Learning Covariances for Estimation with Constrained Bilevel Optimization

Mohamad Qadri, Zachary Manchester, and Michael Kaess

***Abstract*— We consider the problem of learning error covariance matrices for robotic state estimation. The convergence of a state estimator to the correct belief over the robot state is dependent on the proper tuning of noise models. During inference, these models are used to weigh different blocks of the Jacobian and error vector resulting from linearization and hence, additionally affect the stability and convergence of the non-linear system. We propose a gradient-based method to estimate well-conditioned covariance matrices by formulating the learning process as a constrained bilevel optimization problem over factor graphs. We evaluate our method against baselines across a range of simulated and real-world tasks and demonstrate that our technique converges to model estimates that lead to better solutions as evidenced by the improved tracking accuracy on unseen test trajectories.**

I. Introduction

Robot state estimation is the problem of inferring the state of a robot (a set of geometric or physical quantities such as position, orientation, contact forces, etc.) given sensor measurements. The problem is typically formulated as Maximum a Posteriori (MAP) inference over factor graphs where each node (robot state $\mathbf{x}_i$) is connected to other states via factors (potentials) ϕ_i which are distilled from sensor measurements:

$$\mathbf{x}^{\text{MAP}} = \underset{\mathbf{x}}{\operatorname{argmax}} \prod_{i}^{N} \phi_i(\mathbf{x}_i; \theta_i, z_i) \tag{1}$$

The factors typically assume the form:

$$\phi_i \propto \exp\left(-\frac{1}{2}||g_i(\mathbf{x}_i) - z_i||^2_{\theta_i}\right) \tag{2}$$

which leads, after taking the negative log, to the equivalent non-linear least squares objective:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2}||g_i(\mathbf{x}_i) - z_i||^2_{\theta_i} \tag{3}$$

where $\mathbf{x}$ are the state variables, $\mathbf{x}_i$ a subset of $\mathbf{x}$, $\mathbf{z} = \{z_i\}$ the sensor measurements, and g the prediction function which maps states onto the sensor's measurement manifold. *Noise Models* $\{\theta_i\} = \boldsymbol{\theta}$ affect the loss landscape and, as typical in data assimilation procedures [1], correspond to error covariance matrices. These parameters dictate the weight assigned to each measurement which, given an optimal parameter set $\boldsymbol{\theta}^*$, should ideally correlate with the uncertainty of each sensor. Hence, $\{\theta_i\}$ when inaccurately defined will lead to suboptimal solutions. On the other hand, and specifically because each θ_i is used to scale different error and Jacobian terms after relinearization, the condition number of each θ_i is correlated with the overall numerical conditioning and stability of problem (3). Traditionally the set $\boldsymbol{\theta}$ is manually tuned per application. Nevertheless, alternative approaches exist for estimating $\boldsymbol{\theta}$ from data. Zero-order optimization techniques such as [2]–[5] can be leveraged but can also quickly become sample inefficient. Other methods attempt to minimize the final tracking error loss by performing gradient-based parameter updates. These techniques generally either 1) rely on unrolling the optimizer which is sensitive to various hyperparameters [6] or 2) assume that the selected graph optimization algorithm is differentiable [7]. Note however that this assumption does not hold for state-of-the-art optimizers such as iSAM2 [8] due to dependence on relinearization thresholds and non-differentiable operations such as removal/insertion of tree nodes. In addition, these methods do not consider the conditioning of the learned parameters. Hence in this work, we make three key contributions:

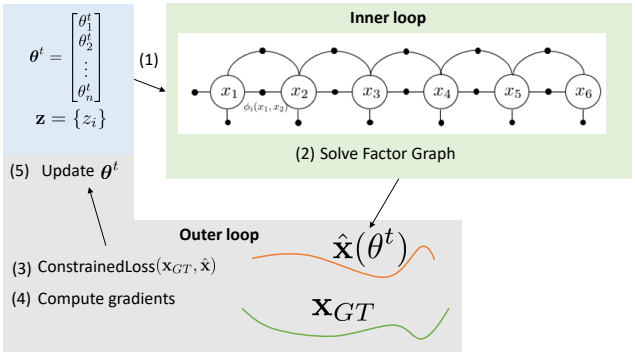

Fig. 1: At each iteration, the parameters $\boldsymbol{\theta}^t$ serve as inputs to to the least squares solver. The inner loop optimization outputs a trajectory estimate $\hat{\mathbf{x}}$ which depends on $\boldsymbol{\theta}^t$. The Jacobian of $\hat{\mathbf{x}}$ with respect to $\boldsymbol{\theta}^t$ is computed via numerical differentiation and used to compute the gradient of the loss with respect to the parameters. Finally, this gradient is used to update the parameters $\boldsymbol{\theta}$.

- We formulate the problem as a bilevel optimization problem over factor graphs and use numerical differentiation to efficiently estimate the required gradients.
- We propose a technique for estimating well-conditioned positive definite matrices by incorporating hard condition number constraints into the learning procedure.
- We evaluate our approach on different synthetic navigation and real-world planar pushing examples in incremental estimation settings.

[1]M. Qadri, Z. Manchester, and M. Kaess are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {mqadri, zmanches, kaess}@andrew.cmu.edu

This work was partially supported by Office of Naval Research award N00014-21-1-2482 and National Science Foundation grant IIS-2008279.

II. Background and Related Work

A. Filtering and Smoothing for State Estimation

Early state estimation techniques such as Kalman Filters [9, 10] rely on the Markov assumption to enable real-time performance. Various methods were proposed to better estimate the process and measurement noise models [11, 12], or to make these filters differentiable [13]–[17]. However, these techniques do not allow for re-linearizing past states which can lead to convergence to poor solutions. Hence, state-of-the-art robotic state estimation algorithms transitioned to factor graph smoothing-based approaches. These methods encode the inherent underlying temporal structure, providing efficient ways to relinearize past estimates and eliminating the need to marginalize past states [8, 18, 19]. Under Gaussian assumptions, the smoothing problem is equivalent to a non-linear least squares objective weighted by error covariance matrices which often require manual tuning tailored to each application [20, 21].

