Gauss–Newton-type methods for bilevel optimization

Jörg Fliege^{1,2} · Andrey Tin^{1,2} · Alain Zemkoho^{1,2}

Received: 6 March 2020 / Accepted: 3 December 2020 / Published online: 10 January 2021
© The Author(s) 2021

Abstract

This article studies Gauss–Newton-type methods for over-determined systems to find solutions to bilevel programming problems. To proceed, we use the lower-level value function reformulation of bilevel programs and consider necessary optimality conditions under appropriate assumptions. First, under strict complementarity for upper- and lower-level feasibility constraints, we prove the convergence of a Gauss–Newton-type method in computing points satisfying these optimality conditions under additional tractable qualification conditions. Potential approaches to address the shortcomings of the method are then proposed, leading to alternatives such as the pseudo or smoothing Gauss–Newton-type methods for bilevel optimization. Our numerical experiments conducted on 124 examples from the recently released Bilevel Optimization LIBrary (BOLIB) compare the performance of our method under different scenarios and show that it is a tractable approach to solve bilevel optimization problems with continuous variables.

Keywords Bilevel optimization · Value function reformulation · Partial exact penalization · Gauss-Newton method

1 Introduction

We aim to solve the bilevel programming problem

$$\min_{x,y} F(x,y) \text{ s.t. } G(x,y) \leq 0, y \in S(x) := \arg \min_y \{f(x,y) : g(x,y) \leq 0\}, \quad (1.1)$$

✉ Alain Zemkoho
a.b.zemkoho@soton.ac.uk

Jörg Fliege
j.fliege@soton.ac.uk

Andrey Tin
a.tin@soton.ac.uk

¹ Centre for Operational Research, Management Sciences and Information Systems (CORMSIS), Southampton SO17 1BJ, UK

² School of Mathematical Sciences, University of Southampton, Southampton SO17 1BJ, UK

where $F : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, $G : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^q$, and $g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$. As usual, we refer to F (resp. f) as the upper-level (resp. lower-level) objective function and G (resp. g) as the upper-level (resp. lower-level) constraint function. Solving problem (1.1) is very difficult because of the implicit nature of the lower-level optimal solution mapping $S : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ defined in (1.1).

There are several ways to deal with the complex nature of problem (1.1). Earliest techniques were based the implicit function and Karush-Kuhn-Tucker (KKT) reformulations. The implicit function approaches rely on the insertion of the lower-level solution function or its approximation in the upper-level objective function; see, e.g., [4, 8] for related algorithms. As for the KKT reformulation, it is strongly linked to MPECs (mathematical programs with equilibrium constraints), see, e.g., [6], which are not necessarily easy to handle, in part due to the extra variables representing the lower-level Lagrangian multipliers. Interested readers are referred to [1, 9, 19] and references therein, for results and methods based on this transformation. More details on methods based on the KKT reformulation and other approaches to deal with bilevel optimization can be found in the books [2, 5]. In this paper, we are going to use the lower-level value function reformulation (LLVF)

$$\min_{x,y} F(x,y) \text{ s.t. } G(x,y) \leq 0, \quad g(x,y) \leq 0, \quad f(x,y) \leq \varphi(x), \quad (1.2)$$

where the optimal value function is defined by

$$\varphi(x) := \inf \{ f(x,y) \mid g(x,y) \leq 0 \}, \quad (1.3)$$

to transform problem (1.1) into a single-level optimization problem. As illustrated in [13], this approach can provide tractable opportunities to develop second order algorithms for the bilevel optimization problem, as it does not involve first order derivatives for the lower-level problem, as in the context of the KKT reformulation.

There are recent studies on solution methods for bilevel programs, based on the LLVF reformulation. For example, [22, 23, 27, 30, 36] develop global optimization techniques for (1.1) based on (1.2)–(1.3). [25, 37, 38] propose algorithms computing stationary points for (1.2)–(1.3), in the case where the upper-level and lower-level feasible sets do not depend on the lower-level and upper-level variable, respectively. [13] is the first paper to propose a Newton-type method for the LLVF reformulation for bilevel programs. Numerical results there show that the approach can be very successful. The system of equations built there is quadratic while our method in this paper is based on a non-quadratic overdetermined system of equations. Hence, there is a need to develop Gauss–Newton-type techniques to capture certain classes of bilevel optimization stationarity points.

One of the main problems in solving (1.2) is that its feasible set systematically fails many constraint qualifications (see, e.g., [10]). To deal with this issue, we will use the partial calmness condition [39], to shift the value function constraint $f(x,y) \leq \varphi(x)$ to the upper-level objective function, as a penalty term with penalty parameter λ . The other major problem with the LLVF reformulation is that φ is typically non-differentiable. This will be handled by using upper estimates of the subdifferential of the function; see, e.g., [7, 9, 10, 39]. Our Gauss–Newton-type

scheme proposed in this paper is based on a relatively simple system of optimality conditions that depend on λ .

To transform these optimality conditions into a system of equations, we substitute the corresponding complementarity conditions using the standard Fischer–Burmeister function [12]. To deal with the non-differentiability of the Fischer–Burmeister function, we consider two approaches in this paper. The first one is to assume strict complementarity for the constraints involved in the upper- and lower-level feasible sets. As second option to avoid non-differentiability; here we investigate a smoothing technique by adding a perturbation inside the Fischer–Burmeister function.

Another important aspect of the aforementioned system of equations is that it is *overdetermined*. Since overdetermined systems have non-square Jacobian, we cannot use a classical Newton-type method as in [13]; for a Newton method addressing the closely related semi-infinite programming problem, interested readers are referred to [32, 33]. Gauss–Newton and Newton-type methods with Moore–Penrose pseudo inverse are both introduced in Sect. 3. It will be shown that these methods are well-defined for solving bilevel programs from the perspective of the LLVF reformulation (1.2). In particular, our framework ensuring that the Gauss–Newton method for bilevel optimization is well-defined does not require any assumption on the lower-level objective function. Links between the two methods are then discussed and it is shown that they should perform very similarly for most problems. However, it is expected that the Newton-type method with pseudo inverse will be more robust, as the evidence from the numerical experiments (Sect. 5) confirms.

In Sect. 5, we present results of extensive experiments on the methods and comparisons with the MATLAB built-in function *fsolve*. Considering the complicated nature of the feasible set of problem (1.1), the results from the algorithms are compared with known solutions of the problems to check if obtained stationary points are optimal solutions of the problems or not. For 124 tested problems (taken from the BOLIB library [41]), more than 80% are solved satisfactorily in the sense that we recover known solutions with $<10\%$ error, or obtain better ones by all methods with CPU time being less than half a second. The number of recovered solutions as well as the performance profiles and feasibility check show that Gauss–Newton and Newton method with pseudo inverse outperform *fsolve*. It is worth mentioning here that it is not typical to conduct such a large number of experiments in the bilevel optimization literature. The conjecture of the similarity of the performance of the tested methods is verified numerically, also showing that Newton’s method with pseudo inverse is indeed more robust than the classic Gauss–Newton one. However, the technique for choosing the penalty parameter λ is still a heuristic, and the result might depend on the structure of the corresponding problem.

2 Optimality conditions and equation reformulation

Let us start with some definitions required to state the main theorem of this section. Define *full convexity* of the lower-level problem as convexity of the lower-level objective function, as well as all lower-level constraints, with respect to (x, y) .

Further on, a point $(\bar{x}, \bar{y}) \in \mathbb{R}^n \times \mathbb{R}^m$, feasible for lower-level problem, is said to be *lower-level regular* if there exists a direction $d \in \mathbb{R}^m$ such that

$$\nabla_y g_i(\bar{x}, \bar{y})^T d < 0 \text{ for } i \in I_g(\bar{x}, \bar{y}) := \{i : g_i(\bar{x}, \bar{y}) = 0\}. \quad (2.1)$$

It is clear that (2.1) corresponds to the Mangasarian-Fromowitz constraint qualification for the lower-level constraint at the point $\bar{y}$, when the upper-level variable is fixed at $x := \bar{x}$. Similarly, the point $(\bar{x}, \bar{y}) \in \mathbb{R}^n \times \mathbb{R}^m$ satisfying the upper- and lower-level inequality constraints is *upper-level regular* if there exists a direction $d \in \mathbb{R}^{n+m}$ such that

$$\begin{aligned} \nabla G_j(\bar{x}, \bar{y})^T d &< 0 \quad \text{for } j \in I_G(\bar{x}, \bar{y}) := \{j : G_j(\bar{x}, \bar{y}) = 0\}, \\ \nabla g_j(\bar{x}, \bar{y})^T d &< 0 \quad \text{for } j \in I_g(\bar{x}, \bar{y}) := \{j : g_j(\bar{x}, \bar{y}) = 0\}. \end{aligned} \quad (2.2)$$

Finally, to describe necessary optimality conditions for problem (1.2), it is standard to use the following partial calmness concept [39]:

Definition 2.1 Let $(\bar{x}, \bar{y})$ be a local optimal solution of problem (1.2). This problem is partially calm at $(\bar{x}, \bar{y})$ if there exists $\lambda > 0$ and a neighbourhood U of $(\bar{x}, \bar{y}, 0)$ such that

$$\begin{aligned} F(x, y) - F(\bar{x}, \bar{y}) + \lambda|u| &\geq 0, \quad \forall (x, y, u) \in U : G(x, y) \leq 0, g(x, y) \leq 0, \\ f(x, y) - \varphi(x) - u &= 0. \end{aligned}$$

According to [39, Proposition 3.3], problem (1.2)–(1.3) being partially calm at one of its local optimal solution $(\bar{x}, \bar{y})$ is equivalent to the existence of a parameter $\lambda > 0$ such that the point $(\bar{x}, \bar{y})$ is also a local optimal solution of problem

$$\min_{x, y} F(x, y) + \lambda(f(x, y) - \varphi(x)) \quad \text{s.t. } G(x, y) \leq 0, g(x, y) \leq 0. \quad (2.3)$$

Partial calmness has been the main tool to derive optimality conditions for (1.1) from the perspective of the optimal value function; see, e.g., [7, 9, 10, 39]. It is automatically satisfied if the upper-level feasible set is independent from y and the lower-level problem is defined by

$$f(x, y) := c^\top y \quad \text{and} \quad g(x, y) := A(x) + By,$$

where $A : \mathbb{R}^n \rightarrow \mathbb{R}^p$, $c \in \mathbb{R}^m$, and $B \in \mathbb{R}^{p \times m}$. More generally, various sufficient conditions ensuring that partial calmness holds have been studied in the literature; see [39] for the seminal work on the subject. More recently, the paper [26] has revisited the condition, proposed a fresh perspective, and established new dual-type sufficient conditions for partial calmness to hold.

It is clear that problem (2.3) is a penalization of (1.2) only w.r.t. the constraint $f(x, y) - \varphi(x) \leq 0$ with penalty parameter λ . Hence, the problem is a usually labelled as a *partial exact penalization* of problem (1.2)–(1.3). With this reformulation it is now reasonable to assume standard constraint qualifications to

derive optimality conditions. Based on this, we have the following result, see, e.g., [7, 9, 10, 39], based on a particular estimate of the subdifferential of φ (1.3).

Theorem 2.2 *Let $(\bar{x}, \bar{y})$ be a local optimal solution to (1.2)–(1.3), where all functions are assumed to be differentiable, φ is finite around $\bar{x}$ and the lower-level problem is fully convex. Further assume that the problem is partially calm at $(\bar{x}, \bar{y})$, the lower-level regularity is satisfied at $(\bar{x}, \bar{y})$ and upper-level regularity holds at $\bar{x}$. Then there exist $\lambda > 0$, and Lagrange multipliers u , v , and w such that*

$$\nabla F(\bar{x}, \bar{y}) + \nabla g(\bar{x}, \bar{y})^T (u - \lambda w) + \nabla G(\bar{x}, \bar{y})^T v = 0, \quad (2.4)$$

$$\nabla_{\mathcal{Y}} f(\bar{x}, \bar{y}) + \nabla_{\mathcal{Y}} g(\bar{x}, \bar{y})^T w = 0, \quad (2.5)$$

$$u \geq 0, \quad g(\bar{x}, \bar{y}) \leq 0, \quad u^T g(\bar{x}, \bar{y}) = 0, \quad (2.6)$$

$$v \geq 0, \quad G(\bar{x}, \bar{y}) \leq 0, \quad v^T G(\bar{x}, \bar{y}) = 0, \quad (2.7)$$

$$w \geq 0, \quad g(\bar{x}, \bar{y}) \leq 0, \quad w^T g(\bar{x}, \bar{y}) = 0. \quad (2.8)$$

Remark 2.3 There are important classes of functions that satisfy the full convexity assumption imposed on the lower-level problem in Theorem 2.2; cf. [24]. However, when it is not possible to guarantee that this assumption is satisfied, there are at least two alternative scenarios to obtain the same optimality conditions. The first is to replace the full convexity assumption by the *inner semicontinuity* of the optimal solution set-valued mapping S (1.1). Secondly, note that a much weaker qualification condition known as *inner semicompactness* can also be used here. However, under the latter assumption, it will additionally be required to have $S(\bar{x}) = \{\bar{y}\}$ in order to arrive at the optimality conditions (2.4)–(2.8). The concept of inner semicontinuity (resp. semicompactness) of S is closely related to the lower semicontinuity (resp. upper semicontinuity) of set-valued mappings; for more details on these notions and their ramifications on bilevel programs, see [7, 9, 10].

