Comput Optim Appl
DOI 10.1007/s10589-017-9903-z

On the resolution of certain discrete univariate max–min problems

Ramsharan Rangarajan[1]

Received: 24 November 2016

Abstract We analyze a class of discrete, univariate, and strictly quasiconcave max–min problems. A distinctive feature of max–min-type optimization problems is the nonsmoothness of the objective being maximized. Here we exploit strict quasiconcavity of the given set of functions to prove existence and uniqueness of the optimizer, and to provide computable bounds for it. The analysis inspires an efficient algorithm for computing the optimizer without having to resort to any regularization or heuristics. We prove correctness of the proposed algorithm and briefly discuss the effect of tolerances and approximate computation. Our study finds direct application in the context of certain mesh deformation methods, wherein the optimal perturbation for a vertex is computed as the solution of a max–min problem of the type we consider here. We include examples demonstrating improvement of simplicial meshes while adopting the proposed algorithm for resolution of the optimization problems involved, and use these numerical experiments to examine the performance of the algorithm.

Keywords Discrete minimax · Quasiconvex optimization · Nonsmooth optimization · Mesh optimization · Mesh smoothing · Polynomial roots

Mathematics Subject Classification 49J35 · 65K10 · 65D18 · 26C10

1 Introduction

Max–min and min–max type optimization problems are ubiquitous in science and engineering, appearing in structural design optimization [1], economics and game theory

✉ Ramsharan Rangarajan
rram@mecheng.iisc.ernet.in

[1] Department of Mechanical Engineering, Indian Institute of Science, Bangalore, India

Published online: 06 March 2017

[2] and image processing [3] to name a few applications. In many instances of such problems, restricting the search for an optimizer in $\mathbb{R}^d$, $d \geq 1$, to a given/computed direction results in a line search problem of the form

$$\text{Find } \lambda_{\text{opt}} = \arg\max_{\lambda \in \mathbb{R}} \min_{1 \leq \alpha \leq n} f_\alpha(\lambda), \tag{1}$$

where $\{f_\alpha\}_{\alpha=1}^n$ is a given set of scalar-valued functions. Equation (1) is termed a discrete max–min problem [4], where *discreteness* emphasizes that $n \in \mathbb{N}$. We are interested here in establishing existence and uniqueness of λ_{opt} when the functions $\{f_\alpha\}_\alpha$ satisfy a specific set of assumptions, and in designing algorithms for computing the optimizer. Most notable among the assumptions on $\{f_\alpha\}_\alpha$ is that of strict quasiconcavity over certain upper level sets. The ensuing analysis of Eq. (1) is then quite straightforward, but crucially, inspires a simple yet robust and efficient algorithm for its resolution.

A characteristic feature of Eq. (1) distinguishing it from straightforward univariate maximization is the fact that even if the given functions $\{f_\alpha\}_\alpha$ are all smooth, the objective

$$F(\lambda) \triangleq \min_{1 \leq \alpha \leq n} f_\alpha(\lambda) \quad \text{for } \lambda \in \mathbb{R},$$

being maximized is in general continuous but not differentiable. Nonsmoothness of F precludes the possibility of using Newton-type methods for maximization in Eq. (1). Similarly, derivative-free methods that require differentiability of the objective to guarantee convergence, are ruled out as well. A recurring idea in the literature for the resolution of such nonsmooth max–min/min–max problems consists in regularizing the objective, cf. [5,6]. In the context of Eq. (1), this consists in replacing F, for instance, by the surrogate

$$F_t(\lambda) = \frac{1}{t} \log \sum_{\alpha=1}^{n} \exp\left(t f_\alpha(\lambda)\right) \quad \text{for } t > 0, \tag{2}$$

which has comparable smoothness as the functions $\{f_\alpha\}_\alpha$. It is then possible to resolve Eq. (1) by computing maximizers of a sequence of functions $\{F_t\}_t$ as defined in Eq. (2) and letting $t \nearrow \infty$, see [7,8]. Equation (2) is an alternative to more commonly employed regularizations using ℓ_p-norms, such as replacing F with the objective $\lambda \mapsto \sum_{\alpha=1}^n |f_\alpha(\lambda)|^p$ with $p = 2$ being a pervasive choice.

The algorithm we propose here consists in restricting the search for the optimizer to an upper level set of F. Exploiting quasiconcavity then narrows the search to a compact interval, and the search is terminated once F is detected to be decreasing. In effect, computing the optimizer entails only computing some of the maxima and pairwise intersections of functions in $\{f_\alpha\}_\alpha$, which in turn reduces to computing roots of smooth functions. We highlight some consequences of quasiconcavity which help in reliably computing these maxima/intersections numerically.

Our motivation to study Eq. (1) stems from designing certain optimization-based mesh deformation methods. We demonstrate an application of the proposed algorithm

in this context in Sect. 4. Mesh deformation methods are routinely used to improve qualities of unstructured meshes in finite element simulations [9–11], to adapt meshes based on *a posteriori* error estimates [12], or even in computer graphics to improve data visualization [13]. In the method discussed in Sect. 4, computing the optimal perturbation for each vertex in a mesh along a prescribed/computed direction requires resolving a max–min problem of type (1), with the qualities of elements around a vertex furnishing the functions $\{f_\alpha\}_\alpha$. The assumption of quasiconcavity is natural in this context [14], as well as in certain generalized linear programs [15].

2 The discrete max–min problem

Let us first introduce the notation necessary to specify the assumptions on the functions $\{f_\alpha\}_\alpha$ in problem (1). For a function $f : \mathbb{R} \to \mathbb{R}$, we denote the y-level set by

$$\mathrm{L}_y(f) \triangleq \{\lambda \in \mathbb{R} \,:\, f(\lambda) = y\},$$

and the corresponding upper level set by

$$\mathrm{L}_y^+(f) \triangleq \{\lambda \in \mathbb{R} \,:\, f(\lambda) \geq y\}.$$

We say that f is *strictly quasiconcave* on a compact interval I if

$$\lambda_1, \lambda_2 \in \mathrm{I} \;\Rightarrow\; f(\eta\lambda_1 + (1-\eta)\lambda_2) > \min\{f(\lambda_1), f(\lambda_2)\} \quad \forall\, \eta \in (0,1). \tag{3}$$

With assumptions **A**0–**A**3 given by

A0 *Feasibility*: $f(0) > 0$,
A1 *Continuity*: f is continuous,
A2 *Decay*: $f(\lambda) \to 0$ as $|\lambda| \to \pm\infty$,
A3 *Strict quasiconcavity*: f is strictly quasiconcave on $\mathrm{L}_y^+(f)$ for each $y \geq f(0)$,

we are interested in resolving the following version of problem (1):

$$\boxed{\text{Given } \{f_\alpha\}_{\alpha=1}^n \text{ satisfying } \mathbf{A}0 - \mathbf{A}3, \text{ find } \lambda_{\mathrm{opt}} = \arg\max_{\lambda\in\mathbb{R}} \min_{1\leq\alpha\leq n} f_\alpha(\lambda).} \tag{4}$$

Some of the specific choices in **A**0–**A**3 are inconsequential for the analysis or resolution of Eq. (4). For instance, we can pick $\lambda_0, y_0 \in \mathbb{R}$ and replace the feasibility condition **A**0 by $f_\alpha(\lambda_0) > y_0$ and the decay condition **A**3 by $\lim_{|\lambda|\to\infty} f_\alpha(\lambda) = y_0$. For definiteness, we have set $\lambda_0 = 0$ and $y_0 = 0$.

2.1 Useful implications of strict quasiconcavity

We record a few consequences of assumptions **A**0–**A**3, and in particular, of strict quasiconcavity, that will be useful for our purposes. Some of these results can be inferred from well known properties of quasiconcave functions [16,17]. Nevertheless,

we include proofs to keep the discussion self-contained, as well as to highlight how the individual assumptions are invoked.

Theorem 1 *Let $f : \mathbb{R} \to \mathbb{R}$ satisfy assumptions* **A***1–***A***3. If $y \geq f(0)$ is such that $L_y(f) \neq \emptyset$, then:*

(i) $\underline{\lambda}_y \triangleq \min\{\lambda \in L_y(f)\}$ *and* $\overline{\lambda}_y \triangleq \max\{\lambda \in L_y(f)\}$ *belong to* $L_y(f)$.
(ii) $L_y^+(f) = [\underline{\lambda}_y, \overline{\lambda}_y]$.
(iii) $L_y(f) = \{\underline{\lambda}_y, \overline{\lambda}_y\}$.

Proof Let y satisfy the hypothesis in the statement.

(i) Continuity of f implies that $L_y(f) = f^{-1}(\{y\})$ is closed. By the decay assumption, we can find $r_y > 0$ such that $|\lambda| > r_y \;\Rightarrow\; f(\lambda) < y/2$. Consequently, $L_y(f)$ is contained in $[-r_y, r_y]$ and is therefore bounded. Hence $L_y(f)$ is compact. Since $L_y(f)$ is nonempty by assumption, it follows that $\underline{\lambda}_y$ and $\overline{\lambda}_y$ both exist. They lie in $L_y(f)$ because $L_y(f)$ is compact.

(ii) Since $L_y(f) \subseteq L_y^+(f)$ and $L_y(f) \neq \emptyset$, $L_y^+(f)$ is nonempty. If $L_y^+(f)$ is a singleton, then we should have $L_y^+(f) = L_y(f) = \{\underline{\lambda}_y = \overline{\lambda}_y\}$, from where claim (ii) follows trivially. In the following therefore, we assume that $L_y^+(f)$ contains at least two distinct points. Notice that if there exists $\lambda \in L_y^+(f) \setminus L_y(f)$, then the decay and continuity assumptions on f imply that $L_y(f)$ contains at least two points, one on either side of λ, by virtue of the intermediate value theorem. In particular,

$$L_y^+(f) \setminus L_y(f) \neq \emptyset \;\Rightarrow\; \underline{\lambda}_y < \overline{\lambda}_y. \tag{5}$$

From the contrapositive of Eq. (5), we get

$$\underline{\lambda}_y = \overline{\lambda}_y \;\Rightarrow\; L_y^+(f) \setminus L_y(f) = \emptyset \;\Rightarrow\; L_y^+(f) = L_y(f) = \{\underline{\lambda}_y = \overline{\lambda}_y\}. \tag{6}$$

Equation (6) shows that

$$\underline{\lambda}_y = \overline{\lambda}_y \;\Rightarrow\; \#L_y^+(f) = 1. \tag{7}$$

Since we know $L_y^+(f)$ contains at least two distinct points, we conclude from Eq. (7) that

$$\#L_y^+(f) \geq 2 \;\Rightarrow\; \underline{\lambda}_y < \overline{\lambda}_y. \tag{8}$$

Let us now proceed to prove claim (ii). Let λ_1 and λ_2 be an arbitrary pair of distinct points in $L_y^+(f)$. Without loss of generality, let $\lambda_1 < \lambda_2$. From Eq. (3), we have

$$\lambda_3 \in (\lambda_1, \lambda_2) \;\Rightarrow\; f(\lambda_3) > \min\{\underbrace{f(\lambda_1)}_{\geq y}, \underbrace{f(\lambda_2)}_{\geq y}\} \;\Rightarrow\; \lambda_3 \in L_y^+(f),$$

leading us to conclude that

$$\lambda_1, \lambda_2 \in \mathrm{L}_y^+(f) \text{ with } \lambda_1 < \lambda_2 \;\Rightarrow\; [\lambda_1, \lambda_2] \subseteq \mathrm{L}_y^+(f). \tag{9}$$

Since $\underline{\lambda}_y, \overline{\lambda}_y \in \mathrm{L}_y^+(f)$ and since Eq. (8) shows $\underline{\lambda}_y < \overline{\lambda}_y$, we set $\lambda_1 = \underline{\lambda}_y$ and $\lambda_2 = \overline{\lambda}_y$ in Eq. (9) to get

$$[\underline{\lambda}_y, \overline{\lambda}_y] \subseteq \mathrm{L}_y^+(f). \tag{10}$$

It only remains to discount the possibility of a strict inclusion in Eq. (10). To this end, suppose there exists $\lambda \in \mathrm{L}_y^+(f)$ such that $\lambda < \underline{\lambda}_y$. Then λ, $\underline{\lambda}_y$ and $\overline{\lambda}_y$ belong to the set $\mathrm{L}_y^+(f)$ over which f is strictly quasiconcave. Then from Eq. (3) we have

$$\lambda < \underline{\lambda}_y < \overline{\lambda}_y \;\Rightarrow\; f(\underline{\lambda}_y) > \min\{\underbrace{f(\lambda)}_{\geq y}, \underbrace{f(\overline{\lambda}_y)}_{=y}\} \;\Rightarrow\; f(\underline{\lambda}_y) > y,$$

which contradicts the fact that $f(\underline{\lambda}_y) = y$. Therefore we conclude that

$$\lambda \in \mathrm{L}_y^+(f) \;\Rightarrow\; \lambda \geq \underline{\lambda}_y. \tag{11}$$

A similar reasoning also shows that

$$\lambda \in \mathrm{L}_y^+(f) \;\Rightarrow\; \lambda \leq \overline{\lambda}_y. \tag{12}$$

Noting Eqs. (11) and (12) in Eq. (10) proves that $\mathrm{L}_y^+(f) = [\underline{\lambda}_y, \overline{\lambda}_y]$.

