FINDING SADDLE POINTS OF MOUNTAIN PASS TYPE WITH QUADRATIC MODELS ON AFFINE SPACES

C. H. JEFFREY PANG

ABSTRACT. The problem of computing saddle points is important in certain problems in numerical partial differential equations and computational chemistry, and is often solved numerically by a minimization problem over a set of mountain passes. We propose an algorithm to find saddle points of mountain pass type to find the bottlenecks of optimal mountain passes. The key step is to minimize the distance between level sets by using quadratic models on affine spaces similar to the strategy in the conjugate gradient algorithm. We discuss parameter choices, convergence results, and how to augment the algorithm to a path based method. Finally, we perform numerical experiments to test the convergence of our algorithm.

CONTENTS

1. INTRODUCTION

We begin with the definition of a mountain pass.

Definition 1.1. Let X be a topological space, and consider $a,b \in X$. For a function $f : X \to \mathbb{R}$, define an *optimal mountain pass* $\bar{p} \in \Gamma(a,b)$ to be a minimizer of the problem

$$\inf_{p\in\Gamma(a,b)} \sup_{0\le t\le 1} f\circ p(t). \tag{1}$$

Here, $\Gamma(a,b)$ is the set of continuous paths $p : [0,1] \to X$ such that $p(0) = a$ and $p(1) = b$.

The point $\bar{x}$ is a *critical point* if $\nabla f(\bar{x}) = 0$, and the critical point $\bar{x}$ is a *saddle point* if it is not a local maximizer or minimizer on X. The value $f(x)$ is a *critical value* if x is a critical point. We say that $\bar{x}$ is a *saddle point of mountain pass type* if there is an open set U containing $\bar{x}$ such that $\bar{x}$ lies in the closure of two path connected components of $\{x \in U : f(x) < f(\bar{x})\}$. In the case where f is smooth and an optimal mountain pass $\bar{p} : [0,1] \to X$ exists, the maximum of f on $\bar{p}([0,1])$ is a saddle point.

The problem of finding saddle points numerically is important in the problem of finding weak solutions to partial differential equations numerically. Some of the theoretical references include

Date: July 22, 2011.

Key words and phrases. Mountain pass algorithm, conjugate gradient, convergence, Krylov subspace, indefinite matrix.

[15, 20, 21, 22, 25]. See also the more accessible reference [11]. The original paper of a mountain pass algorithm to solve partial differential equations is [4], and it contains several semilinear elliptic problems. Particular applications in numerical partial differential equations include finding periodic solutions of a boundary value problem modeling a suspension bridge [6] (introduced by [12]), studying a system of Ginzburg-Landau type equations arising in the thin film model of superconductivity [7], the choreographical 3-body problem [2], and cylinder buckling [10]. Other notable works in computing saddle points for solving numerical partial differential equations include the use of constrained optimization [9], extending the mountain pass algorithm to find saddle points of higher Morse index [5, 14] (See also the theoretical foundations in [19]), extending the mountain pass algorithm to find nonsmooth saddle points [26], and the exploitation of symmetry [23, 24].

The problem of finding saddle points numerically is by now well entrenched in the chemistry curriculum. In transition state theory, the problem of finding the least amount of energy to transition between two stable states is equivalent to finding an optimal mountain pass between these two stable states. The highest point on the optimal mountain pass can then be used to determine the reaction kinetics. The foundations of transition state theory was laid by Marcelin, and important work by Eyring and Polanyi in 1931 and by Pelzer and Wigner a year later established the importance of saddle points in transition state theory. We cite the Wikipedia entry on transition state theory for more on its history and further references. Numerous methods for computing saddle points were suggested through the years, and we refer to [8] for a survey. A software for computing saddle points in chemistry is Gaussian[1]. Tools for computing transition states[2] are also included in VASP[3]. Though the entire optimal mountain pass is needed for such an application, the process of computing saddle points often gives hints on an optimal mountain pass.

As mentioned in [13], our initial interest in the problem of computing saddle points of mountain pass type comes from computing the distance of a matrix $A \in \mathbb{C}^{n\times n}$ to the closest matrix with repeated eigenvalues (also known as the Wilkinson distance problem).

Many of the prevailing methods of finding an optimal mountain pass make use of the formulation (1) directly and discretize a path in $\Gamma(a,b)$. See [8, 17] for example. This discretized path is perturbed so that the maximum value of f along the path is reduced. The proof of the celebrated mountain pass theorem [1] (which establishes the existence of saddle points under some added conditions) shows that such a strategy allows one to find a saddle point.

We recall the classical theory of numerical optimization to get some ideas on how to design algorithms for the mountain pass problem. Global optimization is provably difficult without additional assumptions, so one looks at the local theory. Optimization algorithms are then judged based on how well they perform once the iterates get close to the minimizer. The global mountain pass problem is also provably difficult, so once again we look at local methods. For a local theory of the mountain pass problem, observe that the saddle points of mountain pass type can be seen as the bottlenecks at which an optimal mountain pass has to pass through. The process of identifying such bottlenecks can then give clues to how an optimal mountain pass can be constructed. Algorithms for finding saddle points of mountain pass type can therefore be judge based on how well they perform once they get close to the saddle point.

For a value $l \in \mathbb{R}$, we say that $\{x \in X \mid f(x) \leq l\}$ is a *level set*. The idea of using level sets to establish lower bounds for critical values and to successively find the closest points in the level sets to estimate the saddle point was proposed in [16] and revisited in [13] (written without knowledge of [16]). Suppose $\{l_i\}$ is an increasing sequence and the sequences of points $\{x_i\}$ and $\{y_i\}$ are such that x_i and y_i lie in different components of the level set $\{x \mid f(x) \leq l_i\}$. If the sequences $\{x_i\}$ and $\{y_i\}$ have a common limit, then this common limit is a critical point. Using level sets has several computational advantages. Only two points are needed at any time

[1] http://www.gaussian.com/

[2] http://theory.cm.utexas.edu/vtsttools/neb/

[3] http://cms.mpi.univie.ac.at/vasp/vasp/vasp.html

during the computations instead of a discretized path. Much of the computational effort is then performed near the saddle point, which can be seen as a bottleneck that all optimal mountain passes must pass. The distance $\|x_i - y_i\|$ gives an indication of the algorithm progress. Lastly, it was proven in [13] that, provided black boxes for finding closest points to components of the level set and for the minimization of the function f on an affine space exist, we have local superlinear convergence to the saddle point. Figure 1 contrasts the two strategies for finding saddle points of mountain pass type.

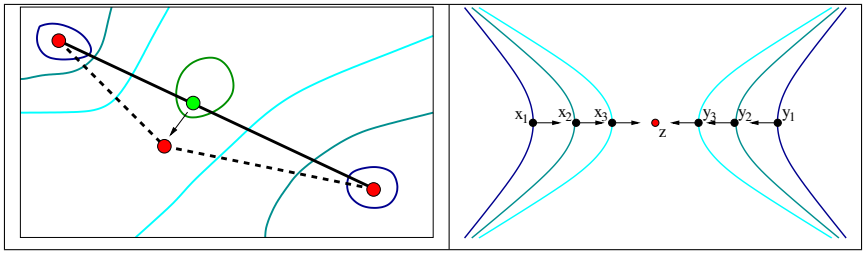

FIGURE 1. The diagram on the left shows the classical method of perturbing paths for the mountain pass problem, while the diagram on the right shows convergence to the critical point by looking at level sets, as was done in [13] and this paper.

Another technique we borrow from optimization theory is to make use of the fact that the function f has a quadratic approximation at where it is smooth, and in particular at the critical points. The quadratic approximation is the basis on which Newton methods, quasi-Newton methods and the conjugate gradient algorithms are derived. All known algorithms achieving fast convergence (i.e., quadratic, superlinear, or linear convergence) are one of the above-mentioned algorithms, trying to find x such that $\nabla f(x) = 0$ by solving the associated linear system obtained from the quadratic approximation. Any algorithm that can converge fast to the saddle point should be similar in some way to the above-mentioned algorithms.

The analysis in [13] uses the following approximation of f at the saddle point $\bar{x}$:

$$f(x) = f(\bar{x}) + (x-\bar{x})^T H(\bar{x})(x-\bar{x}) + o(\|x-\bar{x}\|^2). \tag{2}$$

To simplify our analysis, we assume that $X = \mathbb{R}^n$ throughout so that $f : \mathbb{R}^n \to \mathbb{R}$. A common assumption in finding saddle points of mountain pass type is that of nondegeneracy. The saddle point $\bar{x}$ is said to be *nondegenerate* if the Hessian $H(\bar{x})$ is nonsingular. The more restrictive $\mathscr{C}^{2+}$ condition, equivalent to the Hessian mapping $H : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ being locally Lipschitz at $\bar{x}$, can be realistically assumed for many practical problems.

The smoothness of f at a critical point $\bar{x}$ gives the approximation (2), and a similar approximation can be written for the gradient ∇f. For the analysis in this paper, we concentrate on the theory of finding saddle points in the case where the Hessian $H(\cdot)$ is constant. This is equivalent to assuming that $f : \mathbb{R}^n \to \mathbb{R}$ is defined by $f(x) = \frac{1}{2}x^T Hx + g^T x + c$ and ignoring the higher order terms, which is analogous to the textbook analysis of the steepest descent and conjugate gradient algorithms for optimization in quadratic problems. These assumptions simplify much of the analysis and brings out the main ideas of how an iterative method can find the saddle point with preferably way fewer than n iterations. Since the textbook analysis of conjugate gradient algorithms discusses only the exact quadratic case and not the smooth case, we shall only analyze the exact quadratic in this paper. There are some parallels between the saddle point problem surveyed in [3] and the material in this paper.

The following fact about saddle points of mountain pass type will also be used throughout. The *Morse index* of a nondegerate critical point is the number of negative eigenvalues of its Hessian.

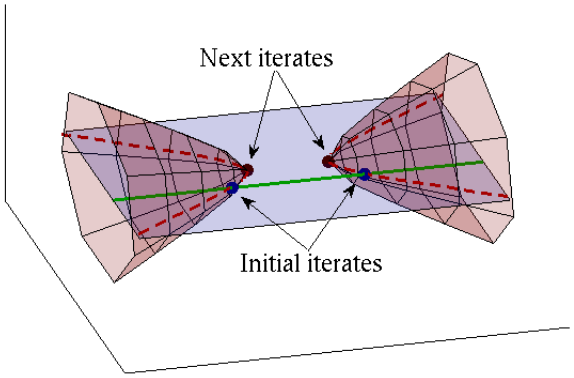

FIGURE 2. Consider $f:\mathbb{R}^3\to\mathbb{R}$ defined by $f(x)=-x_1^2+x_2^2+x_3^2$, which has a critical point at 0 and critical level 0. Let $l<0$. In our algorithm, we first find the closest points of the two components of $\{x:f(x)\le l\}$ on a line. Then through information obtained from the gradients and function values, we approximate the behavior of f on a larger affine space and find pairs of points closer to each other in the respective components. This process continues until we found points sufficiently close to each other.

