On the Solvability Complexity Index
-
Spectra, Inverse Problems and Error Control

Bachelor Thesis
by

Sascha Hauch

Fakultät II - Mathematik und Naturwissenschaften
Technischen Universität Berlin

Examiner:
Prof. Dr. Peter Bürgisser

Berlin 2018

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 12. November 2018

.....
Unterschrift

Zusammenfassung

In der vorliegenden Arbeit beschäftigen wir uns mit dem Solvability Complexity Index. Dabei handelt es sich um ein Konzept von Anders Hansen, dass Berechnungsprobleme, die nicht in endlicher Zeit lösbar sind, danach klassifiziert, wie viele Limites benötigt werden, um sie zu lösen.
Zunächst bauen wir das theoretische Grundgerüst auf das wir benötigen, wenn wir über den Solvability Complexity Index sprechen wollen und untersuchen in ersten Beispielen die Komplexität konkreter Probleme.
Anschließend wenden wir diese Techniken auf Fragestellungen der Funktionalanalysis an: zunächst bei der Berechnung von Spektren beschränkter, selbstadjungierter, orthogonaler, positiv semidefiniter und kompakter Operatoren. Hiernach widmen wir uns der Berechnung von Lösungen wohlgestellter inverser Probleme ebenfalls mit den vorher genannten Operatoren sowie Dreiecksoperatoren.
Letztendlich erörtern wir noch das Thema der Fehlerkontrolle, erzielen abstrakte Resultate darüber, wann Fehlerkontrolle möglich ist, und Übertragen dieses Wissen auf die zuvor diskutierten konkreten Probleme.

Contents

Introduction

For finite binary computations, there is a well established theory initialised by TURING in [8] by the first definition of the so called TURING machine. The corresponding very successful complexity theory was then driven by COOKS result on the NP-completeness of SAT in [3] culminating in the most famous problem in theoretical computer science: P versus NP. These are just two of a whole bunch of important complexity classes, ranging from LOGSPACE over the polynomial hierarchy up to EXPSPACE (and even more) sorting nearly every finitely solvable problem into a proper class of problems having a similar complexity.

If we want to extend the finite computations to (vectors of) real numbers, there are different approaches. One way is to use the BSS machine developed by BLUM, SHUB and SMALE in [2] which also generalises the concept of NP-completeness and the P versus NP question.
Another approach is computable analysis due to WEIHRAUCH (see for example [9]). It even allows to analyse computations using sets of (vectors of) real numbers. Furthermore, one is able to speak about the complexity of infinite computations: For an algorithm which computes approximations whose limit solves the given problem one measures the time needed for a prescribed precision. Unfortunately there are still several drawbacks of this theory.

First, there are problems which cannot be solved using one approximation and limit process, but using a number of these such as polynomial root finding with rational maps, analysed by DOYLE and MCMULLEN in [4], or computation of spectra of operators as we see in Chapter 2. Secondly, those problems cannot be classified in terms of the effort for an approximation up to a prescribed precision, since they do not allow error control as we see in Chapter 3.

In 2011, a new approach was developed by HANSEN in [7] called the *Solvability Complexity Index (SCI)* - first for spectral problems, but soon generalised to almost arbitrary computational problems in [1]. With this general theory one is able to attack the question about the complexity of problems, which deal with real numbers and are not solvable in finite time, by counting how many limits are needed to compute the desired solution. This way we avoid such problems as above and are able to treat computational problems independent of the computation model.

In order to make this precise, in Chapter 1 of this thesis we learn about the general framework of computational problems, general algorithms and towers of algorithms. Finally, we are able to state what the Solvability Complexity Index is. Along this way we see first examples of computational problems as the halting problem and get an impression how the presented techniques work.

Afterwards, in the next chapter, we examine the complexity of computing spectra and apply the tools from Chapter 1. Encountering many negative results about the hardness of this task peak in Theorem 2.4 which states, that even self-adjoint unitary operators require two limits. Another issue we have a look at are inverse problems. As in the first case, we encounter fundamental complexity barriers in solving those - at least if we allow our problems to include general inputs. To lower these bounds we then look at more restricted subclasses which include more structure. This search is inspired by [1, Rem.6.5(ii)], where the question of existence and shape of subclasses providing error control (see Chapter 3) was

raised. As a step in this direction, we extend the results from [1] by proving or disproving approximability of such restricted problems with one limit. For that, we are inspired by the finite dimensional case of matrices, where the solution of inverse problems involving positive semidefinite, unitary or triangular matrices is comparatively easy. Theorem 2.9 then shows that the first property is not sufficient to lower the SCI. Thankfully it looks different concerning the remaining two as Theorem 2.10 and 2.11 verify.

For practise, one of the most important features of algorithms is to provide error control, which is the topic of Chapter 3. Here, we see several negative results such as Theorem 3.2 stating that error control is never possible if we deal with a computational problem which requires two limits for its solution and Theorem 3.8 showing that even very well structured operators as projections do not allow error control while computing their spectrum despite having an SCI of one. Reviewing the results from Chapter 2 which have SCI less equal to one is again a step to answer the question mentioned above, showing how restrictive error control is and why it might by favourable to look for weaker notions fitting more to what intuitively should be easy.

In the last chapter we draw some short conclusions and have an outlook on what results or topics might be tackled in the future especially in the area of error control.

Throughout this thesis we follow the paper [1] concerning the definitions and existing results, but also make some adaptations for simplification. Own contributions are given by the proofs of theorems 2.1 and 2.7 and by the results and proofs of the theorems 2.4, 2.9, 2.10, 2.11, 3.7, 3.8, and 3.10.

Notation

In this thesis we use the following notation:

$\mathbb{N}$	the natural numbers starting at 1
$\mathrm{Map}(\Omega, \mathbb{C})$	the set of all maps $\Omega \to \mathbb{C}$
$\mathfrak{P}(X)$	the power set of a given set X
$\mathfrak{Re}(x)$	the real part of a complex number $x \in \mathbb{C}$
$\mathfrak{Im}(x)$	the imaginary part of a complex number $x \in \mathbb{C}$
δ_x	the KRONECKER-Delta symbol of x, which means $\delta_x(y) = 1$ if $x = y$ and $\delta_x(y) = 0$ otherwise
$B(x, r)$	the open ball of radius $r \in \mathbb{R}^+$ around $x \in \mathcal{M}$ for some metric space $\mathcal{M}$
$L(X)$	the set of all continuous and linear operators on a BANACH space X
$K(X)$	the space of compact linear operators $T\colon X \to X$ for a BANACH space X, where T being compact, means that T maps bounded sets to relatively compact sets
$\sigma(A)$	the spectrum of the operator $A \in L(X)$ or of the matrix $A \in \mathbb{C}^{n\times n}$ respectively
$N(p)$	the zero set of a polynomial p over $\mathbb{C}$
$\ell^2(\mathbb{N})$	the HILBERT space of all square summable sequences with entries in $\mathbb{C}$ equipped with the usual inner product $\langle x, y\rangle = \sum_{k\in\mathbb{N}} x_k\overline{y_k}$
$(e_n)_{n\in\mathbb{N}}$	an orthonormal basis of a separable HILBERT space $\mathcal{H}$; if $\mathcal{H} = \ell^2(\mathbb{N})$ we refer to the standard basis of unit vectors
P_{e_k}	the projection onto the span of the k-th basis vector, $k \in \mathbb{N}$ for an orthonormal basis $(e_n)_{n\in\mathbb{N}} \subseteq \mathcal{H}$; we set $P_k = \sum_{i=1}^{k} P_{e_i}$
$a_i, a_{i,j}$	an entry of a sequence a or a matrix A respectively
$\bigoplus_{i=1}^{\infty} A_i, \oplus_{i=1}^{\infty} a^{(i)}$	for matrices $A_i \in \mathbb{C}^{n_i\times n_i}$ let $A := \bigoplus_{i=1}^{\infty} A_i$ be the linear operator on $\prod_{i=1}^{\infty} \mathbb{C}^{n_i} \cong \prod_{n\in\mathbb{N}} \mathbb{C}$ satisfying $(Ax)_i = A_i x_i$ for $x = (x_i)_{i\in\mathbb{N}} \in \prod_{i=1}^{\infty} \mathbb{C}^{n_i}$; for vectors $a^{(i)} \in \mathbb{C}^{n_i}$, let $a := \oplus_{i=1}^{\infty} a^{(i)}$ be the sequence in $\prod_{i=1}^{\infty} \mathbb{C}^{n_i} \cong \prod_{n\in\mathbb{N}} \mathbb{C}$ satisfying $a_{k+j} = a_j^{(i)}$ for $k = \sum_{l=1}^{i-1} n_l$ and $j = 1, \ldots, n_i$; we apply obvious adaptations for finite sums $\bigoplus_{i=1}^{n} A_i$ and $\oplus_{i=1}^{\infty} a^{(n)}$; in this case we allow A_n to be a linear operator on $\prod_{n\in\mathbb{N}} \mathbb{C}$ and $a^{(n)} \in \prod_{n\in\mathbb{N}} \mathbb{C}$

1 General Algorithms

In this chapter we are concerned with the task to build up a general framework for computational problems and its classification via the Solvability Complexity Index. In order to do so we follow [1] and start with the definition of a computational problem. We want to emphasise that this does not include any condition on the solvability of the problem. That means there might be problems for which no algorithms or more general concepts of solutions, in the sense of Definition 1.3 and 1.7, exist.

Definition 1.1. Let Ω be any set, called primary set, $\Lambda \subseteq \mathrm{Map}(\Omega, \mathbb{C})$ a set of complex valued functions on Ω, called the evaluation set, $\mathcal{M} = (M, d)$ a metric space and $\Xi\colon \Omega \to \mathcal{M}$, called the problem function. Then we refer to $(\Omega, \Lambda, \mathcal{M}, \Xi)$ as a computational problem.

As the next examples (and chapters) show, this definition is general enough to capture a variety of problems. However, one may consider even more general notions, for example where $\mathcal{M}$ is just a pseudo metric space, which needs to be considered for computing zeros of polynomials using rational maps.[1] But for our purposes it suffices to consider this version.

Example 1.2.

(i) An easy example of a spectral problem is the following: Let Ω be the set of all 2×2 matrices $A \in \mathbb{C}^{2\times 2}$, $\Lambda := \{f_{i,j} \mid i, j = 1, 2\}$, where $f_{i,j}$ outputs the (i, j)-entry of a given matrix A, and $\mathcal{M}$ the set of all compact subsets of $\mathbb{C}$ equipped with the HAUSDORFF metric. As a problem function we define $\Xi\colon \Omega \to \mathcal{M}$ by $\Xi(A) = \sigma(A)$ the spectrum of a given matrix A. Then $P_0 := (\Omega, \Lambda, \mathcal{M}, \Xi)$ is a computational problem.

(ii) We can also speak of decision problems. Let Ω be the set of all functions $a\colon \mathbb{N} \to \mathbb{C}$, in other words the sequences $a = (a_n)_{n\in\mathbb{N}}$ in $\mathbb{C}$. For Λ we take the evaluations $f_n\colon a \mapsto a_n$, for $n \in \mathbb{N}$. As metric space $\mathcal{M}$ we use $\{0, 1\}$ equipped with the discrete metric. Then we consider the two problem functions $\Xi_1, \Xi_2\colon \Omega \to \{0, 1\}$ where $\Xi_1(a) = 1$ if and only if $a \in \Omega$ contains a zero and $\Xi_2(a) = 1$ if and only if a contains infinitely many zeros. This way we obtain the two computational problems $P_1 := (\Omega, \Lambda, \mathcal{M}, \Xi_1)$ and $P_2 := (\Omega, \Lambda, \mathcal{M}, \Xi_2)$. This example is inspired by the first examples given in [1, Chap.7].