B. Learned Components in Factor-Graph Inference

Recent works incorporate learned components into factor-graph-based inference models. [22] learns depth codes which are subsequently used to compute various factors for dense monocular SLAM. [23] learns observation models which predict relative sensor poses then used in a factor graph formulation to predict the pose of manipulated objects. Similarly, [24] learns a model to predict relative robot poses from non-sequential ground penetrating radar image pairs then used in a factor graph in GPS-denied localization. However, these methods use surrogate losses for learning independently of the graph optimizer or final tracking error and generally require separate tuning of noise models. While traditionally these models have been manually tuned, novel strategies have emerged to learn them directly from data. [6, 25] proposed differentiating through the argmin operator in Eq. 3 by unrolling the optimizer. However, these techniques are typically sensitive to hyperparameters such as the number of unrolling steps [26] and, in addition, can suffer from vanishing as well as high bias and variance gradients. Few methods use variational inference techniques to refine noise models when no groundtruth trajectories are available [27, 28] or use groundtruth trajectories to learn time-correlated measurement noise models [29]. These methods target batch state estimation problems and do not consider the conditioning of the estimated matrices. Recently, a novel method *LEO* [30] capitalizes on the probabilistic view offered by iSAM2 (as a solver of Eq.1) to provide a way to learn observation models, by minimizing a novel tracking error. In essence, at every training iteration, LEO samples trajectories from the posterior distribution (a joint Gaussian distribution over the states) and the deviation with respect to the groundtruth trajectory is minimized using an energy-based loss. However, LEO exhibits slow convergence speed due to its dependence on sampling from high-dimensional probability distributions and is prone to convergence to poor local minimas.

C. Covariance Estimation in Mathematical Statistics

The estimation of well-conditioned and stable covariances has garnered considerable attention within the mathematical statistics community as their use spans different statistical methods and practical applications ranging from numerical weather prediction to financial portfolio optimization. [31] proposes incorporating a prior involving the nuclear norms of the covariance and its inverse in the estimation process to bound the eigenvalues. [32] performs maximum likelihood estimation of covariances subject to hard condition number constraints. [33] proposes new theoretical perspectives on reconditioning covariances using ridge regression or the minimum eigenvalue method. In this work, we propose to estimate error covariance matrices while imposing condition number constraints for the task of incremental robot state estimation.

D. Conditioning of Non-linear Least-Squares Problems

Iterative methods [34] solve Eq. 3 by relying on a sequence of linearized subproblems. Each subproblem involves solving the linear system $\mathbf{A}\Delta = \mathbf{b}$ where the stability of the solution is influenced by the condition number of $\mathbf{A} : \kappa(\mathbf{A})$. In fact, it has additionally been shown that the convergence rate of specific solvers of the normal equation $\mathbf{A}^T\mathbf{A}\Delta = \mathbf{A}^T\mathbf{b}$, such as conjugate gradient (CG), is upper bounded by $\sqrt{\kappa(\mathbf{A}^T\mathbf{A})}$. When $\sqrt{\kappa(\mathbf{A}^T\mathbf{A})}$ is high and without proper preconditioning, the performance of CG methods can be especially poor. In section III-C.1, we show how estimating well-conditioned error covariances matrices is correlated with the stability of the linearized system.

III. Method

Our goal is to learn the parameters $\{\theta_i\}$ using a gradient-based method from groundtruth robots trajectories $\mathbf{x}_{\text{GT}}$. In this work, we view any unconstrained non-linear least squares solver (e.g. Levenberg–Marquardt (LM) or iSAM2), as a function $f(\boldsymbol{\theta}) : \mathcal{S}^{n_1}_{++} \times \ldots \times \mathcal{S}^{n_p}_{++} \rightarrow \mathcal{X}$ with $\boldsymbol{\theta} = \{\theta_i \mid \theta_i \in \mathcal{S}^{n_i}_{++}\}$ which, given an initial estimate $\mathbf{x}^0 \in \mathcal{X} : \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$, returns an estimate of the optimal state $\hat{\mathbf{x}} \in \mathcal{X}$ after performing N update steps. Here, $\mathcal{M}_i$ is a Lie Group (e.g. the special Euclidean group $SE(n)$) and $\mathcal{S}^{n_i}_{++}$ is the set of $n_i \times n_i$ positive definite matrices. Consider the following bilevel optimization procedure (also illustrated in Fig. 1):

$$\text{Inner Loop: } f(\boldsymbol{\theta}) = \underset{\mathbf{x}}{\operatorname{argmin}}\, H(\mathbf{x}, \boldsymbol{\theta}; \mathbf{z}) = \hat{\mathbf{x}}(\boldsymbol{\theta}) = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \frac{1}{2} ||g_i(\mathbf{x}_i) - z_i||^2_{\theta_i} \quad (4)$$

$$\text{Outer Loop: } \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(f(\boldsymbol{\theta}), \mathbf{x}_{\text{GT}}) \quad (5)$$

where $\mathcal{L}$ is a differentiable loss function capturing the deviation of the estimate $f(\boldsymbol{\theta})$ from the GT. At every iteration, Eq. 5 outputs a set $\boldsymbol{\theta}^t$. Or, in other words, selects an updated loss function for the inner loop optimization such that solving problem 4 leads to a minima/solution that is closer to the groundtruth trajectory $\mathbf{x}_{\text{GT}}$. Let $h(\mathbf{x}, \boldsymbol{\theta}) := \frac{\partial H}{\partial \mathbf{x}}$. The graph of f consists of all points satisfying first order-optimality conditions of problem 4: $\text{gph}(f) = \{(\boldsymbol{\theta}, f(\boldsymbol{\theta})) \mid f(\boldsymbol{\theta}) = H(\hat{\mathbf{x}}, \boldsymbol{\theta}) \text{ and } h(\hat{\mathbf{x}}, \boldsymbol{\theta}) = 0\}$. By the chain rule, the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ requires an estimate of $\frac{\partial f}{\partial \boldsymbol{\theta}}$ (i.e. the Jacobian of the solution with respect to the parameter vector) and by the

implicit function theorem [35], this Jacobian (i.e. $\frac{\partial f}{\partial \boldsymbol{\theta}}$) exists and can be computed as done in existing work in convex optimization [36, 37].

