The Book Review Column [1]

by **Nicholas Tran** (ntran@scu.edu)
Department of Mathematics & Computer Science, Santa Clara University

1 Notable new releases

I am a fan of Sheldon Ross' textbooks on probability; they are concise and full of interesting exercises. He and Erol Peköz have just released a second edition of *A Second Course in Probability* (Cambridge University Press, 2023), which promises a rigorous but accessible and modern introduction to a selection of advanced topics in the field.

Javier Esparza's work includes using automata to study model checking, program analysis and verification. His book (coauthored with Michael Blondin) *Automata Theory: An Algorithmic Approach* (The MIT Press, 2023) emphasizes efficient constructions of (ω-)automata and other algorithmic concerns.

Filterworld: How Algorithms Flattened Culture by New Yorker staff writer Kyle Chayka (Doubleday, 2024) investigates the pervasive impact of algorithms on consumption and distribution of culture and explores ways to reclaim our freedom from the digital overlord.

2 This column

James V. Rauff taught an undergraduate course on quantum computing using *Quantum Computing: An Applied Approach* by Jack D. Hidary (Springer, 2021) and reports the challenges he faced with this textbook.

William Gasarch judges the arguments against and for doing applied mathematics put forth in *A Mathematician's Apology* by G. H. Hardy (Cambridge University Press, 2012) and *An Applied Mathematician's Apology* by Lloyd N. Trefethen (Society for Industrial and Applied Mathematics, 2022).

Mikael Vejdemo-Johansson finds a couple of redeeming features in *Algebra and Geometry with Python* by Sergei Kurgalin & Sergei Borzunov (Springer Nature Switzerland, 2021) despite its tenuous connection to Python.

David J. Littleboy and S. V. Nagaraj separately review *Mathematics in Computing: An Accessible Guide to Historical, Foundational and Application Contexts*, 2nd ed. by Gerard O'Regan (Springer, 2023). Both praise the book's breadth but are not impressed by its depth.

3 How to contribute

Please contact me to write a review! Either choose from the books listed below, or propose your own. In either case, the publisher will send you a free copy of the book. Guidelines and a LaTeX template can be found at `https://algoplexity.com/~ntran`.

BOOKS THAT NEED REVIEWERS FOR THE SIGACT NEWS COLUMN

Algorithms, Complexity, & Computability

1. Knebl, H. (2020). *Algorithms and Data Structures: Foundations and Probabilistic Methods for Design and Analysis.* Springer.
2. Chen, H. (2023). *Computability and Complexity.* The MIT Press.
3. Ferragina. P. (2023). *Pearls of Algorithm Engineering.* Cambridge University Press.
4. Esparza, J., & Blondin, M. (2023). *Automata Theory: An Algorithmic Approach.* The MIT Press.

Miscellaneous Computer Science & Mathematics

1. Nahin, P. (2021). *When Least Is Best: How Mathematicians Discovered Many Clever Ways to Make Things as Small (or as Large) as Possible.* Princeton University Press.
2. Chayka, K. (2024). *Filterworld: How Algorithms Flattened Culture.* Doubleday.

Data Science

1. Hrycej, T., Bermeitinger, B., Cetto, M., & Handschuh, S. (2023). *Mathematical Foundations of Data Science.* Springer.
2. Zhang, T. (2023) *Mathematical Analysis of Machine Learning Algorithms.* Cambridge University Press.

Discrete Mathematics and Computing

1. Harchol-Balter, M. (2023). *Introduction to Probability for Computing.* Cambridge University Press.
2. Ross, S., & Peköz, E. (2023). *A Second Course in Probability.* Cambridge University Press.

Cryptography and Security

1. Tyagi, H., & Watanabe, S. (2023). *Information-Theoretic Security.* Cambridge University Press.

Combinatorics and Graph Theory

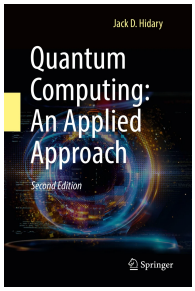
1. Beineke, L., Golumbic, M., & Wilson, R. (Eds.). (2021). *Topics in Algorithmic Graph Theory* (Encyclopedia of Mathematics and its Applications). Cambridge University Press.

2. Landman, B., Luca, F., Nathanson, M., Nešetřil, J., & Robertson, A. (Eds.). (2022). *Number Theory and Combinatorics: A Collection in Honor of the Mathematics of Ronald Graham.* De Gruyter.

