Introduction

The Uncertain Nature of Computation

Computation is the resource that has most transformed our age. Its application has established and extracted value from globe-spanning networks, mobile communication, and big data. As its importance to humankind continues to rise, computation has come at a substantial cost. Consider machine learning, the academic discipline that has underpinned many recent technological advances. The monetary costs of modern massively parallel *graphics processing units* (GPUs), that have proved so valuable to much of machine learning, are prohibitive even to many researchers and prevent the use of advanced machine learning in embedded and other resource-limited systems. Even where there aren't hard resource constraints, there will always be an economic incentive to reduce computation use. Troublingly, there is evidence[1] that current machine learning models are also the cause of avoidable carbon emissions. Computations have to become not just faster, but more efficient.

[1] R. Schwartz et al. "Green AI" (2019); D. Patterson. "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink" (2022).

Much of the computation consumption, particularly within machine learning, is due to problems like solving linear equations, evaluating integrals, or finding the minimum of nonlinear functions, all of which will be addressed in different chapters in this text. These so-called *numerical problems* are studied by the mathematical subfield of *numerics* (short for: *numerical analysis*). What unites these problems is that their solutions, which are numbers, have no *analytic* form. There is no known way to assign an exact numerical value to them purely by structured, rule-based thought. Methods to compute such numbers, by a computer or on paper, are of an approximate nature. Their results are not exactly right, and we do not *know* precisely how far off they are from the true solution; otherwise we could just add that known difference (or error) and be done.

That we are thus *uncertain* about the answer to a numerical

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

problem is one of the two central insights of *probabilistic numerics* (PN). Accordingly, any numerical solver will be uncertain about the accuracy of its output – and, in some cases, also about its intermediate steps. These uncertainties are not ignored by classical numerics, but typically reduced to scalar bounds. As a remedy, PN brings statistical tools for the quantification of uncertainty to the domain of numerics.

What, precisely, are the benefits of quantifying this numerical uncertainty with probability measures?

For a start, a full probability distribution is a richer output than a sole approximation (point estimate). This is particularly useful in sequences of computations where a numerical solution is an input to the next computational task – as is the case in many applications, ranging from the elementary (matrix inversion for least squares) to the complex (solving differential equations for real-word engineering systems). In these settings, a probabilistic output distribution provides the means to propagate uncertainty to subsequent steps.

But PN's uncertainty-aware approach does not stop with quantifying the uncertainty over the final output. Rather, it offers an uncertainty-aware alternative to the design of numerical methods in two ways:

Oftentimes, numerical uncertainty already appears *within* numerical algorithms because its intermediate values are subject to approximations themselves. The resulting *intra-algorithmic* accumulation of such uncertainties calls for an appropriate amount of caution. It appears, e.g., whenever expensive function evaluations are replaced by cheaper simulations (such as when cheaper surrogate functions are used in global optimisation); or when imprecise steps are iterated (such as when ODE solvers concatenate their extrapolation steps). Probabilistic numerical methods account for these uncertainties using probability measures. This enables such methods to make smarter uncertainty-aware decisions, which becomes most salient through their formulation as probabilistic agents (as detailed below).

Moreover, probability distributions allow to more precisely encode the expected structure of the numerical problem into the solver: Numerical tasks can be solved by any number of algorithms, and it is difficult to choose among them. Not all algorithms for, say, integration, work on all integration problems. Some require the integrand to be highly regular, others only that it be continuous, or even just integrable at all. If their requirements are met, they produce a sequence of numbers that converge to the intractable solution. But they do not all

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

converge at the same speed. In particular, algorithms that work on a restricted set of problems, when applied to a problem from that set, often converge faster than methods designed to work on a larger, less-restrictive domain. Academic research in numerics has traditionally concentrated on generic algorithms that work on large spaces of problems. Such methods are now widely available as toolboxes. These collections are valuable to practitioners because they save design time and leverage expert knowledge. But generic methods are necessarily inefficient. Generic methods are, in essence, overly cautious. The more that is known about a computation to be performed before one delves into it, the easier it will be to make progress. Some knowledge is, however, not completely certain but only expected with high probability, and thus cannot be encoded by a function space alone. A probabilistic numerical method, on the other hand, can exploit such less-than-certain expectations by distributing its *prior* probability mass to expected subsets of the function space, and away from less likely scenarios.

