Article type: Overview

Where do Computational Mathematics and Computational Statistics Converge?

Austen C. Duffy

Computational Mathematician, Citilabs

Keywords

Computational Mathematics, Computational Statistics, Linear Least Squares, Monte Carlo Methods, Machine Learning

Abstract

This overview presents a discussion on several topics of importance shared by the fields of computational mathematics and computational statistics. A brief history of the advancements to the fields of mathematics and statistics leading to the modern computational era is provided, along with a discussion of the intersection of topics which comprise the two subjects. The foundational aspects shared by both computational mathematics and computational statistics are explored with elementary discussions suitable to non-experts and aspiring students of the computational sciences. Finally, a discussion of the role played by computational mathematics and computational statistics in a few selected application areas is explored.

An Introduction to Computational Mathematics and Computational Statistics

The various fields that comprise the greater forum of computational science possess a certain core of methodologies necessary for producing numerically founded solutions. Since the advent of the first computers, the fields of computational mathematics and computational statistics have rapidly emerged as distinct disciplines making many important contributions to this area. While each subject possesses it's own primary studies, computational mathematics and computational statistics share common ground owing to the close relationship of the underlying subjects. In this paper we will examine the areas and ways in which the disciplines of computational mathematics and computational statistics converge. The first step in this process will be to define what exactly computational mathematics and computational statistics are and how they relate to their more theoretical parent subjects.

The subject of computational mathematics is primarily concerned with ways in which to efficiently compute numerical solutions to mathematical problems

which may be difficult or impossible to solve otherwise. Computational mathematics can be viewed as the application of the theory of numerical analysis where algorithms must be developed to adhere to the theoretical solution properties of convergence and stability in a finite precision environment, and implemented in a manner which allows for feasible computation with limited computing resources. In computational mathematics, feasible computation may require the use of limited memory algorithms or parallel computation to reduce run times. In the modern many-core computing era parallelization is a vital aspect, and some introductory texts covering the tools of computational mathematics teach parallel computing alongside traditional material such as the text of Karniadakis and Kirby [35]. The practicing computational mathematician is typically well versed in high performance computing and parallel algorithms, as they are often in the business of producing high performance mathematical software. We note here that the use of the term scientific computing is often synonymous with that of computational mathematics, as they cover essentially the same material. Computational statistics shares many of the features of computational mathematics. The Handbook of Computational Statistics [27] describes the subject as one that brings together statistical computing (which [27] lists as numerical analysis, database methodology, computer graphics, software engineering and the computer/human interface) and statistical methods which are computationally heavy. Much like computational mathematics, computational statistics is a subject entrenched in the analysis and application of numerical methods.

A Brief History of Computation Leading to the Era of Electronic Computers

While the primary scope of this paper will focus on topics more relevant to the modern era, it is important to note the origins of computational mathematics and computational statistics and the roles early advancements played in the evolution of the subjects. To be certain, mathematical computation was carried out with the help of machines long before the advent of the modern computer with devices as simple as the abacus and slide rule to more complex mechanical machines which could quite literally crank out the four basic operations of mathematics (explore for example the “Analytic Engine” of Babbage [43]). Watnik [71] provides a concise summary of the early history of computational statistics in which he relays an interesting story concerning the first use of the word “computer” by the *Journal of the American Statistical Association* which he attributes to Porter [48]. In this work, Porter references a machine designed by Herman Hollerith for tabulating census statistics by using a sorted punch card system and, as Watnik notes [71], despite a significant increase in the U.S. population the machine aided tabulation was completed in less than a year despite the previous census requiring seven years to complete the task. Archaic computing machinery though primitive by today’s standards first showed the potential for carrying out tedious mathematical computation on a large scale and helped pave the way for modern computational sci-

ence.

World War II created a massive impetus to the fields of computation and helped forge what would become the modern computer. This was an invention of necessity as mathematicians led by the likes of Alan Turing raced to break the German codes. The inspiration of Turing machines [67] would lead to the invention of modern electrical computers starting with the Electronic Numerical Integrator and Computer (ENIAC) in 1946, giving the calculation heavy fields of mathematics and statistics a powerful new tool which would allow them to push the limits of numerical algorithms. It was also in 1946 that researchers at Los Alamos National Laboratory began the development of Monte Carlo methods, a subject that has carried on with vital importance to both the fields of computational mathematics and computational statistics for its wide ranging uses from computing event probabilities, to chance simulations, to numerical integration. An excellent history of the development of the Mathematical and Numerical Integrator and Computer (MANIAC) which helped spur the development of the Monte Carlo method can be found in [2] which also briefly discusses ENIAC and other early computers.

With the advancement of the modern computer, numerical experiments could be carried out that could validate mathematical and statistical theory, simulations could be used to predict the results of laboratory experiments, and laboratory experiments could be used to improve mathematical and statistical models. The need for advanced computational capabilities drove the advancement of modern computers, and with their emergence in the post World War II era computational mathematics and computational statistics would emerge as essentially new subjects determined to push the limits of what these machines could do.