Programming etc.

1. Sanders, P., Mehlhorn, K., Dietzfelbinger, M., & Dementiev, R. (2019). *Sequential and Parallel Algorithms and Data Structures: The Basic Toolbox.* Springer.

Review of [1]



Quantum Computing: An Applied Approach, 2nd ed.
by **Jack D. Hidary**

Springer Nature Switzerland, 2021
422 pages, Hardcover, $37.99, eBook, $29.99

Review by **James V. Rauff**
(jrauff@millikin.edu)
Department of Mathematics and Computational Sciences
Millikin University

1 Overview

Quantum Computing: An Applied Approach is an introductory textbook on quantum computing. The book features an extensive review of the mathematics underpinning quantum computing, a discussion of the classic algorithms of quantum computing, and implementations of the algorithms using Cirq, an open-source Python framework for Noisy Intermediate Scale Quantum (NISQ) algorithms. The mathematical chapters contain intertextual exercises with solutions. The book relies upon an associated GitHub site for updated code, additional exercises, and other resources.

2 Summary of Contents

The text consists of three parts comprising 15 chapters. Some of these chapters contain intertextual exercises with solutions.

Part I. Foundations

Chapter 1. Superposition, Entanglement and Reversibility.
A short outline of the basic elements of quantum physics relevant to quantum computing is given in this chapter. The topics herein include superposition, entanglement, the Born Rule (relating the amplitude of a quantum state to the probability of that state resulting after measurement), Schrödinger's equation, and reversibility. There are no exercises in the textbook, but some problems are posed in the GitHub resources.

Chapter 2. A Brief History of Quantum Computing.
This very short chapter highlights the contributions of Feynman, Deutsch, Vazirani, Bernstein, Simon, Shor, and Grover. It concludes with DiVincenzo's criteria for a quantum computer. Some problems for this chapter may be found in the GitHub resources.

Chapter 3. Qubits, Operators and Measurement.

This chapter lays the groundwork for the quantum algorithms to be presented later. Here the author defines a qubit and introduces Dirac notation along with quantum circuit diagrams. The fundamental quantum operators/gates (X, Y, Z, $R\varphi$, H, CNOT, CZ, Fredkin, and Toffoli) are explained and given in matrix form and circuit diagrams. A short description of the Bloch sphere is included. No exercises are given in the textbook, but there are several in the GitHub resources.

Chapter 4. Complexity Theory.

This very short chapter sketches the differences between the classical complexity classes P, NP, BPP and the quantum classes BQP, EQP, and QMA. A few problems are available in the GitHub resources.

Part II. Hardware and Applications

Chapter 5. Building a Quantum Computer.

This chapter discusses the leading paradigms of quantum computing hardware. The author views quantum processing units (QPU) as being used in combination with classical CPUs. Nuclear magnetic resonance (NMR) devices, nitrogen-vacancy (NV) center-in-diamond, photonics, and trapped ion approaches are included. There are two problems available in the GitHub resources.

Chapter 6. Development Libraries for Quantum Computer Programming.

A brief presentation of some quantum computing development libraries is the subject of this chapter. Included are Cirq (Google), Qiskit (IBM), Forest (Rigetti), and QDK (Microsoft). Code snippets for each library are presented. There are no exercises provided for this chapter.

Chapter 7. Teleportation, Superdense Coding and Bell's Inequality.

The eponymous quantum circuits are explained and implemented in Cirq. There are no exercises provided for this chapter.

Chapter 8. The Canon: Code Walkthrough.

The content of this chapter would probably form the heart of any beginning quantum computing course. It includes a description and Cirq implementation of the Deutsch-Jozsa algorithm, the Bernstein-Vazirani algorithm, Simon's problem (code on GitHub, but not in the text), the quantum Fourier transform, Shor's algorithm, and Grover's algorithm. The most detailed discussion is focused on Shor's algorithm, during which some exercises (with solutions) are given. Additional exercises are provided in the GitHub resources.

Chapter 9. Quantum Computing Methods.

Several quantum computing programs that can be run on NISQ processors are discussed in this chapter. Included are the variational quantum eigensolver, an application to quantum chemistry, the quantum approximate optimization algorithm (applied to the problem of computing the expectation of the cost Hamiltonian), quantum neural networks, quantum phase estimation, the HHL algorithm for solving linear systems of equations, a quantum random number generator, and quantum walks. All are implemented in Cirq. Some exercises are provided in the GitHub resources.

