Second-Order Sensitivity Analysis for Bilevel Optimization

Robert Dyro  
Stanford University  
rdyro@stanford.edu

Edward Schmerling  
Stanford University  
schmrlng@stanford.edu

Nikos Aréchiga  
Toyota Research Institute  
nikos.arechiga@tri.global

Marco Pavone  
Stanford University  
pavone@stanford.edu

Abstract

In this work we derive a second-order approach to bilevel optimization, a type of mathematical programming in which the solution to a parameterized optimization problem (the “lower” problem) is itself to be optimized (in the “upper” problem) as a function of the parameters. Many existing approaches to bilevel optimization employ first-order sensitivity analysis, based on the implicit function theorem (IFT), for the lower problem to derive a gradient of the lower problem solution with respect to its parameters; this IFT gradient is then used in a first-order optimization method for the upper problem. This paper extends this sensitivity analysis to provide second-order derivative information of the lower problem (which we call the IFT Hessian), enabling the usage of faster-converging second-order optimization methods at the upper level. Our analysis shows that (i) much of the computation already used to produce the IFT gradient can be reused for the IFT Hessian, (ii) errors bounds derived for the IFT gradient readily apply to the IFT Hessian, (iii) computing IFT Hessians can significantly reduce overall computation by extracting more information from each lower level solve. We corroborate our findings and demonstrate the broad range of applications of our method by applying it to problem instances of least squares hyperparameter auto-tuning, multi-class SVM auto-tuning, and inverse optimal control.

1 INTRODUCTION

Optimization is the foundation of modern learning and decision making systems, therefore a natural problem of interest is how to improve, learn, or optimize the optimization itself. Many practitioners of autonomous driving, robotics, and machine learning employ optimization on an everyday basis. Understanding how best to adjust this tool to more accurately suit their application needs is key to improving performance, trust, and reliability of these systems.

A natural way to approach improving optimization comes through formulating a bilevel program—users solve an optimization problem constructed with given data and parameters, and rely on a secondary metric quantifying the quality of the optimization result (it is of course optimal with respect to its own objective) to inform system design. Examples include (i) tuning regularization of a regression model to give good results on the test set, while training it to optimality on the train set, (ii) designing an autonomous car that drives in a human-like fashion, where it optimizes a finite horizon trajectory planning problem at every time step, (iii) setting parallel auction prices in such a way that rational bidding (an optimization in itself) leads to highest auction holder revenue. Any time optimization or decision making is applied, the question of selecting the right (hyper-)parameters arises in order to obtain, for example, (i) statistical models which generalize better, (ii) autonomous agents that behave more like an expert, (iii) auction systems that cannot be exploited. This notion is formalized as a bilevel
program in which the optimization-to-be-improved represents the lower level:

\[
\begin{align*}
\text{minimize}_{p} & \quad f_U(z^*, p) \\
\text{subject to} & \quad z^* = \text{argmin}_{z \in Z} f_L(z, p).
\end{align*}
\] (1)

This bilevel program formulation is general and subsumes the problems of test set model generalization, Stackelberg competition (Von Stackelberg, 2010), meta-learning (Finn et al., 2017) and few-shot learning (Lee et al., 2019). The quality of the optimization result, \(z^*\), of the objective \(f_L\) is quantified via the upper objective, \(f_U\). In the example of a statistical model, \(f_L\) represents the loss on the train dataset and \(f_U\) the loss on the test dataset. Solving the bilevel program requires selecting parameters \(p\) which produce such \(z^*\) that together lead to the minimal upper level loss \(f_U\).

This solution is often approximated by selecting parameters \(p\) by hand or via grid search. However, these approaches suffer from (a) being limited to cases where the dimension of \(p\) is low (usually below 4), (b) requiring parallel computing resources to keep re-evaluating \(z^*(p)\) and most critically, (c) the search is often done manually and wastes the expert’s or practitioner’s time.

A more principled way of solving Problem (1) is to use derivative information and make use of general-purpose solvers developed for optimization problems. This, however, requires the derivative of \(z^*\) with respect to \(p\)—quantifying how small changes in \(p\) affect the upper level objective \(f_U\) not just directly, but also by influencing \(z^*\). Although a closed-form expression of \(z^*\) with respect to \(p\) rarely exists, because \(z^*\) is the result of optimization, the dependence is implicitly defined via the necessary conditions for optimality of the lower optimization. For smooth-in-parameter problems, the implicit function theorem (IFT) may be employed to compute this derivative information relevant to solving the upper problem.

The availability of gradient expressions in bilevel programming obtained via sensitivity analysis enables the use of existing powerful optimizer to tackle these problems when they arise in real-world applications. However, even though many optimization algorithms use second-order information to converge fast in cases where the forward function evaluation is the bottleneck, much attention has been paid to extending sensitivity analysis to second-order information. Doing so would enable another class of faster optimization algorithms to be applied to bilevel programming.

The theoretical application of sensitivity analysis to bilevel programming relies on exact solutions to the lower level problems, but in reality, the numerical limitations rarely allow for that. Existing literature on first-order methods thus focuses on showing that the error in the derivative can be bounded and goes to zero as the approximation approaches the solution to the lower level problem.

### 1.1 Contributions

In this work we extend the application of the implicit function theorem, where gradients of inner optimization result with respect to parameters are found as in many existing works, (Gould et al., 2016; Agrawal et al., 2019b; Barratt, 2018), and derive second-order derivatives, i.e., the IFT Hessian. We leverage this result for three main contributions: (i) We show that the computational complexity of obtaining second-order derivatives is, in many cases, still dominated by the same matrix inversion bottleneck required for the IFT gradient and so our method can be implemented equally efficiently. (ii) We analyze our IFT Hessian expression to derive computational complexity and error bound expressions. We derive a new form of the regularized error bound under diagonal regularization of the matrix inverse operation in the application of IFT. (iii) We use our second-order derivative expression to apply second-order optimization methods to two machine learning datasets and show that these methods lead to faster bilevel optimization, requiring fewer lower level problem evaluations. We then further discuss the practical limitations and advantages of second-order optimization for bilevel optimization.


