INTRODUCTION TO LITERATE SCIENTIFIC COMPUTING

1. Scientific computation

The observation of regular patterns in everyday events led the Ancient Greeks to ascribe the term μάΘημα (máthēma) meaning "learning" to this human endeavor. Different types of "learning" were separated into distinct disciplines, such as φυσική (physikē, literally "nature", whence physics) that deals with the natural world. Mathematics coalesced as the study those abstract patterns that could not be dipatched into specific disciplines such as chemistry, biology, or pschology. Although no consensus has arisen on a definition of "mathematics", some of the earliest and time-honored methods to seek patterns arose from counting (arithmetic), calculation (operating on input quantities, the Latin *calculus* is a small pebble as used in an abacus), and measurement (comparison to some standard object). All of these fall within the realm of computational mathematics, a field of study that stretches from Antiquity to the myriad developments enabled by analog, digital, and quantum computers today.

Computational mathematics can be carried out with numbers (rationals $\mathbb{Q}$, reals $\mathbb{R}$, approximations of reals known as floats $\mathbb{F}$), algebraic structures (computational group theory), symbols (symbolic computaiton), or abstract sets among many other possibilities. This text is mainly concerned with numerical computation, nowadays thought of as a distinct field of applied mathematics. Within this overall field, *Numerical Methods* tends to focus on the development of computational procedures or algorithms, while *Numerical Analysis* studies the properties of such algorithms applying tools from a variety of mathematical disciplines. Neither of these disciplines fully captures the intent of scientific computing since, though often inspired by applications, they tend to focus on abstract concepts in a mathematical or computer science spirit.

An impactful feature of the Modern Age is the rapid introduction of technological advances in computer construction that enables the simulation of natural phenomena with great fidelity. For example, the traditional means of ascertaining aircraft performance was to construct a model at reduced scale and measure aerodynamic forces when placed in a wind tunnel. While such physical experiments and flight tests remain the true validation of a particular design, initial steps in new aircraft construction now typically use "virtual wind tunnels", or computational simulation of the flow around an aircraft. Similar developments are observed in drug design or development of new materials, among many other fields in which computational simulation is significantly aiding researchers. Indeed such computational modeling has developed into a distinct method of scientific inquiry, complementing the traditional methods of theory construction and experimentation.

Scientific computation is a field of inquiry that blends elements of multiple fields of science to construct computational representations of natural phenomena. Concepts are drawn from mathematics, computer science, information theory, and various branches of science. Whereas mathematics is often concerned with aspects such as internal rigor and abstract concepts not necessarily linked to natural phenomena, scientific computation typically adopts the scientific method: acquiring knowledge by observation, formulation of hypotheses, and testing against experiments. A particularity of scientific computation is that the hypothesis concern not only the theoretical framework, say Newtonian mechanics, but also the implementation into executable computer code. Validation of the hypotheses is carried out through numerical experiments, to be compared against analytical predictions or physical measurements.

2. Literate programming

An essential part of science is the ensemble of practices meant to communicate knowledge. The scholarly method includes clear formulation of assumptions, statement of methods of acquiring experimental data, reproducibility of conclusions, and citation of sources. The term itself has an interesting history arising from the desire of medieval philosophers to reconcile contradictions they noticed in classical sources, especially Plato (429-347 BCE) and Aristotle (384-322 BCE), with their Catholic world view. Plato, influenced by Socrates (469-399 BCE), had attacked the teaching of *rhetoric*, or art of persuasive speaking featured in political discourse, distinct from reasoned dialogue or *dialectic*. The founders of Scholasticism, Johanes Scotus Eriugena (815-877), Anselm of Canterbury (1034-1109),

and especially Peter Abelard (1079-1142), further developed the dialectic method of argumentation leading in due course to early formulations of the scientific method by Albertus Magnus (1206-1280), Thomas Aquinas (1225-1274), and Roger Bacon (1220-1292).