Chapter 10. Applications and Quantum Supremacy.

Quantum supremacy refers to a computational task that can be efficiently performed on a quantum computer beyond the capabilities of a classical supercomputer. This chapter discusses some of

the tasks used in demonstrating quantum supremacy. These include random circuit sampling and quantum error correction. There are no exercises provided for this chapter.

Part III. Toolkit

Chapters 11-14 of Part III provide a review of the mathematics of quantum computing. Each chapter has intertextual exercises. These chapters are referred to throughout the first ten chapters of the text. They are designed to provide the definitions, theorems, and techniques needed to write quantum computing algorithms. Here I will simply list the topics appearing in each chapter.

Chapter 11. Mathematical Tools for Quantum Computing I.
Basic vector operation (dot product, norm, etc.), complex numbers, inner product, polar form of complex number, matrices (multiplication, addition, transpose, conjugate), and tensor products.
Set theory, Cartesian product, functions, relations, composition, linear transformations, vector space, subspace, span, linear independence, bases, orthonormal bases, Abelian group, and fields.

Chapter 12. Mathematical Tools for Quantum Computing II.
Matrices associated with linear transformations, determinants, matrix inversion, eigenvectors, eigenvalues, Kronecker delta function, Hermitian operators, unitary operators, direct sum, tensor product, Hilbert space, and a summary of relationship between quantum computing and linear algebra.

Chapter 13. Mathematical Tools for Quantum Computing III.
Boolean functions and Euler's identity.

Chapter 14. Dirac Notation.
Using Dirac notation to represent vectors, vector operations, and tensor products.

Chapter 15. Table of Quantum Operators and Core Circuits.
A table of quantum operators and core circuits.

3 Opinion

I used *Quantum Computing: An Applied Approach* as a textbook for my *Introduction to Quantum Computing* course during the Fall 2023 semester. My students were junior and senior computer science majors who had taken a course in linear algebra and were proficient in Python. None of them had any previous experience with quantum physics or quantum computing.

In the introduction Hidary gives three options for using the book in university courses. The options are keyed to the type of student in the course: STEM majors, physics graduate students, or computer science graduate students. My demographic seemed to fit the author's description for a course in quantum computing for STEM majors as they were not graduate students in either physics or computer science. We covered the material in Chapters 1, 2, 3, 4, 6, 7, 8 and parts of 9 and 10. Because my students had already studied linear algebra, we were able to use Part III as a resource.

I was disappointed with the text. First, the quantum physics discussion in Chapter 1 was too sketchy and confusing for my students. My students responded better to explanations of

superposition and entanglement that were rooted in the matrix algebra of qubits. Hidary does use the polarizing filter example of superposition in Chapter 1, and that's fine. However, it was easier for my undergraduates to see what was happening in the context of linear algebra. Hidary does do some of that in 11.3, but it would improve the text, I think, to have that upfront. Hidary's explanation of entanglement on p. 8 is way too brief. Again, I would have preferred to see the concept arising from the linear algebra calculations. The first chapter could be much longer with some of the material in Part III incorporated. I used another text (*Quantum Computing for Everyone* by Chris Bernhardt) and provided my own supplements to clarify the notions of superposition and entanglement and the mathematics involved therein.

The problems in the GitHub resources were, except for Chapter 8, very difficult and often tangential to the chapter material. This was another disappointment. For example, the problems for Chapter 1 from the GitHub resources required concepts beyond those presented in the chapter. One problem for Chapter 2 is about the CZ-gate, which had not yet been defined. Most of the chapters in Part II had no intertextual exercises, so I needed to write my own.

I was also disappointed with the code provided. I am not surprised when printed textbook code is out-of-date or faulty, but I was hopeful that the GitHub resources would provide up-to-date code. That was not always the case. The faulty code was usually due to mismatches in class names in Cirq. For example, in the text the `Adder` class (p. 124) inherits from `cirq.ArithmeticOperation`, but that superclass is actually `cirq.ArithmeticGate`. My students and I needed to perform numerous searches on the Cirq website (`https://quantumai.google/cirq/start/basics`) to obtain proper syntax, as well as method and class names, in order to make the code execute properly. Fortunately, the Cirq code for the fundamental algorithms is also available on the Cirq website, along with tutorials.