Foundations of Computational Mathematics and Computational Statistics

Though the subjects of mathematics and statistics are distinct in many respects, it is clear that the two are intertwined through their common bases of algebra and calculus. Just as every mathematics student must learn calculus, so must the statistics student. The subjects of computational mathematics and computational statistics are no different in this respect, requiring practitioners of both disciplines to master a certain set of basic skills necessary to solve the many problems arising in both subjects. Here we will examine what computational tools comprise that basic skill set by examining the material which is considered "foundational" to both. To make a valid argument for topics covered by the two subjects we will look at some popular introductory texts of each for a common ground. Here we examine the intersection of those topics taught through traditional computational mathematics introductory texts such as [31, 8, 49] and those covering computational statistics like [28, 26]. While these texts, along with the intro-

ductory texts of essentially all computational disciplines, cover the rudimentary subject of the machine representation of numbers, computer arithmetic and the associated error analysis, we will dedicate this section to the conversation of numerical linear algebra, nonlinear equations, numerical integration and differentiation, numerical optimization and Monte Carlo methods.

Numerical Linear Algebra

The ability to solve linear systems of equations is perhaps the most vital tool in all of computational science and engineering, as many relationships in nature are linear while those that are not can frequently be transformed as such [31]. The processes and discretizations that result in large linear systems of equations can be found in essentially every mathematics based discipline. One of the most frequently encountered problems in this area is the solution to the matrix equation $Ax = b$ where A is a square coefficient matrix, x is a vector representing the solution variable and b is a constant vector frequently referred to as the right hand side (RHS). Much of numerical linear algebra deals with the various methods used to solve this equation from straightforward Gaussian elimination whose techniques one may encounter early on in high school algebra to more algorithm friendly matrix decompositions such as transforming A into the product of a lower and upper diagonal matrix such that $LUx = b$. This results in a system which can be solved with a sequence of forward and backward solves allowing to the decomposed matrices structures. In this manner one would introduce an intermediate variable y which can be found by forward solving $Ly = b$ and then upon arriving at the solution one is ready to solve the equation $Ux = y$ by a backward solve to obtain x.

There exist a wide variety of matrix decompositions that can be computed and may lend themselves to special matrix types, and the young computational mathematician spends many hours studying and developing algorithms for computing them (Interested readers can examine [29, 59] for a thorough collection of algorithms along with [47] for development of parallel and vector versions). One of the most common matrix structures arising from the discretization methods for systems of partial differential equations which dominate the world of physical modelling is the symmetric positive definite matrix. By definition, a symmetric positive definite matrix A is one that satisfies the inequality $x^T Ax > 0$ for any vector x and is of course symmetric as the name implies. A popular decomposition known as the Cholesky decomposition takes advantage of this special matrix structure to produce a system $LL^T x = b$ which requires half the storage of an LU decomposition. As an interesting note, determining whether or not a 'random' matrix with no discernible structure (aside from symmetry) is symmetric positive definite is a difficult task, but can however be determined by attempting a Cholesky factorization and seeing whether or not it fails. The existence of a unique Cholesky decomposition is an equivalent definition of a symmetric positive definite matrix, and an additional equivalence property for positive definite matrices is that all of its eigenvalues are positive. Other special forms of matrices include banded matrices that contain all 0 entries outside of the diagonal and a fixed number of adjacent super

and sub-diagonals and sparse matrices which contain more zero entries than non-zero entries. Banded matrices are particularly nice as one can develop direct algorithms for their solution and they can be efficiently stored. Sparse matrices are frequently stored in special structures omitting zero entries and utilize algorithms that take advantage of their sparse structure to eliminate operations on 0 entries. Some common matrix structures are shown in figure 1.

$$a)\begin{bmatrix} 1 & 0 & 2 & 3 & 5 \\ 8 & 6 & 7 & 8 & 9 \\ 0 & 2 & 4 & 5 & 2 \\ 5 & 6 & 1 & 8 & 3 \\ 1 & 6 & 7 & 0 & 9 \end{bmatrix} \quad b)\begin{bmatrix} 0 & 0 & 2 & 3 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 8 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad c)\begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

$$d)\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 8 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 5 & 6 & 1 & 1 & 0 \\ 1 & 6 & 7 & 0 & 1 \end{bmatrix} \quad e)\begin{bmatrix} 3 & 9 & 2 & 3 & 1 \\ 0 & 6 & 3 & 3 & 7 \\ 0 & 0 & 4 & 5 & 2 \\ 0 & 0 & 0 & 8 & 4 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix} \quad f)\begin{bmatrix} 2 & 1 & 3 & 4 & 0 \\ 1 & 2 & 1 & 3 & 4 \\ 0 & 1 & 2 & 1 & 3 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

Figure 1: Types of Matrices. a) Dense Matrix: Contains mostly non-zero entries. b) Sparse Matrix: Contains mostly zero entries. c) Banded Matrix: Contains non-zero entries only on a fixed number of upper and lower "bands" from the diagonal. Banded matrices arising from discretizations of differential and partial differential equations often contain the same number throughout the band and may be symmetric such as this matrix is. d) Lower Triangular Matrix: Contains non-zero entries only on or below the diagonal. e) Upper Triangular matrix: Contains zero entries only on or above the diagonal. f) Upper Hessenberg Matrix: Contains non-zero entries only above the sub-diagonal. Similarly, there are lower Hessenberg matrices.