Depending on the assumptions made, we can obtain optimality conditions different from the above. The details of different stationarity concepts can be found in the references provided in the remark above, as well as in [40]. Weaker assumptions will typically lead to more general conditions. However, making stronger assumptions allows us to obtain systems that are easier to handle. For instance, it is harder to deal with more general conditions introduced in Theorem 3.5 of [10] or Theorem 3.1 of [7] because of the presence of the convex hull in the corresponding estimate of the subdifferential of φ [7, 9, 10, 39]. The other advantage of (2.4)–(2.8) is that, unlike the system studied in [13], these conditions do not require to introduce a new lower-level variable.

The above optimality conditions involve the presence of complementarity conditions (2.6)–(2.8), which result from inequality constraints present in (1.2)–(1.3). In order to reformulate the complementarity conditions in the form of a

system of equations, we are going to use the concept of NCP-functions; see, e.g., [34]. The function $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}$ is said to be a NCP-function if we have

$$\phi(a, b) = 0 \iff a \geq 0, \quad b \geq 0, \quad ab = 0.$$

In this paper, we use $\phi(a, b) := \sqrt{a^2 + b^2} - a - b$, known as the *Fischer–Burmeister* function [12]. This leads to the reformulation of the optimality conditions (2.4)–(2.8) into the system of equations:

$$\Upsilon^\lambda(z) := \begin{pmatrix} \nabla_x F(x, y) + \nabla_x g(x, y)^T (u - \lambda w) + \nabla_x G(x, y)^T v \\ \nabla_y F(x, y) + \nabla_y g(x, y)^T (u - \lambda w) + \nabla_y G(x, y)^T v \\ \nabla_x f(x, y) + \nabla_y g(x, y)^T w \\ \sqrt{u^2 + g(x, y)^2} - u + g(x, y) \\ \sqrt{v^2 + G(x, y)^2} - v + G(x, y) \\ \sqrt{w^2 + g(x, y)^2} - w + g(x, y) \end{pmatrix} = 0, \quad (2.9)$$

where we have $z := (x, y, u, v, w)$ and

$$\sqrt{u^2 + g(x, y)^2} - u + g(x, y) := \begin{pmatrix} \sqrt{u_1^2 + g_1(x, y)^2} - u_1 + g_1(x, y) \\ \vdots \\ \sqrt{u_p^2 + g_p(x, y)^2} - u_p + g_p(x, y) \end{pmatrix}. \quad (2.10)$$

$\sqrt{v^2 + G(x, y)^2} - v + G(x, y)$ and $\sqrt{w^2 + g(x, y)^2} - w + g(x, y)$ are defined as in (2.10). The superscript λ is used to emphasize the fact that this number is a parameter and not a variable for equation (2.9). One can easily check that this system consists of $n + 2m + p + q + p$ real-valued equations with $n + m + p + q + p$ variables. Clearly, this means that (2.9) is an over-determined system and the Jacobian of $\Upsilon^\lambda(z)$, where it exists, is a non-square matrix.

3 Gauss–Newton-type methods under strict complementarity

To solve equation (2.9), we use a Gauss–Newton-type method, as the system is over-determined. Hence, it is necessary to compute the Jacobian of $\Upsilon^\lambda(z)$ (2.9). However, the function is not differentiable at any point where one of the pairs

$$(u_i, g_i(x, y)), \quad i = 1, \dots, p, \quad (v_j, G_j(x, y)), \quad j = 1, \dots, q, \quad \text{and} \quad (w_i, g_i(x, y)), \quad i = 1, \dots, p$$

vanishes. To avoid this situation, we assume throughout this section that the strict complementarity condition holds:

Assumption 3.1 The strict complementarity condition holds at (x, y, u, v, w) if $(u_i, g_i(x, y)) \neq 0$ and $(w_i, g_i(x, y)) \neq 0$ for all $i = 1, \dots, p$ and $(v_j, G_j(x, y)) \neq 0$ for all $j = 1, \dots, q$.

Under this assumption, the Jacobian of Υ^λ is well-defined everywhere and hence, the *Gauss–Newton step* to solve equation (2.9) can be defined as

$$d^k = -(\nabla \Upsilon^\lambda(z^k)^T \nabla \Upsilon^\lambda(z^k))^{-1} \nabla \Upsilon^\lambda(z^k)^T \Upsilon^\lambda(z^k), \quad (3.1)$$

provided that the involved inverse matrix exists; see, e.g., [15, 28]. This leads to the following algorithm tailored to equation (2.9):

Algorithm 3.2 Gauss–Newton Method for Bilevel Optimization

Step 0: Choose $\lambda > 0$, $\epsilon > 0$, $K > 0$, $z^0 := (x^0, y^0, u^0, v^0, w^0)$, and set $k := 0$.

Step 1: If $\|\Upsilon^\lambda(z^k)\| < \epsilon$ or $k \geq K$, then stop.

Step 2: Calculate Jacobian $\nabla \Upsilon^\lambda(z^k)$ and compute the direction d^k using (3.1).

Step 3: Set $z^{k+1} := z^k + d^k$, $k := k + 1$, and go to Step 1.

In Algorithm 3.2, ϵ denotes the tolerance and K is the maximum number of iterations. Additionally, it is possible to implement a line search technique to control the step size at each iteration, as done in [13] for a Newton-type method. The impact of line search will be analyzed in a separate work, as our main aim here is to study the core step (3.1) of the method. It is clear from (3.1) that for Algorithm 3.2 to be well-defined, the matrix $\nabla \Upsilon^\lambda(z)^T \nabla \Upsilon^\lambda(z)$ needs to be non-singular. In the next subsection, we provide tractable conditions ensuring that this is possible.

3.1 Nonsingularity of $\nabla \Upsilon^\lambda(z)^T \nabla \Upsilon^\lambda(z)$ and Convergence

We begin this subsection by noting that as the Jacobian $\nabla \Upsilon^\lambda(z)$ is a $(n + 2m + 2p + q) \times (n + m + 2p + q)$ matrix with m more rows than columns, and linear independence of its columns ensures that $\nabla \Upsilon^\lambda(z)^T \nabla \Upsilon^\lambda(z)$ is non-singular. It therefore suffices for us to provide conditions guaranteeing the linear independence of the columns of $\nabla \Upsilon^\lambda(z)$.

To present the Jacobian of the system (2.9) in a compact form, let the *upper-level* and *lower-level Lagrangian* functions be defined by

$$L^\lambda(z) := F(x, y) + g(x, y)^T (u - \lambda w) + G(x, y)^T v \quad \text{and} \quad \mathcal{L}(z) := f(x, y) + g(x, y)^T w,$$

respectively. As we need the derivatives of these functions in the sequel, we denote the appropriate versions of the Hessian matrices of L^λ and $\mathcal{L}$, w.r.t. (x, y) , by

$$\nabla^2 L^\lambda(z) := \begin{bmatrix} \nabla_{xx}^2 L^\lambda(z) & \nabla_{yx}^2 L^\lambda(z) \\ \nabla_{xy}^2 L^\lambda(z) & \nabla_{yy}^2 L^\lambda(z) \end{bmatrix} \quad \text{and} \quad \nabla(\nabla_y \mathcal{L}(z)) := \begin{bmatrix} \nabla_{xy}^2 \mathcal{L}(z) & \nabla_{yy}^2 \mathcal{L}(z) \end{bmatrix} \quad (3.2)$$

respectively. Furthermore, by denoting $\nabla g(x, y)^T := \begin{bmatrix} \nabla_x g(x, y)^T \\ \nabla_y g(x, y)^T \end{bmatrix}$ and $\nabla G(x, y)^T := \begin{bmatrix} \nabla_x G(x, y)^T \\ \nabla_y G(x, y)^T \end{bmatrix}$, we can easily check that the Jacobian of $\Upsilon^\lambda(z)$ w.r.t. z can be written as

$$\nabla \Upsilon^\lambda(z) = \begin{bmatrix} \nabla^2 L^\lambda(z) & \nabla g(x, y)^T & \nabla G(x, y)^T & -\lambda \nabla g(x, y)^T \\ \nabla(\nabla_y \mathcal{L}(z)) & O & O & \nabla_y g(x, y)^T \\ \mathcal{T} \nabla g(x, y) & \Gamma & O & O \\ \mathcal{A} \nabla G(x, y) & O & \mathcal{B} & O \\ \Theta \nabla g(x, y) & O & O & \mathcal{K} \end{bmatrix} \quad (3.3)$$

with $\mathcal{T} := \text{diag}\{\tau_1, \dots, \tau_p\}$, $\Gamma := \text{diag}\{\gamma_1, \dots, \gamma_p\}$, $\mathcal{A} := \text{diag}\{\alpha_1, \dots, \alpha_q\}$, $\mathcal{B} := \text{diag}\{\beta_1, \dots, \beta_q\}$, $\Theta := \text{diag}\{\theta_1, \dots, \theta_p\}$, and $\mathcal{K} := \text{diag}\{\kappa_1, \dots, \kappa_p\}$, where the pair (τ_j, γ_j) , $j := 1, \dots, p$ is defined by

$$\tau_j := \frac{g_j(x, y)}{\sqrt{u_j^2 + g_j(x, y)^2}} + 1 \quad \text{and} \quad \gamma_j := \frac{u_j}{\sqrt{u_j^2 + g_j(x, y)^2}} - 1, \quad \text{for } j = 1, \dots, p. \quad (3.4)$$

The pairs (α_j, β_j) , $j = 1, \dots, q$ and (θ_j, κ_j) , $j = 1, \dots, p$ are defined similarly in terms of $(G_j(x, y), v_j)$, $j = 1, \dots, q$ and $(g_j(x, y), w_j)$, $j = 1, \dots, p$, respectively.

Additionally, analogously to the lower-level (resp. upper-level) regularity condition in (2.1) (resp. (2.2)), we will need the lower-level (resp. upper-level) linear independence constraint qualification denoted by LLICQ (resp. ULICQ) that will be said to hold at a point $(\bar{x}, \bar{y})$ if the family of gradients

$$\{\nabla_y g_i(\bar{x}, \bar{y}), i \in I_g(\bar{x}, \bar{y})\} \quad (\text{resp. } \{\nabla g_i(\bar{x}, \bar{y}), i \in I_g(\bar{x}, \bar{y}), \nabla G_j(\bar{x}, \bar{y}), j \in I_G(\bar{x}, \bar{y})\}) \quad (3.5)$$

is linearly independent. Finally, to construct the second order condition necessary in the formulation of the next result, we also need the following index sets:

$$\begin{aligned} v^1 &:= v^1(\bar{x}, \bar{y}, \bar{u}) &:= \{j \mid \bar{u}_j > 0, g_j(\bar{x}, \bar{y}) = 0\}, \\ v^2 &:= v^2(\bar{x}, \bar{y}, \bar{v}) &:= \{j \mid \bar{v}_j > 0, G_j(\bar{x}, \bar{y}) = 0\}, \\ v^3 &:= v^3(\bar{x}, \bar{y}, \bar{w}) &:= \{j \mid \bar{w}_j > 0, g_j(\bar{x}, \bar{y}) = 0\}. \end{aligned}$$

We use these sets to introduce the following linear space:

$$\mathcal{C}(\bar{x}, \bar{y}) := \left\{ d \left| \begin{array}{ll} \nabla g_j(\bar{x}, \bar{y})^\top d = 0 & \text{for } j \in v^1 \\ \nabla G_j(\bar{x}, \bar{y})^\top d = 0 & \text{for } j \in v^2 \\ \nabla g_j(\bar{x}, \bar{y})^\top d = 0 & \text{for } j \in v^3 \end{array} \right. \right\}.$$

Theorem 3.3 *Let the point $\bar{z} = (\bar{x}, \bar{y}, \bar{u}, \bar{v}, \bar{w})$ satisfy (2.9) for some $\lambda > 0$. Suppose that Assumption 3.1 holds at $(\bar{x}, \bar{y}, \bar{u}, \bar{v}, \bar{w})$, while LLICQ and ULICQ are satisfied at $(\bar{x}, \bar{y})$. Furthermore, suppose that we have*

$$d^\top \nabla^2 L^\lambda(\bar{z}) d > 0 \quad \text{for all } d \in \mathcal{C}(\bar{x}, \bar{y}) \setminus \{0\}. \quad (3.6)$$

Then, the columns of the Jacobian matrix $\nabla Y^\lambda(\bar{z})$ are linearly independent.