On the positive side I liked how Hidary broke down some of the standard algorithms, like Shor's, into smaller code sections. That built up the algorithm using easy-to-digest pieces. The result was that my students understood the structure of the algorithm. Also, in general it was helpful to have the algorithms implemented rather than in pseudocode. I was able to guide my students to explore quantum computing ideas and experiment with minimal coding. To improve the text to make it more undergraduate-friendly, I would suggest expanding the first chapter to include the basics of the matrix theory of qubits, design more and easier exercises for each chapter intertextually or in the GitHub resources, and make sure the GitHub code is up to date with the current rendition of Cirq.

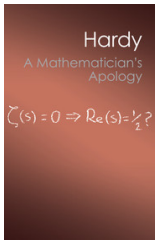
Perhaps graduate students in either physics or computer science would have a much easier go with *Quantum Computing: An Applied Approach* than my class did. The text contains an abundance of information on quantum computing. I think physics students would find Chapter 5 of particular interest. Graduate computer science students would be less bothered by code inconsistencies and be more able to appreciate the program design. Nevertheless, I cannot recommend the book for an undergraduate course.

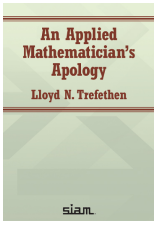
Joint Review of [1]

A Mathematician's Apology
by **G. H. Hardy**
Cambridge University Press, 2012
First published in 1940, Foreword by C. P. Snow added in 1967
$12.00 paperback, free online, 154 pages

and

An Applied Mathematician's Apology
by **Lloyd N. Trefethen**
Society for Industrial and Applied Mathematics, 2022
$36.00 hardcover, 80 pages





Review by **William Gasarch** (gasarch@umd.edu)

1 Introduction

In 1940 G. H. Hardy wrote *A Mathematician's Apology*, which argued why pure mathematics is a worthy endeavor. The book makes some other points and also seems to dismiss applied mathematics as dull. The book was around 80 pages, though with the later editions' long introduction by C. P. Snow, it is 154 pages.

In 2022 Nick Trefethen (he goes by his middle name Nick, not his first name Lloyd) wrote a book that looks at the tension between pure and applied mathematics, defends the latter against Hardy's dismissal, and also makes some criticisms of some types of applied mathematics. This book is also around 80 pages, so the same length as Hardy's book.

We review both books here and give an opinion of who is right. Spoiler alert: Trefethen.

2 A Mathematician's Apology

Hardy begins by saying that works *about math* (like the book he is writing, and that is why he is bringing this up) are somehow inferior to books that *are math*. In what sense? He doesn't really say, which is why I find this point of view misguided[2]. He seems to also think that people should do what they are good at, and that most people are not good at anything. An intelligent discussion about what people are good at, nature vs. nurture, what jobs require what talents, etc., would have been interesting. But none of that is here.

Hardy says that mathematics is a young man's game and claims that no great discoveries in mathematics have been made by anyone over 50. He gives as examples (1) Gauss, Newton, Painlevé (who?), and Laplace, which are fair examples, and (2) Galois, Abel, Ramanujan, Riemann, who all died before they were 50, so these are unfair examples. In my mind Hardy has made an *interesting conjecture* rather than an established fact.

Since Hardy's reasoning is speculative and not rigorous, I will do the same. The following people made important contributions after 50: Michael Rabin, Les Valiant, Avi Wigderson, Roger Apéry

[1] ©2024 William Gasarch

[2] I may be biased, since I've written over 100 book reviews, over 20 open problem columns, and write a blog, most of which are *about math*.

($\zeta(3) \notin Q$ at the age of 62), Yitang Zhang (bounded gaps between primes at the age of 58), Louis de Branges (proof of Bieberbach's conjecture at age 51). André Weil and Jean-Pierre Serre did high-level work past 60; in Serre's case, past 80. Rabin, Valiant, and Wigderson continued to do great work past 50. Valiant and Wigderson are still active.

This may be unfair to Hardy, since the people above were mostly not born before Hardy died. It would be an interesting history project to see which mathematicians before Hardy produced great work after 50. Archimedes may be an example, but our records from that time period are terrible. Euler is a better candidate. Also, Weierstrass proved his approximation theorem at the age of 70.