The mathematical toolkit of PN allows probabilistic algorithms that leverage these benefits of uncertainty quantification. Such algorithms have proven able to achieve dramatic reductions in computation.

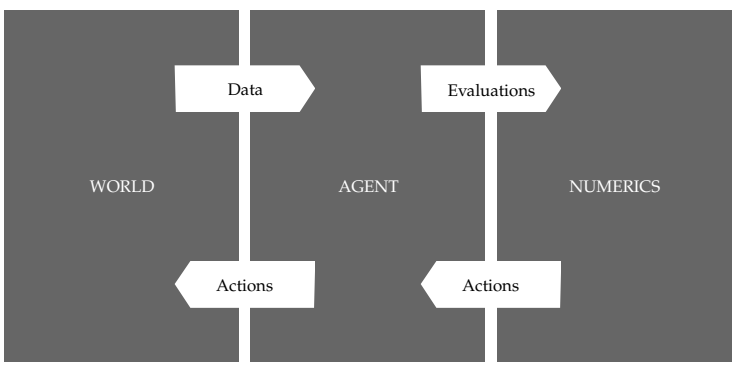

Figure 1: A computational agent interacts with the world just as its numerical algorithms interact with the agent. That is, the agent receives data from the world and selects actions to perform in the world. The numerical algorithm receives evaluations from the agent and selects computations (actions) for the agent to perform. For example, the agent might feed evaluations of an objective to an optimiser, which selects the next evaluations for the agent to make. PN recognises that a numerical algorithm is just as much an agent as one directly interacting with the world.

The second of the central insights of PN is that a numerical algorithm can be treated as an *agent*. For our purposes, an agent is an entity able to take actions so as to achieve its goals. These agents receive data from the environment, use the data to make predictions, and then use the predictions to decide how to interact with the environment. Machine learning often aims to build such agents, most explicitly within its subfield of reinforcement learning. As another example, consider an

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

image classifier using active learning. This classifier receives labelled images, uses those to predict the labels of unlabelled images, and then uses those predictions to decide which new data should be acquired.

It is possible to treat a numerical algorithm as just such an agent, as laid out diagrammatically in Figure 1. Traditionally, a numerical method takes in data, in the form of evaluations (e.g. of an integrand), and returns predictions, or estimates (e.g. of the integral). A numerical method must also provide a rule, perhaps an adaptive one, determining which computations are actually performed (e.g. which nodes to evaluate): these are decisions. There is thus a feedback-loop: an agent that decides itself which data to collect may be inefficient by collecting redundant data, or unreliable if it neglects to probe crucially informative areas of the data-domain. Explicitly, PN treats numerical algorithms just as machine learning often treats its algorithms: as agents.

More precisely, PN is concerned with *probabilistic* agents. As above, PN uses probability distributions to quantify uncertainty. Quantified uncertainty is crucial to all agents, and numerical agents are no exception. In particular, this understanding of numerical solvers brings *probabilistic decision theory*[2] to numerics, yielding a range of advantages.

[2] M. J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. 2015.

For one thing, a numerical agent must decide when to stop. Unlike in the computation of analytic expressions, there is not always an obvious end to a numerical procedure: the error in the current estimate is unknown, so it can be difficult to determine when to stop. Generic and encapsulated methods take a cautious approach, usually aiming to satisfy high demands on precision. Doing so requires many iterations, each further reducing uncertainty and improving precision, but each coming at a cost. That is, this cautious approach consumes much computation. PN provides a new solution to this problem: if uncertainty is well-quantified, we may be satisfied with (quantified) vague, and thus cheap answers. PN hence provides a principled means of stopping *early*, enabling fewer iterations, enabling a reduction in computation.