No discussion of numerical linear algebra would be complete without a treatment of iterative methods for solving the linear system $Ax = b$. Iterative methods can be grouped into stationary and non-stationary. Stationary methods use matrix splitting schemes to create a method that can be iterated towards a solution. For example, if one split the matrix A into two matrices M and N such that $A = M - N$ then they would arrive at the matrix equation $(M - N)x = b$ and using a time discretization one could establish the general iterative method given by

$$Mx_{k+1} = Nx_k + b \tag{1}$$

The stationary iterative method is then defined by the choice of matrix splitting. Some popular choices include the Jacobi method where $M = D$ and $N = -(L + U)$, the faster converging Gauss-Seidel method which makes use of updated solutions as they

become available and uses $M = D + L$ and $N = -U$, and the successive over-relaxation method which improves upon Gauss-Seidel by employing a search direction. Non-stationary methods are powerful tools based on optimization methods and are adept at solving even the most difficult systems. A discussion of these methods is beyond the scope of this overview, but an excellent treatment of these methods including the preconditioned conjugate gradient method (PCG), the stabilized Bi-conjugate gradient method (BiCGSTAB), and the method of generalized minimal residuals (GMRES) can be found in the text of Demmel [13].

A topic of numerical linear algebra of particular interest to the computational statistician is that of linear least squares problems. Linear least squares problems arise when dealing with overdetermined systems such as when fitting multiple data observations to fewer variables and hence result in coefficient matrices with more rows than columns. In these cases, one instead attempts to approximate the solution x by minimizing the norm of the residual equation $r = Ax - b$ rather than by computing the exact solution as through a decomposition or by other means as is done in the square coefficient matrix case. One popular way to minimize the residual norm is to use derivatives to find the minimum of its square. Doing a little matrix calculus we find

$$L(x) = ||r||_2^2 = r^T r = (Ax - b)^T (Ax - b) \tag{2}$$

$$\nabla L(x) = 2A^T (Ax - b) \tag{3}$$

By setting the gradient equal to zero we arrive at the normal equation

$$A^T Ax = A^T b \tag{4}$$

Equation 4 is now a square system and can be solved using the methods discussed above. Like the square matrix case, there are many more sophisticated methods for solving least squares problems such as QR factorizations based on orthogonalization methods and singular value decompositions. The interested reader can learn more about the basics of these methods in the introductory text of Heath [31] or for more advanced treatments see [60, 13]. A good treatment on the use of least squares in linear regression is found in [7] which provides a derivation of point estimates for regression models. Having covered the basics of linear systems, we will now turn focus to nonlinear equations.

Nonlinear Equations

The solution of nonlinear equations is another important topic to both the computational mathematician and computational statistician as evidenced by the attention they receive in the introductory texts [31, 8, 27]. Numerical discretizations aside, science

and engineering problems have a tendency to be nonlinear by nature and provide computational scientists with a wide range of problems not suitable to linearization. Statisticians may construct estimators with nonlinear forms requiring the use of numerical solutions such as covered in the text of Small and Wang [58] for example. In addition, many optimization problems form quadratic cost functions in order to ensure a unique minimum or maximum solution and so these types of problems may need to be solved via nonlinear methods as well.

In the single equation case, some nonlinear problems can be classified as 'root-finding' which seeks the values of x such that $f(x) = 0$. Nonlinear problems of this type can be readily solved by using Newton or secant methods which relate back to basic calculus by essentially finding the zero's of the equation by finding where the tangent and secant line respectively intercept the x-axis through an iterative process. A graphical description of Newton's method is provided in figure 2. Taking the Taylor series of a function $f(x)$ yields

$$f(x+h) = f(x) + f'(x)h + O(h^2) \tag{5}$$

and hence one can find an approximation to the problem by taking $f(x+h) = 0$ implying $h = -f(x)/f'(x)$. Through time discretization we arrive at Newton's method in one dimension where we iterate

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{6}$$

Newton's method for a single equation can be extended to systems of nonlinear equations with a little work. To do this one replaces the function derivative in equation 5 with the Jacobian matrix J representing the derivatives of the system of equations. The Jacobian matrix is described in figure 3. The multidimensional method then follows by solving a system of linear equations for an update vector which is used to iterate the solution vector. While Newton's method provides an elegant solution to the nonlinear system of equations problem, the computation of the Jacobian matrix requires N^2 function evaluations where N is the number of nonlinear equations. For this reason, a class of Quasi-Newton methods have been developed over the years which use cheaper approximations to the Jacobian matrix. The result are methods which converge nearly as fast but require only a fraction of the function evaluations as required by true Newton methods. Numerical optimization texts such as that of Nocedal and Wright [44] are excellent sources for more information on Newton and Quasi-Newton methods.

Numerical Integration and Differentiation

Numerical integration, or quadrature, is one of the first examples of a computational method encountered by students as it is introduced in the form of the trapezoidal and