I believe that people over 50 are *capable* of doing great mathematics; however, there are reasons why it is rare:

1. (Andrew Gleason told me this one.) A mathematician works in one field for a long time and the field dries up: all of the open problems left are too hard to solve at that time. Since mathematics is an old field with much knowledge already built up, changing fields is hard. So the mathematician is stuck. This is why computer science, a newer field, has not had this problem as much.

2. The Peter Principle: Absola is such a great researcher, let's make her department chair, so she will have no time for research.

3. In Hardy's time people often died before 50.

And now for the two main courses: (1) Why does Hardy think pure math is a worthy endeavor? and (2) What does he think of applied math?

Pure math. Hardy says that while some math is useful (more on that later), that is not the real reason why math is worth doing. He gives as examples of beautiful mathematics (a) the proof that $\sqrt{2}$ is irrational, and (b) the proof that the primes are infinite.

He talks about why these theorems, and others, are beautiful and deeper than examples with small numbers (e.g., the theorem that 5 is the sum of 2 squares) or chess. He talks about significance, depth, and generality. He admits that these terms are hard to define. Okay, so in the end was all of this interesting or convincing? *Interesting?* Yes, it's good to see someone struggle, as we all do, to defend our field. *Convincing?* No. I already believed that pure math is worth doing for similar reasons that Hardy did, though he articulates them well. Even so, I doubt this would convince someone who had not already drunk the Kool-Aid.

Applied math. I can do no better than to give a quote which reflects what Hardy says in several parts of the book (I leave out some boring side comments):

> It is undeniable that a fair working knowledge of the differential and integral calculus has considerable practical utility. These parts of mathematics are, on the whole, rather dull. They are just the parts that have the least aesthetic value. The 'real' mathematics of the 'real' mathematicians, like Fermat and Euler and Gauss and Abel and Riemann, is almost wholly useless (and this is as true of 'applied' math as of 'pure' math).

He claims that relativity and quantum mechanics use interesting math (I would say they also motivated interesting math) but are not practical. He was right about that in his time, but wrong in the long term. And of course he thought number theory was interesting math that was not practical.

He was wrong about that, as it is now used heavily in cryptography. It would be unfair to criticize him for not seeing the future. However, he seems to not see the *synergy* that math has with other fields, even in his own time. Physics was the inspiration for much interesting mathematics. Games of chance were the inspiration for much of probability.

I view pure math, applied math, physics, and computer science as all interacting with each other with fuzzy boundaries; each has its interesting and dull parts. As time goes on we may add more fields to that list such as chemistry, biology, philosophy (I am thinking of logic, but there could be other parts), and history (see `https://blog.computationalcomplexity.org/2013/04/a-nice-case-of-interdisciplinary.html` for an example).

3 An Applied Mathematician's Apology

Nick Trefethen wrote this book to discuss what applied math is and why it is a worthy endeavor. The book mostly centers on numerical analysis. His main points are (1) pure math underrates applied math, (2) numerical analysis is *interesting*, and (3) some work in numerical analysis is not practical.

He makes point (1) by looking at the list of Fields Medal winners and noting that not only are none of them in applied math[3], but he has only read the work of one of them (Ahlfors). He also has some anecdotes of pure mathematicians disparaging applied mathematicians. However, my sense is that in recent years this has gotten better. Pure mathematicians are aware of, and respect, for example, the P vs. NP problem. And one of the Millennium Prize problems is squarely in applied math, dealing with the Navier-Stokes equations.

One odd point that may support his contention: Trefethen invented and guided the development of a numerical analysis package `Chebfun` from start to finish. This package is widely used and very practical. On his Wikipedia page there is no mention of `Chebfun`.

Regarding point (2), I suspect that for most of my readers the term *numerical analysis* evokes the thought *dealing with errors, ugh.* This is a misconception. The author defines the term:

Numerical analysis is the study of algorithms for problems in continuous mathematics.

Trefethen then gives very good examples of *interesting* results in numerical analysis. I give one here but also relate his story of how the study of it went off the rails.

Say you want to interpolate a function f by a polynomial p on the interval $-1 \leq x \leq 1$. Which points should you use? Equally spaced points are terrible. The Chebyshev points, which are clustered around -1 and 1, are excellent. This is interesting and was explained by Carl Runge in 1904. So far, so good.