Fact 1.2. *(Morse index one) Suppose $f:\mathbb{R}^n\to\mathbb{R}$ is $\mathscr{C}^2$ and $\bar{x}$ is a saddle point of mountain pass type. If the Hessian $H(\bar{x})\in\mathbb{R}^{n\times n}$ is invertible, then $H(\bar{x})$ has Morse index one. That is, the Hessian has $n-1$ positive eigenvalues and one negative eigenvalue.*

The main strategy in this paper to find saddle points of mountain pass type is as follows. Assume that f has a quadratic approximation near the saddle point $\bar{x}$. Let l_i be a lower bound on the critical value $f(\bar{x})$. Through evaluations of f near $\bar{x}$, we can find the behavior of f on successively larger affine spaces. Such a strategy is analogous to the conjugate gradient algorithm. From these better estimates, we can approximately find the closest pair of points in the respective components of the level set $\{x:f(x)\le l_i\}$. This procedure addresses a difficulty in [13], and is illustrated in Figure 2 and elaborated in Sections 3 and 4 in particular. We can increase the level l_i till l_i is sufficiently close to the critical level $f(\bar{x})$, and thus find the critical point $\bar{x}$.

We outline the sections in this paper. After building the needed background on quadratic models in Section 2, we propose an algorithm in Section 3 to find saddle points of mountain pass type using quadratic approximations on affine spaces of the level set. Sections 4 and 5 explain the choices of d_2 and l_i in the algorithms in Section 3. In Section 6, we explain briefly how our algorithm can be augmented into a path-based algorithm. We prove some convergence results of our methods in Section 7, and show how our algorithms perform for quadratic problems in our numerical experiments in Section 8. These numerical experiments give an indication of how the algorithm can perform once it gets close to the saddle point.

1.1. **Notation.** We denote the set of symmetric matrices in $\mathbb{R}^{n\times n}$ by S^n. The *lineality space* of an affine space A passing through a point z is equal to the subspace $A-z$.

2. QUADRATIC MODELS

For $f:\mathbb{R}^n\to\mathbb{R}$, the second order Taylor approximation motivates the quadratic model $\frac{1}{2}x^THx+g^Tx+c$ to describe the behavior of f near a critical point $\bar{x}$. This section discusses issues related to quadratic models.

We begin with the following elementary result. We say that $f:\mathbb{R}^n\to\mathbb{R}$ is a *quadratic with unknown parameters* if $f(x)=\frac{1}{2}x^THx+g^Tx+c$ for unknown parameters $H\in S^n$, $g\in\mathbb{R}^n$ and $c\in\mathbb{R}$. We say that $d+1$ points are in general position if the affine space containing these points has dimension d.

Proposition 2.1. *(Determining quadratic models) Suppose $f:\mathbb{R}^n\to\mathbb{R}$ is an exact quadratic with unknown parameters. Suppose $L\in\mathbb{R}^{n\times d}$ has linearly independent columns. To determine $\tilde{H}\in S^d$, $\tilde{g}\in\mathbb{R}^d$ and $\tilde{c}\in\mathbb{R}$ such that*

$$\tilde{f}(v):=\frac{1}{2}v^T\tilde{H}v+\tilde{g}^Tv+\tilde{c}=f(Lv+x)\text{ for all }v\in\mathbb{R}^d,$$

we need the value $\tilde{f}(v_i)$ and the gradient $\nabla\tilde{f}(v_i)$ for d points $v_1,\dots,v_d$ in $\mathbb{R}^d$ and $\tilde{f}(v_{d+1})$, where $\{v_1,\dots,v_d,v_{d+1}\}$ are in general position.

Proof. The problem is equivalent to determining $\hat{H}$, $\hat{g}$ and $\hat{c}$ such that

$$\tilde{f}(v)=\frac{1}{2}(v-v_1)^T\hat{H}(v-v_1)+\hat{g}^T(v-v_1)+\hat{c}.$$

The gradient of $\tilde{f}$ is $\nabla f(v)=\hat{H}(v-v_1)+\hat{g}$. Clearly $\hat{c}=\tilde{f}(v_1)=f(Lv_1+x)$ and $\hat{g}=\nabla\tilde{f}(v_1)=L^T\nabla f(Lv+x)$.

With an orthogonal transformation, we can assume that the span of $\{v_2-v_1,v_3-v_1,\dots,v_d-v_1\}$ is equal to the span of the first $d-1$ elementary vectors. Through

$$\hat{H}(v_i-v_1)=\nabla\tilde{f}(v_i)-\nabla\tilde{f}(v_1)=L^T[\nabla f(Lv_i+x)-\nabla f(Lv_1+x)]\text{ for }i=2,\dots d,$$

we can determine $\hat{H}_{i,j}$ for $1\le i\le d-1$ and $1\le j\le d$. Through the symmetry of H, we can determine all entries of $\hat{H}$ except for $\hat{H}_{d,d}$. Since the points $\{v_1,\dots,v_{d+1}\}$ are in general position, $v_{d+1}-v_1$ must have a nonzero d-th component. With

$$\tilde{f}(v_{d+1})=\frac{1}{2}(v_{d+1}-v_1)\hat{H}(v_{d+1}-v_1)+\hat{g}^T(v_{d+1}-v_1)+\hat{c},$$

we can determine the value of $\hat{H}_{d,d}$. □

The next result describes the behavior of the level sets of a quadratic near a saddle point.

Proposition 2.2. *(Optimality in quadratic models) Suppose $f:\mathbb{R}^n\to\mathbb{R}$ is defined by $f(x)=\frac{1}{2}x^THx+g^Tx+c$, where H has eigenvalues $\{\lambda_i\}_{i=1}^n$ arranged in decreasing order, with $\lambda_i>0$ for $1\le i\le n-1$ and $\lambda_n<0$.*

1. *The critical point of f is $-H^{-1}g$, and the critical level of f is $c-\frac{1}{2}g^TH^{-1}g$.*
2. *For a level $l<c-\frac{1}{2}g^TH^{-1}g$, the level set $\{x'\in\mathbb{R}^n:f(x')\le l\}$ consists of two convex components. The points $\tilde{x}$ and $\tilde{y}$ defined by $-H^{-1}g\pm\sqrt{\frac{2l-[2c-g^TH^{-1}g]}{\lambda_n}}q_n$, where q_n is the eigenvector of unit length corresponding to the negative eigenvalue of H, are the minimizers of*

$$\min_{x,y}\|x-y\|$$
$$\text{s.t. } x\text{ and }y\text{ are in different components of }\{x'\in\mathbb{R}^n:f(x')\le l\}.$$

Proof. Part (1) follows by noticing that $\nabla f(x)=Hx+g$ and writing $f(x)$ as

$$f(x)=\frac{1}{2}(x+H^{-1}g)^TH(x+H^{-1}g)+c-\frac{1}{2}g^TH^{-1}g.$$

For part (2), we first show that $\{x \in \mathbb{R}^n : f(x) \le l\}$ is the union of two convex components. Write $H = QDQ^T$ so that D is diagonal and Q is orthogonal, and write $h(u) = \frac{1}{2}u^T Du$. We have $h(u) = f(Q^T u - H^{-1}g) - c + \frac{1}{2}g^T H^{-1} g$. To simplify notation, let

$$\bar{l} := l - [c - \frac{1}{2} g^T H^{-1} g].$$

Consider the set

$$S_+ \quad := \quad \{u \in \mathbb{R}^n : h(u) \le \bar{l}, u_n > 0\}.$$

Now,

$$\begin{aligned} h(u) &\le \bar{l} \\ \iff \frac{1}{2}\sum_{i=1}^{n} \lambda_i u_i^2 &\le \bar{l} \\ \iff \sum_{i=1}^{n-1} \lambda_i u_i^2 &\le 2\bar{l} - \lambda_n u_n^2. \end{aligned}$$

For given values $u_1, \ldots, u_{n-1}$, provided that $u_n \ge 0$, we have

$$u_n \ge \sqrt{\frac{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i u_i^2}{-\lambda_n}}.$$

Consider the function $g : \mathbb{R}^{n-1} \to \mathbb{R}$ defined by $g(v) = \sqrt{\frac{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i v_i^2}{-\lambda_n}}$. The set S_+ is the epigraph of g, so S_+ is a convex set if and only if g is a convex function. We proceed to show that g is convex, that is $\frac{1}{2}[g(v) + g(w)] \ge g(\frac{1}{2}(v+w))$ for all $v, w \in \mathbb{R}^{n-1}$. We have

$$\begin{aligned} \frac{1}{2}[g(v) + g(w)] &\ge g\left(\frac{1}{2}(v+w)\right) \\ \iff \frac{1}{2}\left[\sqrt{\frac{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i v_i^2}{-\lambda_n}} + \sqrt{\frac{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i w_i^2}{-\lambda_n}}\right] &\ge \sqrt{\frac{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i \left[\frac{v_i + w_i}{2}\right]^2}{-\lambda_n}} \\ \iff -2\bar{l} + \sum_{i=1}^{n-1} \lambda_i v_i^2 - 2\bar{l} + \sum_{i=1}^{n-1} \lambda_i w_i^2 + & \\ +2\sqrt{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i v_i^2}\sqrt{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i w_i^2} &\ge -8\bar{l} + 4\sum_{i=1}^{n-1} \lambda_i \left[\frac{v_i + w_i}{2}\right]^2 \\ \iff \sqrt{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i v_i^2}\sqrt{-2\bar{l} + \sum_{i=1}^{n-1} \lambda_i w_i^2} &\ge -2\bar{l} + \sum_{i=1}^{n-1} \lambda_i v_i w_i. \end{aligned} \tag{3}$$

Let $\tilde{v}, \tilde{w} \in \mathbb{R}^n$ be such that

$$\begin{aligned} \tilde{v}_i &= \sqrt{\lambda_i} v_i \text{ for } 1 \le i \le n-1, \\ \tilde{v}_n &= \sqrt{-2\bar{l}}, \\ \tilde{w}_i &= \sqrt{\lambda_i} w_i \text{ for } 1 \le i \le n-1, \\ \text{and } \tilde{w}_n &= \sqrt{-2\bar{l}}. \end{aligned}$$

The Cauchy-Schwarz inequality gives $\|\tilde{v}\|_2 \|\tilde{w}\|_2 \ge \langle \tilde{v}, \tilde{w} \rangle$, which is exactly (3), establishing the convexity of g as needed. Similarly, S_- defined by $\{u \in \mathbb{R}^n : h(u) \le \bar{l}, u_n < 0\}$ is also convex, and so $\{u \in \mathbb{R}^n : h(u) \le \bar{l}\}$, being the union of S_+ and S_-, is the union of two convex sets. The closest points between the sets S_+ and S_- are $\pm\sqrt{\frac{2\bar{l}}{\lambda_n}} e_n$, where e_n is the n-th elementary vector.