Uncertainty also guides exploration. An intelligent agent, making a decision, must weigh the uncertain consequences of its actions, occasionally gambling by taking an uncertain action, in order to explore and learn. Performing such exploration is core to effective numerics. Predictive uncertainty unlocks a fundamental means of quantifying the value of a numerical iteration, and weighing it against the real cost of the computation it

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

will consume. There are almost always choices for the character of an iteration, such as where to evaluate an integrand or an objective function to be optimised. Not all iterations are equal, and it takes an intelligent agent to optimise the cost–benefit trade-off.

On a related note, a well-designed probabilistic numerical agent gives a reliable estimate of its own uncertainty over their result. This helps to reduce bias in subsequent computations. For instance, in ODE inverse problems, we will see how simulating the forward map with a probabilistic solver accounts for the tendency of numerical ODE solvers to systemically over- or underestimate solution curves. While this does not necessarily give a more precise ODE estimate (in the inner loop), it helps the inverse-problem solver to explore the parameter space more efficiently (in the outer loop). As these examples highlight, PN hence promises to make more effective use of computation.

The Deep Roots of Probabilistic Numerics

Probabilistic Numerics has a long history. Quite early in the history of numerical computation, people noted that its demands closely matched what was provided by the professional process of guesswork known as *statistical inference*. It seemed to those people that *probability*, central to the process of inference, might be a natural language in which to describe computation as the gathering of information. In the first chapter to his seminal nineteenth-century *Calcul des Probabilités*,[3] Henri Poincaré mused about assigning probabilities to not-yet-computed numbers:

> *Une question de probabilités ne se pose que par suite de notre ignorance: il n'y aurait place que pour la certitude si nous connaissions toutes les données du problème. D'autre part, notre ignorance ne doit pas être complète, sans quoi nous ne pourrions rien évaluer. Une classification s'opérerait donc suivant le plus ou moins de profondeur de notre ignorance.*
> *Ainsi la probabilité pour que la sixième décimale d'un nombre dans une table de logarithmes soit égale à 6 est* ***a priori*** *de 1/10; en réalité, toutes les données du problème sont bien déterminées, et, si nous voulions nous en donner la peine, nous connaîtrions exactement cette probabilité. De même, dans les interpolations, dans le calcul des intégrales définies par la méthode de Cotes ou celle de Gauss, etc.*

[3] H. Poincaré. *Calcul des Probabilités*. 1896. §I.7, pp. 30–31. Emphasis in the original.

[Roughly:] *The need for probability only arises out of uncertainty: It has no place if we are certain that we know all aspects of a problem. But our lack of knowledge also must not be complete, otherwise we would have nothing to evaluate. There is thus a spectrum of degrees of uncertainty.*
While the probability for the sixth decimal digit of a number in a table of logarithms to equal 6 is 1/10 ***a priori****, in reality, all aspects of the corresponding problem are well determined, and, if we wanted to make the effort, we could find out its exact value. The same holds for interpolation, for the integration methods of Cotes or Gauss, etc.* (Emphasis in the op. cit.)

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

Although Poincaré found it natural to assign probabilities to the value of determined but unknown numbers, it seems the idea of uncertainty about a fully, formally determined quantity did not sit well with a majority of mathematicians. Rather than assigning degrees of certainty about properties of a problem, it seemed more acceptable to formally state assumptions required to be *strictly* true at the onset of a theorem. When considering a particular numerical estimation rule, one can then analyse the convergence of the estimate in an asymptotic fashion, thereby formally proving that (and in which sense) the rule is admissible. This approach leaves the estimation of the rule's error after a finite number of steps as a separate and often subordinate part of the routine.