### 1.2 Related Work

**Practical Deployment.** Optimization improvement or optimization tuning has a long history in practical applications. For systems for which gradient derivation is non-trivial or more generally for systems where local gradient information is not informative of the global scope of the problem, gradient-free proxy models may be employed as in Golovin et al. (2017). Like hand-tuning or grid search, this approach constrains the number of parameters that can practically be tuned to single or low double digits.

\(^{1}\)When \(z^*\) itself has an interpretation as “parameters”, e.g., in model learning, \(p\) may be referred to as “hyperparameters”; we will refer to \(p\) as parameters throughout the remainder of this domain-agnostic work.

\(^{2}\)[pytorch.org](https://pytorch.org)

\(^{3}\)[github.com/google/jax](https://github.com/google/jax)
Formal Literature More formally the Bilevel Programming Problem (BLPP) formulation has a long history in literature (Bard, 1984; Von Stackelberg, 2010; Henrion and Surowiec, 2011; Liu et al., 2001; Bard, 2013). Sinha et al. (2017) contains an extensive review of approaches to solving BLPPs. Many of these works focus on theoretical analysis/characterization of BLPP approaches; the focus of this paper is more on specific concrete applications.

Applications-oriented Renewed interest in applications-oriented gradient-based solutions to BLPPs (Bengio, 2000), led to several works establishing the techniques for obtaining lower level solution gradients with respect to the parameters (Gould et al., 2016) and doing so efficiently for convex problems (Barratt, 2018; Agrawal et al., 2019b,a). Several computationally optimized, program-form specific approaches have been shown (Amos and Kolter, 2017; Amos et al., 2018). Most recently Lorraine et al. (2020); Blondel et al. (2021) apply the techniques to large-scale programs. Most of these applications-oriented works focus on deriving gradient expressions—to be used with a gradient-only BLPP optimizer. These works do not consider the loss landscape or the local curvature of the BLPP; in contrast, in this work, we attempt to quantify that.

Machine Learning BLPPs also found applications in meta-learning literature (Andrychowicz et al., 2016; Finn et al., 2017; Harrison et al., 2018; Bertinetto et al., 2018) with several works making explicit use of the implicit function theorem (Lee et al., 2019; Rajeswaran et al., 2019). While meta-learning literature poses an important application for BLPP, so far little attention has been given to improving the specifics of the solution methods employed in this body of work.

Higher-Order Derivatives Works most closely related to ours, with a focus on finding higher derivative information and analyzing the curvature of the BLPP are Wachsmuth (2014); Mehlibitz and Zemkoho (2021). These works do not demonstrate how to efficiently compute second order derivatives or demonstrate their usage in Newton’s-Method-like optimization, two key focuses of our work.

1.3 Notation

For an argument $x \in \mathbb{R}^d$ we denote the dimension of $x$ by dim$(x) = d$. For a scalar function $f : \mathbb{R}^{\text{dim}(x)} \rightarrow \mathbb{R}$ we denote its gradient and Hessian as $\nabla_x f(x)$ and $\nabla_x^2 f(x)$. For a vector function of $g : \mathbb{R}^{\text{dim}(x)} \rightarrow \mathbb{R}^{\text{dim}(g)}$ we denote its Jacobian matrix as $D_x g(x) \in \mathbb{R}^{\text{dim}(g) \times \text{dim}(x)}$. Where a function takes two arguments, we use the normal and mono-space font respectively to denote whether the differentiation operator is partial or total (i.e. does the derivative capture dependence between variables), e.g. for a function $g : \mathbb{R}^{\text{dim}(x)} \times \mathbb{R}^{\text{dim}(y)} \rightarrow \mathbb{R}^{\text{dim}(g)}$, $D_{x,y} g(x,y)$ denotes the partial Jacobian of $g$ w.r.t. $x$ and $D_x g(x)$ the total Jacobian of $g$ w.r.t. $x$ assuming the possible dependence of $x$ on $y$. In this work, we aim to describe higher order derivatives of vector functions, which leads us to define, for $g : \mathbb{R}^{\text{dim}(x)} \times \mathbb{R}^{\text{dim}(y)} \rightarrow \mathbb{R}^{\text{dim}(g)}$, $D_{x,y} g \in \mathbb{R}^{\text{dim}(g) \times \text{dim}(x) \times \text{dim}(y)}$, represented as a two dimensional matrix s.t.

$$D_{x,y} g(x,y) = \begin{bmatrix} D_y (\nabla_x (g(x,y)_1)) \\ \vdots \\ D_y (\nabla_x (g(x,y)_{\text{dim}(g)})) \end{bmatrix}$$

where $g(x,y)_i$ denotes the $i$-th scalar output of the vector function $g$. We define the operator $H_x \equiv D_{xx}$ (and the total version $H_x \equiv D_{xxx}$). We use $\otimes$ to denote the Kronecker product.

2 PROBLEM STATEMENT

Recall the formulation of bilevel programming:

$$\begin{align*}
\text{minimize} & \quad f_U(z^*, p) \\
\text{subject to} & \quad z^* = \text{argmin}_{z \in \mathbb{Z}} f_L(z, p)
\end{align*}$$

The problem considered in this work is how to obtain an explicit expression for the total second-order derivative, the Hessian matrix, of $f_U$ with respect to $p$, i.e., $H_p f_U(z^*, p)$. Access to this Hessian enables the application of second-order optimization methods for solving the upper problem, with the aim of reducing the total number of lower problem optimizations (and correspondingly, total overall computation).