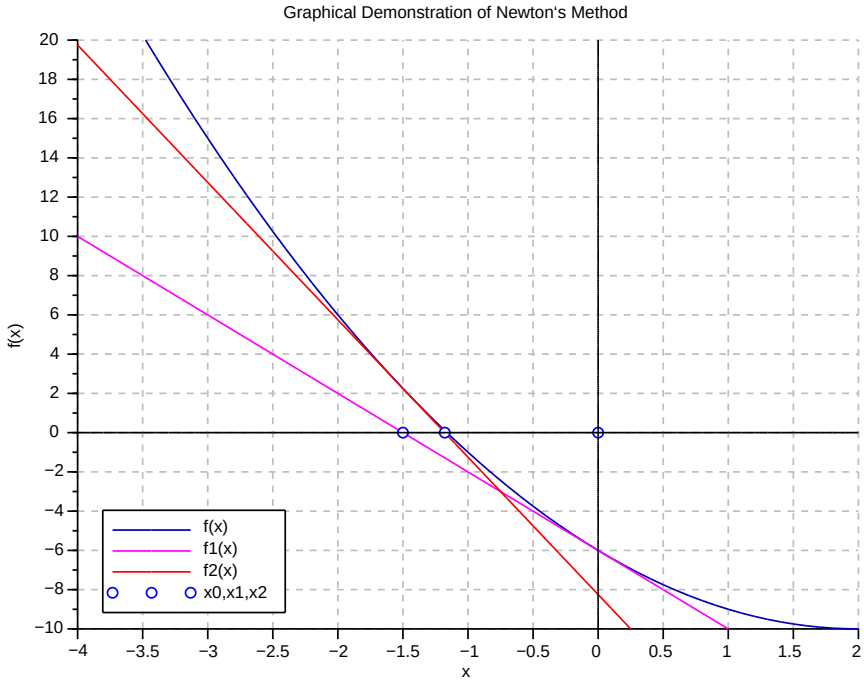

Figure 2: A graphical demonstration of Newton's method. Newton's method finds the zeros of a function by iteratively seeking out the point where the tangent line of the solution point at the current iteration intercepts the X-axis. In this example, one can find the left root of the parabola defined here by $f(x) = (x-2)^2 - 10$ by first taking an initial guess x_0 which is done here using $x_0 = 0$ and finding the corresponding tangent line drawn here as $f_1(x)$. The point where the tangent line $f_1(x)$ crosses the X-axis gives the next approximation point $x_1 = -\frac{3}{2}$. The tangent line at x_1 is then found and the next approximation point $x_2 = -\frac{33}{28}$ lies at the intersection of this second tangent line $f_2(x)$ and the X-axis. The method can be carried on in this fashion by computing new tangent lines for each new approximation point, achieving a closer value to the true solution $2 - \sqrt{10}$ at each iteration until a desired tolerance is achieved.

$$J(X) = \begin{bmatrix} \frac{\partial F_1(X)}{\partial x_1} & \frac{\partial F_1(X)}{\partial x_2} & \cdots & \frac{\partial F_1(X)}{\partial x_n} \\ \frac{\partial F_2(X)}{\partial x_1} & \frac{\partial F_2(X)}{\partial x_2} & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial F_n(X)}{\partial x_1} & \cdots & \cdots & \frac{\partial F_n(X)}{\partial x_n} \end{bmatrix} \tag{7}$$

Figure 3: The Jacobian Matrix $J(X)$ for a function $F(X)$, $X = \{x_1, x_2, ..., x_n\}$ and $F = \{f_1(X), f_2(X), ..., f_n(X)\}$.

Simpson's rules in introductory calculus. Building on this foundation, the computational mathematics and computational statistics student are taught more sophisticated quadrature methods based on quadrature rules such as found in texts like Numerical Mathematics [51]. Quadrature rules approximate an integral by summing a series of weighted functions evaluated at a series of points. Newton-Cotes quadrature, Gaussian quadrature and adaptive quadrature are just some of the more sophisticated methods one can use to approximate an integral. A frequently encountered task in statistics which can require the numerical evaluation of integrals is that of calculating probabilities from representative statistical distributions. While many probability density functions can be easily evaluated, there exist many interesting problems in the numerical integration of more complex distributions as found in the paper of Belov [11]. Numerical integration is inherently well behaved or 'stable' in the eyes of the computational mathematician, one can continually refine the underlying discretization to achieve a more accurate solution without any surprises. Numerical differentiation on the other hand is a different story.

The use of derivatives in mathematics and statistics is highly visible as calculus is a fundamental basis of the subjects and is required in solving varied problems involving differential equations [56, 63, 64] and maximum likelihood estimators [32] for example. In the computational sciences derivatives play special roles in sensitivity and uncertainty analysis and are key components in countless simulation and optimization type problems based on ordinary and partial differential equations. The most elementary form of differentiation is based on finite difference methods and is familiar to anyone who has studied calculus. Looking again to Taylor series expansions we can find formulas for computing derivative approximations, for example solving equation 5 for $f'(x)$ and ignoring higher order terms one arrives at what is referred to as the forward difference approximation to the derivative

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{8}$$

In a similar manner, a backward difference approximation can be derived using $f(x-h)$ in the expansion, and centered difference approximations can be found by combining both versions. As noted earlier, differentiation does not behave as well in a computational frame as integration does. Numerical algorithms developed for solving differen-

tial equations are sensitive to the type (forward, backward, centered) of derivative approximation used and require special conditions on spatial and time discretization steps that must be adhered to ensure algorithmic stability. Determining derivative approximation schemes for complex problems can become a daunting task. A tool known as automatic differentiation can provide a convenient solution to this problem. Automatic differentiation is a special topic in scientific computing where computer programs are written in a special manner so that the program can automatically perform derivative computation when encountered in the code without the need for derivation by the programmer. *The Sourcebook of Parallel Computing* [15] provides a nice introduction to this subject.