By contrast, probabilistic inference makes the formulation of probability distributions, characterising possible error, primary. These distributions will include an explicit prior on the latent quantity to be inferred, and an equally explicit likelihood function capturing the relationship of computable numbers to that latent quantity. This approach may be more or less restrictive than the asymptotic analysis described above. It might well be more cumbersome to state, subject to philosophical intricacies, and requires care to not introduce new intractable tasks along the way. Nonetheless, phrasing computation as probabilistic inference offers substantial advantages. For one thing, the approach yields a posterior probability distribution for quantities of interest that self-consistently combines estimate and uncertainty, and can approximately track their effect through several steps of the computation.

In the twentieth century, the idea of computation as inference lingered in the margins. In the 1950s and 1960s – the golden age of probability theory following the formal works of Kolmogorov – perhaps Sul'din should be mentioned as the first to return to address it in earnest.[4] He focused on the approximation of functions, a task underlying many numerical methods. In statistics, where this process is known as regression, it took on a life of its own, leading to a deep probabilistic framework studied extensively to this day, whose early probabilistic interpretations where driven by people like Sard (1963), or Kimeldorf and Wahba (1970). Parallel to Sul'din's work in Russia, the task of integration found the attention of Ajne and Dalenius (1960) in Scandinavia. The English-speaking audience perhaps first heard of these connections from Larkin (1972), who went on to write several pieces on the connection between inference and computation. Anyone who missed his works might have had

[4] Sul'din (1959); Sul'din (1960).

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

to wait over a decade. By then, the plot had thickened and authors in many communities became interested in Bayesian ideas for numerical analysis. Among them Kadane and Wasilkowski (1985), Diaconis (1988), and O'Hagan (1992). Skilling (1991) even ventured boldly toward solving differential equations, displaying the physicist's willingness to cast aside technicalities in the name of progress. Exciting as these insights must have been for their authors, they seem to have missed fertile ground. The development also continued within mathematics, for example in the advancement of information-based complexity[5] and average-case analysis.[6] But the wider academic community, in particular users in computer science, seem to have missed much of it. But the advancements in computer science *did* pave the way for the second of the central insights of PN: that numerics requires thinking about agents.

[5] Traub, Wasilkowski, and Woźniakowski (1983); Packel and Traub (1987); Novak (2006).

[6] Ritter (2000)

Twenty-First-Century Probabilistic Numerics

The twenty-first century brought the coming-of-age of machine learning. This new field raised new computational problems, foremost among them the presence of big data sets in the computational pipeline, and thus the necessity to sub-sample data, creating a trade-off between computational cost and precision. Numerics is inescapably important to machine learning, with popular tracks at its major conferences, and large tracts of machine learning masters' degrees, devoted to optimisation alone. But machine learning also caused a shift in perspective on modelling itself.

Modelling (or inference) used to be thought of as a passive mathematical map, from data to estimate. But machine learning often views a model as an *agent* in autonomous interaction with its environment, most explicitly in reinforcement learning. This view of algorithms as agents is, as above, central to PN.

Machine learning has been infused with the viewpoints of physicists and other scientists, who are accustomed to limits on precision and the necessity of assumptions. The Bayesian viewpoint on inference soon played a prominent (albeit certainly not the only leading) role in laying its theoretical foundations. Textbooks like those of Jaynes and Bretthorst (2003), MacKay (2003), Bishop (2006), and Rasmussen and Williams (2006) taught a generation of new students – the authors amongst them – to think in terms of generative models, priors, and posteriors. Machine learning's heavy emphasis on numerics couldn't help but lead some of those students to apply their hammer, of probabilis-

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

tic inference, to the nails of the numerical problems that they encountered. These students were bound to revive questions from the history of PN. For instance, if computation is inference, should it then not be possible to build numerical algorithms that:

1. rather than taking in a logical description of their task and returning a (floating-) point estimate of its solution, instead take probability distributions as inputs and outputs?;
2. use an explicit likelihood to capture a richer, generative description of the relation between the computed numbers and the latent, intractable quantity in question?; and
3. treat the CPU or GPU as an interactive source of data?

