Introduction

This book deals with computable analysis. The subject, as its name suggests, represents a marriage between analysis and physics on the one hand, and computability on the other. Computability, of course, brings to mind computers, which are playing an ever larger role in analysis and physical theory. Thus it becomes useful to know, at least theoretically, which computations in analysis and physics are possible and which are not.

In this book, we attempt to develop a coherent framework for solving problems in this area. We will see that a variety of questions in computable analysis can be answered within this framework. For example, we will deal with computability for classical analysis, mathematical physics, Hilbert and Banach spaces, bounded and unbounded linear operators, eigenvalues and eigenvectors, and a variety of other topics. All of these are dealt with from the viewpoint of recursion theory, the theory of computability as treated in mathematical logic. Classical recursion theory provides a definition of computable function from integers to integers. Starting from this, the book develops corresponding notions of computability for real numbers, continuous functions, Hilbert space, L^p-spaces, and, more generally, arbitrary Banach spaces.

The framework used in this book is axiomatic. We axiomatize the notion of a "computability structure" on a Banach space. This allows a variety of applications to be treated under one heading. It is worth mentioning that the concept axiomatized is "computable sequence of vectors" of the Banach space. Then a point x is computable if the sequence $x, x, x, \ldots$ is computable. However, it is natural, and in fact necessary, to deal with sequences rather than individual points. For sequences lie at the center, both of recursion theory and analysis. A cornerstone of recursion theory is the notion of a recursive function, which is nothing more than a computable sequence of integers. In analysis, the topology on a Banach space is given by sequences.

We turn now to a discussion of some of the principal results in the book. These results are contained in Parts II and III. (We will discuss the contents of the more elementary Part I below.) There are three key results, the First and Second Main Theorems and the Eigenvector Theorem.

The First Theorem (in Chapter 3) asserts that, with certain mild side conditions, bounded operators preserve computability and unbounded operators do not. That

is, a linear operator maps every computable element of its domain onto a computable element if and only if it is bounded. The side conditions are satisfied by all of the standard operators of analysis and physics. Hence we obtain a variety of applications merely by taking various well known linear operators and asking whether they are bounded or unbounded. Now, whether or not an operator is bounded, is a classical fact, which is usually well known. Thus, although the conclusion of the First Main Theorem is recursion-theoretic, the boundedness/unboundedness hypothesis is *not* dealt with in a recursion-theoretic manner. For this reason, the First Main Theorem is easy to apply.

Of course it is common practice for an analyst who is studying a particular linear operator to seek norms for which the operator is bounded. Our discussion indicates that this practice is not merely prudent, but necessary. In fact, the device of tailoring the norm to the problem at hand is seen—in a precise and not merely heuristic sense—as a necessary and sufficient means of preserving computability.

The Second Main Theorem (in Chapter 4) asserts that, under mild side conditions, a self-adjoint operator has computable eigenvalues, although the sequence of eigenvalues need not be computable. That is, although for each eigenvalue there is a program which computes it, different eigenvalues may require different programs. There is no master program which computes the entire sequence.

On the other hand, the Second Main Theorem does provide an explicit algorithm for computing individual eigenvalues. This algorithm applies to both bounded and unbounded self-adjoint operators. In particular, it applies to the standard operators of analysis and physics.

Incidentally, the side conditions needed for the Second Main Theorem are given in Chapter 4. Operators which satisfy these conditions are called *effectively determined*.

The Eigenvector Theorem (in Chapter 4), our third major result, asserts that there exists an effectively determined bounded self-adjoint operator T such that 0 is an eigenvalue of T of multiplicity one, but none of the eigenvectors corresponding to 0 is computable.

Relating the three major theorems, we note several contrasts. Begin with the First and Second Main Theorems. The first theorem asserts, in part, that unbounded operators do not preserve computability—i.e. that they map certain computable functions onto noncomputable functions. It may seem surprising, therefore, that all of the eigenvalues of an effectively determined unbounded self-adjoint operator are computable. In fact, the authors were surprised by this result. We had originally suspected that there should exist an effectively determined self-adjoint operator with noncomputable eigenvalues. In this connection, the fact that the eigenvalues *are* computable answers a question raised by Kreisel [1974].

