

***Probabilistic Linear Solvers for Machine Learning***

**By Supervisor :**

Ala'a Alddin Al-Maqtari

**By :**

**IBRAHIM ALEZZI ALHAJJ**

**Dhamar University**

**Faculty of Engineering**

**Mechatronics Engineering Department**

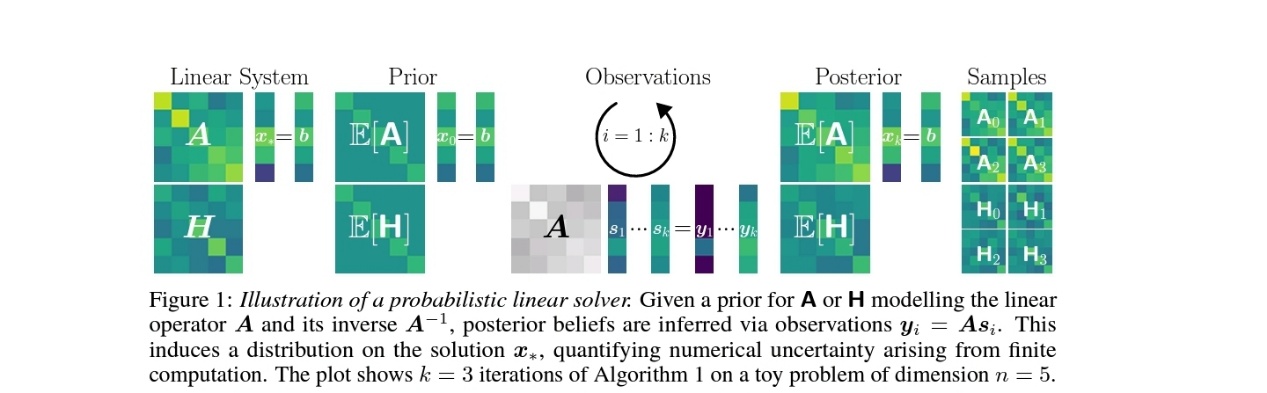
**Fourth Year/ First Semester**

1. **Abstract**

**Linear systems are the bedrock of virtually all numerical computation. Machine learning poses specific challenges for the solution of such systems due to their scale, characteristic structure, stochasticity and the central role of uncertainty in the field. Unifying earlier work we propose a class of probabilistic linear solvers which jointly infer the matrix, its inverse and the solution from matrix-vector product observations. This class emerges from a fundamental set of desiderata which constrains the space of possible algorithms and recovers the method of conjugate gradients under certain conditions. We demonstrate how to incorporate prior spectral information in order to calibrate uncertainty and experimentally showcase the potential of such solvers for machine learning.**

1. **INTRODUCTION**

***guably one of the most fundamental problems in machine learning, statistics and scientific computation at large is the solution of linear systems of the form Ax∗ = b, where A ∈ Rn× n sym is a symmetric positive definite matrix [1–3]. Such matrices usually arise in the context of second-order or quadratic optimization problems and as Gram matrices. Some of the numerous application areas in machine learning and related fields are least-squares regression [4], kernel methods [5], Kalman f iltering [6], Gaussian (process) inference [7], spectral graph theory [8], (linear) differential equations [9] and (stochastic) second-order methods [10]. Linear systems in machine learning are typically large-scale, have characteristic structure arising from generative processes, and are subject to noise. These distinctive features call for linear solvers that can explicitly make use of such structural information. While classic solvers are highly optimized for general problems, they lack key functionality for machine learning. In particular, they do not consider generative prior information about the matrix. Animportant example are kernel Gram matrices, which exhibit specific sparsity structure and spectral properties, depending on the kernel choice and the generative process of the data. Exploiting such prior information is a prime application for probabilistic linear solvers, which aim to quantify numerical uncertainty arising from limited computational resources. Another key challenge, which we will not yet address here, are noisy matrix evaluations arising from data subsampling. Ultimately, linear algebra for machine learning should integrate all sources of uncertainty in a computational pipeline– aleatoric, epistemic and numerical– into one coherent probabilistic framework. Contribution This paper sets forth desiderata for probabilistic linear solvers which establish first principles for such methods. From these, we derive an algorithm incorporating prior information on the matrix A or its inverse A−1, which jointly estimates both via repeated application of A. This results in posterior beliefs over the two operators and the solution which quantify numerical uncertainty. Our approach unifies and extends earlier formulations and constitutes a new way of***