(iii) We can also talk about the famous halting problem. For this let Ω be the set of all pairs (M, x) of TURING machines and input words with input alphabet $\Sigma = \{0, 1\}$, a tape being infinite in one direction, tape alphabet $\Gamma = \{0, 1, -1\}$, a finite set of states $Q \subseteq \mathbb{N} \setminus \{1\}$ containing 2 as initial state, 3 as accepting final state, and a transition function $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$. Further let the set of evaluations be $\Lambda = \{f_{n,k,letter}, f_{n,head}, f_{n,end} \mid n, k \in \mathbb{N}\}$, where $f_{n,k,letter}(M, x)$ gives k-the tape entry, $f_{n,head}(M, x)$ the position of the head in the n-th step in the computation of M on x and $f_{n,end}(M, x)$ outputs 1 or 0 if M holds in the n-th step or not. As metric space $\mathcal{M}$ we choose again $\{0, 1\}$ equipped with the discrete metric and the problem function $\Xi\colon \Omega \to \mathcal{M}$ we define to fulfil $\Xi(M, x) = 1$ if and only if M halts on x. Defining $P_3 := (\Omega, \Lambda, \mathcal{M}, \Xi)$ we obtain a version of the halting problem.

[1] See also [1, Ex.2.1(iii)].

To solve computational problems we need algorithms. One of the most general definitions of an algorithm is the following, which only requires just to use a finite amount of information from the input while computing the output.

Definition 1.3. Let Ω and X be any sets and $\Lambda \subseteq \mathrm{Map}(\Omega, \mathbb{C})$. We define a general (adaptive) algorithm on Ω, with the evaluations Λ and outputs in X as a map $\Gamma\colon \Omega \to X$ with the following property: For all $A \in \Omega$ there is a finite set $\Lambda_\Gamma(A) \subseteq \Lambda$ such that for all $B \in \Omega$ the condition $f(A) = f(B)$ for all $f \in \Lambda_\Gamma(A)$ implies $\Gamma(A) = \Gamma(B)$.

This is based on Definition 2.3 from [1] but slightly adjusted in two ways. Mainly we merged the three conditions on a general algorithm from [1] into one logically equivalent statement for simplification and clarification. Further, when working with general algorithms, in most cases Ω is the primary set and X the metric space of a computational problem.[2] But since the presence of a computational problem is not necessary for talking about an algorithm, we did not incorporate it in the definition. This is also related to the fact that so far there is no connection between an algorithm and the problem function of a given computational problem, which we want to solve.

General algorithms in the sense of Definition 1.3 are allowed to choose adaptively to the input, which amount of information to use within the computation, which is one aspect why they are very powerful. For example there might be a case distinction on a specific key information acquired from every input in the beginning and a calculation based on that distinction afterwards.[3]
On the other hand it might be interesting which computations are possible to do non-adaptively, which is more restrictive. In the following, we therefore speak of adaptive and non-adaptive algorithms.

Definition 1.4. Let Ω and X be any sets, $\Lambda \subseteq \mathrm{Map}(\Omega, \mathbb{C})$ and $\Gamma\colon \Omega \to X$ a general algorithm. We call Γ non-adaptive if $\Lambda_\Gamma(A) = \Lambda_\Gamma(B)$ for all $A, B \in \Omega$ is a possible choice to fulfil the condition of Definition 1.3.

Another aspect of the power of general algorithms is that we do not prescribe which operations they are allowed to use. Intuitively speaking, they can use every mathematical tool in the world to determine the desired answer. With the following definition we obtain algorithms which are more suitable for practical applications.

Definition 1.5. Let Ω and X be any sets, $\Lambda \subseteq \mathrm{Map}(\Omega, \mathbb{C})$ and $\Gamma\colon \Omega \to X$ a general algorithm. We call Γ arithmetic if for each $A \in \Omega$ as input, Γ only performs finitely many arithmetic operations on the set

$$\mathcal{D} := \{f(A), \mathfrak{Re}(f(A)), \mathfrak{Im}(f(A)) \mid f \in \Lambda_\Gamma(A)\}$$

and comparisons of real numbers that appear in the computation. If we extend $\mathcal{D}$ by

$$\left\{ \sqrt[n]{f(A)} \mid f \in \Lambda_\Gamma(A), n \in \mathbb{N} \right\},$$

where for complex numbers we include every n-th root, we call Γ radical.

One might argue that the last definition contains some kind of black box since it is not clear what Γ *only performs finitely many arithmetic operations and comparisons* means and

[2] The only exception in this thesis is the sequence $\{\hat{\Gamma}_n\}_{n\in\mathbb{N}}$ of general algorithms in Definition 3.3 and Theorem 3.5.
[3] See also the proof of Theorem 2.1.

involves and what the output looks like. We do not prescribe the shape of the output but the allowed mathematical tools to include as many cases as possible and treat them altogether. Three different examples of outputs in which we are particularly interested are:

- $\Gamma(A) \in \mathrm{Alg}(\mathcal{D})$
- $\Gamma(A) \in \mathrm{Alg}(\mathcal{D})^m$ for some $m \in \mathbb{N}$
- $\Gamma(A) \subseteq \mathrm{Alg}(\mathcal{D})$,

where $\mathrm{Alg}(\mathcal{D})$ denotes the smallest algebraic number field containing $\mathcal{D}$.

In the sense of Definition 1.3 for example TURING machines which halt on every input are so called general algorithms. To have some more impressions, we review our last example about computational problems and see what algorithms we can speak of in the given context.

Example 1.6.

(i) In the case of P_0 from Example 1.2 we can solve it using a radical algorithm: For a matrix $A = (a_{i,j})_{i,j} \in \Omega = \mathbb{C}^{2\times 2}$ the characteristic polynomial is given by $\det(xI_2 - A) = x^2 - (a_{1,1} + a_{2,2})x + \det(A)$ and with the pq-formula its zeros are

$$x_\pm := \frac{a_{1,1} + a_{1,2}}{2} \pm \sqrt{\left(\frac{a_{1,1} + a_{1,2}}{2}\right)^2 - \det(A)}.$$

Then we define $\Gamma\colon \Omega \to \mathcal{M}$ by $\Gamma(A) := \{x_+, x_-\}$ and obtain a non-adaptive radical algorithm, since its operations only consist of a finite application of arithmetic and radical computations, dependent on $\{a_{i,j} \mid 1 \le i,j \le 2\} = \{f_{i,j}(A) \mid 1 \le i,j \le 2\}$ so we can set $\Lambda_\Gamma(A) = \{f_{i,j} \mid 1 \le i,j \le 2\}$. Moreover we have $\Gamma = \Xi$.

(ii) We consider the computational problems P_1 and P_2 from Example 1.2. Let $m \in \mathbb{N}$. Then we define $\Gamma^1_m\colon \Omega \to \{0,1\}$ by

$$\Gamma^1_m((a_n)_{n\in\mathbb{N}}) = \begin{cases} 1, & a_n = 0 \text{ for at least one } n \in \mathbb{N} \text{ with } 1 \le n \le m \\ 0, & \text{else.} \end{cases}$$

This way we obtain an arithmetic algorithm, since the action of Γ^1_m on $a = (a_n)_{n\in\mathbb{N}} \in \Omega$ only depends on $\{f_n(a) \mid 1 \le n \le m\}$ by performing m comparisons and outputting 1 or 0. In particular Γ^1_m is non-adaptive.

Next, for $m_1, m_2 \in \mathbb{N}$, we define $\Gamma^2_{m_1,m_2}\colon \Omega \to \{0,1\}$ by

$$\Gamma^2_{m_1,m_2}(a) = \begin{cases} 1, & \text{there are at least } m_1 \text{ zeros in the first } m_2 \text{ entries of } a \\ 0, & \text{else.} \end{cases}$$

As above $\Gamma^2_{m_1,m_2}$ is an arithmetic non-adaptive algorithm.

(iii) For the halting problem P_3 we define an algorithm Γ_n, for $n \in \mathbb{N}$, to check whether a given TURING machine halts on an input x after n steps simply by setting $\Gamma_n(M,x) = f_{n,end}(M,x)$. Then $\Gamma_n\colon \Omega \to \mathcal{M}$ is a non-adaptive arithmetic algorithm.

We emphasise that algorithms so far do not have to do anything meaningful. If we want to solve a computational problem, we are looking for an algorithm outputting the same as the problem function Ξ, as in Example 1.6(i). But in general this is not possible. A

typical follow-up question is whether it is possible to compute approximations such that for $n \to \infty$ they converge to the true answer. As we will see, this is not always the case either. Instead, many computational problems require to take several approximations and limits into account in order to compute the solution. The next definition formalises this concept.

Definition 1.7. Let $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ be a computational problem and $k \in \mathbb{N}$. A general tower of algorithms for P of height k is a sequence

$$\{\Gamma_{n_1,\dots,n_k}\}_{(n_1,\dots,n_k)\in\mathbb{N}^k}$$

of general algorithms $\Omega \to \mathcal{M}$ indexed by $\mathbb{N}^k$, such that for all $A \in \Omega$ we have

$$\lim_{n_1\to\infty} \dots \lim_{n_m\to\infty} \Gamma_{n_1,\dots,n_m}(A) = \Xi(A) \tag{1}$$

with respect to the metric of $\mathcal{M}$. A tower of height zero for P is a general algorithm Γ satisfying $\Gamma = \Xi$. If the algorithms (or the algorithm, in case of height zero) are arithmetic or radical, we call the tower arithmetic or radical, respectively.

One now might ask why this sequence of algorithms is called a tower. Probably, this is just due to the way Anders Hansen visualise equation (1) since his first paper [7] on the SCI. There he separated each limit process and stacked one above the other such that it looked like a tower.

Besides *general*, *arithmetic* and *radical* there are even more types of algorithms and towers. The Kleene-Schoenfield towers appear when speaking about connections of the theory of the Solvability Complexity Index to computability and the arithmetical hierarchy. The Doyle-McMullen towers are part of the answer of the question if it is possible to compute polynomial zeros with one iterative generally convergent procedure. Both are treated in [1].

Looking at our examples again, we realise that we have already constructed some towers of algorithms.

Example 1.8.

(i) In Example 1.6 we already constructed a tower of height zero for P_0 since Γ was equal to Ξ.

(ii) We consider P_1 and the collection $\{\Gamma^1_m\}_{m\in\mathbb{N}}$ of arithmetic algorithms from Example 1.6. Then, for all $a \in \Omega$ we have $\lim_{m\to\infty} \Gamma^1_m(a) = \Xi_1(A)$: If a contains a zero, then there is an $n \in \mathbb{N}$ such that $a_n = 0$, hence $\Gamma^1_m(a) = 1 = \Xi_1(a)$ for all $m \geq n$ and if a does not contain any zero, we have $\Gamma^1_m(a) = 0 = \Xi_1(a)$ for all $m \in \mathbb{N}$. Thus $\{\Gamma^1_m\}_{m\in\mathbb{N}}$ is an arithmetic tower of algorithms of height one for P_1.

Considering P_2 and the collection of arithmetic algorithms $\{\Gamma^2_{m_1,m_2}\}_{(m_1,m_2)\in\mathbb{N}^2}$, we obtain

$$\lim_{m_2\to\infty} \Gamma^2_{m_1,m_2}(a) = \begin{cases} 1, & a \text{ contains at least } m_1 \text{ zeros} \\ 0, & \text{else} \end{cases}$$

and hence $\lim_{m_1\to\infty} \lim_{m_2\to\infty} \Gamma^2_{m_1,m_2}(a) = \Xi_2(a)$ for all $a \in \Omega$. Thus $\{\Gamma^2_{m_1,m_2}\}_{(m_1,m_2)\in\mathbb{N}^2}$ is an arithmetic tower of algorithms of height two for P_2.