There is a similar contrast between the Second Main Theorem and the Eigenvector Theorem, relating to the manner in which self-adjoint operators are used in quantum mechanics. As is well known, in quantum mechanics the operators correspond to "observables", the eigenvectors are associated with the states of the system, and the eigenvalues are related to the values actually measured. The Second Main Theorem asserts that the eigenvalues are computable, whereas the Eigenvector Theorem tells us that the eigenvectors need not be.

It is time to spell out a little more fully the contents of the book. We begin with Chapter 2, which deals with the axioms, still postponing our discussion of the elementary Chapters 0 and 1. The axioms play an important role throughout the book. As noted above, the concept axiomatized is "computable sequence of vectors". The axioms relate to the principal structures on a Banach space. We recall that a Banach space is a linear space, which is endowed with a norm and which is complete in terms of this norm. The axioms relate the notion of computability to the three basic concepts of Banach space theory. Thus there are three axioms—one for linearity, one for limits, and one for the norm. When viewed in this light, the axioms appear to be minimal. It turns our that, in most of the interesting cases, the axioms are also maximal. That is, under mild side conditions, the axioms determine the computability structure uniquely. We emphasize that these axioms define a "computability structure" on a preexisting Banach space. We do *not* define a "computable Banach space". All of these matters are discussed fully in the introduction to Chapter 2.

In Chapter 3 we prove the First Main Theorem and give a variety of applications of it. These applications are of two types. In the first type, we apply the First Main Theorem directly to some linear operator. For example, by this means, we show that the heat and potential equations preserve computability, but that the wave equation does not. Similarly, we determine precisely the cases in which the Fourier transform preserves computability and the cases in which it does not. (In particular, the Fourier transform preserves computability from L^2 to L^2, giving an effective version of the Plancherel Theorem.) A host of other applications can be given, merely by taking various standard linear operators of analysis or physics and applying the First Main Theorem to them. The second type of application deals with problems which do not appear to involve linear operators at all. An example is the clasification of those step functions which are L^p-computable ($p < \infty$). Here, although the statement of the theorem does not involve linear operators, the proof does. In fact, in these cases, the introduction of a suitable linear operator—to which the First Main Theorem can be applied—provides the key to the proof.

Chapter 4 deals with topics surrounding the Second Main Theorem and the Eigenvector Theorem. These two theorems have been discussed at some length above. The Second Main Theorem has a number of corollaries. For example, under the hypotheses of that theorem, there exists a bounded operator whose norm is a noncomputable real. However, if in addition the operator is compact, then the norm is computable, and moreover the entire sequence of eigenvalues is computable. It could be asked whether the Second Main Theorem extends to operators which are not self-adjoint, or to Banach spaces other than Hilbert space. The answer is no. We show by an example that even on Hilbert space, when the operator is not self-adjoint, then noncomputable eigenvalues can occur. The chapter contains a number of other results. Several of these are related to the Eigenvector Theorem. In particular, the lemmas used in proving that theorem provide a variety of techniques for dealing with computability questions for Hilbert and Banach spaces.

We remark that, although the Second Main Theorem is stated in Chapter 4, its proof is postponed until Chapter 5.

So far we have mostly discussed Parts II and III, which are the main parts of the book. Now we turn to the introductory Part I. Chapter 0 develops the computability theory of all of the topics, except differentiation, which occur in a standard undergraduate course in real analysis. It provides the basic prerequisites for reading research papers in computable analysis—at least in those cases where the reasoning is classical (see below). Chapter 1 treats differentiation, analytic functions, and a variety of more advanced topics. Both of these chapters are based on the standard Grzegorczyk/Lacombe definition of a computable continuous function, and they make no use of Banach spaces or axioms. Nevertheless, as we will see in Chapter 2, the Grzegorczyk/Lacombe definition does fit naturally into the Banach space framework of Parts II and III.

The book contains a brief Addendum which discusses some open problems.