***interpreting linear solvers. Further, we propose a prior covariance class which recovers the method of conjugate gradients as its posterior mean and uses prior spectral information for uncertainty calibration, one of the primary shortcomings of probabilistic linear solvers. We conclude by presenting simplified examples of promising applications of such solvers within machine learning. [1] [2].***

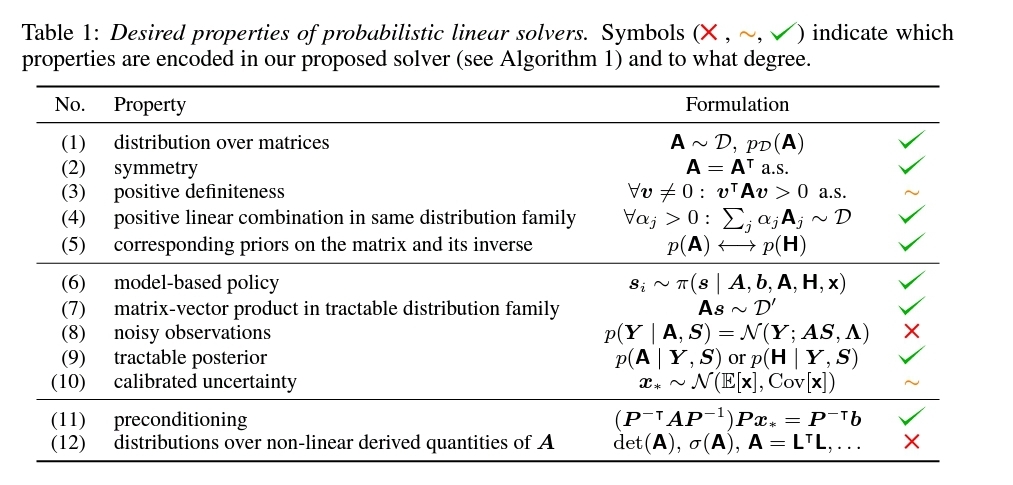
* 1. 2 Probabilistic Linear Solvers

Let Ax∗ = bilinear system with A ∈ Rn× n sym positive definite and b ∈ Rn. Probabilistic linear solvers (PLS) [11–13] iteratively build a model for the linear operator A, its inverse H = A−1 or the solution x∗, represented by random variables A,H or x. In the framework of probabilistic numeric [14, 15] such solvers can be seen as Bayesian agents performing inference via linear observations Y =[y1,...,yk] ∈ Rn× k resulting from actions S = [s1,...,sk] ∈ Rn× k given by an internal policy π(s | A,H,x,A,b). For a matrix-variate prior p(A) or p(H) encoding prior (generative) information, our solver computes posterior beliefs over the matrix, its inverse and the solution of the linear system. An illustration of a probabilistic linear solver is given in Figure 1. Desiderata We begin by stipulating a fundamental set of desiderata for probabilistic linear solvers. To our knowledge such a list has not been collated before. Connecting previously disjoint threads, the following presents a roadmap for the development of these methods. Probabilistic linear solvers modelling A and A−1 must assume matrix-variate distributions which are expressive enough to capture structure and generative prior information either for A or its inverse. The distribution choice must also allow computationally efficient sampling and density evaluation. It should encode symmetry and positive definiteness and must be closed under positive linear combinations. Further, the two models for the system matrix or its inverse should be translatable into and consistent with each other. Actions si of a PLS should be model-based and induce a tractable distribution on linear observations yi = Asi. Since probabilistic linear solvers are low-level procedures, their inference procedure must be computationally lightweight. Given (noise-corrupted) observations this requires tractable posteriors over A, H and x, which are calibrated in the sense that at convergence the true solution x∗ represents a draw from the posterior p(x | Y ,S). Finally, such solvers need to allow preconditioning of the problem and ideally should return beliefs over non-linear properties of the system matrix extending the functionality of classic methods. These desiderata are summarized concisely in Table 1.