(iii) By definition of the halting of a Turing machine the sequence $\{\Gamma_n\}_{n\in\mathbb{N}}$ from Example 1.6 (iii) is an arithmetic tower of algorithms for P_3 of height 1.

Now, with the tools at hand we developed so far, we are able to define the Solvability Complexity Index of a computational problem, which is nothing but the smallest number of limits with which we are able to solve the problem. In the language of towers of algorithms, we have the following:

Definition 1.9. Let $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ be a computational problem. Then we define the Solvability Complexity Index $\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_\alpha$ of P with respect to the algorithm type $\alpha \in \{G := \text{general}, A := \text{arithmetic}, R := \text{radical}\}$ as the smallest number $k \in \mathbb{N}$ such that there exists a tower of algorithms of height k and type α for P. If there is no such tower regardless of the height, we set $\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_\alpha := \infty$.

From this definition we can directly derive an estimate concerning different types of algorithms.

Remark 1.10. For every computational problem P we have $\mathrm{SCI}(P)_G \leq \mathrm{SCI}(P)_R \leq \mathrm{SCI}(P)_A$ since every arithmetic algorithm is radical and every radical one is general in turn. In general, equality as well as strict inequality is possible.[4]

Now, we have a last look at our examples and see what complexity indices they have:

Example 1.11.

(i) Let P_0 be as before. Since $\mathrm{SCI}(P)\alpha \geq 0$ for every computational problem and type of algorithms α, we obtain $\mathrm{SCI}(P_0)_G = \mathrm{SCI}(P_0)_R = 0$ by the last remark and our results from the previous examples. By Theorem 2.1 about computing spectra of finite matrices we also obtain $\mathrm{SCI}(P_0)_A \leq 1$. To prove also $\mathrm{SCI}(P_0)_A \geq 1$, assume there was an arithmetic algorithm $\Gamma = \Xi$. Then we consider the matrix

$$A := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

and conclude $\{i, -i\} = \sigma(A) = \Xi(A) = \Gamma(A)$ contradicting $i \notin \mathbb{Q}$.

(ii) We consider the computational problems P_1 and P_2 from before. By our construction of Example 1.8 we have $\mathrm{SCI}(P_1)_A \leq 1$ and $\mathrm{SCI}(P_2)_A \leq 2$. We will show that both are in fact equalities via proving $\mathrm{SCI}(P_1)_G \geq 1$ and $\mathrm{SCI}(P_2)_G \geq 2$.

For the first assume there was a general algorithm $\Gamma\colon \Omega \to \mathcal{M}$ satisfying $\Gamma = \Xi_1$. Then we consider the sequence $a := (a_n)_{n\in\mathbb{N}} := (1)_{n\in\mathbb{N}} \in \Omega$ and let N be an index for which Γ only takes the items a_n for $n \leq N$ into account while computing the output, which means $\Lambda_\Gamma(a) \subseteq \{f_n \mid n \leq N\}$. We define $b := (b_n)_{n\in\mathbb{N}} \in \Omega$ by $b_n := 1$ for all $n \leq N$ and $b_n := 0$ for $n > N$. Then we have $f(b) = f(a)$ for all $f \in \Lambda_\Gamma(a)$ and thus $\Gamma(a) = \Gamma(b)$ by Definition 1.3 of a general algorithm. This contradicts $\Gamma = \Xi_1$ since $\Xi_1(a) = 0 \neq 1 = \Xi_1(b)$. Thus we obtain $\mathrm{SCI}(P_1)_G = \mathrm{SCI}(P_1)_R = \mathrm{SCI}(P_1)_A = 1$.

For the second we exclude $\mathrm{SCI}(P_2)_G = 0$ with the same counterexample as for the first. To disprove $\mathrm{SCI}(P_2)_G = 1$ we assume the existence of a general tower of algorithms $\{\Gamma_m\}_{m\in\mathbb{N}}$ of height one for P_2. By induction, we will construct sequences $(m_k)_{k\in\mathbb{N}} \subseteq (m)_{m\in\mathbb{N}}$, $(n_k)_{k\in\mathbb{N}} \subseteq \mathbb{N}$ and elements $a^{(k)} \in \mathbb{C}^{n_k}$ such that for $a = \oplus_{k\in\mathbb{N}} a^{(k)} \in \Omega$ we have $\Gamma_{m_k}(a) = \delta_0(k \bmod 2)$ and hence no convergence of $\{\Gamma_m(a)\}_{m\in\mathbb{N}}$.

To this end let $b = (1)_{n\in\mathbb{N}}$ and $c = (0)_{n\in\mathbb{N}}$. Since by assumption $\Gamma_m(b) \to \Xi_2(b) = 0$ as $m \to \infty$ there is an index $m_1 \in \mathbb{N}$ for which $\Gamma_{m_1}(b) = 0$ holds. Since Γ_{m_1} is a general algorithm there is an index $n_1 \in \mathbb{N}$ such that Γ_{m_1} only takes the entries b_i for $i \leq n_1$

[4] See also Example 1.11 and Theorem 2.1.

into account while computing the output, which means $\Lambda_{\Gamma_{m_1}}(b) \subseteq \{f_i \mid i \leq n_1\}$. We set $a^{(1)} := (1)_{n=1}^{n_1} \in \mathbb{C}^{n_1}$. Now assume we already have constructed m_i, n_i and $a^{(i)}$ for $i = 1, \ldots, l$. For $l \bmod 2 = 1$ we consider the sequence $d := \left(\oplus_{k=1}^{l} a^{(k)}\right) \oplus c \in \Omega$ and observe $\Xi_2(d) = 1$. Hence there are indices $m_{l+1} > m_l$ and $s > \sum_{i=1}^{l} n_i$ such that $\Gamma_{m_{l+1}}(d) = \Xi(d) = 1$ and $\Lambda_{\Gamma_{m_{l+1}}}(d) \subseteq \{f_i \mid i \leq s\}$. Finally we define $n_{l+1} := s - \sum_{i=1}^{l} n_i$ and $a^{(l+1)} := (0)_{n=1}^{n_{l+1}} \in \mathbb{C}^{n_{l+1}}$. Similarly, for $l \bmod 2 = 0$ we consider the sequence $e := \left(\oplus_{k=1}^{l} a^{(k)}\right) \oplus b \in \Omega$ with $\Xi(e) = 0$ and choose m_{l+1}, s, n_{l+1} as before and set $a^{(l+1)} := (1)_{n=1}^{n_{l+1}}$.

It remains to prove $\Gamma_{m_k}(a) = \delta_0(k \bmod 2)$ for all $k \in \mathbb{N}$. Therefore let $k \in \mathbb{N}$. Then by construction we have

$$f(a) = \begin{cases} f\left(\left(\oplus_{i=1}^{k} a^{(i)}\right) \oplus c\right), & \text{if } k \bmod 2 = 0 \text{ and } f \in \Lambda_{\Gamma_{m_k}}\left(\left(\oplus_{i=1}^{k} a^{(i)}\right) \oplus c\right), \\ f\left(\left(\oplus_{i=1}^{k} a^{(i)}\right) \oplus b\right), & \text{if } k \bmod 2 = 1 \text{ and } f \in \Lambda_{\Gamma_{m_k}}\left(\left(\oplus_{i=1}^{k} a^{(i)}\right) \oplus b\right) \end{cases}$$

and hence by Definition 1.3

$$\Gamma_{m_k}(a) = \left.\begin{cases} \Gamma_{m_k}\left(\left(\oplus_{i=1}^{k} a^{(i)}\right) \oplus c\right), & k \bmod 2 = 0, \\ \Gamma_{m_k}\left(\left(\oplus_{i=1}^{k} a^{(i)}\right) \oplus b\right), & k \bmod 2 = 1, \end{cases}\right\} = \delta_0(k \bmod 2).$$

All in all we obtain $\mathrm{SCI}(P_2)_G = \mathrm{SCI}(P_2)_R = \mathrm{SCI}(P_2)_A = 2$.

(iii) By Example 1.8 we obtain $\mathrm{SCI}(P_3)_A \leq 1$. We will now prove $\mathrm{SCI}(P_3)_G \geq 1$ to conclude $\mathrm{SCI}(P_3)_G = \mathrm{SCI}(P_3)_R = \mathrm{SCI}(P_3)_A = 1$:

Assume there was a general algorithm $\Gamma\colon \Omega \to \mathcal{M}$ solving the halting problem P_3. Then we consider the word $x = 0$ and a TURING machine which on input x wrights in every step a zero on the right hand side of the existing word. Then we have $\Gamma(M, x) = 0$ by assumption, since M does not halt on x. By the Definition 1.3 of a general algorithm there is an $N \in \mathbb{N}$ for which $\Lambda_\Gamma(M, x) \subseteq \{f_{n,k,letter}, f_{n,head}, f_{n,end} \mid n, k \leq N\}$ holds. Now let M' be another TURING machine doing on x the same as M but after the $(N+1)$-th step halting. We conclude $f(M, x) = f(M', x)$ for all $f \in \Lambda_\Gamma(M, x)$ and hence $\Gamma(M, x) = \Gamma(M', x)$ by the definition of a general algorithm. But this is a contradiction, since M' halts on x. Thus we have $\mathrm{SCI}(P_3)_g \geq 1$.

The last result about the halting problem shows the importance of the information the algorithms is given to solve the problem for the resulting complexity. In this specific setting the algorithm is only able to follow the computations of TURING machine but not allowed to extract information about how the TURING machine is defined. This is crucial, since otherwise the problem would be solvable by general algorithms without applying any limit due to the fact, that a TURING machine in Ω is uniquely determined by its states and transition function, which is finite information.

2 Computational Problems of linear Operators

In this chapter we have a look at various computational problems motivated by functional analysis by applying the tools from Chapter 1 to spectral and inverse problems and analyse what Solvability Complexity Index they have. First, we examine the computability of the spectrum of a finite matrix, the easiest case we can think of in this context. Afterwards we follow Chapter 3 of [1] and present the results related to those about spectra of bounded linear operators on $\ell^2(\mathbb{N})$. Then we look at special cases and prove some hardness results. Going on with inverse problems, we refer to [1, Chap.5] and look again at special cases, where we are able to prove some positive statements.

In both topics to prove upper bounds, concrete algorithms appear, but note that in practise these might look different for example for stability reasons. There, it might be helpful to use radical algorithms instead of the given arithmetic ones.[1]

2.1 Spectral Problems

In the following computational problems Ω always consists of matrices or bounded linear operators on $\ell^2(\mathbb{N})$. As a metric space we take the set of all compact subsets of $\mathbb{C}$ equipped with the HAUSDORFF metric d_H. The problem function Ξ asks for the spectrum denoted by $\sigma(A)$ for $A \in \Omega$.

The following result about the computability of the spectrum of finite matrices gives us another example where the Solvability Complexity Index varies when we consider different types of algorithms.