Reviewing the contents of these chapters as outlined above, we observe that most of the results are not simply effectivizations of classical theorems. Nor are they counterexamples to such effectivizations. In this sense, our results have no classical analog. Of course, there are exceptions. For example, the effective Plancherel theorem, mentioned above, is clearly the effectivization of a classical theorem. In the same vein, the noncomputability of solutions of the wave equation means that a certain existence theorem fails to effectivize. However, most of our results do not fit this format. For instance, as we saw above, the First Main Theorem involves a combination of classical and recursion theoretic hypotheses, although it leads to recursion theoretic conclusions. The same can be said of the Second Main Theorem, the Eigenvector Theorem, and, in fact, most of the results in the book.

We observe that the reasoning in this book is classical. Recall that, at the outset, we mentioned the general problem of deciding which computations in analysis and physics are possible and which are not. From this perspective, it is natural to reason classically, as analysts and physicists do. In particular, we do *not* work within the intuitionist or constructivist framework—e.g. the framework of Brouwer or Bishop. For, just as classical recursion theory allows the use of nonconstructive methods to study computability, our approach to recursive analysis does likewise. Our objective is to delineate the class of computable processes within the larger class of all processes. In this, our viewpoint is analogous to that of the complex analyst, who regards the analytic functions as a special case of the class of all functions, but regards all functions as existing mathematical objects. Of course, we do not wish to deny the value of constructive modes of reasoning. Our purpose here is simply to state, as clearly as possible, the viewpoint adopted in this book.

We have deliberately written the book so as to require a minimal list of prerequisites. As noted above, all of the work in the book is based on the standard notion of a recursive function from $\mathbb{N}$ to $\mathbb{N}$. From logic, we require only a few standard facts about recursive functions. These facts are spelled out in a section entitled *Prerequisites from Logic and Analysis*. All further recursion-theoretic notions, such as the Grzegorczyk/Lacombe definition of computability for continuous functions, are developed from scratch. From analysis we need the following. In Part I, we require only standard calculus. In Parts II and III, some familiarity with the L^p-spaces, Hilbert space, and Banach space is required. While the analysis which we use goes further than this, all additional analytic concepts are defined, frequently with discussion and examples. In the same vein, because the book is written for

a mixed audience—including analysts and logicians—we have taken great pains to make our proofs clear and complete.

It may seen surprising to see an axiomatic approach used in connection with computability. Of course, axioms provide generality. The question is whether, by adopting an axiomatic approach, we lose the intrinsic quality traditionally associated with computability. In traditional recursion theory, for example, the notion of a recursive function from $\mathbb{N}$ to $\mathbb{N}$ is intrinsic. Although many different formulations have been given, they all lead to the same definition of computability. Actually, this intrinsic quality is largely preserved under the axiomatic approach. For, as mentioned above, under mild side conditions, the axioms determine the computability structure uniquely. Of course, the structure will be different for different Banach spaces. Yet, in each case, the axiomatic structure coincides with the natural one which has been extensively studied. For example, when applied to the Banach space of continuous functions on a closed interval, the axiomatic structure coincides with the classical Grzegorczyk/Lacombe definition of computability. For the L^p-spaces, p a computable real, $1 \leqslant p < \infty$, the axiomatic structure coincides with the natural definition of L^p-computability. (L^p-computability is defined in the obvious way—by taking the effective closure in L^p-norm of the Grzegorczyk/Lacombe computable functions.) Even in cases where relatively little work has been done up to now (e.g. Sobolev spaces) the axioms seem to provide a natural starting point. For more details, cf. Chapters 2 and 3.

Returning to L^p-computability, it turns out that certain discontinuous functions, and in particular certain step functions, are L^p-computable. At first glance, this seems to contradict a long standing perception that a computable function must be continuous. However, this perception depends implicitly on the assumption that the function is to be evaluated pointwise. In L^p theory—even classically—a function is never evaluated at a single point. For, as is well known, an L^p function is only defined up to sets of measure zero. So, instead of using pointwise evaluation, we use the L^p norm. Then the L^p-computability of certain step functions emerges, not as an ad hoc postulate, but as a consequence of the basic definition.