* 1. ***.1 Bayesian Inference Framework***

***Guided by these desiderata, we will now outline the inference framework for A,H and x forming the base of the algorithm. The choice of a matrix-variate prior distribution is severely limited by the desideratum that conditioning on linear observations yi = Asi must be tractable. This reduces the choice to stable distributions [16] and thus excludes candidates such as the Wishart, which has measure zero outside the cone of symmetric positive semi-definite matrices. For symmetric matrices, this essentially forces use of the symmetric matrix-variate normal distribution, introduced in this context by Henning [11]. Given A0,WA 0 ∈ Rn× n sym , assume a prior distribution***

***p(A) = N(A;A0,WA 0 WA 0),***



where denotes the symmetric Kronecker product [17].1 The symmetric matrix-variate Gaussian induces a Gaussian distribution on linear observations. While it has non-zero measure only for symmetric matrices, its support is not the positive definite cone. However, positive definiteness can still be enforced post-hoc (see Proposition 1). We assume noise-free linear observations of the form yi = Asi, leading to a Dirac likelihood

p(Y | A,S) = lim ε↓0 N(Y ;AS,ε2I I) = δ(Y −AS).

The posterior distribution follows from the properties of Gaussians [4] and has been investigated in detail in previous work [18, 11, 13]. It is given by

p(A | S,Y ) = N(A;Ak,Σk)

with Ak =A0+∆A 0U+U(∆A 0)−US∆A 0U Σk =WA 0(In −SU)WA 0(In−SU) where

∆A 0 = Y −A0S and U =WA 0S(SWA 0S)−1.

Weaimtoconstructaprobabilistic model H for the inverse H = A−1 consistent with the model A as well. However, not even in the scalar case does the inverse of a Gaussian have finite mean. We ask instead what Gaussian model for H is as consistent as possible with our observational model for A. For a prior of the form p(H) = N(H;H0,WH 0WH 0)andlikelihood

p(S | H,Y ) = δ(S−HY )

,wean logout loathe A-model obtain a posterior distribution p(H | S,Y ) = N(H;Hk,ΣH k)

with Hk =H0+∆H 0(UH)+UH(∆H 0)−UHY∆H 0(UH) ΣH k = WH 0(In −Y(UH))WH 0(In−Y(UH)) where ∆H 0 = S −H0Y and

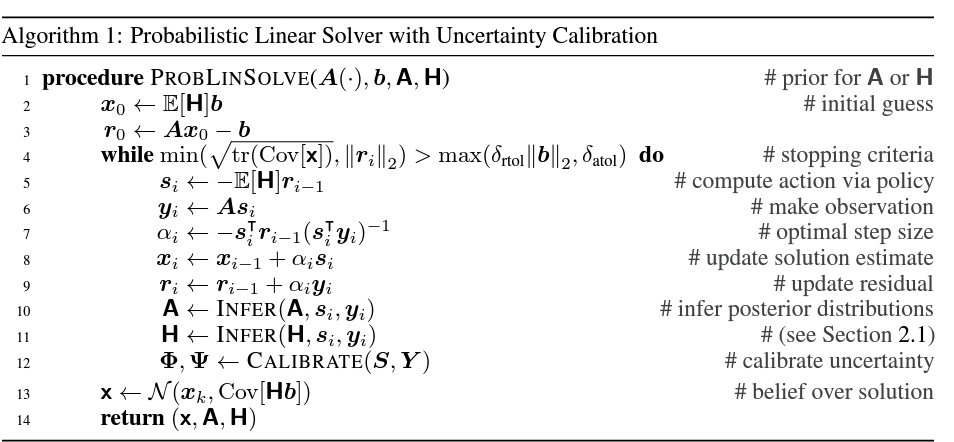
UH =WH 0Y(YWH 0Y)−1.

. InSection3wewillderive a covariance class, which establishes correspondence between the two Gaussian viewpoints for the linear operator and its inverse and is consistent with our desiderata.