Numerically solving ordinary and, in particular, partial differential equations is a major part of the computational sciences. Most students are typically introduced to the basic methods in introductory computational courses, but we will not discuss these here. Texts such as [31, 49] provide good introductions to the numerical solution of both ordinary and partial differential equations. A gentle introduction to the numerical solution of ODEs can be found in [56]. For the more adventurous seeking to delve into the world of numerical PDEs, the texts [61, 63] provide an excellent starting point covering the basics of finite difference methods. The more advanced student may take interest in finite volume methods [40, 64], finite element methods [33] or spectral methods [37].

Numerical Optimization

Optimization type problems can be found in virtually all areas of science, engineering, finance and economics [31]. These typically involve the minimization or maximization of an objective or cost function (these terms are used interchangeably) and may contain a set of constraints that the solution must adhere to. Unconstrained optimization in a single variable is another topic encountered early on in elementary calculus. Finding the minimum of a parabola by equating it's functional derivative to zero and solving is one such example. In numerical optimization one typically deals with problems which are being minimized (or maximized) in more complex and higher dimensional functions. The derivative based methods for solving nonlinear equations discussed previously such as Newton methods or conjugate gradient methods can be naturally extended to the optimization realm. There are numerous ways in which one can deal with additional constraints on optimization problems such as creating penalty or barrier functions which aim to keep a solution in a particular range, or through linear programming methods (which are covered later in the discussion on operations research).

Owing to statistics, the maximum likelihood method is a well known tool for optimizing parameters based on statistical models. Maximum likelihood methods minimize cost functions constructed of likelihood functions and utilize many of the same gradient based solution procedures such as Newton and conjugate gradient methods described

above to obtain the optimal solutions.

Derivative free optimization is sometimes necessary for discontinuous cost functions or those with other complications that prevent the ability to compute a gradient. The class of derivative free optimization techniques is diverse, and many take cues from the processes of nature. Genetic Algorithms [9, 22] mimic the process of natural selection by creating populations of solutions which are used to 'reproduce' new generations and hence combine the best solution traits of both 'parent' solutions to produce more fit 'child' solutions. Simulated annealing [68] mimics the physical process of cooling, and neural networks [38, 69] are patterned from an animal's central nervous system. The probability based Monte Carlo methods [30, 34, 52] are highly robust and can be used for solving many difficult to handle optimization problems, but have a rather slow convergence rate. Hybrid methods exist [18, 19] which seek to exploit the strong points of a certain gradient free method while remedying it's weak points through assistance of another method. For example, a Monte Carlo method could be used to quickly produce a diverse pool of fairly low cost solutions for a particular problem, with the optimization then carried out with a genetic algorithm possessing a faster convergence rate and hence provide a faster solution. Other derivative free methods include direct search methods, and of particular interest to some computational scientists are parallel direct search methods such as the multi-directional search of Torczon [65, 66, 14].

The volume of research on numerical optimization methods is quite extensive owing to the subjects importance in scientific discovery, design in the engineering industry and numerous business aspects. An excellent text covering some of the more advanced topics in numerical optimization is [44].

Monte Carlo Methods

Monte Carlo methods hold a special place in the convergence of computational mathematics and computational statistics for two simple reasons: 1. They are rather versatile and be can be used to solve a wide range of problems in computational mathematics and 2. They are based on probability and hence lend themselves to studies in computational statistics. While their convergence properties often make them a secondary choice (e.g. in optimization we will always opt for a gradient based method if feasible over randomly seeking an optimal solution) they possess a certain robustness that makes them readily available for almost any situation. Problems that are difficult or impossible to solve using other methods (discontinuous solutions, difficult geometries, etc.) often can be solved using Monte Carlo methods. For these reasons, it is no wonder Monte Carlo methods were ranked as the number one algorithm of the twentieth century by [16].

Monte Carlo methods can take on many forms, but at their heart is a series of random experiments which can provide solutions to a problem within some statistical certainty. For example, if one wished to estimate the area of a circle centered in a unit square

they could select points at random to see if they fell inside or outside of the circle and calculate the area based on the proportion of those that fell inside the circle to the total number of points. The level of certainty or the error measure is random but linked directly to the number of random points used in the calculation. As the number of unique points tends towards infinity, the error in the calculation goes to zero. In an optimization realm, Monte Carlo methods can be employed to minimize a function $f(x)$ by randomly choosing solutions x and evaluating them, keeping only the best solution. Some interesting applications of Monte Carlo methods include the direct simulation Monte Carlo method used in rarefied gas dynamics [57] for simulations such as spacecraft re-entry, as an elegant method for solving unsteady adjoint equations [70], and even as a method for inverting matrices [25].

In many instances Monte Carlo methods are embarrassingly parallel, which combined with their robust nature makes them popular in the high performance computing community. In fact, the Sourcebook of Parallel Computing [15] notes that as of it's writing Monte Carlo methods had consumed half of the Department of Energy's high performance computing cycles on it's clusters. Because the Monte Carlo method is dependent on random numbers, random number generation on parallel computers becomes an important topic. Research has been devoted specifically for this need producing special libraries such as SPRNG [42]. Other important research topics in Monte Carlo methods which can improve performance include variance reduction methods and the use of quasi-random numbers instead of pseudo-random numbers (conversations on this topic can also be found in [15]).

Due to the inherent diversity of Monte Carlo methods, their exist numerous texts covering various aspects and application areas. Some general works include [30, 34], while texts such as [52, 5] cover Monte Carlo simulations and [53] demonstrates the methods use in solving differential and partial differential equations.

