{x \in D}{\text{minimize}}\; q_l(x). \tag{11}$$

Either we obtain a new point $\hat{x} \in D$ that violates $q_l(\hat{x}) \geq \tau$ or we show that $q_l(x) > 0$ for all $x \in D$. Formally, this procedure is defined in Algorithm 4.1.

**Algorithm 4.1:** Alternating Algorithm for Pole-Free Rational Approximation.

```
  Input : $\{x^{(0)}, \ldots, x^{(K-1)}\}$
  Output: Pole-free rational approximation $p_l(x)/q_l(x)$
1 Set $l \leftarrow 0$, done $\leftarrow$ false
2 repeat
3     Let $p_l(x)$, $q_l(x)$ be a solution of the relaxation Eq. (10).
4     Let $\hat{x}$ be a (global) minimizer of Eq. (11).
5     if $q_l(\hat{x}) \geq \tau$ then
6         Set done $\leftarrow$ true
7     else
8         Add a new point: $\{x^{(K+l)} := \hat{x}\}$ and set $l \leftarrow l+1$
9 until done is false
```

We note that we can stop the algorithm as soon as $q_l(\hat{x}) > 0$, which indicates that $q(x)$ has no poles in $D$. The final $p_l(x)/q_l(x)$ is the best (least-squares) interpolant that has no poles in $D$. Unfortunately, the algorithm requires the global minimization of the polynomial $q_l(x)$ over $D$. We can either resort to multistarts (multiple local optimizations starting from different points), or compute an underestimator of $q_l(x)$ on $D$ using the reformulation-linearization-technique of [40]. We discuss a practical way of solving the global optimization problem using multistarts in Appendix B.

An alternative approach that avoids the global optimization in Step 4 of Algorithm 4.1 is based on sampling. In particular, [22, Theorem 1], which generalizes an earlier result due to Pomentale [41] to the multivariate setting, provides a criterion guaranteeing that the denominator does not vanish over the approximation domain. Using this criterion, one could verify that $q_l(x) > 0$ on $D$ without having to resort to global optimization. Unfortunately, the criterion is difficult to check in practice, and may require a prohibitively large number of samples as the dimension of the rational approximation grows.

## 5. Numerical experiments

In this section, we compare the approximation quality and computation times of the rational and polynomial approximation approaches. We also study the effects of using different strategies for sampling the interpolation points from the domain and the effects of constraints on the rational approximation.

### 5.1. Experimental setup

Our numerical experiments are conducted on a server with 64 Intel Xeon Gold CPU cores running at 2.30 GHz. There are two threads per core, but each approximation is run on a single thread. The operating system is Linux Ubuntu 16.04. Additionally, the server is equipped with 1.5TB DDR4 2666 MHz of memory. The code is written in Python v3.7.2 where the optimization functions and constraints are compiled with the Numba JIT compiler v0.42.

The experiments are conducted on fast-to-compute analytic test problems whose functional forms are summarized in Table A.5. The use of these analytic test problems enables us to assess the performance of our algorithms efficiently. We show detailed results for five typical test functions that span the range of the functions of interest in this section, and we summarize the remaining results for the other functions, which are included in the electronic supplement sections SM1 and SM2. In the following we show the results for Function A.5.4 whose approximation using Taylor series expansions is a polynomial function; Function A.5.7, which is a rational function; Function A.5.15, which is used to describe a resonant particle of mass $M$ and width $\Gamma$ as a function of the particle's energy $E$ in high-energy physics [42,43]; and Functions A.5.16 and A.5.17, whose approximation using Taylor series expansions is a rational function. Note that the domain of Function A.5.16 is close to the true pole.

We sample the interpolation points $\{x^{(0)}, \ldots, x^{(K-1)}\}$ using sparse grids (SGs) [44] and Latin hypercube sampling (LHS) [45], and we propose a new hybrid strategy called decoupled Latin hypercube design (d-LHD) where the interpolation points are sampled on the faces and inside the domain. A plot of the interpolation points sampled by using the three different strategies is shown in Fig. 2. The approximation results change for interpolation points that are sampled by using the LHS and d-LHD strategies because they have randomness. To account for these changes, each experiment is repeated five times using different random number seeds, and we report the mean and other statistics of the performance metrics for these strategies. We also experimented with uniform randomly sampled points, but we found the results to be inferior and therefore do not include them here.

Each functional value $f_k$ is obtained by evaluating $f$ at $x^{(k)}$. The number of interpolation points $K$ is set as twice the sum of the number of degrees of freedom of the polynomials in each approximation given by $\alpha(M) + \alpha(N)$ for numerator of degree $M$ and denominator of degree $N$. We consider both noise-free and noisy data in the experiments. For noisy data, each functional value $f_k$ is multiplied by a fraction $\epsilon$ of the random value $\phi^{(k)}$ sampled from a standard normal distribution $\mathcal{N}(0, 1)$ as follows:

$$f_k = f_k \left(1 + \epsilon \phi^{(k)}\right), \quad \forall k = 0, \ldots, K-1. \tag{12}$$

The approximation $r(x)$ is computed in four ways: (1) $p(x)$ is the polynomial approximation that is computed by a *NumPy* implementation of finding the linear least-squares solution using singular value decomposition within the driver routine *DGELSD* [46], (2) $r_1(x)$ is the rational approximation using Algorithm 3.1 without degree reduction, (3) $r_2(x)$ is the rational approximation using Algorithm 3.1 with the degree reduction described in Algorithm 3.3, and (4) $r_3(x)$ is the rational approximation using Algorithm 4.1.

To assess the quality of our approximations, we use a second set of testing points $\{x^{(K)}, \ldots, x^{(L-1)}\}$ on the faces and inside of the domain, and we compute their function values $\{f_K, \ldots, f_{L-1}\}$. No noise is added to the testing data. We use the $l_2$-norm error as a test metric to compare the quality of the approximation $r(x)$:

$$\begin{aligned}\Delta_r = ||r - f||_{D,2} &= \left(\int_D (r(x) - f(x))^2\, dx\right)^{1/2} \\ &\approx \left\{\sum_{k=K}^{L-1} \left[r\left(x^{(k)}\right) - f_k\right]^2\right\}^{1/2}.\end{aligned} \tag{13}$$

We consider a solution to be better if it has smaller $\Delta_r$. We assume that the degrees of the numerator and the denominator polynomials of the approximations each are 5. This choice allows us to approximate test functions in which the polynomials are up to degree 4. Choosing the optimal degree of polynomials is a question that is beyond the scope of this paper.

### 5.2. Effects of interpolation point selection method

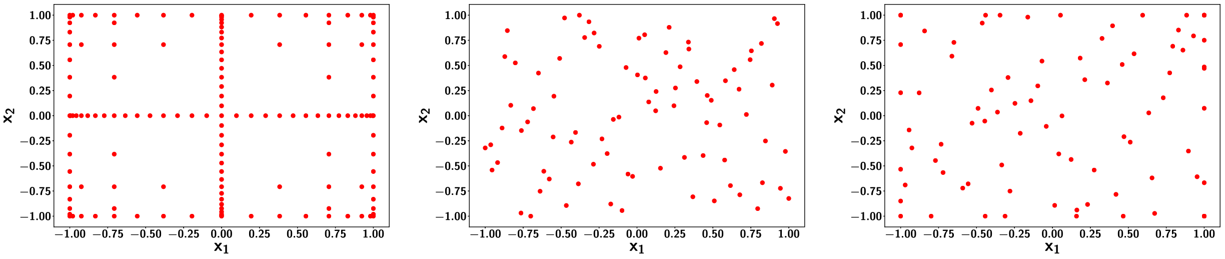
In this section we discuss the choice of the interpolation points using SGs [44], LHS [45] and d-LHD strategies. SGs, first proposed by Smolyak, are sparse tensor product spaces. We also experimented with a uniform random set of points but observed uncompetitive results. With SGs, the grid points are obtained by combining, up to a certain level, the tensor product grid corresponding to the total degree multi-index set. Here, the SG level is chosen such that the number of points in the grid is at least twice the total degrees of freedom of the polynomials in each approximation. Fig. 2a shows a 2D SG. We observe that many points of the SG are collinear, violating the linear independence assumption from Section 3. Hence, for the chosen SG levels, the Hessian matrix of the fitting problem in Eq. (10) is singular. Because of the null space, there are multiple minimizers that result in unbounded values for $p$ and $q$. We overcome this issue by adding a regularization term with weight $\sigma > 0$ to Eq. (10) and bounding the eigenvalues $> \sigma$. The updated fitting problem in iteration $l$ of Algorithm 4.1 is

$$\begin{aligned}&\underset{p,q}{\text{minimize}} \sum_{k=0}^{K-1} \left(f_k q(x^{(k)}) - p(x^{(k)})\right)^2 + \sigma \left(\sum_{j=0}^{\alpha(M)-1} \widehat{a}_j^2 + \sum_{j=0}^{\alpha(N)-1} \widehat{b}_j^2\right) \\ &\text{subject to } q(x^{(k)}) \geq 1,\ \forall k = 0, \ldots, K-1+l,\end{aligned} \tag{14}$$

where $\widehat{a}$ and $\widehat{b}$ are the coefficients of the monomial basis expansion of $p(x)$ and $q(x)$, respectively. To choose $\sigma$, we ran Algorithm 4.1 with Eq. (14) to approximate the data sampled with SG. We found $\sigma$ to be in the vicinity of $10^{-1}$ for all test functions using the L-curve method. An example plot of the L-curve for Function A.5.16 interpolation data is shown in Fig. 3.

When using the SG interpolation points, we observe that Algorithm 4.1 takes only one iteration to converge to a pole-free rational approximation. This is shown in the top plot of Fig. 4. However, penalizing the coefficients of the monomial basis expansions of $p(x)$ and $q(x)$ results in a high testing error thereby deteriorating the approximation quality, as shown in the bottom plot of Fig. 4, which is undesirable.