1. **Algorithm**

**The above inference procedure leads to Algorithm 1. The degree to which the desiderata are encoded in our formulation of a PLS can be found in Table 1. We will now go into more detail about the policy, the choice of step size, stopping criteria and the implementation. Policy and Step Size In each iteration our solver collects information about the linear operator A via actions si determined by the policy**

**π(s |A,H,x,A,b). The next action si = −E[H]ri−1**



chosen based on the current belief about the inverse. If E[H] = A−1, i.e. if the solver’s estimate for the inverse equals the true inverse, then Algorithm 1 converges in a single step since

xi−1 +si = xi−1 −E[H]ri−1 = xi−1 −A−1(Axi−1 −b) = A−1b = x∗.

The step size minimizing the quadratic

q(xi + αsi) = 1 2 (xi +αsi)A(xi +αsi)−b(xi +αsi)

along the action si is given by

αi = argminα q(xi + αsi) = si (b −Axi)(siAsi)−1.

Stopping Criteria Classic linear solvers typically use stopping criteria based on the current residual of the form

Axi −b2 ≤ max(δrtolb2,δatol)

for relative and absolute tolerances δrtol and δatol. However, this residual may oscillate or even increase in all but the last step even if the error x∗ −xi2 is monotonically decreasing [19, 20]. From a probabilistic point of view, we should stop if our posterior uncertainty is sufficiently small. Assuming the posterior covariance is calibrated, it holds that (Ex∗ [x∗ − E[x]2])2 ≤ Ex∗ [x∗ −E[x]2 2] = tr(Cov[x]).

Hence given calibration, we can bound the expected (relative) error between our estimate and the true solution by terminating when

tr(Cov[x]) ≤ max(δrtolb2,δatol).

A probabilistic criterion is also necessary for an extension to the noisy setting, where classic convergence criteria become stochastic. However, probabilistic linear solvers typically suffer from MI calibration [21], an issue we will address in Section 3.

Implementation We provide an open-source implementation of Algorithm 1 as part of PROBNUM, a Python package implementing probabilistic numerical methods, in an online code repository: https://github.com/probabilistic-numerics/probnum The mean and covariance up- and down dates in Section 2.1 when performed iteratively are of low rank. In order to maintain numerical stability these updates can instead be performed for their respective Chelsey factors [22]. This also enables computationally efficient sampling or evaluation of probability density functions downstream.

1. **Theoretical Properties**

This section details some theoretical properties of our method such as its convergence behavior and computational complexity. In particular we demonstrate that for a specific prior choice Algorithm 1 recovers the method of conjugate gradients as its solution estimate. All proofs of results in this section and the next can be found in the supplementary material. We begin by establishing that our solver is a conjugate directions method and therefore converges in at most n steps in exact arithmetic. Theorem 1 (Conjugate Directions Method) Given a prior

p(H) = N(H;H0,WH 0 WH 0)

such that

H0,WH 0 ∈ Rn× n

sym positive definite, then actions is of Algorithm 1 are A-conjugate, i.e. for 0 ≤ i,j ≤ k with i= j it holds that is Asj = 0.

Decantation a better convergence rate by placing stronger conditions on the prior covariance class as outlined in Section 3. Given these assumptions, Algorithm 1 recovers the iterates of (preconditioned) Craned thus inherits its favorable convergence behavior (overviews in [23, 10]). Theorem 2 (Connection to the Conjugate Gradient Method) Given a scalar prior mean

A0 = H−1 0 =αI with α>0

,assume(1)and (2) hold, then the iterates xi of Algorithm 1 are identical to the ones produced by the conjugate gradient method. A common phenomenon observed when implementing conjugate gradient methods is that due to cancellation in the computation of the residuals, the search directions si lose A-conjugacy [24, 25, 3]. In fact, they can become independent up to working precision for i large enough [25].

One way to combat this is to perform complete reorthogonalization of the search directions in each iteration as originally suggested by Lanchow [26]. Algorithm 1 does this implicitly via its choice of policy which depends on all previous search directions as opposed to just si−1 for (naive) CG.