Theorem 1: ***The Implicit Function Theorem:*** *Let* $\hat{\mathbf{x}}(\boldsymbol{\theta}) := \{\mathbf{x} \mid h(\mathbf{x}, \boldsymbol{\theta}) = 0\}$ *where* $\mathbf{x} \in \mathcal{X}$ *and* $\boldsymbol{\theta} = \{\theta_i \mid \theta_i \in \mathcal{S}^n_{++}\}$. *Let* h *be continuously differentiable in the neighborhood of* $(\hat{\mathbf{x}}, \boldsymbol{\theta})$ *namely* $\frac{\partial h(\hat{\mathbf{x}}(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \mathbf{x}}$ *be nonsingular. Then:*

$$\frac{\partial f}{\partial \boldsymbol{\theta}} = \frac{\partial \hat{\mathbf{x}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\left(\frac{\partial h(\hat{\mathbf{x}}(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \mathbf{x}}\right)^{-1} \frac{\partial h(\hat{\mathbf{x}}(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{6}$$

Note that the original theorem is formulated for functions operating on vector spaces. However, the theorem can readily be extended to other manifolds by applying the appropriate group operations [38]. The partial derivatives in Eq. 6 can be derived and computed analytically. However, since the size of the parameter vector $\boldsymbol{\theta}$ is typically small – first, because each θ_i is associated with a single physical sensor and second, since each θ_i has a small number of free parameters (e.g. a maximum of 6 for elements residing in $SE(2)$) – numerical differentiation proved to be efficient, especially when coupled with parallelization on CPU.

A. Numerical Jacobians over Lie Groups

The left Jacobian of functions acting on manifolds $f : \mathcal{N} \to \mathcal{M}$ is defined as the linear map from the Lie algebra $T_{\mathcal{E}}\mathcal{N}$ of $\mathcal{N}$ to $T_{\mathcal{E}}\mathcal{M}$, the Lie algebra of $\mathcal{M}$:

$$\frac{^{\mathcal{E}}Df(\mathcal{Y})}{D\mathcal{Y}} = \lim_{\tau \to 0} \frac{f(\tau \oplus \mathcal{Y}) \ominus f(\mathcal{Y})}{\tau} \tag{7}$$

$$= \lim_{\tau \to 0} \frac{\text{Log}(f(\text{Exp}(\tau) \circ \mathcal{Y}) \circ f(\mathcal{Y})^{-1})}{\tau} \tag{8}$$

where $\mathcal{Y} \in \mathcal{N}$, τ is a small increment defined on $T_{\mathcal{E}}(\mathcal{N})$. The Log operator maps elements from a Lie Group to its algebra while the Exp operator maps elements from the algebra to the group. $\oplus$, $\ominus$, and $\circ$ are the plus, minus, and composition operators respectively [39] where:

$$\tau \oplus \mathcal{Y} = \text{Exp}(\tau) \circ \mathcal{Y} \tag{9}$$

$$\tau = \mathcal{Y}_1 \ominus \mathcal{Y}_2 = \text{Log}(\mathcal{Y}_1 \circ \mathcal{Y}_2^{-1});\ \mathcal{Y}_1, \mathcal{Y}_2 \in \mathcal{N} \tag{10}$$

In this work, $\mathcal{N} = \mathcal{S}^{n_1}_{++} \times \ldots \times \mathcal{S}^{n_p}_{++}$ and $\mathcal{M} = \mathcal{X}$. Additionally, we assume that each vector θ_i corresponds to the non-zero elements of some corresponding diagonal positive definite matrix $\Sigma_i \in \mathcal{S}^{n_i}_{++}$. i.e., we define the following map:

$$\theta_i = \text{diag}^{-1}(\Sigma_i) \in \mathbb{R}^{n_i} \tag{11}$$

where the operator diag^{-1} constructs a vector from the diagonals of a matrix. Hence, $\tau \in \mathbb{R}^{n_i}$ and the operator $\oplus$ in Eq. 7 is the standard addition on vector space $\mathbb{R}^{n_i}$.

B. Constrained Tracking Loss

Consider a parameter estimate $\boldsymbol{\theta} \in \mathbb{R}^m$ with $m = \sum_{i=1}^{p} n_i$. Let $\mathcal{D}$ be the training set, T be the total number of states in groundtruth trajectory $\mathbf{x}_{\text{GT}}$, and D be the sum of the Lie algebra dimensions of all states. Then the outer loss is the constrained mean squared error between the estimated trajectory and $\mathbf{x}_{\text{GT}}$:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} ||f(\boldsymbol{\theta}) \ominus \mathbf{x}_{\text{GT}}||^2_2 \tag{12}$$

$$\text{subject to: } \boldsymbol{\lambda}_i^{\min} \leq \theta_i \leq \boldsymbol{\lambda}_i^{\max} \quad \forall\, \theta_i \in \boldsymbol{\theta}$$

where $\boldsymbol{\lambda}_i^{\min} > 0$ and $\boldsymbol{\lambda}_i^{\max} > \boldsymbol{\lambda}_i^{\min}$ are vectors of minimum and maximum eigenvalues, defined per coordinate and per θ_i, which are enforced to better condition the estimated diagonal matrices as well as ensure their positive definiteness. In other words, these constraints allow to upper bound the condition number $\kappa(\theta_i) = \frac{\boldsymbol{\lambda}_i^{\max}}{\boldsymbol{\lambda}_i^{\min}}$ of the estimated matrices. This, in turn, contributes to better conditioning of the overall linearized system during online inference (see III-C.1). Hence, since the function $\mathcal{L}(\boldsymbol{\theta})$ is non-convex, the constraints help in steering the optimization towards more desirable minimas. This constrained objective is solved by performing iterative Frank-Wolfe update steps [40]. Algorithm 1 outlines the training process. Note that the non-linear least squares (NLLS) in line 3 can be solved by any NLLS optimizer. i.e. our method is agnostic to the choice of optimizer, whether it is differentiable (e.g., Levenberg-Marquardt) or non-differentiable (e.g., iSAM2).