When using LHS, each independent dimension is sampled by using an even sampling method, and then these samples are randomly combined to obtain the sample data. Fig. 2b shows one such 2D LHS sample. The advantage of LHS is that the interpolation points are not collinear. This results in a Hessian matrix of full rank for the fitting problem in Eq. (10) and hence does not require the regularization term to be added. However, the



(a) 2D Sparse gid of level 5 with 145 points.

(b) 2D Latin hypercube sample with 84 points.

(c) 2D decoupled Latin hypercube Design with 84 points.

**Fig. 2.** Location of interpolation points using different sampling strategies for a rational approximation of $M = 5$, $N = 5$ and $\alpha(M) + \alpha(N) = 42$.

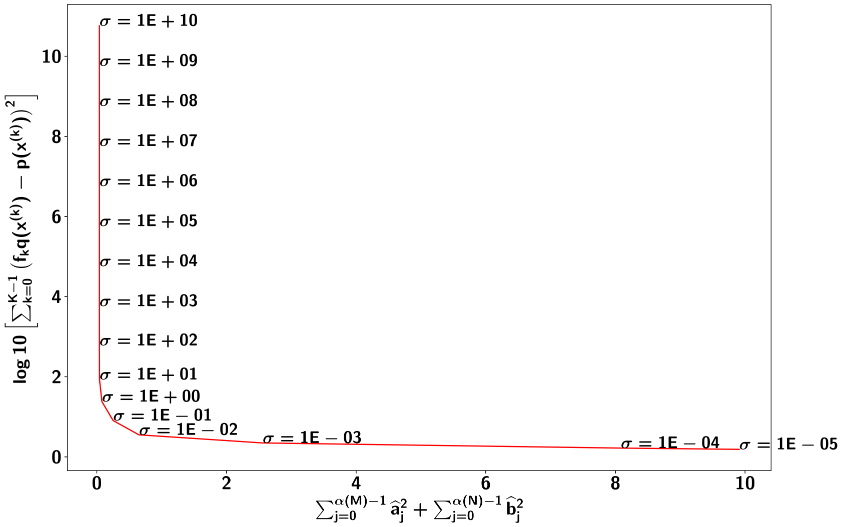


**Fig. 3.** L-curve method to choose $\sigma$. Function A.5.16 interpolation data is sampled by using SG. The approximations are performed by using Algorithm 4.1 with Eq. (14) instead of Eq. (10) for different values of $\sigma$. The degrees of the numerator and denominator polynomials are $M = 5$ and $N = 5$, respectively. The corner of the L is found at $\sigma = 10^{-1}$.

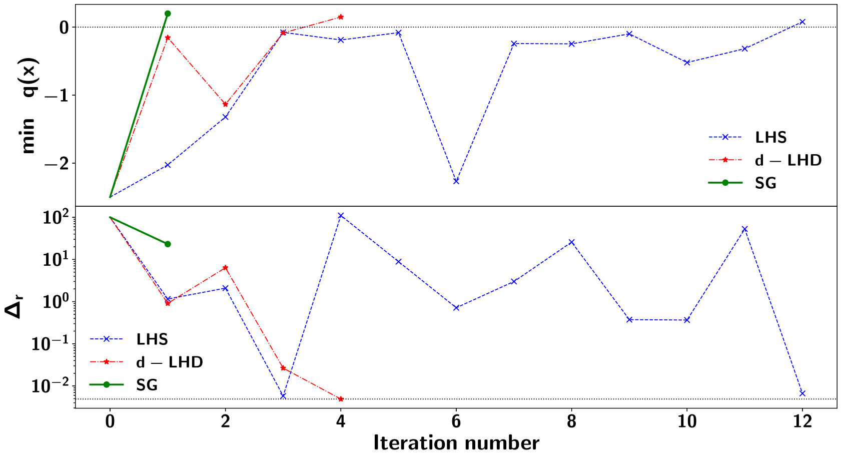


**Fig. 4.** Minimizers of Eq. (11) (top) and testing error (bottom) per iteration of a run of Algorithm 4.1 on Function A.5.15 interpolation data sampled by using all three strategies. In the top plot, Algorithm 4.1 stops iterating when the minimizer is at least 0 (shown with a horizontal dotted line). In the bottom plot, the approximation obtained by using d-LHD sampled data has the lowest testing error (shown with a horizontal dotted line). The numbering of the iterations of Algorithm 4.1 starts at 1. In both plots, the lines for all strategies start from the same extreme point (at iteration 0).

number of iterations of Algorithm 4.1 over noise-free LHS data is on average five times the number of iterations over noise-free SG data (see Fig. 6).

We observe that almost all the minimizers of $q(x)$ found in each iteration of Algorithm 4.1 lie on the faces of the domain as shown in Fig. 5. Hence, when SG places a number of points on the face of the domain, the number of spurious poles is minimized, which on average requires fewer iterations of Algorithm 4.1. So ideally, we want to use a sampling strategy that covers the faces and the inside of the domain evenly such that the points are not collinear, thereby combining the best features of SG and LHS.

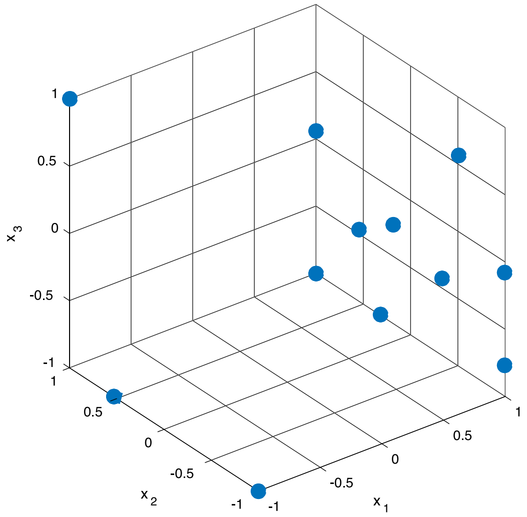


**Fig. 5.** Minimizer of Eq. (11) from all iterations of Algorithm 4.1 on Function A.5.15 interpolation data. The data are sampled by using LHS. All minimizers lie on the face of the cubic domain.

The authors of [47] proposed the maximin augmented nested Latin hypercube design sampling strategy to maximize prediction accuracy. In this strategy, the samples are constructed by augmenting nested LHDs with additional parameters using a modified smart swap algorithm such that the final design satisfies the maxmin property. However, the required properties of the samples to satisfy our goal are simpler, and we therefore use our decoupled Latin hypercube design. We construct nested LHDs over all the $2n$ facets of the domain with dimension $n-1$; in other words, one of the dimensions in each face's sample is fixed. Because these samples are LHDs, the points are not collinear. In order to cover the inside of the domain, an augmented LHD is obtained inside the $n$-dimensional hyper-rectangle. These two steps are independent. Even though the samples on each face and inside the domain satisfy the maxmin property, we do not require that the final design satisfies the maxmin property. We call this sampling strategy decoupled Latin Hypercube Design (d-LHD).

In d-LHD, the number of points sampled is still twice the degrees of freedom of the polynomials in each approximation, namely, $K = 2(\alpha_n(M) + \alpha_n(N))$. On the $2n$ faces, there is an $(n-1)$-dimensional rational function; hence, the number of points sampled along each face of the domain is given as

$$K^{(fc)} = \frac{2(\alpha_{n-1}(M)+\alpha_{n-1}(N))}{2n} = \frac{(\alpha_{n-1}(M)+\alpha_{n-1}(N))}{n}, \qquad (15)$$

and the number of points sampled inside the domain is given as

$$K^{(in)} = K - 2n \cdot K^{(fc)}. \qquad (16)$$

Thus, the d-LHD samples points on each face as well as the inside of the domain as illustrated in Fig. 2c.

Fig. 6 compares the number of iterations performed by Algorithm 4.1 when the function domains are sampled with LHS and d-LHD, respectively. Table 1 shows statistics for the number of iterations performed over all test functions in Table A.5. Fitting the approximation to data sampled using SG takes only one iteration, but it causes a higher testing error compared with the other two strategies (see Fig. 4). Additionally, when using d-LHD to sample points, the number of iterations of Algorithm 4.1 is almost always lower compared with LHS (see Fig. 6) without compromising the approximation quality (see the bottom plot of Fig. 4). Hence, in the remainder of this section, we present results for the approximations performed with data sampled by d-LHD. The results corresponding to the other sampling strategies can be found in the electronic supplement sections SM1 and SM2.

**Table 1**
Number of iterations over all test functions in Table A.5 performed by Algorithm 4.1 over noise-free data sampled using LHS and d-LHD strategies. Here, "Range" is the difference between the maximum and the minimum iterations over all test functions. When the data is sampled by using SG, the number of iterations is 1 for all functions. However, the testing error of the approximation obtained by using SG for all functions is much higher than those obtained by using LHS and d-LHD (see electronic supplement section SM1). For almost all functions, Algorithm 4.1 takes fewer iterations over data sampled using d-LHD, which is evident from the median and the geometric mean. However, the average number of iterations over Function A.5.18 data sampled by using d-LHD is 80.4 whereas that over data sampled by using LHS is 26.4. This outlier makes the arithmetic mean look better for the LHS strategy. The corresponding results for each function can be found in the electronic supplement section SM2.

| Statistic | LHS | d-LHD |
|---|---|---|
| Arithmetic mean | 4.17 | 5.50 |
| Geometric mean | 1.86 | 1.51 |
| Median | 1.20 | 1.00 |
| Range | 26.60 | 79.40 |

## 5.3. Comparison of approximation quality

In this section, we evaluate the ability of constraints in Algorithm 4.1 to remove spurious poles and compare the quality of our rational approximations. More specifically, we examine the number of spurious poles detected and its effect on the testing error in the three rational approximation approaches. Then, we compare the quality of the rational approximations with the polynomial approximation by comparing their testing errors.