Proof Consider an arbitrary vector $d := (d_1^\top, d_2^\top, d_3^\top, d_4^\top)^\top$ such that $\nabla Y^\lambda(\bar{z})d = 0$ with the components $d_1 \in \mathbb{R}^{n+m}$, $d_2 \in \mathbb{R}^p$, $d_3 \in \mathbb{R}^q$, and $d_4 \in \mathbb{R}^p$. Then we have

$$\nabla^2 L^\lambda(\bar{z})d_1 + \nabla g(\bar{x}, \bar{y})^\top d_2 + \nabla G(\bar{x}, \bar{y})^\top d_3 - \lambda \nabla g(\bar{x}, \bar{y})^\top d_4 = 0, \quad (3.7)$$

$$\mathcal{T} \nabla g(\bar{x}, \bar{y})d_1 + \Gamma d_2 = 0, \quad (3.8)$$

$$\mathcal{A} \nabla G(\bar{x}, \bar{y})d_1 + \mathcal{B} d_3 = 0, \quad (3.9)$$

$$\Theta \nabla g(\bar{x}, \bar{y})d_1 + \mathcal{K} d_4 = 0, \quad (3.10)$$

$$\nabla(\nabla_y \mathcal{L}(\bar{z}))d_1 + \nabla_y g(\bar{x}, \bar{y})^\top d_4 = 0. \quad (3.11)$$

On the other hand, it obviously follows from (3.4) that

$$(\tau_j - 1)^2 + (\gamma_j + 1)^2 = 1 \quad \text{for } j = 1, \dots, p. \quad (3.12)$$

Hence, the indices of the pair (τ, γ) satisfying (3.12) can be partitioned into the sets

$$P_1 := \{j : \tau_j > 0, \gamma_j < 0\}, \quad P_2 := \{j : \tau_j = 0\}, \quad \text{and} \quad P_3 := \{j : \gamma_j = 0\}.$$

Similarly, define index sets Q_1, Q_2 , and Q_3 for the pair (α, β) and T_1, T_2 , and T_3 for (θ, κ) . Next, consider the following componentwise description of (3.8), (3.9), and (3.10),

$$\tau_j \nabla g_j(\bar{x}, \bar{y})^\top d_1 + \gamma_j d_{2j} = 0 \quad \text{for } j = 1, \dots, p, \quad (3.13)$$

$$\alpha_j \nabla G_j(\bar{x}, \bar{y})^\top d_1 + \beta_j d_{3j} = 0 \quad \text{for } j = 1, \dots, q, \quad (3.14)$$

$$\theta_j \nabla g_j(\bar{x}, \bar{y})^\top d_1 + \kappa_j d_{4j} = 0 \quad \text{for } j = 1, \dots, p. \quad (3.15)$$

For $j \in P_2$, equation (3.13) becomes $\gamma_j d_{2j} = 0$. Additionally, it follows from (3.12) that for $j \in P_2$, $\gamma_j \neq 0$. Hence $d_{2j} = 0$ for $j \in P_2$. For $j \in P_3$, (3.13) leads to $\tau_j \nabla g_j(\bar{x}, \bar{y})^\top d_1 = 0$, which due to the property above translates into $\nabla g_j(\bar{x}, \bar{y})^\top d_1 = 0$, as $\tau_j \neq 0$. Finally, for $j \in P_1$ equation (3.13) takes the form

$\nabla g_j(\bar{x}, \bar{y})^T d_1 = -\frac{\gamma_j}{\tau_j} d_{2_j}$, where by definition of P_1 we know that $-\frac{\gamma_j}{\tau_j} > 0$. Following the same logic, we respectively have from (3.14) and (3.15) that

$$\begin{aligned} d_{3_j} &= 0 \text{ for } j \in Q_2, \quad \nabla G_j(\bar{x}, \bar{y})^T d_1 = 0 \text{ for } j \in Q_3, \quad \nabla G_j(\bar{x}, \bar{y})^T d_1 = -\frac{\beta_j}{\alpha_j} d_{3_j} \text{ for } j \in Q_1, \\ d_{4_j} &= 0 \text{ for } j \in T_2, \quad \nabla g_j(\bar{x}, \bar{y})^T d_1 = 0 \text{ for } j \in T_3, \quad \nabla g_j(\bar{x}, \bar{y})^T d_1 = -\frac{\kappa_j}{\theta_j} d_{4_j} \text{ for } j \in T_1, \end{aligned}$$

with $-\frac{\beta_j}{\alpha_j} > 0$ for $j \in Q_1$ and $-\frac{\kappa_j}{\theta_j} > 0$ for $j \in T_1$. Multiplying (3.7) by d_1^T ,

$$d_1^T \nabla^2 L^\lambda(\bar{z}) d_1 + d_1^T \nabla g(\bar{x}, \bar{y})^T d_2 + d_1^T \nabla G(\bar{x}, \bar{y})^T d_3 - \lambda d_1^T \nabla g(\bar{x}, \bar{y})^T d_4 = 0. \quad (3.16)$$

Considering the cases defined above, we know that for $j \in P_2$, $j \in Q_2$, and $j \in T_2$, the terms d_{2_j} , d_{3_j} , and d_{4_j} , respectively, disappear. For $j \in P_3$, $j \in Q_3$, and $j \in T_3$, the terms $\nabla g_j(\bar{x}, \bar{y})^T d_1$, $\nabla G_j(\bar{x}, \bar{y})^T d_1$, and $\nabla g_j(\bar{x}, \bar{y})^T d_1$ also vanish. This leads to the equation (3.16) being simplified to

$$d_1^T \nabla^2 L^\lambda(\bar{z}) d_1 + \sum_{j \in P_1} \left(-\frac{\gamma_j}{\tau_j} \right) d_{2_j}^2 + \sum_{j \in Q_1} \left(-\frac{\beta_j}{\alpha_j} \right) d_{3_j}^2 - \lambda \sum_{j \in T_1} \left(-\frac{\kappa_j}{\theta_j} \right) d_{4_j}^2 = 0. \quad (3.17)$$

One can easily check that thanks to Assumption 3.1, the sets P_1 , Q_1 , and T_1 are empty. Hence, (3.17) reduces to $d_1^T \nabla^2 L^\lambda(\bar{z}) d_1 = 0$. Moreover, one can easily check that $v^1 \subseteq P_3$, $v^2 \subseteq Q_3$, and $v^3 \subseteq T_3$. Hence, $d_1 \in \mathcal{C}(\bar{x}, \bar{y})$ and therefore, it follows from (3.6) that $d_1 = 0$.

We have shown that $d_{2_j} = 0$, $d_{3_j} = 0$ and $d_{4_j} = 0$ for $j \in P_2$, $j \in Q_2$ and $j \in T_2$, and $d_1 = 0$. Let us use these results to simplify equations (3.7) and (3.11) as follows

$$\sum_{j \in P_3} \nabla g_j(\bar{x}, \bar{y})^T d_{2_j} + \sum_{j \in Q_3} \nabla G_j(\bar{x}, \bar{y})^T d_{3_j} - \lambda \sum_{j \in T_3} \nabla g_j(\bar{x}, \bar{y})^T d_{4_j} = 0, \quad (3.18)$$

$$\sum_{j \in T_3} \nabla_y g_j(\bar{x}, \bar{y})^T d_{4_j} = 0. \quad (3.19)$$

Equation (3.19) implies that $d_{4_j} = 0$ for all $j \in T_3$, given that $T_3 \subseteq I_g(\bar{x}, \bar{y})$ and the LLICQ holds at $(\bar{x}, \bar{y})$. Then (3.18) becomes

$$\sum_{j \in P_3} \nabla g_j(\bar{x}, \bar{y})^T d_{2_j} + \sum_{j \in Q_3} \nabla G_j(\bar{x}, \bar{y})^T d_{3_j} = 0,$$

which implies $d_{2_j} = 0$ and $d_{3_j} = 0$ for $j \in P_3$ and $j \in Q_3$, given that the ULICQ holds at $(\bar{x}, \bar{y})$. This completes the proof as we have shown that $\nabla Y^\lambda(\bar{z})d = 0$ only if $d = 0$.

Example 3.4 We consider an instance of problem (1.1) taken from the BOLIB Library [41] with

$$\begin{aligned} F(x, y) &:= (x-3)^2 + (y-2)^2, & G(x, y) &:= \begin{pmatrix} x-8 \\ -x \end{pmatrix}, & g(x, y) &:= \begin{pmatrix} -2x+y-1 \\ x-2y-2 \\ x+2y-14 \end{pmatrix}. \\ f(x, y) &:= (y-5)^2, \end{aligned}$$

The point $\bar{z} = (\bar{x}, \bar{y}, \bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{v}_1, \bar{v}_2, \bar{w}_1, \bar{w}_2, \bar{w}_3) = (1, 3, 4\lambda - 2, 0, 0, 0, 0, 4, 0, 0)$ satisfies equation (2.9) for any $\lambda > 1/2$. Obviously, strict complementarity holds at this point, for $\lambda > 1/2$. and the family of vectors $\{\nabla g_j(\bar{x}, \bar{y}), j \in I_g(x, y), \nabla G_j(\bar{x}, \bar{y}), j \in I_G(x, y)\}$ is linearly independent, as $I_g(x, y) = \{1\}$, $I_G(x, y) = \emptyset$. It is easy to see that ULICQ holds as $\nabla g_1(\bar{x}, \bar{y})^T = (-2, 1)^T \neq 0$, and LLICQ holds as $\nabla_y g_1(\bar{x}, \bar{y})^T = 1 \neq 0$. Finally, we obviously have that $\nabla^2 L^\lambda(\bar{z}) = 2e$, where e is the identity matrix of $\mathbb{R}^{2 \times 2}$, is positive definite. In conclusion, this example shows that all assumptions of Theorem 3.3 can hold for a bilevel program and therefore the Gauss–Newton method in () is well-defined.

Based on the result above, we can now state the convergence theorem for our Gauss–Newton Algorithm 3.2. To proceed, first note that by implementing the Gauss–Newton method to solve (2.9), leads to a solution to the least-square problem

$$\min_z \Phi^\lambda(z) := \sum_{i=1}^{N+m} \Upsilon_i^\lambda(z)^2, \quad (3.20)$$

where we define $N := n + m + 2p + q$. The direction of the Newton method for problem (3.20) can be written as

$$d^k := -(\nabla \Upsilon^\lambda(z_k))^T \nabla \Upsilon^\lambda(z_k) + T(z_k))^{-1} \nabla \Upsilon^\lambda(z_k) \Upsilon(z_k),$$

where $T(z_k) := \sum_{i=1}^N \Upsilon_i^\lambda(z_k) \nabla^2 \Upsilon_i^\lambda(z_k)$ is the term that is omitted in the Gauss–Newton direction (3.1). It is well known that the Gauss–Newton method converges with the same rate as Newton method if the term $T(\bar{z})$ is small enough in comparison with the term $\nabla \Upsilon^\lambda(\bar{z})^T \nabla \Upsilon^\lambda(\bar{z})$; see, e.g., [11, 28, 35]. This is the basis of the following convergence result of Algorithm 3.2.

Theorem 3.5 *Let the assumptions in Theorem 3.3 hold at the point $\bar{z} = (\bar{x}, \bar{y}, \bar{u}, \bar{v}, \bar{w})$ (for some $\lambda > 0$), assumed to be a local optimal solution of problem (3.20). Also, let $\{z^k\}$ be a sequence of points generated by Algorithm 3.2 and assume that it converges to $\bar{z}$. Furthermore, suppose that $\nabla^2 L^\lambda$ and $\nabla(\nabla_y) \mathcal{L}$ are well-defined and Lipschitz continuous in a neighbourhood of $\bar{z}$. Then we have*

$$\|z^{k+1} - \bar{z}\| \leq \left\| (\nabla \Upsilon^\lambda(\bar{z}))^T \nabla \Upsilon^\lambda(\bar{z}) \right\|^{-1} \left\| T(\bar{z}) \right\| \|z^k - \bar{z}\| + O\left(\|z_k - \bar{z}\|^2\right). \quad (3.21)$$

Proof Start by recalling that under the assumptions of Theorem 3.3, the matrix $\nabla \Upsilon^\lambda(\bar{z})$ is of full column rank. Hence, it is positive definite. Furthermore, under the strict complementarity condition and well-definedness of the functions $\nabla^2 L^\lambda$ and $\nabla(\nabla_y) \mathcal{L}$ near $\bar{z}$, all components of $z \mapsto \Upsilon^\lambda(z)$ and $z \mapsto \nabla \Upsilon^\lambda(z)$ are differentiable near

$\bar{z}$. Hence, the term $T(z)$ is well-defined near $\bar{z}$. Furthermore, the local Lipschitz continuity of $\nabla^2 L^\lambda$ and $\nabla(\nabla_y \mathcal{L})$ imply that the same holds for $z \mapsto \nabla Y^\lambda(z)^T \nabla Y^\lambda(z)$ and $z \mapsto T(z)$. Hence, the function $z \mapsto \nabla Y^\lambda(z)^T \nabla Y^\lambda(z) + T(z)$ is Lipschitz continuous around $\bar{z}$. Next, note that $z \mapsto \nabla Y^\lambda(z)^T \nabla Y^\lambda(z)$ is differentiable near $\bar{z}$, as the same is satisfied for $z \mapsto \nabla Y^\lambda(z)$. We also know that $\nabla Y^\lambda(z)^T \nabla Y^\lambda(z)$ is non-singular for all z in some neighbourhood of $\bar{z}$, under the assumptions made in Theorem 3.3. By the inverse function theorem, the latter ensures that $(\nabla Y^\lambda)^T \nabla Y^\lambda$ is a diffeomorphism, and hence has a differentiable inverse around $\bar{z}$. Thus, the function $z \mapsto \left(\nabla Y^\lambda(z)^T \nabla Y^\lambda(z) \right)^{-1}$ is Lipschitz continuous around the point $\bar{z}$ and hence, inclusion (3.21) follows by the application of [35, Theorem 7.2.2].