The sets $\{u \in \mathbb{R}^n : h(u) \leq \bar{l}\}$ and $\{u \in \mathbb{R}^n : f(Q^T u - H^{-1}g) - c + \frac{1}{2}g^T H^{-1} g \leq \bar{l}\}$ are identical, and is related to $\{x \in \mathbb{R}^n : f(x) \leq l\}$ by an orthogonal transformation and a translation. This gives the formula of $\tilde{x}$ and $\tilde{y}$ as needed. □

3. Algorithm for finding saddle points

In this section, we present an optimality condition for the closest points of level sets, followed by our main algorithm to find saddle points of mountain pass type. Here is our first observation of level sets.

Proposition 3.1. *(Closest points to level sets of functions) Suppose that two sets A, B in $\mathbb{R}^n$ are defined by $A = \{x : f(x) \leq l\}$ and $B = \{x : g(x) \leq l\}$ for $l \in \mathbb{R}$ and $\mathscr{C}^1$ functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$. Assume that A and B are convex, and that $A \cap B = \emptyset$. The points $\bar{a} \in A$ and $\bar{b} \in B$ are minimizers of*

$$\begin{aligned} &\min \|a - b\|_2 \\ &\text{s.t. } a \in A \text{ and } b \in B, \end{aligned} \tag{4}$$

if and only if

$$f(\bar{a}) = l,\ g(\bar{b}) = l,\ \text{and } \left\langle \frac{\nabla f(\bar{a})}{\|\nabla f(\bar{a})\|}, \frac{\bar{b} - \bar{a}}{\|\bar{b} - \bar{a}\|} \right\rangle = \left\langle \frac{\nabla g(\bar{b})}{\|\nabla g(\bar{b})\|}, \frac{\bar{a} - \bar{b}}{\|\bar{a} - \bar{b}\|} \right\rangle = 1. \tag{5}$$

Proof. For the forward direction, fix $\bar{a}$ and consider

$$\begin{aligned} &\min \|\bar{a} - b\|_2 \\ &\text{s.t. } b \in B. \end{aligned}$$

Since B is convex, it is well known that the minimizer to the problem is the projection of $\bar{a}$ to the set B and is unique. Furthermore, $\left\langle \frac{\nabla g(\bar{b})}{\|\nabla g(\bar{b})\|}, \frac{\bar{a} - \bar{b}}{\|\bar{a} - \bar{b}\|} \right\rangle = 1$. The equation for the other inner product comes by fixing $\bar{b}$ and varying a.

For the backward direction, suppose that (5) holds. So $\|\bar{a} - \bar{b}\|_2$ is an upper bound on (4). The halfspace $A' := \{x \mid \nabla f(\bar{a})^T (x - \bar{a}) \leq 0\}$ contains A, and similarly for $B' = \{x \mid \nabla g(\bar{b})^T (x - \bar{b}) \leq 0\}$. The distance between A' and B' can be easily checked to be $\|\bar{a} - \bar{b}\|_2$. This means that $\|\bar{a} - \bar{b}\|_2$ equals the value of (5), which gives us what we need. □

We now propose an algorithm for finding saddle points of mountain pass type, concentrating on the case where f is an exact quadratic to simplify our analysis.

Algorithm 3.2. *(Estimating critical level) Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is an exact quadratic with unknown parameters that has a Hessian with $n-1$ positive eigenvalues and one negative eigenvalue. This algorithm approximates $\bar{x}$ such that $\nabla f(\bar{x}) = 0$.*

1. *Fix $i = 1$ and $\varepsilon > 0$.*
2. *Let $l_i := \max\{f(x_0), f(y_0)\}$.*
3. *Run Algorithm 3.3 to find 2 points x_i and y_i satisfying (7) for $\tilde{x} = x_i$ and $\tilde{y} = y_i$.*
4. *If one of the convergence criteria holds:*
   (a) *$\|x_i - y_i\|$ is sufficiently small, or*
   (b) *$\frac{1}{2}[\nabla f(x_i) + \nabla f(y_i)]$, which equals $\nabla f(\frac{1}{2}(x_i + y_i))$, is sufficiently small in norm,*

   then return $\frac{1}{2}[x_i + y_i]$. Otherwise, let $l_{i+1} > l_i$ be a lower approximate of the critical value, increase the value of i and return to step 3.

Algorithm 3.2 uses the following algorithm to find iterates in step 2.

Algorithm 3.3. *(Estimating closest points in level sets) Consider $f : \mathbb{R}^n \to \mathbb{R}$ in Algorithm 3.2. Our terminating condition is motivated by the optimality condition (5). For inputs $l_i < f(\bar{x})$ and $x, y \in \mathbb{R}^n$ such that $f(x) = f(y) = l_i$ and a parameter $0 < \varepsilon \ll 1$, this algorithm returns two points $\tilde{x}, \tilde{y} \in \mathbb{R}^n$ such that $f(\tilde{x}) = f(\tilde{y}) = l_i$ and (7) holds.*

(1) *Fix $j = 1$, $\tilde{x}_1 = x$ and $\tilde{y}_1 = y$.*
(2) *Let $d_1 = \frac{\tilde{y}_j - \tilde{x}_j}{\|\tilde{y}_j - \tilde{x}_j\|}$. Choose a second direction d_2 from $\tilde{x}_j$, $\tilde{y}_j$, $\nabla f(\tilde{x}_j)$ and $\nabla f(\tilde{y}_j)$.*
(3) *Let A_j be the affine space passing through $\tilde{x}_j$ with lineality space the span of d_1 and d_2. Use Proposition 2.1 and further evaluations of f on A_j to determine the quadratic model of f on A_j (which will be exact since f is assumed to be quadratic). Consider*

$$\begin{aligned} \min \quad & \|x-y\| \\ s.t. \quad & x \in S(\tilde{x}_j), y \in S(\tilde{y}_j), \end{aligned} \tag{6}$$

where $S(z)$ is the component of the level set $\{u \in A_j : f(u) \le l_i\}$ that contains the point z. Use Proposition 2.2 to find the minimizers to (6)*, and let them be the new iterates $\tilde{x}_{j+1}$ and $\tilde{y}_{j+1}$. Clearly, we have $f(\tilde{x}_{j+1}) = f(\tilde{y}_{j+1}) = l_i$.*
(4) *Increase j, and go back to step 2 unless one of the convergence criteria holds:*
   (a) *For $\tilde{x} = \tilde{x}_j$ and $\tilde{y} = \tilde{y}_j$, we have*

$$\left\langle \frac{\nabla f(\tilde{x})}{\|\nabla f(\tilde{x})\|}, \frac{\tilde{y}-\tilde{x}}{\|\tilde{y}-\tilde{x}\|} \right\rangle \ge (1-\varepsilon) \text{ and } \left\langle \frac{\nabla f(\tilde{y})}{\|\nabla f(\tilde{y})\|}, \frac{\tilde{x}-\tilde{y}}{\|\tilde{x}-\tilde{y}\|} \right\rangle \ge (1-\varepsilon). \tag{7}$$

   (b) *$\frac{1}{2}[\nabla f(\tilde{x}_j) + \nabla f(\tilde{y}_j)]$ has sufficiently small norm.*

Note that in the algorithms above, there are still choices to be made on the value l_{i+1} in step 4 of Algorithm 3.2 and the direction d_2 in Step 2 of Algorithm 3.3, which we will discuss later. In Subsection 4.2, we shall see that the conjugate gradient algorithm is similar to Algorithms 3.2 and 3.3 combined.

Remark 3.4. (Using largest affine space possible) In Algorithm 3.3, the affine space A_j is only of dimension 2. A straightforward extension of Algorithm 3.3 is to amend step (3) in as follows:

(3(H)) In step (3), let A_j be the affine space passing through all previously evaluated points and $\tilde{x}_j + d_2$ instead, and proceed as in the rest of step (3).

This approach also corresponds to finding the quadratic approximation on the largest possible affine space with the data at hand. We refer to this the algorithm with this modification as Algorithm 3.2(H) and Algorithm 3.3(H).

For readers who wish to apply the methods in this paper for finding saddle points of Morse index higher than 1, the algorithms here can be extended in the spirit of [18].

4. Choice of second direction d_2 in Algorithm 3.3

As remarked after Algorithm 3.3, the choice of d_2 has to be made in step (3) there. In this section, we present and explain the different choices of d_2 summarized in Table 2.

4.1. Choosing two additional directions d_2 and d_3 instead of one additional direction d_2.

The strategy in (3D) is to obtain the directions d_2 and d_3 such that both $\nabla f(\tilde{x}_j)$ and $\nabla f(\tilde{y}_j)$ lie in the span of $\{d_1, d_2, d_3\}$. We shall see in Remark 4.4 that this strategy is the best among the five strategies presented.

4.2. Straightforward generalization of the conjugate gradient algorithm.

We recall that the conjugate gradient algorithm, which is now considered classical in optimization, can be stated as follows.