Communication of acquired knowledge is also an important part of scientific computing, and has evolved considerably since the introduction of ENIAC, the first electronic, general-purpose computer in 1945. At that time, the principal communication problem was specifying the steps ENIAC should carry out to compute some quantity of interest. This was initially accomplished through manipulation of physical switches and cable connections (Fig. 1). This burdensome method was soon replaced by specification of instructions directing how to move data between general storage (memory) to specific storage locations where it could be processed (e.g., an accumulator). Here's an example from the IBM 704 computer introduced in 1954:

```
CLA 4  clear contents of accumulator and store value from memory location 4
ADD 5  add the value from memory location 5 to the accumulator
STO 6  store the value in accumulator to memory location 6
```

The above is still a tedious specification of the concise expression $z = y + w$, as was early recognized leading to the development of FORTRAN (FORmula TRANslator), the first general-purpose scientific computing language that allowed a syntax very close to the mathematical formula (input to a computer through punched cards Fig. 1)

```
Z = Y + W
```

Fortran is still in widespread use, though it has since evolved enormously [7], and been joined by a number of programming languages suitable for scientific computing such as C/C++ [2] and Python [6].

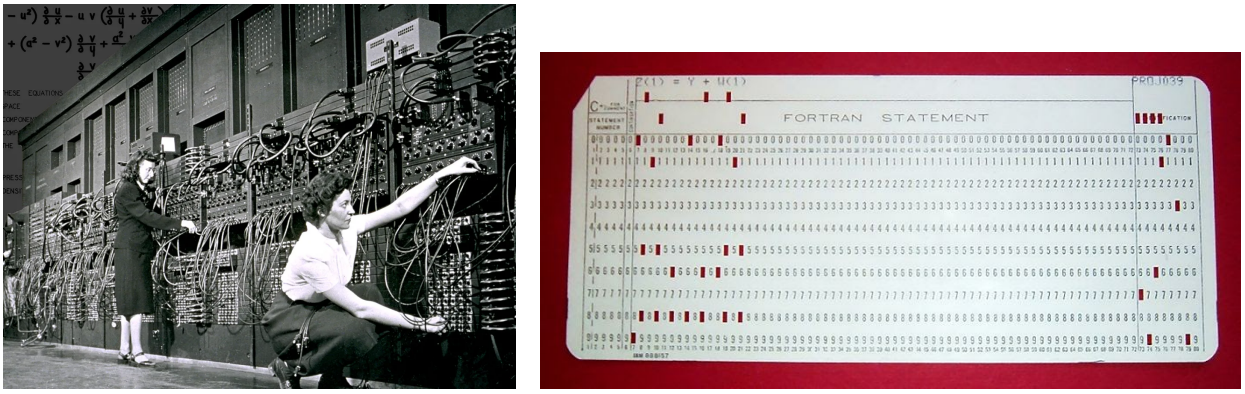

Figure 1. (Left) Early programmers of ENIAC [3]. (Right) Punch card containing FORTRAN statement `Z(1)=Y+W(1)`.

A common feature of all the above programming languages is that they are meant for one-directional communication from a human to a machine. Steps in this human-to-machine communication included translating the human-readable code to machine instructions, a process known as *compilation*, and use of code libraries, a process known as *linking*. Comprehension by another human is not considered, leading to difficulty in interpreting the intent of code such as

```
N = 20
T = 1.0
S = 1
P = 0.0
DO K=0,N-1
  P = P + S/T
  T = T + 2
  S = -S
```

```
END DO
P = 4*P
```

The above Fortran code implements a procedure to compute the irrational mathematical constant π using only operations involving rational numbers known as the Leibniz series

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots,$$

with the approximation after n terms

$$\pi \cong p_n = 4 \cdot \sum_{k=0}^{n-1} s_k = 4 \cdot \sum_{k=0}^{n-1} (-1)^k \frac{1}{2k+1}, s_k = (-1)^k \frac{1}{2k+1}.$$

Donald Knuth [10] recognized that combining programming and documentation would make "… programs more robust, more portable, more easily maintained, and arguably more fun to write …". He called the approach literate programming, and wrote software (the `WEB` system) to document C-language code using his TEX [4] typesetting system for documentation. While TEX, and especially its LATEX derivative [5], have been widely adopted, literate programming has not entered the mainstream of scientific computation, perhaps due to the rather arcane, stream-of-programmer-conciousness approach of the `WEB` system.