3 METHOD

3.1 Preliminaries

Necessary Derivatives As overviewed in Section 1.2, there are several approaches to solving the bilevel problem (1). Here, we focus on derivation of the first and second derivatives of $f_U$ w.r.t. $p$: $D_p f_U$ and $H_p f_U$. Because $z^*$ depends on $p$, the total derivatives of $f_U$ can be written as

$$D_p f_U(z^*, p) = D_p f_U + (D_{z^*} f_U)(D_p z^*)$$

$$H_p f_U(z^*, p) = H_p f_U + (D_{p z^*})^T (H_{z^*} f_U)(D_{p z^*}) + ((D_{z^*} f_U) \otimes I) H_p z^*$$

Importantly, Equations (3) and (4) depend on (i) terms directly obtainable from the upper objective
function via analytical or automatic differentiation: $D_p f_U, H_p f_U, D_z f_U, H_z f_U$ and (ii) quantifying the sensitivity of lower level optimization, i.e., the result $z^*$ w.r.t. $p$: $D_p z^*, H_p z^*$.

The latter two terms are nominally obtainable through automatic differentiation by unrolling the entire $f_L$ optimization process— in many cases this is computationally undesirable or simply too memory intensive to be feasible. An alternative way of obtaining optimization sensitivity terms follows from the IFT.

**Implicit Function** The main insight which allows differentiating $z^* = \text{argmin}_z f_L(z,p)$ is the implicit condition imposed on $z^*$ as a result of $z^*$ being an optimal solution. An optimal solution must satisfy First Order Optimality Conditions (FOOC)

$$k(z^*, p) = 0$$

which define an implicit equation for $z^*$.\footnote{An implicit function is simply defined as a function $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ that equals 0 if its first input $x$ satisfies the implicit definition: $x \in \{x \mid k(x, y) = 0, \exists y \in \mathbb{R}^n\}$. This is in contrast to an explicit function of the form $f : \mathbb{R}^n \to \mathbb{R}^m$: $x \in \{x \mid f(y) = x\}$.}

For unconstrained optimization, the FOOC we consider are explicitly:

$$k(z^*, p) = D_z f_L(z^*, p) = 0.$$  

The following first-order sensitivity analysis is standard in the literature; we reproduce it here to motivate and enable our second-order analysis.

### 3.2 Implicit Function Theorem (IFT)

**Theorem 1 (Implicit Function Theorem (IFT)).** Let $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ be a continuously differentiable multivariate function of two variables $z^* \in \mathbb{R}^m$ and $p \in \mathbb{R}^n$ such that $k$ defines a fixed point for $z^*$, i.e., $k(z^*, p) = 0$ for all values of $p \in \mathbb{R}^n$. Then, the derivative of (implicitly defined) $z^*$ w.r.t. $p$ is given by

$$D_p z^* = -(D_z k(z^*, p))^{-1} D_p k(z^*, p).$$

**Proof.** Take $k(z^*, p) = 0$ and apply the chain rule to differentiate w.r.t. $p$, then, if $D_z k(z^*, p)$ is invertible

$$k(z^*, p) = 0$$

$$D_p k(z^*, p) + (D_z k(z^*, p))(D_p z^*) = 0$$

$$-(D_z k(z^*, p))^{-1} (D_p k(z^*, p)) = D_p z^*.$$  

(6)

Understanding the proof of the equation above is vital to obtaining higher derivatives via the IFT, since under suitable smoothness assumptions, Equation (6) can be differentiated again and the resulting expression solved for $H_p z^*$.

We now present the rarely derived IFT Hessian, which allows us to obtain the second derivative through an optimization. This result follows from a repeated application of the IFT to the same implicit function to obtain higher derivatives of $z^*$ w.r.t. $p$.

### 3.3 Second-Order IFT

**Theorem 2 (Second-Order IFT).** Let $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ be a twice continuously differentiable multivariate function of two variables $z^* \in \mathbb{R}^m$ and $p \in \mathbb{R}^n$ such that $k$ defines a fixed point for $x$, i.e. $k(z^*, p) = 0$ for all values of $p \in \mathbb{R}^n$. Then, the Hessian of (implicitly defined) $z^*$ w.r.t. $p$ is given by

$$H_p z^* = -(D_z k(z^*, p))^{-1} [H_p + (D_p z^*)(D_p z^*) \cdot (I \otimes (D_p z^*)^T)(D_z k) + (I \otimes (D_p z^*)^T)(H_z k)(D_p z^*)].$$

(7)

The proof of Theorem 2 is provided in the Appendix.

### 4 ANALYSIS

#### 4.1 Computational Complexity

The expression for the Second-Order IFT features Kronecker product terms (denoted by $\otimes$) which serve to broadcast over dimensions of either the embedding or the parameter dimension. Efficiently broadcasting over a particular dimension does not require constructing full dense matrices.

The two broadcasting Kronecker product-based forms that appear in Equation (4) are of the form $A \otimes I$ and $I \otimes B$ which both correspond to a broadcasted version of matrix multiplication. We refer the interested reader to the Appendix for the discussion of how computation with these forms can be accomplished efficiently.

Since our analysis here focuses on sensitivity analysis for optimization, we devote most of our attention to analyzing the computational complexity of computing the expression in Equation (4), which we recall here in full

$$H_p f_U(z^*, p) = H_p f_U + (D_p z^*)^T (H_z f_U)(D_p z^*)$$

$$+ (D_z f_U \otimes I) H_p z^*$$

where $H_p f_U, H_z f_U$ and $D_z f_U$ have explicit expressions and the term $D_p z^*$ is either already available
to us from first-order analysis or can be computed cheaply since the matrix $D_zk$ had to be factorized for first-order analysis. Thus, the only term requiring significant computation, substituting Equation (7), is

$$((D_zf_U) \otimes I) \left[ (D_zk)^{-1} \otimes I \right] \left[ \ldots \right] =$$

Equation (7)

$$\left[ (D_zf_U(D_zk)^{-1}) \otimes I \right] \left[ \ldots \right]$$

Since the upper level objective is a scalar by definition, the product

$$D_zf_U(D_zk)^{-1} = \begin{bmatrix} 1 \\ \vdots \end{bmatrix} \in \mathbb{R}^{1 \times m}$$

(7)

can be computed at the cost of a single matrix solve using an already necessarily factorized matrix from first-order analysis, which can be done computationally cheaply. Thus, evaluating the term $((D_zf_U) \otimes I)H_pz^*$ reduces to (a) caching a term from first-order analysis or can be computed efficiently making use of Jacobian-vector and vector-Jacobian products in an automatic differentiation engine.

