


Metadata of the chapter that will be visualized in SpringerLink

Book Title	Monte Carlo and Quasi-Monte Carlo Methods	
Series Title		
Chapter Title	An Adaptive Algorithm Employing Continuous Linear Functionals	
Copyright Year	2020	
Copyright HolderName	Springer Nature Switzerland AG	
Corresponding Author	Family Name	Ding
	Particle	
	Given Name	Yuhan
	Prefix	
	Suffix	
	Role	
	Division	Department of Applied Mathematics
	Organization	Illinois Institute of Technology
	Address	RE 208, 10 W. 32nd St., Chicago, IL, 60616, USA
	Email	yding2@hawk.iit.edu
Author	Family Name	Hickernell
	Particle	
	Given Name	Fred J.
	Prefix	
	Suffix	
	Role	
	Division	Center for Interdisciplinary Scientific Computation and Department of Applied Mathematics
	Organization	Illinois Institute of Technology
	Address	RE 208, 10 W. 32nd St., Chicago, IL, 60616, USA
	Email	hickernell@iit.edu
Author	Family Name	Rugama
	Particle	
	Given Name	Lluís Antoni Jiménez 
	Prefix	
	Suffix	
	Role	
	Division	Department of Applied Mathematics
	Organization	Illinois Institute of Technology
	Address	RE 208, 10 W. 32nd St., Chicago, IL, 60616, USA
	Email	ljimene1@hawk.iit.edu
Abstract	Automatic algorithms attempt to provide approximate solutions that differ from exact solutions by no more than a user-specified error tolerance. This paper describes an automatic, adaptive algorithm for approximating the solution to a general linear problem defined on Hilbert spaces. The algorithm employs continuous linear functionals of the input function, specifically Fourier coefficients. We assume that the Fourier coefficients of the solution decay sufficiently fast, but we do not require the decay rate to be known	

a priori. We also assume that the Fourier coefficients decay steadily, although not necessarily monotonically. Under these assumptions, our adaptive algorithm is shown to produce an approximate solution satisfying the desired error tolerance, without prior knowledge of the norm of the function to be approximated. Moreover, the computational cost of our algorithm is shown to be essentially no worse than that of the optimal algorithm. We provide a numerical experiment to illustrate our algorithm.

Keywords



An Adaptive Algorithm Employing Continuous Linear Functionals



Yuhan Ding, Fred J. Hickernell and Lluís Antoni Jiménez Rugama

Abstract Automatic algorithms attempt to provide approximate solutions that differ from exact solutions by no more than a user-specified error tolerance. This paper describes an automatic, adaptive algorithm for approximating the solution to a general linear problem defined on Hilbert spaces. The algorithm employs continuous linear functionals of the input function, specifically Fourier coefficients. We assume that the Fourier coefficients of the solution decay sufficiently fast, but we do not require the decay rate to be known a priori. We also assume that the Fourier coefficients decay steadily, although not necessarily monotonically. Under these assumptions, our adaptive algorithm is shown to produce an approximate solution satisfying the desired error tolerance, without prior knowledge of the norm of the function to be approximated. Moreover, the computational cost of our algorithm is shown to be essentially no worse than that of the optimal algorithm. We provide a numerical experiment to illustrate our algorithm.

Keywords ■■■

1 Introduction

Adaptive algorithms determine the design and sample size needed to solve problems to the desired accuracy based on the input function data sampled. A priori upper

Y. Ding (✉) · L. A. J. Rugama
Department of Applied Mathematics, Illinois Institute of Technology, RE 208, 10 W. 32nd St.,
Chicago, IL 60616, USA
e-mail: yding2@hawk.iit.edu

L. A. J. Rugama
e-mail: ljimene1@iit.edu

F. J. Hickernell
Center for Interdisciplinary Scientific Computation and Department of Applied Mathematics,
Illinois Institute of Technology, RE 208, 10 W. 32nd St., Chicago, IL 60616, USA
e-mail: hickernell@iit.edu

© Springer Nature Switzerland AG 2020

B. Tuffin and P. L' Ecuyer (eds.), *Monte Carlo and Quasi-Monte Carlo Methods*,
Springer Proceedings in Mathematics & Statistics 324,
https://doi.org/10.1007/978-3-030-43465-6_8



bounds on some norm of the input function are not needed, but some underlying assumptions about the input function are required for the adaptive algorithm to succeed. Here we consider *general linear problems* where a finite number of series coefficients of the input function are used to obtain an approximate solution. The proposed algorithm produces an approximation with guaranteed accuracy. Moreover, we demonstrate that the computational cost of our algorithm is essentially no worse than that of the best possible algorithm. Our adaptive algorithm is defined on a *cone* of input functions.

1.1 Input and Output Spaces

Let \mathcal{F} be a separable Hilbert space of inputs with orthonormal basis $(u_i)_{i \in \mathbb{N}}$, and let \mathcal{G} be a separable Hilbert space of outputs with orthonormal basis $(v_i)_{i \in \mathbb{N}}$. Based on Parseval's identity the norms for these two spaces may be expressed as the ℓ^2 -norms of their series coefficients:

$$f = \sum_{i \in \mathbb{N}} \hat{f}_i u_i \in \mathcal{F}, \quad \|f\|_{\mathcal{F}} = \|(\hat{f}_i)_{i \in \mathbb{N}}\|_2, \quad (1a)$$

$$g = \sum_{i \in \mathbb{N}} \hat{g}_i v_i \in \mathcal{G}, \quad \|g\|_{\mathcal{G}} = \|(\hat{g}_i)_{i \in \mathbb{N}}\|_2. \quad (1b)$$

Let these two bases be chosen so that the linear solution operator, $S : \mathcal{F} \rightarrow \mathcal{G}$, satisfies

$$S(u_i) = \lambda_i v_i, \quad i \in \mathbb{N}, \quad S(f) = \sum_{i \in \mathbb{N}} \lambda_i \hat{f}_i v_i, \quad (1c)$$

$$\lambda_1 \geq \lambda_2 \geq \dots > 0, \quad \lim_{i \rightarrow \infty} \lambda_i = 0, \quad \|S\|_{\mathcal{F} \rightarrow \mathcal{G}} := \sup_{f \neq 0} \frac{\|S(f)\|_{\mathcal{G}}}{\|f\|_{\mathcal{F}}} = \lambda_1. \quad (1d)$$

This setting includes, for example, the recovery of functions, derivatives, indefinite integrals, and solutions of linear (partial) differential equations. We focus on cases where the exact solution requires an infinite number of series coefficients, \hat{f}_i , in general, i.e., all λ_i are positive.

The existence of the $(u_i)_{i \in \mathbb{N}}$, $(v_i)_{i \in \mathbb{N}}$, and $(\lambda_i)_{i \in \mathbb{N}}$ for a given \mathcal{F} , \mathcal{G} , and S follows from the singular value decomposition. The singular value decomposition with singular values tending to zero (but never equal to zero) exists if (and only if) the solution operator is compact and not finite rank. The ease of identifying explicit expressions for these quantities depends on the particular problem of interest. Alternatively, one may start with a choice of $(u_i)_{i \in \mathbb{N}}$, $(v_i)_{i \in \mathbb{N}}$, and $(\lambda_i)_{i \in \mathbb{N}}$, which then determine the solution operator, S , and the spaces \mathcal{F} and \mathcal{G} .

Example 1 in Sect. 1.3 illustrates this general setting. Section 5 provides a numerical example based on this example.

1.2 Solvability

Let \mathcal{H} be any subset of \mathcal{F} , and let $\mathcal{A}(\mathcal{H})$ denote the set of deterministic algorithms that successfully approximate the solution operator $S : \mathcal{H} \rightarrow \mathcal{G}$ within the specified error tolerance for all inputs in \mathcal{H} :

$$\mathcal{A}(\mathcal{H}) := \{\text{algorithms } A : \mathcal{H} \times (0, \infty) \rightarrow \mathcal{G} : \|S(f) - A(f, \varepsilon)\|_{\mathcal{G}} \leq \varepsilon \forall f \in \mathcal{H}, \varepsilon > 0\}. \quad (2)$$

Algorithms in $\mathcal{A}(\mathcal{H})$ are allowed to sample adaptively any bounded, linear functionals of the input function. They must sample only a finite number of linear functionals for each input function and positive error tolerance. The definition of \mathcal{H} can be used to construct algorithms in $\mathcal{A}(\mathcal{H})$, but no other a priori knowledge about the input functions is available. Following [1] we call a problem *solvable* for inputs \mathcal{H} if $\mathcal{A}(\mathcal{H})$ is non-empty.