It is clear from this theorem that the Gauss–Newton method converges quadratically if $T(\bar{z}) = 0$ and Q-linearly if $T(\bar{z})$ is small relative to $\nabla Y^\lambda(\bar{z})^T \nabla Y^\lambda(\bar{z})$. Such properties can be satisfied for small residuals problems and for the problems that are not too nonlinear. For problems with small residuals we have that the components $Y_i^\lambda(\bar{z})$ are small for all i , which makes the term $T(\bar{z})$ small. For the problems with not too much nonlinearity the components $\nabla^2 Y_i^\lambda(\bar{z})$ are small for all i , which also results in small $T(\bar{z})$. If it turns out that we can obtain an exact solution $Y^\lambda(\bar{z}) = 0$, then $T(\bar{z}) = 0$, and we have quadratic convergence. It is worth noting that in general we cannot always have $Y_i^\lambda(\bar{z}) = 0$ for all i as the system is overdetermined, but minimizing the sum of the squares of $Y_i^\lambda(\bar{z})$ we obtain a solution point $\bar{z}$, at which $\sum_{i=1}^{N+m} (Y_i^\lambda(\bar{z}))^2$ is as small as possible. If the problem has small residuals, then small $Y_i^\lambda(\bar{z})$ are naturally obtained by implementing Algorithm 3.2 as this is designed to minimize $\sum_{i=1}^{N+m} Y_i^\lambda(z)^2$. In terms of having small components $\nabla^2 Y_i^\lambda(\bar{z})$ we observe that $\nabla^2 Y^\lambda(\bar{z})$ will involve third derivatives of $F(x, y)$, $G(x, y)$, $f(x, y)$ and $g(x, y)$. Hence, if the original problem is not too nonlinear, the Hessian of the system (2.9) should be small. As a result, if there exists a solution with $Y_i^\lambda(z) \approx 0$ for all i or if the original problem (1.1) is not too nonlinear, then we expect that the Gauss–Newton for Bilevel Programming converges Q-linearly.

The first drawback of Algorithm 3.2 is the requirement of strict complementarity in Assumption 3.1, to help ensure the differentiability of the function Y^λ . Assumption 3.1 is rather strong; for the test problems used for our numerical experiments in Sect. 5, it did not hold at the last iteration for at least one value of λ for a total of 54 out of 124 problems considered. If one wants to avoid the *strict complementarity* assumption, one option is to use *smoothing technique* for Fischer–Burmeister function, to be discussed in Sect. 4. Before we move to this, it is worth mentioning that a second issue faced by our Algorithm 3.2 is the requirement that the matrix $\nabla Y^\lambda(z)^T \nabla Y^\lambda(z)$ is nonsingular at each iteration. To deal with this, one option is to execute a Newton step, where the generalized inverse of $\nabla Y^\lambda(z)^T \nabla Y^\lambda(z)$, which always exists, is used. Such an approach is briefly discussed in the next subsection.

3.2 Newton method with Moore–Penrose pseudo inverse

Indeed, one of the most challenging aspects of the Gauss–Newton step in Algorithm 3.2 is the computation of the inverse of the matrix $\nabla Y^\lambda(z_k)^T \nabla Y^\lambda(z_k)$, as this quantity might not exist at some iterations. To deal with situations where the inverse of the matrix does not exist, various concepts of generalized inverse have been used in the context of Newton’s method; see, e.g., [29] for related details. Although we do not directly compute $(\nabla Y^\lambda(z_k)^T \nabla Y^\lambda(z_k))^{-1}$ in our implementation of Algorithm 3.2 in Sect. 5, we would like to compare the *pure* Gauss–Newton-type method presented in the previous subsection with the Newton method using the Moore–Penrose pseudo inverse. Hence, we present the later approach here and its relationship to Algorithm 3.2.

For an arbitrary matrix $A \in \mathbb{R}^{m \times n}$, its *Moore–Penrose pseudo inverse* (see, e.g., [18]) is defined by

$$A^+ := V \Sigma^+ U^T,$$

where $V \Sigma^+ U^T$ represents a singular value decomposition of A , where Σ^+ corresponds to the pseudo-inverse of Σ that can be given by

$$\Sigma^+ = \text{diag} \left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \dots, \frac{1}{\sigma_r}, 0, \dots, 0 \right) \quad \text{with } r = \text{rank}(A).$$

If A has full column rank, we have an additional property that

$$A^+ := (A^T A)^{-1} A^T.$$

Based on this definition, an iteration of the Newton method with pseudo inverse for system (2.9) can be then stated as

$$z^{k+1} = z^k - \nabla Y(z^k)^+ Y(z^k). \quad (3.22)$$

We are now going to refer to (3.22) as iteration of the *Pseudo-Newton method*. The Pseudo-Newton method for bilevel programming can be defined in the same fashion as Algorithm 3.2 with the difference that direction would be given by $d^k = -\nabla Y^\lambda(z^k)^+ Y^\lambda(z^k)$. Clearly, the Pseudo-Newton method is always well-defined, unlike the Gauss–Newton method, and hence, it will produce some result in the case when the Gauss–Newton method diverges [14]. Based on this general behaviour and interplay between the two approaches, we will be comparing them in the numerical section. For details on the convergence of Newton-type methods with pseudo-inverse, the interested reader is referred to [17].

4 Smoothing Gauss–Newton method

In this section, we relax the strict complementarity assumption, considering the fact that it often fails for many problems as illustrated in the previous section. However, to ensure the smoothness of the function Υ^λ (2.9), the Fischer–Burmeister function is replaced with the *smoothing Fischer–Burmeister function* (see [20]) defined by

$$\phi_{g_j}^\mu(x, y, u) := \sqrt{u_j^2 + g_j(x, y)^2} + 2\mu - u_j + g_j(x, y) \quad \text{for } j = 1, \dots, p, \quad (4.1)$$

where the perturbation parameter $\mu > 0$ helps to guarantee its differentiability at points (x, y, u) satisfying $u_j = g_j(x, y) = 0$. It is well-known (see latter references) that

$$\phi_{g_j}^\mu(x, y, u) = 0 \iff [u_j > 0, -g_j(x, y) > 0, -u_j g_j(x, y) = \mu]. \quad (4.2)$$

The smoothing system of optimality conditions becomes

$$\Upsilon_\mu^\lambda(z) := \begin{pmatrix} \nabla_x F(x, y) + \nabla_x g(x, y)^T(u - \lambda w) + \nabla_x G(x, y)^T v \\ \nabla_y F(x, y) + \nabla_y g(x, y)^T(u - \lambda w) + \nabla_y G(x, y)^T v \\ \nabla_x f(x, y) + \nabla_y g(x, y)^T w \\ \sqrt{u^2 + g(x, y)^2} + 2\mu - u + g(x, y) \\ \sqrt{v^2 + G(x, y)^2} + 2\mu - v + G(x, y) \\ \sqrt{w^2 + g(x, y)^2} + 2\mu - w + g(x, y) \end{pmatrix} = 0, \quad (4.3)$$

following the convention in (2.10), where μ is a vector of appropriate dimensions with sufficiently small positive elements. Under the assumption that all the functions involved in problem (1.1) are continuously differentiable, Υ_μ^λ is also a continuously differentiable function for any $\lambda > 0$ and $\mu > 0$. Additionally, we can easily check that

$$\|\Upsilon_\mu^\lambda(z) - \Upsilon^\lambda(z)\| \longrightarrow 0 \quad \text{as } \mu \downarrow 0.$$

Following the smoothing scheme discussed, for example, in [31], our aim is to consider a sequence $\{\mu_k\}$ decreasing to 0 such that equation (2.9) is approximately solved:

$$\Upsilon_{\mu^k}^\lambda(z) = 0, \quad k = 0, 1, \dots$$

for a fixed value of $\lambda > 0$. Hence, we consider the following algorithm for system (4.3):

Algorithm 4.1 Smoothing Gauss–Newton Method for Bilevel Optimization

Step 0: Choose $\lambda > 0$, $\mu_0 \in (0, 1)$, $z^0 := (x^0, y^0, u^0, v^0, w^0)$, $\epsilon > 0$, $K > 0$, set $k := 0$.

Step 1: If $\|\Upsilon^\lambda(z^k)\| < \epsilon$ or $k \geq K$, then stop.

Step 2: Calculate Jacobian $\nabla \Upsilon_{\mu_k}^\lambda(z^k)$ and find the direction

$$d^k = -(\nabla \Upsilon_{\mu_k}^\lambda(z^k)^T \nabla \Upsilon_{\mu_k}^\lambda(z^k))^{-1} \nabla \Upsilon_{\mu_k}^\lambda(z^k)^T \Upsilon^\lambda(z^k). \quad (4.4)$$

Step 3: Calculate $z^{k+1} = z^k + d^k$.

Step 4: Update $\mu_{k+1} = \mu_k^{k+1}$.

Step 5: Set $k := k + 1$ and go to Step 1.

To implement Algorithm 4.1 numerically, we compute the direction by solving

$$\nabla \Upsilon_{\mu_k}^\lambda(z^k)^T \nabla \Upsilon_{\mu_k}^\lambda(z^k) d^k = -\nabla \Upsilon_{\mu_k}^\lambda(z^k)^T \Upsilon^\lambda(z^k). \quad (4.5)$$

The Pseudo-Newton algorithm for the smoothed optimality conditions (4.3) will be the same as Algorithm 4.1 apart from **Step 2**, where the corresponding direction is given by

$$d^k = -\nabla \Upsilon_{\mu_k}^\lambda(z^k)^+ \Upsilon^\lambda(z^k).$$

Another way to deal with the non-differentiability of the Fischer–Burmeister NCP-function is to introduce a *generalized Jacobian* concept for the system (2.9). A semismooth Newton-type method for bilevel optimization following this type of approach is developed in [13]. However, we will not consider this approach here.

Similarly to (3.4)–(3.3), we introduce the matrices $\mathcal{T}^\mu$, Γ^μ , $\mathcal{A}^\mu$, $\mathcal{B}^\mu$, Θ^μ , and $\mathcal{K}^\mu$, where for instance, the pair $(\mathcal{T}^\mu, \Gamma^\mu)$ is defined by $\mathcal{T}^\mu := \text{diag}\{\tau_1^\mu, \dots, \tau_p^\mu\}$ and $\Gamma^\mu := \text{diag}\{\gamma_1^\mu, \dots, \gamma_p^\mu\}$ with

$$\tau_j^\mu := \frac{g_j(x, y)}{\sqrt{u_j^2 + g_j(x, y)^2 + 2\mu}} + 1 \quad \text{and} \quad \gamma_j^\mu := \frac{u_j}{\sqrt{u_j^2 + g_j(x, y)^2 + 2\mu}} - 1, \quad j = 1, \dots, p. \quad (4.6)$$

With this notation, we can easily check that for $\lambda > 0$ and $\mu > 0$, the Jacobian of Υ_μ^λ is

$$\nabla \Upsilon_\mu^\lambda(z) = \begin{bmatrix} \nabla^2 L^\lambda(z) & \nabla g(x, y)^T & \nabla G(x, y)^T & -\lambda \nabla g(x, y)^T \\ \nabla(\nabla_y \mathcal{L}(z)) & O & O & \nabla_y g(x, y)^T \\ \mathcal{T}^\mu \nabla g(x, y) & \Gamma^\mu & O & O \\ \mathcal{A}^\mu \nabla G(x, y) & O & \mathcal{B}^\mu & O \\ \Theta^\mu \nabla g(x, y) & O & O & \mathcal{K}^\mu \end{bmatrix}. \quad (4.7)$$

The fundamental difference between the framework here and the one in the previous section is that for the pair $(\tau_j^\mu, \gamma_j^\mu)$, $j = 1, \dots, p$, for instance, we have the strict inequalities

$$(\tau_j^\mu - 1)^2 + (\gamma_j^\mu + 1)^2 < 1, \quad j = 1, \dots, p$$

instead of equalities in the context of (τ_j, γ_j) , $j = 1, \dots, p$ (3.4). The next lemma illustrates a further difference between the new coefficients in this section and the ones in (3.4).

