

# Path following in the exact penalty method of convex programming

Hua Zhou · Kenneth Lange

Received: 24 July 2013 / Published online: 18 February 2015  
© Springer Science+Business Media New York 2015

**Abstract** Classical penalty methods solve a sequence of unconstrained problems that put greater and greater stress on meeting the constraints. In the limit as the penalty constant tends to  $\infty$ , one recovers the constrained solution. In the exact penalty method, squared penalties are replaced by absolute value penalties, and the solution is recovered for a finite value of the penalty constant. In practice, the kinks in the penalty and the unknown magnitude of the penalty constant prevent wide application of the exact penalty method in nonlinear programming. In this article, we examine a strategy of path following consistent with the exact penalty method. Instead of performing optimization at a single penalty constant, we trace the solution as a continuous function of the penalty constant. Thus, path following starts at the unconstrained solution and follows the solution path as the penalty constant increases. In the process, the solution path hits, slides along, and exits from the various constraints. For quadratic programming, the solution path is piecewise linear and takes large jumps from constraint to constraint. For a general convex program, the solution path is piecewise smooth, and path following operates by numerically solving an ordinary differential equation segment by segment. Our diverse applications to (a) projection onto a convex set, (b) nonnegative least squares, (c) quadratically constrained quadratic programming, (d) geometric programming, and (e) semidefinite programming illustrate the mechanics

---

**Electronic supplementary material** The online version of this article (doi:[10.1007/s10589-015-9732-x](https://doi.org/10.1007/s10589-015-9732-x)) contains supplementary material, which is available to authorized users.

---

H. Zhou (✉)  
Department of Statistics, North Carolina State University, Raleigh, NC 27695-8203, USA  
e-mail: [hua\\_zhou@ncsu.edu](mailto:hua_zhou@ncsu.edu)

K. Lange  
Departments of Biomathematics, Human Genetics and Statistics, University of California,  
Los Angeles, CA 90095-1766, USA

and potential of path following. The final detour to image denoising demonstrates the relevance of path following to regularized estimation in inverse problems. In regularized estimation, one follows the solution path as the penalty constant decreases from a large value.

**Keywords** Constrained convex optimization · Exact penalty · Geometric programming · Ordinary differential equation · Quadratically constrained quadratic programming · Regularization · Semidefinite programming

**Mathematics Subject Classification** 65K05 · 90C25

## 1 Introduction

Penalties and barriers are both potent devices for solving constrained optimization problems [1–6]. The general idea is to replace hard constraints by penalties or barriers and then exploit the well-oiled machinery for solving unconstrained problems. Penalty methods operate on the exterior of the feasible region and barrier methods on the interior. The strength of a penalty or barrier is determined by a tuning constant. In classical penalty methods, a single global tuning constant is gradually sent to  $\infty$ ; in barrier methods, it is gradually sent to 0. Either strategy generates a sequence of solutions that converges in practice to the solution of the original constrained optimization problem.

Barrier methods are now generally conceded to offer a better approach to solving convex programs than penalty methods. Application of log barriers and carefully controlled versions of Newton's method make it possible to follow the central path reliably and quickly to the constrained minimum [1]. Nonetheless, penalty methods should not be ruled out. Augmented Lagrangian methods [7] and exact penalty methods [4] are potentially competitive with interior point methods for smooth convex programming problems. Both methods have the advantage that the solution of the constrained problem kicks in for a finite value of the penalty constant. This avoids problems of ill conditioning as the penalty constant tends to  $\infty$ .

The disadvantage of exact penalties over traditional quadratic penalties is lack of differentiability of the penalized objective function. In the current paper, we argue that this impediment can be finessed by path following. Our path following method starts at the unconstrained solution and follows the solution path as the penalty constant increases. In the process, the solution path hits, exits, and slides along the various constraint boundaries. The path itself is piecewise smooth with kinks at the boundary hitting and escape times. One advances along the path by numerically solving a differential equation for the Lagrange multipliers of the penalized problem. In the special case of quadratic programming with affine constraints, the solution path is piecewise linear, and one can easily anticipate entire path segments [8]. This special case is intimately related to the linear complementarity problem [9] in optimization theory.