Algorithm 1 Training Loop

1: **Input:** Factor Graph $\mathcal{F}$, initialization $\mathbf{x}^0$
2: **while** itr $<$ max_iter
3: $\quad f(\boldsymbol{\theta}^t) =$ Solve[NLLS($\mathcal{F}$, $\mathbf{x}^0$)]
4: $\quad$ Estimate $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ using Eq. 15
5: $\quad \boldsymbol{\theta}^{t+1}$ = Frank-Wolfe-Step($\frac{\partial L}{\partial \boldsymbol{\theta}}, \boldsymbol{\theta}^t$)
6: $\quad$ itr = itr+1

Algorithm 2 Frank-Wolfe-Step

1: **Inputs**: $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}, \boldsymbol{\theta}^t$
2: Solve (Direction Finding) :
3: $\quad \mathbf{s}^* = \min_s s^T \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ subject to $\boldsymbol{\lambda}^{\min} \leq \mathbf{s} \leq \boldsymbol{\lambda}^{\max}$
4: $\alpha = \frac{2}{M + \text{itr}}$ (Step Size) $\quad \triangleright$ Where M is a parameter
5: return $\boldsymbol{\theta}^t + \alpha(\mathbf{s}^* - \boldsymbol{\theta}^t)$ (Updated parameters)

The linear program in Alg. 2 line 3 has negligible computational burden (m decision variables and $2m$ constraints) and can be solved efficiently using methods such as Interior Point or Dual-Simplex. To estimate, the gradient in Alg. 1 line 4, we propose to perform numerical differentiation by taking advantage of the small parameter space. Taking the gradient of Eq. 12, we get:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} S(f(\boldsymbol{\theta}))^T \cdot \underbrace{(f(\boldsymbol{\theta}) \ominus \mathbf{x}_{\text{GT}})}_{\in \mathbb{R}^{TD}\ (\cong T_{\mathcal{E}}\mathcal{X})} \tag{13}$$

where $S(f(\boldsymbol{\theta})) \in \mathbb{R}^{TD \times m}$ is a matrix such that each row r is equal to:

$$S(f(\boldsymbol{\theta}^t))_r = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_{ij}} \tag{14}$$

$$= \lim_{\tau_{ij} \to 0} \frac{\text{Log}(f(\tilde{\boldsymbol{\theta}}) \circ f^{-1}(\boldsymbol{\theta}))}{\tau_{ij}} \tag{15}$$

where $\tilde{\theta}_{ij} = \theta_{ij} + \tau_{ij}$ and $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}$ otherwise. By the implicit function theorem, $\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_{ij}}$ exists and is estimated using finite differencing by perturbing the parameter θ_{ij} by τ_{ij}.

C. Remarks

1) Condition Number Constraints: The noise models $\boldsymbol{\theta}$ are used to weight the contribution of the error terms during inference and their eigenvalue spread is correlated with the numerical condition of the linear system resulting from the linearization of Eq. 3. Specifically, linearizing Eq. 3 at iterate $\mathbf{x}^t$, we obtain:

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2} ||g_i(\mathbf{x}_i^t) + \frac{\partial g_i}{\partial \mathbf{x}_i} \Delta_i - z_i||_{\theta_i}^2 \tag{16}$$

which as shown in [41] can be re-written as:

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2} ||\theta_i^{-\frac{1}{2}} \frac{\partial g_i}{\partial \mathbf{x}_i} \Delta_i + \theta_i^{-\frac{1}{2}} (g_i(\mathbf{x}_i^t) - z_i)||_2^2 \tag{17}$$

Collecting all Jacobians and prediction errors, the system can be rewritten as:

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{A}\Delta - \mathbf{b}||_2^2 \tag{18}$$

Since both the Jacobian $\mathbf{A}$ and error vector $\mathbf{b}$ are composed of elements which are matrix multiplied by some $\theta_i^{-\frac{1}{2}}$, the eigenvalue spread $\bar{s}$ which we define as the maximum eigenvalue divided by the minimum eigenvalue over all θ_i:

$$\bar{s} = \frac{\max\{eig(\theta_i) \text{ for } \theta_i \in \boldsymbol{\theta}\}}{\min\{eig(\theta_i) \text{ for } \theta_i \in \boldsymbol{\theta}\}} \tag{19}$$

is correlated with the conditioning of matrix $\mathbf{A}$ and the numerical stability of the system in Eq. 18 [1]. Our method enables to constrain $\bar{s}$ by incorporating hard constraints into the learning process.

2) Inner Loop Initialization during Training: We borrow from the imitation learning literature and initialize the inner loop optimizer ($\mathbf{x}^0$ in alg. 1 line 3) with the GT trajectory *during training*. These trajectories form our training set and hence, are known a priori. Such initialization has been shown to improve convergence speed and stability [42] . Note that this is different from baselines such as LEO which, during training, solves the inference problem in Eq. 3 incrementally while initializing $\mathbf{x}^t = \mathbf{x}^{t-1}$ at each new timestep t.