(iii) Since f is strictly quasiconcave on the set $\mathrm{L}_y^+(f) = [\underline{\lambda}_y, \overline{\lambda}_y]$, Eq. (3) implies

$$\begin{aligned} \lambda \in (\underline{\lambda}_y, \overline{\lambda}_y) \;&\Rightarrow\; f(\lambda) > \min\{\underbrace{f(\underline{\lambda}_y)}_{y}, \underbrace{f(\overline{\lambda}_y)}_{y}\} \\ &\Rightarrow\; f(\lambda) > y \;\Rightarrow\; \mathrm{L}_y(f) \cap (\underline{\lambda}_y, \overline{\lambda}_y) = \emptyset. \end{aligned} \tag{13}$$

However, from $\mathrm{L}_y(f) \subseteq \mathrm{L}_y^+(f)$ and $\mathrm{L}_y^+(f) = [\underline{\lambda}_y, \overline{\lambda}_y]$, we have $\mathrm{L}_y(f) \subseteq [\underline{\lambda}_y, \overline{\lambda}_y]$ as well. Equation (13) therefore leads us to conclude that $\mathrm{L}_y(f) \subseteq \{\underline{\lambda}_y, \overline{\lambda}_y\}$. Since $\underline{\lambda}_y, \overline{\lambda}_y \in \mathrm{L}_y(f)$ by definition, we get $\mathrm{L}_y(f) = \{\underline{\lambda}_y, \overline{\lambda}_y\}$, which proves claim (iii). □

Proposition 1 *Let $f : \mathbb{R} \to \mathbb{R}$ satisfy assumptions* **A***1–***A***3. Then f has a unique global maximizer, that lies in $L_y^+(f)$ for each $y \geq f(0)$ which is such that $L_y(f) \neq \emptyset$.*

Proof Let $y \geq f(0)$ be such that $\mathrm{L}_y(f) \neq \emptyset$. Since $\mathrm{L}_y^+(f) \supseteq \mathrm{L}_y(f)$, $\mathrm{L}_y^+(f)$ is nonempty as well. Part (ii) of theorem 1 shows that $\mathrm{L}_y^+(f)$ is a compact interval. Therefore the continuous function f attains its maximum in $\mathrm{L}_y^+(f)$, which proves

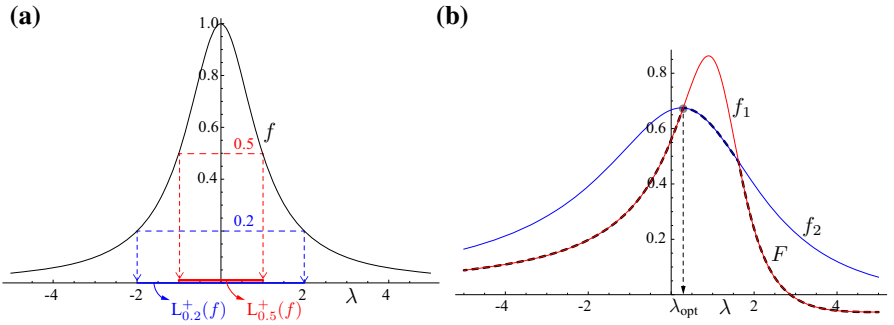


Fig. 1 The function $\lambda \mapsto 1/(1+\lambda^2)$ plotted in (**a**) satisfies conditions **A**0–**A**3. As predicted by Theorem 1, notice that the upper level sets indicated in the figure are compact intervals. Figure (**b**) shows a pair of functions f_1 and f_2 satisfying **A**0–**A**3. As predicted by Theorem 2, their minimum F (*dashed black line*) satisfies these conditions as well, and has a unique maximizer λ_{opt}

the existence of a maximizer in $\mathrm{L}^+_y(f)$. Let us demonstrate its uniqueness next. Let $\hat{y}$ denote the maximum of f over $\mathrm{L}^+_y(f)$. If possible, let λ_1 and λ_2 be distinct maximizers of f in $\mathrm{L}^+_y(f)$. Without loss of generality, let us assume that $\lambda_1 < \lambda_2$. Since $\mathrm{L}^+_y(f)$ is an interval, $[\lambda_1, \lambda_2] \subseteq \mathrm{L}^+_y(f)$. Then from Eq. (3), we have

$$\lambda \in (\lambda_1, \lambda_2) \;\Rightarrow\; f(\lambda) > \min\{\underbrace{f(\lambda_1)}_{\hat{y}}, \underbrace{f(\lambda_2)}_{\hat{y}}\} \;\Rightarrow\; f(\lambda) > \hat{y}. \tag{14}$$

Equation (14) however contradicts the fact that $\hat{y}$ is the maximum of f in $\mathrm{L}^+_y(f)$. Hence we conclude that $f : \mathrm{L}^+_y(f) \to \mathbb{R}$ has a unique maximizer.

It remains to show that the maximizers identified in the upper level sets $\mathrm{L}^+_{y_1}(f)$ and $\mathrm{L}^+_{y_2}(f)$ are identical, provided that $y_1, y_2 \geq f(0)$ and the sets $\mathrm{L}^+_{y_1}(f)$, $\mathrm{L}^+_{y_2}(f)$ are nonempty. Without loss of generality, let us assume that $y_1 > y_2$. Observe that $\mathrm{L}^+_{y_1}(f)$ is a nonempty compact subset of $\mathrm{L}^+_{y_2}(f)$, that $f \geq y_1$ on $\mathrm{L}^+_{y_1}(f)$, and that $f < y_1$ on $\mathrm{L}^+_{y_2}(f_1) \setminus \mathrm{L}^+_{y_1}(f)$ by virtue of part (ii) of Theorem 1. Therefore the maximizer of $f : \mathrm{L}^+_{y_2}(f) \mapsto \mathbb{R}$ necessarily lies in $\mathrm{L}^+_{y_1}(f)$, and hence coincides with the maximizer of $f : \mathrm{L}^+_{y_1}(f) \mapsto \mathbb{R}$. The proposition now follows. □

Figure 1 shows examples of functions satisfying conditions **A**0–**A**3 which are neither convex nor concave. The pair of functions f_1 and f_2 shown in Fig. 1b satisfy **A**0–**A**3. Notice that their minimum F has a unique maximizer. Establishing this in general is the purpose of Theorem 2. It is also clear from the figure that although f_1 and f_2 are smooth, F fails to be differentiable at a few points. Consequently, Newton-type methods may fail to identify the maximizer of F correctly; Algorithms 1 and 2 discussed and analyzed in Sect. 3 provide alternatives instead.

2.2 Existence, uniqueness and bounds for the optimizer

We are now ready to prove existence and uniqueness of the solution λ_{opt} to problem (4). In Theorem 2 stated next, we denote

$$\underline{\lambda}_{y,\alpha} \triangleq \min\{\lambda \in \mathrm{L}_y(f_\alpha)\} \text{ and } \overline{\lambda}_{y,\alpha} \triangleq \max\{\lambda \in \mathrm{L}_y(f_\alpha)\}, \text{ and set } \mathrm{F} \triangleq \min_{1\le\alpha\le n} f_\alpha(lambda).$$

Theorem 2 *Suppose that the functions* $\{f_\alpha\}_{\alpha=1}^n$ *each satisfy* **A***0*–**A***3. Then:*

(i) F *satisfies* **A***0*–**A***3.*
(ii) F *has a unique maximizer. Consequently, Eq. (4) has a unique solution.*
(iii) For each $y \ge F(0)$ *such that* $L_y(F) \ne \emptyset$*, we have*

$$L_y(F) = \{\underline{\Lambda}_y, \overline{\Lambda}_y\} \text{ where } \underline{\Lambda}_y \triangleq \max_{1\le\alpha\le n} \underline{\lambda}_{y,\alpha} \text{ and } \overline{\Lambda}_y \triangleq \min_{1\le\alpha\le n} \overline{\lambda}_{y,\alpha},$$

and consequently the bounds

$$\underline{\Lambda}_y \le \lambda_{opt} \le \overline{\Lambda}_y. \tag{15}$$

(iv) For the specific choice $y_0 = F(0)$*, we have* $0 \in \{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\}$ *and the bounds*

$$\underline{\Lambda}_{y_0} \le \lambda_{opt} \le \overline{\Lambda}_{y_0}. \tag{16}$$

Proof (i) Feasibility, continuity and decay of F follow readily from its definition. Let us verify **A**3, i.e., for $y \ge F(0)$, we need to show that F is strictly quasiconcave on $\mathrm{L}_y^+(F)$. We begin with a few observations. If $\mathrm{L}_y^+(F)$ is either empty or a singleton, there is nothing to prove. In the following therefore, we assume that $\mathrm{L}_y^+(F)$ contains at least two distinct points. Since

$$\mathrm{L}_y^+(F) = \cap_{\alpha=1}^n \mathrm{L}_y^+(f_\alpha), \tag{17}$$

each set $\mathrm{L}_y^+(f_\alpha)$ is nonempty. Then part (ii) of Theorem 1 implies that $\mathrm{L}_y^+(f_\alpha) = [\underline{\lambda}_{y,\alpha}, \overline{\lambda}_{y,\alpha}]$. It therefore follows from Eq. (17) that $\mathrm{L}_y^+(F)$ is itself a compact interval. Notice from

$$y \ge F(0) \;\Rightarrow\; y \ge \min_{1\le\alpha\le n} f_\alpha(0) \;\Rightarrow\; y \ge f_\alpha(0)\ \forall\, \alpha \in \{1, \ldots, n\} \tag{18}$$

that each function f_α is strictly quasiconcave on the set $\mathrm{L}_y^+(f_\alpha)$, and consequently, over the common intersection $\mathrm{L}_y^+(F)$.

We can now prove strict quasiconcavity of F over $\mathrm{L}_y^+(F)$. Consider a pair of distinct points $\lambda_1, \lambda_2 \in \mathrm{L}_y^+(F)$, and without loss of generality, assume $\lambda_1 < \lambda_2$. Since $\mathrm{L}_y^+(F)$ has a nonempty interior (being a compact interval containing at least

two distinct points), we know $(\lambda_1, \lambda_2) \in \mathrm{L}_y^+(F)$ as well. Using the fact that each function f_α is strictly quasiconcave on $\mathrm{L}_y^+(F)$, Eq. (3) yields

$$\lambda_1 < \lambda_3 < \lambda_2 \;\Rightarrow\; f_\alpha(\lambda_3) > \min\{f_\alpha(\lambda_1),\, f_\alpha(\lambda_2)\} \quad \forall\, \alpha \in \{1, \ldots, n\}. \tag{19}$$

Using Eq. (19) and the definition of F, we get

$$\begin{aligned} \lambda_1 < \lambda_3 < \lambda_2 \;\Rightarrow\; F(\lambda_3) &= \min_{1 \le \alpha \le n} f_\alpha(\lambda_3) \\ &> \min_{1 \le \alpha \le n} \min\{f_\alpha(\lambda_1),\, f_\alpha(\lambda_2)\} \\ &\ge \min\{\min_{1 \le \alpha \le n} f_\alpha(\lambda_1),\, \min_{1 \le \alpha \le n} f_\alpha(\lambda_2)\} \\ &= \min\{F(\lambda_1),\, F(\lambda_2)\}, \end{aligned}$$

which verifies condition **A**3 for F.

(ii) Set $y_0 = F(0)$ and notice that $\mathrm{L}_{y_0}(F)$ contains 0 and is therefore nonempty. Having verified in part (i) that F satisfies **A**1–**A**3, invoking proposition 1 with $y = y_0$ and $f = F$ shows that F has a unique global maximizer, namely λ_{opt}.

(iii) Let $y \ge F(0)$ be such that $\mathrm{L}_y(F) \ne \emptyset$. We have

$$\begin{aligned} \mathrm{L}_y^+(F) &= \cap_{1 \alpha \le n} \mathrm{L}_y^+(f_\alpha) && \text{(see Eq. (17))} \\ &= \cap_{\alpha=1}^n \left[\underline{\lambda}_{y,\alpha}, \overline{\lambda}_{y,\alpha}\right] && \text{(using Eq. (18) and Theorem 1)} \\ &= \left[\max_{1 \le \alpha \le n} \underline{\lambda}_{y,\alpha},\, \min_{1 \le \alpha \le n} \overline{\lambda}_{y,\alpha}\right] \\ &= [\underline{\Lambda}_y, \overline{\Lambda}_y] \end{aligned} \tag{20}$$

That $\mathrm{L}_y(F) = \{\underline{\Lambda}_y, \overline{\Lambda}_y\}$ now follows from Eq. (20) and part (iii) of Theorem 1. Invoking Proposition 1 with $f = F$ shows that $\lambda_{\mathrm{opt}} \in \mathrm{L}_y^+(F)$, i.e., $\lambda_{\mathrm{opt}} \in [\underline{\Lambda}_y, \overline{\Lambda}_y]$, which is the bound claimed in Eq. (15).

(iv) That $\underline{\Lambda}_{y_0} \le \lambda_{\mathrm{opt}} \le \overline{\Lambda}_{y_0}$ follows from setting $y = y_0 = F(0)$ in part (iii). Finally,

$$y_0 = F(0) \;\Rightarrow\; 0 \in \mathrm{L}_{y_0}(F) \;\Rightarrow\; 0 \in \{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\},$$

where we have invoked part (iii) of Theorem 1 to deduce the second implication. □

The choice $y_0 = F(0)$ is an obvious candidate satisfying the requirements $y_0 \ge F(0)$ and $\mathrm{L}_{y_0}(F) \ne \emptyset$ in Theorem 2. This is of course a consequence of explicitly assuming that each function in $\{f_\alpha\}_\alpha$ satisfies the feasibility condition **A**0. For $y > y_0$, the strict inclusion $\mathrm{L}_y^+(F) \subsetneq \mathrm{L}_{y_0}^+(F)$ implies that the bounds $\underline{\Lambda}_y \le \lambda_{\mathrm{opt}} \le \overline{\Lambda}_y$ are necessarily tighter than $\underline{\Lambda}_{y_0} \le \lambda_{\mathrm{opt}} \le \overline{\Lambda}_{y_0}$. Indeed, choosing $y = F(\lambda_{\mathrm{opt}})$ narrows the bounds in Eq. (15) to the singleton $\underline{\Lambda}_y = \overline{\Lambda}_y = \lambda_{\mathrm{opt}}$. However, it is not clear how to choose $y > y_0$ *a priori*, while ensuring $\mathrm{L}_y(F) \ne \emptyset$ and without incurring

significant additional computations. Moreover, the examples in Sect. 4 reveal that the bound $\underline{\Lambda}_{y_0} \leq \lambda_{\text{opt}} \leq \overline{\Lambda}_{y_0}$ is quite useful as it is.