Homotopy (continuation) methods for the solution of nonlinear equations and optimization problems have been pursued for many years and enjoyed a variety of successes [4, 10–12]. To our knowledge, however, there has been no exploration of path

following as an implementation of the exact penalty method. Our modest goal here is to assess the feasibility and versatility of exact path following for constrained optimization. Comparing its performance to existing methods, particularly the interior point method, is probably best left for later, more practically oriented papers. In our experience, coding the path algorithm is straightforward in Matlab. The rich numerical resources of MATLAB include differential equation solvers that alert the user when certain events such as constraint hitting and escape occur.

The rest of the paper is organized as follows. Section 2 briefly reviews the exact penalty method for optimization and investigates sufficient conditions for uniqueness and continuity of the solution path. Section 3 derives the path following strategy for general convex programs, with particular attention to the special cases of quadratic programming and convex optimization with affine constraints. Section 4 presents various applications of the path algorithm. Our most elaborate example demonstrates the relevance of path following to regularized estimation. The particular problem treated, image denoising, is typical of many inverse problems in applied mathematics and statistics [13]. In such problems one follows the solution path as the penalty constant decreases. Finally, Sect. 5 discusses the limitations of the path algorithm and hints at future generalizations.

## 2 Exact penalty methods

In this paper we consider the convex programming problem of minimizing the convex objective function  $f(\mathbf{x})$  subject to  $r$  affine equality constraints  $g_i(\mathbf{x}) = 0$  and  $s$  convex inequality constraints  $h_j(\mathbf{x}) \leq 0$ . We will further assume that  $f(\mathbf{x})$  and the  $h_j(\mathbf{x})$  are twice differentiable. The differential  $df(\mathbf{x})$  is the row vector of partial derivatives of  $f(\mathbf{x})$ ; the gradient  $\nabla f(\mathbf{x})$  is the transpose of  $df(\mathbf{x})$ . The second differential  $d^2 f(\mathbf{x})$  is the Hessian matrix of second partial derivatives of  $f(\mathbf{x})$ . Similar conventions hold for the differentials of the constraint functions.

Exact penalty methods [4,5] minimize the surrogate function

$$\mathcal{E}_\rho(\mathbf{x}) = f(\mathbf{x}) + \rho \sum_{i=1}^r |g_i(\mathbf{x})| + \rho \sum_{j=1}^s \max\{0, h_j(\mathbf{x})\}. \tag{1}$$

This definition of  $\mathcal{E}_\rho(\mathbf{x})$  is meaningful regardless of whether the contributing functions are convex. If the program is convex, then  $\mathcal{E}_\rho(\mathbf{x})$  is itself convex. It is interesting to compare  $\mathcal{E}_\rho(\mathbf{x})$  to the Lagrangian function

$$\mathcal{L}(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^r \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^s \mu_j h_j(\mathbf{x}),$$

which captures the behavior of  $f(\mathbf{x})$  near the optimum. At a constrained local minimum  $\mathbf{y}$ , the Lagrangian satisfies the stationarity condition  $\nabla \mathcal{L}(\mathbf{y}) = \mathbf{0}$ ; its inequality multipliers  $\mu_j$  are nonnegative and satisfy the complementary slackness conditions  $\mu_j h_j(\mathbf{y}) = 0$ . In an exact penalty method one takes

$$\rho > \max\{|\lambda_1|, \dots, |\lambda_r|, \mu_1, \dots, \mu_s\}. \tag{2}$$

This choice creates the favorable circumstances

$$\begin{aligned} \mathcal{L}(\mathbf{x}) &\leq \mathcal{E}_\rho(\mathbf{x}) \quad \text{for all } \mathbf{x} \\ \mathcal{L}(\mathbf{z}) &\leq f(\mathbf{z}) = \mathcal{E}_\rho(\mathbf{z}) \quad \text{for all feasible } \mathbf{z} \\ \mathcal{L}(\mathbf{y}) &= f(\mathbf{y}) = \mathcal{E}_\rho(\mathbf{y}) \quad \text{for } \mathbf{y} \text{ optimal} \end{aligned}$$

with profound consequences. As the next proposition proves, minimizing  $\mathcal{E}_\rho(\mathbf{x})$  is effective in minimizing  $f(\mathbf{x})$  subject to the constraints.