IV. Results

A. Baselines

We compare our method against three baselines: Nelder-Mead [5], the Modified Powell's method [4], and LEO [30]. *Nelder-Mead* is a gradient-free simplex-based optimization algorithm that aims at decreasing the value of a given function f at the vertices of a working simplex by performing a sequence of transformation (i.e. reflection, shrinkage, etc.). *Powell's method* is similarly gradient-free and minimizes f by performing a series of one-dimensional line minimization along some search directions. Both methods minimize Eq. 5 where $\mathcal{L}$ is the L2 loss. We use SciPy's implementation of these algorithms and run them until convergence. *LEO* minimizes the following outer-loop energy-based loss:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}_{\text{gt}}^j, \mathbf{z}^j) \in \mathcal{D}} E(\boldsymbol{\theta}, \mathbf{x}_{\text{gt}}^j; \mathbf{z}^j) + \log \int_{\mathbf{x}} e^{-E(\boldsymbol{\theta}, \mathbf{x}; \mathbf{z}^j)} d\mathbf{x} \tag{20}$$

where $(\mathbf{x}_{\text{gt}}^j, \mathbf{z}^j)$ is a groundtruth sample from the training set $\mathcal{D}$, the energy $E(\boldsymbol{\theta}, \mathbf{x}; \mathbf{z}) := H(\boldsymbol{\theta}, \mathbf{x}; \mathbf{z})$ (as defined in Eq. 4), and the integral is over the space of trajectories. We use the official implementation of LEO by Paloma et al.

B. Experimental Details

We perform experiments on different simulated and real tasks and use the initialization proposed in III-C.2 for all methods. For all experiments, we start the learning procedure with initial $\boldsymbol{\theta}^0$ values that steer the optimized trajectories far away from the GT or in other words, $\boldsymbol{\theta}^0$ that are far from the underlying latent unknown parameters. The implementations of Nelder-Mead and Powell in Scipy allow the specification of bound constraints on the parameters. Hence, we perform 2 sets of experiments where 1) the bounds are loosely specified and effectively only used to ensure the positive definiteness of the estimated matrices (i.e. $\boldsymbol{\lambda}^{\text{min}} > \mathbf{0}$) and 2) with tight bounds (denoted with (C) in tables I and II) where $\lambda^{\text{min}} = 0.1$ and $\lambda^{\text{max}} = 10$ are defined such that the maximum condition number $\kappa^{\text{max}} = 100$ and hence, maximum eigenvalue spread $\bar{s}^{\text{max}} = 100$. Note that LEO does not support constraints. After training, the optimal regressed values $\boldsymbol{\theta}^*$ by each method are used as noise models for incremental inference on unseen test samples and the output trajectories are compared against the GT in terms of the root mean square error (RMSE). We use the GTSAM C++ package as the factor graph optimization library and its implementation of iSAM2 as the inner-loop optimizer.

C. 2D Navigation

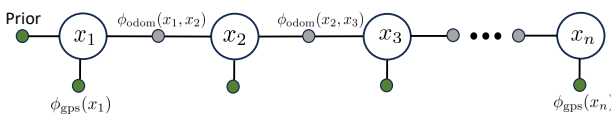

Fig. 3: The factor graph used to solve the synthetic robot navigation estimation problems.

We use 4 synthetic planar (i.e. in $SE(2)$) robotic navigation datasets consisting of GPS and odometric measurements all generated from a different set of parameters $\boldsymbol{\theta}^{\text{latent}}$. Datasets D_1 and D_2 are generated using fixed parameters $\{\theta_{\text{gps}}, \theta_{\text{odom}}\}_{D_i}$ while D_3 and D_4 use varying parameters which are functions of binary variable p: $\{\theta(p)_{\text{gps}}, \theta(p)_{\text{odom}}\}_{D_i}$. p simulates a light detector indicating whether the robot is operating in indoor or outdoor environments. We use 5 sample trajectories for training and 20 for testing. Fig. 3 shows the structure of the graph used to estimate the robot trajectory: A unary GPS factor $\phi_{\text{gps}}(x_i)$ is added to each pose x_i while a binary odometry factor $\phi_{\text{odom}}(x_i, x_{i-1})$ is specified between poses. To simulate realistic robot navigation trajectories, Gaussian noise $\sim \mathcal{N}(0, \theta_{\text{odom}})$ is injected to groundtruth relative odometry measurements while Gaussian noise $\sim \mathcal{N}(0, \theta_{\text{gps}})$ is added to absolute groundtruth GPS measurements.

[1] With factorization-based methods such as QR, poor conditioning can lead to the loss of significant digits or even yield incorrect solutions.

Table I shows the RMSE of the output trajectories compared against GT. Other than D_1 and D_4, with constraints enabled, for which Nelder-Mead or Powell generates parameters with slightly better rotation accuracy, our technique consistently converges to parameters $\boldsymbol{\theta}^*$ that lead to better tracking accuracy on all remaining unseen test trajectories.

TABLE I: Average RMSE over the testing set for each navigation dataset. Translation and rotation errors are in m and rad respectively.

Loose bounds		Initial	LEO	NMead	Powell	Ours
D_1	Transl	1.309	0.401	0.293	2.617	**0.230**
	Rot	1.184	0.070	**0.049**	0.380	0.051
D_2	Transl	2.597	1.226	1.473	7.913	**0.775**
	Rot	2.061	0.572	0.941	0.259	**0.056**
D_3	Transl	0.918	0.640	0.482	3.960	**0.172**
	Rot	0.793	0.318	0.170	0.283	**0.095**
D_4	Transl	0.805	0.671	0.592	0.465	**0.209**
	Rot	1.123	0.816	1.092	0.212	**0.073**
Tight bounds		Initial		NMead(C)	Powell (C)	Ours (C)
D_1	Transl	1.309	-	0.231	0.254	**0.229**
	Rot	1.184	-	**0.048**	0.050	0.051
D_2	Transl	2.597	-	1.467	0.838	**0.777**
	Rot	2.061	-	0.849	0.167	**0.051**
D_3	Transl	0.918	-	0.567	0.376	**0.173**
	Rot	0.793	-	0.271	**0.091**	**0.091**
D_4	Transl	0.805	-	0.681	0.356	**0.210**
	Rot	1.123	-	1.090	**0.074**	0.075