Our problem is not solvable for the whole Hilbert space, \mathcal{F} , as can be demonstrated by contradiction. For any potential algorithm, we show that there exists some $f \in \mathcal{F}$, that looks like 0 to the algorithm, but for which $S(f)$ is far from $S(0) = 0$. Choose any $A \in \mathcal{A}(\mathcal{F})$ and $\varepsilon > 0$, and let L_1, \dots, L_n be the linear functionals that are used to compute $A(0, \varepsilon)$. Since the output space, \mathcal{G} , is infinite dimensional and n is finite, there exists some nonzero $f \in \mathcal{F}$ satisfying that $L_1(f) = \dots = L_n(f) = 0$ with non-zero $S(f)$. This means that $A(\pm cf, \varepsilon) = A(0, \varepsilon)$ for any real c , and $A(\pm cf, \varepsilon)$ both have approximation error no greater than ε , i.e.,

$$\begin{aligned} \varepsilon &\geq \frac{1}{2} [\|S(cf) - A(cf, \varepsilon)\|_{\mathcal{G}} + \|S(-cf) - A(-cf, \varepsilon)\|_{\mathcal{G}}] \\ &= \frac{1}{2} [\|cS(f) - A(0, \varepsilon)\|_{\mathcal{G}} + \|-cS(f) - A(0, \varepsilon)\|_{\mathcal{G}}] \\ &\geq \|cS(f)\|_{\mathcal{G}} = |c| \|S(f)\|_{\mathcal{G}} \quad \text{by the triangle inequality.} \end{aligned}$$

Since $S(f) \neq 0$, it is impossible for this inequality to hold for all real c . The presumed A does not exist, $\mathcal{A}(\mathcal{F})$ is empty, and our problem is not solvable for \mathcal{F} . However, our problem is solvable for well-chosen subsets of \mathcal{F} , as is shown in the sections below.

1.3 Computational Cost of the Algorithm and Complexity of the Problem

The computational cost of an algorithm $A \in \mathcal{A}(\mathcal{H})$ for $f \in \mathcal{H}$ and error tolerance ε is denoted $\text{cost}(A, f, \varepsilon)$, and is defined as the number of linear functional values required to produce $A(f, \varepsilon)$. By overloading the notation, we define the cost of algorithms for sets of inputs, \mathcal{H} , as

$$\text{cost}(A, \mathcal{H}, \varepsilon) := \sup\{\text{cost}(A, f, \varepsilon) : f \in \mathcal{H}\} \quad \forall \varepsilon > 0.$$

For unbounded sets, \mathcal{H} , this cost may be infinite. Therefore, it is meaningful to define the cost of algorithms for input functions in $\mathcal{H} \cap \mathcal{B}_\rho$, where $\mathcal{B}_\rho := \{f \in \mathcal{F} : \|f\|_{\mathcal{F}} \leq \rho\}$ is the ball of radius ρ :

$$\text{cost}(A, \mathcal{H}, \varepsilon, \rho) := \sup\{\text{cost}(A, f, \varepsilon) : f \in \mathcal{H} \cap \mathcal{B}_\rho\} \quad \forall \rho > 0, \varepsilon > 0.$$

Finally, we define the complexity of the problem as the computational cost of the best algorithm:

$$\begin{aligned} \text{comp}(\mathcal{A}(\mathcal{H}), \varepsilon) &:= \min_{A \in \mathcal{A}(\mathcal{H})} \text{cost}(A, \mathcal{H}, \varepsilon), \\ \text{comp}(\mathcal{A}(\mathcal{H}), \varepsilon, \rho) &:= \min_{A \in \mathcal{A}(\mathcal{H})} \text{cost}(A, \mathcal{H}, \varepsilon, \rho). \end{aligned}$$

Note that $\text{comp}(\mathcal{A}(\mathcal{H}), \varepsilon, \rho) \geq \text{comp}(\mathcal{A}(\mathcal{H} \cap \mathcal{B}_\rho), \varepsilon)$. In the former case, the algorithm is unaware that the input function has norm no greater than ρ .

An optimal algorithm for \mathcal{B}_ρ can be constructed in terms of interpolation with respect to the first n series coefficients of the input, namely,

$$A_n(f) := \sum_{i=1}^n \lambda_i \widehat{f}_i v_i, \quad (3)$$

$$\|S(f) - A_n(f)\|_{\mathcal{G}} = \left\| (\lambda_i \widehat{f}_i)_{i=n+1}^{\infty} \right\|_2 \leq \lambda_{n+1} \|f\|_{\mathcal{F}}. \quad (4)$$

Define the non-adaptive algorithm as

$$\widehat{A}(f, \varepsilon) = A_{\widehat{n}}(f), \quad \text{where } \widehat{n} = \min\{n : \lambda_{n+1} \leq \varepsilon/\rho\}, \quad \widehat{A} \in \mathcal{A}(\mathcal{B}_\rho). \quad (5)$$

This algorithm is optimal among algorithms in $\mathcal{A}(\mathcal{B}_\rho)$, i.e.,

$$\text{comp}(\mathcal{A}(\mathcal{B}_\rho), \varepsilon) = \text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon) = \min\{n : \lambda_{n+1} \leq \varepsilon/\rho\}.$$

To prove this, let A^* be an arbitrary algorithm in $\mathcal{A}(\mathcal{B}_\rho)$, and let L_1, \dots, L_{n^*} be the linear functionals chosen when evaluating this algorithm for the zero function with tolerance ε . Thus, $A^*(0, \varepsilon)$ is some function of $(L_1(0), \dots, L_{n^*}(0)) = (0, \dots, 0)$. Let f be a linear combination of u_1, \dots, u_{n^*+1} with norm ρ satisfying $L_1(f) = \dots = L_{n^*}(f) = 0$, then $A^*(\pm f, \varepsilon) = A^*(0, \varepsilon)$, and

$$\begin{aligned}
113 \quad \varepsilon &\geq \max_{\pm} \|S(\pm f) - A^*(\pm f, \varepsilon)\|_{\mathcal{G}} = \max_{\pm} \|\pm S(f) - A^*(0, \varepsilon)\|_{\mathcal{G}} \\
114 \quad &\geq \frac{1}{2} [\|S(f) - A^*(0, \varepsilon)\|_{\mathcal{G}} + \|-S(f) - A^*(0, \varepsilon)\|_{\mathcal{G}}] \\
115 \quad &\geq \|S(f)\|_{\mathcal{G}} = \left\| (\lambda_i \widehat{f}_i)_{i=1}^{n^*+1} \right\|_2 \\
116 \quad &\geq \lambda_{n^*+1} \left\| (\widehat{f}_i)_{i=1}^{n^*+1} \right\|_2 = \lambda_{n^*+1} \|f\|_{\mathcal{F}} = \lambda_{n^*+1} \rho. \\
117
\end{aligned}$$

118 Thus, $\lambda_{n^*+1} \leq \varepsilon/\rho$, and

$$119 \quad \text{cost}(A^*, \mathcal{B}_\rho, \varepsilon) \geq \text{cost}(A^*, 0, \varepsilon) = n^* \geq \min\{n : \lambda_{n+1} \leq \varepsilon/\rho\} = \text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon).$$

120 Hence, the algorithm \widehat{A} defined in (5) is optimal for $\mathcal{A}(\mathcal{B}_\rho)$.

121 **Example 1** Consider the case of function approximation for periodic functions
 122 defined over $[0, 1]$, and the algorithm \widehat{A} defined in (5):

$$\begin{aligned}
123 \quad f &= \sum_{k \in \mathbb{Z}} \widehat{f}(k) \widehat{u}_k = \sum_{i \in \mathbb{N}} \widehat{f}_i u_i, & S(f) &= \sum_{k \in \mathbb{Z}} \widehat{f}(k) \widehat{\lambda}_k \widehat{v}_k = \sum_{i \in \mathbb{N}} \widehat{f}_i \lambda_i v_i, \\
124 \quad \widehat{v}_k(x) &:= \begin{cases} 1, & k = 0, \\ \sqrt{2} \sin(2\pi kx), & k > 0, \\ \sqrt{2} \cos(2\pi kx), & k < 0, \end{cases} & v_i &= \begin{cases} \widehat{v}_{-i/2}, & i \text{ even}, \\ \widehat{v}_{(i-1)/2}, & i \text{ odd}, \end{cases} \\
125 \quad \widehat{\lambda}_k &:= \begin{cases} 1, & k = 0, \\ \frac{1}{|k|^r}, & k \neq 0, \end{cases} & \lambda_i &= \widehat{\lambda}_{\lfloor i/2 \rfloor} = \frac{1}{\max(1, \lfloor i/2 \rfloor)^r}, \\
126 \quad \widehat{u}_k &:= \widehat{\lambda}_k \widehat{v}_k, & u_i &= \lambda_i v_i = \begin{cases} \widehat{u}_{-i/2}, & i \text{ even}, \\ \widehat{u}_{(i-1)/2}, & i \text{ odd}, \end{cases} \\
127 \quad \|f\|_{\mathcal{F}}^2 &= \left(\int_0^1 f(x) dx \right)^2 + \frac{\|f^{(r)}\|_2^2}{(2\pi)^{2r}}, \quad r \in \mathbb{N}, & \widehat{f}_i &= \begin{cases} \widehat{f}(-i/2), & i \text{ even}, \\ \widehat{f}((i-1)/2), & i \text{ odd}, \end{cases} \\
128 \quad \|g\|_{\mathcal{G}} &= \|g\|_2,
\end{aligned}$$

$$\begin{aligned}
130 \quad \text{comp}(\mathcal{A}(\mathcal{B}_\rho), \varepsilon) &= \text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon) = \min\{n : \lambda_{n+1} \leq \varepsilon/\rho\} \\
131 \quad &= \min \left\{ n : \frac{1}{\lfloor (n+1)/2 \rfloor^r} \leq \frac{\varepsilon}{\rho} \right\} = 2 \left\lceil \left(\frac{\rho}{\varepsilon} \right)^{1/r} \right\rceil - 1. \\
132 \\
133
\end{aligned}$$

134 Here, r is positive. If r is an integer, then \mathcal{F} consists of functions that have absolutely
 135 continuous, periodic derivatives of up to order $r-1$. A larger r implies a stronger
 136 \mathcal{F} -norm, a more exclusive \mathcal{B}_ρ , and a smaller $\text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon)$.