3. Literate scientific computing

The further development of both software and hardware since the first formulation of literate programming now allows an approach much closer to scientific communication, called here *literate scientific computing*, and presented in this text through projects from a variety of sciences. The main components of this approach are:

1. Use of an editing platform that allows simultaneous drafting a scientific paper describing the approach and implementation into code. The TEXMACS [9] system is adopted here.
2. Direct inclusion into the document of code that verifies and validates the computational approach ensuring reproducibility of the results as stipulated by the scientific method. The introduction of the Julia programming language [11, 8, 1] allows elegant expression of mathematical algorithms into efficient computer code as explored in this text.

As an example, reconsider the Leibniz series

◦ $\pi \cong p_n = 4 \cdot \sum_{k=0}^{n-1} s_k, s_k = (-1)^k \frac{1}{2k+1}.$

The above formula is actually the first part of a *fold* within this TEXMACS document. Expanded out (by clicking the ◦ fold symbol), the implementation as a Julia programming language function, and typical results appear.

• $\pi \cong p_n = 4 \cdot \sum_{k=0}^{n-1} s_k, s_k = (-1)^k \frac{1}{2k+1}.$

```
∴ function leibniz(n)
    k=0:n-1; s=(-1).^k ./ (2*k.+1)
    4*sum(s)
  end
```

```
leibniz
```

```
∴ leibniz.(25:25:100)'
```

[3.1815766854350316 3.121594652591011 3.15492539446215 3.1315929035585524] (2)

```
∴ leibniz(5000)
```

3.141392653591795

The technicalities of Julia programming will be incrementally introduced in this text. The more important fact is that the presentation of the method to approximate π contains the implementation and is directly executable in this document itself. The implementation can be hidden away to concentrate on the algorithm or mathematics, and can also be extracted into a separate code file if needed.

Literate scientific computing, as understood in this text, is the practice of integrated development of mathematical models and implementation code for description of natural phenomena. This approach is demonstrated in this text through a succession of projects drawn from various scientific fields:

1. *Population models*. Discrete and continuous modeling of the evolution in time of population of interacting species.
2. *Crystals, quasicrystals, and tilings*. Modeling of periodic and aperiodic geometrical structures that can fill space.
3. *Image processing*. Representation of visual information numerically together with procedures to synthesize new visual objects.
4. *Random walks and diffusion in physics and finance*. A study on the consequences of observational scale, relating diffusion of indistinguishable objects at a large observation scale, to random motion of those objects at a smaller observation scale.
5. *Monte Carlo, legislative redistricting*. Use of stochastic techniques to study packing of objects, ranging from molecules to construct materials to "packing" of legislative districts.
6. *Graph Laplacian and spectral clustering in machine learning*. An introduction to the basic problem within machine learning of object classification.
7. *Flocking, schooling, and folding*. Cooperative behavior between interacting entities, be they biological (fish and birds) of inanimate (amino acids in a protein).

BIBLIOGRAPHY

[1] Tobin A. Driscoll and Richard J. Braun. *Fundamentals of Numerical Computation*. SIAM-Society for Industrial & Applied Mathematics, Philadelphia, dec 2017.

[2] Joe Pitt Francis and Jonathan Whiteley. *Guide to Scientific Computing in C++*. Undergraduate Topics in Computer Science. Springer-Verlag, London, 2012.

[3] W.B. Fritz. The women of ENIAC. *IEEE Annals of the History of Computing*, 18(3):13–28, 1996. Conference Name: IEEE Annals of the History of Computing.

[4] Donald E. Knuth. *TeXbook, The*. Addison-Wesley Professional, Reading, Mass, Spi edition edition, jan 1984.

[5] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Professional, Reading, Mass, 2nd edition edition, jun 1994.

[6] Hans Petter Langtangen. *A Primer on Scientific Programming with Python*. Texts in Computational Science and Engineering. Springer-Verlag, Berlin Heidelberg, 5 edition, 2016.

[7] Michael Metcalf, John Reid, and Malcolm Cohen. *Modern Fortran Explained: Incorporating Fortran 2018*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 5 edition, 2018.

[8] Introduction to Applied Linear Algebra – Vectors, Matrices, and Least Squares.

[9] The Jolly Writer - Presentation.

[10] Knuth: Literate Programming.

[11] Think Julia [Book]. ISBN: 9781492045038.