Theorem 2.1. *Let* $\Omega := \bigcup_{n\in\mathbb{N}} \mathbb{C}^{n\times n}$ *and* $\Lambda := \{f_{i,j}^n \mid n,i,j \in \mathbb{N}\} \cup \{f_d\}$, *where*

$$f_{i,j}^n \colon \Omega \to \mathbb{C}, \quad f_{i,j}^n(A) := \begin{cases} a_{i,j} & , A \in \mathbb{C}^{n\times n} \\ 0 & , \textit{else} \end{cases}$$

and

$$f_d \colon \Omega \to \mathbb{C}, \quad f_d(A) := n, \ \textit{for } A \in \mathbb{C}^{n\times n} \subseteq \Omega.$$

Then we have

$$\mathrm{SCI}(\Omega,\Lambda,\mathcal{M},\Xi)_G = 0 \qquad \textit{and} \qquad \mathrm{SCI}(\Omega,\Lambda,\mathcal{M},\Xi)_A = \mathrm{SCI}(\Omega,\Lambda,\mathcal{M},\Xi)_R = 1.$$

Proof. In the general case, we set $\Gamma := \Xi$. To prove that Γ is a general algorithm, let $A \in \Omega$. Then there is an $n \in \mathbb{N}$ such that $A \in \mathbb{C}^{n\times n}$. We define the finite set

$$\Lambda_\Gamma(A) := \{f_{i,j}^n \mid 1 \le i,j \le n\} \cup \{f_d\} \subseteq \Lambda.$$

[1] See [1, Rem.2.8].

Since two matrices with the same dimension and entries are in fact equal and hence they have the same spectrum, Definition 1.3 of a general algorithm is fulfilled. Thus Γ is indeed a general algorithm and we obtain

$$\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_G = 0.$$

Now we show

$$1 \leq \mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_R \leq \mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_A \leq 1,$$

where $\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_R \leq \mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_A$ is obvious by Remark 1.10 about estimates of the Solvability Complexity Index with respect to different types of algorithms.

For $\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_R \geq 1$ we assume $\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_R = 0$. But then we would be able to compute the zero set of any polynomial with finitely many arithmetic operations and radicals since for every normalized polynomial $p \neq 0$ there is a matrix A, with characteristic polynomial $p_A = p$ and hence $\sigma(A) = N(p)$. This is a contradiction to the theorem of Abel and Ruffini, stating that this is not possible in general if the polynomial has degree five or larger. Therefore we have $\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)_R \geq 1$.

For an arithmetic tower of algorithms, we consider any $A \in \Omega$ and its characteristic polynomial $p_A(x) = \sum_{j=0}^{n} a_j x^j$, where $a_n = 1$. Then we have

$$|p_A(x)| = |x|^n \left| \sum_{j=0}^{n} \frac{a_j}{x^{n-j}} \right| \geq |x|^n \left(|a_n| - \left| \sum_{j=0}^{n-1} \frac{a_j}{x^{n-j}} \right| \right) \geq |x|^n \left(1 - \sum_{j=0}^{n-1} \left| \frac{a_j}{x^{n-j}} \right| \right)$$

for all $x \in \mathbb{C} \setminus \{0\}$. If we assume in addition $|x| > 2n \max_j |a_j|$, it even holds

$$|p_A(x)| \geq |x|^n \left(1 - \sum_{j=0}^{n-1} \left| \frac{a_j}{x^{n-j}} \right| \right) \geq |x|^n \left(1 - \sum_{j=0}^{n-1} \frac{1}{2n} \right) = |x| \frac{1}{2} \geq 1.$$

In particular we have $|p_A(x)| \geq 1$ for $|x| > C := 2n \max_j |a_j|^2$ and $|x| > C_1 := C + 1$ respectively.[2] Thus all zeros of p_A are elements of $B(0, C_1)$, meaning $N(p_A) \subseteq B(0, C_1)$. Furthermore, considering the derivative and $x, y \in B(0, C_1)$, we obtain

$$|p'_A(x)| = \left| \sum_{j=1}^{n} j a_j x^{j-1} \right| \leq \sum_{j=1}^{n} j |a_j| |x|^{j-1} \leq n^2 \max_{j \geq 1} |a_j| \, C_1^{n-1} \leq n^2 \max_{j \geq 1} |a_j|^2 \, C_1^{n-1} =: C_2$$

and thus

$$|p_A(x) - p_A(y)| = \left| \int_\gamma p'_A(z) dz \right| \leq C_2 \mathrm{length}(\gamma) = C_2 |x - y|,$$

where $\gamma \colon [0, 1] \to B(0, C_1), \gamma(t) := x + t(y - x)$. We now choose

$$B_k := \left\{ z \in B(0, C_1) \;\middle|\; z = \frac{a}{2kC_2} + i \frac{b}{2kC_2},\ a, b \in \mathbb{Z} \right\}$$

and

$$\Gamma_k(A) := \left\{ z \in B_k \;\middle|\; |p_A(z)|^2 \leq \frac{1}{k} \right\}$$

[2] We choose squares instead of just the absolute value to avoid radicals and obtain an arithmetic algorithm.

for $k \in \mathbb{N}$. First we prove that Γ_k is an arithmetic algorithm for all $k \in \mathbb{N}$. As in the general case, we set

$$\Lambda_{\Gamma_k}(A) := \{f_{i,j}^n \mid 1 \leq i,j \leq n\} \cup \{f_d\} \subseteq \Lambda$$

for $A \in \mathbb{C}^{n\times n}$. For the same reasons as before Γ_k is a general algorithm. Furthermore, we only need finitely many arithmetic operations and comparisons on $f(A)$ for $f \in \Lambda_{\Gamma_k}(A)$ to determine $\Gamma_k(A)$, since B_k is finite and C_1 as well as C_2 are arithmetic expressions in the evaluations. Finally we will show

$$\lim_{k\to\infty} \Gamma_k(A) = \Xi(A).$$

For this let $\epsilon > 0$. We claim, there is an $N_1 \in \mathbb{N}$, such that for all $k \geq N_1$ and $t \in \Xi(A)$ there is a $z \in \Gamma_k(A)$ with $|z-t| < \epsilon$. To this end, let N_1, such that $\frac{1}{N_1} < \epsilon$ and $t \in \Xi(A)$. Then for all $k \geq N_1$ we have

$$t \in B(0,C_1) \subseteq \bigcup_{z\in B_k} B\left(z, \frac{2}{2kC_2}\right) = \bigcup_{z\in B_k} B\left(z, \frac{1}{kC_2}\right).$$

Thus, there is a $z \in B_k \subseteq B(0,C_1)$ such that $t \in B\left(z, \frac{1}{kC_2}\right)$. Particularly

$$|p_A(z)| = |p_A(z) - p_A(t)| \leq C_2|z-t| \leq C_2\frac{1}{kC_2} \leq \frac{1}{k} \qquad \text{and hence} \qquad |p_A(z)|^2 \leq \frac{1}{k}.$$

This implies $z \in \Gamma_k(A)$ as well as $|z-t| < \frac{1}{kC_2} < \frac{1}{N_1} < \epsilon$ so the claim is proven.

Conversely there is an $N_2 \in \mathbb{N}$, such that for all $k \geq N_2$ and $z \in \mathbb{C}$ with $|p_A(z)|^2 \leq \frac{1}{k}$ a $t \in \Xi(A)$ exists, meeting $|z-t| < \epsilon$. To achieve that, we consider the linear factorisation $\prod_{t\in\Xi(A)}(x-t)^{a_t}$ of p_A and choose N_2 so that $\frac{1}{N_2} < \epsilon^{2n}$. If we now assume, there is a $k \geq N_2$ and a $z \in \mathbb{C}$ satisfying $|p_A(z)|^2 \leq \frac{1}{k}$ but also $|z-t| \geq \epsilon$ for all $t \in \Xi(A)$, we would obtain

$$|p_A(z)|^2 = \prod_{t\in\Xi(A)} |z-t|^{2a_t} \geq \prod_{t\in\Xi(A)} \epsilon^{2a_t} = \epsilon^{2n},$$

contradicting the choice of N_2.

Now we set $N := \max\{N_1, N_2\}$ and conclude $d_H(\Xi(A), \Gamma_k(A)) \leq \epsilon$ for all $k \geq N$. Thus $\Gamma_k(A) \to \Xi(A)$ as $k \to \infty$ and hence $\{\Gamma_k\}_{k\in\mathbb{N}}$ is an arithmetic tower of algorithms of height 1 for $(\Omega, \Lambda, \mathcal{M}, \Xi)$. ■

We review this result and its proof in Theorem 3.7 in the context of error control.
The statement from Theorem 2.1 that $\mathrm{SCI}(\Omega,\Lambda,\mathcal{M},\Xi)_G = 0$ but $\mathrm{SCI}(\Omega,\Lambda,\mathcal{M},\Xi)_R = \mathrm{SCI}(\Omega,\Lambda,\mathcal{M},\Xi)_A = 1$ again shows the power of general algorithms as the answer $\Xi(A)$ of a question Ξ on an input A just need to depend on a finite amount of information of A to be computable for general algorithms. In contrast, having $\mathrm{SCI}_G \geq k$ for some $k \in \mathbb{N}$ is thus a very strong property making computation of such a problem reasonably hard. The next two results give us examples of such problems.

In following, Ω is a subset of $L(\ell^2(\mathbb{N}))$, the set of all bounded linear operators on $\ell^2(\mathbb{N})$. The set Λ of evaluations consists of the function $f_{i,j}\colon \Omega \to \mathbb{C}, A \mapsto \langle Ae_i, e_j\rangle$ for $i,j \in \mathbb{N}$ and the metric space $\mathcal{M}$ be the set of all compact subsets of $\mathbb{C}$ equipped with the Hausdorff metric as before. Finally we set $\Xi\colon \Omega \to \mathcal{M}$ as $\Xi(A) := \sigma(A)$.

Theorem 2.2. *Let $\Omega_1 := L(\ell^2(\mathbb{N}))$, Ω_2 be the set of all bounded, normal operators and Ω_3 the set of all bounded, self-adjoint operators. Then we have*

$$\mathrm{SCI}(\Omega_1, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_1, \Lambda, \mathcal{M}, \Xi)_A = 3$$

and

$$\mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_A = 2$$

for $i = 2, 3$.

For the proof of this result from [1] we refer to the same source. If the reader is just interested in the structure of the proof for the lower bounds we emphasise to look at Theorem 2.4. It is inspired by the latter and shows a strengthening of Theorem 2.2 to unitary and positive semidefinite operators. But beforehand we need a lemma which gives us some insight in constructing operators on $\ell^2(\mathbb{N})$ out of finite matrices. It will be useful to find counterexamples.

Lemma 2.3. *Let $A_i \in \mathbb{C}^{n_i \times n_i}$ for $i, n_i \in \mathbb{N}$. Is there an $M > 0$ such that $\sup_{i\in\mathbb{N}} \|A\|_2 \leq M$ then $A := \bigoplus_{i=1}^{\infty} A_i \in L(\ell^2(\mathbb{N}))$.[3] In this case we have*

$$\|A\| \leq M, \quad \sigma(A) = \bigcup_{i=1}^{\infty} \sigma(A_i) \quad \text{as well as} \quad A^* = \bigoplus_{i=1}^{\infty} A_i^*.$$

Further, if A_i is self-adjoint / unitary / positive semidefinite for all $i \in \mathbb{N}$, so is A.

Proof. Let $x \in \ell^2(\mathbb{N})$ and $x_i \in \mathbb{C}^{n_i}$ for $i \in \mathbb{N}$ the vector that contains the components of x starting from index $j_{i,1} := \sum_{k=1}^{i-1} n_k + 1$ up to $j_{i,2} := \sum_{k=1}^{i} n_k$. Then we have

$$\|Ax\|_{\ell^2}^2 = \sum_{i=1}^{\infty} \|A_i x_i\|_2^2 \leq \sum_{i=1}^{\infty} \|A_i\|_2 \|x_i\|_2^2 \leq M \sum_{i=1}^{\infty} \|x_i\|_2^2 = M\|x\|_{\ell^2},$$

hence $\|A\| \geq M$ and $A \in L(\ell^2(\mathbb{N}))$.

