COMPLEXITY OF LINEAR PROBLEMS WITH A FIXED OUTPUT BASIS

ERICH NOVAK AND HENRYK WOŹNIAKOWSKI

Abstract. We use an information-based complexity approach to study the complexity of approximation of linear operators. We assume that the values of a linear operator may be elements of an infinite dimensional space G. It seems reasonable to look for an approximation as a linear combination of some elements g_i from the space G, and compute only the coefficients c_i of this linear combination. We study the case when the elements g_i are fixed and compare it with the case where the g_i can be chosen arbitrarily. We show examples of linear problems where a significant output data compression is possible by the use of a nonlinear algorithm, and this nonlinear algorithm is much better than all linear algorithms. We also provide an example of a linear problem for which one piece of information is enough whereas an optimal (minimal cost) algorithm must use information of much higher cardinality.

1. Introduction

Information-based complexity, IBC, studies the complexity of the approximate solution of continuous problems, see Section 2. The IBC model of computation is based on the real number model with information oracles, see Section 3. Complexity is understood as the minimal cost of computing an approximation with a given error bound ε. The reader interested in IBC is referred to the recent book of Traub and Werschulz (1998) which contains an extensive bibliography.

A typical IBC problem is to approximate the values $S(f)$ in a normed space G over the reals for elements f from a given set F. In the IBC model of computation we assume that we can work with elements g_i of the space G and perform basic operations on them. That is, we can compute linear combinations

$$g = \sum_{k=1}^{m} c_k g_{i_k}.$$

More formally, we assume the real number model and an output of the form

$$\mathrm{out}\,(f) = [i_1, c_1, i_2, c_2, \dots, i_m, c_m], \tag{1}$$

with $i_k \in \mathbb{N}$ and $c_k \in \mathbb{R}$, is identified with the vector g.

We comment on the choice of the sequence $\{g_i\}$ and distinguish between two cases. In the first case, we are free to choose the sequence $\{g_i\}$. That is, we choose $\{g_i\}$ depending on the global parameters of the problem (such as S, F, G and the error ε) but independently

Date: October 1998.

of f. It is desirable to choose $\{g_i\}$ such that the number m in out (f) is minimized and the coefficients c_k and i_k are easy to compute for all $f \in F$. This case is usually studied in IBC and tight complexity bounds are obtained for many problems. We will call the complexity for this case, i.e., when we can freely choose $\{g_i\}$, as the *standard* complexity.

In the second case, we are *not* free to choose the g_i's. The sequence $\{g_i\}$ is given a priori as part of the formulation of the problem. For example, $\{g_i\}$ can be given as a sequence of B-splines and we want to use it for approximate solutions of partial differential equations. Furthermore, we may want to use the same sequence $\{g_i\}$ for different problems, i.e., for different operators S and/or different domains F.

For fixed $\{g_i\}$, it may obviously happen that even $m = \infty$ in (1) is not enough to guarantee a small error of our approximation. This happens if the sequence $\{g_i\}$ is badly chosen for the operator S. It is certainly reasonable to assume that $S(f)$ can be approximated with any given accuracy by a linear combination of the g_i's. This is always the case if the space G is separable and is equal to the closed linear hull of the g_i.

The main problem addressed in this paper is to study complexity for a fixed sequence $\{g_i\}$ and compare it to the standard complexity. Not surprisingly, even if G is the closed linear hull of the g_i, the complexity for a fixed sequence $\{g_i\}$ may be much larger than the standard complexity.

We also define the *output complexity* as the minimal cost of solving the problem assuming that we charge *only* for the output, i.e., all the other operations are free of charge.

With this modification we need to revisit typical questions studied in IBC, and to see whether we get more or less the same answers as for the standard complexity. These questions include the problem of adaption, the existence of linear optimal error and cost algorithms, tradeoffs between optimal information and information used by an optimal cost algorithm. Indeed, a number of results are different and we mention two of them.

We restrict ourselves to *linear* problems, i.e., S is a linear operator. We also assume that we approximate $S(f)$ for elements from F which is convex and symmetric. Sometimes we also assume that F is a subset of a Hilbert space or that G is a space of bounded functions equipped with the sup norm. We find examples of linear problems for which:

(1) *all linear* algorithms must output roughly ε^{-2} numbers whereas the output complexity as well as the total complexity is of order $\varepsilon^{-1} \ln 1/\varepsilon$ and is achieved by a *nonlinear* algorithm. In this case, we achieve a significant output data compression by using a nonlinear algorithm.

(2) the optimal information has cardinality one and the use of it leads to very expensive algorithms. If we, however, use information of higher cardinality then there are algorithms solving the problem with much smaller total cost. Hence we have a *tradeoff* between information cost and computational cost: an algorithm with small information cost has high computational cost and vice versa. The existence of such a problem has long been an open question.

We wish to add that linear problems for which the results mentioned above hold are artificial. We do not know if the same or similar results hold for practically important linear problems.

We end this introduction by a brief discussion of proof techniques for getting *lower bounds* on the complexity. Various lower bounds can be obtained depending on which part of algorithms and their cost is studied:

a) information cost and the notion of the radius of information are used to obtain lower bounds on the number of needed oracle calls;

b) combinatoric and/or arithmetic considerations are used to obtain lower bounds on the number of needed arithmetic operations;

c) topological considerations are used to obtain lower bounds on the number of branchings; although we do not use this approach here, the relevant papers include Hertling (1996), Novak and Woźniakowski (1996), Smale (1987), Vassiliev (1996);

d) best m-term approximation and similar tools from approximation theory are used to obtain lower bounds on the number of output elements.

Which of these lower bounds is the best one depends on the particular problem. In general we should, of course, try to estimate the (suitably weighted) combination of all these lower bounds to have good lower bounds for the complexity.

Sometimes the cost of computation is irrelevant and we are only interested in minimizing the number of outputs. This corresponds to data compression which is formalized by the output complexity. The lower bounds on the output complexity are based on best m-term approximation and they are often sharp. In this paper we allow arbitrary weights when combining the different ingredients of cost. In this way we study the output complexity as well as the other complexities in a uniform way.

2. Linear Problems

In this paper we study linear problems. We first present an example which will be used to motivate our approach. Suppose we want to solve an operator equation

$$(Au)(x) = f(x) \quad \text{ for all } x \in \Omega \tag{2}$$

on a bounded domain $\Omega \subset \mathbb{R}^d$ with known conditions on the boundary of Ω. Here we assume that the operator A is linear and that we have information concerning f such as its values at various points from Ω. We want to compute an approximation of u with a given error in some norm.

This problem is an instance of the following formulation: Compute an approximation of the value $S(f)$ of a linear operator

$$S : X \to G \tag{3}$$

for $f \in F$, where $F \subset X$. In our example, F is typically a ball of a Sobolev space X, and G can be taken as $L_2(\Omega)$. We have $A = S^{-1}$ and $S(f) = u$ iff $Au = f$. Observe that (2) does *not* lead to a linear problem if we want to compute an approximation of S, where $S(A,f) = A^{-1}(f)$, for a class of pairs (A,f) with different operators A.

We assume that X is a normed space of functions and that G is also a normed space both over the real field. We always assume that S is linear. In many cases the space X is infinite dimensional, as in our example, and therefore $f \in X$ cannot directly be an input to a computation. We usually discretize f and S by, for example, a finite element method (FEM). Accordingly, numerical algorithms (methods) are often based on the *a priori (or global) information* $f \in F$ and the *computed (or local) information*

$$N(f) = [L_1(f), L_2(f), \dots, L_n(f)] \in \mathbb{R}^n \tag{4}$$

with linear continuous functionals $L_k : X \to \mathbb{R}$. Hence, numerical algorithms only use partial information $N(f)$ of $f \in F$. The most important example is

$$N(f) = [f(x_1), f(x_2), \dots, f(x_n)], \tag{5}$$

but other functionals are also common. Examples of such functionals include weighted integrals, Fourier coefficients, wavelet coefficients, or derivative values. In general we assume that the linear continuous functionals L_i are from a given class Λ, where $\Lambda \subset X^*$.

An information operator N is called *nonadaptive*, if the functionals L_k are fixed in advance and do not depend on f. One might hope that it is possible to learn about f during the computation of the values $L_1(f), \dots, L_{k-1}(f)$ and to choose the next functional L_k suitably to reduce the error. This leads to *adaptive* information, where the choice of L_k may depend on the already computed values $L_1(f), \dots, L_{k-1}(f)$. For instance, in the case $L_k(f) = f(x_k)$, the knot x_k may depend on the known function values via

$$x_k = \psi_k(f(x_1), \dots, f(x_{k-1})), \tag{6}$$

for some function ψ_k of $(k-1)$ variables. In (4) we define information with *fixed cardinality*. More generally, we may also use an *adaptive stopping rule* and obtain information with *varying cardinality*, where $n = n(f)$ depends on f. After the computation of $L_k(f)$ we decide, on the basis of the computed information $L_1(f), \dots, L_k(f)$, whether additional information is needed or not. If so we select the next functional L_{k+1}, compute $L_{k+1}(f)$ and repeat the same argument with k replaced by $k+1$. If not we set $n(f) = k$.