3 Resolution algorithms

We now take up the question of computing the optimizer λ_{opt} in problem (4). We discuss two algorithms, both of which rely on λ_{opt} coinciding with either a maximum of one of the functions in $\{f_\alpha\}_\alpha$ or with a point where two or more among them intersect. To this end, it is necessary for us to assume that the number of pairwise intersections of functions in $\{f_\alpha\}_\alpha$ is finite[1]. Such an assumption is reasonable in max–min problems arising in the geometric mesh smoothing problem we consider in Sect. 4. In general however, it is possible that distinct functions in $\{f_\alpha\}_\alpha$ overlap over finite intervals or intersect at infinitely many points. It then becomes necessary to resort to interval-based algorithms instead [7].

In contrast to explicitly limiting the number of pairwise intersections, it is not necessary for us to assume that the number of maxima of functions in $\{f_\alpha\}_\alpha$ be finite. As we show next, this is an automatic consequence of strict quasiconcavity.

Proposition 2 *Let $f : \mathbb{R} \to \mathbb{R}$ satisfy assumptions* **A***1–***A***3. If $y \geq f(0)$ is such that $L_y(f) \neq \emptyset$ and $\underline{\lambda}_y < \overline{\lambda}_y$, then:*

(i) f has no local maximizers in $(\underline{\lambda}_y, \overline{\lambda}_y)$ besides the global maximizer, and
(ii) f has no local minimizers in $(\underline{\lambda}_y, \overline{\lambda}_y)$.

Proof Let y satisfy the hypotheses in the statement. We prove both claims by reduction to the absurd.

(i) Proposition 1 shows that f has a unique global maximizer in $\mathrm{L}_y^+(f)$, say $\lambda_\star$. Since part (ii) of Theorem 1 shows $\mathrm{L}_y^+(f) = [\underline{\lambda}_y, \overline{\lambda}_y]$ and we have assumed $\underline{\lambda}_y < \overline{\lambda}_y$, it follows that $\lambda_\star \in (\underline{\lambda}_y, \overline{\lambda}_y)$. If possible, let $\lambda \in (\underline{\lambda}_y, \overline{\lambda}_y)$ be a local maximizer of f that is distinct from $\lambda_\star$. Without loss of generality, let us assume that $\lambda < \lambda_\star$. Then, we can find $\varepsilon > 0$ such that $\lambda + \varepsilon < \lambda_\star$ and $f \leq f(\lambda)$ on $(\lambda, \lambda+\varepsilon)$. Noting that f is strictly quasiconcave on the interval $\mathrm{L}_y^+(f)$ containing λ, $\lambda + \varepsilon/2$ and $\lambda_\star$, Eq. (3) yields

$$\lambda < \lambda + \varepsilon/2 < \lambda_\star \Rightarrow f(\lambda + \varepsilon/2) > \min\{f(\lambda), \underbrace{f(\lambda_\star)}_{\geq f(\lambda)}\}$$
$$\Rightarrow f(\lambda + \varepsilon/2) > f(\lambda),$$

which contradicts the fact that $f \leq f(\lambda)$ on the interval $(\lambda, \lambda + \varepsilon)$.

(ii) Next, suppose that $\lambda \in (\underline{\lambda}_y, \overline{\lambda}_y)$ is a local minimum of f. Then we can find $\varepsilon > 0$ such that $(\lambda - \varepsilon, \lambda + \varepsilon) \subset (\underline{\lambda}_y, \overline{\lambda}_y)$ and $f \geq f(\lambda)$ on $(\lambda - \varepsilon, \lambda + \varepsilon)$. Since f

[1] More precisely, we require the number of pairwise intersections of functions within the interval $[\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}]$ to be finite.

is strictly quasiconcave on the interval $\mathrm{L}_y^+(f)$ containing the points $\lambda \pm \varepsilon/2$ and λ, Eq. (3) shows

$$\lambda - \varepsilon/2 < \lambda < \lambda + \varepsilon/2 \Rightarrow f(\lambda) > \min\{\underbrace{f(\lambda - \varepsilon/2)}_{\geq f(\lambda)}, \underbrace{f(\lambda + \varepsilon/2)}_{\geq f(\lambda)}\}$$
$$\Rightarrow f(\lambda) > f(\lambda),$$

which is of course absurd.

□

3.1 A naïve algorithm

Denote the set of all pairwise intersections of functions in $\{f_\alpha\}_\alpha$ by

$$\Lambda_{\text{int}} = \cup_{1 \leq \alpha < \beta \leq n} \{\lambda \in \mathbb{R} : f_\alpha(\lambda) = f_\beta(\lambda)\}$$

and the set of (global) maximizers of the functions $\{f_\alpha\}_\alpha$ by

$$\Lambda_{\max} = \cup_{1 \leq \alpha \leq n} \{\lambda_\alpha^{\max}\},$$

where $\lambda_\alpha^{\max} \triangleq \arg\max_{\lambda \in \mathbb{R}} f_\alpha(\lambda)$ is the maximizer of f_α (see Proposition 2). The first algorithm for identifying the optimizer in Eq. (4) consists in exhaustively computing all points in $\Lambda_{\text{int}} \cup \Lambda_{\max}$. The rationale underlying Algorithm 1 is that if Λ_{int} is finite, then λ_{opt} necessarily belongs to $\Lambda \triangleq \Lambda_{\text{int}} \cup \Lambda_{\max}$, as we show in Proposition 3 below. Since we also know from Proposition 2 that $\Lambda_{\max}$ is a finite set (containing at most n distinct points), the line search in resolving $\lambda_{\text{opt}} = \arg\max_{\lambda \in \mathbb{R}} F(\lambda)$ is replaced by the computation $\lambda_{\text{opt}} = \arg\max_{\lambda \in \Lambda} F(\lambda)$, which only requires comparing the values of F at finitely many points.

Algorithm 1 Brute force identification of the optimizer λ_{opt} for problem (4)

1: **function** NaïveCalcOptimizer($\{f_\alpha\}_{\alpha=1}^n$)
Ensure: Functions $\{f_\alpha\}_\alpha$ are distinct, satisfy conditions **A**0–**A**3 and the set Λ_{int} is finite
2: Initialize $\Lambda = \emptyset$
3: **for** $\alpha = 1$ to n **do**
4: Update $\Lambda \leftarrow \Lambda \cup \{\lambda_\alpha^{\max}\}$
5: **for** $\beta = \alpha + 1$ to n **do**
6: Update $\Lambda \leftarrow \Lambda \cup \{\lambda \in \mathbb{R} : f_\alpha(\lambda) = f_\beta(\lambda)\}$
7: **end for**
8: **end for**
9: **return** $\lambda_{\text{opt}} = \arg\max_{\lambda \in \Lambda} F(\lambda)$
10: **end function**

Proposition 3 *If the functions $\{f_\alpha\}_\alpha$ satisfy assumptions* **A***0–***A***3 and the set Λ_{int} is finite, then $\lambda_{opt} \in (\Lambda_{int} \cup \Lambda_{max}) \setminus \{\pm\infty\}$.*

Proof Let us show that $\lambda_{\text{opt}} \in \Lambda_{\text{int}} \cup \Lambda_{\max}$. The decay assumption **A**2 on each function f_α implies that Λ_{int} necessarily contains $\pm\infty$. Then, noting that Λ_{int} is a finite set, let

$$\Lambda_{\text{int}} = \{\lambda_0 = \infty < \lambda_1 \ldots < \lambda_m < \lambda_{m+1} = +\infty\}, \tag{21}$$

for some $m \in \mathbb{N}$. Evidently, the functions $\{f_\alpha\}_\alpha$ are non-intersecting on each interval $(\lambda_s, \lambda_{s+1})$, $0 \le s \le m$. Hence for each $s \in \{0, \ldots, m\}$ we can find a unique index $\alpha_s \in \{1, \ldots, n\}$ such that

$$f_{\alpha_s}(\lambda) = F(\lambda) \text{ for } \lambda \in \mathrm{I}_s \triangleq [\lambda_s, \lambda_{s+1}]. \tag{22}$$

Setting $y_0 = F(0)$, observe using Eq. (15) that $\lambda_{\text{opt}} \in \mathrm{L}^+_{y_0}(F)$, and that

$$F = \min_{1\le\alpha\le n} f_\alpha \;\Rightarrow\; \mathrm{L}^+_{y_0}(F) = \cap_{\alpha=1}^n \mathrm{L}^+_{y_0}(f_\alpha) \;\Rightarrow\; \mathrm{L}^+_{y_0}(F) \subseteq \mathrm{L}^+_{y_0}(f_\alpha)\, \forall \alpha \in \{1, \ldots, n\}. \tag{23}$$

We now have

$$\begin{aligned}
&\lambda_{\text{opt}} = \arg\max_{\lambda\in\mathbb{R}} F(\lambda) && \text{(by definition)}\\
\Rightarrow\; &\lambda_{\text{opt}} \in \mathrm{L}^+_{y_0}(F) \cap \left(\cup_{0\le s\le m}\{\arg\max_{\lambda\in\mathrm{I}_s} F(\lambda)\}\right) && (\lambda_{\text{opt}} \in \mathrm{L}^+_{y_0}(F) \text{ and } \cup_{0\le s\le m} \mathrm{I}_s = \mathbb{R})\\
\Rightarrow\; &\lambda_{\text{opt}} \in \mathrm{L}^+_{y_0}(F) \cap \left(\cup_{0\le s\le m}\{\arg\max_{\lambda\in\mathrm{I}_s} f_{\alpha_s}(\lambda)\}\right) && \text{(from Eq. (22))}\\
\Rightarrow\; &\lambda_{\text{opt}} \in \cup_{0\le s\le m}\left(\mathrm{L}^+_{y_0}(F) \cap \{\arg\max_{\lambda\in\mathrm{I}_s} f_{\alpha_s}(\lambda)\}\right)\\
\Rightarrow\; &\lambda_{\text{opt}} \in \cup_{0\le s\le m}\left(\mathrm{L}^+_{y_0}(f_{\alpha_s}) \cap \{\arg\max_{\lambda\in\mathrm{I}_s} f_{\alpha_s}(\lambda)\}\right) && \text{(using Eq. (23))}\\
\Rightarrow\; &\lambda_{\text{opt}} \in \cup_{0\le s\le m}\{\lambda_s, \lambda_{s+1}, \lambda^{\max}_{\alpha_s}\} && \text{(using Propositions 1 and 2)}\\
\Rightarrow\; &\lambda_{\text{opt}} \in \underbrace{\left(\cup_{0\le s\le m}\{\lambda_s, \lambda_{s+1}\}\right)}_{\Lambda_{\text{int}}} \cup \underbrace{\left(\cup_{1\le\alpha\le n}\{\lambda^{\max}_\alpha\}\right)}_{\Lambda_{\max}}\\
\Rightarrow\; &\lambda_{\text{opt}} \in \Lambda_{\text{int}} \cup \Lambda_{\max}. && (24)
\end{aligned}$$

Although it appears from Eqs. (21) and (24) that $\pm\infty$ are candidates for the optimizer, they can be safely discounted because F satisfies the feasibility and decay conditions. That is, since

$$\lim_{\lambda\to\pm\infty} F(\lambda) = \lim_{\lambda\to\pm\infty} \min_{1\le\alpha\le n} f_\alpha(\lambda) = \min_{1\le\alpha\le n} \lim_{\lambda\to\pm\infty} f_\alpha(\lambda) = 0$$

and $F(\lambda_{\text{opt}}) \ge F(0) > 0$, it is evident that $\lambda_{\text{opt}} \notin \{\pm\infty\}$. The claim now follows from Eq. (24). □

A few remarks concerning Algorithm 1 are in order.

(i) Once intersections and maxima constituting the set Λ have been computed, Proposition 3 guarantees that λ_{opt} necessarily belongs to Λ and Theorem 2 guarantees that identifying λ_{opt} simply as the point in Λ where F has maximum value is unambiguous.

(ii) An appealing feature of the algorithm is its simplicity. In particular, it does not require identifying the functions in $\{f_\alpha\}_\alpha$ that are *active*[2] over distinct intervals in $\mathbb{R}$. Notice that in the proof of Proposition 3, specifically in Eq. (22), we do in fact identify the active function f_{α_s} over the interval I_s as an intermediate step. This does not however manifest in the algorithm itself.

(iii) It may be possible to generalize Proposition 3 to include cases when Λ_{int} is not finite. Even if correct, Algorithm 1 as given is unlikely to be applicable (or useful) in such scenarios simply because Λ would still contain infinitely many points.

(iv) It is evident that the algorithm requires computing n maximizers of the functions $\{f_\alpha\}_\alpha$ and intersections of $n(n+1)/2$ distinct pairs of functions. However, a large number of these points are irrelevant for the purpose of identifying λ_{opt}. The examples discussed in Sects. 4.4 and 4.6 convey this observation quite emphatically. The algorithm also requires evaluating the objective at each point in Λ, which is an expensive operation in itself because computing F at a point in turn requires evaluating each of the functions $\{f_\alpha\}_\alpha$.