### 5.3.1. Ability to remove spurious poles

In this section, we compare the number of spurious poles found in the three rational approximation approaches, since we are interested in separating the error due to these poles from the actual approximation error. Detecting these poles is difficult, however, because multivariate rational approximations are more complicated than univariate ones. They may have uncountably many singular points, and these singularities are typically not isolated. Hence, to perform this comparison, we find the testing points near poles or pole-like points that have large function deviations. As shown in Fig. 5, the minimizers of $q_l(x)$ in each iteration of Algorithm 4.1 tend to be on the boundary of the domain. Hence, we choose the testing points randomly on the faces of the domain in addition to randomly chosen points inside of the domain. For these testing points, we define $W_{r,t}$ as the index set of points whose absolute approximated value is much larger than the corresponding absolute value of the function indicating a possible spurious pole. More formally, we define $W_{r,t}$ as:

$$W_{r,t} = W_{r,t}^{(fc)} \cup W_{r,t}^{(in)} = \left\{ j \,\middle|\, \frac{|r(x^{(j)})|}{\max\left(1, |f_{max}^{(fc)}|\right)} > t \right\} \cup \left\{ k \,\middle|\, \frac{|r(x^{(k)})|}{\max\left(1, |f_{max}^{(in)}|\right)} > t \right\}, \qquad (17)$$

where $x^{(j)}$ and $x^{(k)}$ are testing points on the face and inside of the domain respectively, $j \in I^{(fc)}, k \in I^{(in)}, I^{(fc)} \cup I^{(in)} = \{K, \dots, L-1\}, I^{(fc)} \cap I^{(in)} = \emptyset, f_{max}^{(fc)} = \max |f_j|, f_{max}^{(in)} = \max |f_k|$, and $t$ is a large threshold.

Fig. 7 shows the average number of pole-like points found over all functions in Table A.5 when the interpolation data for these functions was sampled by using d-LHD. The number of pole-like points per function in Table A.5 for d-LHD, SG, and LHS-based approximations is given in the electronic supplement section SM1. We observe pole-like points in $r_1(x)$ and $r_2(x)$ for noise-free and

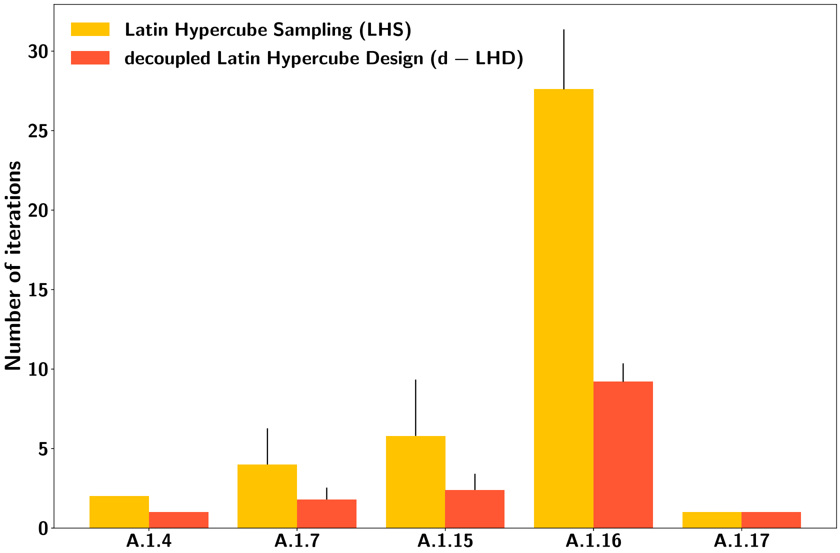
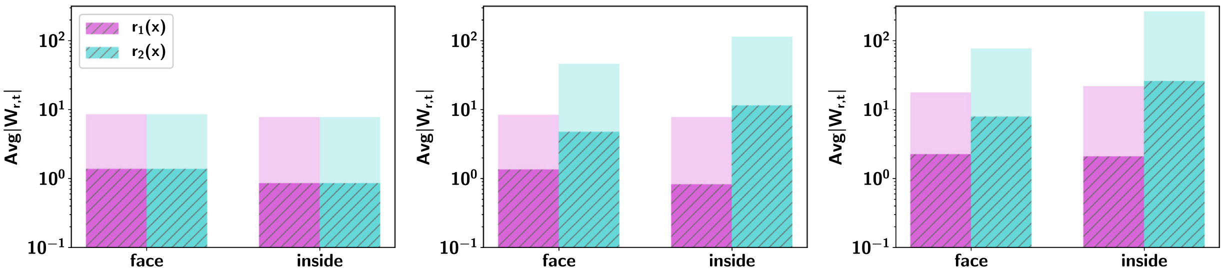


**Fig. 6.** Number of iterations performed by Algorithm 4.1 over noise-free data sampled by using LHS and d-LHD strategies. The standard deviation is shown as a black vertical line. When the data are sampled by using SG, the number of iterations is 1 for all functions. As shown in the bottom plot of Fig. 4, however, the testing error of the approximation obtained by using SG is much higher than those obtained using LHS and d-LHD.



(a) $\epsilon = 0$ (b) $\epsilon = 10^{-6}$ (c) $\epsilon = 10^{-2}$

**Fig. 7.** Comparison of the average number of pole-like points found over all functions in Table A.5 for different relative noise levels $\epsilon$. The data is sampled with d-LHD. Each bar represents the average number of pole-like points found when $t \geq 10^2$. The average number of pole-like points found when $t \geq 10^3$ is shown as a hatched bar. The average number of pole-like points found when $10^2 \leq t < 10^3$ is shown as a faded bar. The number of pole-like points found in $r_3(x)$ is 0 for all noise levels.

noisy interpolation data. In contrast, the approximation $r_3(x)$ does not have these pole-like points. As discussed in Section 4.1, this is due to the iterative removal of poles by Algorithm 4.1 by design, thereby giving a pole-free $r_3(x)$. When no noise is added to the interpolation data, that is, when $\epsilon = 0$, the number of pole-like points found on the faces of the domain is larger than inside of the domain in $r_1(x)$ and $r_2(x)$. This difference is more prominent when the interpolation data are sampled by using LHS. The number of pole-like points found in $r_1(x)$ and $r_2(x)$ on the face is 24% higher than those found on the inside whereas this difference is only 8.5% when the interpolation data are sampled by using d-LHD. The reason is that LHS samples fewer interpolation points on the faces of the domain, causing the LHS-based approximations to be less accurate on the boundary of the domain, especially when the function domain is in close proximity to the true poles. Also, this result is consistent with our earlier observation that the minimizers found in each iteration of Algorithm 4.1 tend to lie on the face of the domain.

In the presence of noise, that is, when $\epsilon \neq 0$, the number of pole-like points found in $r_1(x)$ and $r_2(x)$ increases inside as well as on the faces of the domain. As above, $r_3(x)$ does not suffer from spurious poles. The number of pole-like points found for $r_2(x)$ is much higher than for $r_1(x)$. As discussed before, the reason is that multivariate rational approximations are more complicated than univariate ones. Their singularities are never isolated; and even if we eliminate unnecessary degrees of freedom, they will still, in general, have uncountably many singular points. This problem is compounded when the input data are noisy. Thus, we cannot hope that the degree-reduction approach will work as well in the multivariate case as it does in the univariate case. On the other hand, the optimization approach by design eliminates poles in $D$.

### *5.3.2. Comparison of the testing error*

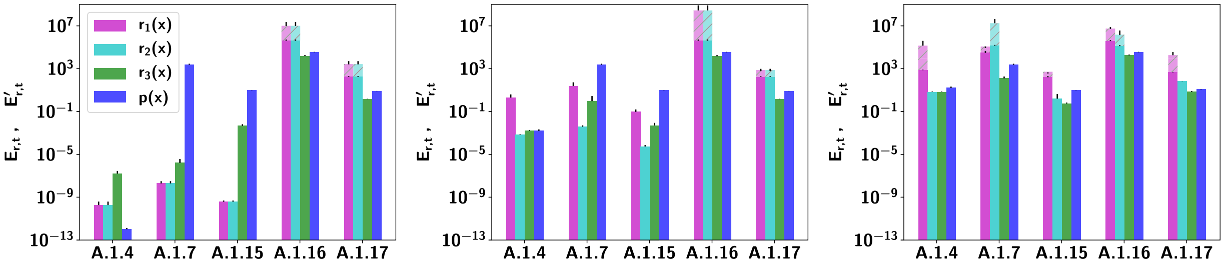
To better compare the testing error, we divide it into two parts: the component due to poles and the remainder. Given the definition of $W_{r,t}$ in Eq. (17), the error due to pole-like points is defined as

$$E_{r,t} = \left[ \sum_{j \in W_{r,t}} (r(x^{(j)}) - f_j)^2 \right]^{1/2}, \tag{18}$$

and the error not due to pole-like points is

$$E'_{r,t} = \left[ \Delta_r^2 - E_{r,t}^2 \right]^{1/2}. \tag{19}$$

The testing error for the three rational approximations $r_1(x)$, $r_2(x)$, and $r_3(x)$, as well as the polynomial approximation $p(x)$ is given in Fig. 8 and Table 2. The data for these plots are given in the electronic supplement section SM1. In order to ensure a fair comparison, the degrees of freedom are the same among the



(a) $\epsilon = 0$ (b) $\epsilon = 10^{-6}$ (c) $\epsilon = 10^{-2}$

**Fig. 8.** Comparison of the quality of rational and polynomial approximations. The data are sampled with d-LHD, the threshold is $t = 10^2$ and $\epsilon$ is the level of relative noise added to the data. For approximations where pole-like points were found, the average error due to pole-like points is shown as a faded hatched bar. The average errors not due to pole-like points are shown as solid bars and are superimposed on the faded hatched bar where applicable. All the faded hatched bars, when not 0, are taller than the solid bars. The standard deviation is shown as a black vertical line. Function A.5.4 is an exponential function, Function A.5.7 is rational function, Function A.5.15 is a Breit–Wigner function, and Functions A.5.16 and A.5.17 are functions whose denominator is a polynomial.