4.1.1 Big-Ω Notation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Computational Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st IFT</td>
<td>$\mathcal{O}(m^3 + mn)$</td>
</tr>
<tr>
<td>1st IFT w/ sens.</td>
<td>$\mathcal{O}(m^3 + n)$</td>
</tr>
<tr>
<td>2nd IFT</td>
<td>$\mathcal{O}(m^3 + m^2n^2)$</td>
</tr>
<tr>
<td>2nd IFT w/ sens.</td>
<td>$\mathcal{O}(m^3 + mn^2)$</td>
</tr>
</tbody>
</table>

Table 1: Computational complexity of applying the first- and second-order implicit function theorem, omitting computation of partial explicit derivatives. In this table we define $m = \dim(z)$, $n = \dim(p)$, “w/ sens.” denotes that a sensitivity vector is available in which case the matrix inverse can be computed for a single left hand side as in Equation (7).

Following Section 4.1 we show computational complexity of applying the first- (1st) and second-order (2nd) implicit function theorem in Table 1 and observe that the second-order expression with a sensitivity left-hand side—the term $D_zf_U$ in Equations (3) (4)—has a computational complexity that differs from first-order expression with a sensitivity left hand side only by the additional quadratic term in $n = \dim(p)$, which is expected as the resulting matrix is in $\mathbb{R}^{n \times n}$.

The increase in computational complexity of obtaining the partial derivatives of $f_U$ and $k$, which we denote as partial explicit derivatives, primarily because: they are heavily problem dependent; they vanish; can be precomputed; efficient analytical expressions exist; or they can be obtained by efficiently making use of Jacobian-vector and Jacobian products in an automatic differentiation engine.

4.2 Error Analysis

Error analysis is vital for sensitivity analysis because numerical limitations often result in lower level solutions that are not quite optimal, but are instead at a small distance from the optimum. Developing confidence in the sensitivity methods described here requires the ability to bound the error caused by applying sensitivity analysis developed for optimal points to suboptimal lower level solutions.

4.2.1 First-Order Error Bound

Assuming an inexact local solution, we state the following error bound on the Jacobian. Theorem 3 is heavily inspired by Blondel et al. (2021), Theorem 1, which is in turn inspired by Higham (2002), Theorem 7.2.

Theorem 3 (First-Order Error Bound). Given the result of IFT applied to an exact solution $D_zp = g = -A^{-1}B$, where $A(z^*,p) = D_zk(z^*,p)$, $B(z^*,p) = D_pk(z^*,p)$ and an inexact solution $\tilde{A}(z,p) = D_zk(z,p)$ (where $k(z,p) \neq 0$) and $\tilde{B}(z,p) = D_pk(z,p)$. Assume $\|z - z^*\| \leq \delta$, $\|\tilde{A} - A\|_F \leq \gamma \delta$, $\|\tilde{B} - B\|_F \leq \beta \delta$, $\|B\|_F \leq R$, $\|\tilde{A}v\| \geq \alpha_1 \|v\|$, $\|Av\| \geq \alpha_2 \|v\|$, then

$$\|\tilde{J} - J\|_F \leq \frac{\beta}{\alpha_1} + \frac{\gamma R}{\alpha_1 \alpha_2} \delta$$

(8)

We point the reader to the Appendix for the proof.

4.2.2 Second-Order Error Bound

Assuming an inexact local solution, we state the following error bound on the Jacobian.

Theorem 4 (Second-Order Error Bound). Given the result of IFT applied to an exact solution, for $z^* \in \mathbb{R}^m$, $H_zp = H = -A^{-1}B$, where $A(z^*,p) = D_zk(z^*,p) \otimes I$, $B(z^*,p) = H_pk + (D_{pz^*}k)(D_{pz^*})^T(D_{pz^*}k) + (I \otimes (D_{pz^*})^T)(H_zk)(D_{pz^*})$ and an inexact solution
We point the reader to the Appendix for the proof.

4.2.3 Error Bound Optimization

Now, assuming an inexact solution we analyze the diagonal regularization of the inverse in the application of IFT. We let $\bar{A} = A + \epsilon I$.

**Theorem 5** (Regularized First-Order Error Bound). Given the result of regularized IFT applied to an exact solution $D_z \cdot p = g = -A^{-1}B$, where $A(z^*, p) = D_z \cdot k(z^*, p)$, $B(z^*, p) = D_p \cdot k(z, p)$ and an inexact solution $\bar{A}(z, p) = D_z \cdot k(z, p) + \epsilon I$, $\bar{B}(z, p) = D_p \cdot k(z, p)$ (where $k(z, p) \neq 0$). Assume $\|z - z^*\| \leq \delta$, $\|\bar{A} - A\|_{op} \leq \gamma \delta$, $\|\bar{B} - B\|_F \leq \beta \delta$, $\|B\|_F \leq R$, $\|\bar{A}v\| \geq \alpha_1 \|v\|$, and $v^T \bar{A}v \geq 0$, so $\|\bar{A}v\| \geq (\alpha_1 + \epsilon) \|v\|$, $\|Av\| \geq \alpha_2 \|v\|$. Then

$$
\|\bar{H} - H\|_F \leq \frac{\zeta + 2\mu \kappa + \nu \kappa^2 \delta}{\alpha_1} + \frac{\gamma R}{\alpha_1 \alpha_2} \delta. \tag{9}
$$

Selecting a post hoc arbitrary regularization allows to tighten the bound. We refer the reader to the Appendix for the proof.