3.2 An algorithm based on limited construction of the objective

Scrutinizing Eq. (24) in the proof of Proposition 3 reveals that not tracking the active functions defining F introduces a large number of unnecessary points in the set Λ of candidate optimizers, which in turn renders Algorithm 1 expensive in practice. We remedy this in Algorithm 2 by explicitly constructing F over a compact interval containing λ_{opt}. Three aspects of the algorithm contribute to its efficiency:

- Limiting the construction of F to the the upper level set $\mathrm{L}^+_{y_0}(F) = [\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}]$, where $y_0 = F(0)$.
- Tracking the active functions defining F within $[\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}]$.
- Terminating the search for λ_{opt} once F is detected to be decreasing.

Algorithm 2 calls three functions CalcBounds, CalcMaximizer and FindActiveFunc. For the purpose of our current discussion, it suffices for us to know what these functions compute:

(i) The function CalcBounds with signature

$$\{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\} = \textsc{CalcBounds}(\{f_\alpha\}_\alpha)$$

computes the bounds $\underline{\Lambda}_{y_0} = \max_{1\le\alpha\le n} \underline{\lambda}_{y_0,\alpha}$ and $\overline{\Lambda}_{y_0} = \min_{1\le\alpha\le n} \overline{\lambda}_{y_0,\alpha}$ identified in Eq. (16) for the optimizer. We adopt the convention that the function returns two values even if $\underline{\Lambda}_{y_0}$ and $\overline{\Lambda}_{y_0}$ coincide.

[2] We say that f_α is active over the interval I if $F = f_\alpha$ on I.

Algorithm 2 Compute λ_{opt} in problem (4) by constructing F within $[\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}]$

1: **function** CALCOPTIMIZER($\{f_\alpha\}_{\alpha=1}^n$)
Ensure: Functions $\{f_\alpha\}_\alpha$ are distinct, satisfy **A**0–**A**3 and the set Λ_{int} is finite
2: Compute $\{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\} \leftarrow$ CALCBOUNDS($\{f_\alpha\}_\alpha$) ▷ Note: $0 \in \{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\}$
3: **if** $\underline{\Lambda}_{y_0} = \overline{\Lambda}_{y_0}$ **then**
4: **return** 0
5: **end if**
6: Initialize $\{\alpha_{\text{act}}, \lambda_{\text{next}}\} \leftarrow$ FINDACTIVEFUNC($\{f_\alpha\}_\alpha, \underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}$)
7: Initialize $\lambda_{\text{prev}} \leftarrow \underline{\Lambda}_{y_0}$
▷ $F = f_{\alpha_{\text{act}}}$ on $[\lambda_{\text{prev}}, \lambda_{\text{next}}]$
8: **while** true **do**
9: $\{\lambda_{\alpha_{\text{act}}}^{\max}, \text{flag}\} \leftarrow$ CALCMAXIMIZER($f_{\alpha_{\text{act}}}, \lambda_{\text{prev}}, \lambda_{\text{next}}$)
10: **if** flag = true **then**
11: **return** $\lambda_{\alpha_{\text{act}}}^{\max}$
12: **end if**
13: **if** $f_{\alpha_{\text{act}}}(\lambda_{\text{next}}) < f_{\alpha_{\text{act}}}(\lambda_{\text{prev}})$ **then**
14: **return** λ_{prev}
15: **end if**
16: Update $\lambda_{\text{prev}} \leftarrow \lambda_{\text{next}}$
17: Update $\{\alpha_{\text{act}}, \lambda_{\text{next}}\} \leftarrow$ FINDACTIVEFUNC($\{f_\alpha\}_\alpha, \lambda_{\text{prev}}, \overline{\Lambda}_{y_0}$)
18: **end while**
19: **assert**(false) ▷ Impossible to be here
20: **end function**

(ii) The function CALCMAXIMIZER with signature

$$\{\lambda_\alpha^{\max}, \text{flag}\} = \text{CALCMAXIMIZER}(f_\alpha, \lambda_{\text{prev}}, \lambda_{\text{next}})$$

returns the boolean-valued parameter *flag* as true if the maximizer of f_α lies in the interval $[\lambda_{\text{prev}}, \lambda_{\text{next}}]$, and as false otherwise. If *flag* is returned as true, then the maximizer $\arg\max_{\lambda\in\mathbb{R}} f_\alpha(\lambda)$ is returned as $\lambda_\alpha^{\max}$.

(iii) The function FINDACTIVEFUNC with signature

$$\{\alpha_{\text{act}}, \lambda_{\text{next}}\} = \text{FINDACTIVEFUNC}(\{f_\alpha\}_\alpha, \lambda_{\text{prev}}, \lambda_{\max})$$

computes a coordinate λ_{next} and an index $\alpha_{\text{act}} \in \{1, \ldots, n\}$ such that

$$\lambda_{\text{next}} \in \big((\lambda_{\text{prev}}, \lambda_{\max}] \cap \Lambda_{\text{int}}\big) \cup \{\lambda_{\max}\} \text{ and } F = f_{\alpha_{\text{act}}} \text{ on } [\lambda_{\text{prev}}, \lambda_{\text{next}}]. \quad (25)$$

The details of how these three functions are implemented is not important. A few ideas to facilitate a robust implementation are mentioned in Sect. 3.3.4 and at the end of the discussion in Sect. 4.6.

3.3 Analysis of Algorithm 2

Our first goal in discussing Algorithm 2 is establishing the following result:

Theorem 3 *If the functions $\{f_\alpha\}_\alpha$ satisfy* **A**0–**A**3 *and the set Λ_{int} is finite, then Algorithm 2 is well defined, terminates and returns λ_{opt}.*

In the statement, by the algorithm being well defined, we mean that each step is unambiguous. Specifically, the function call CALCMAXIMIZER appearing in Step 9 requires that $\lambda_{\text{next}} \geq \lambda_{\text{prev}}$. This is automatically ensured by eq. (25). The function calls FINDACTIVEFUNC appearing in Steps 6, and 17 require that $\lambda_{\text{prev}} < \overline{\Lambda}_{y_0}$. We will therefore have to demonstrate that FINDACTIVEFUNC is never invoked with $\lambda_{\text{prev}} = \overline{\Lambda}_{y_0}$.

For the proof of Theorem 3, we proceed through a sequence of simple steps detailed in Sects. 3.3.1 and 3.3.2. In the remainder of this section, we assume without explicit mention that the hypotheses in Theorem 3 hold, i.e., that the functions $\{f_\alpha\}_\alpha$ satisfy **A**0–**A**3 and that the set Λ_{int} is finite. Similarly, we will freely use the fact that λ_{opt} is unique as shown in Theorem 2.

The choice $y_0 = F(0)$ in the algorithm implies that $\mathrm{L}_{y_0}(F) \neq \emptyset$. Hence Theorem 2 shows that $\lambda_{\text{opt}} \in \mathrm{L}^+_{y_0}(F) = [\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}]$. Notice that if $\underline{\Lambda}_{y_0} = \overline{\Lambda}_{y_0}$, then the bounds $\underline{\Lambda}_{y_0} \leq \lambda_{\text{opt}} \leq \overline{\Lambda}_{y_0}$ from Eq. (16) implies that λ_{opt} coincides with the common value of $\underline{\Lambda}_{y_0}$ and $\overline{\Lambda}_{y_0}$, which by virtue of part (iv) of Theorem 2, is zero. This justifies Step 4 in Algorithm 2, see Proposition 7. On the other hand, if $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$, then Proposition 1 shows that $\lambda_{\text{opt}} \in (\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0})$. Therefore, we conclude that

$$\begin{cases} \lambda_{\text{opt}} \in \{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\} & \iff \quad \underline{\Lambda}_{y_0} = \overline{\Lambda}_{y_0} = 0 \\ \lambda_{\text{opt}} \in (\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}) \text{ otherwise.} \end{cases} \tag{26}$$

3.3.1 Iterations

If $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$, it follows from Eq. (25) that recursive calls of the form

$$\{\alpha_s, \lambda_s\} = \text{FINDACTIVEFUNC}(\{f_\alpha\}_\alpha, \lambda_{s-1}, \overline{\Lambda}_{y_0})\} \text{ with } \lambda_0 \triangleq \underline{\Lambda}_{y_0}, \ s = 1, 2, \ldots \tag{27}$$

identifies the sequences of points

$$\{\lambda_0 = \underline{\Lambda}_{y_0} < \lambda_1 < \ldots < \lambda_N = \overline{\Lambda}_{y_0}\} \subseteq \left([\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}] \cap \Lambda_{\text{int}}\right) \cup \{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\}$$

and a corresponding sequence of indices $\{\alpha_s\}_{s=1}^N \subseteq \{1, \ldots, n\}$ such that

$$F = f_{\alpha_s} \text{ over } [\lambda_{s-1}, \lambda_s] \text{ for each } 1 \leq s \leq N. \tag{28}$$

That N is finite in Eq. (28) is a consequence of our assumption that Λ_{int} is a finite set.

Although an iteration counter is not explicitly used in the algorithm, it is convenient to introduce one, say s, for the ensuing arguments. Initializing s to be 1 prior to Step 8 and incrementing it by 1 after Step 17, the s^{th} iteration of the algorithm consists in executing Steps 9 to 17 enclosed in the *while* loop when the iteration counter has value s. In view of Eq. (28), we also identify $\lambda_{\text{prev}} = \lambda_{s-1}$, $\lambda_{\text{next}} = \lambda_s$ and $\alpha_{\text{act}} = \alpha_s$ during the s^{th} iteration.

Proposition 4 *If* $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$, *then*

$$\lambda_{opt} \in \{\lambda_1, \ldots, \lambda_{N-1}\} \cup \left(\cup_{s=1}^{N} \left(\{\lambda_{\alpha_s}^{max}\} \cap (\lambda_{s-1}, \lambda_s) \right) \right). \tag{29}$$

Proof From $\mathrm{L}_{y_0}^{+}(F) = [\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}]$ and Eq. (28), we have

$$\begin{aligned} & \lambda_{\mathrm{opt}} \in \underset{\lambda \in \mathrm{L}_{y_0}^{+}(F)}{\arg\max}\ F(\lambda) \\ \Rightarrow\ & \lambda_{\mathrm{opt}} \in \cup_{1 \le s \le N} \underset{\lambda \in [\lambda_{s-1}, \lambda_s]}{\arg\max}\ F(\lambda) \\ \Rightarrow\ & \lambda_{\mathrm{opt}} \in \cup_{1 \le s \le N} \underset{\lambda \in [\lambda_{s-1}, \lambda_s]}{\arg\max}\ f_{\alpha_s}(\lambda) \end{aligned} \tag{30}$$

In Eq. (30), observe that[3]

$$\underset{\lambda \in [\lambda_{s-1}, \lambda_s]}{\arg\max}\ f_{\alpha_s}(\lambda) \subseteq \{\lambda_{s-1}, \lambda_s\} \cup \left(\{\lambda_{\alpha_s}^{\max}\} \cap (\lambda_{s-1}, \lambda_s) \right), \tag{31}$$

where we have exploited the fact that f_{α_s} has no local maxima besides $\lambda_{\alpha_s}^{\max}$, see Propositions 1 and 2. Using Eq. (31) in Eq. (30), we get

$$\begin{aligned} & \lambda_{\mathrm{opt}} \in \cup_{1 \le s \le N} \left(\{\lambda_{s-1}, \lambda_s\} \cup \left(\{\lambda_{\alpha_s}^{\max}\} \cap (\lambda_{s-1}, \lambda_s) \right) \right) \\ \Rightarrow\ & \lambda_{\mathrm{opt}} \in \underbrace{\left(\cup_{1 \le s \le N} \{\lambda_{s-1}, \lambda_s\} \right)}_{\{\lambda_0, \lambda_1, \ldots, \lambda_N\}} \cup \left(\cup_{s=1}^{N} \left(\{\lambda_{\alpha_s}^{\max}\} \cap (\lambda_{s-1}, \lambda_s) \right) \right). \end{aligned} \tag{32}$$

Equation (29) now follows from Eq. (32), because Eq. (26) specifically shows $\lambda_{\mathrm{opt}} \notin \{\lambda_0, \lambda_N\}$ when $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$. □

Equation (29) represents a clear improvement over Proposition 3 for identifying λ_{opt}, especially because $\{\lambda_1, \ldots, \lambda_{N-1}\}$ is generally a small subset of Λ_{int}. In this sense at least, we expect Algorithm 2 to be more efficient than Algorithm 1. This prospect however comes at the expense of tracking the active functions defining F, as required in the definition of the sequence $\{\lambda_s\}_s$.

Proposition 5 *If* $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$ *and* $\eta \in [\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}]$, *then*

$$\left.\begin{array}{l} \eta < \lambda_s \le \lambda_{opt} \\ \lambda_{opt} \le \lambda_s < \eta \end{array}\right\} \ \Rightarrow\ F(\eta) < F(\lambda_s) \text{ for } 1 \le s < N. \tag{33}$$

Notice that it is not necessary to consider the case $s = N$ in Eq. (33)— while $\lambda_N \le \lambda_{\mathrm{opt}}$ is ruled out by the assumption $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$ and Eq. (26), we are not interested in the case $\eta > \lambda_N$.

[3] Arguing as we did in Propositions 1 and 2, we can shows that f_{α_s} has a unique maximizer over $[\lambda_{s-1}, \lambda_s]$. For the purpose of arriving at Eq. (32) however, the inclusion noted in Eq. (31) is sufficient.