Summary

In this section we have attempted to assemble a brief overview of the foundational topics that lie at the intersection of the subjects of computational mathematics and computational statistics. While they may be used in different ways and for different reasons by the computational mathematician and computational statistician, the tools provided here lay a numerical foundation for problem solving in more advanced subject matter. Having covered the common ground of the two subjects, we will now move on to discuss application areas that are heavily dependent on the tools provided by computational mathematics and computational statistics.

Application Areas

In this section we will take a brief look at some of the external fields which rely heavily on computational mathematics and computational statistics. Those subjects chosen here embody the convergence of computational mathematics and computational statistics but by no means are they exclusive in this right, and we note that this is just a sampling of the many application areas that share the computational rigors of mathematics and statistics.

Operations Research

One field that stands out as a prime example of the convergence of computational mathematics and computational statistics is that of operations research. The text of Marlow [41] states that "Operations research is the application of scientific methods, and especially mathematical and statistical ones, to problem making decisions." These decisions are frequently ones arising in industry in areas like supply chain logistics, and this is in fact how the field got it's start in England during World War II. Much like the various areas of computational science, operations research relies heavily on the foundational concepts of computational mathematics and computational statistics presented in this paper. Taha's text on operations research [62] provides an excellent introduction to the fundamental topics of the subject.

One of the first major concepts an operations research student must master is linear programming. Linear programming solves a constrained optimization problem whose constraints are linear equations of the solution variable. The solution of such problems is heavily dependent on the fundamentals of numerical optimization and numerical linear algebra. The simplex method as described in [62], for example, is used to solve the constrained optimization problem

$$\begin{aligned} &\textit{min } c^T x \\ &\textit{subject to } Ax = b, \\ &x \geq 0 \end{aligned} \tag{9}$$

by transforming the equation set into a matrix form (or tableau as it is frequently referred to)

$$\begin{bmatrix} 1 & -c^T & 0 \\ 0 & A & b \end{bmatrix} \tag{10}$$

and performing elementary pivot operations to obtain a continually improving feasible solution.

Operations research also relies heavily on probability theory and what is known as probabilistic dynamic programming which is used for solving stochastic inventory

models and problems involving Markovian processes [62]. In addition, Monte Carlo simulations play a vital role in the simulation of certain processes such as waiting lines or queues and is a topic also covered in introductory operations research texts [62]. The solution of nonlinear equations and the study of numerical optimization are also topics of interest in operations research as they play a vital role in quadratic programming.

Stochastic Differential and Partial Differential Equations

The foundations of computational mathematics and computational statistics section of this work briefly discussed the numerical solution of differential and partial differential equations which are of great interest across computational science and engineering, and whose solutions are a popular research area of computational mathematics. Here we will take this notion one step in the direction of statistics by discussing the numerical solution of differential and partial differential equations whose variables are random. Stochastic differential and stochastic partial differential equations contain a stochastic process, such as random forcing terms for physics based problems, which make their solution very different from their non-stochastic counterparts. This topic is one that began to receive a lot of attention in the past decade, particularly due to readily available computing power which has allowed for the real time solution of the stochastic partial differential equations describing financial stock option prices such as the classic Black-Scholes equation [6].

The need to numerically solve stochastic differential and partial differential equations arises in many areas including computational cell biology [21], computational finance [17] and of course computational physics from which the subject spawned with Einstein's seminal paper [20] which described the stochastic process known as Brownian motion. Much as a detailed discussion on the solution of ordinary and partial differential equations exceeds the limits of this review, so does that of their stochastic brethren, but we refer the reader here to texts such as [12, 36].

Machine Learning

A popular topic in today's data driven age, machine learning is a subject more attributed to computer science than either computational mathematics or computational statistics, but after some examination one cannot help but see how it's foundations rely heavily on these two disciplines. So what exactly is machine learning? Whether called machine learning, artificial intelligence or pattern recognition [54] it is a subject devoted to using the numerical algorithms of computational mathematics and observational insight derived from statistics to extract information from sets of data. The similarities between machine learning and statistics are not lost on the authors of [1] as they note that the former could be considered a process with less restrictive assumptions and more general models than the latter.

Much like statistics, machine learning uses observational data to infer specific relationships based on a set of hypothesis. It applies probability theory and algorithms familiar

to numerical optimization to derive meaning from data sets. An excellent example of machine learning in action that most people can relate to is the Netflix million dollar programming prize [4, 50]. In this contest, Netflix sought an algorithm for it's popular streaming media service which could improve viewing recommendations for viewers based on perceived preferences derived from things they watched previously. For example, a learning algorithm may take variables such as actors, directors, genre, etc. into account and test hypotheses against a set of training data (e.g. previously viewed movies) to create a formula for what one likes in order to suggest future movie viewing choices for the user. A more detailed description of this example along with an excellent introduction to the field of machine learning can be found in [1].

Computational Fluid Dynamics

Computational fluid dynamics is a branch of fluid mechanics requiring the use of many of the numerical methods described above to study fluid flows [39]. While much of computational fluid dynamics is of a more deterministic nature studying the macroscopic nature of flows, there exists a range of methods within the subject which focus on the microscopic level and whose mathematical models are based in statistical mechanics. These methods include those which model fluids at the particle level such as the direct simulation Monte Carlo method used in rarefied gas dynamics [57] (and which was discussed previously in the section on Monte Carlo methods). There also exists a popular class known as Lattice Boltzmann methods for solving general Newtonian fluid flow models which lies at the intersection of computational mathematics and computational statistics.