Now let $\lambda \in \sigma(A_i)$ for some $i \in \mathbb{N}$ and $y \in \mathbb{C}^{n_i}$ a corresponding eigenvector. Further, let $\hat{y} \in \ell^2(\mathbb{N})$ be the sequence satisfying $P_{e_j}\hat{y} = 0$ for all $j < j_{i,1}$ and $j > j_{i,2}$ and being the k-th entry of y for $j = j_{i,1} + k - 1$ and $1 \leq k \leq j_{i,2} - j_{i,1} + 1$. By this we get $A\hat{y} = \lambda\hat{y}$ and thus

$$\bigcup_{i=1}^{\infty} \sigma(A_i) \subseteq \sigma(A).$$

For $\lambda \notin \bigcup_{i=1}^{\infty} \sigma(A_i)$ we conclude

$$A - \lambda I_{\ell^2(\mathbb{N})} = \bigoplus_{i=1}^{\infty} (A_i + \lambda I_{\mathbb{C}^{n_i}})$$

and hence the invertibility of $A - \lambda I_{\ell^2(\mathbb{N})}$ by the invertibility of $A_i + \lambda I_{\mathbb{C}^{n_i}}$ for all $i \in \mathbb{N}$. Since $\ell^2(\mathbb{N})$ is a Banach space and $A - \lambda I_{\ell^2(\mathbb{N})}$ is bounded it is in fact boundedly invertible using the Open Mapping Theorem. Thus $\lambda \notin \sigma(A)$ and the desired equality holds.

To show the third statement let $B := \bigoplus_{i=1}^{\infty} A_i^*$. Note that B is well defined by our step of the proof since $\|A_i^*\|_2 = \|A_i\|_2$. Then, for $x \in \ell^2(\mathbb{N})$ with $x_i \in \mathbb{C}^{n_i}$ defined as before we

[3] We consider operatornorm on $L(\mathbb{C}^{n_i})$ induced by the 2-norm.

have

$$\langle A^*x, x\rangle_{\ell^2} = \overline{\langle Ax, x\rangle}_{\ell^2} = \overline{\sum_{i=1}^{\infty} \langle A_i x_i, x_i\rangle_{\mathbb{C}^{n_i}}} = \sum_{i=1}^{\infty} \overline{\langle A_i x_i, x_i\rangle}_{\mathbb{C}^{n_i}} = \sum_{i=1}^{\infty} \langle A_i^* x_i, x_i\rangle_{\mathbb{C}^{n_i}}$$

which implies $\langle A^*x, x\rangle_{\ell^2} = \langle Bx, x\rangle_{\ell^2}$ and hence $A^* = B$.

Now let A_i be self-adjoint for all $i \in \mathbb{N}$. Then $A^* = \bigoplus_{i=1}^{\infty} A_i^* = \bigoplus_{i=1}^{\infty} A_i = A$, so A is self-adjoint.

If A_i is unitary for all $i \in \mathbb{N}$ we have

$$\left(\bigoplus_{i=1}^{\infty} A_i^*\right)\left(\bigoplus_{i=1}^{\infty} A_i\right) = \bigoplus_{i=1}^{\infty} A_i^* A_i = \bigoplus_{i=1}^{\infty} A_i^{-1} A_i = I_{\ell^2(\mathbb{N})}$$

as well as $\left(\bigoplus_{i=1}^{\infty} A_i\right)\left(\bigoplus_{i=1}^{\infty} A_i^*\right) = I_{\ell^2(\mathbb{N})}$. Therefore, A is invertible with $A^{-1} = \bigoplus_{i=1}^{\infty} A_i^* = A^*$ and hence unitary.

Finally, if A_i is positive semidefinite for all $i \in \mathbb{N}$ and $x \in \ell^2(\mathbb{N})$ with $x_i \in \mathbb{C}^{n_i}$ defined as before, we conclude

$$\langle Ax, x\rangle = \sum_{i=1}^{\infty} \langle A_i x_i, x_i\rangle \geq 0$$

and thus the corresponding property for A. ■

Now we can prove the result mentioned in the previous paragraph. Especially the case of unitary *and* self-adjoint operators shows how hard it is to compute spectra, even if they are subsets of $\{1, -1\}$.

Theorem 2.4. *Let Ω_4 be the set of all unitary, self-adjoint operators and Ω_5 the set of all bounded, positive semidefinite operators. Then we have*

$$\mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_A = 2,$$

for $i = 4, 5$.

Proof. First we have $\mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_A \leq 2$ for $i = 4, 5$ by Theorem 2.2, since positive semidefinite operators are in particular self-adjoint.

It remains to prove $\mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_G \geq 2$ for $i = 4, 5$. Towards a contradiction, we assume the existence of a tower of algorithms $\{\Gamma_k\}_{k\in\mathbb{N}}$ of height one for $(\Omega_i, \Lambda, \mathcal{M}, \Xi)$ for $i = 4$ and $i = 5$ respectively. For any $A \in \Omega_4 \cup \Omega_5$ and $k \in \mathbb{N}$ let $N(A, k) \in \mathbb{N}$ be the largest index, such that Γ_k only takes the entries $f_{i,j}(A) = \langle Ae_i, e_j\rangle$ with $i, j \leq N(A, k)$ into account. Then we consider the operators $A_i := \bigoplus_{n=1}^{\infty} A_{l_n^{(i)},i}$ for $i = 4, 5$ with

$$A_{n,4} := \begin{bmatrix} \frac{1}{\sqrt{2}} & & & & \frac{1}{\sqrt{2}} \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ \frac{1}{\sqrt{2}} & & & & -\frac{1}{\sqrt{2}} \end{bmatrix} \in \mathbb{C}^{n\times n} \quad \text{and} \quad A_{n,5} := \begin{bmatrix} \frac{1}{\sqrt{2}} & & & & \frac{1}{\sqrt{2}} \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ \frac{1}{\sqrt{2}} & & & & \frac{1}{\sqrt{2}} \end{bmatrix} \in \mathbb{C}^{n\times n}$$

where $n \in \mathbb{N}$ and $(l_n^{(i)})_{n\in\mathbb{N}} \subseteq \mathbb{N}$ is a sequence which we construct hereafter by induction. First of all every $A_{n,4}$ is unitary and self-adjoint satisfying $\sigma(A_{n,4}) = \{1, -1\}$ and every $A_{n,5}$ positive semidefinite satisfying $\sigma(A_{n,5}) = \{1, 0, \sqrt{2}\}$. By Lemma 2.3, A_4 and A_5 inherit this

properties. Further, we consider $C = \mathrm{diag}(\frac{1}{\sqrt{2}}, 1, 1, 1, \ldots)$ with $\sigma(C) = \{1, \frac{1}{\sqrt{2}}\}$ and choose $k_0 = 1$ as well as $l_1 > N(C, k_0)$. Suppose, $l_1^{(i)}, \ldots, l_m^{(i)}$ and $k_0, \ldots, k_{m-1}$ are already chosen. Then we have $\sigma\left(\bigoplus_{n=1}^m A_{l_n^{(4)},4} \oplus C\right) = \{1, -1, \frac{1}{\sqrt{2}}\}$, $\sigma\left(\bigoplus_{n=1}^m A_{l_n^{(5)},5} \oplus C\right) = \{1, 0, \sqrt{2}, \frac{1}{\sqrt{2}}\}$ and hence there is a $k_m > k_{m-1}$ such that

$$\Gamma_{k_m}\left(\bigoplus_{n=1}^m A_{l_n^{(i)},i} \oplus C\right) \cap B\left(\frac{1}{\sqrt{2}}, \frac{1}{m}\right) \neq \emptyset$$

for $i = 4, 5$. Now we choose

$$l_{n+1}^{(i)} > N\left(\bigoplus_{n=1}^m A_{l_n^{(i)},i} \oplus C\right) - \sum_{n=1}^m l_n^{(i)}.$$

By construction we have

$$\Gamma_{k_m}\left(\bigoplus_{n=1}^\infty A_{l_n^{(i)},i}\right) \cap B\left(\frac{1}{\sqrt{2}}, \frac{1}{m}\right) = \Gamma_{k_m}\left(\bigoplus_{n=1}^m A_{l_n^{(i)},i} \oplus C\right) \cap B\left(\frac{1}{\sqrt{2}}, \frac{1}{m}\right) \neq \emptyset,$$

using Definition 1.3 of a general algorithm, since for all $f_{i,j} \in \Lambda_{\Gamma_{k_m}}\left(\bigoplus_{n=1}^m A_{l_n^{(i)},i} \oplus C\right)$ we have

$$f_{i,j}\left(\bigoplus_{n=1}^m A_{l_n^{(i)},i} \oplus C\right) = f_{i,j}\left(\bigoplus_{n=1}^\infty A_{l_n^{(i)},i}\right).$$

Thus it has to be $\frac{1}{\sqrt{2}} \in \lim_{m\to\infty} \Gamma_{k_m}\left(\bigoplus_{n=1}^\infty A_{l_n^{(i)},i}\right)$ contradicting $\{1, -1\} = \sigma\left(\bigoplus_{n=1}^\infty A_{l_n^{(4)},4}\right) = \lim_{k\to\infty} \Gamma_k\left(\bigoplus_{n=1}^\infty A_{l_n^{(4)},4}\right)$ and $\{1, 0, \sqrt{2}\} = \sigma\left(\bigoplus_{n=1}^\infty A_{l_n^{(5)},5}\right) = \lim_{k\to\infty} \Gamma_k\left(\bigoplus_{n=1}^\infty A_{l_n^{(5)},5}\right)$ respectively. ∎

There is actually a class of operators for which the computation of the spectrum just requires one limit: The compact operators. This comes not as a surprise, since compact operators can be approximated by finite dimensional operators for which we already showed that their spectra are computable within one limit. Also, this result was already proven in [1]. We give a more detailed proof. To this end, we first discuss two lemmata.

Lemma 2.5. *Let $K \in \mathcal{K}(H)$ for some separable* H *with orthonormal basis $(e_n)_{n\in\mathbb{N}}$. Then it holds $P_nKP_n \to K$ as $n \to \infty$ with respect to the operator norm.*

Proof. Since

$$\begin{aligned}\|P_nKP_n - K\| &\leq \|P_nKP_n - KP_n\| + \|KP_n - K\| \\ &\leq \|P_nK - K\| + \|KP_n - K\| \\ &= \|K^*P_n - K^*\| + \|KP_n - K\|\end{aligned}$$

for all $n \in \mathbb{N}$ and K^* is compact too, it suffices to show $KP_n \to K$ as $n \to \infty$. Suppose, this is not the case. Then there is a $c > 0$ and a sequence $(x_n)_{n\in\mathbb{N}} \subseteq H$ with $\|x_n\| = 1$ and $\|K(P_n - I)x_n\| = \|KP_nx_n - Kx_n\| \geq c$ for all $n \in \mathbb{N}$. On the other hand we have

$$\langle (P_n - I)x_n, e_k\rangle = \langle x_n, (P_n - I)e_k\rangle = 0$$

for all $k \in \mathbb{N}$ and $n \geq k$. Thus $(P_n - I)x_n \xrightarrow{w} 0$ as $n \to \infty$ and hence $K(P_n - I)x_n \to 0$ as $n \to \infty$, which is a contradiction. ■

Lemma 2.6. *Let H be any* HILBERT *space, $K \in \mathcal{K}(H)$ and $(K_n)_{n\in\mathbb{N}} \subseteq \mathcal{K}(H)$ such $K_n \to K$ as $n \to \infty$ with respect to the operator norm. If $(\lambda_m)_{m\in\mathbb{N}} \subseteq \mathbb{C}$ is a a enumeration of the non-zero eigenvalues of K, then there is a enumeration $(\lambda_{m,n})_{m\in\mathbb{N}} \subseteq \mathbb{C}$ of the non-zero eigenvalues of K_n, such that*

$$\lim_{n\to\infty} \lambda_{m,n} = \lambda_m,$$

where the limit is uniform in $m \in \mathbb{N}$.