Algorithm 4.1. *(Conjugate gradient algorithm) Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is defined by $f(x) = \frac{1}{2}x^T Hx + g^T x + c$, where $H \in \mathbb{R}^{n\times n}$ is positive definite. Then the conjugate gradient algorithm with starting iterate z_0 can be expressed as follows:*

(1) *Start with iterate $z_0 \in \mathbb{R}^n$, and let $i = 0$.*
(2) *Evaluate $\nabla f(z_i)$, and let A_i be the affine space through z_0 with lineality space spanned by $\{\nabla f(z_0), \ldots, \nabla f(z_i)\}$. Let z_{i+1} be the point on A_i such that $\nabla f(z_{i+1})$ is orthogonal to all elements in $\{\nabla f(z_0), \ldots, \nabla f(z_i)\}$.*

	Choice of d_2 in Algorithm 3.3
(3D)	(Three directions) Instead of choosing one direction d_2, choose two directions d_2 and d_3 such that both $\nabla f(\tilde{x}_j)$ and $\nabla f(\tilde{y}_j)$ lie in the span of $\{d_1, d_2, d_3\}$. The affine space in in Algorithm 3.3 is 3-dimensional instead of 2-dimensional. See Subsection 4.1.
(MG)	(Midpoint gradient) Let $d_2 = \frac{1}{2}[\nabla f(\tilde{x}_j) + \nabla f(\tilde{y}_j)]$, which is an approximate of $\nabla f(\frac{1}{2}[\tilde{x}_j + \tilde{y}_j])$. See Subsection 4.2.
(MV)	(Maximum violation of (7)) Choose d_2 by $d_2 = \begin{cases} \nabla f(\tilde{x}_j) & \text{if } \left\langle \frac{\nabla f(\tilde{x}_j)}{\|\nabla f(\tilde{x}_j)\|}, \frac{\tilde{y}_j - \tilde{x}_j}{\|\tilde{y}_j - \tilde{x}_j\|} \right\rangle \leq \left\langle \frac{\nabla f(\tilde{y}_j)}{\|\nabla f(\tilde{y}_j)\|}, \frac{\tilde{x}_j - \tilde{y}_j}{\|\tilde{x}_j - \tilde{y}_j\|} \right\rangle \\ \nabla f(\tilde{y}_j) & \text{otherwise.} \end{cases}$ See Subsections 4.3 and 4.4.
(PM)	(Power maximization) Choose d_2 so that $\|v_x\|^2 + \|v_y\|^2$ is maximal, where v_x is the projection of $\nabla f(\tilde{x}_j)$ onto the space spanned by d_1 and d_2, and similarly for v_y. See Subsection 4.3.
(MD)	(Midpoint distance) Find the minimizer $\bar{z}$ of $\min_z \left\{ \left\| \frac{\tilde{x}_j + \tilde{y}_j}{2} - z \right\| : \langle z - \tilde{x}_j, \nabla f(\tilde{x}_j) \rangle = 0, \langle z - \tilde{y}_j, \nabla f(\tilde{y}_j) \rangle = 0 \right\}$, and let $d_2 = \bar{z} - \frac{\tilde{x}_j + \tilde{y}_j}{2}$. If $\frac{\nabla f(\tilde{x}_i)}{\|\nabla f(\tilde{x}_i)\|} + \frac{\nabla f(\tilde{y}_j)}{\|\nabla f(\tilde{y}_j)\|}$ gets too close to 0, we use (MV) instead to avoid numerical difficulties. See Subsection 4.4.
(H)	(Using largest affine space possible) In Remark 3.4, we suggested using the largest affine space possible that can be created from previous evaluations of $f(\cdot)$ and the gradients $\nabla f(\cdot)$. The Hessian $\tilde{H}$ of the model $\tilde{f}$ in Assumption 4.2 grows in size as the algorithm progresses.

TABLE 2. List of strategies to find second direction d_2 in step 2 of Algorithm 3.3.

(3) *Increase i by* 1 *and go to step 2 till* $\|\nabla f(z_i)\|$ *is sufficiently small.*

In step 2, the point z_{i+1} also minimizes f on the affine space A_i. The gradient $\nabla f(z_{i+1})$ is also orthogonal to the lineality space of A_i.

We now explain the strategy (MG). Let the midpoint of the iterates $\tilde{x}_j$ and $\tilde{y}_j$ in Algorithm 3.3 be $\tilde{z}_j$. We see that $\nabla f(\tilde{z}_j)$ is orthogonal the lineality space of the affine space A_j. The midpoint $\tilde{z}_j$ plays the role of z_i in the conjugate gradient algorithm above. If the direction d_2 in Algorithm 3.3(H) was chosen to be $\nabla f(\tilde{z}_j)$, then Algorithm 3.3(H)(MG) is identical to the conjugate gradient algorithm stated in Algorithm 4.1 but with the condition that H be positive definite dropped. The midpoint $\tilde{z}_j$ can also be calculated without computing $\tilde{x}_j$ and $\tilde{y}_j$.

This strategy is also appealing for a few reasons. In Algorithm 4.1, finding the point z_{i+1} in Step 2 using Proposition 2.2 requires the solution of a linear system with a symmetric (though not necessarily positive definite) matrix but not an eigen-decomposition like in Algorithm 3.3. Algorithm 4.1 also does not require the knowledge of the Morse index of the critical point.

4.3. **Preserving the optimality condition** (7)**.** We now state an assumption that will be used later.

Assumption 4.2. *Suppose* $f(x) = \frac{1}{2}x^T Hx + g^T x + c$. *At iteration j of Algorithm 3.3, let* $L_j \in \mathbb{R}^{n\times 2}$ *be such that the columns of* L_j *are orthogonal and span the directions* d_1 *and* d_2 *in Algorithm 3.3, with the first column of* L_j *being* d_1. *Let* $\tilde{H}_j \in S^2$, $\tilde{g}_j \in \mathbb{R}^2$ *and* $\tilde{c}_j \in \mathbb{R}$ *be defined by*

$$\tilde{H}_j = L_j^T H L_j, \qquad \tilde{g}_j = L_j^T [H\tilde{x}_j + g] \qquad \text{and } \tilde{c}_j = \frac{1}{2}\tilde{x}_j H \tilde{x}_j + g^T \tilde{x}_j + c. \tag{8}$$

so that $\tilde{f}_j : \mathbb{R}^2 \to \mathbb{R}$ *defined by* $\tilde{f}_j(v) = f(L_j v + \tilde{x}_j)$ *equals* $\frac{1}{2} v^T \tilde{H}_j v + \tilde{g}_j^T v + \tilde{c}_j$.

Other facts immediate from Assumption 4.2 are that the mapping $v \mapsto L_j v + \tilde{x}_j$ is a bijection between $\mathbb{R}^2$ and the affine space A_j passing through $\tilde{x}_j$ with lineality space spanned by d_1 and d_2.

We now explain the strategies (MV) and (PM), and first note the following easy result.

Fact 4.3. *(Preservation of constraint violation) Suppose Assumption 4.2 holds. Let* e_k *be the kth elementary vector in* $\mathbb{R}^n$. *We have*

$$\nabla \tilde{f}_j(0) = \tilde{g} = L_j^T H \tilde{x}_j = L_j^T \nabla f(\tilde{x}_j),$$

and

$$\begin{aligned} \nabla \tilde{f}_j(\|\tilde{y}_j - \tilde{x}_j\| e_1) &= \tilde{H}_j \|\tilde{y}_j - \tilde{x}_j\| e_1 + \tilde{g}_j \\ &= L_j^T (H L_j \|\tilde{y}_j - \tilde{x}_j\| e_1 + H \tilde{x}_j) \\ &= L_j^T H \tilde{y}_j. \end{aligned}$$

If d_2 *is chosen to be* $\nabla f(\tilde{x}_j)$, *then* $\|\nabla \tilde{f}_j(0)\| = \|L_j^T \nabla f(\tilde{x}_j)\| = \|\nabla f(\tilde{x}_j)\|$, *which gives*

$$\begin{aligned} \left\langle \frac{\nabla \tilde{f}_j(0)}{\|\nabla \tilde{f}_j(0)\|}, e_1 \right\rangle &= \left\langle \frac{L_j^T \nabla f(\tilde{x}_j)}{\|\nabla f(\tilde{x}_j)\|}, e_1 \right\rangle \\ &= \left\langle \frac{\nabla f(\tilde{x}_j)}{\|\nabla f(\tilde{x}_j)\|}, L_j e_1 \right\rangle \\ &= \left\langle \frac{\nabla f(\tilde{x}_j)}{\|\nabla f(\tilde{x}_j)\|}, \frac{\tilde{y}_j - \tilde{x}_j}{\|\tilde{y}_j - \tilde{x}_j\|} \right\rangle. \end{aligned} \tag{9}$$

Similarly, if d_2 *is chosen to be* $\nabla f(\tilde{y}_j)$, *then* $\left\|\nabla \tilde{f}_j(\|\tilde{y}_j - \tilde{x}_j\| e_1)\right\| = \|\nabla f(\tilde{y}_j)\|$, *which gives*

$$\left\langle \frac{\nabla \tilde{f}_j(\|\tilde{y}_j - \tilde{x}_j\| e_1)}{\left\|\nabla \tilde{f}_j(\|\tilde{y}_j - \tilde{x}_j\| e_1)\right\|}, -e_1 \right\rangle = \left\langle \frac{\nabla f(\tilde{y}_j)}{\|\nabla f(\tilde{y}_j)\|}, \frac{\tilde{x}_j - \tilde{y}_j}{\|\tilde{x}_j - \tilde{y}_j\|} \right\rangle. \tag{10}$$

In view of the optimality conditions in (5), one choice of d_2 is marked as (MV) in Table 2. In other words, the maximum violation of the optimality conditions is preserved in the model $\tilde{f}_j : \mathbb{R}^2 \to \mathbb{R}$. It is clear that (3D) preserves the violation in both optimality conditions.

Notice also in Fact 4.3 that if d_2 were chosen to be $\nabla f(\tilde{x}_j)$, then the projection of $\nabla f(\tilde{x}_j)$ onto the subspace spanned by d_1 and d_2 is exactly $\nabla f(\tilde{x}_j)$ itself. A similar thing happens if d_2 were chosen to be $\nabla f(\tilde{y}_j)$. This motivates another strategy (PM) in Table 2.

4.4. **Using affine spaces to approximate the level set.** We now explain strategy (MD), and give a second explanation for (MV). As illustrated in Figure 3, the region $\{u \in \mathbb{R}^n : f(u) = l_i\}$ near $\tilde{x}_j$ and $\tilde{y}_j$ can be approximated by

$$\begin{aligned} & \{u \in \mathbb{R}^n : \nabla f(\tilde{x}_j)^T (u - \tilde{x}_j) = 0\} \\ \text{and} \quad & \{u \in \mathbb{R}^n : \nabla f(\tilde{y}_j)^T (u - \tilde{y}_j) = 0\} \text{ respectively.} \end{aligned} \tag{11}$$

Provided that $\nabla f(\tilde{x}_j)$ and $\nabla f(\tilde{y}_j)$ are not multiples of each other, the two affine spaces intersect in an affine space of dimension $n-2$. Let the projection of $\frac{1}{2}(\tilde{x}_j + \tilde{y}_j)$ onto this affine space be $\bar{z}$. We shall choose d_2 so that the affine space through $\tilde{x}_j$ with lineality space spanned by d_1 and d_2 passes through $\bar{z}$. Such a strategy is sensible because if perturbing $\tilde{x}_j$ and $\tilde{y}_j$ with $\bar{z} - \tilde{x}_j$ and $\bar{z} - \tilde{y}_j$ as tangent directions respectively can give good decrease, though not necessarily optimal decrease. To calculate d_2, we observe that the normals of the intersection of the affine spaces in (11) are linear combinations of $\nabla f(\tilde{x}_j)$ and $\nabla f(\tilde{y}_j)$. We can then write $d_2 = \alpha \nabla f(\tilde{x}_j) + \beta \nabla f(\tilde{y}_j)$,