137 Our goal is to construct algorithms in $\mathcal{A}(\mathcal{H})$ for some \mathcal{H} and also to deter-
 138 mine whether the computational cost of these algorithms is reasonable. We define
 139 $\text{cost}(A, \mathcal{H}, \varepsilon, \rho)$ to be *essentially no worse* than $\text{cost}(A^*, \mathcal{H}^*, \varepsilon, \rho)$ if there exists
 140 $\omega > 0$ such that

$$\text{cost}(A, \mathcal{H}, \varepsilon, \rho) \leq \text{cost}(A^*, \mathcal{H}^*, \omega\varepsilon, \rho) \quad \forall \varepsilon, \rho > 0. \quad (6)$$

We extend this definition analogously if $\text{cost}(A, \mathcal{H}, \varepsilon, \rho)$ is replaced by $\text{cost}(A, \mathcal{H}, \varepsilon)$ and/or $\text{cost}(A^*, \mathcal{H}^*, \omega\varepsilon, \rho)$ is replaced by $\text{cost}(A^*, \mathcal{H}^*, \omega\varepsilon)$. If these inequalities are not satisfied, we say that the cost of A is *essentially worse* than the cost of A^* . If the costs of two algorithms are essentially no worse than each other, then we call their costs essentially the same. An algorithm whose cost is essentially no worse than the best possible algorithm, is called *essentially optimal*.

Our condition for essentially no worse cost in (6) is not the same as

$$\text{cost}(A, \mathcal{H}, \varepsilon, \rho) \leq \omega \text{cost}(A^*, \mathcal{H}^*, \varepsilon, \rho) \quad \forall \varepsilon, \rho > 0. \quad (7)$$

If the cost grows polynomially in ε^{-1} , then conditions (6) and (7) are basically the same. If for some positive p and p^* ,

$$\text{cost}(A, \mathcal{H}, \varepsilon, \rho) \leq C(1 + \varepsilon^{-p}\rho^p) \quad \text{and} \quad C^*(1 + \varepsilon^{-p^*}\rho^{p^*}) \leq \text{cost}(A^*, \mathcal{H}^*, \varepsilon, \rho),$$

then $\text{cost}(A, \mathcal{H}, \varepsilon, \rho)$ is essentially no worse than $\text{cost}(A^*, \mathcal{H}^*, \varepsilon, \rho)$ iff $p \leq p^*$ under either condition (6) or (7). However, if the cost grows only logarithmically in ε^{-p} , then condition (6) makes more sense than (7). Specifically, if

$$\text{cost}(A, \mathcal{H}, \varepsilon, \rho) \leq C + \log(1 + \varepsilon^{-p}\rho^p) \quad \text{and} \quad C^* + \log(1 + \varepsilon^{-p^*}\rho^{p^*}) \leq \text{cost}(A^*, \mathcal{H}^*, \varepsilon, \rho),$$

then condition (6) requires $p \leq p^*$ for $\text{cost}(A, \mathcal{H}, \varepsilon, \rho)$ to be essentially no worse than $\text{cost}(A^*, \mathcal{H}^*, \varepsilon, \rho)$, whereas (7) allows this to be true even for $p > p^*$. We grant that if the cost grows faster than polynomially in ε^{-1} , then (7) may be the preferred condition.

To illustrate the comparison of costs, consider a non-increasing sequence of positive numbers, $\lambda^* = (\lambda_1^*, \lambda_2^*, \dots)$, which converges to 0, where $\lambda_i^* \geq \lambda_i$ for all $i \in \mathbb{N}$. Also consider an unbounded, strictly increasing sequence of non-negative integers $\mathbf{n} = (n_0, n_1, \dots)$. Define an algorithm A^* analogously to \hat{A} defined in (5):

$$A^*(f, \varepsilon) = A_{n^*}(f), \quad \text{where } n^* = n_{j^*}, \quad j^* = \min\{j : \lambda_{n_j+1}^* \leq \varepsilon/\rho\}, \quad A^* \in \mathcal{A}(\mathcal{B}_\rho).$$

By definition, the cost of algorithm A^* is no smaller than that of \hat{A} . Algorithm A^* may or may not have essentially worse cost than \hat{A} depending on the choice of λ^* and \mathbf{n} . The table below shows some examples. Each different case of A^* is labeled as having a cost that is either essentially no worse or essentially worse than that of \hat{A} .

	$\lambda_i = \frac{C}{i^p}$	$\text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon) \geq \left(\frac{C\rho}{\varepsilon}\right)^{1/p} - 1$
		$\text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon) < \left(\frac{C\rho}{\varepsilon}\right)^{1/p}$
no worse	$\lambda_i^* = \frac{C^*}{i^p}, n_j = 2^j$	$\text{cost}(A^*, \mathcal{B}_\rho, \varepsilon) \leq 2 \left(\frac{C^*\rho}{\varepsilon}\right)^{1/p}$
worse	$\lambda_i^* = \frac{C^*}{i^q}, q < p, n_j = j$	$\text{cost}(A^*, \mathcal{B}_\rho, \varepsilon) \geq \left(\frac{C^*\rho}{\varepsilon}\right)^{1/q} - 1$
	$\lambda_i = \frac{C}{i^p}, p > 1$	$\text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon) \geq \frac{\log(C\rho/\varepsilon)}{\log(p)} - 1$
		$\text{cost}(\widehat{A}, \mathcal{B}_\rho, \varepsilon) < \frac{\log(C\rho/\varepsilon)}{\log(p)}$
no worse	$\lambda_i^* = \frac{C^*}{i^p}, n_j = 2^j$	$\text{cost}(A^*, \mathcal{B}_\rho, \varepsilon) < \frac{\log(C^*\rho/\varepsilon)}{\log(p)} + 1$
worse	$\lambda_i^* = \frac{C^*}{i^p}, n_j = 2^j$	$\text{cost}(A^*, \mathcal{B}_\rho, \varepsilon) > 1.999 \frac{\log(C^*\rho/\varepsilon)}{\log(p)}$ for some ε
worse	$\lambda_i^* = \frac{C^*}{i^q}, q < p, n_j = j$	$\text{cost}(A^*, \mathcal{B}_\rho, \varepsilon) \geq \frac{\log(C^*\rho/\varepsilon)}{\log(q)} - 1$

1.4 The Case for Adaptive Algorithms

For bounded sets of input functions, such as balls, non-adaptive algorithms like \widehat{A} make sense. However, an a priori upper bound on $\|f\|_{\mathcal{F}}$ is typically unavailable in practice, so it is unknown which \mathcal{B}_ρ contain the input function f . Our aim is to consider unbounded sets of f for which the error of the interpolatory algorithm $A_n(f)$, defined in (3), can be bounded without an a priori upper bound on $\|f\|_{\mathcal{F}}$.

Popular adaptive algorithms encountered typically employ heuristic error bounds. While any algorithm can be fooled, we would like to have precise necessary conditions for being fooled, or equivalently, sufficient conditions for the algorithm to succeed. Our adaptive algorithm has such conditions and follows in the vein of adaptive algorithms developed in [2–6].

Our rigorous, data-driven error bound assumes that the series coefficients of the input function, f , decay steadily—but not necessarily monotonically. The cone of nice input functions, \mathcal{C} , is defined in Sect. 2. For such inputs, we construct an adaptive algorithm, $\tilde{A} \in \mathcal{A}(\mathcal{C})$, in Sect. 3, where $\tilde{A}(f, \varepsilon) = A_{\tilde{n}}(f)$ for some \tilde{n} depending on the input data and the definition of \mathcal{C} . The number of series coefficients sampled, \tilde{n} , is adaptively determined so that $\tilde{A}(f, \varepsilon)$ satisfies the error condition in (2). The computational cost of \tilde{A} is given in Theorem 1. Section 4 shows that our new algorithm is essentially optimal (see Theorem 3). Section 5 provides a numerical example. We end with concluding remarks.

2 Assuming a Steady Decay of the Series Coefficients of the Solution

Recall from (4) that the error of the fixed sample size interpolatory algorithm A_n is $\|S(f) - A_n(f)\|_{\mathcal{G}} = \|(\lambda_i \widehat{f}_i)_{i=n+1}^\infty\|_2$. The error depends on the series coefficients

not yet observed, so at first it seems impossible to bound the error in terms of observed series coefficients.