D. Real-world planar pushing

We perform experiments on real-world tactile pushing example where an end-effector pushes an object and the goal is to estimate the pose of both the end-effector and that of the object from sensor data. Groundtruth end-effector and object poses are obtained using a motion capture system. We follow the formulation in [23] where the graph includes relative pose factor ϕ_{tac} (in which measurements predict the difference in the pose of the end-effector relative to the object between times t and $t-n$ with $n > 1$), quasi-static dynamics factor ϕ_{dyn}, geometric constraints ϕ_{geo}, and end-effector pose priors ϕ_{prior}. Each factor involves a different parameter $\in \{\theta_{\text{tac}}, \theta_{\text{dyn}}, \theta_{\text{geo}}, \theta_{\text{prior}}\} = \boldsymbol{\theta}$ which collectively form our optimization set. We perform two sets of experiments where 1) ϕ_{tac} is provided by a noisy oracle (i.e. simulating an accurate sensor with confident measurements) and 2) ϕ_{tac} being computed from a stream of real tactile measurement.

TABLE II: Test RMS translation and rotation errors in cm and rad respectively (averaged over the testing set). We report end-effector (ee) and object (obj) trajectory error for both the noisy oracle and network experiments.

ELLIP

Loose bounds			Initial	LEO	NMead	Powell	Ours
Noisy Ora	ee	Transl	0.784	0.569	0.817	**0.012**	**0.012**
		Rot	0.004	0.041	0.004	$6e^{-5}$	$\mathbf{2e^{-5}}$
	obj	Transl	2.625	1.875	2.581	0.510	**0.273**
		Rot	0.234	0.209	0.229	0.018	**0.008**
Network	ee	Transl	0.821	0.274	0.845	0.015	**0.012**
		Rot	0.004	$9e^{-4}$	0.004	$1e^{-4}$	$\mathbf{1e^{-5}}$
	obj	Transl	2.431	2.456	2.477	1.543	**0.982**
		Rot	0.271	0.162	0.265	0.168	**0.155**
Tight bounds			Initial		NMead (C)	Powell (C)	Ours (C)
Noisy Ora	ee	Transl	0.784	-	0.710	**0.006**	0.018
		Rot	0.004	-	0.005	$\mathbf{2e^{-5}}$	$\mathbf{2e^{-5}}$
	obj	Transl	2.625	-	1.534	0.364	**0.287**
		Rot	0.234	-	0.031	0.023	**0.007**
Network	ee	Transl	0.821	-	0.900	0.010	$\mathbf{1e^{-4}}$
		Rot	0.004	-	0.007	$\mathbf{2e^{-5}}$	$7e^{-5}$
	obj	Transl	2.431	-	1.813	1.521	**0.980**
		Rot	0.271	-	0.159	0.159	**0.141**

RECT

Loose bounds			Initial	LEO	NMead	Powell	Ours
Noisy Ora	ee	Transl	1.027	1.381	0.912	0.009	$\mathbf{8e^{-4}}$
		Rot	0.006	0.015	0.006	$\mathbf{8e^{-5}}$	0.007
	obj	Transl	5.571	5.724	4.308	0.575	**0.396**
		Rot	0.471	0.197	0.464	0.007	**0.009**
Network	ee	Transl	0.729	2.887	0.862	0.041	**0.014**
		Rot	0.004	0.001	0.004	$2e^{-4}$	$\mathbf{6e^{-5}}$
	obj	Transl	4.300	4.935	6.464	2.505	**1.609**
		Rot	0.529	0.287	0.556	0.229	**0.212**
Tight bounds			Initial		NMead (C)	Powell (C)	Ours (C)
Noisy Ora	ee	Transl	1.027	-	1.643	**0.007**	0.019
		Rot	0.006	-	0.016	$3e^{-5}$	$\mathbf{1e^{-5}}$
	obj	Transl	5.571	-	2.993	0.484	**0.413**
		Rot	0.471	-	0.061	0.026	**0.013**
Network	ee	Transl	0.729	-	1.293	0.026	**0.014**
		Rot	0.004	-	0.012	$\mathbf{5e^{-5}}$	$1e^{-4}$
	obj	Transl	4.300	-	4.229	1.645	**1.609**
		Rot	0.529	-	0.240	0.214	**0.207**

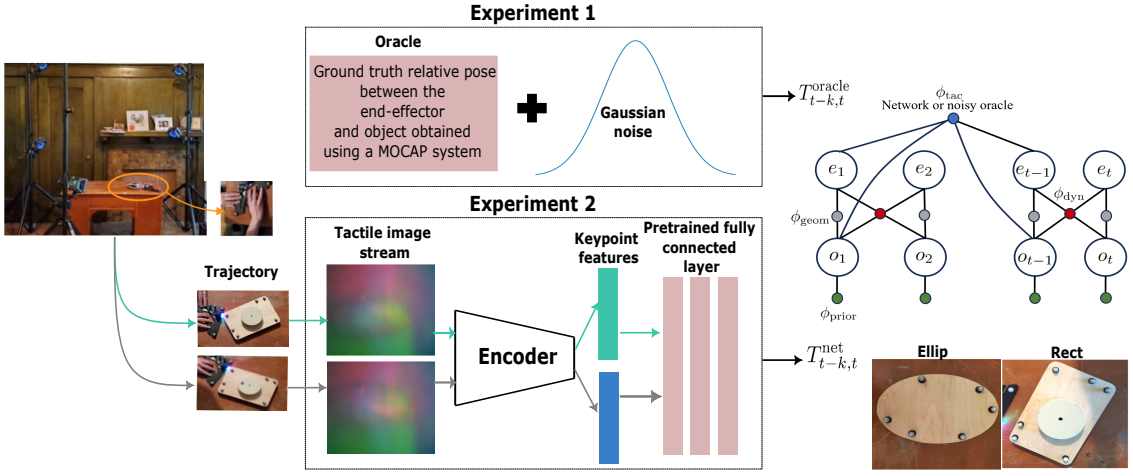

Fig. 2: Experimental setup and factor graph used for the planar pushing task. Figures showing the data collection setup were borrowed from [23, 30].