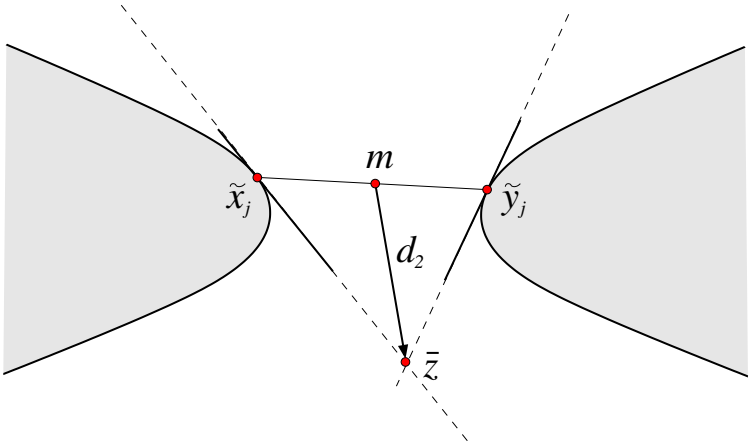

FIGURE 3. We illustrate the method (MD) of finding the direction d_2. The boundaries of the level set $\{x : f(x) \le l\}$ is approximated by affine spaces in (11), and the direction d_2 equals the vector $\bar{z} - m$, where $m = \frac{\tilde{x}_j + \tilde{y}_j}{2}$ and $\bar{z}$ is the closest point from m to the intersection of the affine spaces.

where α and β are determined by

$$\begin{aligned} \nabla f(\tilde{x}_j)^T \left[\frac{1}{2}(\tilde{x}_j + \tilde{y}_j) + d_2 - \tilde{x}_j \right] &= 0 \\ \text{and } \nabla f(\tilde{y}_j)^T \left[\frac{1}{2}(\tilde{x}_j + \tilde{y}_j) + d_2 - \tilde{y}_j \right] &= 0 \\ \Rightarrow \begin{pmatrix} \nabla f(\tilde{x}_j)^T \nabla f(\tilde{x}_j) & \nabla f(\tilde{x}_j)^T \nabla f(\tilde{y}_j) \\ \nabla f(\tilde{x}_j)^T \nabla f(\tilde{y}_j) & \nabla f(\tilde{y}_j)^T \nabla f(\tilde{y}_j) \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} &= \frac{1}{2} \begin{pmatrix} \nabla f(\tilde{x}_j)^T [\tilde{y}_j - \tilde{x}_j] \\ \nabla f(\tilde{y}_j)^T [\tilde{x}_j - \tilde{y}_j] \end{pmatrix}. \end{aligned}$$

If $\frac{\nabla f(\tilde{x}_i)}{\|\nabla f(\tilde{x}_i)\|} + \frac{\nabla f(\tilde{y}_j)}{\|\nabla f(\tilde{y}_j)\|}$ gets too close to 0, the value of $\|\frac{1}{2}(\tilde{x}_j + \tilde{y}_j) - \bar{z}\|$ is likely to be much larger than $\|\tilde{x}_j - \tilde{y}_j\|$, and is likely to cause numerical difficulties. For this reason, the method (MD) needs to switch to some other method if $\|\frac{1}{2}(\tilde{x}_j + \tilde{y}_j) - \bar{z}\|$ is greater than a fixed multiple of $\|\tilde{x}_j - \tilde{y}_j\|$.

More sophisticated estimates of the closest points can be devised, but we shall see in Section 8 that (MD) performs well compared to the other algorithms.

Lastly, we consider the case of fixing $\tilde{y}_j$ and perturbing $\tilde{x}_j$. The region $\{u \in \mathbb{R}^n : f(u) = l_i\}$ near $\tilde{x}_j$ can be approximated by $\{u \in \mathbb{R}^n : \nabla f(\tilde{x}_j)^T (u - \tilde{x}_j) = 0\}$. With elementary geometry, we can show that the best tangent direction to perturb $\tilde{x}_j$ in is $d_1 - \frac{1}{\|\nabla f(\tilde{x}_j)\|^2} [\nabla f(\tilde{x}_j)^T d_1] \nabla f(\tilde{x}_j)$. Choosing d_2 to be $\nabla f(\tilde{x}_j)$ allows one to perturb $\tilde{x}_j$ in that direction. This strategy gives the same choice of d_2 as (MV) earlier. The case of fixing $\tilde{x}_j$ and perturbing $\tilde{y}_j$ is similar.

Remark 4.4. (Directions in (3D)) The directions d_2 in the methods (MV), (PM), (MD) and (MG) in Table 2 are linear combinations of $\nabla f(\tilde{x}_j)$ and $\nabla f(\tilde{y}_j)$. Hence the method (3D) will always be better than the other methods.

5. MORE ON l_i IN ALGORITHM 3.2, AND METHODS (MG) AND (3D)

In the case where $f : \mathbb{R}^n \to \mathbb{R}$ is not necessarily a quadratic, the level l_i in Algorithm 3.2 clearly needs to converge to a critical value so that the iterates converge to the corresponding critical point. Once near the critical point, the quadratic approximation gets better, and Algorithms 3.2 and 3.3 become more effective. In this section, we clarify that when f is a quadratic function, the role of l_i is, on the contrary, not as important.

We look at the method (MG) in Table 2. As remarked in Subsection 4.2, the strategy (MG) does not use the level set structure for its computations. The next result explains that the choice of l_i is irrelevant for both (MG) and (3D).

Proposition 5.1. *(Irrelevance of l_i in (MG) and (3D)) If $f:\mathbb{R}^n\to\mathbb{R}$ is a quadratic $f(x)=\frac{1}{2}x^THx+g^Tx+c$, then the two dimensional space spanned by d_1 and d_2 in Algorithm 3.3(MG) does not depend on l_i. Similarly, the three dimensional subspace spanned by d_1, d_2 and d_3 does not depend on l_i in Algorithm 3.3(3D). Similar conclusions hold for Algorithm 3.3(H)(MG) and Algorithm 3.3(H)(3D).*

Proof. Let $\tilde{x}_j$ and $\tilde{y}_j$ be iterates in Algorithm 3.3. The direction d_1 is common to both strategies (MG) and (3D). By looking at the quadratic models on the affine space and using Proposition 2.2, we see that the direction of $\tilde{y}_j-\tilde{x}_j$ is independent of l_i.

We first look at the case of strategy (MG). The corresponding d_2 equals $\nabla f(\frac{1}{2}(\tilde{x}_j+\tilde{y}_j))$, which in turn does not depend on l_j. For strategy (3D), the directions d_2 and d_3 can be chosen to be $d_2=\nabla f(\frac{1}{2}(\tilde{x}_j+\tilde{y}_j))$, and $d_3=\nabla f(\tilde{x}_j)-d_2$. Given that f is quadratic, the gradient map $\nabla f:\mathbb{R}^n\to\mathbb{R}^n$ can be written as $\nabla f(x)=Hx+g$, and is affine. We have

$$\begin{aligned}\nabla f(\tilde{x}_j) &= H\tilde{x}_j+g\\ &= H\Big(\frac{1}{2}[\tilde{x}_j-\tilde{y}_j]\Big)+H\Big(\frac{1}{2}[\tilde{x}_j+\tilde{y}_j]\Big)+g\\ &= H\Big(\frac{1}{2}[\tilde{x}_j-\tilde{y}_j]\Big)+d_2.\end{aligned}$$

Since the direction of $\tilde{x}_j-\tilde{y}_j$ does not depend on l_i, the direction of $d_3=H(\frac{1}{2}[\tilde{x}_j-\tilde{y}_j])$ does not depend on l_j too. The analysis for (H)(MG) and (H)(3D) is similar. □

The above result shows that in the case when f is quadratic, we can rewrite Algorithm 3.2 (MG) and (3D) as the following equivalent algorithm without making use of the variable l_i.

Algorithm 5.2. *(Equivalent algorithms for (MG) and (3D)) Given a quadratic function $f:\mathbb{R}^n\to\mathbb{R}$ where $f(x)=\frac{1}{2}x^THx+g^Tx+c$, and the critical point $\bar{x}=-H^{-1}g$ has Morse index one,*

(1) *Let $i=0$. Start with approximate critical point z_0, and let A_0 be some affine space containing z_0.*
(2) *Use Proposition 2.1 and further evaluations of f on A_i to determine the quadratic model of f on A_i (which will be exact since f is assumed to be quadratic). From the quadratic model, find the point z_{i+1} in A_i where $\nabla f(z_{i+1})$ is orthogonal to the lineality space of A_i.*
(3) *Let the lineality space of A_i be spanned by the columns of the matrix L_i, where L_i has orthogonal columns. To find the next affine space A_{i+1}, we need to figure out the directions d_1, d_2 and d_3 as follows:*
   (a) *d_1 is the eigenvector corresponding to the negative eigenvector of $\tilde{H}=L_i^THL_i$.*
   (b) *d_2 equals $\nabla f(z_{i+1})$.*
   (c) *d_3 equals $\nabla f(z_{i+1}+\lambda d_1)$, where λ is any nonzero scalar. The vector d_3 needs to be calculated for (3D), but not for (MG).*

   The affine space A_{i+1} for the different strategies all pass through z_{i+1}, but have different lineality spaces:
   (i) *For (MG), the lineality space of A_{i+1} is spanned by $\{d_1,d_2\}$.*
   (ii) *For (3D), the lineality space of A_{i+1} is spanned by $\{d_1,d_2,d_3\}$.*
   (iii) *For (H)(MG), the lineality space of A_{i+1} is spanned by the columns of L_i and d_2. (Note that d_1 lies in the column space of L_i.)*
   (iv) *For (H)(3D), the lineality space of A_{i+1} is spanned by the columns of L_i, d_2 and d_3.*

   From this information we can deduce L_{i+1}. Increase the counter i by one and return to step 2 until the convergence criteria of $\nabla f(z_i)$ being small in norm is met.

As can be seen from Algorithm 5.2, methods (H)(MG) and (H)(3D) are equivalent to a Krylov subspace method. In (H)(3D), the Krylov subspace grows by two dimensions instead of one in each iteration.

It is clear that while the other strategies in Table 2 do not enjoy the property in Proposition 5.1, they can be more effective when f is not a quadratic function. We have the following heuristic on the choice of l_{i+1}.

Remark 5.3. (Choice of l_{i+1}) To choose l_{i+1} from l_i in Algorithm 3.2, one possible strategy is to make use of Proposition 2.2. Given $f(x) = \frac{1}{2}x^T Hx + g^T x + c$, the critical level is $c - \frac{1}{2}g^T H^{-1} g$, the distance between the components is $2\sqrt{\frac{1}{\lambda_n}[2l - (2c - g^T H^{-1} g)]}$, where the negative eigenvalue λ_n of H is approximated in step 3 of Algorithm 3.3. The critical level can be estimated from λ_n and the distance between the components.