**Table 2**
Testing error ($\Delta_r$) for all test functions in Table A.5 of rational and polynomial approximations. The data are sampled with d-LHD, and $\epsilon$ is the level of relative noise added to the data. Since the scale of the error for each function is different, the error is first normalized to a 0–1 scale before calculating each statistic over all functions. The corresponding results for each function can be found in the electronic supplement section SM1.

| Statistic | $\epsilon = 0$ | | | | $\epsilon = 10^{-6}$ | | | | $\epsilon = 10^{-2}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $r_1(x)$ | $r_2(x)$ | $r_3(x)$ | $p(x)$ | $r_1(x)$ | $r_2(x)$ | $r_3(x)$ | $p(x)$ | $r_1(x)$ | $r_2(x)$ | $r_3(x)$ | $p(x)$ |
| Arithmetic mean | 6.34E−02 | 6.34E−02 | 5.36E−02 | 8.38E−02 | 6.43E−02 | 5.33E−02 | 5.37E−02 | 8.38E−02 | 6.23E−02 | 7.49E−02 | 5.80E−02 | 8.37E−02 |
| Median | 1.88E−15 | 1.88E−15 | 4.48E−08 | 3.19E−04 | 1.85E−08 | 5.86E−07 | 5.16E−06 | 3.19E−04 | 3.65E−03 | 2.63E−03 | 8.71E−04 | 5.72E−04 |

rational approximations. The degrees of freedom of the polynomial approximation are at least as large as those of the rational approximation. For the rational approximations, the error due to pole-like points is given for the threshold value of $t = 10^2$. From Table 2, we observe that the approximation $r_3(x)$ performs best overall for all functions and all noise levels. More specifically, the approximation $r_3(x)$ has the lowest error when there is no noise or there are high levels of noise in the interpolation data. However, the quality of the approximation of $r_2(x)$ matches that of $r_3(x)$ when the level of noise is low ($\epsilon = 10^{-6}$). The reason is that the degree reduction in Algorithm 3.1 is able to reduce poles and give a better-quality approximation for low noise levels. From Fig. 8, we observe that whenever pole-like points are found, their contribution to the testing error is high, as defined in Eq. (18). The polynomial approximation yields a lower testing error than rational approximations do for the noise-free case of Function A.5.4 because Function A.5.4 is approximated by a polynomial. Conversely, for rational functions such as Function A.5.6, the rational approximations over noise-free data yield better testing errors than the polynomial approximation does. Moreover, for rational functions, the errors without pole-like points in approximations $r_1(x)$ and $r_2(x)$ is on the order of $10^{-8}$ and is lower than $10^{-6}$ for $r_3(x)$. We believe the reason is that the approximations $r_1(x)$ and $r_2(x)$ are obtained from Algorithm 3.1, whose orthonormal basis implementation is numerically more accurate than the constrained optimization approach of Algorithm 4.1 in the monomial basis. We also observe that the testing errors of the approximations of noise-free data of Function A.5.15 show trends similar to those described above for the rational functions. The reason is that the denominator of this function approximates to a polynomial of degree 4 and has the unit of physical energy $E^4$. Since Function A.5.15 is unitless, the numerator also has a unit of $E^4$. Thus, the entire function can be approximated by a rational function with numerator of degree 4 and denominator of degree 4 [42,43]. For other functions, the approximation $r_3(x)$ over noise-free data performs better than the other approximations due to the lack of spurious poles as well as due to better goodness of fit.

**Table 3**
Total CPU time over all test functions in Table A.5 for all four approximation approaches when the interpolation data is sampled using d-LHD and is noise-free. The total time taken by Algorithm 4.1 is shown as its fit time and multistart time to perform the global optimization of $q_l(x)$. Algorithm 4.1 is more expensive than the other approaches are, and this time is clearly dominated by the multistart time. The corresponding results for each function can be found in the electronic supplement section SM2.

| Statistic | $p(x)$ | $r_1(x)$ | $r_2(x)$ | $r_3(x)$: Fit time | $r_3(x)$: Multistart time |
|---|---|---|---|---|---|
| Arithmetic mean | 0.66 | 3.17 | 4.00 | 8.60 | 88.95 |
| Geometric mean | 0.63 | 3.11 | 3.94 | 0.19 | 31.44 |
| Median | 0.62 | 3.30 | 4.02 | 0.04 | 15.85 |

Generally all approximations for noise-free data are better than for noisy data. For $r_1(x)$ and $r_2(x)$, one reason is the higher number of pole-like points in the noisy data case. Another reason is the poor quality of the fit of the approximations to the data. This is because the degrees of freedom are the same in both the noise-free and the noisy data cases. Hence, for higher levels of noise, the approximation underfits the data because there may not be any spare degrees of freedom to fit the data and the noise. We would prefer to prevent overfitting the data, but for high noise levels, more degrees of freedom may be required in order to better fit the data.

### 5.4. Computational effort of computing approximations

In this section, we compare the computational effort required to compute all four approximations. Fig. 9 and Table 3 show the total CPU time taken by the four approximation approaches when the interpolation data are sampled by using d-LHD. In Fig. 9 each bar is the average CPU time, and the error bars at the top of each bar indicate the standard deviation. The time taken by Algorithm 4.1 is split into the time taken to fit the data by solving Eq. (10) and the time to perform the global minimization of $q(x)$ by using the multistart approach across all iterations. Because the CPU times are generally consistent across the different noise levels,

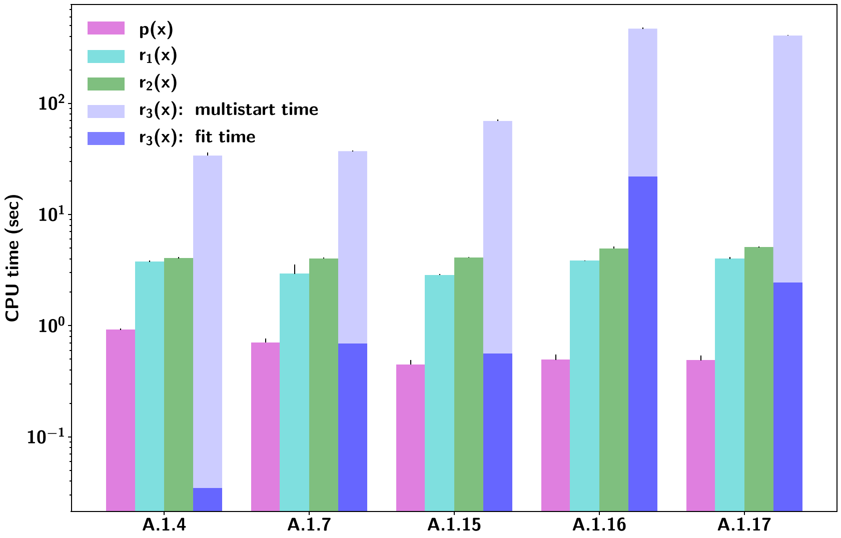


**Fig. 9.** Total CPU time taken by the four approximation approaches when the interpolation data is sampled by using d-LHD and is noise-free. Each bar is the average CPU time, and the error bars at the top of each bar are the standard deviations. The total time taken by Algorithm 4.1 is shown as its fit time and multistart time to perform the global optimization of $q_l(x)$. Function A.5.4 is an exponential function, Function A.5.7 is a rational function, Function A.5.15 is a Breit–Wigner function, and Functions A.5.16 and A.5.17 are functions whose denominator is a polynomial. The multistart time clearly dominates the total time taken by Algorithm 4.1.

we show the results only for the noise-free case. The CPU times for all functions, sampling strategies and noise levels are given in the electronic supplement section SM2.

We observe that the multistart time of Algorithm 4.1 clearly dominates the total CPU time. The reason is that the fitting time grows with the number of iterations of Algorithm 4.1, as shown in Fig. 6. However, the multistart time increases exponentially with the degrees of freedom, as is expected because the global optimization cost grows exponentially. On the other hand, the compute cost of the other approaches grows more slowly with the degrees of freedom because these algorithms are polynomial in time. Despite the computation overhead, the cost of obtaining $r_3(x)$ may be negligible when they are used as surrogates for the expensive simulations of the physics processes, as we show in the next section.

### 5.5. Summary of computational results

From our experiments, we conclude that among the sampling strategies considered, d-LHD performs the best when the goal is to fit a rational approximation to data generated from a black box. We found this result to be true even when d-LHD was compared with the uniform random sampling of points over the domain. The d-LHD method samples both on the faces and on the inside of the domain evenly and requires only a few interpolation points more than the degrees of freedom of the approximation. This is especially useful for applications whose function evaluations are computationally extremely expensive (minutes to hours per evaluation). We also find that the approximations based on d-LHD-generated samples require overall fewer iterations of Algorithm 4.1 and produce better-quality approximations than the LHS-based approximations do.

The approximation approach using Algorithm 3.1 with and without degree reduction is computationally more efficient than the approach using Algorithm 4.1. However, our goal was to develop an approximation method for computationally expensive simulations that performs overall well when the underlying simulation function is unknown (black box). Thus, the computational overhead of the algorithms is negligible; and when applied to a true black-box simulation, Algorithm 4.1 is more likely to give low errors, in particular when the interpolation data are noisy. More specifically, no spurious poles are found in $r_3(x)$ for nonrational functions as well as noisy problems, and the goodness of fit of $r_3(x)$ is much better than $r_1(x)$ or $r_2(x)$. We note here that these claims are based on the assumption that the data are sampled over a domain that does not include any true poles.