We want to compute an approximation of $S(f) \in G$ based on the information $N(f)$ for $f \in F$. Let $\varphi(N(f)) \in G$ be such an approximation. We say that $\varphi \circ N$ is an adaptive algorithm if N is adaptive, and $\varphi \circ N$ is a nonadaptive algorithm if N is nonadaptive.

If G is a one dimensional space it is clear that G can be identified with $\mathbb{R}$ and in this case the approximation $\varphi(N(f))$ is a real number. If G is a m dimensional space then G can be identified as $\mathbb{R}^m$ and the approximation $\varphi(N(f))$ is a vector of m components. In both cases, we can output $\varphi(N(f))$ as m real numbers.

Assume now that G is infinite dimensional as it is the case for our example of solving operator equations. Then it is not clear how we can output the approximation $\varphi(N(f))$. In this case, we may choose a sequence $g_1, g_2, \ldots \in G$ and assume that the approximation

$\varphi(N(f))$ is of the form

$$\varphi(N(f)) = \sum_{k=1}^{m} c_k \cdot g_{i_k} \in G \tag{7}$$

for some finite m, where m and the real numbers c_k and the integer indices i_k may depend on the information $N(f)$. We then output $2m$ numbers,

$$\text{out}\,(f) = [i_1, c_1, \dots, i_m, c_m] \in \mathbb{R}^{2m}. \tag{8}$$

We comment on (7) and (8). To compute out (f) we only perform operations on real numbers. Knowing out (f) we may use (7) as follows. Assume that g_{i_k} are functions. We need to have subroutines which compute $g_{i_k}(x)$ for a given x. Then we can compute $\varphi(N(f))(x) = \sum_{k=1}^{m} c_k \cdot g_{i_k}(x)$ by performing only real multiplications and additions.

The error of $\varphi(N(f))$ is

$$e(\varphi, N, f) = \Big\| S(f) - \sum_{k=1}^{m} c_k \cdot g_{i_k} \Big\|_G. \tag{9}$$

We are interested in the *worst case error* which is defined as

$$e(\varphi, N, F) = \sup_{f \in F} \Big\| S(f) - \sum_{k=1}^{m} c_k \cdot g_{i_k} \Big\|_G. \tag{10}$$

We want to find φ and N for which $e(\varphi, N, F) \le \varepsilon$, where ε is a given error threshold.

We now discuss the choice of the g_i. As already mentioned in the introduction we have two cases. In the first case, we are allowed to choose the sequence $\{g_i\}$ depending on the global parameters of the problem. That is, $\{g_i\}$ may depend on S, F, G, Λ and ε but it is independent of elements f's. Usually it is a good idea to find a sequence $\{g_i\}$ which minimizes m and/or allows a "simple" computation of i_k and c_k in (8). This may, however, lead to "complicated" sequences $\{g_i\}$. This case is usually studied in the information-based complexity literature.

In the second case, the sequence $\{g_i\}$ is fixed and given a priori. Then we are looking for suitable coefficients c_k and indices i_k in (8). In most cases, we will assume that $S(f)$ can be approximated with an arbitrary small error by a linear combination of the g_i's.

3. Model of computation, cost, and complexity

The standard model of computation used in numerical analysis and scientific computing is the real number model. This is also the underlying model of this paper. For a formal description see Blum, Shub and Smale (1989) and Blum, Cucker, Shub and Smale (1997), as well as Novak (1995) and Novak and Woźniakowski (1996), and also Woźniakowski (1997) for a discussion of this model. Here we only mention some properties of this model.

We assume that the reader is familiar with the concept of an algorithm over a ring presented by Blum, Shub and Smale (1989). We sketch how computation is performed by such algorithms and we restrict ourselves only to algorithms over the reals. Input and output consist of finitely many real numbers. We have arithmetic instructions, a jump instruction, and several copy instructions. We now describe these instructions.

The standard arithmetic operations are the following: addition of a constant, addition of two numbers, multiplication by a constant, multiplication of two numbers, and division of two numbers. Division by 0 is equivalent to a non-terminating computation. Other instructions could be allowed but are not studied in this paper.

The algorithm is able to perform backwards and forwards jumps in the list of instructions. We also have the usual copy instructions including indirect addressing, see also Novak (1995). It is clear that many problems of computational mathematics are computable by such algorithms. Examples include problems which are determined by finitely many parameters and whose solutions may be obtained by performing a finite number of arithmetic operations and comparisons. This holds for problems involving polynomials and matrices.

To deal also with problems that are defined for general functions, we need to have an *information* operation. Typically a black box computes $f(x)$ for any x. This black box (oracle) is an additional computational device. Observe that, in this case, the information about the function f is restricted to $f(x_i)$ for finitely many i. This information, which in general does not determine the function f uniquely, is *partial.*

The cost of computation can be simply defined by counting how many operations of various types are performed. In computational mathematics, one usually counts the number of arithmetic operations, the number of comparisons, and the number of information operations (oracle calls). The cost of input, output as well as copy instructions is usually neglected. For simplicity, we also assume that input, and copy instructions cost zero, although it is obvious how to proceed without this assumption.

We now discuss output instructions. We will present examples where the length $2m$ of the output $\mathrm{out}\,(f) = [i_1, c_1, \ldots, i_m, c_m] \in \mathbb{R}^{2m}$ is much larger than the actual cost of the other (arithmetic, branching and information) instructions. Hence, it seems reasonable to add the cost of output. Let $\mathbf{c}_{\mathrm{out}}$ be the cost of outputting one pair $[i_j, c_j]$. Then $m \cdot \mathbf{c}_{\mathrm{out}}$ is the cost of $\mathrm{out}\,(f)$. Here we assume $\mathbf{c}_{\mathrm{out}} \geq 0$; hence the case with zero cost of output can be also included.

Assume that $\mathbf{c}_{\mathrm{ari}}$ denotes the cost of performing one arithmetic operation, whereas $\mathbf{c}_{\mathrm{top}}$ denotes the cost of one comparison, i.e., the cost of performing one comparison of two real numbers. We assume that $\mathbf{c}_{\mathrm{info}}$ is the cost of one information operation. Suppose that the algorithm terminates on input $f \in F$. Let the computation require n_{ari} arithmetic operations, n_{top} comparisons, and n_{info} information operations. The cost of computing the output, $\mathrm{out}\,(f)$, is

$$\mathrm{cost}\,(\varphi, N, f) = n_{\mathrm{ari}}\,\mathbf{c}_{\mathrm{ari}} + n_{\mathrm{top}}\,\mathbf{c}_{\mathrm{top}} + n_{\mathrm{info}}\,\mathbf{c}_{\mathrm{info}} + m\,\mathbf{c}_{\mathrm{out}}. \tag{11}$$

Sometimes we call $n_{\mathrm{info}}\,\mathbf{c}_{\mathrm{info}}$ the information cost and $n_{\mathrm{ari}}\,\mathbf{c}_{\mathrm{ari}} + n_{\mathrm{top}}\,\mathbf{c}_{\mathrm{top}} + m\,\mathbf{c}_{\mathrm{out}}$ the computational cost.

If data compression is the main concern one would like to output as few real numbers. This corresponds to the case where the constants $\mathbf{c}_{\text{ari}}$, $\mathbf{c}_{\text{top}}$ and $\mathbf{c}_{\text{info}}$ all are relatively small or even zero while $\mathbf{c}_{\text{out}}$ is positive.

We assumed, for simplicity, that the cost of all arithmetic operations is the same. Of course, this could be easily modified. Usually, it is much more expensive to compute an information operation than any other operation. For some practical problems, computation of $L(f) = f(t)$ may require millions of arithmetic operations and comparisons. That is, it can take hours even on a modern computer whereas arithmetic operations or comparisons can be done in a fraction of a second. That is the reason why we use different parameters for the cost of permissible operations. We also want to see how arithmetic, comparison, information and output operations affect the complexity of a problem. The global cost of an algorithm $\varphi \circ N$ over F in the worst case setting is defined as

$$\text{cost}\,(\varphi, N, F) = \sup_{f\in F} \text{cost}\,(\varphi, N, f). \tag{12}$$

Here we identify, for simplicity, the mapping $\varphi \circ N$ with the simplest algorithm for its computation.

Many problems of computational mathematics can be studied within the model of computation given by oracles and algorithms over the reals. We give a few trivial examples. Assume that we want to compute a constant vector

$$g = \varphi(N(f)) = \sum_{k=1}^{m} c_k \cdot g_{i_k} \in G.$$

Then the output is

$$\text{out}\,(f) = [i_1, c_1, \dots, i_m, c_m] \in \mathbb{R}^{2m}$$

and does not depend on $f \in F$, i.e., the numbers i_k and c_k just have to be copied, and no computation is needed. Hence the constant function $g = \varphi(N(f))$ can be computed with cost $m \cdot \mathbf{c}_{\text{out}}$ if g is of the form $\sum_{k=1}^{m} c_k \cdot g_{i_k}$. Observe that we have zero cost in the case where we do not charge for the output, even if m is very large.