For $\tilde{x}_j$ and $\tilde{y}_j$ to be well defined, l_i needs to be a lower bound of the critical level. If l_i is found to be larger than the critical value, then l_i can be reduced so that it is below the critical value. Contrast the management of l_i to that in the main algorithm in [13], where a sequence of lower bounds $\{l_i\}$ of the critical value is obtained through an optimization procedure. The $\{l_i\}$ there converges superlinearly to the critical value.

6. Augmenting ideas to a path based algorithm

A commonly used method of finding an optimal mountain pass is still to discretize and perturb a path between two endpoints so that the maximum along the path decreases. We now remark on how the idea of finding a quadratic expression near a critical point can be augmented to a path based algorithm.

A basic path-based algorithm can be described as follows.

Algorithm 6.1. *(Basic path based algorithm) Given $f : X \to \mathbb{R}$ and two points $a, b \in X$, find an optimal mountain pass $p : [0,1] \to X$ connecting a and b.*

1. *Consider a discretized path $p_1, p_2, \ldots, p_k$, where $p_1 = a$ and $p_k = b$.*
2. *Find the maximizer of f on the line segments $[p_1, p_2], [p_2, p_3], \ldots, [p_{k-1}, p_k]$, say $\bar{p}$.*
3. *If $\|\nabla f(\bar{p})\|$ sufficiently close to 0, then algorithm ends. Otherwise, perturb the path $p_1, p_2, \ldots, p_k$ based on the gradient $\nabla f(\bar{p})$ and other information. The path may also be refined (i.e., more points can be used to describe the path) as necessary. Return to step 1.*

The ideas on finding a quadratic approximation near the saddle point $\bar{x}$ can be incorporated into the basic path based algorithm. The point $\bar{p}$ is the point most likely to be closest to the saddle point, and the evaluations of f near $\bar{p}$ can be used to deduce the quadratic approximation of f near $\bar{x}$. The quadratic approximation of f on an affine space through $\bar{p}$ can be constructed through Propositions 2.1 and 2.2. The critical point is estimated to be $\bar{p} - H(\bar{p})^{-1}\nabla f(\bar{p})$ like in a Newton method, but because the full information of $H(\bar{p})^{-1}$ may not be easily available, ideas from the algorithm we proposed can give a good indication of how to perturb the path $p_1, p_2, \ldots, p_k$ to reduce the maximum value of f along the path.

7. Convergence analysis

In this section, we prove in Theorem 7.1 a formula describing the rate of convergence of algorithm (MG), and prove in Theorem 7.2 that Algorithm 3.3 will eventually find two points so that (7) is satisfied. We begin with the analysis of Algorithm 5.2(H)(MG). For a positive definite matrix A, let the norm $\|\cdot\|_A$ be defined by $\|v\|_A := \sqrt{v^T A v}$.

Theorem 7.1. *(Rate of convergence of (MG)) Suppose that $H \in \mathscr{S}^n$ has $n-1$ positive eigenvalues and one negative eigenvalue, all of which lie in $[\underline{\lambda}, \overline{\lambda}] \cup \{-\hat{\lambda}\}$, where $0 < \underline{\lambda} < \overline{\lambda}$ and $\hat{\lambda} > 0$. Consider Algorithm 5.2(H)(MG) applied to finding the saddle point $\bar{z} = -H^{-1}g$ for*

$f(x) = \frac{1}{2}x^T Hx + g^T x + c$. For the ith iterate z_i, let the error $\mathbf{e}_i$ be $z_i - \bar{z}$. Then for any positive definite matrix A, the errors $\mathbf{e}_i$ satisfy

$$\frac{\|\mathbf{e}_i\|_A}{\|\mathbf{e}_0\|_A} \leq 2\frac{\overline{\lambda}+\hat{\lambda}}{\hat{\lambda}}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{i-1},$$

where $\kappa = \overline{\lambda}/\underline{\lambda}$.

Proof. Let Π_i be the set of polynomials p such that $p(0) = 1$. Then by using standard methods in the study of the convergence of the conjugate gradient algorithm (see for example [3, Section 9]), we have

$$\begin{aligned}
\frac{\|\mathbf{e}_i\|_A}{\|\mathbf{e}_0\|_A} &\leq \min_{p\in\Pi_i} \max_{\lambda\in\left([\underline{\lambda},\overline{\lambda}]\cup\{-\hat{\lambda}\}\right)} p(\lambda) \\
&\leq \min_{p\in\Pi_{i-1}} \max_{\lambda\in[\underline{\lambda},\overline{\lambda}]} p(\lambda)\frac{\lambda+\hat{\lambda}}{\hat{\lambda}} \\
&\leq \frac{\overline{\lambda}+\hat{\lambda}}{\hat{\lambda}} \min_{p\in\Pi_{i-1}} \max_{\lambda\in[\underline{\lambda},\overline{\lambda}]} p(\lambda) \\
&\leq 2\frac{\overline{\lambda}+\hat{\lambda}}{\hat{\lambda}}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{i-1}.
\end{aligned}$$

□

Theorem 7.2. *(Convergence of iterates of Algorithm 3.3(MV)) Suppose Assumption 4.2 holds at iteration j. Then for any $\varepsilon > 0$, Algorithm 3.3(MV) produces some iterate $\tilde{x}_{j^*}$ and $\tilde{y}_{j^*}$ satisfying (7) for $\tilde{x} = \tilde{x}_{j^*}$ and $\tilde{y} = \tilde{y}_{j^*}$.*

Proof. From Assumption 4.2, we infer that the second column of L is the unit vector in the direction of $d_2 - \frac{1}{\|d_1\|^2}(d_1^T d_2)d_1$. Consider the model $\tilde{f}_j(v) = \frac{1}{2}v^T\tilde{H}_j v + \tilde{g}_j^T v + \tilde{c}_j$, where $\tilde{H}_j$, $\tilde{g}_j$ and $\tilde{c}_j$ are as chosen in (8) based on the iterates $\tilde{x}_j$ and $\tilde{y}_j$. Seeking a contradiction, suppose that (7) is violated for all iterates.

Recall that at iteration j, if d_2 was chosen to be $\nabla f(\tilde{x}_j)$, then from Fact 4.3,

$$\begin{aligned}
\left\langle \frac{\nabla\tilde{f}_j(0)}{\|\nabla\tilde{f}_j(0)\|}, e_1 \right\rangle &= \left\langle \frac{\nabla f(\tilde{x}_j)}{\|\nabla f(\tilde{x}_j)\|}, \frac{\tilde{y}_j - \tilde{x}_j}{\|\tilde{y}_j - \tilde{x}_j\|} \right\rangle \\
&< 1-\varepsilon.
\end{aligned}$$

A similar inequality can be obtained if d_2 was chosen to be $\nabla f(\tilde{y}_j)$ instead. This means that the pair $(0, \|\tilde{y}_j - \tilde{x}_j\|e_1)$ are not the points in the model that minimize the distance between the components of $\{u \in \mathbb{R}^2 : \tilde{f}_j(u) \leq l_i\}$. Step 3 in Algorithm 3.3 chooses iterates $\tilde{x}_{j+1}$ and $\tilde{y}_{j+1}$ such that $\|\tilde{x}_{j+1} - \tilde{y}_{j+1}\| < \|\tilde{x}_j - \tilde{y}_j\|$.

From the formulas of the columns of L_j in terms of d_1 and d_2, we see that L_j depends continuously on d_1 and d_2. We shall first assume that

$$\left\langle \frac{\nabla f(\tilde{x}_j)}{\|\nabla f(\tilde{x}_j)\|}, \frac{\tilde{y}_j - \tilde{x}_j}{\|\tilde{y}_j - \tilde{x}_j\|} \right\rangle \neq \left\langle \frac{\nabla f(\tilde{y}_j)}{\|\nabla f(\tilde{y}_j)\|}, \frac{\tilde{x}_j - \tilde{y}_j}{\|\tilde{x}_j - \tilde{y}_j\|} \right\rangle. \tag{12}$$

Under this assumption, d_1 and d_2 are continuous on the iterates $(\tilde{x}_j, \tilde{y}_j)$, so the parameters $\tilde{H}_j$, $\tilde{g}_j$ and $\tilde{c}_j$ in (8) depend continuously on $(\tilde{x}_j, \tilde{y}_j)$ as well. This implies that the eigenvalues and eigenvectors of $\tilde{H}_j$ also depend continuously, and thus the next iterates also depend continuously on that of the previous iterates through Proposition 2.2.

Recall our earlier assumption that Algorithm 3.3 does not produce the iterates satisfying (7). By compactness arguments in $\mathbb{R}^n$, there is a subsequence of the iterates $\{(\tilde{x}_j, \tilde{y}_j)\}$ not satisfying (7) converging to some (x', y'). Since $f(\tilde{x}_j) = f(\tilde{y}_j) = l_i$ for all i, it follows that $f(x') = f(y') = l_i$, and that (x', y') do not satisfy (7) for $\tilde{x} = x'$ and $\tilde{y} = y'$. If a step of Algorithm 3.3 were to be

	Objective
(1)	Minimizing $\|\tilde{x}_j - \tilde{y}_j\|$, the distance between points in different components of $\{u \mid f(u) \le l_i\}$
(2)	Maximizing $\min\left(\left\langle \frac{\nabla f(\tilde{x}_j)}{\|\nabla f(\tilde{x}_j)\|}, \frac{\tilde{y}_j - \tilde{x}_j}{\|\tilde{y}_j - \tilde{x}_j\|} \right\rangle, \left\langle \frac{\nabla f(\tilde{y}_j)}{\|\nabla f(\tilde{y}_j)\|}, \frac{\tilde{x}_j - \tilde{y}_j}{\|\tilde{x}_j - \tilde{y}_j\|} \right\rangle\right)$, an optimality condition for (1). (See Proposition 3.1.)
(3)	Minimizing $\left\|\nabla f\left(\frac{1}{2}(\tilde{x}_j + \tilde{y}_j)\right)\right\|$, the gradient at the midpoint.
(4)	Minimizing $\|\bar{z} - \frac{1}{2}(\tilde{x}_j + \tilde{y}_j)\|$, the distance between the critical point and the midpoint.

TABLE 3. Objective functions used in our numerical experiments and the associated greedy algorithms.

applied to $\{(x', y')\}$, then we get new iterates (x'', y'') such that $\|x'' - y''\| < \|x' - y'\|$. If the assumption in (12) were dropped, then if the iterates $(\tilde{x}_j, \tilde{y}_j)$ approach (x', y'), then next iterates $(\tilde{x}_{j+1}, \tilde{y}_{j+1})$ approach two possible limits, say $(\hat{x}, \hat{y})$ and $(\check{x}, \check{y})$, and both $\|\hat{x} - \hat{y}\| < \|x' - y'\|$ and $\|\check{x} - \check{y}\| < \|x' - y'\|$.