In 2012, the authors of this text co-organised, with J. P. Cunningham, a workshop at the *Neural Information Processing Systems* conference on the shores of Lake Tahoe to discuss these questions. We were motivated by our own work on Bayesian Optimisation and Bayesian Quadrature. At the time, the wider issue seemed new and unexplored to us. By a stroke of luck, we managed to convince Persi Diaconis to make the trip up from Stanford and speak. His talk pointed us to a host of prior work.

In search of an inclusive, short title for the workshop, we had chosen to call it *Probabilistic Numerics* (PN). In the years since, this label has been increasingly used by a growing number of researchers who, like us, feel that the time has come to more clearly and extensively connect the notions of inference and computation. The 2012 workshop also marked the beginning of a fruitful collaboration between the machine learning community and statisticians like M. Girolami[7] and C. J. Oates, as well as applied mathematicians like T. J. Sullivan, I. Ipsen, H. Owhadi and others. They ensured that existing knowledge in either community was not forgotten,[8] and their research groups have since laid an increasingly broad and deep foundation to the notion of computation as *probabilistic* and also more narrowly, carefully defined *Bayesian* inference.[9] They have also undertaken a commendable effort to build a community for PN within the mathematical fields.

[7] Hennig, Osborne, and Girolami (2015)

[8] Owhadi and Scovel (2016); Oates and Sullivan (2019).

[9] Cockayne et al. (2019b)

Although numerous interesting insights have already been reached, just as many questions are still scarcely explored. Many of them emerge from new application areas, and new associated computational problems, in the age of Big Data, GPUs, and distributed, compartmental, computation.

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

We made a conscious decision not to use the word *Bayesian* in naming Probabilistic Numerics. We will unashamedly adopt a Bayesian view within this text, which forms a natural framework for the core ideas within PN; those ideas can also be found in many alternative approaches to machine learning and statistics. But there is also an important way in which PN can be viewed as non-Bayesian. The Bayesian norm enforces hygiene between modelling and decision-making. That is, you write down a prior capturing as much of your background knowledge as possible, and do inference to compute a posterior. Then, with that posterior, you write down a loss function and use expected loss to choose an action. The Bayesian's counterpart, the frequentist, has the loss function in mind from the outset.

However, in numerics, we are rarely afforded the luxury of using models informed by all available prior knowledge. Such models are usually more computationally expensive, in themselves, than models that are built on only weak assumptions. Sometimes, we are willing to spend a little more computation on a model to save even more computation in solving its numerical task, but other times we are not. That is, in considering an additional computation cost on a model, we must consider whether it is justified in improving performance for the given numerical task: this performance is measured by a loss function.

PN is hence, in this way, more akin to the frequentist view in muddling the loss function and the prior. That is, numerics requires us to make, in some cases, drastic simplifications to our models in order to achieve usable computational complexity. This can be conceived as letting some (vaguely specified) loss function on computation dictate which elements of the prior can be incorporated.

This Book

This book aims to give an overview of the emerging new area of Probabilistic Numerics, particularly influenced by contemporary machine learning. Even at this early point in the field, we have to concede that a complete survey is not possible: we are bound to under-represent some rapidly developing viewpoints, and apologise in advance to its authors. Our principal goals will be to study uses and roles for *uncertainty* in numerical computation, and to employ such uncertainty in making *optimal decisions* about computation.

Invariably, we will capture uncertainty in the language of *probabilities*. Any quantity that is not known to perfect (machine)

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

precision will ideally be assigned an explicit probability distribution or measure. We will study algorithms that take and return probabilities as inputs and outputs, respectively, but also allow for uncertain, imprecise, computations to take place within the algorithm itself. These algorithms will explicitly be treated as agents, intelligently making decisions about which computations to perform. Within our probabilistic framework, these decisions will be selected as those that *minimise an expected loss*.