Lattice Boltzmann methods [10] are used to solve fluid dynamics problems modeled using the Boltzmann equation which is founded in statistical mechanics as opposed to more traditional macroscopic flow problems typically handled by the Navier-Stokes equations. The method allows for more microscopic physical properties such as thermal reactions from particle collisions to be present in simulations. Other examples of methods of computational physics which rely on both mathematics and statistics abound. These include the random projection method for hyperbolic conservation laws [3], the previously mentioned Monte Carlo method for solving unsteady adjoint equations [70], and emerging research on fluctuating hydrodynamics [46, 72] to name but a few.

Conclusion

The convergence of computational mathematics and computational statistics is comprised of many of the fundamental algorithms of the fields of numerical linear algebra, the numerical solution of nonlinear equations, numerical integration and differentiation, numerical optimization and Monte Carlo methods. The larger class of numerical methods found within these fields provides a basis of tools for the computational sciences. External fields such as operations research and machine learning benefit from the research and resulting

algorithms that have been compiled by the fields of computational mathematics and computational statistics. We encourage the interested reader to pursue more advanced studies through explorations of the introductory texts referenced within this work. Societies with information pertaining to computational mathematics include the Society for Industrial and applied mathematics [23] and Foundations of Computational Mathematics [45]. Computational statistics societies include the International Association for Statistical Computing [24] and the Statistical Computing Section of the American Statistics Association [55].

References

[1] Yasser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data: A Short Course*. AMLbook.com, 2012.

[2] Herbert L. Anderson. Metropolis, Monte Carlo and the MANIAC. *Los Alamos Science*, 14:96–108, 1986.

[3] Weizhou Bao and Shi Jin. The random projection method for hyperbolic conservation laws with stiff reaction terms. *Journal of Computational Physiscs*, 163:216–248, 2000.

[4] Robert M. Bell, Jim Bennett, Yehuda Koren, and Chris Volinsky. The million dollar programming prize. *IEEE Spectrum*, 2009.

[5] B. A. Berg. *Markov Chain Monte Carlo Simulations and Their Statistical Analysis*. World Scientific Publishers, Singapore, 2004.

[6] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[7] Bruce L. Bowerman and Richard T. O'Connell. *Linear Statistical Models: An Applied Approach*. Duxbury Classic Series. Brooks/Cole, Pacific Grove, CA, second edition, 1990.

[8] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Thomson Brooks/Cole, Belmont, CA, eigth edition, 2005.

[9] Erick Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles*, 10, 1998.

[10] Shiyi Chen and Gary D. Doolen. Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30:329–364, 1998.

[11] Shiyi Chen and Gary D. Doolen. On the computation of the probability density function of α-stable distributions. *Mathematical Modeling and Analysis*, pages 333–341, 2005.

[12] Pao-Liu Chow. *Stochastic Partial Differential Equations*. Applied Mathematics and Nonlinear Science Series. CRC Press, Boca Raton, FL, 2007.

[13] James W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[14] J.E. Dennis and V. Torczon. Direct search methods on parallel machines. *SIAM Journal of Optimization*, 1(4):448–474, 1991.

[15] Jack Dongarra, Ian Foster, Geoffrey Fox, William Grop, Ken Kennedy, Linda Torczon, and Andy White. *Sourcebook of Parallel Computing*. Morgan Kaumann, San Francisco, CA, 2003.

[16] J.J. Dongarra and F. Sullivan. Top ten algorithms of the century. *IEEE Computing in Science and Engineering*, 2(1):22–23, January/February 2000.

[17] Jin-Chuan Duan, Wolfgang Karl Härdle, and James E. Gentle. *Handbook of Computational Finance*. Springer Handbooks of Computational Statistics. Springer-Verlag, Berlin, 2012.

[18] Nazim Dugan and Sakir Erkoc. Genetic algorithm-Monte Carlo hybrid geometry optimization method for atomic clusters. *Computational Materials Science*, 45:127–132, 2008.

[19] R. Duvigneau and M. Visonneau. Hybrid genetic algorithms and artificial neural networks for complex design optimization in cfd. *International Journal for Numerical Methods in Fluids*, 44(11):1257–1278, 2004.

[20] Albert Einstein. Investigations on the theory of the Brownian movement. *Dover Publications*, 1956. English Translation of the original 1926 paper.

[21] Christopher P. Fall, Eric S. Marland, John M. Wagner, and John J. Tyson. *Computational Cell Biology*. Interdisciplinary Applied Mathematics: Mathematical Biology. Springer-Verlag, New York, NY, 2002.

[22] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, July 1993.

[23] The Society for Industrial and Applied Mathematics. http://www.siam.org. Accessed 12/22/2013.

[24] International Association for Statistical Computing. http://www.iasc-isi.org/. Accessed 12/22/2013.

[25] G.E. Forsythe and R.A. Leibler. Matrix inversion by a Monte Carlo method. *Mathematical Tables and Other Aids to Computation*, 4(31):127–129, 1950.

[26] James E. Gentle. *Computational Statistics*. Statistics and Computing. Springer, New York, NY, second edition, 2009.

[27] James E. Gentle, Wolfgang Karl Härdle, and Yuichi Mori. *Handbook of Computational Statistics: Concepts and Methods*. Springer Handbooks of Computational Statistics. Springer-Verlag, Berlin, second edition, 2012.