The assumption that a subsequence of $\{(\tilde{x}_j, \tilde{y}_j)\}$ converges to (x', y') implies that the distance between iterates will not go below $\|x' - y'\|$, while the continuity of new iterates from the old iterates implies that there are sequences of iterates whose distances is arbitrarily close to $\|x'' - y''\|$. With minor adjustments, we can show that a similar condition holds for the case where (12) fails. This is a contradiction, and gives us the conclusion we seek. □

This theorem also tells us that Algorithm 3.3(H)(MV), Algorithm 3.3(3D) and Algorithm 3.3(H)(3D) converge.

While the convergence analysis pales in comparison to analogous results in optimization, it highlights that the second direction d_2 should be chosen so that $\left\langle \frac{\nabla \tilde{f}_j(0)}{\|\nabla \tilde{f}_j(0)\|}, e_1 \right\rangle$ and $\left\langle \frac{\nabla \tilde{f}_j(\|\tilde{y} - \tilde{x}\| e_1)}{\left\|\nabla \tilde{f}_j(\|\tilde{y} - \tilde{x}\| e_1)\right\|}, -e_1 \right\rangle$ should be as far away from 1 as possible to obtain decrease in the distance between components in the next iterate.

Note that if f were assumed to be such that the Hessian is locally Lipschitz instead, the statement in Theorem 7.2 need not hold for all $\varepsilon > 0$ because the errors in estimating a quadratic model may lead to an inaccurate estimate of the points minimizing the distance between the components of the level sets in the affine space.

8. NUMERICAL EXPERIMENTS

We now describe our numerical experiments in Matlab to test our algorithm.[4] Specifically, we shall test Algorithm 3.3 for the various choices of d_2 in Table 2.

8.1. **Objectives and greedy algorithms.** We shall only be concerned with running Algorithm 3.3(H) for a particular value l_i. While the clear objective in Algorithm 3.3(H) is to find the two closest points of the components of $\{u \mid f(u) \le l_i\}$, we also use other objectives listed in Table 3 in our numerical experiments. Objectives (1) to (3) can be calculated as the algorithm progresses, but objective (4) is what one really wants to compute. In ill-conditioned problems, objective (2) may be close to the optimal value of 1, but far from achieving the minimum distance in objective (1).

Other than the choice of directions in Table 2, we introduce greedy algorithms to study whether the choice of direction d_2 in Algorithm 3.3. In what follows, the strategy (G1) will mean that at each step of the iteration, Algorithm 3.3 tries out all directions d_2 in Table 2, then chooses the direction that best minimizes objective (1) in Table 3. The strategies (G2), (G3) and (G4) are similar. Strategy (G4) is an "invisible hand" that brings the iterates as close to the true saddle

[4]The Matlab codes are available in http://math.mit.edu/∼chj2pang/mtn_code.tar.gz

point 0 as possible. It is not practical because the knowledge of the true saddle would mean that there is no need to run the algorithm.

In typical applications of the problem of finding a saddle point of mountain pass type, the cost of evaluating the function and its gradient is high, and may take hours or longer. Compared to Algorithm 3.3, Algorithm 3.3(H) (see Remark 3.4) takes advantage of the quadratic formulation to obtain fast convergence to the saddle point. Therefore, we shall only perform experiments to study Algorithm 3.3(H). Our experiments would give an indication of how fast the convergence to the critical point would be once the quadratic approximation is reliable.

8.2. **Numerical experiments.** If $f : \mathbb{R}^n \to \mathbb{R}$ is a quadratic, then we can, with an orthogonal transformation and translation, assume that $f(x) = \frac{1}{2}x^T Dx$, where $D \in \mathbb{R}^{n\times n}$ is diagonal. The critical point is 0, and the critical value is 0. We shall also assume that the diagonal entries in D are arranged in descending order. This method also produces ill-conditioned matrices D. In an implementation of Algorithm 3.3, invoking Proposition 2.1 in step 3 to approximate the parameters of the quadratic approximation can be numerically difficult. We ignore such difficulties for the time being and study the effects of the different strategies discussed in this paper instead.

To start off our experiments, we generate the diagonal entries of D randomly from the uniform distribution on $[0,1]$ using the "rand" function in Matlab. The last eigenvalue will be chosen to be negative, and the rest will be positive. The critical level of f is 0, and we choose two points $\tilde{x}_0$ and $\tilde{y}_0$ such that $f(\tilde{x}_0) = f(\tilde{y}_0) = -1$. The points $\tilde{x}_0$ and $\tilde{y}_0$ are chosen as follows. First, we choose the first $n-1$ coordinates of $\tilde{x}_0$ and $\tilde{y}_0$ randomly from the normal distribution using the "randn" function in Matlab. Next, we choose the last coordinate of $\tilde{x}_0$ and $\tilde{y}_0$ so that $f(\tilde{x}_0) = f(\tilde{y}_0) = -1$.

We first observe the effect of different strategies in the Tables 2 and 3 (except for (3D), which is provably superior to the others). The results are summarized in Tables 4 and 5.

The following observations can be made for the different strategies:

(1) A greedy method may not be better than a pure strategy in the long term.
(2) The iterates produced by (MV) is the best for objectives (1) and (2) in the short and medium term.
(3) The iterates produced by (MG) has the best convergence of $\frac{1}{2}(\tilde{x}_j + \tilde{y}_j)$ to the critical point (objective (4)) and of reducing the norm of the gradient (objective (3)). Note that as the algorithm progresses, $\|\tilde{x}_j - \tilde{y}_j\|$ will get smaller, so $\nabla f(\tilde{x}_j)$ and $\nabla f(\tilde{y}_j)$ will be far too similar after some point. While (3D) can provably do a much better job if such numerical errors are not encountered, the above observation suggests that (MG) is a good strategy once close to the critical point.
(4) The iterates produced by (MD) has the best decrease for objectives (1), (3) and (4) in the first iteration, and this decrease is sustained for the next few iterations for (3) and (4). For objectives (1) and (2), once (MD) switches to (MV) to overcome ill-conditioning, the performance of the iterates catches up quickly to do just as well as the pure strategy (MV).

We now study the performance of all pure strategies, including (3D). The performance is shown in Figure 4. Here are some observations from Figure 4:

(1) The strategy (3D) is the best among all choices in Table 2, as expected.
(2) The strategy (MG) performs the poorest in the long run for objectives (1) and (2), as is consistent with the data in Tables 4 and 5.
(3) The strategy (MG) performs better than (MV), (PM) for objectives (3) and (4), as is consistent with the data in Tables 4 and 5.
(4) The strategy (MD) is the best in the first few iterations for objectives (3) and (4). This behavior persists even as (MD) switches to (MG) when ill conditioning is encountered. For objectives (1) and (2), (MD) becomes competitive with the other strategies after it switches to (MV). This switching explains the sharp decline in the graphs in objectives (1) and (2) in Figure 4.

Percentage of times Objective (1) is optimal for different strategies												
#	(H) + Strategy											
	Pure strategies				Pure and greedy strategies							
	(MV)	(PM)	(MD)*	(MG)	(MV)	(PM)	(MD)*	(MG)	(G1)	(G2)	(G3)	(G4)
1	1	2	75	22	1	2	75	22	0	0	0	0
2	51	29	0	13	21	0	14	13	39	0	0	0
3	34	23	0	43	0	3	0	0	63	34	0	0
4	70	23	0	7	1	6	0	0	36	56	1	0
5	78	21	0	1	33	14	0	0	24	28	1	0
6	83	17	0	0	12	10	0	0	43	35	0	0
7	74	22	4	0	3	10	0	0	47	40	0	0
8	48	22	30	0	9	13	4	0	20	54	0	0
9	37	14	49	0	8	8	16	0	34	33	1	0
10	27	20	53	0	0	13	15	0	45	26	1	0
11	23	21	56	0	4	10	13	0	38	34	1	0
12	27	22	51	0	4	10	10	0	34	39	3	0
13	24	27	49	0	1	18	11	0	35	32	3	0
14	22	31	47	0	1	13	14	0	32	37	3	0
15	22	34	44	0	3	19	10	0	39	26	3	0

Percentage of times Objective (2) is optimal for different strategies												
#	(H) + Strategy											
	Pure strategies				Pure and greedy strategies							
	(MV)	(PM)	(MD)*	(MG)	(MV)	(PM)	(MD)*	(MG)	(G1)	(G2)	(G3)	(G4)
1	84	10	4	2	84	10	4	2	0	0	0	0
2	94	6	0	0	46	5	0	0	0	49	0	0
3	75	24	0	1	1	4	0	0	38	57	0	0
4	68	32	0	0	41	26	0	0	5	28	0	0
5	45	55	0	0	42	51	0	0	2	5	0	0
6	64	36	0	0	28	21	0	0	10	41	0	0
7	59	41	0	0	13	17	0	0	12	58	0	0
8	57	41	2	0	27	28	2	0	8	35	0	0
9	53	37	10	0	37	26	6	0	9	21	1	0
10	53	37	10	0	22	23	6	0	10	39	0	0
11	42	34	24	0	15	24	10	0	7	44	0	0
12	40	34	26	0	19	23	15	0	7	36	0	0
13	35	33	32	0	23	28	18	0	6	24	1	0
14	31	33	36	0	22	24	24	0	2	27	1	0
15	36	40	24	0	19	34	16	0	4	26	1	0

TABLE 4. In a run of 100 experiments for $n = 100$ for the first 15 iterations, the percentage of times for which the corresponding strategy is optimal is recorded. We compare among pure strategies in the first four columns, and compare the pure strategies together with the greedy strategies in the next eight columns. In strategy (MD), once ill-conditioning is encountered, we switch to (MV).