However, we can observe a finite number of partial sums,

$$\sigma_j(f) := \left\| (\lambda_i \widehat{f}_i)_{i=n_{j-1}+1}^{n_j} \right\|_2, \quad j \in \mathbb{N}, \quad (8)$$

where $\mathbf{n} = (n_0, n_1, \dots)$ is an unbounded, strictly increasing sequence of non-negative integers. We define the cone of nice input functions to consist of those functions for which the $\sigma_j(f)$ decay at a given rate with respect to one another:

$$\begin{aligned} \mathcal{C} = \mathcal{C}_{\mathbf{n},a,b} = & \{f \in \mathcal{F} : \sigma_{j+r}(f) \leq ab^r \sigma_j(f) \quad \forall j, r \in \mathbb{N}\} \\ & = \left\{f \in \mathcal{F} : \sigma_j(f) \leq \min_{1 \leq r < j} ab^r \sigma_{j-r}(f) \quad \forall j \in \mathbb{N}\right\}. \end{aligned} \quad (9)$$

Here, a and b are positive numbers that define the inclusivity of the cone \mathcal{C} and satisfy

$$b < 1 < a.$$

The constant a is an inflation factor, and the constant b defines the general rate of decay of the $\sigma_j(f)$ for $f \in \mathcal{C}$. Because ab^r may be greater than one, we do not require the series coefficients of the solution, $S(f)$, to decay monotonically. However, we expect their partial sums to decay steadily.

From the expression for the error in (4) and the definition of the cone in (9), one can now derive a data-driven error bound for $j \in \mathbb{N}$:

$$\begin{aligned} \|S(f) - A_{n_j}(f)\|_{\mathcal{G}} &= \left\| (\lambda_i \widehat{f}_i)_{i=n_j+1}^{\infty} \right\|_2 = \left\{ \sum_{r=1}^{\infty} \sum_{i=n_{j+r-1}+1}^{n_{j+r}} |\lambda_i \widehat{f}_i|^2 \right\}^{1/2} \\ &= \|(\sigma_{j+r}(f))_{r=1}^{\infty}\|_2 \\ &\leq \|(ab^r \sigma_j(f))_{r=1}^{\infty}\|_2 = \frac{ab}{\sqrt{1-b^2}} \sigma_j(f). \end{aligned} \quad (10)$$

This upper bound depends only on the function data and the parameters defining \mathcal{C} . The error vanishes as $j \rightarrow \infty$ because $\sigma_j(f) \leq ab^{j-1} \sigma_1(f) \rightarrow 0$ as $j \rightarrow \infty$. Moreover, the error of $A_{n_j}(f)$ is asymptotically no worse than $\sigma_j(f)$. Our adaptive algorithm in Sect. 3 increases j until the right hand side is smaller than the error tolerance.

Consider the choice $n_j = 2^j n_0$, where the number of terms in the sums, $\sigma_j(f)$, are doubled at each step. If the series coefficients of the solution decay like $|\lambda_i \widehat{f}_i| = \Theta(i^{-p})$ for some $p > 1/2$, then it is reasonable to expect that the $\sigma_j(f)$ are bounded above and below as

$$C_{\text{lo}}(n_0 2^j)^{1/2-p} \leq \sigma_j(f) \leq C_{\text{up}}(n_0 2^j)^{1/2-p}, \quad j \in \mathbb{N}, \quad (11)$$

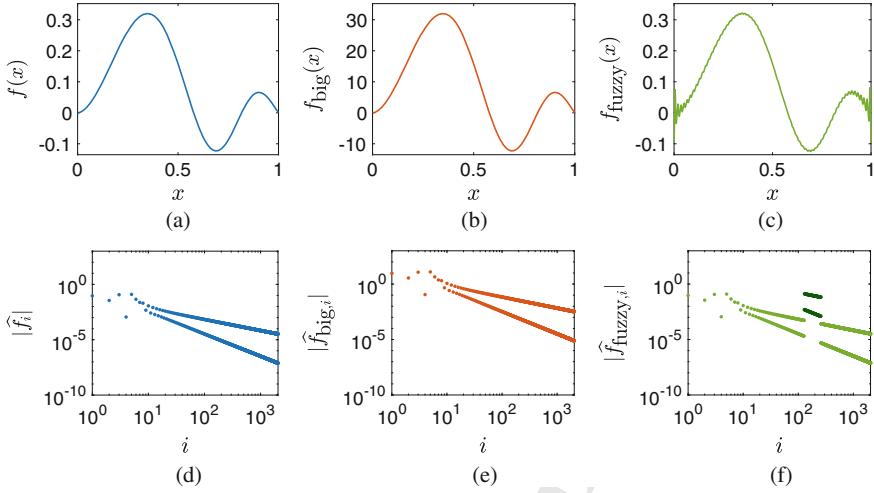


Fig. 1 **a** The function $f(x) = e^{-3x} \sin(3\pi x^2)$; **b** The function $f_{\text{big}} = 100f$; **c** The function f_{fuzzy} which results from modifying some of f coefficients, in dark green; **d** The first Fourier coefficients of f ; **e** The first Fourier coefficients of f_{big} ; **f** The first Fourier coefficients of f_{fuzzy}

for some constants C_{lo} and C_{up} , unless the series coefficients drop precipitously in magnitude for some $n_{j-1} < i \leq n_j$, and then jump back up for larger i . When (11) holds, it follows that

$$\frac{\sigma_{j+r}(f)}{\sigma_j(f)} \leq \frac{C_{\text{up}}(n_0 2^{j+r})^{1/2-p}}{C_{\text{lo}}(n_0 2^j)^{1/2-p}} = \frac{C_{\text{up}} 2^{r(1/2-p)}}{C_{\text{lo}}} \quad j \in \mathbb{N}.$$

Thus, choosing $a \geq C_{\text{up}}/C_{\text{lo}}$ and $b \geq 2^{1/2-p}$ ensures that reasonable inputs f lie inside the cone \mathcal{C} .

Although the definition of the cone in (9) constrains the decay rate of $\lambda_i |\widehat{f}_i|$ it is a rather weak constraint. Specifying a and b only implies a lower bound—but not an upper bound—on the p for which $\lambda_i |\widehat{f}_i|$ can be $\Theta(i^{-p})$.

Figure 1 shows three functions and their coefficients for the case of Example 1 when $r = 1$ and $n_j = 2^j$. These functions are $f(x) = e^{-3x} \sin(3\pi x^2)$, $f_{\text{big}} = 100f$ and f_{fuzzy} . The function f_{fuzzy} is obtained by taking $\sigma_8(f_{\text{fuzzy}}) = 250\sigma_8(f)$ and $\sigma_j(f_{\text{fuzzy}}) = \sigma_j(f)$ for $j \neq 8$. Both f and f_{big} lie in the same cones, $\mathcal{C}_{n,a,b}$, for all n , a , and b , and both appear similarly nice to the eye. Therefore, we expect adaptive algorithms that are successful for f to also be successful for f_{big} . However, f_{big} does not lie in some of the balls, \mathcal{B}_ρ , that f lies in. On the other hand, the high frequency noise in f_{fuzzy} suggests that only more robust and costly adaptive algorithms would succeed for such an input. This corresponds to the fact that f_{fuzzy} does not lie in all cones, $\mathcal{C}_{n,a,b}$, that f lies in.

3 Adaptive Algorithm

Now we introduce our adaptive algorithm, $\tilde{A} \in \mathcal{A}(\mathcal{C})$, which yields an approximate solution to the problem $S : \mathcal{C} \rightarrow \mathcal{G}$ that meets the absolute error tolerance ε .

Algorithm 1 Given a, b , the sequence \mathbf{n} , the cone \mathcal{C} , the input function $f \in \mathcal{C}$, and the absolute error tolerance ε , set $j = 1$.

Step 1. Compute $\sigma_j(f)$ as defined in (8).

Step 2. Check whether j is large enough to satisfy the error tolerance, i.e.,

$$\sigma_j(f) \leq \frac{\varepsilon \sqrt{1-b^2}}{ab}.$$

If this is true, then return $\tilde{A}(f, \varepsilon) = A_{n_j}(f)$, where A_n is defined in (3), and terminate the algorithm.

Step 3. Otherwise, increase j by 1 and return to Step 1.

Theorem 1 The algorithm, \tilde{A} , defined in Algorithm 1 lies in $\mathcal{A}(\mathcal{C})$ and has computational cost $\text{cost}(\tilde{A}, f, \varepsilon) = n_{j^*}$, where j^* is defined implicitly by the inequalities

$$j^* = \min \left\{ j \in \mathbb{N} : \sigma_j(f) \leq \frac{\varepsilon \sqrt{1-b^2}}{ab} \right\}. \quad (12)$$

Moreover, $\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho) = n_{j^\dagger}$, where j^\dagger satisfies the following upper bound:

$$j^\dagger \leq \min \left\{ j \in \mathbb{N} : F(j) \geq \frac{\rho^2 a^4}{\varepsilon^2 (1-b^2)} \right\}, \quad (13)$$

and F is the strictly increasing function defined as

$$F(j) := \sum_{k=0}^{j-1} \frac{b^{2(k-j)}}{\lambda_{n_k}^2}, \quad j \in \mathbb{N}. \quad (14)$$

Proof This algorithm terminates for some $j = j^*$ because $\sigma_j(f) \leq ab^{j-1} \sigma_1(f) \rightarrow 0$ as $j \rightarrow \infty$. The value of j^* follows directly from this termination criterion in Step 2. It then follows that the error bound on $A_{n_{j^*}}(f)$ in (10) is no greater than the error tolerance ε . So, $\tilde{A} \in \mathcal{A}(\mathcal{C})$.