Along the way, we will make several foundational observations. Here is a preview of some of them, forming a quick tour through the text:

Classical methods are probabilistic Classical methods often have clear probabilistic interpretations. Across the spectrum of numerical tasks, from integration to linear algebra, nonlinear optimisation, and solving differential equations, many of the foundational, widely used numerical algorithms can be explicitly motivated as maximum a posteriori or mean *point-estimates* arising from concrete (typically Gaussian) prior assumptions. The corresponding derivations take up a significant part of this text. These insights are crucial for two reasons.

First, finding a probabilistic interpretation for existing methods shows that the idea of a probabilistic numerical method is not some philosophical pipe dream. Probabilistic methods tangibly exist, and are as fast, and as reliable as the methods people trust and use every day – because those very methods already *are* probabilistic numerical methods, albeit they are not usually presented as such.

Second, once phrased as probabilistic inference, the classical methods provide a solid, well-studied and understood foundation for the development of *new* numerical methods and novel functionality addressing modern challenges. It may be tempting to try and invent probabilistic methods de novo; but the analytical knowledge and practical experience embodied in numerical libraries is the invaluable condensed labor of generations of skilled applied mathematicians. It would be a mistake to throw them overboard. Classical numerical methods are computationally lightweight (their cost per computational step is often constant, and small), numerically stable (they are not thrown off by small machine errors), and analytically efficient (they converge to the true solution at a "good" rate). When developing new functionality, one should strive to retain these properties as much as possible, or at least to take inspiration

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

from the form of the classical methods. At the same time, we also have to leave a disclaimer at this point: Given the breadth and depth of numerical analysis, it is impossible to give a universal introduction in a book like this. The presentations of classical methods in these pages are designed to give a concise introduction to specific ideas, and highlight important aspects. We refer interested readers from non-numerical backgrounds to the established literature, referenced in the margins.[10]

[10] The text contains a large number of notes in the margins. It is generally possible to read just the main text and ignore these offshoots. Some of them provide short reference to background knowledge for the readers' convenience. Others are entry points to related literature.

Numerical methods are autonomous agents We will employ the decision-theoretic, expected-loss-minimisation, framework to design numerical algorithms. Often, it is not just that the way a classical numerical method combines collected floating-point numbers into a numerical estimate can be interpreted as a posterior expectation or estimate. Additionally, the decision rule for computing those numbers in the first place also arises naturally from the underlying prior probabilistic model, through a decision-theoretic treatment. Thus, such a classical numerical algorithm can indeed be interpreted as an *autonomous agent* acting consistently with its internal probabilistic "beliefs". At first glance, this insight has primarily aesthetic value. But we will find that it directly motivates novel algorithms that act in an adaptive fashion.

Numerics should not be random Importantly, we will *not* identify probabilistic uncertainty with *randomness*.[11] Randomness is but one possible way for uncertainty to arise. This kind is sometimes called *aleatory* or *stochastic*, in contrast to the *epistemic* uncertainty capturing a lack of *knowledge*. Probability theory makes no formal distinction between the two, they are both captured by spreading unit measure over a space of hypotheses. But there are some concepts, notably that of *bias*, which require a careful separation of these types of uncertainty. Furthermore, randomness is often used within numerics today to make (tough!) decisions, for instance, about where to make evaluations of an integrand or objective. We will argue that randomness is ill-suited to this role, as can be seen in describing a numerical algorithm as an agent (whose expected-loss-minimising action will never be returned by a random number generator). We will show (§12.3) how non-random, expected-loss-minimisation, decisions promise the reward of dramatically lowered computation consumption. This is not a fundamental rejection of the concept of Monte Carlo methods, but it reveals deep philosophical subtleties surrounding these algorithms that raise concrete

[11] §12.3 delves into this topic.

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

research questions.