But are the Chebyshev points optimal? The answer is no. Okay, then what are the optimal points? Bernstein made a conjecture about this in 1931 that was solved in 1981 by Kilgore and de Boor. Great! So can we now do interpolation much better? No. The improvement is 0%. Hence the work after Runge was academic and has no real-world impact. Yet even if it has no application, is it worth knowing? I leave that as an open question.

[3]The following Fields Medal winners are candidates for exceptions: Wendelin Werner (2006, probability theory and mathematical physics), Cédric Villani (2010, PDEs and mathematical physics), Martin Hairer (2014, stochastic partial differential equations), Alessio Figalli (2018, calculus of variations and PDEs), Hugo Duminil-Copin (2020, statistical physics). Note that the last one got his Fields Medal after the book was written.

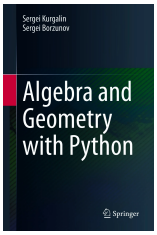
The book gives other examples of interesting applied mathematics and sometimes, even when work went astray, where that work led to. Also the book tells the author's own story of how he got into the field and how he thinks about research.

4 Opinion

Hardy's book has some interesting points to make, even if you disagree with them; however, he seems to go on and on at times before getting to his point, which is not worth it in the end. It's dull reading, almost as dull as he claims applied mathematics is. Should you read it? Oddly enough, yes, to see how one could try to defend pure math. Should you buy it? I borrowed mine from a colleague in the math department, and I recommend you do the same.

Trefethen's book is interesting. His life story is fascinating, and the math he talks about made me curious to learn more. He makes the point that numerical analysis (and to a lesser extent applied math) is fundamentally interesting. You should buy and read his book.

Mini-review of [1]



Algebra and Geometry with Python
by **Sergei Kurgalin & Sergei Borzunov**

Springer Nature Switzerland, 2021
425 pages, Hardcover, $99.99, Softcover, $64.99, eBook, $49.99

Review by **Mikael Vejdemo-Johansson**
(mvj@math.csi.cuny.edu)
Computer Science Program, CUNY Graduate Center,
Department of Mathematics, CUNY College of Staten Island

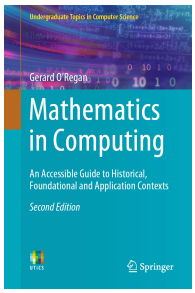
This is an introduction to elementary linear algebra and analytical geometry with some code snippets and references to performing computations in Python - suitable for a first-year course for STEM students. In the book, a reader will meet most of the topics expected in such a course: matrices, vectors and matrix algebra; matrix reduction to solve systems of linear equations; equations of straight lines and planes; structure of euclidean vector spaces with particular focus on $\mathbb{R}^3$. Towards the end, it goes into a selection of geometric applications of the principles developed earlier in the book: complex numbers, quantum computing, bilinear and quadratic forms with their vector-matrix representations $w(x) = x^T Ax$, generic quadratic curves, and finally even a foray into elliptic curves and their group structure.

Each chapter starts with the text developing the material, with concrete examples and worked calculations. This is followed by a good selection of review questions and then problems that range from mechanic training on the specific skills of the chapter to proof-based and exploratory problems that go into some depth in associated topics that were not covered in the text. They are followed by solutions to most (but not quite all) of the given problems. While making the book better for self-study through this choice, the solutions may decrease the usefulness of the book problems for homework in less trusting academic settings. Index and reference list are solid; no clear or obvious problems with either. The book also includes three appendices: very bare-bones intro to Python, a trigonometry cheat sheet, and the Greek alphabet. The Python appendix gives me the impression that the authors fully expect any reader to already be comfortable with programming.

The quantum computing excursion is more thorough than many short intros that I have seen so far, and it may well set the reader up for reading more thorough introductory texts with more success than without reading this chapter.

As an introduction to linear algebra it is not bad, but the Python connection is both outdated and sporadic at best. The authors give a few code snippets: exchanging values of variables, I/O, matrix addition, scalar multiplication and transposition, matrix multiplication, determinants, Hilbert matrices, Gaussian and Gauss-Jordan elimination, determining quadrant of a point, elliptic curve addition. They are writing the linear algebra Python code without any reference to the matrix multiplication operator (in Python since 2015) - much less any use of it - but since they also barely ever use their own matrix multiplication method, there is not that much lost by skipping it. The authors do say in the Preface that they are using "Python 3" without specifying a sub-version. Most remarkable in how little impact Python has on the book is the entire Section 11.5 with a title starting with "Algorithms" and containing not a single line of either code or algorithm pseudo-code.