5 EXPERIMENTS

5.1 Regression Model Auto-Tuning

We compare the performance of three commonly used nonlinear optimization algorithms on the problem of linear model improvement via smooth hyperparameter tuning. Using the above analysis we are able to apply a second-order optimization method, which offers to dramatically reduce the number of lower function evaluations in BLPP and significantly speed up optimization problems where the lower level constraint evaluation dominates. To compare, we optimize the hyperparameters of 3 linear models on MNIST (LeCun,
verges to a lower classification loss much faster. Adam exhibits poor performance on the convolutional SVM, outperforming L-BFGS in terms of test loss, but it takes BFGS goes unstable and the Adam optimizer tends to converge much quicker in terms of wall-clock time. Finally, the L-BFGS also reaches a higher test accuracy, but also a higher classification loss.

5.2 Hyperparameter Optimization

We further verify the usefulness of computing second-order derivatives in practical problem instances by applying the Newton’s Method to tuning a multi-class SVM model (Crammer and Singer, 2001) on the Fashion-MNIST (Xiao et al., 2017) dataset for best performance on the test set. We reformulate the constraints as log-barrier penalty\(^6\) to ensure smoothness, with the refinement value of \(\alpha = 10^2\). We solve the resulting problem with the Mosek optimizer.\(^7\)

We scale the images down to 14 × 14 to aid with the memory requirements and fit a multi-class SVM model to 2,000 randomly selected samples in the train set. We define upper level loss as the cross-entropy classification loss between the predictions and the labels. We further verify the usefulness of computing second-order derivatives derived in this paper leads to much faster convergence.

\(^6\)The log-barrier is defined for a constraint \(z \leq 0\) as \(-\log(-\alpha x)/\alpha\) for a tunable refinement constant \(\alpha\).

\(^7\)www.mosek.com

\(t_{i,j} = \delta_{i,y_i}\) indicates that \(t_{i,j}\) is equal to 1 if example \(j\) is of the class \(i\) and 0 otherwise.
5.3 Parameter Loss Landscape in Inverse Optimal Control

We investigate the Inverse Optimal Control (IOC) problem as a case study in loss landscape or curvature analysis. IOC has a natural formulation as a BLPP.

IOC has a large body of literature of its own, e.g. Keshavarz et al. (2011); Johnson et al. (2013); Terekhov and Zatsiorsky (2011), but we focus here on some simple examples that are illustrative to the general BLPP curvature analysis. We show that the unconstrained Optimal Control Problem (OCP) problem we formulate is globally convex in parameter, but that in the presence of constraints in the OCP, the resulting BLPP requires more care—linear inequalities, without reformulation, violate our smoothness assumption in Theorem 2.

Given known linear time-invariant discrete time dynamics \( x^{(i+1)} = Ax^{(i)} + Bu^{(i)} \) and state and control cost matrices \( Q, R \), and control limits \( u^{(i)} \in \mathbb{U} = \{ u \mid ||u||_\infty \leq u_{\text{lim}} \} \), we assume we observe some expert’s trajectory \( X_e = [x^{(0)}, \ldots, x^{(N)}] \) and control history \( U_e = [u^{(0)}, \ldots, u^{(N)}] \). We seek to learn a reference trajectory we assume the expert is tracking, i.e., the expert behaves optimally under the cost \( J(X,U) \):

\[
X_e, U_e = \arg\min_{X^{(i)}=Ax^{(i)}+Bu^{(i)}, U \in \mathbb{U}} J(X,U, X_e, U_e) \tag{11}
\]

where \( J(X,U, X_e, U_e) = \sum_i (x^{(i)} - x^{(i)}_{e,\text{ref}})^T Q (x^{(i)} - x^{(i)}_{e,\text{ref}}) + (u^{(i)} - u^{(i)}_{e,\text{ref}})^T R (u^{(i)} - u^{(i)}_{e,\text{ref}}) \). This leads to a BLPP

\[
\min_{X_{\text{ref}}, U_{\text{ref}}} \left\| X_e - X^* \right\|_2^2 + \| U_e - U^* \|
\]

s.t. \( X^*, U^* = \arg\min_{X^{(i)}=Ax^{(i)}+Bu^{(i)}, U \in \mathbb{U}} J(X,U, X_{\text{ref}}, U_{\text{ref}}) \).

(12)

In the typical notation of this paper, \( z^* = (X^*, U^*) \), \( p = (X_{\text{ref}}, U_{\text{ref}}) \). The problem corresponds to discovering the reference trajectory of an optimal agent, e.g., the centerline of a road using an observed trajectory of an autonomous car.

We show the comparison between the upper loss landscapes for IOC with unconstrained/constrained controls in Figure 5. We employ the Principle Component Analysis (PCA) dimension reduction technique on the optimization path to visualize a many dimensional upper loss in 2 dimensions, inspired by Li et al. (2017).

In the absence any constraints, the resulting problem as stated is convex which follows from the application of Equations (4) and (7).

Naive application of Equation (7) to a constrained IOC problem yields a globally positive definite Hessian, but violates the assumptions of Theorem 2, since FOOC with linear constraints are not twice continuously differentiable. In the presence of even the simple maximum control value constraints considered, the loss-in-parameter (the upper loss) is highly non-convex. We observe that any constraints can be smoothly approximated to any degree of precision using the log-barrier function which is infinitely differentiable. Figure 6 shows the 2D projected loss-in-parameter \( p \) landscape for a successive refinement of the log-barrier constraints; for high values of refinement \( \alpha \to \infty \), the loss landscape closely approximates the exactly constrained version, yet remains differentiable, so Theorem 2 can be applied.
6 DISCUSSION

Limitations & Promises The method we propose here offers to make better use of every single lower level problem evaluation, but comes with limitations. Firstly, like any second-order optimization method, it might not be best suited for highly non-convex landscapes, which can be common in BLPPs as Figure 5 shows. Secondly, in computational complexity analysis we do not focus on obtaining partial explicit derivatives necessary to compute the second-order sensitivity expression. We do so because these can be problem specific and often be zero or have an analytic form, but if they are obtained via automatic differentiation (the method we employ), their computation time highly depends on chosen software package. In general, it might turn out that computing second-order sensitivity information might take longer than several evaluations of the lower level and the first-order sensitivity information evaluation—at which point gradient-only optimization methods will likely function better than our proposed approach. Nevertheless, our work expands the optimization toolbox where some examples of BLPPs can be optimized much quicker, as Figure 4 shows.