Here, we pre-train a fully connected network on a small training set to output the relative pose measurements from tactile input images. These measurements are designed to be relatively noisy and inaccurate. Both experiments are performed on 2 objects of different shapes: ellipsoidal and rectangular each featuring different contact patches. Finally, we initialize all $\theta_i \in \boldsymbol{\theta}^0$ such that 1) they are far from the underlying unknown latents and 2) the spread $\bar{s}$ is high. We use a 5/10 train/test split. The experimental setup is further illustrated in Fig. 2. Table II shows the RMSE of the output trajectories when compared against GT for the pushing task: LEO can converge to poor local minimas which lead at times to worse tracking error on the testing set compared to our initial estimate. Similarly, for Nelder-Mead, the method can fail to decrease the function value. In fact, it has practically been observed to stagnate at non-optimal points [43]. While Powell's method generates reasonable parameter estimates, our technique converges to solutions $\boldsymbol{\theta}^*$ that lead to better or comparable tracking accuracy on all unseen test trajectories.

TABLE III: The spread $\bar{s}$ (to the nearest integer) of the final output values $\boldsymbol{\theta}^*$ learned by our method.

	Ellip Ora	Ellip Net	Rect Ora	Rect Net
Ours (Loose bounds)	610	3615	460	420
Ours (Tight bounds)	70	62	67	75

Table III shows the eigenvalue value spread $\bar{s}$ of the optimized set $\boldsymbol{\theta}^*$ estimated by our method under both tight and loose eigenvalue bounds. Note that the generated solution indeed satisfies the hard bound constraints ($\bar{s}^{\max} < 100$) when specified. Conversely, in the absence of upper-bound constraints, the eigenvalue spread can effectively grow unbounded.

V. Commentary

A. Invariance to the Specified Constraint Bounds

We note from Tables I and II, that the performance of the Nelder-Mead and Powell's algorithms, in terms of tracking accuracy and variance of the output is influenced by the specified bound constraints. In contrast, our algorithm converges to minimas that lead to similar tracking performance (albeit with different eigenvalue spread) regardless of the specified bounds. Note that the parameter M in alg 2 requires tuning in order to damp the step size if the bound interval is large. In addition, and as typical in optimization problems, a solution needs to exist in the feasible region.

B. Runtime

Fig. 4 shows the translation and rotation accuracy on the training set as a function of runtime for the navigation dataset D_3. Nelder-Mead and Powell's method, being zero-order methods, exhibit slower convergence rates compared to gradient-based optimizers. LEO does leverage the gradient of the energy-based loss (Eq. 20). However, it requires samples from a high dimensional probability distribution to approximate the integral term at each training iteration which is a time-consuming process. Our technique generates gradients by directly comparing deviations from the training trajectories leading to faster convergence. However, note that our method's running time is expected to increase proportionally to the dimension of $\boldsymbol{\theta}$ since although parallelizable, the perturbations in Eq. 15 need to be performed per parameter and per dimension.

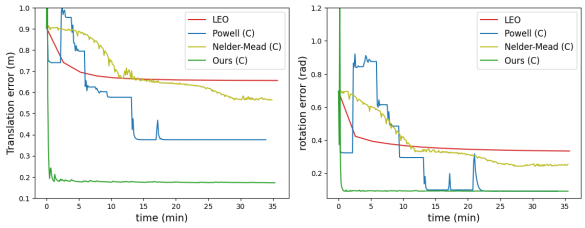

Fig. 4: Training translation and rotation error vs runtime for all methods on the navigation D_3 dataset.

C. Varying the Initialization

We show in Fig. 5 the training error curves for different initializations of the parameter vector $\boldsymbol{\theta}^0$ for dataset D_3. We observe that across all initializations, our method converges to model estimates that minimize the translation and rotation error (deviation from groundtruth pose) on the training set.

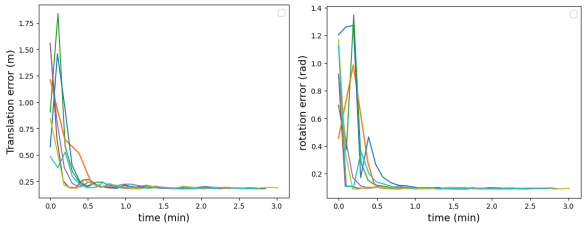

Fig. 5: Convergence with varying parameter initialization $\boldsymbol{\theta}^0$.

D. Varying the number of training trajectories

We used 5 training samples across experiments and found that increasing the number of training trajectories does not lead to improved generalization and testing accuracy. As noted in [44], given the relatively small parameter set $\boldsymbol{\theta}$ and the fact that both train and test trajectories are sampled from the same distribution, the learning process only requires a few samples.

VI. Conclusion and Future Work

We introduce a gradient-based algorithm to learn error covariance matrices for robotic state estimation. Our technique formulates the problem as a bilevel optimization procedure and generates required gradients through numerical differentiation. Our method results in parameters that generalize better compared to baselines with the added benefit of incorporating hard condition number constraints. In future work, we want to extend our algorithm to learn parameters $\{\theta_i\}$ that are themselves functions of observations i.e. $\theta_i(z_i, \Theta_i)$ where Θ_i can, for example, be the weights of a jointly trained neural network. Indeed, the outputs of the network can be perturbed to approximate $\frac{\partial \hat{\mathbf{x}}}{\partial \theta_i}$ (where $\hat{\mathbf{x}}$ is the solution returned by the graph optimizer) as proposed in this work and then chained with $\frac{\partial \theta_i}{\partial \Theta}$ (obtainable from existing auto-differentiation packages such as PyTorch [45]) to get the Jacobian of the optimized output trajectory with respect to network weights. Finally, enforcing constraints on the output of a neural network offers interesting related avenues for future research.