Assume now that $\varphi \circ N$ is linear,

$$A_{n,m}(f) := \varphi(N(f)) = \sum_{j=1}^{n} \sum_{k=1}^{m} L_j(f) \cdot c_{j,k} \cdot g_{i_k} \tag{13}$$

with i_k and $c_{j,k}, g_{i_k}$ independent of f. We assume for simplicity that the mapping given by (13) cannot be written as a $A_{\tilde{n},\tilde{m}}$ with $\tilde{n} < n$ or $\tilde{m} < m$. The output for $A_{n,m}$ is given by

$$\text{out}\,(f) = \left[i_1,\, \sum_{j=1}^{n} L_j(f) \cdot c_{j,1},\, i_2,\, \sum_{j=1}^{n} L_j(f) \cdot c_{j,2}, \dots, i_m,\, \sum_{j=1}^{n} L_j(f) \cdot c_{j,m} \right]. \tag{14}$$

The i_k's do not depend on f, hence they can be copied (without cost) from the algorithm to the output. The cost of computing $A_{n,m}$ is at least

$$\text{cost}\,(A_{n,m}) \geq n \cdot \mathbf{c}_{\text{info}} + m \cdot \mathbf{c}_{\text{out}}\,,$$

and can be much larger, depending on the cost of computing the coefficients

$$\sum_{j=1}^{n} L_j(f) \cdot c_{j,k}, \qquad k = 1, \ldots, m.$$

We always have

$$\mathrm{cost}\,(A_{n,m}) \le n \cdot \mathbf{c}_{\mathrm{info}} + m \cdot \mathbf{c}_{\mathrm{out}} + (2n-1)m \cdot \mathbf{c}_{\mathrm{ari}}. \tag{15}$$

In general, we expect the cost $(A_{n,m})$ to be of the same order as the right hand side of (15).

It is interesting to compare (15) with the cost of $A_{n,m}$ in the standard IBC model, where we can choose some $\tilde{g}_i$'s instead of the g_i's. Define

$$\tilde{g}_i = \sum_{k=1}^{m} c_{j,k} \cdot g_{i_k}.$$

Then

$$A_{n,m}(f) = \sum_{j=1}^{n} L_j(f) \cdot \tilde{g}_j$$

and the output with respect to the $\tilde{g}_j$'s is

$$\mathrm{out}\,(f) = [i_1,\, L_1(f),\, i_2,\, L_2(f), \ldots, i_n,\, L_n(f)]\,,$$

with cost

$$\mathrm{cost}\,(A_{n,m}) = n \cdot (\mathbf{c}_{\mathrm{info}} + \mathbf{c}_{\mathrm{out}}\,). \tag{16}$$

Hence, in the standard IBC model, the cost of a linear algorithm is always proportional to n and we can easily ignore the arithmetic and the output cost in the presence of the information cost.

The ε-complexity of the problem is the minimal cost needed to compute an ε-approximation for every $f \in F$. It depends, of course, on F and S as well as on the set Λ of admissible information functionals and on the sequence of the g_i. We therefore write

$$\mathrm{comp}\,(\varepsilon, S, F, \Lambda, \{g_i\}) = \inf\{\mathrm{cost}\,(\varphi, N, F) \mid\ e(\varphi, N, F) \le \varepsilon\}. \tag{17}$$

In the case when the sequence $\{g_i\}$ can be chosen arbitrarily we have

$$\mathrm{comp}\,(\varepsilon, S, F, \Lambda) = \inf_{\{g_i\}} \mathrm{comp}\,(\varepsilon, S, F, \Lambda, \{g_i\}). \tag{18}$$

We also use a short form such as comp (ε) for comp $(\varepsilon, S, F, \Lambda, \{g_i\})$ or comp $(\varepsilon, S, F, \Lambda)$ if it is clear from the context what we mean by S, F, Λ and $\{g_i\}$. By convention we set $\inf \emptyset = \infty$. That is, if there exists no algorithm with finite cost with error at most ε then comp $(\varepsilon) = \infty$.

This defines complexity in the worst case setting. Complexity depends, in particular, on the constants $\mathbf{c}_{\mathrm{ari}}$, $\mathbf{c}_{\mathrm{top}}$, $\mathbf{c}_{\mathrm{info}}$, and $\mathbf{c}_{\mathrm{out}}$. Since complexity minimizes the total cost, it minimizes the number of arithmetic, comparison and information operations weighted by $\mathbf{c}_{\mathrm{ari}}$, $\mathbf{c}_{\mathrm{top}}$, $\mathbf{c}_{\mathrm{info}}$, and $\mathbf{c}_{\mathrm{out}}$. For example, if $\mathbf{c}_{\mathrm{info}}$ is much larger than the other constants then we should perform less information operations at the expense of additional arithmetic and comparison operations. On the other hand, if $\mathbf{c}_{\rm top}$ is very large then we should eliminate as many comparisons as possible. In any case, depending on the values of the constants we may perform a different number of arithmetic, comparison, information, and output operations.

It is interesting to study how many operations of different types are necessary independently of the other operations. This can be achieved by taking three of $\mathbf{c}_{\rm ari}$, $\mathbf{c}_{\rm top}$, $\mathbf{c}_{\rm info}$, $\mathbf{c}_{\rm out}$ equal to zero. For simplicity we take the fourth one equal to one. In this way we obtain different complexities. More precisely, we obtain the *arithmetic* complexity $\mathrm{comp}_{\rm ari}(\varepsilon)$, the *branching* or *topological* complexity $\mathrm{comp}_{\rm top}(\varepsilon)$, the *information* complexity $\mathrm{comp}_{\rm info}(\varepsilon)$ and the *output* complexity $\mathrm{comp}_{\rm out}(\varepsilon)$. Clearly, the (total) complexity $\mathrm{comp}\,(\varepsilon)$ is bounded from below by

$$\mathrm{comp}\,(\varepsilon) \ge \mathbf{c}_{\rm ari}\,\mathrm{comp}_{\rm ari}(\varepsilon) + \mathbf{c}_{\rm top}\,\mathrm{comp}_{\rm top}(\varepsilon) + \mathbf{c}_{\rm info}\,\mathrm{comp}_{\rm info}(\varepsilon) + \mathbf{c}_{\rm out}\mathrm{comp}_{\rm out}(\varepsilon). \tag{19}$$

It may happen that the total complexity goes to infinity as ε goes to zero, whereas one or even several of the other complexities are bounded or even zero. In these definitions of the ε-complexity, the error parameter ε is fixed, i.e., for different ε one may take completely different algorithms. We studied uniform algorithms (which take $\varepsilon > 0$ as an input) recently in Novak, Woźniakowski (1997).

We end this section by a simple example which illustrates differences between various complexities. Consider the mapping

$$S(f) = L^*(f)\cdot g, \tag{20}$$

where $L^* \in \Lambda$ is a nonzero functional, $g \in G$ is a nonzero element, and $F \subset X$ is convex and symmetric. Let

$$K := \sup_{f\in F} L^*(f).$$

Observe that for $\varepsilon \ge K\|g\|$ the problem is trivial since it can be solved by the zero algorithm with cost zero. In this case, all complexities are just zero.

Assume then that $\varepsilon < K\|g\|$. We first consider the information complexity. We show that

$$\mathrm{comp}_{\rm info}(\varepsilon) \in \{1, \infty\}.$$

First, we exclude the case $\mathrm{comp}_{\rm info}(\varepsilon) = 0$. Indeed, if we do not compute any linear functionals from Λ then no algorithm depends on f, and for all f from F the algorithm outputs the same numbers which correspond to the same vector, say, h. Since $S(F) = [-K, K]g$ we have $\sup_{f\in F}\|S(f) - h\| \ge K\|g\| > \varepsilon$. This means that the error of any algorithm is larger than ε. Hence, we must compute at least one functional. We obviously can compute $L^*(f)$ and it may seem that always $\mathrm{comp}_{\rm info}(\varepsilon) = 1$. In general, this is not true. Indeed, assume that $K = \infty$ and

$$a_m(g, \{g_i\}) := \inf_{c_k,\, i_k} \Big\| g - \sum_{k=1}^m c_k \cdot g_{i_k} \Big\|_G$$

is positive for all m. Then the error of any algorithm is infinity since any algorithm corresponds to $c(f)h$ for some $c(f)$, which can be equal to $L^*(f)$, but h is such that $\inf_\alpha \|g-\alpha h\| > 0$. Then

$$\sup_{f\in F} \|L^*(f) - c(f)h\| \;\ge\; \sup_{f\in F} |L^*(f)| \inf_\alpha \|g-\alpha h\| \;=\; \infty.$$

In this case, $\mathrm{comp}_{\,\mathrm{info}}(\varepsilon) = \infty$ for all $\varepsilon \le K\|g\|$. On the other hand, if K is finite then

$$\mathrm{comp}_{\,\mathrm{info}}(\varepsilon) \;=\; 1 \quad \text{if } \varepsilon > K\, a_m(g,\{g_i\})$$

for some finite m. Indeed, we can take the algorithm $L^*(f)h$ with h such that $\|g-h\|$ is sufficiently close to $a_m(g,\{g_i\})$. This algorithm is well defined and has error at most ε.

Observe that if the span of the g_i's is dense in G then $\lim_m a_m(g,\{g_i\}) = 0$ and $\mathrm{comp}_{\,\mathrm{info}}(\varepsilon) = 1$ for all positive ε. For $\varepsilon = 0$, $\mathrm{comp}_{\,\mathrm{info}}(0) = 1$ iff g can be expressed as a finite linear combination of g_i's. If g is not a finite linear combination of the g_i, then the error of any algorithm is positive and therefore we cannot solve the problem exactly in a finite cost.