Computational Complexity The solver has time complexity O(kn2) for k iterations without uncertainty calibration. Compared to CG, inferring the posteriors in Section

2.1 adds an overhead of four outer products and four matrix-vector products per iteration, given (1) and (2). Uncertainty calibration outlined in Section 3 adds between O(1) and O(k3) per iteration depending on the sophistication of the scheme. Already for moderate n this is dominated by the iteration cost. In practice, means and covariance's do not need to be formed in memory. Instead they can be evaluated lazily as linear operators v → Lv, if S and Y are stored. This results in space complexity O(kn)..

1. **Related Work**

**Numerical methods for the solution of linear systems have been studied in great detail since the last century. Standard texts [1, 2, 10, 3] give an in-depth overview. The conjugate gradient method recovered by our algorithm for a specific choice of prior was introduced by Hastens and Stifle [19].**

**Recently, randomization has been exploited to develop improved algorithms for large-scale problems arising from machine learning [27, 28].**

**The key difference to our approach is that we do not rely on sampling to approximate large-scale matrices, but instead perform probabilistic inference. Our approach is based on the framework of probabilistic numerics [14, 15] and is a natural continuation of previous work on probabilistic linear solvers.**

**In historical order, Hennig and Keitel [18] provided a probabilistic interpretation of Quasi-Newton methods, which was expanded upon in [11]. This work also relied on the symmetric matrix-variate Gaussian as used in our paper. Bartels and Hennig [29]**

**estimate numerical error in approximate least-squares solutions by using a probabilistic model.**

**More recently, Cocaine et al. [21] proposed a Bayesian conjugate gradient method performing inference on the solution of the system.**

**This was connected to the matrix-based view by Bartels.**

**6. Experiments**

**This section demon stratosphere functionality of Algorithm1.Wechoosesome–deliberatelysimpleexampleproblemsfrommachinelearningandscientificcomputation,wherethesolvercanbeusedto quantifyuncertaintyinducedbyfinitecomputation,solvemultipleconsecutivelinearsystems,and propagate information between problems. GaussianProcessRegressionGPregression[7]infersalatentfunctionf:RN→Rfromdata D=(X,y),whereX∈Rn×Nandy∈Rn.Givenapriorp(f)=GP(f;0,k)withkernelkfor theunknownfunctionf,theposteriormeanandmarginalvarianceatmnewinputs˜x∈RN×mare E[˜f]=˜k(K+ε2I)−1yandV[˜f]=k(˜x,˜x)−˜k(K+ε2I)−1˜k,whereK=k(X,X)∈Rn×n isGrammatrixofthekerneland˜k=k(X,˜x)∈Rn×m.Thebulkofcomputationduring predictionarisesfromsolvingthelinearsystem(K+ε2I)z=bforsomeright-handsideb∈Rn repeatedly.Whenusingaprobabilisticlinearsolverforthistask,wecanquantifytheuncertainty arisingfromfinitecomputationaswellasthebeliefofthesolverabouttheshapeoftheGPatasetof notyetcomputedinputs.Figure3illustratesthis.Infact,wecanestimatethemarginalvarianceofthe GPwithoutsolvingthelinearsystemagainbymultiplying˜kwiththeestimatedinverseofK+ε2I. Inlarge-scaleapplications,wecantradeoffcomputationalexpenseforincreaseduncertaintyarising fromthenumericalapproximationandquantifiedbytheprobabilisticlinearsolver.Byassessing thenumericaluncertaintyarisingfromnotexploringthefullspace,wecanjudgethequalityofthe estimated GP maenad marginal variance.**

**KernelGramMatrixInversionConsideralinearproblemKx∗=b,whereKisgeneratedby aMercerkernel.Foraν-timescontinuouslydifferentiablekerneltheeigenvaluesλn(K)decay**