The approximation using Algorithm 4.1 is computationally more expensive than the other approaches because it solves a harder problem of removing the poles iteratively. Most of this expense is due to the multistart optimization whose time grows exponentially with the degrees of freedom since it is tasked to the perform global minimization of $q(x)$. As we will see in the next section, however, the additional time to get a pole-free and a superior quality of approximation is a small price to pay considering that this approximation replaces expensive simulations of the physics processes.

## 6. Rational approximation for high energy physics

A common problem in HEP is to infer information on *unobservable* parameters, $x$, from experimentally measured data, $d$. Typically, this is achieved by using a dedicated physics simulation program and statistical measures. Since it is particularly well suited for rational approximations, we will discuss a measure called "binned likelihood" in which the measured and the simulated data take on the form of histograms with the same binning [48]. The binning of the histogram is driven by experimental constraints such as how precise the quantity in question can actually be measured. An illustrative example of the problem setup is shown in Fig. 10.

In our example, the simulation predicts how postulated dark matter particles interact with a Xenon-based detector in a process called "direct detection" (see [49, Sec. 26]). Our physics simulation has three parameters, $x = (m_\chi, c_+, c_\pi)$, which represent the dark matter particle mass and two couplings to ordinary matter. The parameter domain is $[10, 100] \times [0.0001, 0.001] \times [0.001, 0.1]$. The particle mass has the dimension of GeV/c$^2$, while the couplings are dimensionless.

We note that at the time of writing, no experimental result on the direct detection of dark matter has been published.

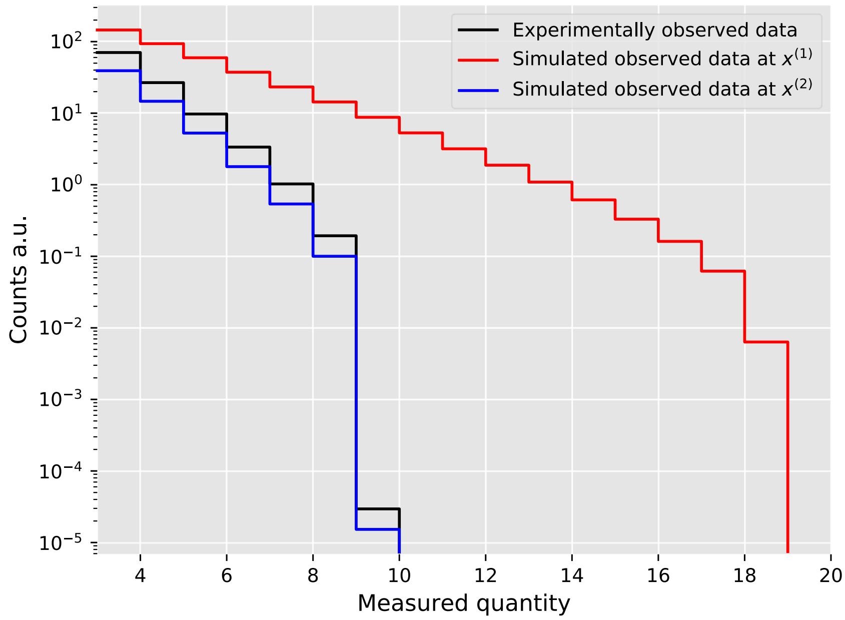


**Fig. 10.** Illustration of a typical problem setup in particle physics. Shown are histogram skylines of observable experimental data (black) with observable predictions coming from simulations at different points $x^{(k)}$ in the same parameter space (blue and red). Numerical comparisons of the experimental with the simulated quantities are typically used to infer quantitative statements on the (unobservable) parameters of the simulation. Here, for example, one would be interested in finding parameter points $x$ such that the corresponding simulation prediction resembles the experimental observation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Here, we assume a signal consistent with a dark matter mass $m_\chi = 10$ GeV/c$^2$ and an interaction strength large enough to produce approximately 100 events in future xenon detectors. The simulated experimental data for each bin is: $\{d_1, d_2, \ldots, d_6\} = \{70.4, 26.7, 9.8, 3.4, 1.0, 0.2\}$. The values $d_b, b = 1, \ldots, 6$, are simulated by using a specific framework of the generalized spin-independent response to dark matter in direct detection experiments [50,51].

At a given point $x^{(k)}$ in the parameter domain, we define the likelihood function $\mathcal{L}(x^{(k)}|d_1, d_2, \ldots, d_6)$ as the product of independent Poisson processes over all bins (generalized for non-integer variables):

$$\mathcal{L}(x^{(k)}|d_1, d_2, \ldots, d_6) = \prod_{b=1}^{6} \frac{N_b(x^{(k)}) d_b e^{N_b(x^{(k)})}}{\Gamma(d_b + 1)}, \tag{20}$$

where the $N_b(x^{(k)})$ denote the simulated quantities for a point $x^{(k)}$ that correspond to the $d_b$; in other words, one simulation returns the values for all bins.

By numerically maximizing Eq. (20), we infer information about the parameters $x$ or regions of the parameter space that yields simulated data consistent with their experimentally observed counterpart. We use MultiNest [52–54] for this purpose, which requires the evaluation of Eq. (20) at tens of thousands[2] of $x^{(k)}$ to succeed. The computational cost of this operation is driven by the cost to obtain $N_b(x^{(k)})$ and can be substantial.

In the following we will discuss how rational approximations can be used to significantly reduce the required CPU cost of maximizing the likelihood. We will show results for rational approximations of degree $M = 4, N = 4$ as well as polynomial approximations of degree 7.[3]

**Table 4**
Comparison of computational cost when maximizing the likelihood Eq. (20) using the true simulation and maximizing the approximate likelihood Eq. (22) using a rational approximation for the data in each bin.

| | Likelihood evaluations | Total run-time [s] |
|---|---|---|
| Using full simulation Fig. 11a | 29 459 | 14 594 |
| Using $r_b$ with Algorithm 4.1, $M = 4, N = 4$ | 29 612 | 288 |

First, we calculate separate rational approximations $r_b(x)$ that approximates $N_b(x)$ for each bin $b$. This calculation requires evaluating the exact simulation at sufficiently many training points. We use $N_{\text{point}} = 500$ points sampled using the Latin hypercube method from the parameter space, $x^{(k)}, k = 1, \ldots, N_{\text{points}}$, at which we evaluate $N_b(x^{(k)}), k = 1, \ldots, N_{\text{points}}$. We compute the $r_b$ from the input–output data pairs

$$\left\{\left(x^{(k)}, N_b(x^{(k)})\right)\right\}_{k=1}^{N_{\text{points}}} \quad \text{for } b = 1, \ldots, N_{\text{bins}}. \tag{21}$$

By replacing the expensive simulations to obtain $N_b(x^{(k)})$ with cheap-to-evaluate rational (or polynomial) approximations $r_b$ in Eq. (20) we can define an approximate likelihood:

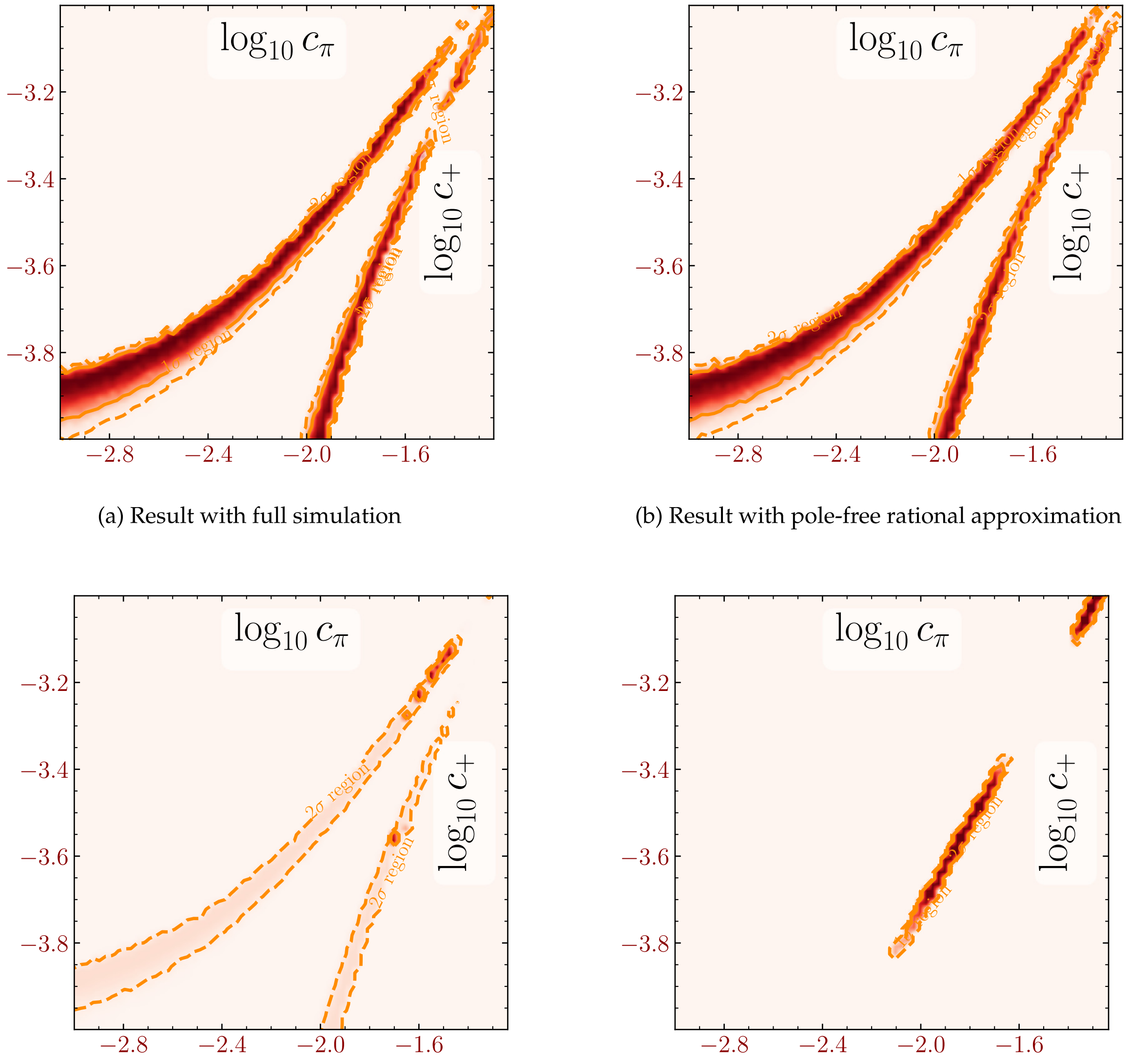
$$\begin{aligned}\mathcal{L}(x^{(k)}|d_1, d_2, \ldots, d_{N_{\text{bins}}}) &\approx \tilde{\mathcal{L}}(x^{(k)}|d_1, d_2, \ldots, d_{N_{\text{bins}}}) \\ &= \prod_{b=1}^{N_{\text{bins}}} \frac{r_b(x^{(k)}) d_b e^{r_b(x^{(k)})}}{\Gamma(d_b + 1)}.\end{aligned} \tag{22}$$