We now turn to the output and total complexities for $\varepsilon < K\|g\|$. If we are free to choose the g_i's, or if g happens to be one of the g_i then the problem is trivial and S can be computed exactly with cost $\mathbf{c}_{\mathrm{info}} + \mathbf{c}_{\mathrm{out}}$. This means

$$\mathrm{comp}_{\,\mathrm{out}}(\varepsilon,S,F,\Lambda) \;=\; \mathrm{comp}_{\,\mathrm{out}}(\varepsilon,S,F,\Lambda,\{g_i\}) \;=\; 1$$

and

$$\mathrm{comp}\,(\varepsilon,S,F,\Lambda) \;=\; \mathrm{comp}\,(\varepsilon,S,F,\Lambda,\{g_i\}) \;=\; \mathbf{c}_{\mathrm{info}} + \mathbf{c}_{\mathrm{out}}.$$

It is easy to show that in general we have

$$\mathrm{comp}\,(\varepsilon,S,F,\Lambda,\{g_i\}) \;=\; \mathbf{c}_{\mathrm{info}} + m\cdot(\mathbf{c}_{\mathrm{out}} + \mathbf{c}_{\mathrm{ari}}), \quad \text{and} \quad \mathrm{comp}_{\,\mathrm{out}}(\varepsilon,S,F,\Lambda,\{g_i\}) \;=\; m,$$

where m is the smallest integer for which $a_m(g,\{g_i\}) \le \varepsilon/K$.

If the span of the g_i is dense in G and if K is finite then the ε-complexity will be finite for any $\varepsilon > 0$, but can increase to infinity very fast as ε goes to zero. If $K = \infty$, for example if $F = X$, then the ε-complexity is infinite for all $\varepsilon > 0$ unless g is a finite linear combination of the g_i. If the sequence of the $a_m(g,\{g_i\})$ does not converge to zero then the ε-complexity is infinity for some or all $\varepsilon > 0$, unless $K = 0$.

Observe that in all cases linear algorithms are optimal and are of the form $L^*(f)h$ with h in the span of the g_i's. Hence, $\mathrm{comp}\,(\varepsilon,S,F,\Lambda) = \mathbf{c}_{\mathrm{info}} + \mathbf{c}_{\mathrm{out}}$ whereas $\mathrm{comp}\,(\varepsilon,S,F,\Lambda,\{g_i\})$ can be much larger and may tend to infinity as ε goes to zero.

4. Lower bounds for arbitrary algorithms

The notion of radius of information $r(N)$ is fundamental in IBC. It gives a sharp lower bound for the error of any algorithm using N, i.e.,

$$r(N) = \inf_\varphi e(\varphi, N). \tag{21}$$

We take here the infimum over *all* $\varphi : N(F) \to G$ and we do not require that φ is computable, i.e., that $\varphi(N(f))$ can be computed in a finite number of operations. We denote by

$$r_n = r_n(S, F, \Lambda) \tag{22}$$

the minimal radius of information, for all N using at most n linear functionals from Λ. If $r_{n+1} \le \varepsilon < r_n$ then we define the ε-cardinality as

$$\text{card}\,(\varepsilon) = \text{card}\,(\varepsilon, S, F, \Lambda) = n + 1. \tag{23}$$

This yields a lower bound on the total and information complexities given in the simple lemma below.

Lemma 1.

$$\text{comp}\,(\varepsilon, S, F, \Lambda) \ge \text{comp}_{\,\text{info}}(\varepsilon, S, F, \Lambda) \ge \text{card}\,(\varepsilon) \cdot \mathbf{c}_{\text{info}}. \tag{24}$$

Observe that the minimal radii of information r_n are strictly related to *approximation* rather than to *computation* since they are based on all noncomputable or computable and possible very expensive φ's. Still their inverses provide a lower bound on the total and information complexities. Sometimes this lower bound is bad, i.e., far away from the actual complexity. Surprisingly often, however, this lower bound is very good. Indeed, for many problems we have essentially equalities in (24).

Remark 1. For many problems, including all problems studied in this paper, we have $\text{comp}_{\,\text{info}}(\varepsilon, S, F, \Lambda) = \text{card}\,(\varepsilon) \cdot \mathbf{c}_{\text{info}}$ In fact, in some papers $\text{card}\,(\varepsilon) \cdot \mathbf{c}_{\text{info}}$ is defined as information complexity. In general, $\text{comp}_{\,\text{info}}(\varepsilon, S, F, \Lambda)$ and $\text{card}\,(\varepsilon) \cdot \mathbf{c}_{\text{info}}$ may differ since for $\text{card}\,(\varepsilon) \cdot \mathbf{c}_{\text{info}}$ we allow arbitrary φ's while for $\text{comp}_{\,\text{info}}(\varepsilon, S, F, \Lambda)$ we allow only computable φ's. It seems to us that both definitions of information complexity are useful and, as already mentioned, coincide in most practical cases.

By (24) we get a lower bound for $\text{comp}\,(\varepsilon, S, F, \Lambda, \{g_i\})$ which does not depend on the sequence $\{g_i\}$. Now we want to study the dependence on the $\{g_i\}$. We use the following definition.

Definition 1. Assume that $g \in G$ and a sequence $\{g_i\}$ in G are given. A *best m-term approximation* of g (by elements of the given sequence) is a sum $\tilde{g} = \sum_{k=1}^{m} c_k \cdot g_{i_k}$ such that $\|g - \tilde{g}\|$ is minimal. We define $a_m\,(g, \{g_i\})$ by

$$a_m(g, \{g_i\}) = \inf_{c_k,\, i_k} \Big\| g - \sum_{k=1}^{m} c_k \cdot g_{i_k} \Big\|_G. \tag{25}$$

We call this number *the error of the best m-term approximation*, even if the inf in (25) is not attained. The following lemma is obvious.

Lemma 2. *If* $\varepsilon < \sup_{f \in F} a_m\,(S(f), \{g_i\})$ *then*

$$\text{comp}\;(\varepsilon, S, F, \Lambda, \{g_i\}) \ge \text{comp}_{\,\text{out}}\,(\varepsilon, S, F, \Lambda, \{g_i\}) \ge (m+1) \cdot \mathbf{c}_{\text{out}}\,. \tag{26}$$

Remark 2. Which one of the lower bounds (24) or (26) is larger depends on the specific problem. It is easy to construct examples where the numbers $\mathrm{comp}_{\mathrm{out}}\,(\varepsilon, S, F, \Lambda, \{g_i\})$ are much larger than the numbers $\mathrm{card}\,(\varepsilon) \cdot \mathbf{c}_{\mathrm{info}}$. We have presented such an example already in Section 3.

For some standard nonadaptive algorithms, for example such as FEM with a quasi uniform partition, good upper bounds for the error are often known. Sometimes these upper bounds are compared with the lower bounds from Lemma 2 and then there is a huge gap between them, see Dahlke et al. (1997) and DeVore (1998). It would be, however, wrong to conclude that adaptive algorithms could be much better than optimal nonadaptive algorithms in this case. The first reason is that there might be a much better nonadaptive method, for example a FEM based on an irregular partition. Another reason is that for $g = S(f)$ the best choice of elements g_{i_k} in (25) depends on $S(f)$ and this information may not be available even by adaptive information.

In some cases it is very easy to see that the lower bound (26) cannot be sharp. Indeed, for integration, $S(f) = \int_0^1 f(t)dt$ and $G = \mathbb{R}$, we have

$$a_1(S(f), g_1) \;=\; 0, \qquad \text{with } g_1 = 1.$$

We only obtain the trivial lower bound $\mathrm{comp}\,(\varepsilon, S, F, \Lambda, \{g_i\}) \geq 1 \cdot \mathbf{c}_{\mathrm{out}}$. For a continuous function f, we know from the mean value theorem that $S(f) = f(x^*)\cdot g_1$ for some unknown $x^* \in [0, 1]$. The point is that x^* is not available even for adaptive information.

5. On the power of adaption

In this section we briefly discuss adaptive information. We first recall a result of Bakhvalov (1971), Gal and Micchelli (1980) and Traub and Woźniakowski (1980) about the power of adaption on the error level.

Theorem 1. *Assume that $S : X \to G$ is a linear operator and that F is a symmetric and convex subset of X. Assume that $\varphi \circ N$ is an arbitrary adaptive algorithm using, for $f = 0 \in F$, the functionals*

$$N^0(f) = [L_1^0(f), L_2^0(f), \ldots, L_n^0(f)].$$

Then there is a nonadaptive algorithm of the form $\varphi^* \circ N^0$ such that

$$e(\varphi^* \circ N^0) \leq 2\, e(\varphi \circ N).$$

Hence, adaption can only reduce the error at most by a factor of 2 as compared to non-adaption.

There are examples of linear problems for which adaption is (slightly) better than nonadaption, see Kon and Novak (1990).