Mathematics in Computing:
An Accessible Guide to Historical, Foundational and Application Contexts, 2nd ed.
by **Gerard O'Regan**



Springer Nature Switzerland, 2020
458 pages, Softcover

Mini-review by [1]
David J. Littleboy (djl@alum.mit.edu)
Technical translator, retired
Tokyo, Japan

1 Overview

I think this book succeeds and would be useful as a text for an introductory and/or overview course for the mathematics used in computer science.

The book consists of 28 (yes, 28!) short chapters, each one of which consists of a short introduction, an exposition of the topic at hand, some review questions, and a summary. Each chapter could be covered in one or two lectures plus a TA-led session, and the professor could pick and choose from the later chapters. The questions are excellent, mostly conceptual, with an occasional calculation problem to keep the students on their toes.

The author deserves praise for his content selection. The first three chapters are necessary background/introductory material, and then individual chapters cover individual topics: algorithms, number theory, algebra, combinatorics, and a lot more. I was particularly pleased that the author covers both number

Mini-review by [1]
S. V. Nagaraj (svnagaraj@acm.org)
Vellore Institute of Technology
Chennai Campus, India

1 Summary

This book, authored by a prolific writer, is about the applications of mathematics in computing and places emphasis on historical contexts where applicable. Aimed at undergraduate students of computer science, the book provides helpful pedagogical features: key topics covered in each chapter are listed at the beginning, followed by an introduction, a summary, questions for review, and references at the end. Its second edition includes 28 chapters and runs to 458 pages, a significant expansion of the first edition, which has 16 chapters and runs to 288 pages. The chapter on foundations of computing introduces binary numbers, early computers such as the analytical engine, and symbolic logic. The chapter on algebra looks very briefly at structures such as monoids, groups, rings, fields and vector spaces. The chapter on coding theory focuses on block codes. The chapter on advanced topics in logic contains a brief description of

theory and abstract algebra, not just as incidental material relevant to computer science, but as independent intellectual areas. I also liked the author's example of a proof using strong induction. I've seen too many books that fail to explain where strong induction is needed and used.

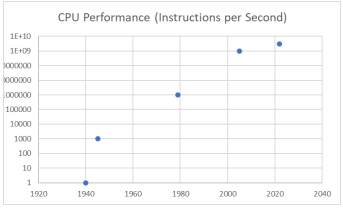
While I did notice some editing issues and typos, I like both the overall design of the book and the content covered. The writing style is on the dense, punchy, and declarative side, but not excessively so. Lastly, the book makes a point of introducing and crediting the figures who made the major contributions to the areas discussed, that is, it keeps its eye on the historical and foundational aspects it aims to.

2 A Historical Aside

While I strongly believe that the brevity of the chapters in this book is a feature, it doesn't allow space for deep historical analysis. In the introductory historical material, the author emphasizes the uninspired performance of early digital computers. How slow were they, actually?

While electro-mechanical computers could perform roughly one operation per second, the early tube computers could perform around 1,000 operations per second. That's three order of magnitude. Not too shabby.

The next gain of three orders of magnitude took another 30 years for mainframes and another 10 years after that for personal machines. And it took another 20 years for the next three orders of magnitude speed improvement (see the plot below); many commentators see those 20 years as a period of unequaled technological progress.



fuzzy logic, temporal logic, intuitionistic logic and undefined values. The chapter on overview of formal methods discusses topics such as the need for formal methods, approaches, the Z specification language, the B method, and model checking. The chapter on automata theory is very brief, running to just ten pages; finite state machines, pushdown automata and Turing machines are considered. The epilogue essentially summarizes the concepts discussed in the chapters. The list of figures, glossary, and the index are helpful. The bibliography surprisingly has just one reference by the same author.

2 Opinion

It was not easy to find out what changes were made in the second edition of the book. The companion website gives some information, although inadequate: "This fully updated new edition has been expanded with a more comprehensive treatment of algorithms, logic, automata theory, model checking, software reliability and dependability, algebra, sequences and series, and mathematical induction."

Since 458 pages span the 28 chapters, readers will find that some of the chapters are very abbreviated. However, given the fact that entire books have been devoted to the topics covered in each of the chapters, and that the book is aimed at undergraduate students in computer science, the shortcoming may be excused. The book achieves its objective of providing a flavor of the mathematics used in computing, so the general reader will likely benefit from it as well.