The maximization of Eqs. (20) and (22) requires about 30,000 evaluations of $\mathcal{L}(x^{(k)}|d_1, d_2, \ldots, d_{N_{\text{bins}}})$ and $\tilde{\mathcal{L}}(x^{(k)}|d_1, d_2, \ldots, d_{N_{\text{bins}}})$, respectively. The run time of the latter is, however, about a factor 50 faster (Table 4).

To demonstrate that the results obtained with the rational approximations $r_b$ are in agreement with the full simulation, we present our results in terms of two-dimensional profile-likelihood projections (Fig. 11). We limit the discussion to the projection onto the $c_\pi - c_+$ plane because it exhibits the most interesting

[2] The dimension of the problem and the convergence criteria of the MultiNest algorithm strongly influence the number of required function calls.

[3] The degree is chosen such that the number of coefficients is comparable to the number of coefficients used in the rational approximations.



(a) Result with full simulation

(b) Result with pole-free rational approximation

(c) Result with non-pole-free rational approximation

(d) Result with polynomial approximation

**Fig. 11.** Two-dimensional profile-likelihood projections of a 3-dimensional parameter space with superplot [55]. Regions of higher likelihood are shown darker. The data are normalized to the maximum likelihood observed before plotting. We compare the result obtained with the full physics simulation (Fig. 11a) to the result obtained when using pole-free rational approximations ($M = 4, N = 4$) calculated with the semi-infinite approach (Fig. 11b). Fig. 11c shows the effect of poles in the relevant parameter domain. The poles are visible as dark dots. For completeness, Fig. 11d shows the result when using polynomial approximations with a similar number of coefficients as in Fig. 11b.

pattern. In those plots, dark regions indicate higher likelihood values and therefore high level of compatibility with experimentally observed data.

The top-left plot (Fig. 11a) shows the result obtained with the full simulation Eq. (20). We observe two ridges of equal likelihood, meaning that there are very different parameter combinations that are equally in agreement with the experimentally observed data. This is our ground truth for comparison with the approximation based results.

The result obtained with pole-free (Algorithm 4.1) rational approximations in Fig. 11b is in excellent agreement with the ground truth both qualitatively and quantitatively. The rational approximations obtained with Eq. (4), shown in Fig. 11c, demonstrates the impact of spurious poles: although we find qualitative similarities with the ground truth, the poles that are present in some of the $r_b$ lead to a complete distortion of the evaluated likelihoods and therefore to a quantitatively wrong interpretation. For completeness, we show in Fig. 11d the result obtained with polynomials of order 7. It clearly shows the advantage of rational approximations since the polynomial approximations are apparently not able to capture the true likelihood at all. Thus, the information inferred by using the polynomial approximation would be misleading.

## 7. Conclusions

We have presented two approaches for computing rational approximations for computationally expensive black-box functions. Our first approach uses linear algebra to construct the rational approximation, but it does not guarantee that the approximation is pole-free. Our second approach exploits a semi-infinite optimization problem formulation that leads to accurate rational approximations without poles.

Our numerical study shows that the selection of interpolation points for fitting the approximations has a major impact on the approximation error and the number of iterations taken by the pole-free rational approximation. We find that a Latin hypercube design that is augmented with sample points on the boundary of

**Table A.5**
Description of fast-to-compute test problems. Here, $n$ is the number of variables, $M$ is the degree of the numerator, $N$ is the degree of the denominator, $f$ is the functional form, and *Domain* for each dimension is the interval in which no poles exist. If either the numerator or denominator is not a polynomial, then the entry for $M$ or $N$ is a dash, respectively.

| No. | Description | $n$ | $M$ | $N$ | $f$ | *Domain* |
|---|---|---|---|---|---|---|
| A.5.1 | Function whose denominator is a polynomial | 2 | – | 4 | $\frac{e^{x_1x_2}}{(x_1^2-1.44)(x_2^2-1.44)}$ | $x \in [-1, 1]^2$ |
| A.5.2 | Log function | 2 | – | – | $\log(2.25-x_1^2-x_2^2)$ | $x \in [-1, 1]^2$ |
| A.5.3 | Hyperbolic tangent function | 2 | – | – | $\tanh(5(x_1-x_2))$ | $x \in [-1, 1]^2$ |
| A.5.4 | Exponential function | 2 | – | – | $e^{\frac{-(x_1^2+x_2^2)}{1000}}$ | $x \in [-1, 1]^2$ |
| A.5.5 | Absolute value function | 2 | – | – | $\lvert(x_1-x_2)\rvert^3$ | $x \in [-1, 1]^2$ |
| A.5.6 | Rational function | 2 | 3 | 3 | $\frac{x_1+x_2^3}{x_1x_2^2+1}$ | $x \in [0, 1]^2$ |
| A.5.7 | Rational function | 2 | 2 | 2 | $\frac{x_1^2+x_2^2+x_1-x_2-1}{(x_1-1.1)(x_2-1.1)}$ | $x \in [-1, 1]^2$ |
| A.5.8 | Rational function | 2 | 4 | 4 | $\frac{x_1^4+x_2^4+x_1^2x_2^2+x_1x_2}{(x_1^2-1.1)(x_2^2-1.1)}$ | $x \in [-1, 1]^2$ |
| A.5.9 | Rational function | 4 | 2 | 2 | $\frac{x_1^2+x_2^2+x_1-x_2+1}{(x_3-1.5)(x_4-1.5)}$ | $x \in [-1, 1]^4$ |
| A.5.10 | Rational function | 2 | 2 | 3 | $\frac{x_1^2+x_2^2+x_1-x_2-1}{x_1^3+x_2^3+4}$ | $x \in [-1, 1]^2$ |
| A.5.11 | Rational function | 2 | 3 | 2 | $\frac{x_1^3+x_2^3}{x_1^2+x_2^2+3}$ | $x \in [-1, 1]^2$ |
| A.5.12 | Rational function | 2 | 4 | 4 | $\frac{x_1^4+x_2^4+x_1^2x_2^2+x_1x_2}{x_1^2x_2^2-2x_1^2-2x_2^2+4}$ | $x \in [-1, 1]^2$ |
| A.5.13 | Rational function | 2 | 3 | 4 | $\frac{x_1^3+x_2^3}{x_1^2x_2^2-2x_1^2-2x_2^2+4}$ | $x \in [-1, 1]^2$ |
| A.5.14 | Rational function | 2 | 4 | 3 | $\frac{x_1^4+x_2^4+x_1^2x_2^2+x_1x_2}{x_1^3+x_2^3+4}$ | $x \in [-1, 1]^2$ |
| A.5.15 | Breit–Wigner function | 3 | – | – | $\frac{2\sqrt{2}M\Gamma\gamma}{(\pi\sqrt{M^2+\gamma})[(E^2-M^2)^2+M^2\Gamma^2]}$<br>where $\gamma=\sqrt{M^2(M^2+\Gamma^2)}$ | $E \in [80, 100]$,<br>$\Gamma \in [5, 10]$,<br>$M \in [90, 93]$ |
| A.5.16 | Function whose denominator is a polynomial | 4 | – | 4 | $\frac{\tan^{-1}(x_1)+\cdots+\tan^{-1}(x_4)}{x_1^2x_2^2-x_1^2-x_2^2+1}$ | $x \in [-0.95, 0.95]^4$ |
| A.5.17 | Function whose denominator is a polynomial | 4 | – | 2 | $\frac{e^{x_1x_2x_3x_4}}{x_1^2+x_2^2-x_3x_4+3}$ | $x \in [-1, 1]^4$ |
| A.5.18 | Sinc function | 4 | – | – | $10\prod_{i=1}^{4}\frac{\sin x_i}{x_i}$ | $x \in [10^{-6}, 4\pi]^4$ |
| A.5.19 | Sinc function | 2 | – | – | $10\frac{\sin x_1}{x_1}\frac{\sin x_2}{x_2}$ | $x \in [10^{-6}, 4\pi]^2$ |
| A.5.20 | Polynomial function | 2 | 2 | – | $x_1^2+x_2^2+x_1x_2-x_2+1$ | $x \in [-1, 1]^2$ |

the parameter domain leads to improved approximations more efficiently. We hypothesize that this is due to close proximity of the function domain to the true poles and the approximations fare poorly without the sample points on the boundary. We showed that for a variety of analytic fast-to-compute test problems with and without noise the rational approximations generally perform better than the polynomial approximations do. The result was further confirmed by approximating data generated from an expensive HEP simulation. The polynomial did not capture the true underlying functional relationship at all. Thus, for black-box simulations whose true underlying functional forms are unknown, using a polynomial may lead to incorrect conclusions.