We stress that all the assumptions of Theorem 1 are essential. If one of them is not satisfied then adaption may help significantly on the error level. In particular, Theorem 1 assumes that F is convex and symmetric. The set F reflects the *a priori (global) information* concerning the problem; often it is known that f has a certain smoothness and this knowledge may be expressed by $f \in F$. If our a priori information about the problem leads to a set F that is either nonsymmetric or nonconvex (or both) then we certainly cannot apply Theorem 1 and it is possible that adaption is significantly better. The same is true if we consider *nonlinear* problems like $S(A, f) = A^{-1}(f)$, for a general set of operators and a set of right hand sides f. See the survey Novak (1996) for more details.

The idea behind the proof of Theorem 1 is that nonadaptive information that is good for the zero function $0 \in F$ is also good for any other $f \in F$. This is true for *any linear problem with any norm.* However, this result does *not* automatically lead to good nonadaptive algorithms. In particular, we do not claim that the optimal nonadaptive knots are somehow uniformly distributed or equidistant. There are important examples where regular grid points are bad and the optimal (nonadaptive) knots are much more complicated. We stress this fact because we noticed that some authors compare poor nonadaptive algorithms based, for example, on a regular grid with sophisticated adaptive algorithms and (wrongly) conclude that adaptive algorithms are superior to *all* nonadaptive algorithms.

Theorem 1 states that as long as we are interested in errors it is enough to study nonadaptive algorithms. However, if we are interested in cost and complexity we cannot disregard adaptive algorithms. It may happen that for some linear problems any nonadaptive algorithm is much more expensive than a good adaptive one. Hence, adaption will reduce the cost not the error in this case. Although the existence of such linear problems has yet to be established, we cannot rule out this situation today.

6. Fast linear algorithms

In this section we assume that $F \subset X$ is convex and symmetric. Furthermore we assume that X is a Hilbert space or that G is a space $B(\Omega)$ of bounded functions equipped with the sup-norm. Under these conditions it is known that adaption does not help at all and that linear algorithms are optimal, see Traub, Wasilkowski, Woźniakowski (1988) and Mathé (1990). That is for any nonadaptive information N,

$$r(N) = \inf_{\varphi} e(\varphi, N), \quad \varphi \text{ linear.} \tag{27}$$

For general linear problems, (27) is not true. If S is continuous then we may loose at most a factor of $(1 + \sqrt{n})$ by replacing general algorithms by linear ones both using at most n linear functionals, see Mathé (1990). If S is not continuous then $r(N)$ may be small but all linear algorithms may have infinite error, see Werschulz and Woźniakowski (1986).

Under the above conditions the analysis of the ε-complexity in the sense of (18) (where the $\{g_i\}$ can be chosen arbitrarily) is easy. Linear nonadaptive algorithms are optimal in the class of all adaptive and nonlinear algorithms. The complexity is then roughly given by

$$\text{comp}\,(\varepsilon, S, F, \Lambda) \approx \text{card}\,(\varepsilon) \cdot \mathbf{c}_{\text{info}}.$$

The additional cost (for arithmetic operations and/or the output) is at most proportional to card$\,(\varepsilon)$, see (16). In this case, Lemma 1 provides a sharp lower bound.

Hence, we restrict our attention to the ε-complexity in the sense of (17), where the $\{g_i\}$ are given and fixed. In this case we study linear as well as nonlinear algorithms. The linear algorithms are of the form (13) or (14), where the i_k as well as the $c_{j,k}$ and g_{i_k} do not depend on f.

Lemma 3. *Assume that $F \subset X$ is convex and symmetric. Furthermore assume that X is a Hilbert space or that G is a space $B(\Omega)$ with the sup-norm. If, in addition, F is bounded and* span $\{g_i,\ i \in \mathbb{N}\}$ *is dense in G, then*

$$\inf\{e(A_{n,m}) \mid m \in \mathbb{N}\} = r_n, \tag{28}$$

where the infimum is taken over all linear algorithms of the form (13) *and r_n is as in* (22).

Proof. Since adaption and nonlinear algorithms do not help we know that

$$\inf\{e(A_n) \mid m \in \mathbb{N}\} = r_n,$$

where now the infimum is over all algorithms of the form

$$A_n(f) = \sum_{j=1}^{n} L_j(f)\, g^j$$

with arbitrary $g^j \in G$. Since span $\{g_i,\ i \in \mathbb{N}\}$ is dense in G, for any positive δ we can find a finite m and real numbers $c^j_{i_k}$ and elements $g^j_{i_k}$ from span $\{g_i,\ i \in \mathbb{N}\}$ such that

$$\Big\| g^j - \sum_{k=1}^{m} c^j_{i_k} g^j_{i_k} \Big\| \le \delta.$$

Letting $A_{n,m}(f) = \sum_{j=1}^{n} L_j(f) \sum_{k=1}^{m} c^j_{i_k} g^j_{i_k}$, we obtain a linear algorithm of the form (13) with error bounded by

$$\begin{aligned} \|S(f) - A_{n,m}(f)\| &\le \|S(f) - A_n(f)\| + \|A_n(f) - A_{n,m}(f)\| \\ &\le \|S(f) - A_n(f)\| + \delta \sum_{j=1}^{k} |L_j(f)| \\ &\le \|S(f) - A_n(f)\| + \delta\, M \sum_{j=1}^{k} \|L_j\|, \end{aligned}$$

where M is a bound on elements of F. Since δ can be arbitrarily small, the error of $A_{n,m}$ can be arbitrarily close to the error of A_n. This proves that the corresponding infimum of the errors of $A_{n,m}$ is r_n. □

Now we discuss the cost of linear algorithms. Linear algorithms are of the form (13), their error is at least r_n. The cost of such an algorithm is bounded by

$$\text{cost}\,(A_{n,m}) \le n \cdot \mathbf{c}_{\text{info}} + m \cdot \mathbf{c}_{\text{out}} + (2n-1)m \cdot \mathbf{c}_{\text{ari}} \tag{29}$$

but of course it can be much smaller. Observe that this upper bound on the cost of linear algorithms using n linear functionals also depends on m. Usually, m need not be related to n.

There are, however, many problems, where fast linear algorithms exist, according to the following definition. For solving PDEs, such fast algorithms often can be given by multiscale or multigrid methods, see Dahmen (1997), D'yakonov (1996), Hackbusch (1985).

Definition 2. We say that the problem admits a *fast* sequence of linear algorithms if there is a sequence $\{A_{n,m}\}$ of the form (13) with the following properties:
a) there is a $K > 0$ such that $\mathrm{cost}\,(A_{n,m}) \le n \cdot (\mathbf{c}_{\mathrm{info}} + K)$ for all n;
b) there is a $C \ge 1$ such that $e(A_{n,m}) \le C\, r_n$.

It is easy to check the power of a fast sequence of linear algorithms and a relation between their cost and complexity.

Theorem 2. *Assume that the problem admits a fast sequence of linear algorithms. If we define*

$$\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon, S, F, \Lambda, \{g_i\}) = \inf \mathrm{cost}\,(A_{n,m}), \tag{30}$$

where the infimum is over linear algorithms (13) *with error bounded by* ε*, then we obtain*

$$\frac{\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon, S, F, \Lambda, \{g_i\})}{\mathrm{comp}\,(\varepsilon, S, F, \Lambda)} \le \frac{\mathbf{c}_{\mathrm{info}} + K}{\mathbf{c}_{\mathrm{info}}} \cdot \frac{\mathrm{card}\,(\varepsilon/C)}{\mathrm{card}\,(\varepsilon)}. \tag{31}$$

Proof. By (24) we have

$$\mathrm{comp}\,(\varepsilon, S, F, \Lambda) \ge \mathrm{card}\,(\varepsilon) \cdot \mathbf{c}_{\mathrm{info}}.$$

Hence, it is enough to prove

$$\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon, S, F, \Lambda, \{g_i\}) \le (\mathbf{c}_{\mathrm{info}} + K) \cdot \mathrm{card}\,(\varepsilon/C). \tag{32}$$

Assume that $\varepsilon > 0$ is given. We define n by

$$r_n \le \varepsilon < r_{n-1}$$

or $n = \mathrm{card}\,(\varepsilon)$. We choose $\tilde{n}$ such that

$$C \cdot r_{\tilde{n}} \le r_n < C \cdot r_{\tilde{n}-1}.$$

With fast linear algorithms $A_{\tilde{n},m}$ we can achieve an error

$$e(A_{\tilde{n},m}) \le C \cdot r_{\tilde{n}} \le r_n \le \varepsilon.$$

Then we have $\mathrm{card}\,(r_n/C) = \tilde{n}$ and $\mathrm{cost}\,(A_{\tilde{n},m}) \le \tilde{n}(\mathbf{c}_{\mathrm{info}} + K)$. Since $r_n/C \le \varepsilon/C$ we have $\tilde{n} = \mathrm{card}\,(r_n/C) \le \mathrm{card}\,(\varepsilon/C)$ which completes the proof. □

Remark 3. A few comments on (31) are in order. For many problems we have

$$\mathrm{card}\,(\alpha\varepsilon) \le \alpha^{-\beta} \cdot \mathrm{card}\,(\varepsilon) \tag{33}$$

for a certain $\beta > 0$ and all $0 < \alpha \leq 1$. In this case we even obtain

$$\frac{\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon, S, F, \Lambda, \{g_i\})}{\mathrm{comp}\,(\varepsilon, S, F, \Lambda)} \leq \frac{\mathbf{c}_{\mathrm{info}} + K}{\mathbf{c}_{\mathrm{info}}} \cdot C^{\beta}. \tag{34}$$

Observe that the right side of (34) is a constant; it does not depend on ε. This means that fast linear algorithms are optimal modulo a multiplicative factor. Furthermore this factor is close to one if K is small relative to c_{info} and if C is close to one. Of course, this factor is large if the opposite is true. In this case, we may still want to use nonlinear algorithms.