Lemma 4.2 *For a point $z := (x, y, u, v, w)$ such that $\Upsilon_\mu^\lambda(z) = 0$ with $\lambda > 0$ and $\mu > 0$, it holds that*

$$\begin{aligned} \tau_j^\mu &> 0, & \gamma_j^\mu &< 0, & j &= 1, \dots, p, \\ \alpha_j^\mu &> 0, & \beta_j^\mu &< 0, & j &= 1, \dots, q, \\ \theta_j^\mu &> 0, & \kappa_j^\mu &< 0, & j &= 1, \dots, p. \end{aligned}$$

Proof We prove that $\tau_j^\mu > 0$ and $\gamma_j^\mu < 0$ for $j = 1, \dots, p$; the other cases can be done similarly. For $j = 1, \dots, p$, it follows from (4.2) that $g_j(x, y) = -\frac{\mu}{u_j}$. Hence, we can rewrite τ_j^μ and γ_j^μ as

$$\tau_j^\mu = 1 - \frac{\mu}{u_j \sqrt{u_j^2 + \frac{\mu^2}{u_j^2} + 2\mu}} \quad \text{and} \quad \gamma_j^\mu = \frac{u_j}{\sqrt{u_j^2 + \frac{\mu^2}{u_j^2} + 2\mu}} - 1, \quad (4.8)$$

respectively. Next, we consider the following three scenarios:

Case 1 Suppose that $u_j = \mu$. Substituting this value into (4.8), we arrive at

$$\tau_j^\mu = 1 - \frac{1}{\mu + 1} > 0 \quad \text{and} \quad \gamma_j^\mu = \frac{\mu}{\mu + 1} - 1 < 0 \quad \text{as} \quad \mu > 0.$$

Case 2 Suppose that $u_j = \mu + \delta$ for some $\delta > 0$ and substituting this in (4.8) leads to

$$\begin{aligned} \tau_j^\mu &= 1 - \frac{\mu}{(\mu + \delta) \sqrt{(\mu + \delta)^2 + \frac{\mu^2}{(\mu + \delta)^2} + 2\mu}} = 1 - \frac{1}{\sqrt{\frac{(\mu + \delta)^4}{\mu^2} + 1 + 2 \frac{(\mu + \delta)^2}{\mu}}} > 0, \\ \gamma_j^\mu &= \frac{\mu + \delta}{\sqrt{(\mu + \delta)^2 + \frac{\mu^2}{(\mu + \delta)^2} + 2\mu}} - 1 = \frac{1}{\sqrt{1 + \frac{\mu^2}{(\mu + \delta)^4} + 2 \frac{\mu}{(\mu + \delta)^2}}} - 1 < 0. \end{aligned}$$

Case 3 Finally, suppose that $u_j = \mu - \delta$ for some $\delta > 0$. Then substituting this in (4.8),

$$\begin{aligned} \tau_j^\mu &= 1 - \frac{\mu}{(\mu - \delta) \sqrt{(\mu - \delta)^2 + \frac{\mu^2}{(\mu - \delta)^2} + 2\mu}} = 1 - \frac{1}{\sqrt{\frac{(\mu - \delta)^4}{\mu^2} + 1 + 2 \frac{(\mu - \delta)^2}{\mu}}} > 0, \\ \gamma_j^\mu &= \frac{\mu - \delta}{\sqrt{(\mu - \delta)^2 + \frac{\mu^2}{(\mu - \delta)^2} + 2\mu}} - 1 = \frac{1}{\sqrt{1 + \frac{\mu^2}{(\mu - \delta)^4} + 2 \frac{\mu}{(\mu - \delta)^2}}} - 1 < 0. \end{aligned}$$

Note that $u_j = \mu - \delta > 0$ for in the third case, which can be used to ensure that

$$\mu - \delta = \sqrt{(\mu - \delta)^2}.$$

Next, we use this lemma to provide a condition ensuring that the matrix $\nabla Y_\mu^\lambda(z)^T \nabla Y_\mu^\lambda(z)$ is nonsingular. This will allow the smoothed Gauss–Newton step (4.4) to be well-defined. As in the previous section, it suffices to show that the columns of $\nabla Y_\mu^\lambda(\bar{z})$ are linearly independent.

Theorem 4.3 *For a point $\bar{z} := (\bar{x}, \bar{y}, \bar{u}, \bar{v}, \bar{w})$ verifying (4.3) for some $\mu > 0$ and $0 < \lambda < \frac{\kappa_j^\mu \tau_j^\mu}{\theta_j^\mu}$, $j = 1, \dots, p$, suppose that $\nabla^2 L^\lambda(\bar{z})$ is positive definite. Then, the columns of the matrix $\nabla Y_\mu^\lambda(\bar{z})$ are linearly independent.*

Proof Similarly to the proof of Theorem 3.3, $\nabla Y_\mu^\lambda(\bar{z})(d_1^\top, d_2^\top, d_3^\top, d_4^\top)^\top = 0$ is equivalent to

$$\nabla^2 L^\lambda(\bar{z})d_1 + \nabla g(\bar{x}, \bar{y})^T d_2 + \nabla G(\bar{x}, \bar{y})^T d_3 - \lambda \nabla g(\bar{x}, \bar{y})^T d_4 = 0, \quad (4.9)$$

$$\tau_j^\mu \nabla g_j(\bar{x}, \bar{y})^T d_1 + \gamma_j^\mu d_{2_j} = 0, \quad (4.10)$$

$$\alpha_j^\mu \nabla G_j(\bar{x}, \bar{y})^T d_1 + \beta_j^\mu d_{3_j} = 0, \quad (4.11)$$

$$\theta_j^\mu \nabla g_j(\bar{x}, \bar{y})^T d_1 + \kappa_j^\mu d_{4_j} = 0, \quad (4.12)$$

$$\nabla(\nabla_y \mathcal{L}(\bar{z}))d_1 + \nabla_y g(\bar{x}, \bar{y})^T d_4 = 0, \quad (4.13)$$

where $j = 1, \dots, p$ in (4.10) and (4.12), while $j = 1, \dots, q$ in (4.11). Thanks to Lemma 4.2, we can rewrite equations (4.10), (4.11), and (4.12) as

$$\nabla g_j(\bar{x}, \bar{y})^\top d_1 = -\frac{\gamma_j^\mu}{\tau_j^\mu} d_{2_j}, \quad \nabla G_j(\bar{x}, \bar{y})^\top d_1 = -\frac{\beta_j^\mu}{\alpha_j^\mu} d_{3_j}, \quad \text{and} \quad \nabla g_j(\bar{x}, \bar{y})^\top d_1 = -\frac{\kappa_j^\mu}{\theta_j^\mu} d_{4_j}, \quad (4.14)$$

respectively, with $-\frac{\gamma_j^\mu}{\tau_j^\mu} > 0$, $-\frac{\beta_j^\mu}{\alpha_j^\mu} > 0$, and $-\frac{\kappa_j^\mu}{\theta_j^\mu} > 0$. Now, let us multiply (4.9) by $d_1^\top$:

$$d_1^\top \nabla^2 L^\lambda(\bar{z})d_1 + d_1^\top \nabla g(\bar{x}, \bar{y})^\top d_2 + d_1^\top \nabla G(\bar{x}, \bar{y})^\top d_3 - \lambda d_1^\top \nabla g(\bar{x}, \bar{y})^\top d_4 = 0. \quad (4.15)$$

Using the results above, the equation (4.15) can be written as

$$d_1^\top \nabla^2 L^\lambda(\bar{z})d_1 + \sum_{j=1}^p \left(-\frac{\gamma_j^\mu}{\tau_j^\mu} \right) d_{2_j}^2 + \sum_{j=1}^q \left(-\frac{\beta_j^\mu}{\alpha_j^\mu} \right) d_{3_j}^2 - \lambda \sum_{j=1}^p \left(-\frac{\kappa_j^\mu}{\theta_j^\mu} \right) d_{4_j}^2 = 0. \quad (4.16)$$

Furthermore, it follows from the first and last items of (4.14) that

$$d_{4_j} = -\frac{\theta_j^\mu}{\kappa_j^\mu} \nabla g_j(\bar{x}, \bar{y})^\top d_1 = \frac{\theta_j^\mu \gamma_j^\mu}{\kappa_j^\mu \tau_j^\mu} d_{2_j}. \quad (4.17)$$

Substituting (4.17) into (4.16)

$$d_1^\top \nabla^2 L^\lambda(\bar{z}) d_1 + \sum_{j=1}^p \left(-\frac{\gamma_j^\mu}{\tau_j^\mu} \right) d_{2_j}^2 + \sum_{j=1}^q \left(-\frac{\beta_j^\mu}{\alpha_j^\mu} \right) d_{3_j}^2 - \lambda \sum_{j=1}^p \left(-\frac{\kappa_j^\mu}{\theta_j^\mu} \right) \left(\frac{\theta_j^\mu \gamma_j^\mu}{\kappa_j^\mu \tau_j^\mu} \right)^2 d_{2_j}^2 = 0. \quad (4.18)$$

Rearranging this equation, we get

$$d_1^\top \nabla^2 L^\lambda(\bar{z}) d_1 + \sum_{j=1}^q \left(-\frac{\beta_j^\mu}{\alpha_j^\mu} \right) d_{3_j}^2 + \sum_{j=1}^p \left(1 - \lambda \frac{\theta_j^\mu \gamma_j^\mu}{\kappa_j^\mu \tau_j^\mu} \right) \left(-\frac{\gamma_j^\mu}{\tau_j^\mu} \right) d_{2_j}^2 = 0. \quad (4.19)$$

Then, with Lemma 4.2, and under the assumptions that $\nabla^2 L^\lambda(\bar{z})$ is positive definite and $\lambda < \frac{\kappa_j^\mu \tau_j^\mu}{\theta_j^\mu \gamma_j^\mu}$ for $j = 1, \dots, p$, equation (4.19) is the sum of non-negative terms, which can only be a sum of zeros if all components of d_1, d_2 and d_3 are zeros. Since all components of d_2 are zeros, we can look back to (4.17) or (4.12) to deduce that $d_{4_j} = 0$ for $j = 1, \dots, p$, completing the proof.

It is important to note that the assumption that $\lambda < \frac{\kappa_j^\mu \tau_j^\mu}{\theta_j^\mu \gamma_j^\mu}$ does not necessarily conflict with the requirement that λ be strictly positive, as due to Lemma 4.2, we have $\frac{\kappa_j^\mu \tau_j^\mu}{\theta_j^\mu \gamma_j^\mu} > 0$. In Sect. 5.5, a numerical analysis of this condition is conducted. Next, we provide an example of bilevel program where the assumptions made in the Theorem 4.3 are satisfied.

Example 4.4 Consider the instance of problem (1.1) from the BOLIB Library [41] with

$$\begin{aligned} F(x, y) &:= x^2 + (y_1 + y_2)^2, & G(x, y) &:= -x + 0.5, & g(x, y) &:= \begin{pmatrix} -x - y_1 - y_2 + 1 \\ -y \end{pmatrix}. \\ f(x, y) &:= y_1, \end{aligned}$$

The point $\bar{z} = (\bar{x}, \bar{y}_1, \bar{y}_2, \bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{v}, \bar{w}_1, \bar{w}_2, \bar{w}_3) = (0.5, 0, 0.5, 1, \lambda, 0, 0, 0, 1, 0)$ satisfies equation (2.9) for any $\lambda > 0$. Strict complementarity does not hold at this point as $(\bar{v}, G(\bar{x}, \bar{y})) = (0, 0)$ and $(\bar{w}_1, g_1(\bar{x}, \bar{y})) = (0, 0)$. We observe that $\nabla^2 L^\lambda(\bar{z}) = 2e$, where e is the identity matrix of $\mathbb{R}^{3 \times 3}$, is positive definite. As for the conditions $\lambda < \frac{\kappa_j^\mu \tau_j^\mu}{\theta_j^\mu \gamma_j^\mu}$, $j = 1, 2, 3$, they hold for any value of λ such that

$$0 < \lambda < \min \left\{ \frac{1}{1 - 1/(2\mu + 1)^{1/2}}, \frac{1 - 1/(2\mu + 1)^{1/2}}{1 - \lambda/(\lambda^2 + 2\mu)^{1/2}}, 1 \right\}.$$

This is automatically the case if, for example, we set $\mu = 2 \times 10^{-2}$ and $\lambda = 10^{-2}$. $\square$

There is at least one other way to show that $\nabla Y_\mu^\lambda(\bar{z})^T \nabla Y_\mu^\lambda(\bar{z})$ is nonsingular. The approach is based on the structure of the matrix, as it will be clear in the next result. To proceed, we need the following two assumptions.

Assumption 4.5 Each row of the following matrix is a nonzero vector:

$$[\nabla^2 L^\lambda(z)^T \quad \nabla(\nabla_y \mathcal{L}(z))^T \quad \nabla g(x, y)^T \quad \nabla G(x, y)^T].$$

Assumption 4.6 For $\lambda > 0$ and $\mu > 0$, the diagonal elements of the matrix $\nabla Y_\mu^\lambda(z)^T \nabla Y_\mu^\lambda(z)$ dominate the other terms row-wise; i.e., $a_{ii} > \sum_{j=1, j \neq i}^N |a_{ij}|$ for $i = 1, \dots, N$, where a_{ij} denotes the element in the cell (i, j) of $\nabla Y_\mu^\lambda(z)^T \nabla Y_\mu^\lambda(z)$ for $i = 1, \dots, N$ and $j = 1, \dots, N$.

Lemma 4.7 Let Assumption 4.5 hold at the point $z := (x, y, u, v, w)$. Then for any $\lambda > 0$ and $\mu > 0$, the diagonal elements of the matrix $\nabla Y_\mu^\lambda(z)^T \nabla Y_\mu^\lambda(z)$ are strictly positive.