An outstanding challenge for using rational approximations is the determination of the "correct" polynomial degrees in the numerator and denominator. We have experimented with a heuristic method to determine these degrees, but noisy data pose an additional challenge, and more research is needed.

The structural constraint considered in the pole-free rational approximation is mitigating poles through enforcing nonnegativity of the denominator $q(x)$. Other structural constraints arise in the solution of chance-constraint optimization, where we wish to approximate an empirical cumulative density function. By construction, the function should be monotonic, which again imposes a constraint on the rational approximation. Such structural constraints should also be modeled in the future.

The rational approximations require a minimum number of interpolation points to fit the model. One drawback is in obtaining these interpolation points, since the number of points required increases significantly with the number of parameters and the degrees of the polynomials of the approximation. Hence, obtaining these interpolation points may become computationally too expensive. Additionally, the multistart global optimization of the denominator $q(x)$ in the pole-free rational approximation will become computationally significantly more expensive as the number of parameters increase. We have tested problems with up to 7 parameters, but especially in high energy physics dozens of parameters are commonly encountered. Thus, the question of scalability of the proposed rational approximation approaches must be addressed in the future.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. Support for this work was provided through the SciDAC program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research. This work was also supported by the U.S. Department of Energy through grant DE-FG02-05ER25694, and by Fermi Research Alliance, LLC, United States of America under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of Nuclear Physics, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute under Contract No. DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. http://energy.gov/downloads/doe-public-access-plan.

## Appendix A. Description of test problems

The functional forms of the test problems that we use in our numerical experiments are shown in Table A.5.

## Appendix B. Checking for poles

In this section, we discuss practical ways to solve the global optimization subproblem in line 4 of Algorithm 4.1. We compare different strategies to perform the global minimization of $q_l(x)$ to detect poles in $D$ for a number of fast-to-compute test functions described in Table A.5. The benchmark for the comparison is the Baron global optimization solver for nonlinear and mixed-integer nonlinear problems [56,57]. The other strategies include "singlestart", in which we choose one point randomly from $D$ as starting point for the optimization; "multistart", which starts multiple optimizations from different points in $D$; and "sampling", where $q_l(x)$ is evaluated at multiple random points in $D$ to check if any evaluation of $q_l(x) < 0$. We allow multistart and sampling to run for the same amount of time as Baron to ensure a fair comparison of these approaches. However, multistart and sampling stop as soon as the first $x$ with $q_l(x) < 0$ is detected and do not continue toward finding the global minimum.

The results from this comparison are summarized in Table B.6. We observe that multistart detects poles almost as well as Baron in a much shorter time. The reason is that multistart stops as soon as some $x$ with $q_l(x) < 0$ is detected, whereas Baron tries to solve the problem to optimality in each iteration of Algorithm 4.1. Also, multistart detects poles almost as well as Baron does when the time taken by both approaches is the same. Therefore, to set a suitable time limit for multistart a priori, we estimate the amount of time Baron would take to solve the problem given the number of nonlinearities. Then we compute the number of multistart iterations that can be completed within this time. The goal of this heuristic is to minimize the occurrence of poles in $q(x)$ without spending the effort required to run Baron. The number of multistart iterations needed is approximately an exponential function, $\phi$, of the number of nonlinearities, nnl, when multistart ran for the same time as Baron.

$$\phi(\text{nnl}) = 2042.023e^{0.029\text{nnl}} \tag{B.1}$$

**Table B.6**
Comparing global optimization strategies for detecting poles. Here, $n$ denotes the number of variables, *nnl* is the number of nonlinearities in $q_l(x)$ that is obtained by subtracting the constant and linear terms from the total degrees of freedom of $q_l(x)$. *Time* is the CPU time in seconds, and *%FN* is the percentage of false negatives for detecting poles, that is, when Baron identifies the existence of a pole, while the corresponding other method did not. The results are more informative for problems where $n > 2$. Hence, results are only obtained for some functions with $n = 2$.

| Function No. | $n$ | *nnl* | Baron | Singlestart | | Multistart | | Sampling | |
|---|---|---|---|---|---|---|---|---|---|
| | | | *Time* | *% FN* | *Time* | *% FN* | *Time* | *% FN* | *Time* |
| A.5.12 | 2 | 7 | 0.0809 | 0.68 | 0.0021 | 0.00 | 0.0319 | 1.35 | 0.0306 |
| A.5.13 | 2 | 7 | 0.0575 | 2.67 | 0.0017 | 0.00 | 0.0541 | 2.00 | 0.0539 |
| A.5.14 | 2 | 7 | 0.0564 | 1.29 | 0.0018 | 0.00 | 0.0506 | 0.65 | 0.0503 |
| A.5.15 | 3 | 16 | 0.1066 | 9.66 | 0.0057 | 0.00 | 0.0742 | 1.70 | 0.0743 |
| A.5.16 | 4 | 30 | 0.2653 | 23.63 | 0.0082 | 1.10 | 0.0757 | 19.23 | 0.1270 |
| A.5.17 | 4 | 30 | 0.1202 | 0.00 | 0.0051 | 0.00 | 0.0539 | 5.00 | 0.0756 |
| A.5.18 in 7D | 7 | 112 | 259.0448 | 0.29 | 0.0078 | 0.00 | 0.3549 | 2.20 | 0.4579 |

## Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2020.107663.

## References

[1] A. Booker, J. Dennis Jr, P. Frank, D. Serafini, V. Torczon, M. Trosset, Struct. Multidiscip. Optim. 17 (1999) 1–13.
[2] G. Matheron, Econ. Geol. 58 (1963) 1246–1266.
[3] M. Powell, Advances in Numerical Analysis, Vol. 2: Wavelets, Subdivision Algorithms and Radial Basis Functions, Oxford University Press, Oxford, Oxford University Press, London, 1992, pp. 105–210, Chapter. The Theory of Radial Basis Function Approximation in 1990.
[4] J. Friedman, Ann. Statist. 19 (1991) 1–141.
[5] R. Myers, D. Montgomery, Response Surface Methodology, Process and Product Optimization using Designed Experiments, Wiley-Interscience Publication, 1995.
[6] R. Devore, X.-M. Yu, Trans. Amer. Math. Soc. 293 (1) (1986) 161–169.
[7] D.J. Newman, Michigan Math. J. 11 (1964) 11–14.
[8] P. Gonnet, R. Pachón, L.N. Trefethen, Electron. Trans. Numer. Anal. 38 (2011) 146–167.
[9] R. Pachón, P. Gonnet, J. Van Deun, SIAM J. Numer. Anal. 50 (3) (2012) 1713–1734, http://dx.doi.org/10.1137/100797291.
[10] D. Braess, Nonlinear Approximation Theory, Springer, Berlin, 1986.
[11] E.W. Cheney, Introduction to Approximation Theory, McGraw-Hill, New York, 1966.
[12] L.N. Trefethen, Approximation Theory and Approximation Practice, SIAM, Philadelphia, 2013.
[13] A.A.M. Cuyt, J. Math. Anal. Appl. 96 (1983) 283–293, http://dx.doi.org/10.1016/0022-247x(83)90041-0.
[14] A.A.M. Cuyt, B.M. Verdonk, Computing 34 (1985) 41–61, http://dx.doi.org/10.1007/bf02242172.
[15] A. Cuyt, X. Yang, Numer. Algorithms 55 (2010) 233–243.
[16] P. Seshadri, P. Constantine, P. Gonnet, G. Parks, S. Shahpar, 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2013, http://dx.doi.org/10.2514/6.2013-1680.
[17] G.H. Golub, V. Pereyra, SIAM J. Numer. Anal. 10 (2) (1973) 413–432.
[18] C.F. Borges, Electron. Trans. Numer. Anal. 35 (2009) 57–68.
[19] Y. Nakatsukasa, O. Sète, L.N. Trefethen, SIAM J. Sci. Comput. 40 (2018) A1494–A1522, http://dx.doi.org/10.1137/16m1106122.

[20] S.-I. Filip, Y. Nakatsukasa, L.N. Trefethen, B. Beckermann, SIAM J. Sci. Comput. 40 (2018) A2427–A2455, http://dx.doi.org/10.1137/17m1132409.

[21] A. Cuyt, O. Salazar Celis, BIT 59 (2019) 35–55, http://dx.doi.org/10.1007/s10543-018-0721-1.

[22] A. Cuyt, O. Salazar Celis, M. Lukach, Multidimens. Syst. Signal Process. 25 (2014) 447–471, http://dx.doi.org/10.1007/s11045-012-0208-1.

[23] O. Salazar Celis, A. Cuyt, B. Verdonk, Numer. Algorithms 45 (2007) 375–388.

[24] C.B. Dunham, J. Approx. Theory 37 (1) (1983) 5–11.

[25] E. Kaufman Jr, G. Taylor, J. Approx. Theory 32 (1) (1981) 9–26.

[26] R. Hettich, K.O. Kortanek, SIAM Rev. 35 (3) (1993) 380–429, http://dx.doi.org/10.1137/1035089.

[27] A. Buckley, et al., Phys. Rep. 504 (2011) 145–233, http://dx.doi.org/10.1016/j.physrep.2011.03.005, arXiv:1101.2599.