For this lemma we omit the proof and refer to [5, XI.9 Lemma 5].

Now we are able to state and show the result on compact operators, which can also be seen as a generalisation of the Theorem 2.1 about spectra of finite matrices.

Theorem 2.7. *Let Ω_6 be the set $\mathcal{K}(\ell^2(\mathbb{N}))$ of compact operators. Then we have*

$$\mathrm{SCI}(\Omega_6, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_6, \Lambda, \mathcal{M}, \Xi)_A = 1.$$

Proof. First we prove $\mathrm{SCI}(\Omega_6, \Lambda, \mathcal{M}, \Xi)_G \geq 1$. For this we assume $\mathrm{SCI}(\Omega_6, \Lambda, \mathcal{M}, \Xi)_G = 0$, that is, the existence of a general algorithm Γ satisfying $\Gamma(K) = \Xi(K)$ for all $K \in \mathcal{K}(\ell^2(\mathbb{N}))$. Let N be the largest index such that $f_{i,j} \in \Lambda_\Gamma(0)$ for $i, j \leq N$. Then we consider $P_{e_{N+1}}$ and conclude $f_{i,j}(0) = f_{i,j}(P_{e_{N+1}})$ for all $i, j \leq N$. Definition 1.3 of a general algorithm implies $\Gamma(0) = \Gamma(P_{e_{N+1}})$. But this is a contradiction since

$$\Gamma(0) = \sigma(0) = \{0\} \neq \{0, 1\} = \sigma(P_{e_{N+1}}) = \Gamma(P_{e_{N+1}}).$$

Thus we have $\mathrm{SCI}(\Omega_6, \Lambda, \mathcal{M}, \Xi)_G \geq 1$.

Now we show $\mathrm{SCI}(\Omega_6, \Lambda, \mathcal{M}, \Xi)_A \leq 1$. For this occasion let $K \in \mathcal{K}(\ell^2(\mathbb{N}))$ and $\{\tilde{\Gamma}_k\}_{k\in\mathbb{N}}$ the tower of algorithms from Corollary 3.7, where we will construct a better version of the algorithm from the proof of Theorem 2.1 about spectra of finite matrices. For $k \in \mathbb{N}$ we set $\Gamma_k(K) := \tilde{\Gamma}_k(P_kKP_k) \cup \{0\}$.[4] Then Γ_k is an arithmetic algorithm, since $\tilde{\Gamma}_k$ already was. In order to prove $\Gamma_k \to \Xi$ as $k \to \infty$ let $\epsilon > 0$. By Corollary 3.7 there is a $K_1 \in \mathbb{N}$ such that $d_H(\tilde{\Gamma}_k(P_kKP_k) \cup \{0\}, \sigma(P_kKP_k)) < \frac{\epsilon}{2}$ for all $k \geq K_1$. Further, by Lemma 2.5 and 2.6 there is a $K_2 \in \mathbb{N}$ such that $d_H(\sigma(P_kKP_k) \setminus \{0\}, \sigma(K) \setminus \{0\}) < \frac{\epsilon}{2}$ for all $k \geq K_2$. Since $0 \in \sigma(K) \cap \sigma(P_kKP_k)$, in particular $d_H(\sigma(P_kKP_k), \sigma(K)) < \frac{\epsilon}{2}$ and hence

$$d_H(\Gamma_k(K), \Xi(K)) \leq d_H(\tilde{\Gamma}_k(P_kKP_k) \cup \{0\}, \sigma(P_kKP_k) \cup \{0\}) + d_H(\sigma(P_kKP_k) \cup \{0\}, \sigma(K)) < \epsilon$$

for all $k \geq \max(K_1, K_2)$. That is $\Gamma_k(K) \to \Xi(K)$ for $k \to \infty$ and thus $\mathrm{SCI}(\Omega_6, \Lambda, \mathcal{M}, \Xi)_A \leq 1$. ■

Note that this result is related to Theorem 3.8 where we see that despite the comparably low SCI no error control is possible in the computation of spectra of compact operators.

[4] Since $\tilde{\Gamma}_k$ is only defined for finite matrices, more precisely, instead P_kKP_k we consider the representation matrix of the linear mapping $A\colon \mathbb{C}^k \to \mathbb{C}^k, x \mapsto R_kP_kKP_kL_kx$ where $L_k\colon \mathbb{C}^k \to \ell^2(\mathbb{N}), x \mapsto x \oplus 0$ and $R_k\colon \ell^2(\mathbb{N}) \to \mathbb{C}^k, x \mapsto (x_1, \ldots, x_k)$.

2.2 Inverse Problems

Next we focus on well-posed[5] inverse problems. Also here even very structured operator classes such as self-adjoint operators do not imply any lowering of the bounds for computing solutions. And even more striking as in the spectral space: They have the same Solvability Complexity Index as the class of all operators.

In the following computational problems Ω is always a subset of $L_{inv}(\ell^2(\mathbb{N})) \times \ell^2(\mathbb{N})$. Thereby $L_{inv}(\ell^2(\mathbb{N}))$ denotes the set of all boundedly invertible operators, meaning $T \in L_{inv}(\ell^2(\mathbb{N}))$ if and only if T is bijective and $T, T^{-1} \in L(\ell^2(\mathbb{N}))$. The set Λ consists of all function $f_{i,j}$ and f_i for $i, j \in \mathbb{N}$ mapping from Ω to $\mathbb{C}$ such that $f_{i,j}(A, b) = \langle Ae_i, e_j \rangle$ and $f_i(A, b) = \langle b, e_i \rangle$ for $(A, b) \in \Omega$. The metric space $\mathcal{M}$ is just $\ell^2(\mathbb{N})$. The problem function Ξ we define to be $\Xi(A, b) = A^{-1}b$.

Theorem 2.8. *Let $L_{inv,sa}(\ell^2(\mathbb{N})) \subseteq L_{inv}(\ell^2(\mathbb{N}))$ be the set of all operators, which are boundedly invertible and self-adjoint. We define $\Omega_1 := L_{inv}(\ell^2(\mathbb{N})) \times \ell^2(\mathbb{N})$ and $\Omega_2 := L_{inv,sa}(\ell^2(\mathbb{N})) \times \ell^2(\mathbb{N})$. Then we have*

$$\mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_i, \Lambda, \mathcal{M}, \Xi)_A = 2$$

for $i = 1, 2$.

This result is again from [1] and again we are able to strengthen this to the following:

Theorem 2.9. *Let $L_{inv,psd}(\ell^2(\mathbb{N}))$ be the set of all boundedly invertible and positive semidefinite operators and $\Omega_3 := L_{inv,psd}(\ell^2(\mathbb{N})) \times \ell^2(\mathbb{N})$. Then we have*

$$\mathrm{SCI}(\Omega_3, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_3, \Lambda, \mathcal{M}, \Xi)_A = 2.$$

Proof. Using Theorem 2.8 yields immediately $\mathrm{SCI}(\Omega_3, \Lambda, \mathcal{M}, \Xi)_A \leq 2$. Therefore, we assume $\mathrm{SCI}(\Omega_3, \Lambda, \mathcal{M}, \Xi)_G < 2$ and the existence of a tower of algorithms $\{\tilde{\Gamma}_k\}_{k \in \mathbb{N}}$ of height one. This we utilise to construct a tower of algorithms of height one for $(\Omega_2, \Lambda, \mathcal{M}, \Xi)$ and to obtain a contradiction to Theorem 2.8. Let $(A, b) \in \Omega_2$. Then A^2 is positive semidefinite and invertible and we have

$$\lim_{k \to \infty} \tilde{\Gamma}_k(A^2, b) = (A^2)^{-1}b = (A^{-1})^2 b.$$

We now define $\Gamma_k(A, b) := P_k A P_k \tilde{\Gamma}_k(A^2, b)$ for $k \in \mathbb{N}$ and obtain a general algorithm. Further we have

$$\lim_{k \to \infty} \Gamma_k(A, b) = A(A^{-1})^2 b = A^{-1}b,$$

which implies the desired contradiction. All in all, we have $\mathrm{SCI}(\Omega_3, \Lambda, \mathcal{M}, \Xi)_A = 2$. ∎

In contrast to the spectral case, solving inverse problems with unitary operators is easier than with positive semidefinite operators. This comes not as a surprise since already in the finite dimensional setting, inverting unitary matrices is very simple.

Theorem 2.10. *Let $\mathrm{U}(\ell^2(\mathbb{N}))$ be the set of all unitary operators $\Omega_4 := \mathrm{U}(\ell^2(\mathbb{N})) \times \ell^2(\mathbb{N})$. Then we have*

$$\mathrm{SCI}(\Omega_4, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_4, \Lambda, \mathcal{M}, \Xi)_A = 1.$$

[5] In the sense of Hadamard, going back to [6].

Proof. To falsify $\mathrm{SCI}(\Omega_4, \Lambda, \mathcal{M}, \Xi)_G = 0$, we assume the existence of a general algorithm Γ with $\Gamma(A,b) = \Xi(A,b)$ for all $(A,b) \in \Omega$ an. Let $N \in \mathbb{N}$ be the largest index such that $f_{i,j}, f_k \in \Lambda_\Gamma(I,0)$ for $i,j,k \leq N$. Then we have $f_{i,j}(I,0) = f_{i,j}(I, e_{N+1})$ and $f_k(I,0) = f_k(I,e_{N+1})$ for all $i,j,k \leq N$ and therefore

$$0 = \Gamma(I,0) = \Gamma(I, e_{N+1}) = e_{N+1}$$

by Definition 1.3 of a general algorithm. This is a contradiction and implies $\mathrm{SCI}(\Omega_4, \Lambda, \mathcal{M}, \Xi)_G \geq 1$.

We will now show $\mathrm{SCI}(\Omega_4, \Lambda, \mathcal{M}, \Xi)_A \leq 1$. For this let $(A,b) \in \Omega_4$. We define

$$\Gamma_k(A,b) := (P_k A P_k)^* b = P_k^* A^* P_k^* b = P_k A^{-1} P_k b$$

for $k \in \mathbb{N}$ and set $\Lambda_{\Gamma_k}(A,b) := \{f_{i,j}, f_l \mid i,j,l \leq k\} \subseteq \Lambda$. Then Γ_k is a non-adaptive general algorithm, since for all $(A_1,b_1),(A_2,b_2) \in \Omega_4$ such that $(A_1)_{i,j} = f_{i,j}(A_1,b_1) = f_{i,j}(A_2,b_2) = (A_2)_{i,j}$ and $(b_1)_l = f_l(A_1,b_1) = f_l(A_2,b_2) = (b_2)_l$ for all $i,j,l \leq k$ we have $(P_k A_1 P_k)^* b_1 = (P_k A_2 P_k)^* b_2$, which gives the condition from Definition 1.3 of a general algorithm. Moreover it is arithmetic, since we need only finitely many arithmetic operations on $\{f_{i,j}(A,b), f_l(A,b) \mid i,j,l \leq k\}$ to compute $(P_k A P_k)^* b$, since we just have to transpose, conjugate a $k \times k$ matrix and apply the result to $P_k b$.[6] Finally, by the pointwise convergence $(P_k)_{k \in \mathbb{N}}$ to the identity, we deduce

$$\lim_{k \to \infty} \Gamma_k(A,b) = \lim_{k \to \infty} P_k A^{-1} P_k b = A^{-1} b = \Xi(A,b)$$

and hence $\mathrm{SCI}(\Omega_4, \Lambda, \mathcal{M}, \Xi)_A \leq 1$. ∎

Another of finite matrices where the corresponding inverse problem is easy to solve is the class of lower triangular ones. The following result extends this to infinite matrices.