Proof Considering the Jacobian matrix in (4.7), its transpose can be written as

$$\nabla Y_\mu^\lambda(z)^T = \begin{bmatrix} \nabla^2 L^\lambda(z)^T & \nabla(\nabla_y \mathcal{L}(z))^T & \nabla g(x, y)^T \mathcal{T}^{\mu T} & \nabla G(x, y)^T \mathcal{A}^{\mu T} & \nabla g(x, y)^T \Theta^{\mu T} \\ \nabla g(x, y) & \mathcal{O} & \Gamma^{\mu T} & \mathcal{O} & \mathcal{O} \\ \nabla G(x, y) & \mathcal{O} & \mathcal{O} & \mathcal{B}^{\mu T} & \mathcal{O} \\ -\lambda \nabla g(x, y) & \nabla_y g(x, y) & \mathcal{O} & \mathcal{O} & \mathcal{K}^{\mu T} \end{bmatrix}.$$

Denote by r_i , $i = 1, \dots, 4$, respectively, the first, second, third, and fourth row-block of this matrix. Then the desired product can be represented as

$$\nabla Y_\mu^\lambda(z)^T \nabla Y_\mu^\lambda(z) = \begin{bmatrix} r_1 r_1^T & r_1 r_2^T & r_1 r_3^T & r_1 r_4^T \\ r_2 r_1^T & r_2 r_2^T & r_2 r_3^T & r_2 r_4^T \\ r_3 r_1^T & r_3 r_2^T & r_3 r_3^T & r_3 r_4^T \\ r_4 r_1^T & r_4 r_2^T & r_4 r_3^T & r_4 r_4^T \end{bmatrix}.$$

Obviously, the diagonal elements of $\nabla Y_\mu^\lambda(z)^T \nabla Y_\mu^\lambda(z)$ are the diagonal elements of $r_1 r_1^T$, $r_2 r_2^T$, $r_3 r_3^T$, and $r_4 r_4^T$. We can check that for $j = 1, \dots, n + m$, a diagonal element of $r_1 r_1^T$ has the form

$$\begin{aligned} (r_1 r_1^T)_{jj} &= \sum_{k=1}^{n+m} \nabla_{j,k}^2 L^\lambda(z)^T \nabla_{j,k}^2 L^\lambda(z) + \sum_{k=1}^m \nabla_j(\nabla_{y_k} \mathcal{L}(z))^T \nabla_j(\nabla_{y_k} \mathcal{L}(z)) \\ &\quad + \sum_{k=1}^p \nabla_j g_k(x, y)^T \nabla_j g_k(x, y) (\tau_k^\mu)^2 + \sum_{k=1}^q \nabla G_k(x, y)^T \nabla G_k(x, y) (\alpha_k^\mu)^2 \\ &\quad + \sum_{k=1}^p \nabla g_k(x, y)^T \nabla g_k(x, y) (\theta_k^\mu)^2, \end{aligned}$$

where ∇_j stands for the j th element of $\nabla := (\nabla_{x_1}, \dots, \nabla_{x_n}, \nabla_{y_1}, \dots, \nabla_{y_m})$ and $\nabla_{j,k}^2$ corresponds to an element in the j th row and k th column of

$$\nabla^2 := \begin{bmatrix} \nabla_{x_1 x_1} & \cdots \nabla_{x_1 x_n} & \nabla_{x_1 y_1} & \cdots \nabla_{x_1 y_m} \\ \vdots & \ddots & \vdots & \vdots \\ \nabla_{x_n x_1} & \cdots \nabla_{x_n x_n} & \nabla_{x_n y_1} & \cdots \nabla_{x_n y_m} \\ \nabla_{y_1 x_1} & \cdots \nabla_{y_1 x_n} & \nabla_{y_1 y_1} & \cdots \nabla_{y_1 y_m} \\ \vdots & \ddots & \vdots & \vdots \\ \nabla_{y_m x_1} & \cdots \nabla_{y_m x_n} & \nabla_{y_m y_1} & \cdots \nabla_{y_m y_m} \end{bmatrix}.$$

Combining Assumption 4.5 and Lemma 4.2, it is clear that $(r_1 r_1^T)_{jj} > 0$ for $j = 1, \dots, n + m$. Similarly, the diagonal elements of $r_2 r_2^T$, $r_3 r_3^T$, and $r_4 r_4^T$ can respectively be written as

$$\begin{aligned} (r_2 r_2^T)_{jj} &= \nabla g_j(x, y) \nabla g_j(x, y)^T + (\gamma_j^\mu)^2 & \text{for } j = 1, \dots, p, \\ (r_3 r_3^T)_{jj} &= \nabla G_j(x, y) \nabla G_j(x, y)^T + (\beta_j^\mu)^2 & \text{for } j = 1, \dots, q, \\ (r_4 r_4^T)_{jj} &= \nabla g_j(x, y) \nabla g_j(x, y)^T + (\kappa_j^\mu)^2 & \text{for } j = 1, \dots, p. \end{aligned}$$

Thanks to Lemma 4.2, it is also clear that these items are all strictly positive. $\square$

Theorem 4.8 *Let $z = (x, y, u, v, w)$ be a stationary point of the system (4.3) for some $\lambda > 0$ and $\mu > 0$. If Assumptions 4.5 and 4.6 are satisfied, then the matrix $\nabla \Upsilon_\mu^\lambda(z)^T \nabla \Upsilon_\mu^\lambda(z)$ is nonsingular.*

Proof It is known that the matrix is positive definite if it is symmetric, its diagonal elements are strictly positive, and diagonal elements dominate elements of the matrix in the corresponding row. This property is the consequence of the *Gershgorin circle theorem* [18, page 320]. As $\nabla \Upsilon_\mu^\lambda(z)^T \nabla \Upsilon_\mu^\lambda(z)$ is symmetric, then combining Assumptions 4.5 and 4.6 to Lemma 4.7, we have the result. $\square$

Next, we provide an example where the assumptions required for Theorem 4.8 are satisfied.

Example 4.9 Consider the instance of problem (1.1) taken from the BOLIB Library [41] with

$$F(x, y) := (x - 1)^2 + y^2, \quad f(x, y) := x^2 y, \quad g(x, y) := y^2$$

and no upper-level constraint. For this problem, the function Υ^λ (2.9) can be written as

$$\Upsilon^\lambda(z) = \left(2x - 2, 2y + 2yu - 2\lambda yw, x^2 + 2yw, \sqrt{u^2 + y^4} - u + y^2, \sqrt{w^2 + y^4} - w + y^2 \right)^\top.$$

The first item of note about this example is that the optimal solution $(\bar{x}, \bar{y}) = (1, 0)$ does not satisfy the optimality conditions (2.4)–(2.8), given that $\Upsilon^\lambda(\bar{x}, \bar{y}, \bar{u}, \bar{w}) \neq 0$ for any values of $\bar{u}$ and $\bar{w}$. However, Algorithm 4.1 identifies the solution for λ

taking the values 0.6, 0.7 or 0.8 with the smoothing parameter set to $\mu = 10^{-11}$. Indeed, the convergence of the method seems to be justified as for this problem, we can easily check that for $\bar{x} = 1$ and $\bar{y} = 0$,

$$\nabla^2 L^\lambda(\bar{z}) = 2 \operatorname{diag} (1, 1 + \bar{u} - \lambda \bar{w}), \quad \nabla(\nabla_y \mathcal{L}(\bar{z})) = (2 \ 2\bar{w}), \quad \nabla g(\bar{x}, \bar{y}) = (0 \ 0),$$

$$\Gamma^\mu = \frac{\bar{u}}{\sqrt{\bar{u}^2 + 2\mu}} - 1, \quad \text{and} \quad \mathcal{K}^\mu = \frac{\bar{w}}{\sqrt{\bar{w}^2 + 2\mu}} - 1$$

and subsequently, we have the product

$$\nabla Y_\mu^\lambda(\bar{z})^T \nabla Y_\mu^\lambda(\bar{z}) = \begin{bmatrix} 8 & 4\bar{w} & 0 & 0 \\ 4\bar{w} & 4\bar{w}^2 + (2\bar{u} - 2\lambda\bar{w})^2 & 0 & 0 \\ 0 & 0 & \left(\frac{\bar{u}}{\sqrt{\bar{u}^2 + 2\mu}} - 1\right)^2 & 0 \\ 0 & 0 & 0 & \left(\frac{\bar{w}}{\sqrt{\bar{w}^2 + 2\mu}} - 1\right)^2 \end{bmatrix}.$$

Hence, Assumption 4.5 is clearly satisfied; for Assumption 4.6 to hold, we need

$$8 > 4\bar{w}, \quad 4\bar{w}^2 + (2\bar{u} - 2\lambda\bar{w})^2 > 4\bar{w}, \quad \left(\frac{\bar{u}}{\sqrt{\bar{u}^2 + 2\mu}} - 1\right)^2 > 0, \quad \left(\frac{\bar{w}}{\sqrt{\bar{w}^2 + 2\mu}} - 1\right)^2 > 0,$$

which holds for any $\mu > 0$, $\lambda > 0$, $\bar{u} > 0$, and $1 < \bar{w} < 2$. $\square$

We further note that the assumptions made in Theorem 4.3 hold for the problem in this example. Firstly, we observe that $\nabla^2 L^\lambda(\bar{z})$ is positive definite if $\lambda\bar{w} < \bar{u} + 1$. Subsequently, we can check that both assumptions of Theorem 4.3 are satisfied if

$$\lambda < \min \left\{ \frac{\bar{u} + 1}{\bar{w}}, \frac{\left(\frac{\bar{w}}{\sqrt{\bar{w}^2 + 2\mu}} - 1\right)}{\left(\frac{\bar{u}}{\sqrt{\bar{u}^2 + 2\mu}} - 1\right)} \right\} \quad \text{with} \quad \bar{w} \neq 0.$$

For instance, choosing $\bar{u} = \sqrt{8} \times 10^{-6}$, $\bar{w} = 10^{-6}$, $\mu = 4 \times 10^{-12}$, and $\lambda < 2.25$ gives the result.

To conclude this section, we would like to analyse the *Jacobian consistency* of Y^λ . Recall that according to [3], the Jacobian consistency property will hold for Y^λ if this mapping is Lipschitz continuous and there exists a constant $\epsilon > 0$ such that for any $z \in \mathbb{R}^N$ and $\mu \in \mathbb{R}_+$, we have

$$\|Y_\mu^\lambda(z) - Y^\lambda(z)\| \leq \mu\epsilon \quad \text{and} \quad \lim_{\mu \downarrow 0} \operatorname{dist} \left(\nabla Y_\mu^\lambda(z), \partial_C Y^\lambda(z) \right) = 0. \quad (4.20)$$

Here, *dist* represents the standard distance between a point and a set while $\partial_C Y^\lambda(z)^T$ denotes the C-subdifferential

$$\partial_C \Upsilon^\lambda(z)^T := \partial \Upsilon_1^\lambda(z) \times \cdots \times \partial \Upsilon_{N+m}^\lambda(z), \quad (4.21)$$

commonly used in this context; see, e.g., [21]. In (4.21), $N := n + m + 2p + q$ and $\partial \Upsilon_i^\lambda$, $i = 1, \dots, N + m$ represents the subdifferential in the sense of Clarke. Note that $\partial_C \Upsilon^\lambda(z)^T$ contains the generalized Jacobian in the sense of Clarke of the function Υ^λ . Roughly speaking, the Jacobian consistency property (4.20) translates to a framework ensuring that when the smoothing parameter μ converges to zero, the Jacobian $\nabla \Upsilon_\mu^\lambda(z)$ converges to an element in the C-subdifferential $\partial_C \Upsilon^\lambda(z)$. This property is important in determining the accuracy of the smoothing method in Algorithm 4.1.

