We thank the reviewers for their insightful feedback! We are encouraged they recognize the importance of probabilistic linear solvers (PLS) for ML (R1, R2, R4) and the need to quantify uncertainty arising from finite computation (R1, R4).

Contributions and Novelty We are pleased the reviewers appreciate the originality of this work (R2, R3), its methodological contributions (R1, R2, R3), the stipulated desiderata marking a roadmap for PLS research (R2, R3), the value of the proposed prior class (R2, R4), and the importance of the novel uncertainty calibration procedure (R2, R3). It remedies a primary shortcoming of PLS, enables probabilistic stopping criteria and estimates $\log(\det(\boldsymbol{A}))$ (see below). We also provide the first practical open-source implementation of a PLS returning distributions over $\boldsymbol{A}$, $\boldsymbol{A}^{-1}$ and $\boldsymbol{x}$. This is an important step towards a framework which targets computational resources "to reduce uncertainty as required for a downstream task" (R4). The returned posterior means by the PLS are low-rank approximations to $\boldsymbol{A}$ and $\boldsymbol{A}^{-1}$, which among others find use in kernel methods. The returned covariances provide a bound to the error and their structure can be exploited for novel use cases, e.g. as proposed for probabilistic mesh refinement in Galerkin's method.

Bayesian Interpretation and Prior Class (R1, R3) The generic inference procedure in Section 2 for a given prior covariance $\boldsymbol{W}_0^{\mathsf{A}} \otimes \boldsymbol{W}_0^{\mathsf{A}}$ is Bayesian since it relies on Bayes' theorem. Algorithm 1 performs sequential Bayesian updates for single action - observation pairs $(\boldsymbol{s}_i, \boldsymbol{y}_i)$. This can be seen by recognizing that the posterior (see Section 2.1) is of the same form as the prior (for any $1 \leq k \leq n$). Guided by the desiderata in Table 1, we restrict the $n \times n$ DoFs in the prior. This results in the proposed prior class in eq. (3). This prior and our calibration procedure depend on the entire collected 'data' during a run of Algorithm 1. When using the proposed prior class our method is thus not strictly Bayesian in the philosophical sense, but empirical Bayesian (i.e. it uses data to fit hyperparameters of the prior). As this approach is standard in GP regression (where kernel parameters are set by type-II maximum likelihood), we neglected to make this distinction. We will clarify this in the final version. This leaves the question how the algorithm is realizable for the proposed prior (3) given its dependence on future data. The posterior mean in Section 2.1 only depends on $\boldsymbol{W}_0^{\mathsf{A}}\boldsymbol{S} = \boldsymbol{Y}$ *not* on $\boldsymbol{W}_0^{\mathsf{A}}$ alone. By eq. (3), this product is given by the previously made observations $\boldsymbol{Y}$. Similar reasoning applies for the inverse. Now, the posterior covariances do depend on $\boldsymbol{W}_0^{\mathsf{A}}$, resp. $\boldsymbol{W}_0^{\mathsf{H}}$ alone, but during a run of Algorithm 1, we only require $\mathrm{tr}(\mathrm{Cov}[\mathsf{x}])$ for the stopping criterion. We show in Section S4.5 under the assumptions of Theorem 2 how to compute this at any iteration i without access to future actions and observations.

Calibration Procedure (R1, R3) Calibration ensures that the uncertainty returned by the solver has the right scale, i.e. it bounds the expected (relative) error (see Sections 2.2 and S4.5). Since the policy π only depends on the posterior mean $\mathbb{E}[\mathsf{H}]$ and not the covariance, the hyperparameters $\boldsymbol{\Phi}, \boldsymbol{\Psi}$ and thus calibration do *not* change the solution estimate $\boldsymbol{x}_i$ only its associated covariance at iteration i. While structure in the uncertainty is preserved, miscalibration negatively impacts the probabilistic termination criterion. In our experiments in Table 2 the solver without calibration terminates early, since it is overconfident and thus has larger error than with calibration. *Why not return* $\mathcal{N}(\hat{x}^{\mathrm{CG}}, \mathrm{span}(\boldsymbol{S})^{\perp}$*(GP output at* $t+1$*))?* (R1): This is similar to Algorithm 1's output assuming Theorem 2 holds. However, it omits prior knowledge about the space $\mathrm{span}(\boldsymbol{S})$ explored by the algorithm (e.g. information about the dominant eigenspectrum). Further, only using the GP prediction at $t+1$ implies that the algorithm's uncertainty about the action of $\boldsymbol{A}$ in $\boldsymbol{S}^{\perp}$ is of the same order as the next eigenvalue. This ignores any information about eigenvalues $\lambda_{t+2}, \ldots, \lambda_n$ contained in Rayleigh quotients and a priori known decay patterns for specific matrix classes.