Although so far we have mainly discussed L^p spaces, many other computability structures on Banach spaces are related in a similar way to the standard Grzegorczyk-Lacombe definition. This holds, for example, for the energy norm and Sobolev spaces discussed in Chapter 3. However, there are other cases, e.g. those involving "ad hoc" computability structures, which bear no relationship to the Grzegorczyk-Lacombe definition. All of these cases, whether intrinsic or ad hoc, satisfy the axioms for a computability structure.

Although self-contained, the work in this monograph does not appear in a vacuum. There is a long tradition of research in recursive analysis using classical reasoning. Among those who have worked within this tradition are: Aberth, Grzegorczyk, Kreisel, Lachlan, Lacombe, Mazur, Metakides, Moschovakis, Mostowski, Myhill, Nerode, Rice, Robinson, Rogers, Shepherdson, Shore, Simpson, and Specker. Their work is cited in various places in the text, and also in the bibliography.

In the opinion of the authors, the field of computable analysis is in its infancy. There are numerous open problems, some hard and some easier. A small sample of these is given in the Addendum. Our hope is that this brief monograph will provide an easy introduction to the subject.

Prerequisites from Logic and Analysis

We begin with logic.

This book does not require any prior knowledge of formal logic.

First we expand upon some points made in the introduction to the book. All of our notions of computability (for real numbers, continuous functions, and beyond) are based on the notion of a recursive function a: $\mathbb{N} \to \mathbb{N}$ or a: $\mathbb{N}^q \to \mathbb{N}$. ($\mathbb{N}$ denotes the set of non-negative integers.) On the other hand, this book does not require a detailed knowledge of recursion theory. For reasons to be explained below, an intuitive understanding of that theory will suffice.

We continue for now to consider only functions from $\mathbb{N}$ to $\mathbb{N}$. Intuitively, a recursive function is simply a "computable" function. More precisely, a recursive function is a function which is computable by a Turing machine. The weight of fifty years experience leads to the conclusion that "recursive function" is the correct definition of the intuitive notion of a computable function. The definition is as solid as the definition of a group or a vector space. By now, the theory of recursive functions is highly developed.

However, as we have said, this book does not require a detailed knowledge of that theory. We avoid the need for heavy technical prerequisites in two ways.

1. Whenever we prove that some process is computable, we actually give the algorithm which produces the computation. As we shall see, some of these algorithms are quite intricate. But each of them is built up from scratch, so that the book is self-contained.
2. To prove that certain processes are not computable, we shall find that it suffices to know one basic result from recusive function theory—the existence of a recursively enumerable nonrecursive set. This we now discuss.

Imagine a computer which has been programmed to produce the values of a function a from nonnegative integers to nonnegative integers. We set the program in motion, and have the computer list the values $a(0), a(1), a(2), \ldots$ in order. This set A of values, a subset of the natural numbers, is an example of a "recursively enumerable set". If we take a general all purpose computer—e.g. a Turing machine —and consider the class of all such programs, we obtain the class of all recursively enumerable sets.

Suppose now that we have a recursively enumerable set A. Can we find an effective procedure which, for arbitrary n, determines whether or not $n \in A$? If $n \in A$, then

clearly we have a procedure which tells us so. Namely, we simply list the values $a(0)$, $a(1)$, $a(2)$, ..., and if $n \in A$, then the value n will eventually appear. The difficulty comes if $n \notin A$. For then it seems possible that we might have no method of ascertaining this fact. We can, of course, list the set A to an arbitrarily large finite number of its elements, and we can observe that the value n has not occurred *yet*. However, we might have no way of determining whether n will show up at some later stage.

It turns out that, in general, there is no effective procedure which lists the elements $n \notin A$. Thus, there is no effective procedure which answers, for every n, the question: is $n \in A$?

On the other hand, for some sets A, the question of membership in A can be effectively answered. This is true for most of the sets of natural numbers commonly encountered in number theory, e.g. the set of primes. Such sets are called "recursive". By contrast, those sets A for which we can ascertain when $n \in A$ but not when $n \notin A$, are called "recursively enumerable nonrecursive". It is a fundamental result of logic that such sets exist (Proposition A below).