7 CONCLUSIONS

In this work we derive the second-order derivatives of the upper level objective in a general bilevel program via sensitivity analysis using the implicit function theorem. Second-order information enables second-order optimization to be applied to these problems, which we argue can drastically reduce the number of lower level function evaluations, and speed up optimization, in cases where the lower level evaluation dominates. We further show that the computational complexity of our proposed approach is comparable to first-order IFT and we adapt error bound analysis for first-order IFT derivatives to our second-order IFT derivatives.

Future Work Future work includes quantifying how well various approximations suggested for first-order IFT apply to second-order IFT and whether (and when), for optimization purposes, some terms in our second-order sensitivity expression can be omitted or approximated to make their computation quicker. Finally, we are interested in further investigating loss landscapes, and the associated difficulty of optimization, for constrained bilevel problems.

Acknowledgments

Toyota Research Institute and the NASA University Leadership Initiative (grant #80NSSC20M0163) provided funds to support this work; this article solely reflects the opinions and conclusions of its authors and not any Toyota or NASA entity.

References

Second-Order Sensitivity Analysis for Bilevel Optimization


Higham, N. J. (2002). *Accuracy and stability of numerical algorithms*. SIAM.


Appendix

A PROOFS

Theorem 2 (Second-Order IFT). Let $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ be a twice continuously differentiable multivariate function of two variables $z^* \in \mathbb{R}^m$ and $p \in \mathbb{R}^n$ such that $k$ defines a fixed point for $x$, i.e. $k(z^*, p) = 0$ for all values of $p \in \mathbb{R}^n$. Then, the Hessian of (implicitly defined) $z^*$ w.r.t. $p$ is given by

$$H_{p z^*} = -\left( (D_{z^*} k)^{-1} \otimes I \right) \left[ H_{p k} + (D_{p z^*} k)(D_{p z^*}) ight] \left( I \otimes (D_{p z^*})^T \right) (D_{z^*} k)$$

$$+ (I \otimes (D_{p z^*})^T) (H_{z^*} k)(D_{p z^*}) \right].$$

(7)

Proof. Differentiation of the implicit expression further requires establishing a convention for matrix expression with vector inputs derivatives. We keep the standard convention of representing the partial Jacobian (with operator $D$) and scalar function Hessian (with operator $H$). The missing representations include the Hessian and the 2nd order mixed derivative. Let $f : \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}^p$, then we represent Hessian and 2nd order mixed derivatives as

$$D_{ab} f(a, b) = \begin{bmatrix} D_b (\nabla_a (f_1)) \\ \vdots \\ D_b (\nabla_a (f_p)) \end{bmatrix} \in \mathbb{R}^{pm \times n}$$

$$H_{a} f(a, b) = \begin{bmatrix} H_{a} f_1 \\ \vdots \\ H_{a} f_p \end{bmatrix} \in \mathbb{R}^{pm \times m}.$$  (13)

We introduce the following derivatives rules

$$D_{a} (A(a) B) = (I \otimes B^T)(D_{a} A(a))$$

$$D_{a} (BA(a)) = (B \otimes I)(D_{a} A(a))$$

(14)

(15)

where the chain rule applies as

$$D_{a} (A(c(a))) = (D_{c} A(c))(D_{a} c(a))$$

(16)

This allows to differentiate $H_{p k} + (D_{z^*} k)(D_{p z^*}) = 0$ giving

$$H_{p k}(z^*, p) + (D_{p z^*} k(z^*, p))(D_{p z^*})$$

$$+ (I \otimes (D_{p z^*})^T) (D_{z^*} k(z^*, p))$$

$$+ (I \otimes (D_{p z^*})^T) (H_{z^*} k(z^*, p))(D_{p z^*})$$

$$+ ((D_{z^*} k(z^*, p) \otimes I)(H_{p z^*})$$

$$= 0$$

(17)
The implicit Hessian is given by solving Equation (17) for $H_p z^*$.

$$H_p z^* = -\left( (D_z k(z^*,p))^{-1} \otimes I \right) \left[ H_p k(z^*,p) + (D_{p^2} k(z^*,p)) (D_p z^*) \right. 
+ \left. (I \otimes (D_p z^*)^T)(D_{2^p} k(z^*,p)) \right]$$

(7)

exploiting the identity $(A \otimes I)^{-1} = A^{-1} \otimes I$.

\[ \text{Theorem 3 (First-Order Error Bound).} \]

Given the result of IFT applied to an exact solution $D_z p = g = -A^{-1} B$, where $A(z^*,p) = D_z k(z^*,p)$, $B(z^*,p) = D_p k(z^*,p)$ and an inexact solution $\tilde{A}(z,p) = D_z k(z,p)$ (where $k(z,p) \neq 0$) and $\tilde{B}(z,p) = D_p k(z,p)$. Assume $\|z - z^*\| \leq \delta$, $\|\tilde{A} - A\|_{op} \leq \gamma \delta$, $\|\tilde{B} - B\|_F \leq \beta \delta$, $\|B\|_F \leq R$, $\|\tilde{A}v\| \geq \alpha_1 \|v\|$, $\|Av\| \geq \alpha_2 \|v\|$, then

$$\|\tilde{J} - J\|_F \leq \frac{\beta}{\alpha_1} \delta + \frac{\gamma R}{\alpha_1 \alpha_2} \delta$$