In any case, if the problem admits a fast sequence of linear algorithms and (33) holds then linear nonadaptive algorithms are order-optimal with respect to cost as well as with respect to error.

7. A finite dimensional example

We present an example of a linear problem for which there exists a nonlinear algorithm that is better than *all* linear algorithms. Linear algorithms need a large amount of output while an effective data compression is possible for this nonlinear algorithm. In the following we assume that $\mathbf{c}_{\mathrm{out}} > 0$, i.e., we charge for the output. Define

$$S^n : \ell_1^n \to \ell_\infty^m, \qquad m = n(n-1)/2, \quad n \geq 2, \tag{35}$$

between finite dimensional spaces. For $1 \leq j < i \leq n$ we define the successive components of the vector $S^n(x)$ as

$$(S^n(x))_{i\cdot(i-1)/2+j-1} := x_i + x_j. \tag{36}$$

Hence, $S^n(x)$ is a vector with coordinates $(x_i + x_j)$, where the set (i,j) runs through all subsets of indices with two elements. As information we use functionals of the form $L(x) = x_i$, i.e., we can compute components of the vector x.

We consider the standard basis $g_1, \ldots, g_m$ in ℓ_∞^m. We present some results about the numbers

$$\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon, S^n, F, \Lambda, \{g_i\}) \quad \text{and} \quad \mathrm{comp}\,(\varepsilon, S^n, F, \Lambda, \{g_i\}).$$

We take F as the unit ball of ℓ_1^n. For $\varepsilon \geq 1$, the problem is trivial and can be solved by the zero algorithm without any cost. Hence

$$\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon, S^n, F, \Lambda, \{g_i\}) = \mathrm{comp}\,(\varepsilon, S^n, F, \Lambda, \{g_i\}) = 0, \quad \forall\, \varepsilon \geq 1. \tag{37}$$

Assume that $\varepsilon < 1$. It is easy to show that $\mathrm{card}\,(\varepsilon) = n$. Indeed, if we use less than n information operations, we do not know some component of x. Let i be this component. Then using x with all components different from i as zero we observe that all components of $S^n(x)$ with x_i vary from -1 to 1. This implies that the error of any algorithm must be at least 1, and we cannot find an ε-approximation. Obviously, when we compute all n components of x we can recover $S^n(x)$ exactly and this yields zero error. Hence, $\mathrm{card}\,(\varepsilon) = n$, $\forall\, \varepsilon \in [0,1)$, as claimed. We know from Lemma 1 that then

$$\mathrm{comp}\,(\varepsilon, S^n, F, \Lambda) \geq n \cdot \mathbf{c}_{\mathrm{info}}. \tag{38}$$

Is this bound sharp? That is, can we solve the problem with cost of the order n by using only linear or arbitrary algorithms?

We first study linear algorithms. For $\varepsilon \in [0.5, 1)$ we propose the following linear algorithm.

First step: Compute $c = \frac{1}{2} \sum_{i=1}^{n} x_i$. The cost is $n(\mathbf{c}_{\text{info}} + \mathbf{c}_{\text{ari}})$.

Second step: Take the approximation $\sum_{i=1}^{m} c\, g_i$. This is the vector with each coordinate equal to c. Observe that no further computation is needed. We just output m times the number c with cost $m\mathbf{c}_{\text{out}}$.

It is easy to prove that the (worst case) error of this linear algorithm is 1/2. Indeed, for $x \in F$ we have $\sum_{k=1}^{n} |x_k| \le 1$ and

$$\Big|x_i + x_j - c\Big| \;\le\; \Big|0.5(x_i + x_j) + 0.5 \sum_{k \neq \{i,j\}} x_k\Big| \;\le\; 0.5 \sum_{k=1}^{n} |x_k| \;\le\; 0.5.$$

Hence, we have proved that

$$\text{comp}^{\,\text{lin}}(\varepsilon, S^n, F, \Lambda, \{g_i\}) \le m \cdot \mathbf{c}_{\text{out}} + n \cdot (\mathbf{c}_{\text{info}} + \mathbf{c}_{\text{ari}}). \tag{39}$$

On the other hand, it is easy to see that, for $\varepsilon < 1$, a linear algorithm has to output an approximation of all m components of $S^n(x)$. Indeed, if one of the components is not outputted, then since the same component of the solution varies in $[-1, 1]$, the error of an algorithm must be at least 1. Hence we obtain

$$\text{comp}^{\,\text{lin}}(\varepsilon, S^n, F, \Lambda, \{g_i\}) \ge m \cdot \mathbf{c}_{\text{out}} + n \cdot \mathbf{c}_{\text{info}}. \tag{40}$$

By (39) and (40) we have tight bounds for the linear complexity,

$$\text{comp}^{\,\text{lin}}(\varepsilon, S^n, F, \Lambda, \{g_i\}) = m \cdot \mathbf{c}_{\text{out}} + n \cdot (\mathbf{c}_{\text{info}} + \, a\, \mathbf{c}_{\text{ari}}), \quad a \in [0,1], \quad \forall\, \varepsilon \in [1/2, 1).$$

We now study the case $\varepsilon < 1/2$ for linear algorithms. Observe that we can even compute the exact $S^n(x)$ with the cost $n \cdot \mathbf{c}_{\text{info}} + m \cdot (\mathbf{c}_{\text{out}} + \mathbf{c}_{\text{ari}})$. Combining this upper bound with the lower bound (40), which is also true for $\varepsilon < 1/2$, we obtain

$$\text{comp}^{\,\text{lin}}(\varepsilon, S^n, F, \Lambda, \{g_i\}) = n \cdot \mathbf{c}_{\text{info}} + m \cdot (\mathbf{c}_{\text{out}} + a\, \mathbf{c}_{\text{ari}}), \quad a \in [0,1], \quad \forall\, \varepsilon < 1/2. \tag{41}$$

It is interesting that the linear complexity is almost independent on ε, as long as $\varepsilon < 1$. It is always of the order n^2. We will see in a moment that a cost of the order n can be achieved by nonlinear algorithms, for any fixed $\varepsilon > 0$.

We now turn to nonlinear algorithms. We prove that for $\varepsilon < 1$ the complexity is of the order

$$\text{comp}\,(\varepsilon, S^n, F, \Lambda, \{g_i\}) \asymp n \cdot \mathbf{c}_{\text{info}} + n \cdot \min\,(n, \varepsilon^{-1})(\mathbf{c}_{\text{out}} + \mathbf{c}_{\text{ari}} + \mathbf{c}_{\text{top}}). \tag{42}$$

The lower bound follows from (38) and from the fact that we have to output at least $\Omega(n \cdot \min\,(n, \varepsilon^{-1}))$ numbers. Indeed, consider $1/(k+1) \le \varepsilon < 1/k$, where k is an integer.

For $n \ge k$ consider the input vector $f = [1/k, ..., 1/k, 0, 0, ..., 0] \in F$ with k coordinates equal to $1/k$. For $n < k$ we take instead the vector $f = [1/n, ..., 1/n] \in F$. Then, using

Lemma 2, we see that we have to output at least $k \cdot (n-1)/2$ or $n \cdot (n-1)/2$ numbers, respectively. Hence we have to output about $n \cdot \min(n, \varepsilon^{-1})$ numbers.

To prove an upper bound, we first define n_0 by

$$n_0 = \lceil 2/\varepsilon \rceil - 2. \tag{43}$$

Observe that n_0 can be precomputed. If $n \le n_0$ then we just take the obvious linear algorithm, with cost given by (41). Assume then that $n > n_0$. We propose a special nonlinear algorithm which is based on the idea of data compression.

First step. Given $x \in F$, compute all coordinates x_i of x. The cost of this step is $n \cdot \mathbf{c}_{\text{info}}$.

Second step. Compute $a_i = |x_i|$ for all $i = 1, 2, \ldots, n$. This can be done by using branching and arithmetic operations: if $x_i \ge 0$ then $a_i = x_i$ else $a_i = -x_i$. The cost is at most $n(\mathbf{c}_{\text{top}} + \mathbf{c}_{\text{ari}})$. Then compute n_0 indices of the set $I = \{i_1, i_2, \ldots i_{n_0}\}$ which correspond to the n_0 largest numbers among a_i, i.e.,

$$a_i \le a_{i_k} \quad \text{for } k = 1, 2 \ldots, n_0$$

for all i different from the indices i_k's. This can be done in cost proportional to

$$n \, \min\{n_0, \log_2 \, n\}(\mathbf{c}_{\text{top}} + \mathbf{c}_{\text{ari}}).$$

Third step. Observe that

$$|(S^n(x))_{i \cdot (i-1)/2 + j - 1}| = |x_i + x_j| \le \varepsilon$$

whenever both i and j are not in the set I.