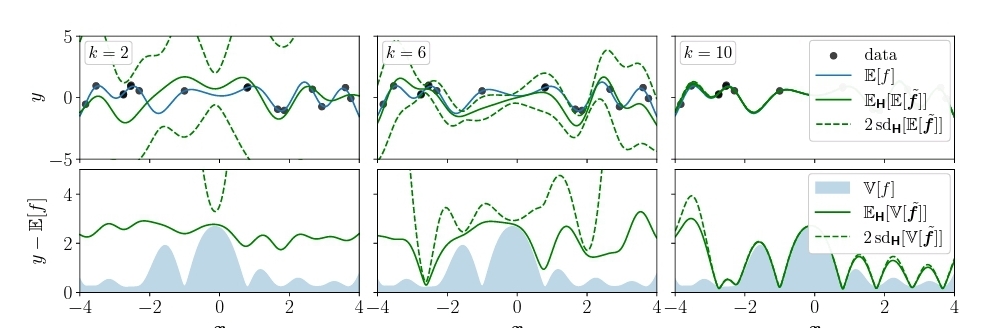


Figure 3: Numerical uncertainty in GP inference. Computing posterior mean and covariance of a GP regression using a PLS. Top: GP mean for a toy data set (n = 16) computed with increasing number of iterations k of Algorithm 1. The numerical estimate of the GP mean approaches the true mean. Note that the numerical variance is different from the marginal variance of the GP. Bottom: GPvariance and estimate of GP variance with numerical uncertainty. The GP variance estimate is computed using the estimated inverse from computing E[ ˜ f] without any additional solver iterations.

approximately as |λn| ∈ O(n−ν−1 2 ) [33]. We can make use of this generative prior information by specifying a parametrized prior mean µ(n) = lnθ0n−θ1= θ0 − θ1ln(n) for the ln-Rayleigh quotient model. Typically, such Gram matrices are ill-conditioned and therefore

K= K +ε2I is used instead, implying λ(K)i ≥ ε2. In order to assess calibration we apply various differentiable kernels to the airline delay dataset from January 2020 [34]. We compute the ln-ratio statistic

w(x∗) = 1 2 ln(tr(Cov[x])) − ln(x∗ − E[x]2)

for no calibration, calibration via Rayleigh quotient GP regression using µ(n) as a prior mean, calibration by setting

φ = ε2 and calibration using the average spectrum

φ = λk+1:n. The average ¯ w for 105/n randomly sampled test problems is shown in Table 2.2 Without any calibration the solver is generally overconfident. All tested calibration procedures reverse this, resulting in more cautious uncertainty estimates. We observe that Rayleigh quotient regression overcorrects for larger problems. This is due to the fact that its model correctly predicts K to be numerically singular from the dominant Rayleigh quotients, however it misses the information that the spectrum of Kis bounded from below by ε2. If we know the (average) of the remaining spectrum, significantly better calibration can be achieved, but often this information is not available. Nonetheless, since in this setting the majority of eigenvalues satisfy λ(K)i ≈ ε2 by choosing φ = ψ−1 = ε2, we can get to the same degree of calibration. Therefore, we can improve the solver’s uncertainty calibration at constant cost O(1) per iteration. For more general problems involving Gram matrices without damping we may want to rely on Rayleigh regression instead.

Galerkin’s Method for PDEs In the spirit of applying machine learning approaches to problems in the physical sciences and vice versa [35], we use Algorithm 1 for the approximate solution of a PDE via Galerkin’s method [9]. Consider the Ridiculed problem for the Poisson equation given by

−∆u(x,y) = f(x,y) (x,y) ∈ int Ω

u(x,y) = u∂Ω(x,y) (x,y) ∈ ∂Ω

where Ω is a connected open region with sufficiently regular boundary and u∂Ω : ∂Ω → R defines the boundary conditions. One obtains an approximate solution by projecting the weak formulation of the PDE to a finite dimensional subspace. This results in the Galerkin equation Au = f, i.e. a linear system where A is the Gram matrix of the associated bilinear form. Figure 4 shows the induced uncertainty on the solution of the Dirichlet problem for f(x,y) = 15 and u∂Ω(x,y) = (x2 −2y)2(1+sin(2πx)). The mesh and corresponding Gram matrix were computed using FENICS [36]. We can exploit two properties of Algorithm 1 in this setting. First, if we need to solve multiple related problems (Aj,fj)j, by solving a single problem we obtain an estimate of the solution to all other problems. We can successively use the posterior over the inverse as a prior for the next problem. This approach is closely related to subspace recycling in numerical linear algebra [37, 38]