(18)

Proof.

$$-(\tilde{J} - J) = \tilde{A}^{-1} \tilde{B} - A^{-1} B$$

(19)

$$= \tilde{A}^{-1} \tilde{B} - \tilde{A}^{-1} B - A^{-1} B$$

(20)

$$= \tilde{A}^{-1} (\tilde{B} - B) + (\tilde{A}^{-1} - A^{-1}) B$$

(21)

which allows to bound the implicit gradient error as

$$\|\tilde{J} - J\|_F \leq \|\tilde{A}^{-1}\|_{op} \|\tilde{B} - B\|_F + \|\tilde{A}^{-1} - A^{-1}\|_{op} \|B\|_F$$

$$\leq \|\tilde{A}^{-1}\|_{op} \|\tilde{B} - B\|_F + \|\tilde{A}^{-1}\|_{op} \|\tilde{A} - A\|_{op} \|A^{-1}\|_{op} \|B\|_F$$

$$\leq \frac{\beta}{\alpha_1} \delta + \frac{\gamma R}{\alpha_1 \alpha_2} \delta$$

exploiting the fact that for any invertible matrices $M_1, M_2$ (where $\tilde{A}$, $A$ are invertible from $\|\tilde{A}v\| \geq \alpha_1 \|v\|$, $\|Av\| \geq \alpha_2 \|v\|$) $(M_1^{-1} - M_2^{-2}) = M_1^{-1} (M_1 - M_2) M_2^{-1}$.

\[ \text{Theorem 4 (Second-Order Error Bound).} \]

Given the result of IFT applied to an exact solution for $z^* \in \mathbb{R}^m$, $H_z p = H = -A^{-1} B$, where $A(z^*,p) = D_z k(z^*,p) \otimes I$, $B(z^*,p) = H_p k + (D_{p^2} k)(D_p z^*) + (I \otimes (D_p z^*)^T)(D_{2^p} k) k^T + (D_{p^2} z^*)^T)(D_{2^p} k) k^T$ and an inexact solution $\tilde{A}(z,p), \tilde{B}(z,p)$ (where $k(z,p) \neq 0$). Assume $\|z - z^*\| \leq \delta$, $\|\tilde{A} - A\|_{op} \leq \gamma \delta$, $\|H_p k(z,p) - H_p k(z^*,p)\|_F \leq \gamma \delta$, $\|D_{p^2} k(z,p) - D_{p^2} k(z^*,p)\|_F \leq \gamma \delta$, $\|H_z k(z,p) - H_z k(z^*,p)\|_F \leq \nu \delta$, $\|D_p p - D_z p\|_F \leq \kappa \delta$ defined in Theorem 3, $\|B\|_F \leq R_H$, $\|\tilde{A}v\| \geq \alpha_1 \|v\|$, $\|Av\| \geq \alpha_2 \|v\|$, then

$$\|\tilde{H} - H\|_F \leq \frac{\zeta + 2 \eta k \delta + \nu \kappa^2 \delta}{\alpha_1} + \frac{\gamma R H}{\alpha_1 \alpha_2} \delta.$$ 

(22)

Proof.

$$-(\tilde{H} - H) = \left( \tilde{A}^{-1} \otimes I \right) \tilde{B} - \left( A^{-1} \otimes I \right) B$$

(23)

$$= \left( \tilde{A}^{-1} \otimes I \right) \tilde{B} - \left( \tilde{A}^{-1} \otimes I \right) B + \left( \tilde{A}^{-1} \otimes I \right) B - \left( A^{-1} \otimes I \right) B$$

(24)

$$= \left( \tilde{A}^{-1} \otimes I \right) (\tilde{B} - B) + \left( \tilde{A}^{-1} - A^{-1} \right) \otimes I \right) B$$

(25)

(26)
which allows to bound the implicit Hessian error as

\[
\| \tilde{H} - H \|_F \leq \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{B} - B \|_F + \| \tilde{A}^{-1} - A^{-1} \|_{\text{op}} \| B \|
\]

\[
\leq \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{B} - B \|_F + \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{A} - A \|_{\text{op}} \| A^{-1} \|_{\text{op}} \| B \|
\]

\[
\leq \frac{\zeta + 2\eta\kappa \beta \delta}{\alpha_1} + \frac{\nu \kappa^2 \delta}{\alpha_1 \alpha_2} + \frac{\gamma R H}{\alpha_1 \alpha_2} \delta
\]

exploiting the fact that for any invertible matrices $M_1$, $M_2$ (where $\tilde{A}$, $A$ are invertible from $\| \tilde{A}v \| \geq \alpha_1 \| v \|$, $\| Av \| \geq \alpha_2 \| v \|$) $(M_1^{-1} - M_2^{-2}) = M_1^{-1} (M_1 - M_2) M_2^{-1}$.

**Theorem 5** (Regularized First-Order Error Bound). *Given the result of regularized IFT applied to an exact solution $D_z, p = g = -A^{-1}B$, where $A(z^*, p) = D_z, k(z^*, p), B(z^*, p) = D_p, k(z^*, p)$ and an inexact solution $\tilde{A}(z, p) = D_z, k(z, p) + \epsilon I$, $\tilde{B}(z, p) = D_p, k(z, p)$ (where $k(z, p) \neq 0$). Assume $\| z - z^* \| \leq \delta$, $\| \tilde{A} - A \|_{\text{op}} \leq \gamma \delta$, $\| \tilde{B} - B \|_F \leq \beta \delta, \| B \|_F \leq R$, $\| \tilde{A}v \| \geq \alpha_1 \| v \|$ and $v^T \tilde{A}v \geq 0$, so $\| \tilde{A}v \| \geq (\alpha_1 + \epsilon) \| v \|$, $\| Av \| \geq \alpha_2 \| v \|$ then

\[
\| \tilde{J} - J \|_F \leq \frac{\beta \delta}{\alpha_1 + \epsilon} + \frac{R (\gamma \delta + \epsilon)}{(\alpha_1 + \epsilon) \alpha_2}
\]