Hence, we compute and output $(S^n(x))_k$ only for $k = i \cdot (i-1)/2 + j - 1$ for which at least one of i or j is in I. All other coordinates of $S^n(x)$ are ignored. The third step can be done in cost proportional to $n \cdot n_0(\mathbf{c}_{\text{out}} + \mathbf{c}_{\text{ari}})$.

Combining the cases $n \le n_0$ and $n > n_0$ we get a nonlinear algorithm whose error is at most ε and whose cost is bounded by

$$n \cdot \mathbf{c}_{\text{info}} + c \cdot n \cdot \min(n, \varepsilon^{-1})(\mathbf{c}_{\text{out}} + \mathbf{c}_{\text{ari}} + \mathbf{c}_{\text{top}}),$$

where c is a positive number which does not depend on n or ε.

We now compare the linear and total complexities for $0 < \varepsilon < 1$. Ignoring the constants, we have

$$\operatorname{comp}^{\text{lin}}(\varepsilon, S^n, F, \Lambda, \{g_i\}) \asymp n^2 \quad \text{and} \quad \operatorname{comp}(\varepsilon, S^n, F, \Lambda, \{g_i\}) \asymp n \cdot \min(n, \varepsilon^{-1}). \tag{44}$$

This means that nonlinear algorithms are much better than linear algorithms if n is much larger than ε^{-1}. We achieve a significant data compression by the use of a nonlinear algorithm which is impossible for linear algorithms.

Knowing the complexity of this linear problem we can also test the qualities of the lower bounds provided by Lemmas 1 and 2. Observe that for this problem the lower bound of Lemma 1 is of order n which is bad whereas the lower bound of Lemma 2 is of order $n \cdot \min(n, \varepsilon^{-1})$ which is very good. The latter can be proved exactly in the same way as above.

The results for this example will be used in the next (infinite dimensional) examples.

8. Infinite dimensional examples

We present infinite dimensional examples, based on the mappings S^n, see (35), from the previous section. Assume that a sequence $\{n_i\}$ of natural numbers and a sequence of weights $\{\gamma_i\}$ are given. We also assume

$$\gamma_1 \geq \gamma_2 \geq \cdots > 0 \quad \text{and} \quad \lim_{i\to\infty} \gamma_i = 0. \tag{45}$$

We then define S as a direct product of the S^{n_i},

$$S = \prod_{i=1}^{\infty} \gamma_i S^{n_i} : \prod_{i=1}^{\infty} \ell_1^{n_i} \to \prod_{i=1}^{\infty} \ell_\infty^{m_i}, \tag{46}$$

where $m_i = n_i(n_i - 1)/2$. Hence, if $x^i \in \ell_1^{n_i}$ for $i \in \mathbb{N}$ then

$$S([x^1, x^2, \dots]) = [\gamma_1 S^{n_1}(x^1), \gamma_2 S^{n_2}(x^2), \dots]. \tag{47}$$

For $x = [x^1, x^2, \dots]$, we define

$$\|x\| := \sup_{i\in\mathbb{N}} \|x^i\|_{\ell_1^{n_i}} \tag{48}$$

and define F as the unit ball in this norm, $F = \{x \mid \|x\| \leq 1\}$. Similarly, we define the norm $\|y\|$ in the image space by

$$\|y\| := \sup_{i\in\mathbb{N}} \|y^i\|_{\ell_\infty^{m_i}}. \tag{49}$$

From the definition of S and the norms, we easily conclude that solving the problem (S, F, ε) is as difficult as solving all the problems $(\gamma_i S^{n_i}, F_i, \varepsilon)$, for all $i \in \mathbb{N}$. Here, of course F_i, is the unit ball in $\ell_1^{n_i}$.

By Lemma 1 and the discussion in Section 7 we obtain the exact value of the information complexity. For $\gamma_{k+1} \leq \varepsilon < \gamma_k$ we get

$$\mathrm{comp}_{\,\mathrm{info}}(\varepsilon) = \mathbf{c}_{\mathrm{info}} \cdot (n_1 + \cdots + n_k). \tag{50}$$

Ignoring the constants which are proportional to $\mathbf{c}_{\mathrm{out}} + \mathbf{c}_{\mathrm{info}} + \mathbf{c}_{\mathrm{ari}} + \mathbf{c}_{\mathrm{top}}$ we also obtain

$$\mathrm{comp}\,(\varepsilon) = \sum_{i=1}^{k} \mathrm{comp}\,(\varepsilon/\gamma_i, S^{n_i}) \asymp \sum_{i=1}^{k} n_i \cdot \min(n_i, \gamma_i/\varepsilon) \tag{51}$$

and

$$\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon) = \sum_{i=1}^{k} \mathrm{comp}^{\,\mathrm{lin}}(\varepsilon/\gamma_i, S^{n_i}) \asymp \sum_{i=1}^{k} n_i^2. \tag{52}$$

To give a specific example we put

$$n_i = 2^i \quad \text{and} \quad \gamma_i = 2^{-i}. \tag{53}$$

For this example we obtain

$$\mathrm{comp}\,(\varepsilon) \asymp k \cdot 2^k \quad \text{or} \quad \mathrm{comp}\,(\varepsilon) \asymp -\varepsilon^{-1} \cdot \log \varepsilon \tag{54}$$

and

$$\mathrm{comp}^{\mathrm{lin}}(\varepsilon) \asymp 4^k \quad \text{or} \quad \mathrm{comp}^{\mathrm{lin}}(\varepsilon) \asymp \varepsilon^{-2}. \tag{55}$$

Hence, the linear complexity is much larger than the complexity. This shows that nonlinear algorithms are much more efficient than linear algorithms.

For optimal nonlinear algorithms we compute

$$\varphi(N(f)) = \sum_{k=1}^{n} c_k \cdot g_{i_k} \in G,$$

where the n, i_k and c_k depend on f. It is important that the choice of the elements g_{i_k} depends on f. It fact, it is easy to check that if we use the same g_{i_k} for all f then we cannot improve linear algorithms and even the best nonlinear algorithm with error ε would have cost proportional to ε^{-2} instead of $\varepsilon^{-1} \cdot \log(1/\varepsilon)$.

Observe that adaption does not help for this example. The information N of an optimal algorithm can be chosen linear, i.e., nonadaptive.

9. Tradeoff between information and total complexity

In this section we present an example of a linear problem for which there is a tradeoff between the information, and output and total complexity. More precisely, for any integer n we find an ε such that $\mathrm{comp}_{\mathrm{info}}(\varepsilon) = 1$, and the minimal output and total cost of any algorithm using this information of cardinality one is 2^n. On other hand, the output and total complexity are at most of order n.

Define

$$S^n : \ell_\infty^n \to \ell_\infty^m, \qquad m = 2^n, \tag{56}$$

between finite dimensional spaces, where

$$S^n(x) = \frac{1}{n}\left(\sum_{i=1}^{n} \pm x_i\right)_{j=1,\dots,m},$$

for all $m = 2^n$ possible distributions of the signs. To have a more formal definition, we put

$$S^n(x) = \frac{1}{n}\left(\sum_{i=1}^{n} (-1)^{j_i}\, x_i\right)_{j=1,\dots,m}, \tag{57}$$

where

$$j - 1 = \sum_{i=1}^{n} j_i\, 2^{i-1} \qquad \text{with} \quad j_i \in \{0,\, 1\}.$$

The simplest case is when

$$(n-1)/n \le \varepsilon < 1. \tag{58}$$

Then we have $\mathrm{comp}_{\mathrm{info}}(\varepsilon) = 1$. Indeed, it is clear that $\mathrm{comp}_{\mathrm{info}}(\varepsilon) > 0$ since without any information the problem cannot be solved. It is enough, however, to compute one of the x_i, say x_1. As an approximation of $S^n(x)$ we can take

$$\varphi \circ N(x) = \frac{1}{n}\Big((-1)^{j_1}\, x_1 \Big)_{j=1,\dots,m}, \tag{59}$$

and the error of this algorithm is $(n-1)/n \le \varepsilon$. Observe that the (total) cost of this algorithm is very high since we have to output all components,

$$\mathrm{cost}\,(\varphi \circ N) \approx \mathbf{c}_{\mathrm{info}} + 2^n \mathbf{c}_{\mathrm{out}}. \tag{60}$$

There is no algorithm, based on the same information, which is essentially cheaper. This can be seen as follows. Assume that $x_1 = 1$. Then $|(S^n(x)_j| = 1$ is possible for each coordinate of $S^n(x)$ and hence we have to output a number, different from zero, for each of the 2^n coordinates. Hence we have the lower bound $\mathrm{cost}\,(\varphi \circ N) \ge \mathbf{c}_{\mathrm{info}} + 2^n\, \mathbf{c}_{\mathrm{out}}$ for each algorithm which uses only one oracle call.

We now propose a much cheaper algorithm $\varphi \circ N$ which computes complete information and whose output cost is low,

$$\mathrm{cost}\,(\varphi \circ N) \approx n \cdot (\mathbf{c}_{\mathrm{info}} + \mathbf{c}_{\mathrm{ari}} + \mathbf{c}_{\mathrm{top}}) + 2\mathbf{c}_{\mathrm{out}}. \tag{61}$$

After computing the complete information we evaluate $a = (|x_1| + |x_2| + \cdots + |x_n|)/n$, and output only two coordinates $(S^n(x))_j$ of $S^n(x)$. Namely, the two coordinates a and $-a$ which correspond to all $(-1)^{j_i}\, x_i$ with the same (positive or negative) sign.