Numerics must report calibrated uncertainty To complete the description of classical methods as probabilistic, it does not suffice to note these methods give point estimates that arise from the most probable or expected value, the "location" of a posterior distribution. We also have to worry about the "width", captured by variance and support, of the posterior around this point estimate. We must ask whether the posterior can indeed be endowed with an interpretation as a notion of uncertainty, connected to the probable error of the numerical method. The sections that study connections to existing methods will provide some answers in this regard. We will frequently argue that certain classical methods arise from a family of *Gaussian* prior measures, parametrised by either a scalar or multivariate scale of uncertainty. All members of that family give rise to Gaussian posteriors with the same mean (which is identical to the classical method) and a posterior standard deviation that contracts at the same rate, and thus only differs by the constant scale. We will see that the contraction rate of this posterior variance is related to classical convergence rates of the point estimate (in its non-adaptive form, it is a conservative, worst-case bound on the error). We will further show that the remaining constant parameter can be inferred at runtime with minimal computational overhead, using either probabilistic, statistical, or algebraic estimation rules. Throughout, we will argue that the provision of well-calibrated quantifications of uncertainty is crucial to numerical algorithms, whether classical or new. Such reliable uncertainty underpins both the trust that can be placed in numerical algorithms and effective decision-making about computation.

Imprecise computation is to be embraced Having adopted reliable quantifications of uncertainty, we are freed from the burden of ensuring that numerical calculations must always be highly precise. Not all numerical problems are equal: computation can be saved, in lowering precision, for the least important. From our perspective, some of the most pressing contemporary numerical problems arise in data science and machine learning, in the processing pipelines of big data sets. In these settings, data are frequently sub-sampled at runtime. This introduces stochastic disturbances to computed numbers that significantly lowers the *precision* of the computation. Yet sub-sampling also drastically reduces computational cost compared to computations on the

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

entire data set. The trade-off between computation cost and precision results in *tuning parameters* (e.g. specifying how large a random sub-sample should be chosen) being exposed to the user, a major irk to practitioners of data science. PN here offers value in enabling optimisation of this trade-off, freeing the user of the nasty task of fiddling with parameters.

PN *consolidates numerical computation and statistical inference* Both numerical solvers and statistical-inference methods convert the information available to them into an approximation of their quantity of interest. In numerics, this information consists of evaluations of analytic expressions (or functions) – e.g. of the integrand for quadrature. In statistics, it comes as measurements (data) of observable variables. But ultimately, when viewed through the lens of information theory, these types of information are essentially the same – namely nothing but the output of a (noisy) communication channel,[12] either through an (imprecise) function or a (noisy) statistical observation. PN exploits this analogy by recasting numerical problems as statistical inference. More precisely, probabilistic numerical methods work by providing a statistical model linking the accessible function evaluations to the solution of a numerical problem, and then approximating the solution by use of statistical inference in this model. But this statistical model can be useful beyond the numerical problem at hand: should the occasion arise, it can be extended to include additional observational data containing more information. This way, the computational and observational data (information) can work together, in a *single* model, to improve the inference of the numerical solution, and of other latent variables of this joint model. Real payoffs of this probabilistic consolidation of numerics and statistics have, for example, been demonstrated for differential equations – as we will detail in §41.3.

[12] MacKay (2003), §9

Probabilistic numerical algorithms are already adding value The PN approach to global optimisation is known as Bayesian optimisation, and, in fact, Bayesian optimisation was conceived as probabilistic since its invention. Bayesian optimisation is widely used today to automatically make decisions otherwise made by human algorithm designers. Machine learning's growth has created a bewildering variety of algorithms, each with their own design decisions: the choices of an algorithm and its detailed design to be made can be framed as an "outer-loop" global optimisation problem. This problem makes careful selection of evaluations of the algorithm and its design primary, as the cost

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

of evaluating an algorithm on a full data set is expensive enough to prevent anything approaching exhaustive search. In finding performant evaluations sooner, Bayesian optimisation saves, relative to the alternative of grid search, considerable computation on evaluations. Pre-dating the recent surge of research in wider Probabilistic Numerics, Bayesian optimisation has already blossomed into a fertile sub-community of its own, and produced significant economic impact. Chapter V is devoted to this domain, highlighting a host of new questions and ideas arising when uncertainty plays a prominent role in computation.