Theorem 2.11. *Let $L_{inv,\triangle}(\ell^2(\mathbb{N}))$ be the set of all boundedly invertible lower triangular operators, where we call $T \in L(\ell^2(\mathbb{N}))$ lower triangular operator, if $\langle Te_i, e_j \rangle = 0$ for $i > j$. Further we define $\Omega_5 := L_{inv,\triangle}(\ell^2(\mathbb{N})) \times \ell^2(\mathbb{N})$. Then we have*

$$\mathrm{SCI}(\Omega_5, \Lambda, \mathcal{M}, \Xi)_G = \mathrm{SCI}(\Omega_5, \Lambda, \mathcal{M}, \Xi)_A = 1.$$

Proof. To obtain $\mathrm{SCI}(\Omega_5, \Lambda, \mathcal{M}, \Xi)_G \geq 1$, one can use the same counterexample as in the proof of Theorem 2.10.

Now let $(A,b) \in \Omega_5$. Then we define $\Gamma_k(A,b) := P_k A^{-1} P_k b$ for any $k \in \mathbb{N}$ and obtain

$$\lim_{k \to \infty} \Gamma_k(A,b) = A^{-1} b = \Xi(A,b)$$

immediately. Therefore, it remains to prove that Γ_k is an arithmetic Algorithm for all $k \in \mathbb{N}$. To this end, we set $\Lambda_{\Gamma_k}(A,b) := \{f_{i,j}, f_l \mid i,j,l \leq k\} \subseteq \Lambda$. Therefore Γ_k is non-adaptive. Further, A^{-1} is again a lower triangular operator. We prove this by induction. First, we have $\langle Ae_i, e_1 \rangle = 0$ for all $i > 1$, thus $A^* e_1 \in \mathrm{span}(\{e_1\}) \backslash \{0\}$ and by $\langle AA^{-1} e_i, e_1 \rangle = 0$ we conclude $\langle A^{-1} e_i, e_1 \rangle = 0$. Now we assume to know $\langle A^{-1} e_i, e_j \rangle = 0$ for all $1 \leq j \leq k$ and $i > j$. Then we deduce $A^* e_{k+1} \in \mathrm{span}(\{e_1, \ldots, e_{k+1}\})$ by $\langle Ae_i, e_{k+1} \rangle = 0$ for all $i > k+1$. Moreover it is $A^* e_{k+1} \notin \mathrm{span}(\{e_1, \ldots, e_k\})$ since otherwise $A^* e_{k+1}$ and $A^* e_k$ would be linearly dependent.

[6] Note that we have $(P_k A P_k)^* b = (P_k A P_k)^* P_k b$.

Thus, using the induction hypothesis,

$$0 = \left\langle AA^{-1}e_i, e_{k+1}\right\rangle = \left\langle A^{-1}e_i, A^*e_{k+1}\right\rangle = \sum_{l=1}^{k+1} \lambda_l \left\langle A^{-1}e_i, e_l\right\rangle = \lambda_{k+1}\left\langle A^{-1}e_i, e_{k+1}\right\rangle$$

for all $i > k+1$ and some $\lambda_1, \ldots, \lambda_{k+1} \in \mathbb{C}$ satisfying $\lambda_{k+1} \neq 0$. Hence $\left\langle A^{-1}e_i, e_{k+1}\right\rangle = 0$ for all $i > k+1$, which finishes the inductive proof.
By this we obtain

$$\left\langle A^{-1}e_i, e_j\right\rangle = \frac{1}{\langle Ae_j, e_j\rangle}\left(\delta_{i,j} - \sum_{k=i}^{j-1}\left\langle A^{-1}e_i, e_k\right\rangle \langle Ae_k, e_j\rangle\right)$$

for all $i, j \in \mathbb{N}$ since

$$\begin{aligned}
\delta_{i,j} &= \langle e_i, e_j\rangle \\
&= \left\langle AA^{-1}e_i, e_j\right\rangle \\
&= \left\langle A^{-1}e_i, A^*e_j\right\rangle \\
&= \sum_{k\in\mathbb{N}} \left\langle A^{-1}e_i, e_k\right\rangle \langle e_k, A^*e_j\rangle \\
&= \sum_{k\in\mathbb{N}} \left\langle A^{-1}e_i, e_k\right\rangle \langle Ae_k, e_j\rangle \\
&= \sum_{k=i}^{j} \left\langle A^{-1}e_i, e_k\right\rangle \langle Ae_k, e_j\rangle .
\end{aligned}$$

Note that $\langle Ae_j, e_j\rangle \neq 0$ for all $j \in \mathbb{N}$ since A is invertible. Using this formula we can compute $\left\langle A^{-1}e_i, e_j\right\rangle$ iteratively for $i, j \leq k$. Thus $P_kA^{-1}P_kb$ is uniquely obtained by finitely many arithmetic operations on $f(A, b)$ for $f \in \Lambda_{\Gamma_k}(A, b)$. Hence also Definition 1.3 of a general algorithm is fulfilled and $\{\Gamma_k\}_{k\in\mathbb{N}}$ is an arithmetic tower of algorithms of height one for $(\Omega_5, \Lambda, \mathcal{M}, \Xi)$. ■

Both Theorem 2.10 and Theorem 2.11 are related to Theorem 3.10 where we prove that even for very simple inverse problems including special cases of the two previous settings no error control is possible.

3 Error Control

In the last chapter we have seen various examples of famous computational problems for which the Solvability Complexity Index is larger than one, even when using general algorithms. One might wonder why this should be bad for computing solutions of the considered computational problem. In all cases we also have algorithms which still approximate the desired quantity, so why should the complexity index be relevant in practise?

This chapter we have a look at a crucial property algorithms should have for practical applications: error control. We see some results when in general and in specific cases this property is fulfilled or just satisfiable by following [1, Chap. 6] and making own contributions.

But first let us start with the definition of global error control.

Definition 3.1. Let $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ be a computational problem and

$$\mathcal{T} := \{\Gamma_{n_1,\dots,n_m}\}_{(n_1,\dots,n_m)\in\mathbb{N}^m}$$

a tower of algorithms for P. Then $\mathcal{T}$ is said to provide global error control, if for every $N \in \mathbb{N}$ there are $n_1, \dots, n_k \in \mathbb{N}$ satisfying

$$d\left(\Gamma_{n_1,\dots,n_k}(A), \Xi(A)\right) < \frac{1}{N} \tag{2}$$

for all $A \in \Omega$, where d denotes the metric associated to $\mathcal{M}$.

With the following theorem we now see why having Solvability Complexity Index greater or equal to two could be impractical, since in this case we never have global error control. This is due to [1], but we state and prove a slightly more general version.

Theorem 3.2. *Let $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ be a computational problem, such that $\mathrm{SCI}(P)_\alpha \geq 2$ for some $\alpha \in \{G, A, R\}$ and*

$$\mathcal{T} := \{\Gamma_{n_1,\dots,n_m}\}_{(n_1,\dots,n_m)\in\mathbb{N}^m}$$

be a tower of algorithms of type α for P. Then $\mathcal{T}$ does not provide global error control.

Proof. Towards a contradiction we assume the opposite, that is $\mathcal{T}$ provides global error control. By definition, for $N \in \mathbb{N}$ there are integers $n_1, \dots, n_k \in \mathbb{N}$ satisfying

$$d\left(\Gamma_{n_1,\dots,n_k}(A), \Xi(A)\right) < \frac{1}{N}$$

for all $A \in \Omega$. Hence, setting $\Gamma_N := \Gamma_{n_1,\dots,n_k}$ we obtain a tower of algorithms $\tilde{\mathcal{T}} = \{\Gamma_N\}_{N\in\mathbb{N}}$ of height one and type α for P, since by Definition 1.7 of towers of algorithms $\Gamma_{n_1,\dots,n_k}$ is an algorithm of type α. This is a contradiction to $\mathrm{SCI}(\Omega, \Lambda, \mathcal{M}, \Xi)\alpha \geq 2$. ∎

With this result in mind one could ask if it is possible to obtain a weaker version of error control. One option would be to demand local error control in the sense that (2) holds not for all but for every fixed $A \in \Omega$. This is indeed always satisfied by Definition 1.7 of a tower

of algorithms. In practise then one would desire to be able to compute the corresponding indices for which (2) is attained.

Definition 3.3. Let $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ be a computational problem and

$$\mathcal{T} := \{\Gamma_{n_1,\dots,n_m}\}_{(n_1,\dots,n_m)\in\mathbb{N}^m}$$

a tower of algorithms for P. Then we say $\mathcal{T}$ provides computable local error control of there is a sequence $\{\hat{\Gamma}_N\}_{N\in\mathbb{N}}$ of general algorithms $\Gamma_N \colon \Omega \to \mathbb{N}^k$ such that

$$d\left(\Gamma_{\hat{\Gamma}_N(A)}(A), \Xi(A)\right) < \frac{1}{N}$$

for all $N \in \mathbb{N}$ and $A \in \Omega$, where d denotes the metric associated to $\mathcal{M}$.

And indeed this is a property implied by global error control.

Remark 3.4. If $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ is a computational problems that can be solved by a tower of algorithms providing error control, then it can also be solved by a tower providing computable local error control, since the tower which is constructed in the proof of Theorem 3.2 has the trivial error control $\{\hat{\Gamma}_N\}_{N\in\mathbb{N}}$ with $\hat{\Gamma}_N \equiv N$ for all $N \in \mathbb{N}$.

Unfortunately, also computable local error control can not be achieved by towers of algorithms that solve problems of SCI ≥ 2.

Theorem 3.5. *Let $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ be a computational problem, such that* $\mathrm{SCI}(P)\alpha \geq 2$ *for some $\alpha \in \{G, A, R\}$ and*

$$\mathcal{T} := \{\Gamma_{n_1,\dots,n_m}\}_{(n_1,\dots,n_m)\in\mathbb{N}^m}$$

a tower of algorithms of type α for P. Then $\mathcal{T}$ does not provide computable local error control.

Proof. Assume there was a sequence $\{\hat{\Gamma}_N\}_{N\in\mathbb{N}}$ of general algorithms $\hat{\Gamma}_N \colon \Omega \to \mathbb{N}^k$ such that

$$d\left(\Gamma_{\hat{\Gamma}_N(A)}(A), \Xi(A)\right) < \frac{1}{N}$$

for all $N \in \mathbb{N}$ and $A \in \Omega$ and define $\bar{\Gamma}_N(A) := \Gamma_{\hat{\Gamma}_N(A)}(A)$. Then obviously $\bar{\Gamma}_N(A) \to \Xi(A)$ as $N \to \infty$ for all $A \in \Omega$. We show that $\{\bar{\Gamma}_N\}_{N\in\mathbb{N}}$ is arithmetic tower of algorithms for $(\Omega, \Lambda, \mathcal{M}, \Xi)$ and obtain a contradiction. To this end, it remains to prove that Γ_N is an arithmetic algorithm for all $N \in \mathbb{N}$. Let $N \in \mathbb{N}$ and set

$$\Lambda_{\bar{\Gamma}_N}(A) := \Lambda_{\Gamma_{\hat{\Gamma}_N(A)}}(A) \cup \Lambda_{\hat{\Gamma}_N}(A).$$

To prove Definition 1.3 of a general algorithm let $A, B \in \Omega$ be such that $f(A) = f(B)$ for all $f \in \Lambda_{\Gamma_N}(A)$. In particular, this holds for all $f \in \Lambda_{\hat{\Gamma}_N}(A)$ and thus $\hat{\Gamma}_N(A) = \hat{\Gamma}_N(B)$ by Definition 1.3. Further we conclude $f(A) = f(B)$ for all $f \in \Lambda_{\Gamma_{\hat{\Gamma}_N(A)}}(A)$ and since $\Gamma_{\hat{\Gamma}_N(A)}$ is a general algorithm also $\Gamma_{\hat{\Gamma}_N(A)}(A) = \Gamma_{\hat{\Gamma}_N(A)}(B) = \Gamma_{\hat{\Gamma}_N(B)}(B)$. Thus we have $\bar{\Gamma}_N(A) = \bar{\Gamma}_N(B)$. Finally $\bar{\Gamma}_N$ is of type α, since already $\Gamma_{\hat{\Gamma}_N(A)}$ was of type α. ∎

Again the statement and the proof of Theorem 3.5 is due to [1] for case of general algorithms. Interestingly, the proof of this theorem shows that computable local error control implies global error control, so we have in fact equivalent notions. Since for both it is necessary to have SCI ≤ 1 we just consider these.