The cost of such an algorithm is $n \cdot \mathbf{c}_{\mathrm{info}}$ for the information plus arithmetic cost of order $n\mathbf{c}_{\mathrm{ari}}$, branching cost of the same order, and output cost $2\mathbf{c}_{\mathrm{out}}$. Hence the complexity of this problem is at most of the order n.

For the linear complexity we have

$$\mathrm{comp}^{\mathrm{lin}}(\varepsilon) \approx \mathbf{c}_{\mathrm{info}} + 2^n\, \mathbf{c}_{\mathrm{out}}\,, \tag{62}$$

since we have to output each coordinate, if we only allow linear algorithms, and we have to compute at least one information functional. In fact, the algorithm (59) achieves this bound.

Let now $\varepsilon \ge 0$ and $n \in \mathbb{N}$ be arbitrary. The case $\varepsilon \ge 1$ is trivial since we can take the zero algorithm and obtain

$$\mathrm{comp}\,(\varepsilon) = \mathrm{comp}_{\mathrm{info}}(\varepsilon) = \mathrm{comp}^{\mathrm{lin}}(\varepsilon) = 0. \tag{63}$$

Hence we assume

$$(\ell - 1)/n \le \varepsilon < \ell/n, \qquad \ell \in \mathbb{N},\ \ell \le n. \tag{64}$$

As already mentioned, for the linear complexity we have to output an approximation to each of the $m = 2^n$ coordinates, the cost is at least

$$\mathrm{comp}^{\mathrm{lin}}(\varepsilon) \ge 2^n\, \mathbf{c}_{\mathrm{out}}\,, \tag{65}$$

independently of $\varepsilon < 1$.

A completely naive way to compute the exact value of $S^n(x)$ would cost about $n \cdot 2^n$. It is not difficult, however, to improve this upper bound and obtain

$$\mathrm{comp}^{\,\mathrm{lin}}(\varepsilon) \le K \cdot 2^n \tag{66}$$

with $K > 0$ which does not depend on n or ε.

We now study the information complexity. As an approximation of $S^n(x)$ we can take

$$\varphi \circ N(x) = \frac{1}{n} \left(\sum_{i=1}^{n+1-\ell} (-1)^{j_i}\, x_i \right)_{j=1,\ldots,m}, \tag{67}$$

and this is optimal with respect to the used information. Hence the information complexity is given by

$$\mathrm{comp}_{\,\mathrm{info}}(\varepsilon) = (n+1-\ell) \cdot \mathbf{c}_{\mathrm{info}} \approx n\,(1-\varepsilon) \cdot \mathbf{c}_{\mathrm{info}}. \tag{68}$$

We now provide an upper bound for the (total) complexity. We suggest an algorithm which is much better than every linear algorithm, for given $\varepsilon > 0$ and large n.

Even if ε is relatively large then we compute complete information. Then we sort the x_i. Without loss of generality we may assume that

$$1 \ge x_1 \ge x_2 \ge \cdots \ge x_n \ge 0. \tag{69}$$

The sorting and the respective computations on indices cost of the order $n \log n$. This is a little more than (61) but still it is much less than the cost of optimal linear algorithms. This is the first step of the algorithm. Due to (69) the second step of the algorithm will be linear. (Hence the complete algorithm will be "piecewise linear".)

We denote the set of all $x \in \mathbb{R}^n$ satisfying (69) by X_n. Observe that X_n is a convex set with $(n+1)$ extremal points, all of the form $(1,\ldots,1,0,\ldots,0)$. Assume now that an arbitrary sequence of signs

$$k = (k_1, \ldots, k_n) \in \{-1,\, 1\}^n$$

is given. Consider now the linear functional

$$L_k(x) = \frac{1}{n} \sum_{i=1}^{n} k_i\, x_i.$$

Define $\min(k)$ by

$$\min(k) := \min \left\{ \frac{1}{n} \sum_{i=1}^{j} k_i \mid j = 1, \ldots, n \right\}$$

and define $\max(k)$ by

$$\max(k) := \max \left\{ \frac{1}{n} \sum_{i=1}^{j} k_i \mid j = 1, \ldots, n \right\}.$$

Due to the geometrical structure of X_n we easily conclude that

$$\{L_k(x) \mid x \in X_n\} = [\min(k),\, \max(k)].$$

For the given ℓ, we precompute all $k \in \{-1, 1\}^n$ such that

$$\max\{-\min(k),\ \max(k)\} \geq \ell/n.$$

We denote by N_ℓ the number of such k's. This is not really part of the algorithm, so there is no cost of this step. Now we compute and output the coordinates of $S^n(x)$ which correspond to all such k's. The total cost of this algorithm is bounded by

$$\text{cost}\,(\varphi \circ N) \leq K \cdot (n \log n + n \cdot N_\ell). \tag{70}$$

Using a well known result, see Feller (1957, Ch. 3, Sec. 4), we obtain

$$N_\ell \leq 2 \cdot \sum_{m=\ell}^{n} \frac{\ell}{m} \binom{m}{\frac{m+\ell}{2}} 2^{n-m}.$$

The sum is over all m such that $m + \ell$ is even or we put $\binom{m}{r} = 0$ if r is not an integer. For a fixed $\varepsilon \approx \ell/n$ and large n, the upper bound (70) is much smaller then the cost of optimal linear algorithms.

We could use this example, as in Section 8, to construct infinite dimensional examples. We omit this construction since it does not lead into new insights.

Acknowledgments. We are grateful to Joseph F. Traub for valuable comments on our paper.

References

1. N. S. Bakhvalov, On the optimality of linear methods for operator approximation in convex classes of functions, USSR Comput. Maths. Math. Phys., 11 (1971), pp. 244–249.

2. L. Blum, M. Shub and S.Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, Bull. of the AMS 21 (1989), pp. 1–46.

3. L. Blum, F. Cucker, M. Shub and S. Smale, Complexity and Real Computation, Springer, New York, 1998.

4. S. Dahlke, W. Dahmen and R. A. DeVore, Nonlinear approximation and adaptive techniques for solving elliptic operator equations, In: Multiscale Wavelet Methods for PDEs, W. Dahmen, A. J. Kurdila, P. Oswald (eds.), Academic Press, San Diego, 1997, pp. 237–283,

5. W. Dahmen, Wavelet and multiscale methods for operator equations, Acta Numerica **6** (1997), pp. 55–228.

6. R. A. DeVore, Nonlinear approximation, Acta Numerica **7** (1998), pp. 51–150.

7. E. G. D'yakonov, Optimization in Solving Elliptic Problems, CRC Press, Boca Raton, 1996.

8. W. Feller, An Introduction to Probability Theory and its Applications, Volume 1, Second Edition, Wiley, New York, 1957.

9. S. Gal, C. A. Micchelli, Optimal sequential and non-sequential procedures for evaluating a functional, Appl. Anal., 10 (1980), pp. 105–120.

10. W. Hackbusch, Multi-grid Methods and Applications, Springer-Verlag, New York, 1985.

11. P. Hertling, Topological complexity with continuous operations, J. Complexity 12 (1996), pp. 315-338.

12. P. Mathé, s-numbers in information-based complexity, J. Complexity 6 (1990), pp. 41–66.

13. E. Novak, The real number model in numerical analysis, J. Complexity 11 (1995), pp. 57–73.

14. E. Novak, On the power of adaption, J. Complexity 12 (1996), pp. 199–237.

15. E. Novak, H. Woźniakowski, Topological complexity of zero finding, J. Complexity 12 (1996), pp. 380–400.

16. E. Novak, H. Woźniakowski, On the cost of uniform and nonuniform algorithms, 1997, to appear in Theoretical Computer Science.

17. S. Smale, On the topology of algorithms, J. Complexity 3 (1987), pp. 81-89.

18. J. F. Traub and H. Woźniakowski, A General Theory of Optimal Algorithms, Academic Press, New York, 1980.

19. J. F. Traub, G. W. Wasilkowski, and H. Woźniakowski, Information-Based Complexity, Academic Press, Boston, 1988.

20. J. F. Traub, A. G. Werschulz, Complexity and Information, Cambridge University Press, 1998.

21. V. A. Vassiliev, Topological complexity of root-finding algorithms, in: The Mathematics of Numerical Analysis, J. Renegar, M. Shub, S. Smale (eds.), Lectures in Applied Mathematics 32, AMS, 1996, pp. 831-856.

22. A. G. Werschulz and H. Woźniakowski, Are linear algorithms always good for linear problems?, Aequationes Math. 31 (1986), pp. 202–212.

23. H. Woźniakowski, Why does information-bause the real number model?, 1997, to appear in Theoretical Computer Science.

Mathematisches Institut, Universität Erlangen-Nürnberg, Bismarckstrasse 1 1/2, 91054 Erlangen, Germany

E-mail address: novak@mi.uni-erlangen.de

Department of Computer Science, Columbia University, New York 10027

E-mail address: henryk@cs.columbia.edu