Proof Let us prove that

$$\eta < \lambda_s \leq \lambda_{\text{opt}} \;\Rightarrow\; F(\eta) < F(\lambda_s) \text{ for } 1 \leq s < N. \tag{34}$$

For $s \in \{1, \ldots, N-1\}$, we have

$$\eta < \lambda_s = \lambda_{\text{opt}} \;\Rightarrow\; \eta \neq \lambda_{\text{opt}} \;\Rightarrow\; F(\eta) < F(\lambda_{\text{opt}}), \tag{35}$$

which proves Eq. (34) when $\lambda_s = \lambda_{\text{opt}}$. If $\lambda_s < \lambda_{\text{opt}}$ on the other hand, we use the fact that F is strictly quasiconcave over $\mathrm{L}^+_{y_0}(F)$ from part (i) of Theorem 2. Noting that $\eta, \lambda_s, \lambda_{\text{opt}} \in \mathrm{L}^+_{y_0}(F)$ in Eq. (3) shows

$$\eta < \lambda_s < \lambda_{\text{opt}} \;\Rightarrow\; F(\lambda_s) > \min\{\underbrace{F(\eta)}_{<F(\lambda_{\text{opt}})}, F(\lambda_{\text{opt}})\} \;\Rightarrow\; F(\lambda_s) > F(\eta). \tag{36}$$

Equations (35) and (36) prove Eq. (34). We omit a proof of the remaining case $\lambda_{\text{opt}} \leq \lambda_s < \eta$ in Eq. (33), which follows along the same lines as Eqs. (35) and (36). □

For the proof of Theorem 3, we will mainly need the next corollary of Proposition 5, which follows from choosing η to be either λ_{s-1} or λ_{s+1} in Eq. (33).

Corollary 1 *If* $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$ *and* $1 \leq s < N$, *then*

$$\lambda_s \leq \lambda_{opt} \;\Rightarrow\; F(\lambda_{s-1}) < F(\lambda_s). \tag{37a}$$
$$\lambda_s \geq \lambda_{opt} \;\Rightarrow\; F(\lambda_{s+1}) < F(\lambda_s). \tag{37b}$$

Proposition 6 *If* $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$, *then* $\lambda^{max}_{\alpha_1} \neq \lambda_0, \lambda^{max}_{\alpha_N} \neq \lambda_N$ *and*

$$s \in \{1, \ldots, N\},\ \lambda^{max}_{\alpha_s} \in [\lambda_{s-1}, \lambda_s] \;\Rightarrow\; \lambda_{opt} = \lambda^{max}_{\alpha_s}. \tag{38}$$

Proof Consider the possibility that $\lambda^{\max}_{\alpha_1} = \lambda_0$. Since λ_0 is the maximizer of f_{α_1}, we have

$$\begin{aligned}\lambda_0 < \lambda \leq \lambda_N \;\Rightarrow\; F(\lambda) &= \min_{1 \leq \alpha \leq n} f_\alpha(\lambda) \\ &\leq f_{\alpha_1}(\lambda) < f_{\alpha_1}(\lambda_0) = F(\lambda_0) \;\Rightarrow\; F(\lambda) < F(\lambda_0),\end{aligned}$$

showing that λ_0 is the maximizer of F in $[\lambda_0, \lambda_N]$. Hence $\lambda^{\max}_{\alpha_1} = \lambda_0 \Rightarrow \lambda_{\text{opt}} = \lambda_0$, which contradicts Eq. (26). Therefore we conclude that $\lambda^{\max}_{\alpha_1} \neq \lambda_0$. A similar argument shows that $\lambda^{\max}_{\alpha_N} = \lambda_N$ implies $\lambda_{\text{opt}} = \overline{\Lambda}_{y_0}$, contradicting Eq. (26) again. Therefore $\lambda^{\max}_{\alpha_N} \neq \lambda_N$ either.

Let us proceed to proving Eq. (38). If $\lambda^{\max}_{\alpha_s} \in (\lambda_{s-1}, \lambda_s)$, then $\lambda^{\max}_{\alpha_s}$ is a local maximizer of F in the interval $(\lambda_{s-1}, \lambda_s)$ and Proposition 2 implies that $\lambda^{\max}_{\alpha_s} = \lambda_{\text{opt}}$.

It remains to consider the possibilities $\lambda_{\alpha_s}^{\max} \in \{\lambda_{s-1}, \lambda_s\}$. Suppose that $\lambda_{\alpha_s}^{\max} = \lambda_{s-1}$. Since we have already shown that $\lambda_{\alpha_1}^{\max} \neq \lambda_0$, we have $2 \leq s \leq N$. Then,

$$F = \min_{1 \leq \alpha \leq n} f_\alpha \leq f_{\alpha_s} \leq f_{\alpha_s}(\lambda_{\alpha_s}^{\max}) = F(\lambda_{s-1})$$

reveals that λ_{s-1} is a maximizer of F. Proposition 2 then confirms that $\lambda_{\text{opt}} = \lambda_{s-1}$. The remaining possibility, namely, $\lambda_{\alpha_s}^{\max} = \lambda_s$ is handled in a similar manner. □

3.3.2 Termination and correctness

There are exactly three ways in which Algorithm 2 can terminate— at Steps 4, 11 or 14. We examine conditions for termination at each one of these steps and show that the algorithm always returns the correct value of λ_{opt} in each case.

Proposition 7 *If $\underline{\Lambda}_{y_0} = \overline{\Lambda}_{y_0}$, then Algorithm 2 terminates at Step 4 and returns λ_{opt}.*

Proof If $\underline{\Lambda}_{y_0} = \overline{\Lambda}_{y_0}$, Eq. (26) shows that λ_{opt} equals the common value, which by virtue of part (iv) of Theorem 2 is indeed zero. This observation justifies Step 4, where the algorithm terminates. □

Proposition 8 *Suppose that $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$ and $\lambda_{\alpha_s}^{max} \in [\lambda_{s-1}, \lambda_s]$ for some $s \in \{1, \ldots, N\}$. Then Algorithm 2 terminates at Step 11 in the j^{th} iteration, where $j \triangleq \min\{s : \lambda_{\alpha_s}^{max} \in [\lambda_{s-1}, \lambda_s]\}$, and returns λ_{opt}.*

Proof Since $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$, the algorithm does not terminate at Step 4. Therefore $j \geq 1$. If the algorithm reaches the j^{th} iteration, then it is evident from the hypothesis $\lambda_{\alpha_j}^{\max} \in [\lambda_{j-1}, \lambda_j]$ that the check in Step 10 is passed and consequently the algorithm terminates at Step 11. Proposition 6 also shows that the algorithm returns the correct value of λ_{opt}. To prove the proposition, we therefore need to show that the conditions in Steps 10 and 13 are not satisfied during iterations 1 through $j-1$, i.e., that

$$1 \leq s \leq j-1 \;\Rightarrow\; \begin{cases} \lambda_{\alpha_s}^{\max} \notin [\lambda_{s-1}, \lambda_s], & \text{(39a)} \\ F(\lambda_{s-1}) < F(\lambda_s). & \text{(39b)} \end{cases}$$

While (39a) is a consequence of the definition of index j in the statement, using Eq. (37a) of corollary 1 shows

$$\lambda_{\text{opt}} \in [\lambda_{j-1}, \lambda_j] \;\Rightarrow\; \lambda_{j-1} \leq \lambda_{\text{opt}} \;\Rightarrow\; F(\lambda_{s-1}) < F(\lambda_s) \text{ for } 1 \leq s \leq j-1,$$

which proves (39b). □

Proposition 9 *Suppose that $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$. If $\lambda_{opt} = \lambda_j$ for some $j \in \{\lambda_1, \ldots, \lambda_{N-1}\}$ and $\lambda_{\alpha_j}^{max} \neq \lambda_j$, then Algorithm 2 terminates at Step 14 during the j^{th} iteration and returns λ_{opt}.*

Proof Since $\underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$, the algorithm does not terminate at Step 4. To prove the proposition, we need verify three conditions—that the algorithm does not terminate at Step 11 during iterations 1 through j, i.e.,

$$1 \leq s \leq j \;\Rightarrow\; \lambda_{\alpha_s}^{\max} \notin [\lambda_{s-1}, \lambda_s], \tag{40}$$

that the algorithm does not terminate at Step 14 during iterations 1 through $j-1$, i.e.,

$$1 \leq s \leq j-1 \;\Rightarrow\; F(\lambda_{s-1}) < F(\lambda_s), \tag{41}$$

and that the algorithm terminates at Step 14 at the j^{th} iteration, i.e.,

$$F(\lambda_{j+1}) < F(\lambda_j). \tag{42}$$

Let us prove Eqs. (40) to (42). From Proposition 6, we know that

$$s \in \{1, \ldots, N\},\; \lambda_{\alpha_s}^{\max} \in [\lambda_{s-1}, \lambda_s] \;\Rightarrow\; \lambda_{\text{opt}} = \lambda_{\alpha_s}^{\max}. \tag{43}$$

Since we have assumed $\lambda_{\text{opt}} = \lambda_j$, the contrapositive of Eq. (43) yields

$$\lambda_{\alpha_s}^{\max} \notin [\lambda_{s-1}, \lambda_s] \text{ for } 1 \leq s \leq j-1 \tag{44a}$$

$$\lambda_{\alpha_j}^{\max} \notin [\lambda_{j-1}, \lambda_j). \tag{44b}$$

The explicit assumption $\lambda_{\alpha_j}^{\max} \neq \lambda_j$ together with Eq. (44b) shows that

$$\lambda_{\alpha_j}^{\max} \notin [\lambda_{j-1}, \lambda_j]. \tag{45}$$

Equation (40) now follows from Eqs. (44a) and (45). Noting $\lambda_j = \lambda_{\text{opt}}$ in Eqs. (37a) and (37b) of corollary 1 proves Eqs. (41) and (42) respectively. □

3.3.3 *Proof of Theorem 3*

From Eq. (26) and Proposition 4, we have

$$\lambda_{\text{opt}} \in \underbrace{\{\lambda_0, \lambda_N\}}_{\Lambda^0} \cup \underbrace{\{\lambda_1, \ldots, \lambda_{N-1}\}}_{\Lambda^0_{\text{int}}} \cup \underbrace{\left(\cup_{s=1}^{N} \left(\{\lambda_{\alpha_s}^{\max}\} \cap (\lambda_{s-1}, \lambda_s)\right)\right)}_{\Lambda^0_{\max}}, \tag{46}$$

In Eq. (46), observe that the sets Λ^0, Λ^0_{int} and $\Lambda^0_{\max}$ are disjoint. If $\lambda_{\text{opt}} \in \Lambda^0$, Eq. (26) shows $\underline{\Lambda}_{y_0} = \overline{\Lambda}_{y_0}$. Then Proposition 7 proves our claim. Otherwise, Eq. (26) shows that $\lambda_{\text{opt}} \notin \Lambda^0 \;\Rightarrow\; \underline{\Lambda}_{y_0} < \overline{\Lambda}_{y_0}$. If $\lambda_{\text{opt}} \in \Lambda^0_{\max}$, then there exists $s \in \{1, \ldots, N\}$ such that $\lambda_{\text{opt}} = \{\lambda_{\alpha_s}^{\max}\} \cap (\lambda_{s-1}, \lambda_s)$. In particular, $\lambda_{\alpha_s}^{\max} \in [\lambda_{s-1}, \lambda_s]$ for some s, and hence Proposition 8 proves our claim. Finally, consider the possibility $\lambda_{\text{opt}} \in \Lambda^0_{\text{int}}$. Let $j \in \{1, \ldots, N-1\}$ be the (unique) index such that $\lambda_{\text{opt}} = \lambda_j$. If $\lambda_j = \lambda_{\alpha_j}^{\max}$,

Proposition 8 proves our claim. Otherwise, $\lambda_j \neq \lambda_{\alpha_j}^{\max}$ and the theorem follows from Proposition 9.

3.3.4 Remarks on implementing Algorithm 2

(i) Owing to the choice $y_0 = F(0)$, Theorem 2 shows that 0 necessarily belongs to $\{\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}\}$. Hence only one of $\underline{\Lambda}_{y_0}$ and $\overline{\Lambda}_{y_0}$ needs to be computed by CALCBOUNDS. With α_0 such that $f_{\alpha_0}(0) = F(0)$, it is straightforward to identify which one among $\underline{\Lambda}_{y_0}$ and $\overline{\Lambda}_{y_0}$ equals zero and which one needs to be computed. For instance, if $\underline{\lambda}_{y_0,\alpha_0} = 0$, then $\underline{\Lambda}_{y_0} = 0$ and it suffices to compute $\overline{\Lambda}_{y_0}$.

(ii) When computing $\underline{\Lambda}_{y_0}$ or $\overline{\Lambda}_{y_0}$, it is possible to avoid evaluating some of the $\underline{\lambda}_{y_0,\alpha}$'s and $\overline{\lambda}_{y_0,\alpha}$'s by resorting to simple bracketing rules.

(iii) Routines to compute the level sets $\mathrm{L}_{y_0}(f_\alpha)$ can exploit the fact that these sets contains precisely one or two points. In the former case, $\underline{\lambda}_{y_0,\alpha} = \overline{\lambda}_{y_0,\alpha} = 0$ and consequently, $\lambda_{\mathrm{opt}} = 0$ as well.

(iv) Implementations of the routine CALCMAXIMIZER can limit the search for the maximizer $\lambda_{\alpha_s}^{\max}$ to the interval $[\lambda_{s-1}, \lambda_s]$ during the s^{th} iteration of the algorithm. The parameter *flag* returned by the routine is specifically meant to facilitate checking necessary conditions for this purpose, rather than explicitly computing $\lambda_{\alpha_s}^{\max}$ and then verifying if it lies in $[\lambda_{s-1}, \lambda_s]$. Implementations can also take advantage of the fact that f_α has precisely one maximizer. We caution that despite the claims in Proposition 2, the condition $f'_\alpha = 0$ is not sufficient to identify $\lambda_\alpha^{\max}$ because such points may be points of inflection (assuming that f_α is differentiable).

(v) Since F is continuous at the points $\{\lambda_s\}_s$ identified in Eq. (28), the routine FINDACTIVEFUNC can use the active function f_{α_s} known at the s^{th} iteration to efficiently identify the function $f_{\alpha_{s+1}}$ that is active at the next iteration. The arguments passed to the routine will of course have to be modified accordingly.