**Proposition 1** *Suppose the objective function  $f(\mathbf{x})$  and the constraint functions are twice differentiable and satisfy the Lagrange multiplier rule at the local minimum  $\mathbf{y}$ . If inequality (2) holds and  $\mathbf{v}^*d^2\mathcal{L}(\mathbf{y})\mathbf{v} > 0$  for every vector  $\mathbf{v} \neq \mathbf{0}$  satisfying  $dg_i(\mathbf{y})\mathbf{v} = 0$  and  $dh_j(\mathbf{y})\mathbf{v} \leq 0$  for all active inequality constraints, then  $\mathbf{y}$  furnishes an unconstrained local minimum of  $\mathcal{E}_\rho(\mathbf{x})$ . For a convex program satisfying Slater’s constraint qualification and inequality (2),  $\mathbf{y}$  is a minimum of  $\mathcal{E}_\rho(\mathbf{x})$  if and only if  $\mathbf{y}$  is a minimum of  $f(\mathbf{x})$  subject to the constraints. No differentiability assumptions are required for convex programs.*

*Proof* The conditions imposed on the quadratic form  $\mathbf{v}^*d^2\mathcal{L}(\mathbf{y})\mathbf{v}$  are well-known sufficient conditions for a local minimum. Theorems 6.9 and 7.21 of the reference [5] prove all of the foregoing assertions.

As previously stressed, the exact penalty method turns a constrained optimization problem into an unconstrained minimization problem. Furthermore, in contrast to the quadratic penalty method [4, Section17.1], the constrained solution in the exact method is achieved for a finite value of  $\rho$ . Despite these advantages, minimizing the surrogate function  $\mathcal{E}_\rho(\mathbf{x})$  is complicated. For one thing, it is no longer globally differentiable. For another, one must minimize  $\mathcal{E}_\rho(\mathbf{x})$  along an increasing sequence  $\rho_n$  because the Lagrange multipliers (2) are usually unknown in advance. These hurdles have prevented wide application of exact penalty methods in convex programming.

As a prelude to our derivation of the path following algorithm for convex programs, we record several properties of  $\mathcal{E}_\rho(\mathbf{x})$  that mitigate the failure of differentiability.

**Proposition 2** *The surrogate function  $\mathcal{E}_\rho(\mathbf{x})$  is increasing in  $\rho$ . Furthermore,  $\mathcal{E}_\rho(\mathbf{x})$  is strictly convex for one  $\rho > 0$  if and only if it is strictly convex for all  $\rho > 0$ . Likewise, it is coercive for one  $\rho > 0$  if and only if is coercive for all  $\rho > 0$ . Finally, if  $f(\mathbf{x})$  is strictly convex (or coercive), then all  $\mathcal{E}_\rho(\mathbf{x})$  are strictly convex (or coercive).*

*Proof* The first assertion is obvious. For the second assertion, consider more generally a finite family  $u_1(\mathbf{x}), \dots, u_q(\mathbf{x})$  of convex functions, and suppose a linear combination  $\sum_{k=1}^q c_k u_k(\mathbf{x})$  with positive coefficients is strictly convex. It suffices to prove that any other linear combination  $\sum_{k=1}^q b_k u_k(\mathbf{x})$  with positive coefficients is strictly convex. For any two points  $\mathbf{x} \neq \mathbf{y}$  and any scalar  $\alpha \in (0, 1)$ , we have

$$u_k[\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}] \leq \alpha u_k(\mathbf{x}) + (1 - \alpha)u_k(\mathbf{y}). \tag{3}$$

Since  $\sum_{k=1}^q c_k u_k(\mathbf{x})$  is strictly convex, strict inequality must hold for at least one  $k$ . Hence, multiplying inequality (3) by  $b_k$  and adding gives

$$\sum_{k=1}^q b_k u_k[\alpha \mathbf{x} + (1 - \alpha)\mathbf{y}] < \alpha \sum_{k=1}^q b_k u_k(\mathbf{x}) + (1 - \alpha) \sum_{k=1}^q b_k u_k(\mathbf{y}).$$