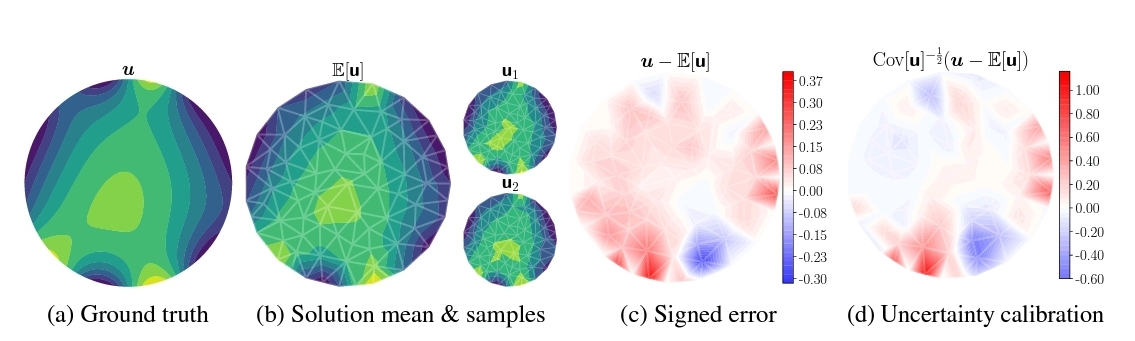


Figure 4: Solving the Ridiculed problem with a probabilistic linear solver. Figures 4a and 4b show the ground truth and mean of the solution computed with Algorithm 1 after k = 23 iterations along with samples from the posterior. The posterior on the coarse mesh can be used to assess uncertainty about the solution on a finer mesh. The signed error computed on the coarse mesh in Figure 4c shows that the approximation is better near the top boundary of Ω. Given perfect uncertainty calibration, Figure 4d represents a sample from N(0,I). The apparent structure in the plot and smaller than expected deviations in the upper part of Ω indicate the conservative confidence estimate of the solver. Second, suppose we first compute a solution in a low-dimensional subspace corresponding to a coarse discretization for computational efficiency. We can then leverage the estimated solution to extrapolate to an (adaptively) refined discretization based on the posterior uncertainty. In machine learning lingo these two approaches can be viewed as forms of transfer learning.

.

1. **Conclusion**

In this work, we condensed a line of previous research on probabilistic linear algebra into agit st self-contained algorithm for the solution of linear problems in machine learning. We proposed first principles to constrain the space of possible generative models and derived a suitable covariance class. In particular, our proposed framework incorporates prior knowledge on the system matrix or its inverse and performs inference for both in a consistent fashion. Within our framework we identified parameter choices that recover the iterates of conjugate gradients in the mean, but add calibrated uncertainty around them in a computationally lightweight manner.

To our knowledge our solver, available as part of the PROBNUM package, is the first practical implementation of this kind. In the f inal parts of this paper we showcased applications like kernel matrix inversion, where prior spectral information can be used for uncertainty calibration and outlined example use-cases for propagation of numerical uncertainty through computations. Naturally, there are also limitations remaining. While our theoretical framework can incorporate noisy matrix-vector product evaluations into its inference procedure via a Gaussian likelihood, practically tractable inference in the inverse model is more challenging. Our solver also opens up new research directions. In particular, our outlined regression model on the Rayleigh quotient may lead to a probabilistic model of the eigenspectrum.

Finally, the matrix-based view of probabilistic linear solvers could inform probabilistic approaches to matrix decompositions, analogous to the way Lanchow methods are used in the classical setting.

**Broader Impact**

Our research on probabilistic linear solvers is primarily aimed at members of the machine learning f field working on uncertainty estimation which use linear solvers as part of their toolkit.

We are convinced that numerical uncertainty induced by finite computational resources is a key missing component to be quantified in machine learning settings. By making numerical uncertainty explicit like our solver does, holistic probabilistic models incorporating all sources of uncertainty become possible. In fact, we hope that this line of work stimulates further research into numerical linear algebra for machine learning, a topic that has been largely considered solved by the community. This is first and foremost a methods paper aiming to improve the quantification of numerical uncertainty in linear problems. While methodological papers may seem far removed from application and questions of ethical and societal impact, this is not the case. Precisely due to the general nature of the problem setting, the linear solver presented in this work is applicable to a broad range of applications,