We now spell out these definitions and results in a formal manner. These are the only recursion-theoretic facts which are used in this book.

A set $A \subseteq \mathbb{N}$ is called *recursively enumerable* if $A = \phi$ or A is the range of a recursive function a. When A is infinite, the function a can be chosen to be one-to-one.

A set $A \subseteq \mathbb{N}$ is called *recursive* if both A and its complement $\mathbb{N} - A$ are recursively enumerable.

A cornerstone of recursion theory is the following result. For a proof see e.g. Kleene [1952], Rogers [1967], Davis [1958], Cutland [1980], Soare [1987].

Proposition A. *There exists a set $A \subseteq \mathbb{N}$ which is recursively enumerable but not recursive.*

Occasionally we will have to use a slightly stronger result—the existence of a recursively inseparable pair of sets.

Proposition B. *There exists a recursively inseparable pair of sets, i.e. a pair of subsets A, B of $\mathbb{N}$ such that:*
a) *A and B are recursively enumerable.*
b) *$A \cap B = \phi$.*
c) *There is no recursive set C with $A \subseteq C$ and $B \subseteq \mathbb{N} - C$.*

We need one further result from logic—the existence of a recursive pairing function J from $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, together with recursive inverse functions $K: \mathbb{N} \to \mathbb{N}$ and $L: \mathbb{N} \to \mathbb{N}$ such that:

$$J[K(n), L(n)] = n.$$

In fact, a standard form for J is:

$$J(x, y) = \frac{(x + y)(x + y + 1)}{2} + x$$

A technical comment. When we use the *characteristic function* of a set, we follow the custom of analysts rather than that of logicians. Thus we define the characteristic function χ_S of a set S by

$$\chi_S(x) = \begin{cases} 1 & \text{for } x \in S, \\ 0 & \text{for } x \notin S. \end{cases}$$

Now we turn to analysis. We assume the contents of a standard undergraduate course in real variables, together with the rudiments of measure theory. We also need the following well known definitions.

A *Banach space* is a real or complex vector space with a norm $\| \ \|$ such that:

$$\|x + y\| \leqslant \|x\| + \|y\|,$$
$$\|\alpha x\| = |\alpha| \cdot \|x\| \quad \text{for all scalars } \alpha,$$
$$\|x\| \geqslant 0 \quad \text{with equality if and only if } x = 0,$$

and such that the space is complete in the metric $\|x - y\|$.

A *Hilbert space* is a Banach space in which the norm is given by an inner product (x, y): the form (x, y) is linear in the first variable and conjugate linear in the second variable, and it is related to the norm by $\|x\| = (x, x)^{1/2}$.

The following spaces are used frequently in this book. Except for $C^\infty[a, b]$, they are all Banach spaces.

$L^p[a, b]$: for $1 \leqslant p < \infty$, the space of all measurable functions f on $[a, b]$ for which the L^p-norm $\|f\|_p = \left(\int_a^b |f(x)|^p \, dx \right)^{1/p}$ is finite.

l^p: for $1 \leqslant p < \infty$, the space of all real or complex sequences $\{c_n\}$ for which the l^p-norm $\|\{c_n\}\|_p = \left(\sum_{n=0}^{\infty} |c_n|^p \right)^{1/p}$ is finite.

$C[a, b]$: the space of all continuous functions f on $[a, b]$, endowed with the uniform norm $\|f\|_\infty = \sup_x \{|f(x)|\}$.

$C^n[a, b]$: the space of all n times continuously differentiable functions f on $[a, b]$; here the norm can be taken as the sum of the uniform norms of $f^{(k)}$, $0 \leqslant k \leqslant n$.

$C^\infty[a, b]$: the space of infinitely differentiable functions on $[a, b]$.

By obvious modifications, the interval $[a, b]$ can be replaced by the real line $\mathbb{R}$ or euclidean q-space $\mathbb{R}^q$. For the space $C(\ )$, we use $C_0(\mathbb{R}^q)$, the space of continuous functions which vanish at infinity.

Finally, in the important case where $p = 2$, L^p and l^p are Hilbert spaces.