(vi) Ensuring that successive active functions are distinct, i.e., that $\alpha_s \neq \alpha_{s+1}$, helps avoid extraneous iterations in the algorithm. This can be realized by setting

$$\lambda_{s+1} = \max\left\{\lambda \in \left((\lambda_s, \overline{\Lambda}_{y_0}] \cap \Lambda_{\mathrm{int}}\right) \cup \{\overline{\Lambda}_{y_0}\} \,:\, F = f_{\alpha_{s+1}} \text{ over } [\lambda_s, \lambda]\right\}.$$

(vii) The algorithm is not limited to the choice $y_0 = F(0)$, but applies for any $y > F(0)$ which is such that $\mathrm{L}_y(F) \neq 0$. As mentioned in the remarks following Theorem 2, choosing $y > F(0)$ necessarily narrows the bounds for λ_{opt}.

(viii) If n is large or if λ_{opt} is far away from 0, it is possible that the algorithm requires a large number of iterations to terminate. If it is acceptable to return an approximate optimizer, then the algorithm can be terminated after a fixed number of iterations. At the end of s iterations, λ_{s+1} is the best approximation to λ_{opt}. Alternatively, with $y = F(\lambda_s)$, the algorithm can return the bounds $\mathrm{L}_y^+(F) = [\overline{\Lambda}_y, \underline{\Lambda}_y]$ as the interval containing the optimizer at the end of s iterations.

We intentionally refrain from providing generic algorithms for the routines CalcBounds, CalcMaximizers and FindActiveFunc because incorporating the ideas mentioned above is important in practice, but inevitably requires additional information about the functions $\{f_\alpha\}_\alpha$.

3.4 Tolerances

An important caveat in our description and analysis of Algorithms 1 and 2 is the consideration of tolerances. Tolerances are necessarily introduced by numerical computations adopted for root finding and computations performed with finite precision. We limit the discussion that follows to approximate calculation of roots (level sets, intersections and maximizers) in Algorithms 1 and 2. In particular, we assume that the objective is evaluated exactly and hence do not account for the errors introduced by performing arithmetic with finite precision.

3.4.1 Effect of approximate root finding in Algorithm 1

Suppose that the points in the set $\Lambda = \{\xi_i\}_{i=1}^m$ in Algorithm 1 are computed approximately, say as $\Lambda' = \{\xi_i'\}_{i=1}^m$, where ξ_i' is understood to be an approximation of ξ_i. We are interested in comparing $\lambda_{\text{opt}} = \arg\max_{\lambda\in\Lambda} F(\lambda)$ and $\lambda'_{\text{opt}} = \arg\max_{\lambda'\in\Lambda'} F(\lambda')$, with the intention of showing that they are close to each other if corresponding points in Λ and Λ' are themselves close.

Noting that Λ' is a finite set with bounded elements, let $X \subset \mathbb{R}$ be a compact set containing Λ'. Since F is uniformly continuous on X, given $\varepsilon > 0$, there exists $\delta > 0$ such that

$$\lambda, \lambda' \in X,\ |\lambda - \lambda'| < \delta \ \Rightarrow\ |F(\lambda) - F(\lambda')| < \varepsilon. \tag{47}$$

Let us suppose that $\max_{1\le i\le m} |\xi_i - \xi_i'| < \delta$. Let $\lambda_{\text{opt}} = \xi_I$ for some $I \in \{1, \ldots, m\}$, while $\lambda'_{\text{opt}} = \xi'_J$ for some $J \in \{1, \ldots, m\}$ instead. Using Eq. (47) and the fact that $F(\xi'_J) \ge F(\xi'_I)$, we get

$$F(\lambda_{\text{opt}}) - F(\lambda'_{\text{opt}}) = F(\xi_I) - F(\xi'_J) \le F(\xi_I) - F(\xi'_I) < \varepsilon. \tag{48}$$

We would like to claim using Eq. (48) that λ'_{opt} is close to λ_{opt}. However, mere continuity of F does not suffice for this. As the next proposition shows, it is necessary to exploit the fact that F has a unique maximizer[4].

Proposition 10 *Let $f : X \to \mathbb{R}$ be a continuous map on a compact set $X \subset \mathbb{R}$ such that f has a unique maximum at $\lambda_\star \in X$. Then the function*

$$\delta(\varepsilon) \triangleq \max_{\lambda\in X} \{|\lambda - \lambda_\star| \ :\ f(\lambda) \ge f(\lambda_\star) - \varepsilon\} \text{ for } \varepsilon \ge 0 \tag{49}$$

is well defined, monotonically non-increasing as $\varepsilon \searrow 0$ and $\lim_{\varepsilon\searrow 0} \delta(\varepsilon) = 0$.

[4] We thank Prof. Adrian Lew (Stanford University) for helpful discussions regarding Proposition 10.

Proof Let $\varepsilon \geq 0$ be arbitrary. Since f is continuous, $X_\varepsilon \triangleq \{\lambda : f(\lambda) \geq f(\lambda_\star) - \varepsilon\}$ is a closed subset of X. Compactness of X hence implies that X_ε is compact. We know X_ε is non-empty because $\lambda_\star \in X_\varepsilon$. Therefore $\delta(\varepsilon) = \max_{\lambda \in X_\varepsilon} |\lambda - \lambda_\star|$ is well defined. That $\delta(\varepsilon)$ is non-decreasing is evident from $0 \leq \varepsilon_1 \leq \varepsilon_2 \Rightarrow X_{\varepsilon_1} \subseteq X_{\varepsilon_2} \Rightarrow \delta(\varepsilon_1) \leq \delta(\varepsilon_2)$.

Since $\delta(\varepsilon) \geq 0$ for each $\varepsilon \geq 0$, to prove $\lim_{\varepsilon \searrow 0} \delta(\varepsilon)$ exists and equals zero, it suffices to demonstrate that $\limsup_{\varepsilon \searrow 0} \delta(\varepsilon) = 0$. We argue by contradiction. Suppose that $\limsup_{\varepsilon \searrow 0} \delta(\varepsilon) = 2\eta > 0$. Then we can find a sequence $\{\varepsilon_n > 0\}_n$ such that $\delta(\varepsilon_n) > \eta$ for each $n \in \mathbb{N}$ and $\varepsilon_n \searrow 0$. Noting that X_{ε_n} is non-empty and compact for each n, set $\lambda_n \triangleq \arg\max_{x \in X_{\varepsilon_n}} |\lambda - \lambda_\star|$ so that $|\lambda_n - \lambda_\star| = \delta(\varepsilon_n) > \eta$ and $f(\lambda_n) \geq f(\lambda_\star) - \varepsilon_n$. Since X is compact, we can find a convergent subsequence $\{\lambda_{n_i}\}_i$, that converges to a limit, say, $\lambda_\circ \in X$. Observe that $|\lambda_\circ - \lambda_\star| \geq \eta > 0$ and $f(\lambda_\circ) = f(\lambda_\star)$, which contradicts our assumption that f has a unique maximum at $\lambda_\star$. □

From Eqs. (48) and (49), we infer that

$$F(\lambda_{\text{opt}}) - F(\lambda'_{\text{opt}}) < \varepsilon \;\Rightarrow\; |\lambda'_{\text{opt}} - \lambda_{\text{opt}}| < \delta_F(\varepsilon), \tag{50}$$

for some function $\delta_F(\varepsilon)$ that is well defined, monotonically non-increasing, and such that $\lim_{\varepsilon \searrow 0} \delta_F(\varepsilon) = 0$. This guarantees that Algorithm 1 computes an approximate optimizer λ'_{opt} that can be made arbitrarily close to the exact one λ_{opt} by improving the accuracy of root finding algorithms used in computing pairwise intersections and maxima of the functions $\{f_\alpha\}_\alpha$.

3.4.2 Effect of approximate root finding in Algorithm 2

In the context of Algorithm 2, approximate root finding translates to adding small perturbations to the intersections $\{\lambda_s\}_{s=0}^N$ and to the set of maxima $\{\lambda_\alpha^{\max}\}_{\alpha=1}^n$. It is clear that the algorithm is meaningful only if the errors in computing $\{\lambda_s\}_{s=0}^N$ do not alter the identification of the active functions defining F. Owing to continuity of the functions $\{f_\alpha\}_\alpha$, small perturbations to $\{\lambda_s\}_{s=0}^N$ result in small perturbations to F. Then Proposition 10 ensures that the consequent changes in λ_{opt} remain small. This, roughly speaking, is why we anticipate the algorithm to be stable to small approximations in computing intersections and maxima. Although detailed arguments remain to be worked out, it is instructive to consider an example.

Let $\lambda_{\alpha_s}^{\max} \in [\lambda_{s-1}, \lambda_s]$ so that $\lambda_{\text{opt}} = \lambda_{\alpha_s}^{\max}$ by virtue of Proposition 6. Suppose that the correct value of $\lambda_{\alpha_s}^{\max}$ is replaced by the approximate one $\hat{\lambda}_{\alpha_s}^{\max} = \lambda_{\alpha_s}^{\max} + \delta$ for some small δ. Denoting the optimizer returned by the algorithm as $\hat{\lambda}_{\text{opt}}$, let us examine the difference $|\lambda_{\text{opt}} - \hat{\lambda}_{\text{opt}}|$. Without loss of generality, we assume that the algorithm does not terminate during the first $(s-1)$ iterations.

(i) If $\hat{\lambda}_{\alpha_s}^{\max} \in [\lambda_{s-1}, \lambda_s]$, then $\hat{\lambda}_{\text{opt}} = \hat{\lambda}_{\alpha_s}^{\max}$ and hence

$$|\lambda_{\text{opt}} - \hat{\lambda}_{\text{opt}}| = |\lambda_{\alpha_s}^{\max} - \hat{\lambda}_{\alpha_s}^{\max}| = \delta.$$

(ii) If $\hat{\lambda}_{\alpha_s}^{\max} < \lambda_{s-1}$ and $f_{\alpha_s}(\lambda_{s-1}) > f_{\alpha_s}(\lambda_s)$, the algorithm terminates during the s^{th} iteration at Step 14 and returns $\hat{\lambda}_{\text{opt}} = \lambda_{s-1}$. Then

$$|\lambda_{\text{opt}} - \hat{\lambda}_{\text{opt}}| = |\lambda_{\alpha_s}^{\max} - \lambda_{s-1}| < |\lambda_{\alpha_s}^{\max} - \hat{\lambda}_{\alpha_s}^{\max}| = \delta.$$

(iii) If $\hat{\lambda}_{\alpha_s}^{\max} < \lambda_{s-1}$ and $f_{\alpha_s}(\lambda_{s-1}) \leq f_{\alpha_s}(\lambda_s)$, then the algorithm terminates during the $(s+1)^{\text{th}}$ iteration at Step 14 (see corollary 1) and returns $\hat{\lambda}_{\text{opt}} = \lambda_s$. Setting $y = f_{\alpha_s}(\lambda_{\alpha_s}^{\max})$ and $\varepsilon = f_{\alpha_s}(\lambda_{\alpha_s}^{\max}) - f_{\alpha_s}(\lambda_{s-1})$, notice that $\lambda_{s-1} \in \mathrm{L}_{y-\varepsilon}^{+}(f_{\alpha_s})$ by definition of ε and that $\lambda_s \in \mathrm{L}_{y-\varepsilon}^{+}(f_{\alpha_s})$ because $f_{\alpha_s}(\lambda_s) \geq f_{\alpha_s}(\lambda_{s-1})$. Proposition 10 therefore shows that $\lambda_s - \lambda_{s-1} < \delta_F(\varepsilon)$. We now have

$$\begin{aligned} |\lambda_{\text{opt}} - \hat{\lambda}_{\text{opt}}| &= |\lambda_{\alpha_s}^{\max} - \lambda_s| \\ &\leq \underbrace{|\lambda_{\alpha_s}^{\max} - \hat{\lambda}_{\alpha_s}^{\max}|}_{=\delta} + \underbrace{|\hat{\lambda}_{\alpha_s}^{\max} - \lambda_{s-1}|}_{\leq |\hat{\lambda}_{\alpha_s}^{\max} - \lambda_{\alpha_s}^{\max}| = \delta} + \underbrace{|\lambda_{s-1} - \lambda_s|}_{\leq \delta_F(\varepsilon)} \leq 2\delta + \delta_F(\varepsilon). \end{aligned} \tag{51}$$

In Eq. (51), $\delta_F(\varepsilon)$ is small because

$$\lambda_{\alpha_s}^{\max} - \lambda_{s-1} < \lambda_{\alpha_s}^{\max} - \hat{\lambda}_{\alpha}^{\max} = \delta,$$

which together with continuity of f_{α_s}, implies that ε is small.