[28] J. Albrecht, et al., HEP Software Foundation Collaboration Collaboration, Comput. Softw. Big Sci. 3 (1) (2019) 7, http://dx.doi.org/10.1007/s41781-018-0018-8, arXiv:1712.06982.

[29] P. Gonnet, S. Güttel, L.N. Trefethen, SIAM Rev. 55 (1) (2013) 101–117, http://dx.doi.org/10.1137/110853236.

[30] M. Carter, B. van Brunt, The Lebesgue-Stieltjes Integral: A Practical Introduction, in: Undergraduate Texts in Mathematics, Springer New York, 2000, URL https://books.google.com/books?id=qgiLag9dPpMC.

[31] M. Huhtanen, R.M. Larsen, BIT 42 (2002) 393–407, http://dx.doi.org/10.1023/A:1021907210628.

[32] M. Zaccaron, Discrete Orthogonal Polynomials and Hyperinterpolation Over Planar Regions (M.Sc. thesis), University of Padova, 2014.

[33] D. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, third ed., Springer, New York, 2010.

[34] L. Giraud, J. Langou, M. Rozložník, J. van den Eshof, Numer. Math. 101 (2005) 87–100.

[35] S.J. Leon, Å. Björck, W. Gander, Numer. Linear Algebra Appl. 20 (2013) 492–532, http://dx.doi.org/10.1002/nla.1839.

[36] B.N. Parlett, The Symmetric Eigenvalue Problem, SIAM, Philadelphia, 1998.

[37] O. Stein, Bi-Level Strategies in Semi-Infinite Programming, Vol. 71, Springer Science & Business Media, 2013.

[38] A. Ben-Tal, L. El Ghaoui, A. Nemirovski, Robust Optimization, Princeton University Press, 2009.

[39] E. Levitin, B. Polyak, USSR Comput. Math. Math. Phys. 6 (5) (1966) 1–50.

[40] H. Sherali, W. Adams, A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, Kluwer, Dordrecht, 1998.

[41] T. Pomentale, Numer. Math. 12 (1968) 40–46, http://dx.doi.org/10.1007/bf02170995.

[42] G. Breit, E. Wigner, Phys. Rev. 49 (1936) 519–531, http://dx.doi.org/10.1103/PhysRev.49.519, URL https://link.aps.org/doi/10.1103/PhysRev.49.519.

[43] A.R. Bohm, Y. Sato, Phys. Rev. D71 (2005) 085018, http://dx.doi.org/10.1103/PhysRevD.71.085018, arXiv:hep-ph/0412106.

[44] V. Barthelmann, E. Novak, K. Ritter, Adv. Comput. Math. 12 (4) (2000) 273–288.

[45] M.D. McKay, R.J. Beckman, W.J. Conover, Technometrics 21 (2) (1979) 239–245.

[46] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK users' guide, third ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[47] E.R. Leatherman, A.M. Dean, T.J. Santner, Comput. Statist. Data Anal. 113 (2017) 346–362, http://dx.doi.org/10.1016/j.csda.2016.07.013, URL http://www.sciencedirect.com/science/article/pii/S0167947316301773.

[48] R.J. Barlow, C. Beeston, Comput. Phys. Comm. 77 (1993) 219–228, http://dx.doi.org/10.1016/0010-4655(93)90005-W.

[49] M. Tanabashi, K. Hagiwara, K. Hikasa, K. Nakamura, Y. Sumino, F. Takahashi, J. Tanaka, K. Agashe, G. Aielli, C. Amsler, M. Antonelli, D.M. Asner, H. Baer, S. Banerjee, R.M. Barnett, T. Basaglia, C.W. Bauer, J.J. Beatty, V.I. Belousov, J. Beringer, S. Bethke, A. Bettini, H. Bichsel, O. Biebel, K.M. Black, E. Blucher, O. Buchmuller, V. Burkert, M.A. Bychkov, R.N. Cahn, M. Carena, A. Ceccucci, A. Cerri, D. Chakraborty, M.-C. Chen, R.S. Chivukula, G. Cowan, O. Dahl, G. D'Ambrosio, T. Damour, D. de Florian, A. de Gouvêa, T. DeGrand, P. de Jong, G. Dissertori, B.A. Dobrescu, M. D'Onofrio, M. Doser, M. Drees, H.K. Dreiner, D.A. Dwyer, P. Eerola, S. Eidelman, J. Ellis, J. Erler, V.V. Ezhela, W. Fetscher, B.D. Fields, R. Firestone, B. Foster, A. Freitas, H. Gallagher, L. Garren, H.-J. Gerber, G. Gerbier, T. Gershon, Y. Gershtein, T. Gherghetta, A.A. Godizov, M. Goodman, C. Grab, A.V. Gritsan, C. Grojean, D.E. Groom, M. Grünewald, A. Gurtu, T. Gutsche, H.E. Haber, C. Hanhart, S. Hashimoto, Y. Hayato, K.G. Hayes, A. Hebecker, S. Heinemeyer, B. Heltsley, J.J. Hernández-Rey, J. Hisano, A. Höcker, J. Holder, A. Holtkamp, T. Hyodo, K.D. Irwin, K.F. Johnson, M. Kado, M. Karliner, U.F. Katz, S.R. Klein, E. Klempt, R.V. Kowalewski, F. Krauss, M. Kreps, B. Krusche, Y.V. Kuyanov, Y. Kwon, O. Lahav, J. Laiho, J. Lesgourgues, A. Liddle, Z. Ligeti, C.-J. Lin, C. Lippmann, T.M. Liss, L. Littenberg, K.S. Lugovsky, S.B. Lugovsky, A. Lusiani, Y. Makida, F. Maltoni, T. Mannel, A.V. Manohar, W.J. Marciano, A.D. Martin, A. Masoni, J. Matthews, U.-G. Meißner, D. Milstead, R.E. Mitchell, K. Mönig, P. Molaro, F. Moortgat, M. Moskovic, H. Murayama, M. Narain, P. Nason, S. Navas, M. Neubert, P. Nevski, Y. Nir, K.A. Olive, S. Pagan Griso, J. Parsons, C. Patrignani, J.A. Peacock, M. Pennington, S.T. Petcov, V.A. Petrov, E. Pianori, A. Piepke, A. Pomarol, A. Quadt, J. Rademacker, G. Raffelt, B.N. Ratcliff, P. Richardson, A. Ringwald, S. Roesler, S. Rolli, A. Romaniouk, L.J. Rosenberg, J.L. Rosner, G. Rybka, R.A. Ryutin, C.T. Sachrajda, Y. Sakai, G.P. Salam, S. Sarkar, F. Sauli, O. Schneider, K. Scholberg, A.J. Schwartz, D. Scott, V. Sharma, S.R. Sharpe, T. Shutt, M. Silari, T. Sjöstrand, P. Skands, T. Skwarnicki, J.G. Smith, G.F. Smoot, S. Spanier, H. Spieler, C. Spiering, A. Stahl, S.L. Stone, T. Sumiyoshi, M.J. Syphers, K. Terashi, J. Terning, U. Thoma, R.S. Thorne, L. Tiator, M. Titov, N.P. Tkachenko, N.A. Törnqvist, D.R. Tovey, G. Valencia, R. Van de Water, N. Varelas, G. Venanzoni, L. Verde, M.G. Vincter, P. Vogel, A. Vogt, S.P. Wakely, W. Walkowiak, C.W. Walter, D. Wands, D.R. Ward, M.O. Wascko, G. Weiglein, D.H. Weinberg, E.J. Weinberg, M. White, L.R. Wiencke, S. Willocq, C.G. Wohl, J. Womersley, C.L. Woody, R.L. Workman, W.-M. Yao, G.P. Zeller, O.V. Zenin, R.-Y. Zhu, S.-L. Zhu, F. Zimmermann, P.A. Zyla, J. Anderson, L. Fuller, V.S. Lugovsky, P. Schaffner, Particle Data Group Collaboration Collaboration, Phys. Rev. D 98 (2018) 030001, http://dx.doi.org/10.1103/PhysRevD.98.030001, URL https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[50] M. Hoferichter, P. Klos, J. Menéndez, A. Schwenk, Phys. Rev. D94 (6) (2016) 063505, http://dx.doi.org/10.1103/PhysRevD.94.063505, arXiv:1605.08043.

[51] D.G. Cerdeño, A. Cheek, E. Reid, H. Schulz, J. Cosmol. Astropart. Phys. 1808 (08) (2018) 011, http://dx.doi.org/10.1088/1475-7516/2018/08/011, arXiv:1802.03174.

[52] F. Feroz, M.P. Hobson, M. Bridges, Mon. Not. R. Astron. Soc. 398 (2009) 1601–1614, http://dx.doi.org/10.1111/j.1365-2966.2009.14548.x, arXiv:0809.3437.

[53] F. Feroz, M.P. Hobson, E. Cameron, A.N. Pettitt, Instrum. Methods Astrophys. (2013) arXiv:1306.2144.

[54] J. Buchner, A. Georgakakis, K. Nandra, L. Hsu, C. Rangel, M. Brightman, A. Merloni, M. Salvato, J. Donley, D. Kocevski, aap 564 (2014) A125, http://dx.doi.org/10.1051/0004-6361/201322971, arXiv:1402.0004.

[55] A. Fowlie, M.H. Bardsley, Eur. Phys. J. Plus 131 (11) (2016) 391, http://dx.doi.org/10.1140/epjp/i2016-16391-0, arXiv:1603.00555.

[56] M. Tawarmalani, N.V. Sahinidis, Math. Program. 103 (2005) 225–249.

[57] N.V. Sahinidis, BARON 17.8.9: Global optimization of mixed-integer nonlinear programs, user's manual, 2017.