For the remainder of the proof consider ρ and ε to be fixed. To derive an upper bound on $n_{j^\dagger} = \text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho)$ we first note some properties of $\sigma_j(f)$ for all $f \in \mathcal{C}$:

$$\begin{aligned} \lambda_{n_j} \|(\hat{f}_i)_{i=n_{j-1}+1}^{n_j}\|_2 &\leq \|(\lambda_i \hat{f}_i)_{i=n_{j-1}+1}^{n_j}\|_2 = \sigma_j(f) \\ &\leq \lambda_{n_{j-1}+1} \|(\hat{f}_i)_{i=n_{j-1}+1}^{n_j}\|_2. \end{aligned} \quad (15)$$

A rough upper bound on j^\dagger may be obtained by noting that for any $f \in \mathcal{C} \cap \mathcal{B}_\rho$ and for any $j < j^* \leq j^\dagger$, it follows from (12) and (15) that

$$\rho \geq \|f\|_{\mathcal{F}} \geq \|(\widehat{f}_i)_{i=n_{j-1}+1}^{n_j}\|_2 \geq \frac{\sigma_j(f)}{\lambda_{n_{j-1}+1}} > \frac{\varepsilon\sqrt{1-b^2}}{ab\lambda_{n_{j-1}+1}} \geq \frac{\varepsilon\sqrt{1-b^2}}{ab\lambda_{n_{j-1}}}$$

Thus, one upper bound on j^\dagger is the smallest j violating the above inequality:

$$j^\dagger \leq \min \left\{ j \in \mathbb{N} : \lambda_{n_{j-1}}^{-1} \geq \frac{\rho ab}{\varepsilon\sqrt{1-b^2}} \right\}. \quad (16)$$

The tighter upper bound in Theorem 1 may be obtained by a more careful argument in a similar vein. For any $f \in \mathcal{C} \cap \mathcal{B}_\rho$ satisfying $n_{j^*} = \text{cost}(\tilde{A}, f, \varepsilon) = \text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho) = n_{j^*}$ and for any $j < j^* = j^\dagger$,

$$\begin{aligned} \rho^2 &\geq \|f\|_{\mathcal{F}}^2 = \|(\widehat{f}_i)_{i=1}^\infty\|_2^2 \geq \sum_{k=1}^j \|(\widehat{f}_i)_{i=n_{k-1}+1}^{n_k}\|_2^2 \\ &\geq \sum_{k=1}^j \frac{\sigma_k^2(f)}{\lambda_{n_{k-1}+1}^2} \quad \text{by (15)} \\ &\geq \sum_{k=1}^{j-1} \frac{b^{2(k-j)}\sigma_j^2(f)}{a^2\lambda_{n_{k-1}+1}^2} + \frac{\sigma_j^2(f)}{\lambda_{n_{j-1}+1}^2} \quad \text{by (9)} \\ &= \sigma_j^2(f) \left[\sum_{k=1}^{j-1} \frac{b^{2(k-j)}}{a^2\lambda_{n_{k-1}+1}^2} + \frac{1}{\lambda_{n_{j-1}+1}^2} \right] \\ &> \frac{\sigma_j^2(f)}{a^2} \sum_{k=1}^j \frac{b^{2(k-j)}}{\lambda_{n_{k-1}+1}^2} \quad \text{since } a > 1 \\ &= \frac{\sigma_j^2(f)}{a^2} \sum_{k=0}^{j-1} \frac{b^{2(k+1-j)}}{\lambda_{n_k+1}^2} \\ &\geq \frac{b^2\sigma_j^2(f)}{a^2} \sum_{k=0}^{j-1} \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \quad \text{since } \lambda_{n_k} \geq \lambda_{n_k+1} \\ &= \frac{b^2\sigma_j^2(f)}{a^2} F(j) \quad \text{by (14).} \end{aligned}$$

So, for any $j < j^* = j^\dagger$ it follows from the termination criterion in (12) that

$$F(j) < \frac{\rho^2 a^4}{\varepsilon^2(1-b^2)}.$$

Note from (14) that F is an increasing function because as j increases, the sum defining F includes more positive terms and $b^{2(k-j)}$ also increases. Thus, any j that violates the above inequality, must satisfy $j \geq j^\dagger$. This establishes (13). \square

We note in passing that for our adaptive algorithm,

$$\min\{\text{cost}(\tilde{A}, f, \varepsilon) : f \in \mathcal{C} \setminus \mathcal{B}_\rho\} \begin{cases} = n_1, & n_0 > 0, \\ \leq n_2, & n_0 = 0, \end{cases} \quad \forall \rho > 0, \varepsilon > 0.$$

This result may be obtained by considering functions where only \hat{f}_1 is nonzero. For $n_0 > 0$, $\sigma_1(f) = 0$, and for $n_0 = 0$, $\sigma_2(f) = 0$.

The upper bound on $\text{cost}(\tilde{A}, \mathcal{C}, \rho, \varepsilon)$ in Theorem 1 is a non-decreasing function of ρ/ε , which depends on the behavior of the sequence $(\lambda_{n_j})_{j=0}^\infty$. This in turn depends both on the increasing sequence \mathbf{n} and on the non-increasing sequence $(\lambda_i)_{i=1}^\infty$. Considering the definition of F , one can imagine that in some cases the first term in the sum dominates, while in other cases the last term in the sum dominates, all depending on how b^{k-j}/λ_{n_k} behaves with k and j . These simplifications lead to two simpler, but coarser upper bounds on the cost of \tilde{A} .

Corollary 1 *For the algorithm, \tilde{A} , defined in Algorithm 1, we have $\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho) \leq n_{j^\dagger}$, where j^\dagger satisfies the following upper bound:*

$$j^\dagger \leq \left\lceil \log \left(\frac{\rho a^2 \lambda_{n_0}}{\varepsilon \sqrt{1-b^2}} \right) / \log \left(\frac{1}{b} \right) \right\rceil. \quad (17)$$

Moreover, if the $\lambda_{n_{j-1}}$ decay as quickly as

$$\lambda_{n_{j-1}} \leq \alpha \beta^j, \quad j \in \mathbb{N}, \quad \text{for some } \alpha > 0, 0 < \beta < 1. \quad (18)$$

then j^\dagger also satisfies the following upper bound:

$$j^\dagger \leq \left\lceil \log \left(\frac{\rho a^2 \alpha b}{\varepsilon \sqrt{1-b^2}} \right) / \log \left(\frac{1}{\beta} \right) \right\rceil. \quad (19)$$

Proof Ignoring all but the first term in the definition of F in (14) implies that

$$\begin{aligned} j^\dagger &\leq \min \left\{ j \in \mathbb{N} : b^{-2j} \geq \frac{\rho^2 a^4 \lambda_{n_0}^2}{\varepsilon^2 (1-b^2)} \right\} \\ &= \min \left\{ j \in \mathbb{N} : j \geq \log \left(\frac{\rho a^2 \lambda_{n_0}}{\varepsilon (\sqrt{1-b^2})} \right) / \log \left(\frac{1}{b} \right) \right\}. \end{aligned}$$

This implies (17).

Ignoring all but the last term of the sum leads to the simpler upper bound similar to (16):

$$j^\dagger \leq \min \left\{ j \in \mathbb{N} : \lambda_{n_{j-1}}^{-1} \geq \frac{\rho a^2 b}{\varepsilon \sqrt{1-b^2}} \right\}.$$

If the $\lambda_{n_{j-1}}$ decay as assumed in (18) then

$$j^\dagger \leq \min \left\{ j \in \mathbb{N} : \alpha \beta^{-j} \leq \frac{\rho a^2 b}{\varepsilon \sqrt{1-b^2}} \right\},$$

which implies (19). \square

This corollary highlights two limiting factors on the computational cost of our adaptive algorithm, \tilde{A} . When j is large enough to make $\lambda_{n_{j-1}} \|f\|_{\mathcal{F}} / \varepsilon$ small enough, $\tilde{A}(f, \varepsilon)$ stops. This is statement (19) and its precursor, (16). Alternatively, the assumption that the $\sigma_j(f)$ are steadily decreasing, as specified in the definition of \mathcal{C} in (9), means that $\tilde{A}(f, \varepsilon)$ also must stop by the time j becomes large enough with respect to $\lambda_{n_0} \|f\|_{\mathcal{F}} / \varepsilon$.

Assumption (18) is not very restrictive. It holds if the λ_i decay algebraically and the n_j increase geometrically. It also holds if the λ_i decay geometrically and the n_j increase arithmetically.

The adaptive algorithm \tilde{A} , which does not know an upper bound on $\|f\|_{\mathcal{F}}$ a priori, may cost more than the non-adaptive algorithm \hat{A} , which assumes an upper bound on $\|f\|_{\mathcal{F}}$, but under reasonable assumptions, the extra cost is small.

Corollary 2 Suppose that the sequence \mathbf{n} is chosen to satisfy

$$\lambda_{n_{j+1}} \geq c_\lambda \lambda_{n_j}, \quad j \in \mathbb{N}, \quad (20)$$

for some positive c_λ . Then $\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho)$ is essentially no worse than $\text{cost}(\hat{A}, \mathcal{B}_\rho, \varepsilon)$ in the sense of (6).