Corollary 3.6. *Let $P = (\Omega, \Lambda, \mathcal{M}, \Xi)$ be a computational problem with $\mathrm{SCI}(P)_\alpha \leq 1$ for some $\alpha \in \{G, A, R\}$. Then P can be solved by a height one tower of algorithms of type α with global error control if and only if with computable local error control.*

Proof. The proof from global error control to computable local error control we already saw in Remark 3.4. For the other direction we assume the existence of a tower of algorithms of type α for P that provides computable local error control. Then we use the same construction as in Theorem 3.5 to obtain a tower of algorithms of type α for P that provides global error control. ∎

With this knowledge we summarise having Solvability Complexity Index less or equal to one is necessary for error control. Reviewing our computational problems from Chapter 2 which have an index greater than one we conclude that all of them are not solvable with an algorithm providing one of the error controls mentioned above. Thus we focus on the examples with index less or equal to one and see when we can achieve global error control.

First we consider the finite dimensional case.

Theorem 3.7. *Let $\Omega, \Lambda, \mathcal{M}$ and Ξ be as in Theorem 2.1 about computing spectra of finite matrices. Then there is an arithmetic tower of algorithms $\{\Gamma_n\}_{n\in\mathbb{N}}$ of height one for $(\Omega, \mathcal{M}, \Lambda, \Xi)$, such that for all $n \in \mathbb{N}$ and $A \in \Omega$ we have*

$$d_H(\Gamma_n(A), \Xi(A)) < \frac{1}{n}.$$

In particular, it is possible to have global error control within the arithmetic approximation of Ξ.

Proof. Let $\left\{\tilde{\Gamma}_k\right\}_{k\in\mathbb{N}}$ be the arithmetic tower of algorithms from the proof of Theorem 2.1. Further let $k \in \mathbb{N}$ and $A \in \Omega$. Then there is an $n \in \mathbb{N}$ such that $A \in \mathbb{C}^{n\times n}$ and by $\Gamma_k(A) := \tilde{\Gamma}_{k^{2n}}(A)$ we define an arithmetic algorithm. From the proof of Theorem 2.1 we deduce

$$d_H(\Gamma_N(A), \Xi(A)) < \frac{1}{N}$$

for all $A \in \Omega$ and all $N \in \mathbb{N}$. ∎

Unfortunately this result does not generalise to the compact operators as before, since we have the following far more general statement, where we restrict ourselves to projections.

Theorem 3.8. *Let $\Xi, \mathcal{M}, \Lambda$ be as in the Section 2.1 about computing spectra of bounded operators on $\ell^2(\mathbb{N})$ and $\Omega \subseteq L(\ell^2(\mathbb{N}))$ a family of operators containing 0 and the projections P_{e_n} onto the n-th component for arbitrary large $n \in \mathbb{N}$. Suppose $\mathcal{T} := \{\Gamma_n\}_{n\in\mathbb{N}}$ is any tower of algorithms of height 1 for $(\Omega, \mathcal{M}, \Lambda, \Xi)$.[1] Then $\mathcal{T}$ does not provide global error control.*

Proof. Towards a contradiction, we assume the opposite, that is global error control. Then there is an $N \in \mathbb{N}$ satisfying $d_H(\Gamma_k(T), \sigma(T)) < \frac{1}{2}$ for all $k \geq N$. In particular,

$$d_H(\Gamma_N(0), \{0\}) = d_H(\Gamma_N(0), \sigma(T \equiv 0)) < \frac{1}{2}$$

and hence $\Gamma_N(0) \subseteq B(0, \frac{1}{2})$. Let further M be the largest index, such that $f_{i,j} \in \Lambda_{\Gamma_N}(0)$ for all $i, j \leq M$. By assumption there is an $n \in \mathbb{N}$ fulfilling $n > M$ and $P_{e_n} \in \Omega$. We

[1] Note that this includes the case of a tower of height 0 where we would assume the existence of a general algorithm Γ satisfying $\Gamma = \Xi$.

obtain $f(0) = f(P_{e_n})$ for all $f \in \Lambda_{\Gamma_N}(0)$ and by Definition 1.3 of a general algorithm, we deduce $B(0, \frac{1}{2}) \supseteq \Gamma_N(0) = \Gamma_N(P_{e_n})$. But this is a contradiction, since by choice of N it is $d_H(\Gamma_N(P_{e_n}), \sigma(P_{e_n})) < \frac{1}{2}$ and thus $\Gamma_N(P_{e_n}) \cap B(1, \frac{1}{2}) \neq \emptyset$. ■

Since the class of compact operators fulfils the assumption of Theorem 3.8 we deduce the same negative conclusion.

Corollary 3.9. *We consider the computational problem $P = (\Omega, \mathcal{M}, \Lambda, \Xi)$ of computing spectra of compact operators from Theorem 2.7. Then there is no tower of algorithms for P providing global error control.*

Similarly in the case of inverse problems, even in very restricted situations where we just consider the identity no global error control is possible.

Theorem 3.10. *Let $\Xi, \mathcal{M}, \Lambda$ be as in Section 2.2 about computing well-posed inverse problems on $\ell^2(\mathbb{N})$ and $\Omega := \Omega_l \times \Omega_r \subseteq L_{inv}(\ell^2(\mathbb{N})) \times \ell^2(\mathbb{N})$ such that $I \in \Omega_l$ and $e_n \in \Omega_r$ for arbitrary large $n \in \mathbb{N}$ as well as $0 \in \Omega_r$. Then there is no tower of algorithms for $(\Omega, \mathcal{M}, \Lambda, \Xi)$ providing global error control.*

Proof. Suppose there is a tower of algorithms $\{\Gamma_k\}_{k\in\mathbb{N}}$ of height one for $(\Omega, \mathcal{M}, \Lambda, \Xi)$ providing global error control.[1] Then there is an $N \in \mathbb{N}$ satisfying $\|\Gamma_k(A, b) - A^{-1}b\|_{\ell^2} < \frac{1}{2}$ for all $k \geq N$, so that particularly $\|\Gamma_N(I, 0)\|_{\ell^2} < \frac{1}{2}$. Now let M be the largest index with $f_i \in \Lambda_{\Gamma_N}(I, 0)$ for all $i \leq M$. By assumption there is an $n \in \mathbb{N}$ with $n > M$ and $e_n \in \Omega_2$. Then we have $f(I, 0) = f(I, e_n)$ for all $f \in \Lambda_{\Gamma_N}(I, 0)$ and by the definition of a general algorithm also $\Gamma_N(I, 0) = \Gamma_N(I, e_n)$. But we have

$$\frac{1}{2} > \|\Gamma_N(I, e_n) - I^{-1}e_n\|_{\ell^2} \geq \|e_n\|_{\ell^2} - \|\Gamma_N(I, 0)\|_{\ell^2} > 1 - \frac{1}{2} = \frac{1}{2},$$

and therefore a contradiction. ■

And again all our operator classes and corresponding inverse problems from Chapter 2 for which we showed solvability within one limit fail to provide global error control.

Corollary 3.11. *We consider the computational problems $P_4 = (\Omega_4, \mathcal{M}, \Lambda, \Xi)$ from Theorem 2.10 and $P_5 = (\Omega_5, \mathcal{M}, \Lambda, \Xi)$ from Theorem 2.11 of computing solutions to inverse problems with unitary operators and lower triangular operators respectively. Then neither for P_4 nor for P_5 there is a tower of algorithms providing global error control.*

Theorem 3.10 implies also that one has to impose restrictions on the right hand side when dealing with inverse problems since even the most simple invertible operator does not allow error global control with arbitrary right hand sides. In fact, in Theorem 3.10 we assumed a lot structure of the right hand sides as we just considered unit vectors. We believe that requiring a sort of decay of the components of the right hand side is the way to obtain global error control.
Besides, another view of Theorem 3.10 delivers an additional negative result: We showed that the identity itself can not be computed with global error control.

4 Conclusions and Outlook

With those negative results from the previous chapter in mind one hopes also for positive ones.
In the case of inverse problems by Theorem 3.10 about error control of inverse problems using the identity this is only possible by restricting the class of operators *and* the right hand sides used. An even nicer result would be the determination of the exact class of pairs of operators and vectors for which error control is possible while solving the corresponding inverse problem, as well as a similar result for computing spectra.
Another way to obtain positive statements could also be to consider the weaker notions of error control which might also be satisfiable in the case of $\mathrm{SCI} \geq 2$.
In contrast, an even stronger notion of computable global error control would also be interesting, to divide the $\mathrm{SCI} = 1$ problems into smaller subclasses.

Apart from error control another interesting direction would be to extend the theory of inverse problem from Chapter 2 to ill-posed and ill-conditioned problems.
Further, the topic of computing spectra can be extended to unbounded operators. For the famous Schrödinger operator this is already done in [1]. But it might also be interesting to look at other examples of unbounded operators and maybe it is possible to characterise the complexity of (all) unbounded operators as in the bounded case.

Finally, for the general framework of the Solvability Complexity Index we look forward to working out more differences between higher SCI classes starting from two and to answer questions like: What makes a problem having $\mathrm{SCI} = 3$ reasonably harder than those with $\mathrm{SCI} = 2$? Are there properties which can be provided in the $\mathrm{SCI} = 2$ class and not above? If we compare $\mathrm{SCI} = 1$ with $\mathrm{SCI} \geq 2$ this gap is filled by error control and maybe another version of it does it for higher classes.

Bibliography

[1] J. Ben-Artzi, A. C. Hansen, O. Nevanlinna, and M. Seidel. Can everything be computed? - On the solvability complexity index and towers of algorithms. arXiv:1508.03280, 2015.

[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. Am. Math. Soc., New Ser.*, 21(1):1–46, 1989.

[3] S. A. Cook. The complexity of theorem-proving procedures. ACM, Proc. 3rd ann. ACM Sympos. Theory Computing, Shaker Heights, Ohio 1971, 151-158 (1971)., 1971.

[4] P. Doyle and C. McMullen. Solving the quintic by iteration. *Acta Math.*, 163(3-4):151–180, 1989.

[5] N. Dunford and J. T. Schwartz. Linear operators. Part II: Spectral theory. Self-adjoint operators in Hilbert space. With the assistance of William G. Bade and Robert G. Bartle. Pure and Applied Mathematics. Vol. 7. New York and London: Interscience Publishers, a division of John Wiley and Sons 1963. ix, 859-1923 (1963)., 1963.

[6] M. J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, 13(4):49–52, 1902.

[7] A. C. Hansen. On the solvability complexity index, the n-pseudospectrum and approximations of spectra of operators. *J. Am. Math. Soc.*, 24(1):81–124, 2011.

[8] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc. (2)*, 42:230–265, 1936.

[9] K. Weihrauch. *Computable analysis. An introduction.* Berlin: Springer, 2000.