Importance of Noise (R4) When referring to noise, we consider matrix-vector products of the form $\boldsymbol{v} \mapsto (\boldsymbol{A} + \mathsf{E}_i)\boldsymbol{v}$, where $\mathsf{E}_i \in \mathbb{R}^{n\times n}_{\mathrm{sym}}$ is Gaussian with zero mean. CG fails to converge in such a setting. While this approach can model floating-point arithmetic, typically in ML settings noise from subsampling dominates. An important example is large-scale empirical risk minimization. Due to memory constraints data needs to be batched and thus only approximate Hessian-vector products $\boldsymbol{H}_{\mathrm{batch}}\boldsymbol{v} = (\boldsymbol{H} + \mathsf{E}_{\mathrm{batch}})\boldsymbol{v}$ are available. One could use a PLS for Hessian-free optimization in this setting. This results in a trade-off between computing an accurate Hessian by sampling new batches in each iteration of the solver and taking more optimization steps in parameter space. This approach results in an optimizer which interpolates between SGD and Newton's method, depending on the batch size and number of PLS iterations k.

Applications *Transfer Learning* (R1): Using a posterior from a related problem as a prior on a new problem has the advantage over only setting $\boldsymbol{x}_0^{\mathrm{new}} = \boldsymbol{x}_k^{\mathrm{prev}}$, that uncertainty in already explored directions $\boldsymbol{S}$ is low. Hence, if the new problem $(\boldsymbol{A}^{\mathrm{new}}, \boldsymbol{b}^{\mathrm{new}})$ is similar, the covariance will contract faster. In turn also convergence will be faster (as in subspace recycling). *Kernel matrix inversion* (R4): We recognize the variety of methods available for Gram matrix inversion in the Gaussian process setting. While a comparison for different priors adapted to the kernel choice vs. a set of inducing point methods is an interesting experiment, this would have exceeded the scope of this paper. *Log-Determinant Estimation* (R3, R4): The PLS can estimate the log-determinant in $\mathcal{O}(n)$ using the proposed ln-Rayleigh regression model for uncertainty calibration via $\ln(\det(\boldsymbol{A})) = -\sum_{i=1}^{n} \ln R(\boldsymbol{A}, \boldsymbol{s}_i) \approx -(\sum_{i=1}^{k} \ln R(\boldsymbol{A}, \boldsymbol{s}_i) + \sum_{i=k+1}^{n} \mathbb{E}[\ln R_i \mid \mathsf{A}, \boldsymbol{S}])$. *Galerkin's Method* (R2, R4): When using a PLS as part of Galerkin's method the posterior (predictive) on a refined mesh can be derived analytically (see Proposition S6). We leave comparisons to multi-grid methods for future work.

Other (R3, R4) *Reorthogonalization* (R3) is a consequence of the policy choice $\boldsymbol{s}_i = -\mathbb{E}[\mathsf{H}]\boldsymbol{r}_i$, where $\mathbb{E}[\mathsf{H}]$ depends on all previous search directions as opposed to just $\boldsymbol{s}_{i-1}$ for (naive) CG. *Unification of PLS theory* (R4): Bartels et al. [13] demonstrate that the matrix-based view generalizes the solution-based view. We focus on presenting a unified matrix-based framework, which among others, connects the inference perspectives for $\boldsymbol{A}$ and $\boldsymbol{A}^{-1}$ in a rigorous way.