Proof Combining the upper bound on $n_{j^\dagger} = \text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho)$ in (16) plus (20) above, it follows that

$$\lambda_{n_{j^\dagger}} \geq c_\lambda^2 \lambda_{n_{j^\dagger-2}} > \frac{\varepsilon c_\lambda^2 \sqrt{1-b^2}}{\rho a b} = \frac{\omega \varepsilon}{\rho} \geq \lambda_{\hat{n}+1}, \quad \omega := \frac{c_\lambda^2 \sqrt{1-b^2}}{a b},$$

where $\hat{n} = \text{cost}(\hat{A}, \mathcal{B}_\rho, \omega \varepsilon)$. Since the λ_i are non-increasing and $\lambda_{n_{j^\dagger}} > \lambda_{\hat{n}+1}$, it follows that $n_{j^\dagger} < \hat{n} + 1$, and so $n_{j^\dagger} \leq \hat{n}$. Thus,

$$\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho) = n_{j^\dagger} \leq \hat{n} = \text{cost}(\hat{A}, \mathcal{B}_\rho, \omega \varepsilon).$$

\square

4 Essential Optimality of the Adaptive Algorithm

From Corollary 2 it is known that $\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho)$ is essentially no worse than $\text{cost}(\tilde{A}, \mathcal{B}_\rho, \varepsilon) = \text{comp}(\mathcal{A}(\mathcal{B}_\rho), \varepsilon)$. We would like to show that $\tilde{A} \in \mathcal{A}(\mathcal{C})$ is essentially optimal, i.e., $\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho)$ is essentially no worse than $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho)$. However, $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho)$ may be smaller than $\text{comp}(\mathcal{A}(\mathcal{B}_\rho), \varepsilon)$ because $\mathcal{C} \cap \mathcal{B}_\rho$ is a strict subset of \mathcal{B}_ρ . This presents a challenge.

A lower bound on $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho)$ is established by constructing fooling functions in \mathcal{C} with norms no greater than ρ . To obtain a result that can be compared with the cost of our algorithm, we assume that

$$R = \sup_{j \in \mathbb{N}} \frac{\lambda_{n_{j-1}}}{\lambda_{n_j}} < \infty. \quad (21)$$

That is, the subsequence $\lambda_{n_1}, \lambda_{n_2}, \dots$ cannot decay too quickly.

The following theorem establishes a lower bound on the complexity of our problem for input functions in \mathcal{C} . The theorem after that shows that the cost of our algorithm as given in Theorem 1 is essentially no worse than this lower bound.

Theorem 2 *Under assumption (21), a lower bound on the complexity of the linear problem defined in (1) is*

$$\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho) \geq n_{j^\dagger} - 1,$$

where

$$j^\dagger = \min \left\{ j \in \mathbb{N} : F(j+2) \geq \frac{\rho^2}{\varepsilon^2 b^2} \left[\frac{(a+1)^2 R^2}{(a-1)^2} + 1 \right]^{-1} \right\},$$

and F is the function defined in (14).

Proof Consider a fixed ρ and ε . Choose any positive integer j such that $n_j \geq \text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho) + 2$. The proof proceeds by carefully constructing three test input functions, f and f_\pm , lying in $\mathcal{C} \cap \mathcal{B}_\rho$, which yield the same approximate solution but different true solutions. This leads to a lower bound on n_j , which can be translated into a lower bound on $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho)$.

The first test function $f \in \mathcal{C}$ is defined in terms of its series coefficients as follows:

$$\hat{f}_i := \begin{cases} \frac{cb^{k-j}}{\lambda_{n_k}}, & i = n_k, k = 1, \dots, j, \\ 0, & \text{otherwise,} \end{cases}$$

$$c^2 := \rho^2 \left[\left(1 + \frac{(a-1)^2}{(a+1)^2 R^2} \right) \sum_{k=0}^j \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \right]^{-1}.$$

It can be verified that the test function lies both in \mathcal{B}_ρ and in \mathcal{C} :

$$\|f\|_{\mathcal{F}}^2 = c^2 \sum_{k=1}^j \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \leq \rho^2,$$

$$\sigma_k(f) = \begin{cases} cb^{k-j}, & k = 1, \dots, j, \\ 0, & \text{otherwise,} \end{cases}$$

$$\sigma_{k+r}(f) = \begin{cases} b^r \sigma_k(f) \leq ab^r \sigma_k(f), & k+r \leq j, r \geq 1, \\ 0 \leq ab^r \sigma_k(f), & k+r > j, r \geq 1. \end{cases}$$

Now suppose that $A^* \in \mathcal{A}(\mathcal{C})$ is an optimal algorithm, i.e., $\text{cost}(A^*, \mathcal{C}, \varepsilon, \rho) = \text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho)$ for all $\varepsilon, \rho > 0$. For our particular input f defined above, suppose that $A^*(f, \varepsilon)$ samples $L_1(f), \dots, L_n(f)$ where

$$n+2 \leq \text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho) + 2 \leq n_j.$$

Let u be a linear combination of u_1, \dots, u_{n_j} , expressed as

$$u = \sum_{k=0}^j \frac{b^{k-j} u^{(k)}}{\lambda_{n_k}},$$

where $u^{(0)}$ is a linear combination of u_1, \dots, u_{n_0} , and each $u^{(k)}$ is a linear combination of $u_{n_{k-1}+1}, \dots, u_{n_k}$, for $k = 1, \dots, j$. We constrain u to satisfy:

$$L_1(u) = \dots = L_n(u) = 0, \quad \langle u, f \rangle_{\mathcal{F}} = 0, \quad \max_{0 \leq k \leq j} \|u^{(k)}\|_{\mathcal{F}} = 1.$$

Since u is a linear combination of $n_j \geq n+2$ basis functions, these $n+2$ constraints can be satisfied.

Let the other two test functions be constructed in terms of u as

$$f_{\pm} := f \pm \eta u, \quad \eta := \frac{(a-1)c}{(a+1)R}, \quad (22)$$

$$\begin{aligned} \|f_{\pm}\|_{\mathcal{F}}^2 &\leq \|f\|_{\mathcal{F}}^2 + \|\eta u\|_{\mathcal{F}}^2 \\ &\leq \sum_{k=1}^j \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \left(c^2 + \eta^2 \|u^{(k)}\|_{\mathcal{F}}^2 \right) + \eta^2 \|u^{(0)}\|_{\mathcal{F}}^2 \frac{b^{-2j}}{\lambda_{n_0}^2} \\ &\leq (c^2 + \eta^2) \sum_{k=0}^j \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \\ &= \left(1 + \frac{(a-1)^2}{(a+1)^2 R^2} \right) c^2 \sum_{k=0}^j \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \\ &\leq \rho^2 \quad \text{by the definition of } c \text{ above.} \end{aligned}$$

So, $f_{\pm} \in \mathcal{B}_{\rho}$. By design, $A^*(f_{\pm}, \varepsilon) = A^*(f, \varepsilon)$, which will be used below.

Now, we must check that $f_{\pm} \in \mathcal{C}$. From the definition in (8) it follows that for $k = 1, \dots, j$ and $r \geq 1$,

$$\sigma_k(f_{\pm}) \begin{cases} \leq \sigma_k(f) + \sigma_k(\eta u) \leq cb^{k-j} + \eta \lambda_{n_{k-1}+1} \frac{b^{k-j} \|u^{(k)}\|_{\mathcal{F}}}{\lambda_{n_k}} \leq b^{k-j} (c + \eta R) \\ \geq \sigma_k(f) - \sigma_k(\eta u) \geq cb^{k-j} - \eta \lambda_{n_{k-1}+1} \frac{b^{k-j} \|u^{(k)}\|_{\mathcal{F}}}{\lambda_{n_k}} \geq b^{k-j} (c - \eta R), \end{cases}$$

Therefore,

$$\sigma_{k+r}(f_{\pm}) \leq b^{k+r-j} (c + \eta R) = ab^r b^{k-j} \frac{2c}{a+1} = ab^r b^{k-j} (c - \eta R) \leq ab^r \sigma_k(f_{\pm}),$$

which establishes that $f_{\pm} \in \mathcal{C}$.