REFERENCES

[1] J. M. Tabeart, S. L. Dance, S. A. Haben, A. S. Lawless, N. K. Nichols, and J. A. Waller, "The conditioning of least-squares problems in variational data assimilation," *Numerical Linear Algebra with Applications*, vol. 25, no. 5, p. e2165, 2018.

[2] H. Hu and G. Kantor, "Parametric covariance prediction for heteroscedastic noise," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3052–3057.

[3] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[4] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.

[5] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[6] B. Yi, M. A. Lee, A. Kloss, R. Martín-Martín, and J. Bohg, "Differentiable factor graph optimization for learning smoothers," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1339–1345.

[7] J. Krishna Murthy, S. Saryazdi, G. Iyer, and L. Paull, "gradslam: Dense slam meets automatic differentiation," in *arXiv*, 2020.

[8] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[9] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[10] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[11] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation," in *2017 IEEE power & energy society general meeting*. IEEE, 2017, pp. 1–5.

[12] B. Feng, M. Fu, H. Ma, Y. Xia, and B. Wang, "Kalman filter with recursive covariance estimation—sequentially estimating process noise covariance," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6253–6263, 2014.

[13] A. Kloss, G. Martius, and J. Bohg, "How to train your differentiable filter," *Autonomous Robots*, vol. 45, no. 4, pp. 561–578, 2021.

[14] W. Sun, A. Venkatraman, B. Boots, and J. A. Bagnell, "Learning to filter with predictive state inference machines," in *International conference on machine learning*. PMLR, 2016, pp. 1197–1205.

[15] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," *Advances in neural information processing systems*, vol. 29, 2016.

[16] M. A. Lee, B. Yi, R. Martín-Martín, S. Savarese, and J. Bohg, "Multimodal sensor fusion with differentiable filters," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 444–10 451.

[17] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," *arXiv preprint arXiv:1805.11122*, 2018.

[18] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The bayes tree: An algorithmic foundation for probabilistic robot mapping," in *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2011, pp. 157–173.

[19] M. Qadri, P. Sodhi, J. G. Mangelson, F. Dellaert, and M. Kaess, "Incopt: Incremental constrained optimization using the bayes tree," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6381–6388.

[20] A. S. Lambert, M. Mukadam, B. Sundaralingam, N. Ratliff, B. Boots, and D. Fox, "Joint inference of kinematic and force trajectories with visuo-tactile sensing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3165–3171.

[21] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*. Springer, 2014, pp. 834–849.

[22] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[23] P. Sodhi, M. Kaess, M. Mukadam, and S. Anderson, "Learning tactile models for factor graph-based estimation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 686–13 692.

[24] A. Baikovitz, P. Sodhi, M. Dille, and M. Kaess, "Ground encoding: Learned factor graph-based models for localizing ground penetrating radar," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5476–5483.

[25] S. Bechtle, A. Molchanov, Y. Chebotar, E. Grefenstette, L. Righetti, G. Sukhatme, and F. Meier, "Meta learning via learned loss," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4161–4168.

[26] B. Amos and D. Yarats, "The differentiable cross-entropy method," in *International Conference on Machine Learning*. PMLR, 2020, pp. 291–302.

[27] D. J. Yoon, H. Zhang, M. Gridseth, H. Thomas, and T. D. Barfoot, "Unsupervised learning of lidar features for use ina probabilistic trajectory estimator," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2130–2138, 2021.

[28] J. N. Wong, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, "Variational inference with parameter learning applied to vehicle trajectory estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5291–5298, 2020.

[29] D. J. Yoon and T. D. Barfoot, "Towards consistent batch state estimation using a time-correlated measurement noise model," *arXiv preprint arXiv:2303.06507*, 2023.

[30] P. Sodhi, E. Dexheimer, M. Mukadam, S. Anderson, and M. Kaess, "Leo: Learning energy-based models in factor graph optimization," in *Conference on Robot Learning*. PMLR, 2022, pp. 234–244.

[31] E. C. Chi and K. Lange, "Stable estimation of a covariance matrix guided by nuclear norm penalties," *Computational statistics & data analysis*, vol. 80, pp. 117–128, 2014.

[32] J.-H. Won, J. Lim, S.-J. Kim, and B. Rajaratnam, "Condition-number-regularized covariance estimation," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 75, no. 3, pp. 427–450, 2013.

[33] J. M. Tabeart, S. L. Dance, A. S. Lawless, N. K. Nichols, and J. A. Waller, "Improving the condition number of estimated covariance matrices," *Tellus A: Dynamic Meteorology and Oceanography*, vol. 72, no. 1, pp. 1–19, 2020.

[34] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[35] A. L. Dontchev, R. T. Rockafellar, and R. T. Rockafellar, *Implicit functions and solution mappings: A view from variational analysis*. Springer, 2009, vol. 11.

[36] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[37] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[38] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson *et al.*, "Theseus: A library for differentiable nonlinear optimization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3801–3818, 2022.

[39] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[40] M. Frank, P. Wolfe *et al.*, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[41] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[42] G. Swamy, D. Wu, S. Choudhury, D. Bagnell, and S. Wu, "Inverse reinforcement learning without reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 33 299–33 318.

[43] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[44] M. Qadri and M. Kaess, "Learning observation models with incremental non-differentiable graph optimizers in the loop for robotics state estimation," in *ICML 2023 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, 2023.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.