Based on (4.21), at all points $z := (x, y, u, v, w)$ satisfying Assumption 3.1,

$$\partial_C \Upsilon^\lambda(z)^T := \{\nabla \Upsilon^\lambda(z)^T\}, \quad (4.22)$$

where $\nabla \Upsilon^\lambda(z)$ is defined by (3.3). For the case when strict complementarity does not hold, elements of $\partial_C \Upsilon^\lambda(z)$ have the same structure as in (3.3) with the only differences being in the terms τ_j , γ_j , α_j , β_j , θ_j , and κ_j for indices j where strict complementarity does not hold. We still have $\partial \Upsilon_i^\lambda(z)$ is the same as $\nabla \Upsilon_i^\lambda(z)$ for rows $i = 1, \dots, n + 2m$. To determine the remaining rows, consider

$$\begin{aligned} \Omega_1 &:= \{j : (u_j, g_j(x, y)) = (0, 0)\}, \\ \Omega_2 &:= \{j : (v_j, G_j(x, y)) = (0, 0)\}, \\ \Omega_3 &:= \{j : (w_j, g_j(x, y)) = (0, 0)\}. \end{aligned}$$

Obviously τ_j and γ_j introduced in (3.4) are not well-defined for $j \in \Omega_1$. Similarly, the same holds for α_j and β_j for $j \in \Omega_2$ and θ_j and κ_j for $j \in \Omega_3$. Following the same procedure as in [21, Proposition 2.1], we define

$$\begin{aligned} \tau_k &:= \zeta_k + 1, \quad \gamma_k := \rho_k - 1 \text{ for some } (\zeta_k, \rho_k) \in \mathbb{R}^2 \text{ such that } \|(\zeta_k, \rho_k)\| \leq 1 \text{ if } k \in \Omega_1, \\ \alpha_k &:= \sigma_k + 1, \quad \beta_k := \delta_k - 1 \text{ for some } (\sigma_k, \delta_k) \in \mathbb{R}^2 \text{ such that } \|(\sigma_k, \delta_k)\| \leq 1 \text{ if } k \in \Omega_2, \\ \theta_k &:= \iota_k + 1, \quad \kappa_k := \eta_k - 1 \text{ for some } (\iota_k, \eta_k) \in \mathbb{R}^2 \text{ such that } \|(\iota_k, \eta_k)\| \leq 1 \text{ if } k \in \Omega_3. \end{aligned}$$

Since we do not assume strict complementarity here, then in contrast to Sect. 3.1, we have

$$(\tau_j - 1)^2 + (\gamma_j + 1)^2 \leq 1, \quad (\alpha_j - 1)^2 + (\beta_j + 1)^2 \leq 1, \quad (\theta_j - 1)^2 + (\kappa_j + 1)^2 \leq 1.$$

Theorem 4.10 *For $\lambda > 0$, the Jacobian consistency property holds for the approximation Υ_μ^λ of Υ^λ .*

Proof First of all, note that Υ^λ is locally Lipschitz continuous. Proceeding as in [21, Corollary 2.4], we can easily check that we have

$$\left\| \Upsilon_\mu^\lambda(z) - \Upsilon^\lambda(z) \right\| \leq \epsilon \sqrt{\mu} \quad \text{with} \quad \epsilon := 2\sqrt{2p} + \sqrt{2q}.$$

$$\lim_{\mu \downarrow 0} \nabla \Upsilon_\mu^\lambda(z) = \lim_{\mu \downarrow 0} \begin{bmatrix} \nabla^2 L^\lambda(z) & \nabla g(x, y)^T & \nabla G(x, y)^T & -\lambda \nabla g(x, y)^T \\ \nabla(\nabla_y \mathcal{L}(z)) & O & O & \nabla_y g(x, y)^T \\ \mathcal{T}^\mu \nabla g(x, y) & \Gamma^\mu & O & O \\ \mathcal{A}^\mu \nabla G(x, y) & O & \mathcal{B}^\mu & O \\ \Theta^\mu \nabla g(x, y) & O & O & \mathcal{K}^\mu \end{bmatrix}, \quad (4.23)$$

where it is easy to see that the first two rows of (4.23) are the same as the first two rows of (3.3), as these are not involving perturbation μ . For the rest of Jacobian, we observe that

$$\lim_{\mu \downarrow 0} \left[\tau_j^\mu \nabla g_j(x, y), \gamma_j^\mu, 0, 0 \right] = \begin{cases} (\tau_j \nabla g_j(x, y) \gamma_j, 00) & \text{for } j \notin \Omega_1 \\ (\nabla g_j(x, y) - 100) & \text{for } j \in \Omega_1 \end{cases} \quad (4.24)$$

and similarly for $\lim_{\mu \downarrow 0} [(\alpha_j^\mu \nabla G_j(x, y) \beta_j^\mu, 00)]$ and $\lim_{\mu \downarrow 0} [(\theta_j^\mu \nabla g_j(x, y) \kappa_j^\mu, 00)]$. This leads to $\lim_{\mu \downarrow 0} \text{dist}(\nabla \Upsilon_\mu^\lambda(z), \partial_C \Upsilon^\lambda(z)) = 0$ as $\partial_C \Upsilon^\lambda(z)$ has the same form in (3.3) with corresponding adjustments to $\zeta_j, \rho_j, \sigma_j, \delta_j, \iota_j$ and η_j for the cases when strict complementarity does not hold.

5 Numerical results

The focus of our experiments in this section will be on the smoothing system (4.3), where we set $\mu := 10^{-11}$ constant throughout all iterations. Based on this system, we test and compare the Gauss–Newton method, the Pseudo-Newton method, and the MATLAB built-in method called *fsolve* (with Levenberg-Marquardt chosen as option). The examples used for the experiments are from the Bilevel Optimization LIBrary of Test Problems (BOLIB) [41], which contains 124 nonlinear examples. The experiments are run in MATLAB, version R2016b, on a MACI64. Here, we present a summary of the results obtained; more details for each example are reported in [16].

For Step 0 of Algorithm 4.1 and the corresponding smoothed Pseudo-Newton algorithm, we set the tolerance to $\epsilon := 10^{-5}$ (see Sect. 5.4 for a justification) and the maximum number of iterations to be $K := 1000$. As for stopping criterion of *fsolve*, the tolerance is set to 10^{-5} as well. For the numerical implementation we calculate the direction d^k by solving (4.5) with Gaussian elimination. Five different values of the penalty parameter are used for all the experiments; i.e., $\lambda \in \{100, 10, 1, 0.1, 0.01\}$, see [16] for details of the values of each solution for a selection of λ . The motivation of using different values of λ comes from the idea of not over-penalizing and not under-penalizing deviation of lower-level objective values from the minimum, as bigger (resp. smaller) values of λ seem to perform better for small (resp. big) values of lower-level objective. After running the experiments for all values of $\lambda \in \{100, 10, 1, 0.1, 0.01\}$, the best one is chosen (see

Table 1 Convexity of the lower-level functions

$f(\cdot, y)$	$g_i(\cdot, y), i = 1, \dots, p$	Total count
Convex	Convex (linear)	55
Convex	Convex (nonlinear)	14
Convex	No constraints	5
Convex	Convex (with CQ satisfied)	74

Table 1 in [16]), i.e. for which the best feasible solution is produced for the particular problem by the tested algorithms. Later in this section, we present the comparison of the performance of the algorithms for the best value of λ . The experiments have shown that the algorithms perform much better if the starting point (x^0, y^0) is feasible. As a default setup, we start with $x^0 = 1_n$ and $y^0 = 1_m$. If the default starting point does not satisfy at least one constraint, we choose a feasible starting point; see [16]. Subsequently, the Lagrange multipliers are initialised at $u^0 = \max \{0.01, -g(x^0, y^0)\}$, $v^0 = \max \{0.01, -G(x, y)\}$, and $w^0 = w^0$.

5.1 Performance profiles

Performance profiles are widely used to compare characteristics of different methods. In this section we consider performance profiles, where $t_{i,s}$ denotes the CPU time to solve problem i by algorithm s . If the optimal solution of problem i is known but it cannot be solved by algorithm s (i.e., upper-level objective function error $> 60\%$), we set $t_{i,s} := \infty$. We then define the *performance ratio* by

$$r_{i,s} := \frac{t_{i,s}}{\min\{t_{i,s} : s \in S\}},$$

where S is the set of solvers. The performance ratio is the ratio of how algorithm s performed to solve problem i compared to the performance of the best performed algorithm from the set S . The *performance profile* can be defined as the cumulative distribution function of the performance ratio:

$$\rho_s(\tau) := \frac{|\{i \in P : r_{i,s} \leq \tau\}|}{n_p},$$

where P is the set of problems and τ is the number measuring performance ratio. The performance profile, $\rho_s(\tau)$, is counting the number of examples for which the performance ratio of the algorithm s is better (smaller) than τ . The performance profile $\rho_s : \mathfrak{R} \rightarrow [0, 1]$ is a non-decreasing function, where the value of $\rho_s(1)$ shows the fraction of the problems for which solver s was the best.

Comparing line-graphs of the performance profiles (cf. Fig. 1), a higher position of a graph indicates better performance of the corresponding algorithm. The value on the y-axis shows the fraction of examples for which the performance ratio is better than T (presented on the x-axis). Figure 1 clearly shows that the Gauss–Newton and Pseudo-Newton method perform better than fsolve. Since the variable for the comparison was CPU time, based on the values of $\rho_s(1)$, we can claim that

Fig. 1 Performance profiles of the methods for 124 problems

Gauss–Newton was the fastest algorithm for about 70% of the problems, Pseudo-Newton for about 60% of the problems and fsolve was the quickest for about 20% of the problems. From the graph, one can also see that Gauss–Newton and Pseudo-Newton methods have $\rho_s(2) = 80\%$, while fsolve only has the value $\rho_s(2) = 30\%$, meaning that fsolve was more than twice worse than the best algorithm for 70% of the problems. Approaching $T = 6$, the Gauss–Newton and Pseudo-Newton methods obtain a performance ratio close to $\rho_s(T) = 90\%$, where fsolve obtains $\rho_s(6) \approx 65\%$. This shows that Gauss–Newton and Pseudo-Newton methods show quite similar performance in terms of CPU time. Clearly, from the perspective of the performance profiles discussed in this subsection, both of these algorithms clearly outperform fsolve for solving our test problems.

5.2 Feasibility check

Considering the structure of the feasible set of problem (1.2), it is critical to check whether the points computed by our algorithms satisfy the value function constraint $f(x, y) \leq \varphi(x)$, as it is not explicitly included in the expression of Υ^λ (2.9). If the lower-level problem is convex in y and a solution generated by our algorithms satisfies (2.5) and (2.8), then it will verify the value function constraint. Conversely, to guaranty that a point (x, y) such that $y \in S(x)$ satisfies (2.5) and (2.8), a constraint qualification (CQ) is necessary. Note that conditions (2.5) and (2.8) are incorporated in the stopping criterion of Algorithm 4.1. To check whether the points obtained are feasible, we first identify the BOLIB examples, where the lower-level problem is convex w.r.t. y ; see the summary of these checks in Table 1. It turns out that a significant number of test examples have linear lower-level constraints. For these examples, the lower-level convexity is automatically satisfied.

There are 14 problems, where $f(\cdot, y)$ and $g_i(\cdot, y)$ with $i = 1, \dots, p$, are convex, but the constraints are not all linear w.r.t. y . For these examples, the lower-level regularity (2.1) has been shown to hold at the points computed by our algorithms. The rest of the problems have non-convex lower-level objective or some of the lower-level constraints are nonconvex. For these examples, we compare the solutions obtained with those known from the literature. Let f_A stand for $f(\bar{x}, \bar{y})$, the

Fig. 2 Lower-level optimality check for examples with a nonconvex lower-level problem

lower-level function value obtained by one of the algorithms tested at convergence, and let f_K be the known optimal value of lower-level objective function. In the graph below we plot the lower-level relative error, $(f_A - f_K)/(1 + |f_K|)$ on the y-axis, against different problems. The errors are plotted in increasing order. Note that for the 25 problems with smallest relative error the error is smaller than 5%.

From the figure above we can see that for 30 problems the relative error of lower-level objective is negligible ($< 5\%$) for all three methods. For almost all of the remaining 19 examples, the Gauss–Newton and Pseudo–Newton methods obtain smaller errors than fsolve, while the Pseudo–Newton method seems to obtain slightly smaller errors than the Gauss–Newton method for some of the examples. We have seen that convexity and a constraint qualification hold for the lower-level problem hold for 74 test examples. Accordingly, (2.5) and (2.8) are guaranteed to hold and hence, the corresponding points are feasible for our bilevel programs. If we allow for a feasibility error of up to 20%, feasibility is satisfied for 113 (91.13%) problems through the Gauss–Newton and Pseudo–Newton methods, and for 110 (88.71%) problems if we proceed with fsolve (Fig. 2).

5.3 Accuracy of the upper-level objective function

Here, we compare the values of the upper-level objective functions at points computed by the algorithms under study, i.e., Gauss–Newton and Pseudo–Newton algorithms and fsolve. For this comparison, we focus our attention only on 116 BOLIB examples [41], as solutions are not known for six of them and the Gauss–Newton algorithm diverges for *NieEtal2017e*, possibly due the singularity of the matrix $\nabla Y_\mu^\lambda(z)^T \nabla Y_\mu^\lambda(z)$; see [16] for more details. To proceed, let $\bar{F}_A$ be the value of upper-level objective function at the point $(\bar{x}, \bar{y})$ at which the algorithm under

Fig. 3 Comparison of upper-level objective values for examples with known solutions

consideration stops, and let $\bar{F}_K$ the value of this function at the known best solution point reported in the literature (see corresponding references in [41]). The comparison is shown in Fig. 3, where we plot the relative error $(\bar{F}_A - \bar{F}_K)/(1 + |\bar{F}_K|)$ on the y -axis against examples on the x -axis. Again, the graph is plotted in the order of increasing error. The 85 test cases with smallest error all exhibit a relative error smaller than 5%.

From Fig. 3, we can see that most of the known best values of upper-level objective functions were recovered by all the methods, as the relative error is close to zero. For 93 of the tested problems, the upper-level objective function error is negligible (i.e., less than 5%) for the solutions obtained by all the three methods. For the remaining 23 examples, it is clear that the errors resulting from *fsolve* are much higher than the ones from the Gauss–Newton and Pseudo–Newton methods. It is worth noting that algorithms perform fairly well for most of the problems. With the accuracy error $\leq 20\%$ our algorithms recovered solutions for 92.31% of the problems, while *fsolve* recovered only 88.03% of the solutions.