Although two test functions f_{\pm} yield the same approximate solution, they have different true solutions. In particular,

$$\begin{aligned} \varepsilon &\geq \max \{ \|S(f_+) - A^*(f_+, \varepsilon)\|_{\mathcal{G}}, \|S(f_-) - A^*(f_-, \varepsilon)\|_{\mathcal{G}} \} \\ &\geq \frac{1}{2} [\|S(f_+) - A^*(f, \varepsilon)\|_{\mathcal{G}} + \|S(f_-) - A^*(f, \varepsilon)\|_{\mathcal{G}}] \\ &\quad \text{since } A^*(f_{\pm}, \varepsilon) = A^*(f, \varepsilon) \\ &\geq \frac{1}{2} \|S(f_+) - S(f_-)\|_{\mathcal{G}} \quad \text{by the triangle inequality} \\ &\geq \frac{1}{2} \|S(f_+ - f_-)\|_{\mathcal{G}} \quad \text{since } S \text{ is linear} \\ &= \eta \|S(u)\|_{\mathcal{G}}. \end{aligned}$$

Thus, we have

$$\begin{aligned} \varepsilon^2 &\geq \eta^2 \|S(u)\|_{\mathcal{G}}^2 = \eta^2 \sum_{k=0}^j \|S(u^{(k)})\|_{\mathcal{G}}^2 \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \\ &\geq \eta^2 \sum_{k=0}^j \|u^{(k)}\|_{\mathcal{F}}^2 b^{2(k-j)} \quad \text{since } \|S(u^{(k)})\|_{\mathcal{G}} \geq \lambda_{n_k} \|u^{(k)}\|_{\mathcal{F}} \\ &\geq \eta^2 b^{2(k^*-j)} \quad \text{where } k^* = \operatorname{argmax}_{0 \leq k \leq j} \|u^{(k)}\|_{\mathcal{F}} \\ &\geq \eta^2 = \frac{(a-1)^2 c^2}{(a+1)^2 R^2} \quad \text{since } b < 1 \\ &= \frac{(a-1)^2 \rho^2}{(a+1)^2 R^2} \left[\left(1 + \frac{(a-1)^2}{(a+1)^2 R^2} \right) \sum_{k=0}^j \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \right]^{-1} \quad \text{by (22)} \end{aligned}$$

$$\begin{aligned}
&= \rho^2 \left[\left(\frac{(a+1)^2 R^2}{(a-1)^2} + 1 \right) \sum_{k=0}^j \frac{b^{2(k-j)}}{\lambda_{n_k}^2} \right]^{-1} \\
&= \rho^2 \left[\left(\frac{(a+1)^2 R^2}{(a-1)^2} + 1 \right) b^2 F(j+1) \right]^{-1} \quad \text{by (14).}
\end{aligned}$$

This inequality is equivalent to

$$F(j+1) \geq \frac{\rho^2}{\varepsilon^2 b^2} \left[\frac{(a+1)^2 R^2}{(a-1)^2} + 1 \right]^{-1}.$$

This lower bound must be satisfied by j to be consistent with the assumption $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho) \leq n_j - 2$. Thus, any j violating this inequality satisfies $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho) \geq n_j - 1$. Since F is strictly increasing, the largest j violating this inequality must satisfy

$$F(j+1) < \frac{\rho^2}{\varepsilon^2 b^2} \left[\frac{(a+1)^2 R^2}{(a-1)^2} + 1 \right]^{-1} \leq F(j+2),$$

which is the definition of j^\ddagger above in the statement of this theorem. This proves the lower bound on $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho)$. \square

The next step is to show that the cost of our algorithm is essentially no worse than that of the optimal algorithm.

Theorem 3 Under assumption (21) $\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho)$ is essentially no worse than $\text{comp}(\mathcal{A}(\mathcal{C}), \varepsilon, \rho)$.

Proof We need an inequality relating $F(j)$, which appears in the definition of j^\dagger in Theorem 1 and $F(j+3)$, which is related to the definition of j^\ddagger in Theorem 2. According to (21) and the definition of F in (14), for any $\ell \in \mathbb{N}$,

$$R^{2\ell} F(j) \geq R^{2\ell} \frac{b^{-2}}{\lambda_{n_{j-1}}^2} \geq \frac{b^{-2}}{\lambda_{n_{j+\ell-1}}^2},$$

and so

$$\begin{aligned}
b^{-2\ell} [1 + b^2 R^2 \cdots + (b^2 R^2)^\ell] F(j) &\geq F(j) b^{-2\ell} + \frac{b^{-2\ell}}{\lambda_{n_j}^2} + \cdots + \frac{b^{-2}}{\lambda_{n_{j+\ell-1}}^2} \\
&= \sum_{j=0}^{j-1} \frac{b^{2(k-j-\ell)}}{\lambda_{n_k}^2} + \frac{b^{-2\ell}}{\lambda_{n_j}^2} + \cdots + \frac{b^{-2}}{\lambda_{n_{j+\ell-1}}^2} \\
&= F(j+\ell).
\end{aligned}$$

The $\ell = 3$ case of this inequality implies the following lower bound on $F(j)$ in terms of $F(j + 3)$:

$$F(j) \geq \frac{b^6 F(j + 3)}{1 + b^2 R^2 + b^4 R^4 + b^6 R^6} = \frac{\omega^2 a^4 b^2}{1 - b^2} \left[\frac{(a + 1)^2 R^2}{(a - 1)^2} + 1 \right] F(j + 3),$$

where

$$\omega = \sqrt{\frac{(1 - b^2)b^4}{a^4(1 + b^2 R^2 + b^4 R^4 + b^6 R^6)} \left[\frac{(a + 1)^2 R^2}{(a - 1)^2} + 1 \right]^{-1}}, \quad (23)$$

Note that ω does not depend on ρ or ε but only on the definition of \mathcal{C} .

For any positive ρ and ε , let $j^\dagger = j^\dagger(\varepsilon, \rho)$ be defined as in Theorem 1 and let $j^\ddagger = j^\ddagger(\varepsilon, \rho)$ be defined as in Theorem 2. Then by those Theorems, $j^\dagger(\varepsilon, \rho)$ can be bounded above in terms of $j^\ddagger(\omega\varepsilon, \rho)$ for ω defined in (23):

$$\begin{aligned} j^\dagger(\varepsilon, \rho) &\leq \min \left\{ j \in \mathbb{N} : F(j) \geq \frac{\rho^2 a^4}{\varepsilon^2(1 - b^2)} \right\} \\ &\leq \min \left\{ j \in \mathbb{N} : \frac{\omega^2 a^4 b^2}{1 - b^2} \left[\frac{(a + 1)^2 R^2}{(a - 1)^2} + 1 \right] F(j + 3) \geq \frac{\rho^2 a^4}{\varepsilon^2(1 - b^2)} \right\} \\ &= \min \left\{ j \in \mathbb{N} : F(j + 3) \geq \frac{\rho^2}{\omega^2 \varepsilon^2 b^2} \left[\frac{(a + 1)^2 R^2}{(a - 1)^2} + 1 \right]^{-1} \right\} \\ &= j^\ddagger(\omega\varepsilon, \rho) - 1. \end{aligned}$$

By the argument above, it follows that $j^\dagger(\varepsilon, \rho) \leq j^\ddagger(\omega\varepsilon, \rho) - 1$, and again by Theorems 1 and 2 it follows that

$$\text{cost}(\tilde{A}, \mathcal{C}, \varepsilon, \rho) = n_{j^\dagger(\varepsilon, \rho)} \leq n_{j^\ddagger(\omega\varepsilon, \rho) - 1} \leq n_{j^\ddagger(\omega\varepsilon, \rho)} - 1 \leq \text{comp}(\mathcal{A}(\mathcal{C}), \omega\varepsilon, \rho).$$

Therefore, our algorithm is essentially no more costly than the optimal algorithm. \square

5 Numerical Example

Consider the case of approximating the partial derivative with respect to x_1 of periodic functions defined on the d -dimensional unit cube:

$$f = \sum_{\mathbf{k} \in \mathbb{Z}^d} \hat{f}(\mathbf{k}) \hat{u}_{\mathbf{k}} = \sum_{i \in \mathbb{N}} \hat{f}_i u_i,$$

$$\begin{aligned}
\widehat{u}_k(\mathbf{x}) &:= \prod_{j=1}^d \frac{2^{(1-\delta_{k_j,0})/2} \cos(2\pi k_j x_j + \mathbb{1}_{(-\infty,0)}(k_j)\pi/2)}{[\max(1, \gamma_j k_j)]^4}, \\
S(f) &:= \frac{\partial f}{\partial x_1} = \sum_{\mathbf{k} \in \mathbb{Z}^d} \widehat{f}(\mathbf{k}) \lambda(\mathbf{k}) \widehat{v}_k(\mathbf{x}) = \sum_{i \in \mathbb{N}} \widehat{f}_i \lambda_i v_i, \\
\widehat{v}_k(\mathbf{x}) &:= -\text{sign}(k_1) \sin(2\pi k_1 x_1 + \mathbb{1}_{(-\infty,0)}(k_1)\pi/2) \\
&\quad \times \prod_{j=2}^d \cos(2\pi k_j x_j + \mathbb{1}_{(-\infty,0)}(k_j)\pi/2), \\
\lambda(\mathbf{k}) &:= 2\pi |k_1| \frac{\prod_{j=1}^d 2^{(1-\delta_{k_j,0})/2}}{\prod_{j=1}^d [\max(1, \gamma_j k_j)]^4}, \\
\boldsymbol{\gamma} &:= (1, 1/2, 1/4, \dots, 2^{-d+1}).
\end{aligned}$$

Note that $\lambda_1 \geq \lambda_2 \geq \dots$ is an ordering of the $\lambda(\mathbf{k})$. That ordering then determines the \widehat{f}_i , u_i , and v_i in terms of the $\widehat{f}(\mathbf{k})$, $\widehat{u}(\mathbf{k})$, and $\widehat{v}(\mathbf{k})$, respectively.