(27)

**Proof.** We observe that from singular value decomposition

\[
\| \tilde{A}v \| \geq \alpha_1 \| v \|, v^T \tilde{A}v \geq 0 \implies \| \tilde{A} + \epsilon I \|_{\text{op}} \geq \alpha_1 + \epsilon
\]

(28)

which gives

\[
\| \tilde{J} - J \|_F \leq \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{B} - B \|_F + \| \tilde{A}^{-1} - A^{-1} \|_{\text{op}} \| B \|
\]

\[
\leq \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{B} - B \|_F + \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{A} - A \|_{\text{op}} \| A^{-1} \|_{\text{op}} \| B \|
\]

\[
\leq \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{B} - B \|_F + \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{A} + \epsilon I - A \|_{\text{op}} \| A^{-1} \|_{\text{op}} \| B \|
\]

\[
\leq \| \tilde{A}^{-1} \|_{\text{op}} \| \tilde{B} - B \|_F + \| \tilde{A}^{-1} \|_{\text{op}} \left( \| \tilde{A} - A \|_{\text{op}} + \| \epsilon I \|_{\text{op}} \right) \| A^{-1} \|_{\text{op}} \| B \|
\]

\[
\leq \frac{\beta}{\alpha_1 + \epsilon} \delta + \frac{R}{(\alpha_1 + \epsilon) \alpha_2} (\gamma \delta + \epsilon)
\]

which gives

\[
\| \tilde{J} - J \|_F \leq \frac{\beta \delta}{\alpha_1 + \epsilon} + \frac{R (\gamma \delta + \epsilon)}{(\alpha_1 + \epsilon) \alpha_2}
\]

(29)
B  COMPUTATIONAL COMPLEXITY

Evaluating $A \otimes I$ Efficiently  The product $A \otimes I$ features in the Hessian expression

$$H_{pz^*} = \left[ (D_{z^*}k)^{-1} \otimes I \right] \ldots$$

Given the expression $M = (A \otimes I)C$, let $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{(np) \times r}$, such that

$$C = \begin{bmatrix} C_1 \in \mathbb{R}^{p \times r} \\ \vdots \\ C_n \in \mathbb{R}^{p \times r} \end{bmatrix}$$

Let $M = (A \otimes I)C$ and define $\tilde{C} \in \mathbb{R}^{n \times p \times r}$ s.t. $\tilde{C}_{i*} \in \mathbb{R}^{p \times r} \forall i \in [1..n]$, $\tilde{M} \in \mathbb{R}^{m \times p \times r}$ s.t. $\tilde{M}_{j*} \in \mathbb{R}^{p \times r} \forall j \in [1..m]$—a 3-d representation of $C$ and $M$ where the stacked matrices in $C$, $M$ are concatenated along a third, first, dimension in $\tilde{C}$, $\tilde{M}$.

The operation can now be defined formally in Einstein summation notation as

$$(\tilde{M})_{ijk} = (A)_{jl}(\tilde{C})_{ilk}$$

Intuitively, the operation $M = (A \otimes I)$ corresponds to a matrix multiplication performed for every vector in $C$ built from $n$ elements, one from each $C_i$.

Evaluating $I \otimes B$ Efficiently  The product $I \otimes B$ features in the Hessian expression

$$H_{pz^*} = \ldots \left[ \cdots + (I \otimes (D_{p}z^*)^T)(D_{z^*}k) + \ldots \right]$$

Given the expression $M = (I \otimes B)C$, let $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{(np) \times r}$, such that

$$C = \begin{bmatrix} C_1 \in \mathbb{R}^{n \times r} \\ \vdots \\ C_p \in \mathbb{R}^{n \times r} \end{bmatrix}$$

Let $M = (I \otimes B)C$ and define $\tilde{C} \in \mathbb{R}^{p \times n \times r}$ s.t. $\tilde{C}_{i*} \in \mathbb{R}^{n \times r} \forall i \in [1..n]$, $\tilde{M} \in \mathbb{R}^{m \times p \times r}$ s.t. $\tilde{M}_{j*} \in \mathbb{R}^{p \times r} \forall j \in [1..p]$—a 3-d representation of $C$ and $M$ where the stacked matrices in $C$, $M$ are concatenated along a third, first, dimension in $\tilde{C}$, $\tilde{M}$.

The operation can now be defined formally in Einstein summation notation as

$$(\tilde{M})_{ijk} = (A)_{jl}(\tilde{C})_{ilk}$$

Intuitively, the operation $M = (I \otimes B)$ corresponds to a matrix multiplication performed by broadcasting $B$ and performing matrix multiplication of $B$ with every $p$ element of $C$, so that $BC_i \forall i \in [1..p]$—these products are then stacked together to form $M$. 
C ADDITIONAL INSIGHTS

Convexity-in-parameter Global convexity-in-parameter is obtained if the second-order order derivative of the upper loss function w.r.t. to the parameter is globally positive semi-definite (PSD) while both the objective functions are globally twice continuously differentiable. Recalling the full derivative expressions

\[
H_f(z^*, p) = H_p f_U + (D_p z^*)^T (H_{z^*} f_U) (D_p z^*) + \left( (D^{*} f_U) \otimes I \right) H_p z^*
\]

we conclude that it is not possible to guarantee the global PSD property for this Hessian in general, because of the reduction \( (D^{*} f_U) \otimes I \) as outlined in Section B.\(^8\) However, in the special case when \( H_p z^* = 0 \) and the problem is globally twice differentiable, global convexity is obtained, e.g., unconstrained quadratic programs with linear parameterizations.

---

\(^8\) A stack of Hessian matrices are reduced by a weighted summation with unknown sign weights.