5.4 Variation of the tolerance in the stopping criterion

Let us now evaluate the performance of Algorithm 4.1 as we relax the tolerance in the stopping criterion. Setting $\epsilon := 10^{-8}$ (as opposed to $\epsilon := 10^{-5}$ used so far) it turns out that for most of the examples our algorithms stop at the maximum number of iterations, or when the gap between improvement from step to step is too small.

The values of $\|\Upsilon_\mu^\lambda(z)\|$ produced by the algorithms are presented in increasing order in Fig. 4. We can see that *fsolve* performs slightly better for 40 examples, where all algorithms achieve $\|\Upsilon_\mu^\lambda(z)\| \leq 10^{-8}$. For those examples, *fsolve* recovered solutions with better tolerance than the Gauss–Newton and Pseudo–Newton

Fig. 4 Performance of the methods in terms of solving $\Upsilon_\mu^\lambda(z) = 0$

algorithms. This shows that whenever we are able to solve a problem almost exactly, fsolve's stopping criteria is more strict and obtains slightly better values for the system. This can be explained by an additional stopping criteria that we use for Gauss–Newton and Pseudo–Newton methods if the improvement between the steps of the algorithms gets too small. Accordingly, whenever we are able to solve the system $\Upsilon_\mu^\lambda(z) = 0$ almost exactly, fsolve's stopping criteria is somewhat less strict and keeps iterating to produce values of $\|\Upsilon_\mu^\lambda(z)\|$ that are closer to 0 than for the other two methods. The explanation could be that due to an additional stopping criteria for the Gauss–Newton and Pseudo–Newton algorithms stop earlier once significant improvement of the solution is not observed from step to step. If we now look at the performance of the algorithms with tolerances between 10^{-8} and 10^{-6} , the Pseudo–Newton method shows better performance than the other algorithms and fsolve is the weakest among the three methods. This means that if we want to solve a problem with the tolerance of 10^{-6} or better, the Pseudo–Newton algorithm is more likely to recover solutions than the other two methods. The other important observation from Fig. 4 is that choosing $\epsilon := 10^{-5}$ is the most sensible tolerance for our problem set, as better tolerance only allow about 50% of the examples to be solved (i.e., just over 60 examples as we can see from the graph).

5.5 Checking assumption on λ

Considering the importance of the requirement that $\lambda < \frac{\kappa_i^\mu}{\sigma_j^\mu} \frac{c_j^\mu}{r_j^\mu}$ for $j = 1, \dots, p$ in Theorem 4.3, let us analyze its ramifications regarding the solution point generated from Algorithm 4.1 for each value of $\lambda \in \{100, 10, 1, 0.1, 0.01\}$. To simplify the analysis, we introduce

Fig. 5 Checking assumption $\lambda < c^\mu(\lambda)$ for best and optimal values of λ

$$c^\mu(\lambda) := \min_{j=1,\dots,p} \frac{\kappa_j^\mu \tau_j^\mu}{\theta_j^\mu \gamma_j^\mu}.$$

It suffices to check that $\lambda < c^\mu(\lambda)$. We set $\lambda - c^\mu(\lambda) := 100$ if the difference is undefined, and we do not consider problems with no lower-level constraints. (Because the assumption on λ is not necessary in Theorem 4.3 for problems with no lower-level constraint.) Let us also introduce the notions of the best λ and optimal λ , whereby the best λ we mean the values of λ for which $\lambda - c^\mu(\lambda)$ is the smallest and by the optimal we mean the values of λ that were the best to obtain the solution according to [16]. In the figure below we present the difference $\lambda - c^\mu(\lambda)$ on the y-axis and number of the example on the x-axis, following ascending order w.r.t. the values on the y-axis.

Clearly, condition $\lambda < c^\mu(\lambda)$ holds for the values of $\lambda - c^\mu(\lambda)$ lying below the x-axis. From Fig. 5, we can see that the assumption holds for 42 (out of 116) problems for the optimal λ . This means that the condition can hold for many examples. But, obviously, as it is not a necessary condition, our Algorithm 4.1 still converges for many other problems, where the condition is not necessarily satisfied. For the best values of λ , condition $\lambda < c^\mu(\lambda)$ holds for 101 problems. Hence, showing that for most of the examples, there is at least one value of $\lambda \in \{100, 10, 1, 0.1, 0.01\}$ for which the condition is satisfied.

6 Final comments

In this paper, a class of the lower-level value function-based optimality conditions for the bilevel optimization problems has been reformulated as a system of equations using the Fischer–Burmeister function. We have shown that the Gauss–Newton method for such systems can be well-defined and provide a framework for convergence. We then test the method and its smoothed version numerically, alongside with Newton’s method with Moore–Penrose pseudo inverse. The comparison of solutions obtained with known best ones shows that the methods

are appropriate to be used for bilevel programs, as they recover optimal solutions (when known) for most of the problems tested.

It is worth mentioning that whenever $\nabla Y_{\mu}^{\lambda}(z)^T \nabla Y_{\mu}^{\lambda}(z)$ is non-singular throughout all iterations, Gauss–Newton and Pseudo-Newton methods produced the same results as expected. More interestingly, for the 38 test problems for which the Gauss–Newton method could not be implemented due to singularity of the direction matrix for one or more values of λ (see [16]), our conjecture that the Pseudo-Newton would produce reasonable solutions for these cases was correct for 14 problems (e.g., *CalamaiVicente1994c*, *DempeDutta2012b*, and *DempeFranke2011a* in [16]), but fails for the remaining 24 examples (e.g., *Bard1988c*, *Colson2002BIPA3*, and *DempeDutta2012a* in [16]). Nevertheless, we can conclude that our proposed Pseudo-Newton method is indeed somewhat more robust than the Gauss–Newton method. Overall, the Gauss–Newton and Pseudo-Newton methods are more efficient at recovering solutions for BOLIB test problems [41] than the MATLAB built-in function `fsolve`. We also show that our methods are better at producing feasible solutions (i.e., optimal points for the lower-level problem) than `fsolve`; cf. Sect. 5.2.

Acknowledgements The authors would like to thank the two anonymous referees for their constructive remarks that have helped us to improve the quality of this paper. The work of the second author was supported by the University of Southampton Presidential Scholarship and that of third author was funded by an EPSRC grant with reference number EP/P022553/1.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Allende, G.B., Still, G.: Solving bi-level programs with the KKT-approach. *Math. Program.* **131**, 37–48 (2012)
2. Bard, J.F.: *Practical Bilevel Optimization: Algorithms and Applications*. Kluwer Academic Publishers, Cambridge (1998)
3. Chen, X., Qi, L., Sun, D.: Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities. *Math. Comput.* **67**(222), 519–540 (1998)
4. Dempe, S.: A bundle algorithm applied to bilevel programming problems with non-unique lower level solutions. *Comput. Optim. Appl.* **15**(2), 145–166 (2000)
5. Dempe, S.: *Foundations of Bilevel Programming*. Kluwer Academic Publishers, Cambridge (2002)

6. Dempe, S., Dutta, J.: Is bilevel programming a special case of mathematical programming with equilibrium constraints? *Math. Program.* **131**, 37–48 (2010)
7. Dempe, S., Dutta, J., Mordukhovich, B.S.: New necessary optimality conditions in optimistic bilevel programming. *Optimization* **56**(5–6), 577–604 (2007)
8. Dempe, S., Schmidt, H.: On an algorithm solving two-level programming problems with nonunique lower level solutions. *Comput. Optim. Appl.* **6**(3), 227–249 (1996)
9. Dempe, S., Zemkoho, A.B.: The bilevel programming problem: reformulations, constraint qualification and optimality conditions. *Math. Program.* **138**, 447–473 (2013)
10. Dempe, S., Zemkoho, A.B.: The generalized Mangasarian-Fromowitz constraint qualification and optimality conditions for bilevel programs. *J. Optim. Theory Appl.* **148**(1), 46–68 (2011)
11. Dennis, J.E., Schnabel, R.B.: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM Classics in Applied Mathematics, New York (1996)
12. Fischer, A.: A special Newton-type optimization method. *Optimization* **24**(3), 269–284 (1992)
13. Fischer, A., Zemkoho, A.B., Zhou, S.: Semismooth Newton-type method for bilevel optimization: global convergence and extensive numerical experiments (2019). [arXiv:1912.07079](https://arxiv.org/abs/1912.07079)
14. Fletcher, R.: Generalized inverse methods for the best least squares solution of systems of non-linear equations. *Comput. J.* **10**(4), 392–399 (1968)
15. Fletcher, R.: *Practical Methods of Optimization*, 2nd edn. Wiley, Hoboken (1987)
16. Fliege, J., Tin, A., Zemkoho, A.B.: Supplementary material for “Gauss-Newton-type methods for bilevel optimization”. University of Southampton, UK, School of Mathematical Sciences (2020)
17. Gatilov, S.Y.: Using low-rank approximation of the Jacobian matrix in the Newton–Raphson method to solve certain singular equations. *J. Comput. Appl. Math.* **272**, 8–24 (2014)
18. Golub, G.H., Van Loan, C.F.: *Matrix Computations*. The John Hopkins University Press, Baltimore (1996)
19. Jiang, Y., Li, X., Huang, C., Wu, X.: Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bi-level programming problem. *Appl. Math. Comput.* **219**, 4332–4339 (2013)
20. Kanzow, C.: Some noninterior continuation methods for linear complementarity problems. *SIAM J. Matrix Anal. Appl.* **17**(4), 851–868 (1996)
21. Kanzow, C., Pieper, H.: Jacobian smoothing methods for general nonlinear complementarity problems. *SIAM J. Optim.* **9**, 342–372 (1999)
22. Kleniati, P., Adjiman, C.S.: Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: theoretical development. *J. Glob. Optim.* **60**(3), 425–458 (2014)
23. Kleniati, P., Adjiman, C.S.: Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part II: convergence analysis and numerical results. *J. Glob. Optim.* **60**(3), 459–481 (2014)
24. Lampariello, L., Sagratella, S.: Numerically tractable optimistic bilevel problems. *Comput. Optim. Appl.* **76**(2), 277–303 (2020)
25. Lin, G.-H., Xu, M., Ye, J.J.: On solving simple bilevel programs with a nonconvex lower level program. *Math. Program.* **144**(1–2), 277–305 (2014)
26. Mehlitz, P., Minchenko, L.I., Zemkoho, A.B.: A note on partial calmness for bilevel optimization problems with linear structures at the lower level (2020). [arXiv preprint arXiv:2003.06138](https://arxiv.org/abs/2003.06138)
27. Mitsos, A., Lemonidis, P., Barton, P.I.: Global solution of bilevel programs with a nonconvex inner program. *J. Glob. Optim.* **42**(4), 475–513 (2008)
28. Nocedal, J., Wright, S.J.: *Numerical Optimization*. Springer, Berlin (1999)
29. Pan, V., Schreiber, R.: An improved newton iteration for the generalized inverse of a matrix, with applications. *SIAM J. Sci. Stat. Comput.* **12**(5), 1109–1130 (1991)
30. Paulavicius, R., Gao, J., Kleniati, P., Adjiman, C.S.: BASBL: branch-and-sandwich bilevel solver. Implementation and computational study with the BASBLib test sets. *Comput. Chem. Eng.* **132**, 106609 (2020)
31. Qi, L., Sun, D.: A survey of some nonsmooth equations and smoothing Newton methods. In: *Progress in Optimization* Vol. 30 (pp. 121–146), Springer (1999)
32. Stein, O., Tezel, A.: The semismooth approach for semi-infinite programming under the Reduction Ansatz. *J. Glob. Optim.* **41**(2), 245–266 (2008)
33. Stein, O., Tezel, A.: The semismooth approach for semi-infinite programming without strict complementarity. *SIAM J. Optim.* **20**(2), 1052–1072 (2009)
34. Sun, D., Qi, L.: On NCP Functions. *Comput. Optim. Appl.* **13**(1–3), 201–220 (1999)

35. Sun, W., Yuan, Y.-X.: Optimization Theory and Methods. Springer, Berlin (2006)
36. Wiesemann, W., Tsoukalas, A., Kleniati, P., Rustem, B.: Pessimistic bilevel optimization. *SIAM J. Optim.* **23**(1), 353–380 (2013)
37. Xu, M., Ye, J.J.: A smoothing augmented lagrangian method for solving simple bilevel programs. *Comput. Optim. Appl.* **59**(1–2), 353–377 (2014)
38. Xu, M., Ye, J.J., Zhang, L.: Smoothing SQP methods for solving degenerate nonsmooth constrained optimization problems with applications to bilevel programs. *SIAM J. Optim.* **25**(3), 1388–1410 (2015)
39. Ye, J.J., Zhu, D.L.: Optimality conditions for bilevel programming problems. *Optimization* **33**, 9–27 (1995)
40. Zemkoho, A.B.: Bilevel programming: reformulations, regularity and stationarity. Ph.D. thesis, Department of Mathematics and Computer Science, TU Bergakademie Freiberg, Freiberg, Germany (2012)
41. Zhou, S., Zemkoho, A.B., Tin, A.: BOLIB: bilevel optimization LIBrary of test problems. In: Dempe, S., Zemkoho, A.B. (eds.) *Bilevel Optimization: Advances and Next Challenges*. Springer, Berlin (2020)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.