(iv) If $\hat{\lambda}_{\alpha_s}^{\max} > \lambda_s$ and $f_{\alpha_s}(\lambda_{s-1}) < f_{\alpha_s}(\lambda_s)$, then the algorithm terminates during the $(s+1)^{\text{th}}$ iteration at Step 14 and returns $\hat{\lambda}_{\text{opt}} = \lambda_s$. Hence

$$|\lambda_{\text{opt}} - \hat{\lambda}_{\text{opt}}| = |\lambda_{\alpha_s}^{\max} - \lambda_s| \leq |\lambda_{\alpha_s}^{\max} - \hat{\lambda}_{\alpha_s}^{\max}| = \delta.$$

(v) Finally, suppose that $\hat{\lambda}_{\alpha_s}^{\max} > \lambda_s$ and $f_{\alpha_s}(\lambda_{s-1}) > f_{\alpha_s}(\lambda_s)$. The the algorithm terminates during the s^{th} iteration at Step 14 and returns $\hat{\lambda}_{\text{opt}} = \lambda_{s-1}$. Setting $y = f_{\alpha_s}(\lambda_{\alpha_s}^{\max})$ and $\varepsilon = f_{\alpha_s}(\lambda_{\alpha_s}^{\max}) - f_{\alpha_s}(\lambda_s)$, we have $\lambda_s \in \mathrm{L}_{y-\varepsilon}^{+}(f_{\alpha_s})$ by definition of ε and $\lambda_{s-1} \in \mathrm{L}_{y-\varepsilon}^{+}(f_{\alpha_s})$ because $f_{\alpha_s}(\lambda_{s-1}) > f_{\alpha_s}(\lambda_s)$. Reasoning as we did in Eq. (51), we get

$$\begin{aligned} |\lambda_{\text{opt}} - \hat{\lambda}_{\text{opt}}| = |\lambda_{\alpha_s}^{\max} - \lambda_{s-1}| &\leq |\lambda_{\alpha_s}^{\max} - \hat{\lambda}_{\alpha_s}^{\max}| + |\hat{\lambda}_{\alpha_s}^{\max} - \lambda_s| + |\lambda_s - \lambda_{s-1}| \\ &\leq 2\delta + \delta_F(\varepsilon), \end{aligned}$$

where δ being small implies that ε is small and in turn that $\delta_F(\varepsilon)$ is small.

The above example substantiates our intuition on the stability of Algorithm 2 to small approximations in root finding, but also confirms that the claim is a nontrivial one requiring detailed arguments.

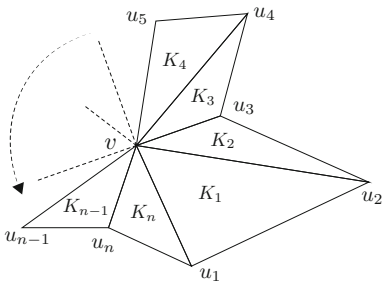


Fig. 2 Ring of triangles $\{K_\alpha\}_{\alpha=1}^n$ around a vertex v

4 Application to geometric mesh improvement

By way of demonstrating an application of Algorithms 1 and 2, and as a means of examining their performance, we consider an optimization-based method for improving qualities of unstructured meshes used in finite element simulations. The method, termed *directional vertex relaxation* (dvr) and introduced in [18], perturbs a specified set of vertices in a mesh along prescribed directions without altering element connectivities. The key ingredient in dvr is the sense of optimality defining vertex perturbations, which naturally conjures a univariate discrete max–min problem analogous to Eq. (1). Although applicable to general mesh types, we limit our explanations of dvr to the case of planar triangle meshes and numerical experiments to triangle and tetrahedral meshes for the sake of simplicity.

4.1 Optimal vertex perturbations in dvr

Referring to Fig. 2, consider planar triangles $K_1, K_2, \ldots, K_n$ sharing a common vertex v and label the vertices in triangle K_α by $\{v, u_\alpha, u_{\alpha+1}\}$ for $\alpha = 1, \ldots, n$. To remain consistent with the figure, we identify vertex u_{n+1} with u_1. Introduce a scalar-valued function Q called the *element quality metric*, that is defined over the set of planar triangles[5] and that serves to measure triangle qualities. A triangle K is therefore assigned a quality $Q(K)$. We adopt the convention that better- or more desirably-shaped triangles are assigned higher qualities by Q. The metric Q can be defined based on purely geometric criteria or in a data dependent manner using *a priori* or *a posteriori* error estimates.

Our goal is to perturb v along a prescribed direction $\mathbf{d}$ such that the qualities of triangles $\{K_\alpha\}_{\alpha=1}^n$ improve in a certain sense. To help formulate the condition defining the optimal perturbation for v, let us denote the triangle with vertices $\{v+\lambda\mathbf{d}, u_\alpha, u_{\alpha+1}\}$ by K_α^λ. Notice that K_α^λ is the result of perturbing vertex v in triangle K_α to the location $v + \lambda\mathbf{d}$. The optimal perturbation of v along the direction $\mathbf{d}$ is defined in dvr as

$$\lambda_{\text{opt}} \triangleq \arg\max_{\lambda\in\mathbb{R}} \min_{1\le\alpha\le n} Q(K_\alpha^\lambda). \tag{52}$$

[5] We only request that Q be defined over triangles whose vertices do not all coincide.

The identification $f_\alpha(\lambda) = Q(K_\alpha^\lambda)$ for $\alpha = 1, \ldots, n$, transforms Eq. (52) into

$$\lambda_{\text{opt}} = \arg\max_{\lambda \in \mathbb{R}} \min_{1 \le \alpha \le n} f_\alpha(\lambda), \tag{53}$$

which of course resembles Eq. (1). The rationale behind the vertex update $v \mapsto v + \lambda_{\text{opt}}\mathbf{d}$ is evident— it maximizes the poorest quality among the triangles around the perturbed vertex.

4.2 Mean ratio metric

Whether λ_{opt} as defined in Eq. (52) exists and is unique, depends on the choice of Q. In our numerical experiments here, we choose it to be the mean ratio metric [12], which assigns the the quality

$$Q(K) \triangleq \frac{4\sqrt{3}\,\mu(K)}{\sum_{1 \le p < q \le 3} \ell_{pq}^2(K)}, \tag{54}$$

to a triangle K with signed area $\mu(K)$, where the sign is decided using the right hand rule for example, and where $\ell_{pq}(K)$ is the length of the edge joining the p^{th} and q^{th} vertices. The normalizing factor $4\sqrt{3}$ is chosen so that equilateral triangles are assigned unit quality. By adopting Eq. (54), the goal of mesh improvement is then to render triangles that are as close to equilateral as possible. Properties of Q and its equivalences with other commonly used quality metrics can be found in [19,20]. We omit arguments demonstrating that if triangle K has strictly positive quality, then the function $\lambda \mapsto f(\lambda) = Q(K^\lambda)$ with Q chosen to be the mean ratio metric satisfies assumptions **A**0–**A**3. While feasibility, continuity and decay conditions are immediate, we mention that the map $v \mapsto Q(K)$ is strictly quasiconcave over circular upper level sets corresponding to strictly positive element qualities. Consequently, $\lambda \mapsto f(\lambda)$ is strictly quasiconcave irrespective of the choice of $\mathbf{d}$.

4.3 Roots, intersections and maxima

Since $v \mapsto \mu(K_\alpha)$ is an affine function, so is $\lambda \mapsto \mu(K_\alpha^\lambda)$. Similarly, the sum of squared edge lengths of K_α^λ is a quadratic function of λ. Hence the functions $\lambda \mapsto f_\alpha(\lambda)$ are rational polynomials. This has a host of useful consequences, foremost among which is the guarantee that the number of pairwise intersections of distinct functions in $\{f_\alpha\}_\alpha$ is finite. The requirement that Λ_{int} be a finite set in Algorithms 1 and 2 is therefore trivially satisfied. Besides, root finding in the algorithms is significantly simplified—computing level sets and intersections requires the solution of cubic polynomials while maximizers are roots of quadratics. Identifying distinct functions among $\{f_\alpha\}_\alpha$ becomes straightforward as well, requiring only a comparison of polynomial coefficients.

4.4 An example

Figure 3 shows an example of improving the qualities of triangles $\{K_\alpha\}_{\alpha=1}^{6}$ by perturbing their common vertex v. The poorest quality among the triangles shown in Fig. 3a is $Q(K_4) = 0.103$. Figure 3b shows the result of relocating v to v' following the Laplacian smoothing algorithm [21], wherein the Cartesian coordinates of v' is defined to be the arithmetic mean of the coordinates of vertices $u_1, \ldots, u_6$. Notice that v' computed this way coincides with vertex u_4, causing triangles K_3 and K_4 to collapse. It is well known that Laplacian smoothing does not always yield admissible meshes, much less guarantee mesh improvement. Nevertheless, it continues to be widely used because of its simplicity and ease of implementation.

Figure 3c shows the result of perturbing v to $\hat{v} = v + \lambda_{\text{opt}}\mathbf{d}$, where $\mathbf{d}$ is prescribed to be the unit vector oriented at 45° to the horizontal and λ_{opt} is computed as the

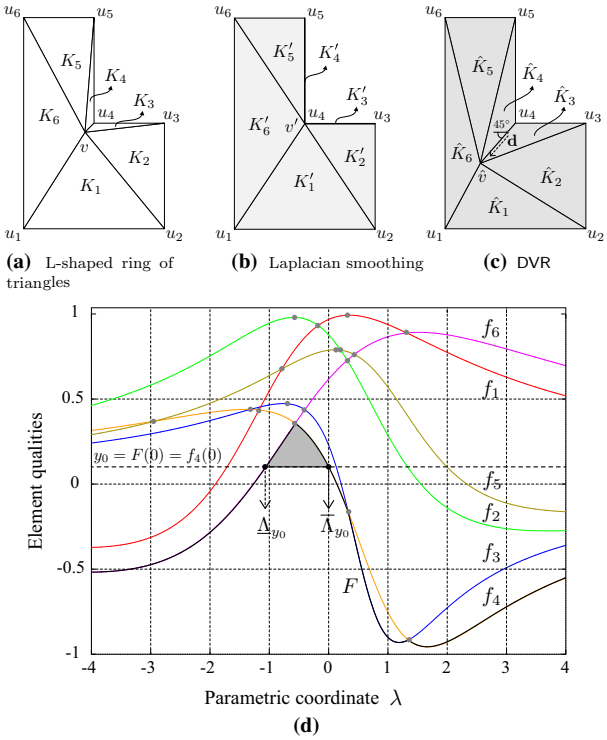


Fig. 3 Improving qualities of triangles in (**a**) by perturbing their common vertex v. Figure (**b**) shows that relocating v to v' using the Laplacian smoothing algorithm results in triangles K_3 and K_4 becoming degenerate. Optimizing the location of v along the direction $\mathbf{d}$ oriented at 45° to the horizontal by resolving Eq. (52) results in the improved mesh shown in (**c**). The poorest element quality around v' is improved to 0.357 from 0.103 around v. Details of the computation of λ_{opt} are shown in (**d**). Algorithm 1 exhaustively computes all maximizers and intersections indicated by gray dots and identifies λ_{opt} from among them. Algorithm 2 first computes the bounds $\underline{\Lambda}_{y_0} = -1.06$ and $\overline{\Lambda}_{y_0} = 0$, and then identifies the active functions defining F as f_4 over $[-0.563, 0]$ and f_6 over $[-1.06, -0.563]$. The optimizer is found to be the coordinate $\lambda_{\text{opt}} = -0.563$ where f_4 and f_6 intersect

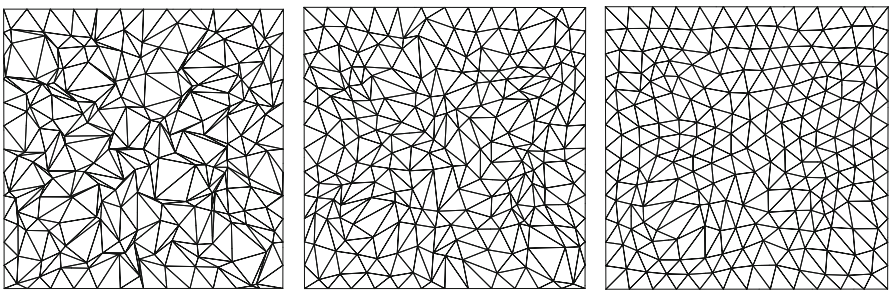
solution of Eq. (52). The minimum element quality among triangles sharing vertex $\hat{v}$ is improved to $Q(\hat{K}_4) = Q(\hat{K}_6) = 0.357$.

Figure 3d depicts the details involved in computing λ_{opt} using Algorithms 1 and 2. The figure shows the functions $\lambda \mapsto f_\alpha(\lambda)$ for $\alpha = 1, \ldots, 6$ and the objective $\lambda \mapsto F(\lambda)$ that needs to be maximized. Observe that although the functions $\{f_\alpha\}_\alpha$ are smooth, F is not. The maximizer of F is located at $\lambda_{\text{opt}} = -0.563$ where the functions f_4 and f_6 intersect. Algorithm 1 computes λ_{opt} by identifying all maximizers and pairwise intersections. Algorithm 2 on the other hand, first computes the interval $[\underline{\Lambda}_{y_0}, \overline{\Lambda}_{y_0}] = [-1.06, 0]$, which evidently brackets λ_{opt} quite well, and then reconstructs F within this interval while computing only a small fraction of the maximizers and intersections compared to Algorithm 1.

4.5 Mesh improvement

Supplementing the example in Fig. 3, the numerical experiment discussed next consists in iteratively perturbing an entire collection of vertices in a mesh to improve the qualities of its elements. This examples serves to highlight the improvement in mesh quality possible by sequentially repeating the procedure followed in Fig. 3 at the given set of vertices.

Figure 4a shows a distorted mesh of triangles over a unit square. The poorest element quality in the mesh is 0.0219. We seek to improve the mesh by perturbing its interior vertices, while leaving the ones along the boundary undisturbed. Each iteration in the dvr algorithm then consists in sequentially relaxing the interior vertices, with each vertex perturbation computed by resolving a problem of type (52). For lack of better alternatives, vertices are enumerated according to the node numbers assigned by the mesh generator, and relaxed along the horizontal and vertical directions during odd and even iterations, respectively. Figure 4b, c reveal the dramatic improvement achieved within a few iterations of the dvr algorithm. The minimum element quality in the mesh improves to 0.424 after 2 iterations and to 0.652 after 6 iterations. The minimum element quality continues to improve monotonically with each iteration,



(a) Input mesh **(b)** 2 iterations **(c)** 6 iterations

Fig. 4 Mesh improvement with dvr. Figure (**a**) shows a distorted mesh over a unit square. Figures (**b**) and (**c**) reveal that iteratively relaxing the interior vertices results in dramatic improvement in element qualities

reaching a value of 0.815 at 20 iterations and converges approximately to the value 0.896 beyond about 30 iterations.