We construct a function by choosing its Fourier coefficients $\widehat{f}(\mathbf{k}) \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1)$ for $d = 3$, $\mathbf{k} \in \{-30, -29, \dots, 30\}^3$, and $\widehat{f}(\mathbf{k}) = 0$ otherwise. This corresponds to $61^3 \approx 2 \times 10^5$ nonzero Fourier coefficients. Let $a = 2$ and $b = 1/2$ and choose $\mathbf{n} = (0, 16, 32, 64, \dots)$. To compute $\sigma_j(f)$, $j \in \mathbb{N}$ by (8), we need to sort $(\lambda(\mathbf{k}))_{\mathbf{k} \in \mathbb{Z}^d}$ in descending order, $\lambda_1, \lambda_2, \dots$. Given ε , we can then find the number of series coefficients needed to satisfy the error criterion, i.e., n_{j^\dagger} where

$$j^\dagger = \min \left\{ j \in \mathbb{N} : \frac{ab\sigma_j(f)}{\sqrt{1-b^2}} \leq \varepsilon \right\}$$

Figure 2 shows the input function, the solution, the approximate solution, and the error of the approximate solution for $\varepsilon = 0.1$. For this example, $n_{j^\dagger} = 8192$ is sufficient to satisfy the error tolerance, as is clear from Fig. 2d. Figure 3 shows the sample size, n_{j^\dagger} needed for ten different error tolerances from 0.1 to 10. Because the possible sample sizes are powers of 2, some tolerances require the same sample size.

6 Discussion and Conclusion

Many practical adaptive algorithms lack theory, and many theoretically justified algorithms are non-adaptive. We have demonstrated for a general setting how to construct a theoretically justified, essentially optimal algorithm. The decay of the singular values determines the computational complexity of the problem and the computational cost of our algorithm.

The key idea of our algorithm is to derive an adaptive error bound by assuming the steady decay of the Fourier series coefficients of the solution. The set of such functions

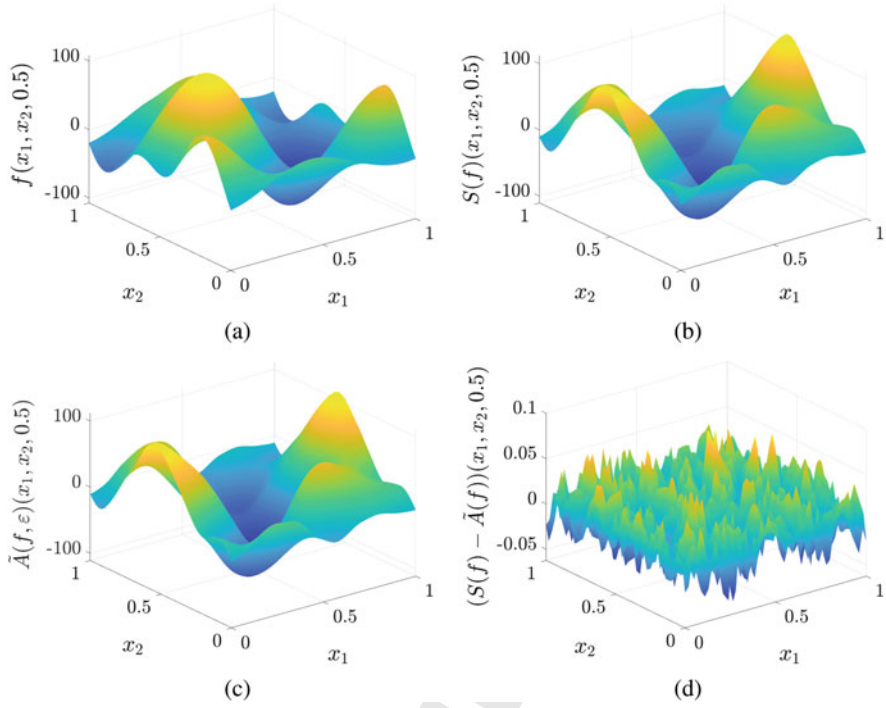


Fig. 2 For $\varepsilon = 0.1$: **a** The input function, f ; **b** The true first partial derivative of f ; **c** The approximate first partial derivative of f ; **d** The approximation error

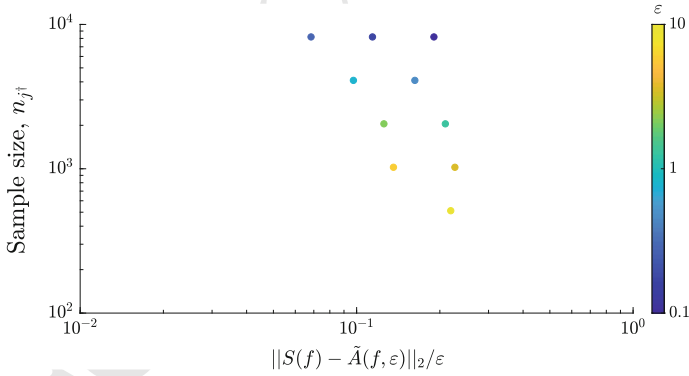


Fig. 3 Sample size n_{j^*} , error tolerance ε , and ratio of true error to error tolerance

constitutes a cone. We do not need to know the decay rate of these coefficients a priori. The cost of our algorithm also serves as a goal for an algorithm that uses function values, which are more commonly available than Fourier series coefficients. An important next step is to identify an essentially optimal algorithm based on function values. Another research direction is to extend this setting to Banach spaces of inputs and/or outputs.

Acknowledgements The authors would like to thank the organizers for a wonderful MCQMC 2018. We also thank the referees for their many helpful suggestions. This work is supported in part by National Science Foundation grants DMS-1522687 and DMS-1638521 (SAMSI). Yuhan Ding acknowledges the support of the Department of Mathematics, Misericordia University.


References

1. Kunsch, R.J., Novak, E., Rudolf, D.: Solvable integration problems and optimal sample size selection. *J. Complex.* **53**, 40–67 (2019)
2. Hickernell, F.J., Jiang, L., Liu, Y., Owen, A.B.: Guaranteed conservative fixed width confidence intervals via Monte Carlo sampling. In: Dick, J., Kuo, F.Y., Peters, G.W., Sloan, I.H. (eds.) *Monte Carlo and Quasi-Monte Carlo Methods 2012*. Springer Proceedings in Mathematics and Statistics, vol. 65, pp. 105–128, Springer, Berlin (2013). <https://doi.org/10.1007/978-3-642-41095-6>
3. Clancy, N., Ding, Y., Hamilton, C., Hickernell, F.J., Zhang, Y.: The cost of deterministic, adaptive, automatic algorithms: cones, not balls. *J. Complex.* **30**, 21–45 (2014). <https://doi.org/10.1016/j.jco.2013.09.002>
4. Hickernell, F.J., Jiménez Rugama, L.A.: Reliable adaptive cubature using digital sequences. In: Cools and Nuyens [7], pp. 367–383 (2016). [arXiv:1410.8615](https://arxiv.org/abs/1410.8615) [math.NA]
5. Jiménez Rugama, L.A., Hickernell, F.J.: Adaptive multidimensional integration based on rank-1 lattices. In: Cools and Nuyens [7], pp. 407–422 (2016). [arXiv:1411.1966](https://arxiv.org/abs/1411.1966)
6. Hickernell, F.J., Jiménez Rugama, L.A., Li, D.: Adaptive quasi-Monte Carlo methods for cubature. In: Dick, J., Kuo, F.Y., Woźniakowski, H. (eds.) *Contemporary Computational Mathematics—a Celebration of the 80th Birthday of Ian Sloan*, pp. 597–619. Springer, Berlin (2018). <https://doi.org/10.1007/978-3-319-72456-0>
7. Cools, R., Nuyens, D. (eds.): *Monte Carlo and Quasi-Monte Carlo methods: MCQMC*, Leuven, Belgium, April 2014. In: Springer Proceedings in Mathematics and Statistics, vol. 163. Springer, Berlin (2016)



Author Queries

Chapter 8

Query Refs.	Details Required	Author's response
AQ1	Please check and confirm if the author names and initials are correct	
AQ2	As keywords are mandatory for this chapter, please provide 3–6 keywords.	
AQ3	Reference [7] is given in list but not cited in text. Please cite in text or delete from list.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

<i>Instruction to printer</i>	<i>Textual mark</i>	<i>Marginal mark</i>
Leave unchanged	... under matter to remain	Ⓟ
Insert in text the matter indicated in the margin	⧵	New matter followed by ⧵ or ⧵ [Ⓢ]
Delete	/ through single character, rule or underline or ⎯ through all characters to be deleted	⧻ or ⧻ [Ⓢ]
Substitute character or substitute part of one or more word(s)	/ through letter or ⎯ through characters	new character / or new characters /
Change to italics	— under matter to be changed	↵
Change to capitals	≡ under matter to be changed	≡
Change to small capitals	≡ under matter to be changed	≡
Change to bold type	~ under matter to be changed	~
Change to bold italic	≈ under matter to be changed	≈
Change to lower case	Encircle matter to be changed	≡
Change italic to upright type	(As above)	⧻
Change bold to non-bold type	(As above)	⧻
Insert 'superior' character	/ through character or ⧵ where required	Y or Y under character e.g. Y or Y
Insert 'inferior' character	(As above)	⧵ over character e.g. ⧵
Insert full stop	(As above)	⊙
Insert comma	(As above)	,
Insert single quotation marks	(As above)	Y or Y and/or Y or Y
Insert double quotation marks	(As above)	Y or Y and/or Y or Y
Insert hyphen	(As above)	⎯
Start new paragraph	┐	┐
No new paragraph	┐	┐
Transpose	↔	↔
Close up	linking ○ characters	○
Insert or substitute space between characters or words	/ through character or ⧵ where required	Y
Reduce space between characters or words		↑