Pipelines of computation demand harmonisation A probabilistic-numerics framework, encompassing all numerical algorithms, grants the ability to efficiently allocate computation, and manage uncertainty, amongst them. A newly prominent computational issue is that realistic data processing in science and industry happens not in a single computational step, but in highly engineered pipelines of compartmental computations. Each step of these chains consumes computation, and depends on and propagates errors and uncertainties. The area of uncertainty quantification[13] has developed methods to study and identify such problems, but its methods tend to add computational overhead that is not recouped in savings elsewhere. Probabilistic numerical methods, with their ability to handle uncertain inputs and produce calibrated uncertain outputs, offer a natural notion of uncertainty propagation through computational graphs. The framework of graphical models provides a scaffolding for this process. Depending on the user's needs, it is then possible to scale between simple Gaussian forms of uncertainty propagation that produce simple error bars at only minimal computational overhead, to full-fledged uncertainty propagation, with more significant computational demands. With a harmonised treatment of the uncertainty resulting from each step, and the computational costs of reducing such uncertainty, PN allows the allocation of computational resources to those steps that would benefit from it most.

[13] Sullivan (2015)

Open questions Finally, the text also highlights some areas of ongoing research, again with a focus on desirable functionality for data-centric computation.

We will *not* pay very close attention to machine precision, machine errors, and problems of numerical stability. These issues have been studied widely and deeply in numerical analysis. They play an important role, of course, and their ability to cause

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

havoc should not be underestimated. But in many pressing computational problems of our time, especially those involving models defined on external data sets, the dominant sources of uncertainty lie elsewhere. Sub-sampling of big data sets to speed up a computation regularly causes computational uncertainty many orders of magnitude above the machine's computational precision. In fact, in areas like deep learning, the noise frequently dominates the signal in magnitude. This is also the reason why we spend considerable time on methods of low order: advanced methods of high order are often not applicable in the context of high computational uncertainty.

This Book and You

We wrote this book for anyone who needs to use numerical methods, from astrophysicists to deep learning hackers. We hope that it will be particularly interesting for those who are, or are aiming at, becoming a developer of numerical methods, perhaps those with machine learning or statistical training.

We invite you to join the Probabilistic Numerics community. Why should you care?

Probabilistic Numerics is beautiful The study of PN is its own reward. It offers a unified treatment of numerical algorithms that recognises them as first-class citizens, agents in their own right.

Probabilistic Numerics is just beginning to bloom The PN banner has been borne since Poincaré, and ours will not be the generation to let it slip. Despite these deep roots, the field's branches are only now beginning to be defined, and we can only guess at what wonderful fruit they will produce. In Chapter VII, we will describe some of the many problems open to your contributions to PN.

Probabilistic Numerics is your all-in-one toolkit for numerics Numerics need not be considered foreign to those with statistical or machine learning expertise. PN offers a machine learning or statistics researcher the opportunity to deploy much of their existing skillset in tackling the numerical problems with which they are so commonly faced. PN allows the design of numerical algorithms that are perfectly tailored to the needs of your problem.

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press

Probabilistic Numerics grants control of computation PN's promise is to transform computation, sorely needed in an age of ballooning computation demands and the ever-growing evidence that its costs cannot continue to be borne.

With this, it is time to get to the substance. The next chapter provides a concise introduction to the most central arithmetic framework of probabilistic inference: Gaussian probability distributions, which provide the basic toolbox for computationally efficient reasoning with uncertainty. Readers fully familiar with this area can safely skip this chapter, and move directly to the discussion of the first and arguably the simplest class of numerical problems, univariate integration, in Chapter II.

https://doi.org/10.1017/9781316681411.003 Published online by Cambridge University Press