We conclude this example mentioning that the mesh improvement evident in this example is not a coincidence, but is in fact guaranteed irrespective of the choice of relaxation directions and the order in which vertices are relaxed. Monotonic improvement of the poorest element quality is particularly significant in finite element simulations, where the conditioning of linear systems of equations to be resolved depends strongly on the minimum element quality in the underlying mesh [22]. We refer to [18] for more details on dvr, its analysis, the guarantees for mesh improvement it provides, and for its representative application to simulating moving boundary problems.

4.6 Performance of Algorithms 1 and 2—a case study

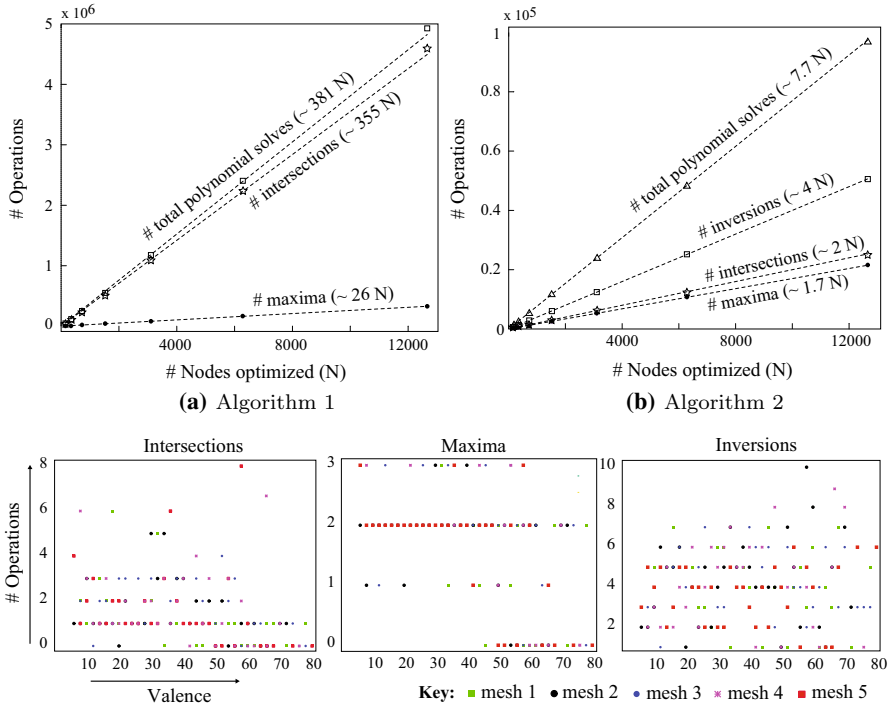
In this third example, we consider improving qualities of elements in tetrahedral meshes using dvr. Vertices in such meshes have reasonably large valencies (number of elements incident at a vertex), at least when compared to those in planar triangle meshes. For instance, valencies in the meshes used in the experiments discussed here range from under 10 to over 70. These examples therefore help us to examine and compare the performance of Algorithms 1 and 2 for a correspondingly large range of n in problem (4).

We adopt the mean ratio metric for defining qualities of tetrahedra as well [23]. Analogous to Eq. (54), the quality of a tetrahedron K is given by

$$Q(K) \triangleq \frac{12\sqrt[3]{9}\,\mathrm{sign}(\mu(K))\,|\mu(K)|^{2/3}}{\sum_{1\le p<q\le 4}\ell_{pq}^2(K)}, \tag{55}$$

where the constant $12\sqrt[3]{9}$ ensures that regular tetrahedra are assigned unit quality, $\mu(K)$ is the signed volume of K and $\ell_{pq}(K)$ is the length of the edge joining its p^{th} and q^{th} vertices. Arguing along the same lines as the two dimensional case, it is possible to show that if K has positive quality, then $\lambda \mapsto Q(K^\lambda)$ satisfies assumptions **A**0–**A**3. The map $\lambda \mapsto Q(K^\lambda)$ is of the form $a(\lambda)^{2/3}/b(\lambda)$, where $\lambda \mapsto a(\lambda)$ is affine and $\lambda \mapsto b(\lambda)$ is quadratic. In particular, $Q^3(K)$ is a rational polynomial. Therefore distinct quality curves intersect at finitely many points, as required in Algorithms 1 and 2. Furthermore, maxima, intersections and level sets of element quality curves can be computed as roots of polynomials.

The experiment discussed here consists in visiting all interior vertices in a tetrahedral mesh, computing the optimal perturbation of each vertex along the direction $\mathbf{d} = (1, 0, 0)$ by resolving Eq. (52), and recording the number of maxima, intersections and function inversions required. To this end, we construct five tetrahedral meshes with progressively larger number of vertices. Each mesh is the Delaunay triangulation of a randomly generated set of points [24], and is explicitly checked to contain no degen-



(a) Algorithm 1 **(b)** Algorithm 2

(c) Operation count for Algorithm 2

Fig. 5 A comparison of the performance of Algorithms 1 and 2 for optimizing the locations of interior vertices in five tetrahedral meshes with randomly distributed vertices. Vertices in these meshes have widely varying valencies. However, the operation count for Algorithm 1 reported in (**a**) corresponds to the valence $n \approx 26$. Then, comparing figures (**a**) and (**b**) reveals that Algorithm 1 requires the resolution of approximately 381 polynomials to compute λ_{opt}, while Algorithm 2 requires fewer than 8. The plots in (**c**) reveal that the number of operations required in Algorithm 2 for a range of n remains relatively small

erate elements. Each interior vertex in such a mesh therefore furnishes a problem of type Eq. (52), with n corresponding to the valence of the vertex[6].

Figure 5a, b report the total number of maxima, intersections and function inversions required in the five meshes when using Algorithms 1 and 2, respectively. Figure 5a shows that Algorithm 1 computes approximately $26N$ maxima and $355N$ intersections to optimize the locations of N vertices. Despite the widely varying vertex valencies, by coincidence perhaps, Fig. 5a reflects the operation count of Algorithm 1 corresponding to an "average valence" of $n \approx 26$.

Contrasting Fig. 5a with the data for Algorithm 2 in Fig. 5b reveals the drastic reduction in the number of computations required—identifying λ_{opt} using Algorithm 2 with $n \approx 26$ roughly requires the resolution of only 8 polynomials compared to 381 in Algorithm 1. Figure 5c provides a representative snapshot of the computations in Algorithm 2 for specific values of n realized at vertices with corresponding valencies,

[6] In general, n is less than or equal to the valence of a vertex since the quality curves of two or more elements may coincide. For the representative data in Fig. 5c, n equals the vertex valence.

in each of the five meshes. The plot shows that the operation count in the algorithm does not appear to increase with n in any significant way. The bound $\underline{\Lambda}_{y_0} \leq \lambda_{\text{opt}} \leq \overline{\Lambda}_{y_0}$ plays a critical role in relieving Algorithm 2 from many of the unnecessary calculations performed in Algorithm 1. These bounds also help to conveniently circumvent the need for any geometrically motivated restrictions on the magnitude of perturbations commonly used in mesh smoothing methods, cf. [13].

It should be noted that the comparison in Fig. 5 does not reflect the number of function evaluations required in the two algorithms. Unlike Algorithm 2, Algorithm 1 requires a large number of function evaluations since the objective F is evaluated at each point in the set $\Lambda_{\text{int}} \cup \Lambda_{\text{max}}$, which in turn requires computing each of the n functions in $\{f_\alpha\}_\alpha$ at these points. In fact, profiling our implementation of Algorithm 1 reveals that close to half the execution time is consumed in evaluating F.

The exact details of our implementation of CALCBOUNDS, CALCMAXIMIZER and FINDACTIVEFUNC are not crucial for interpreting the data in Fig. 5 for Algorithm 2. We mention that checking necessary conditions for existence of roots (within relevant intervals) using des Cartes rule of signs [25] in the routines CALCBOUNDS and FINDACTIVEFUNC helps to avoid many needless polynomial resolutions in the algorithm. We did not employ such checks in CALCMAXIMIZER since maximizers are just roots of quadratic polynomials. Polynomial roots themselves are computed as eigenvalues of a companion matrix [26,27]. Failure of Algorithm 2, though very rare, is limited to nonconvergence of QR factorizations required in eigenvalue computations. A simple rescaling of the polynomial equations improves the conditioning of the companion matrix and renders the algorithm more reliable overall.

We recognize that it is unwise to draw general conclusions about the performance of Algorithm 2 from Fig. 5; the operation count in the algorithm necessarily depends on the data (i.e., the functions $\{f_\alpha\}_\alpha$). Nevertheless, this case study provides encouraging evidence that the algorithm is a useful and computationally efficient tool in the challenging context of geometric mesh improvement.

5 Concluding remarks

We hope that the analysis and algorithms discussed here will benefit a general class of problems where max–min/min–max criteria and quasiconvex/quasiconcave functions appear naturally [15]. At least in the univariate case, Algorithm 2 provides a compelling alternative to ad hoc solution strategies often resorted to in practice. For the algorithm to be adopted in practical engineering applications, a detailed analysis of the effect of tolerances on its accuracy, which was only briefly discussed in Sect. 3.4, will be important. It may also be possible to improve the algorithm, for instance, by identifying initial guesses closer to the optimizer (choosing $y > y_0$) or by discarding inactive functions from the computations altogether. We intend to pursue applications of this work to problems in geometric mesh optimization, a topic replete with heuristic methods lacking meaningful guarantees and with justified concerns about computational costs.

References

1. Cherkaev, E., Cherkaev, A.: Minimax optimization problem of structural design. Comput. Struct. **86**(13), 1426–1435 (2008)
2. Du, D., Pardalos, P.: Minimax and Applications, vol. 4. Springer, Berlin (2013)
3. Ke, Q., Kanade, T.: Quasiconvex optimization for robust geometric reconstruction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(10), 1834–1847 (2007)
4. Demyanov, V., Malozemov, V.: Introduction to Minimax. Dover Publications (1990).
5. Polyak, R.: Smooth optimization methods for minimax problems. SIAM J. Control Optim. **26**(6), 1274–1286 (1988)
6. Xu, S.: Smoothing method for minimax problems. Comput. Optim. Appl. **20**(3), 267–279 (2001)
7. Sotiropoulos, D.: Solving discrete minimax problems using interval arithmetic. Pac. J. Optim. **2**(2), 241–259 (2006)
8. Wolfe, M.: On discrete minimax problems in R using interval arithmetic. Reliab. Comput. **5**(4), 371–383 (1999)
9. Dari, E., Buscaglia, G.: Mesh optimization: how to obtain good unstructured 3D finite element meshes with not-so-good mesh generators. Struct. Optim. **8**(2–3), 181–188 (1994)
10. Freitag, L., Jones, M., Plassmann, P.: An efficient parallel algorithm for mesh smoothing. In: Proceedings of the 4th International Meshing Roundtable, pp. 47–58 (1995)
11. Zavattieri, P., Dari, E., Buscaglia, G.: Optimization strategies in unstructured mesh generation. Int. J. Numer. Methods Eng. **39**(12), 2055–2071 (1996)
12. Bank, R., Smith, R.: Mesh smoothing using a posteriori error estimates. SIAM J. Numer. Anal. **34**(3), 979–997 (1997)
13. Alliez, P., Ucelli, G., Gotsman, C., Attene, M.: Recent advances in remeshing of surfaces. In: De Floriani, L., Spangnuolo, M. (eds.) Shape Analysis and Structuring, pp. 53–82. Springer, Berlin (2008)
14. Amenta, N., Bern, M., Eppstein, D.: Optimal point placement for mesh smoothing. J. of Algorithms **30**(2), 302–322 (1999)
15. Eppstein, D.: Quasiconvex programming. Comb. Comput. Geom. **52**, 287–331 (2005)
16. Ansari, Q., Lalitha, C., Mehta, M.: Generalized Convexity, Nonsmooth Variational Inequalities, and Nonsmooth Optimization. CRC Press, Boca Raton (2013)
17. Greenberg, H., Pierskalla, W.: A review of quasi-convex functions. Oper. Res. **19**(7), 1553–1570 (1971)
18. Rangarajan, R., Lew, A.: Provably robust directional vertex relaxation for geometric mesh optimization (submitted, 2016)
19. Knupp, P.: Algebraic mesh quality metrics. SIAM J. Sci. Comput. **23**(1), 193–218 (2001)
20. Pébay, P., Baker, T.: Analysis of triangle quality measures. Math. Comput. **72**(244), 1817–1839 (2003)
21. Frey, W., Field, D.: Mesh relaxation: a new technique for improving triangulations. Int. J. Numer. Methods Eng. **31**(6), 1121–1133 (1991)
22. Du, Q., Wang, D., Zhu, L.: On mesh geometry and stiffness matrix conditioning for general finite element spaces. SIAM J. Numer. Anal. **47**(2), 1421–1444 (2009)
23. Liu, A., Joe, B.: On the shape of tetrahedra from bisection. Math. Comput. **63**(207), 141–154 (1994)
24. Barber, B., Dobkin, D., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**(4), 469–483 (1996)
25. Kobel, A., Rouillier, F., Sagraloff, M.: Computing real roots of real polynomials... and now for real! arXiv preprint arXiv:1605.00410 (2016)
26. Edelman, A., Murakami, H.: Polynomial roots from companion matrix eigenvalues. Math. Comput. **64**(210), 763–776 (1995)
27. Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., Rossi, F.: GNU scientific library reference manual. *URL* http://www.gnu.org/software/gsl, version 1 (2010)