9. CONCLUSION AND OPEN QUESTIONS

We presented an algorithm for finding a saddle point of mountain pass type using quadratic models on affine spaces. Our algorithm is similar in some ways to the conjugate gradient algorithm. The choices one has to make in implementing the algorithm are explained, and some

Percentage of times Objective (3) is optimal for different strategies												
#	(H) + Strategy											
	Pure strategies				Pure and greedy strategies							
	(MV)	(PM)	(MD)*	(MG)	(MV)	(PM)	(MD)*	(MG)	(G1)	(G2)	(G3)	(G4)
1	0	5	68	27	0	5	68	27	0	0	0	0
2	1	1	83	15	0	1	52	4	2	1	26	14
3	0	0	93	7	0	0	53	2	1	0	24	20
4	0	0	90	10	0	0	54	2	0	0	25	19
5	0	0	90	10	0	0	57	3	0	0	19	21
6	0	0	88	12	0	0	46	3	0	0	21	30
7	0	0	84	16	0	0	44	6	0	0	15	35
8	0	1	78	21	0	1	37	9	0	1	19	33
9	1	1	68	30	1	1	35	12	0	0	13	38
10	1	1	63	35	1	1	36	15	0	0	10	37
11	2	1	58	39	2	1	31	22	1	1	6	36
12	3	2	15	45	2	2	37	24	0	1	4	30
13	4	2	51	43	2	2	31	22	1	1	7	34
14	3	4	51	42	2	4	30	23	0	1	6	36
15	5	5	50	40	2	5	29	25	1	3	6	29

Percentage of times Objective (4) is optimal for different strategies												
#	(H) + Strategy											
	Pure strategies				Pure and greedy strategies							
	(MV)	(PM)	(MD)*	(MG)	(MV)	(PM)	(MD)*	(MG)	(G1)	(G2)	(G3)	(G4)
1	0	3	66	31	0	3	66	31	0	0	0	0
2	0	1	84	15	0	0	52	6	2	0	33	7
3	0	0	81	19	0	0	43	8	0	0	33	16
4	0	0	78	22	0	0	44	10	0	0	32	14
5	0	0	78	22	0	0	44	9	0	0	33	14
6	0	0	74	26	0	0	40	13	0	0	29	18
7	0	0	60	40	0	0	28	28	0	0	29	15
8	0	0	42	58	0	0	21	42	0	0	18	19
9	0	0	27	73	0	0	15	54	0	0	18	13
10	0	0	34	66	0	0	21	43	0	0	15	21
11	1	0	44	55	0	0	29	35	0	1	16	19
12	1	1	47	51	0	0	26	28	1	0	11	34
13	1	0	46	53	0	0	22	28	1	0	14	35
14	0	3	51	46	0	0	30	24	0	0	14	32
15	3	2	46	49	1	0	24	28	1	0	16	30

TABLE 5. Continuation of the same experiment from Table 4. In strategy (MD), once ill-conditioning is encountered, we now switch to (MG) instead.

theoretical and numerical results are presented. We also explain briefly how our ideas can be implemented in a path-based mountain pass algorithm.

Much still needs to be done. For example, formulas similar to that of Theorem 7.1 describing the convergence of the various implementations will be helpful in fine tuning the algorithm. Lastly, our algorithm is only "local" in the sense that it works well only when the iterates are close to the saddle point where the quadratic approximation becomes more accurate. An emphasis should also be placed on designing a "global" algorithms.

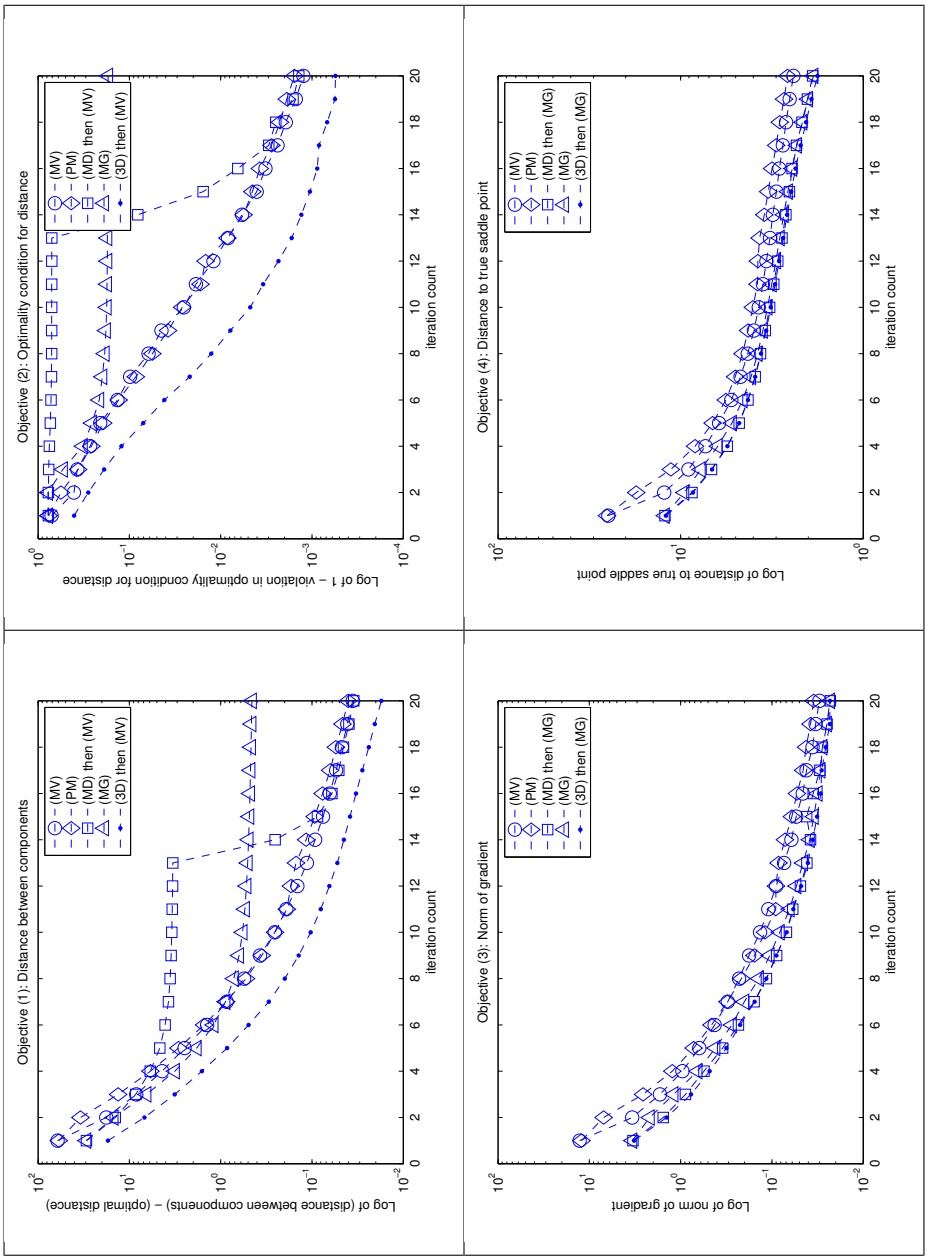

FIGURE 4. Performance of various strategies for a random example. For objectives (1) and (2), the strategies (MD) and (3D) switch to (MV) once ill-conditioning in encountered. For objectives (3) and (4), the strategies (MD) and (3D) switch to (MG) instead if ill-conditioning is encountered.

Acknowledgement. We thank Jiahao Chen for discussions that led to the main ideas in this paper and for references to the literature on computing saddle points, and also to James Renegar for some helpful discussions.

REFERENCES

1. A. Ambrosetti and P.H. Rabinowitz, *Dual variational methods in critical point theory and applications*, J. Funct. Anal. **14** (1973), 349–381.

2. G. Arioli, V. Barutello, and S. Terracini, *A new branch of mountain pass solutions for the choreographical 3-body problem*, Commun. Math. Phys. **268** (2006), 439–463.
3. M. Benzi, G.H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta. Numer. **14** (2005), 1–137.
4. Y.S. Choi and P.J. McKenna, *A mountain pass method for the numerical solution of semilinear elliptic equations*, Nonlinear Anal. **20:4** (1993), 417–437.
5. Zhonghai Ding, David Costa, and Goong Chen, *A high-linking algorithm for sign-changing solutions of semilinear elliptic equations*, Nonlinear Anal. **38** (1999), 151–172.
6. Y. Feng, *The study of nonlinear flexings in a floating beam by variational methods. oscillations in nonlinear systems: Applications and numerical aspects*, J. Comput. Appl. Math. **52:1–3** (1994), 91–112.
7. D. Glotov and P.J. McKenna, *Numerical mountain pass solutions of Ginzburg-Landau type equations*, Comm. Pure Appl. Anal. **7:6** (2008), 1345–1359.
8. G. Henkelman, G. Jóhannesson, and H. Jónsson, *Methods for finding saddle points and minimum energy paths*, Progress in Theoretical Chemistry and Physics (S.D. Schwartz, ed.), Kluwer, Amsterdam, 2000, pp. 269–300.
9. J. Horák, *Constrained mountain pass algorithm for the numerical solution of semilinear elliptic problems*, Numer. Math. **98** (2004), 251–276.
10. J. Horák, G.J. Lord, and M.A. Peletier, *Cylinder buckling: The mountain pass as an organizing center*, SIAM J. Appl. Math. **66:5** (2006), 1793–1824.
11. Youssef Jabri, *The mountain pass theorem*, Cambridge, 2003.
12. A.C. Lazer and P.J. McKenna, *Nonlinear flexings in a periodically forced floating beam*, Math. Method. Appl. Sci. **14:1** (1991), 1–33.
13. A.S. Lewis and C.H.J. Pang, *Level set methods for finding critical points of mountain pass type*, Nonlinear Anal. **74:12** (2011), 4058–4082.
14. Yongxin Li and Jianxin Zhou, *A minimax method for finding multiple critical points and its applications to semilinear PDEs*, SIAM J. Sci. Comput. **23:3** (2001), 840–865.
15. J. Mawhin and M. Willem, *Critical point theory and Hamiltonian systems*, Springer, New York, 1989.
16. R.A. Miron and K.A. Fichthorn, *The step and slide method for finding saddle points on multidimensional potential surfaces*, J. Chem. Phys. **115:9** (2001), 8742–8747.
17. J.J. Moré and T.S. Munson, *Computing mountain passes and transition states*, Math. Program. **100:1** (2004), 151–182.
18. C.H.J. Pang, *Level set methods for finding saddle points of general Morse index*.
19. P.H. Rabinowitz, *Some critical point theorems and applications to semilinear elliptic partial differential equations*, Ann. Scuola Norm. Sup. Pisa **5** (1977), 412–424.
20. ———, *Minimax methods in critical point theory with applications to differential equations*, CBMS regional Conf. Ser. in Math, 65, AMS, Providence, RI, 1986.
21. M. Schechter, *Linking methods in critical point theory*, Birkhäuser, Boston, 1999.
22. M. Struwe, *Variational methods*, 4th ed., Springer, New York, 2008.
23. Zhi-Qiang Wang and Jianxin Zhou, *A local minimax-Newton's method for finding critical points with symmetries*, SIAM J. Num. Anal. **42** (2004), 1745–1759.
24. ———, *An efficient and stable method for computing multiple saddle points with symmetries*, SIAM J. Num. Anal. **43** (2005), 891–907.
25. M. Willem, *Minimax theorems*, Birkhäuser, Boston, 1996.
26. Xudong Yao and Jianxin Zhou, *A local minimax characterization of computing multiple nonsmooth saddle critical points*, Math. Program., Ser. B **104** (2005), 749–760.

Current address: Massachusetts Institute of Technology, Department of Mathematics, 2-334, 77 Massachusetts Avenue, Cambridge MA 02139-4307.

E-mail address: chj2pang@mit.edu