[28] Geof H. Givens and Jennifer A. Hoeting. *Computational Statistics*. Wiley Series in Computational Statistics. John Wiley & Sons, Inc., Hoboken, New Jersey, second edition, 2013.

[29] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[30] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Chapman and Hall, London and New York, 1964.

[31] Michael T. Heath. *Scientific Computing: An Introductory Survey*. McGraw Hill, New York, NY, second edition, 2002.

[32] Robert V. Hogg and Elliot Tanis. *Probability and Statistical Inference*. Macmillan, New York, NY, 8th edition, 2009.

[33] Thomas J.R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Mineola, NY, 2000.

[34] M. H. Kalos and P. A. Whitlock. *Monte Carlo Methods, Volume I: Basics*. Wiley-Interscience Publications. John Wiley and Sons, New York, NY, 1986.

[35] George Em Karniadakis and Robert M. Kirby II. *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and Their Implementation*. Cambridge University Press, New York, NY, 2003.

[36] P.E. Kloeden and E. Platen. A survey of numerical methods for stochastic differential equations. *Stochastic Hydrology and Hydraulics*, 3(3):155–178, 1989.

[37] David A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. Scientific Computation. Springer, New York, NY, 2009.

[38] Arun D. Kulkarni. *Artificial Neural Networks for Image Understanding*. Van Nostrand Reinhold, New York, NY, 1994.

[39] Pijush K. Kundu and Ira M. Cohen. *Fluid Mechanics*. Elsevier Academic Press, San Diego, California, third edition, 2004.

[40] Randall J. Leveque. *Finite Volume Methods for Hyberbolic Problems*. Cambridge University Press, New York, NY, 2002.

[41] W.H. Marlow. *Mathematics for Operations Research*. Dover Publications, New York, NY, 1978.

[42] M. Mascagni and A. Srinivasan. Algorithm 806: SPRNG: A scalable library for pseudorandom number generation. *ACM Transactions on Mathematical Software*, 26:436–461, 2000.

[43] L. F. Menabrea. Sketch of the analytical engine invented by Charles Babbage. *Bibliothéque Universelle de Genéve*, (82), October 1842.

[44] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, second edition, 2006.

[45] Foundations of Computational Mathematics. http://focm-society.org/. Accessed 12/22/2013.

[46] Research Group of Professor Paul J. Atzberger. http://www.math.ucsb.edu/ atzberg/. Accessed 3/11/2014.

[47] James M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems*. Frontiers of Computer Science. Plenum Press, New York, NY, 1988.

[48] R. P. Porter. The eleventh census. *Publications of the American Statistical Association*, 2:321–379, 1891.

[49] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, second edition, 1992.

[50] Netflix prize Official Website. http://www.netflixprize.com/. Accessed 12/22/2013.

[51] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Texts in Applied Mathematics. Springer, New York, NY, second edition, 2006.

[52] R. Y. Rubenstein. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York, NY, 1981.

[53] K. K. Sabelfeld. *Monte Carlo Methods in Boundary Value Problems*. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[54] Jürgen Schürmann. *Pattern Classification: A Unified View of Statistical and Neural Approches*. John Wiley and Sons, New York, NY, 1996.

[55] American Statistical Association Statistical Computing Section. http://statcomputing.org/. Accessed 12/22/2013.

[56] L.F. Shampine, I. Gladwell, and S. Thompson. *Solving ODE's with Matlab*. Cambridge University Press, New York, NY, 2003.

[57] C. Shen. *Rarefied Gas Dynamics: Fundamentals, Simulations and Micro FLows*. Heat and Mass Transfer. Springer, Berlin, 2005.

[58] Christopher G. Small and Jinfang Wang. *Numerical Methods for Nonlinear Estimating Equations*. Oxford Statistical Science Series. Oxford University Press, Oxford, 2003.

[59] G.W. Stewart. *Matrix Algorithms Volume 1: Basic Decompositions*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1998.

[60] G.W. Stewart. *Matrix Algorithms Volume 2: Eigensystems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

[61] John C. Strickwerda. *Finite Difference Schemes and Partial Differential Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2004.

[62] Hamdy A. Taha. *Operations Research: An Introduction*. Prentice Hall, ninth edition, 2010.

[63] J.W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Texts in Applied Mathematics. Springer, New York, NY, 1995.

[64] J.W. Thomas. *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*. Texts in Applied Mathematics. Springer, New York, NY, 1999.

[65] V. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Rice University, May 1989.

[66] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM J. Optim.*, 1(1):123–145, 1991.

[67] A.M. Turing. On computable numbers, with an application to the entscheidungsproblem: A correction. *Proceedings of the London Mathematical Society*, 43(6):544–546, 1937.

[68] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2010.

[69] L.P.J. Veelenturf. *Analysis and Applications of Artificial Neural Networks*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[70] Qiqi Wang, David Gleich, Amin Saberi, Nasrolla Etemadi, and Parviz Moin. A Monte Carlo method for solving unsteady adjoint equations. *Journal of Computational Physics*, 227:6184–6205, 2008.

[71] Mitchell Watnik. Early computational statistics. *Journal of Computational and Graphical Statistics*, 20(4):811–817, 2011.

[72] Radhakrishnan Lab Research Group Webpage. https://fling.seas.upenn.edu/ biophys/dynamic/wordpress/. Accessed 3/11/2014.