The third assertion follows from the fact that a convex function is coercive if and only if its restriction to each half-line is coercive [14, Proposition 3.2.2]. Given this result, suppose  $\mathcal{E}_\rho(\mathbf{x})$  is coercive, but  $\mathcal{E}_{\rho^*}(\mathbf{x})$  is not coercive. Then there exists a point  $\mathbf{x}$ , a direction  $\mathbf{v}$ , and a sequence of scalars  $t_n$  tending to  $\infty$  such that  $\mathcal{E}_{\rho^*}(\mathbf{x} + t_n \mathbf{v})$  is bounded above. This requires the sequence  $f(\mathbf{x} + t_n \mathbf{v})$  and each of the sequences  $|g_i(\mathbf{x} + t_n \mathbf{v})|$  and  $\max\{0, h_j(\mathbf{x} + t_n \mathbf{v})\}$  to remain bounded above. But in this circumstance the sequence  $\mathcal{E}_\rho(\mathbf{x} + t_n \mathbf{v})$  also remains bounded above. The final two assertions are obvious.

### 3 The path following algorithm

In this section, we take a different point of view. Instead of minimizing  $\mathcal{E}_\rho(\mathbf{x})$  for an increasing sequence  $\rho_n$ , we study how the solution  $\mathbf{x}(\rho)$  changes continuously with  $\rho$  and devise a path following strategy starting from  $\rho = 0$ . For some finite value of  $\rho$ , the path locks in on the solution of the original convex program. In regularized statistical estimation and inverse problems, the primary goal is to select relevant predictors rather than to find a constrained solution. Thus, the entire solution path commands more interest than any single point along it [8, 13, 15–17]. Although our theory will focus on constrained estimation, readers should bear in mind this second application area of path following.

The path algorithm relies critically on the first order optimality condition that characterizes the optimum point of the convex function  $\mathcal{E}_\rho(\mathbf{y})$ .

**Proposition 3** *For a convex program, a point  $\mathbf{x} = \mathbf{x}(\rho)$  minimizes the function  $\mathcal{E}_\rho(\mathbf{y})$  if and only if  $\mathbf{x}$  satisfies the stationarity condition*

$$\mathbf{0} = \nabla f(\mathbf{x}) + \rho \sum_{i=1}^r s_i \nabla g_i(\mathbf{x}) + \rho \sum_{j=1}^s t_j \nabla h_j(\mathbf{x}) \tag{4}$$

for coefficient sets  $\{s_i\}_{i=1}^r$  and  $\{t_j\}_{j=1}^s$ . These sets can be characterized as

$$s_i \in \begin{cases} \{-1\} & g_i(\mathbf{x}) < 0 \\ [-1, 1] & g_i(\mathbf{x}) = 0 \\ \{1\} & g_i(\mathbf{x}) > 0 \end{cases} \quad \text{and} \quad t_j \in \begin{cases} \{0\} & h_j(\mathbf{x}) < 0 \\ [0, 1] & h_j(\mathbf{x}) = 0 \\ \{1\} & h_j(\mathbf{x}) > 0 \end{cases}. \tag{5}$$

At most one point achieves the minimum of  $\mathcal{E}_\rho(\mathbf{y})$  for a given  $\rho$  when  $\mathcal{E}_\rho(\mathbf{y})$  is strictly convex.

*Proof* According to Fermat’s rule,  $\mathbf{x}$  minimizes  $\mathcal{E}_\rho(\mathbf{y})$  if and only if  $\mathbf{0}$  belongs to the subdifferential  $\partial\mathcal{E}_\rho(\mathbf{x})$  of  $\mathcal{E}_\rho(\mathbf{y})$ . To derive the subdifferential displayed in Eqs. (4) and (5), one applies the addition and chain rules of the convex calculus. The sets defining the possible values of  $s_i$  and  $t_j$  are the subdifferentials of the functions  $|s|$  and  $t_+ = \max\{t, 0\}$ , respectively. For more details see Theorem 3.5 and ancillary material in the book [5]. Finally, it is well known that strict convexity guarantees a unique minimum.

To speak coherently of solution paths, one must validate the existence, uniqueness, and continuity of the solution  $\mathbf{x}(\rho)$  to the system of Eq. (1). Uniqueness follows from strict convexity as already noted. Existence and continuity are more subtle.

**Proposition 4** *If  $\mathcal{E}_\rho(\mathbf{y})$  is strictly convex and coercive, then the solution path  $\mathbf{x}(\rho)$  of Eq. (1) exists and is continuous in  $\rho$ . If the gradient vectors  $\{\nabla g_i(\mathbf{x}) : g_i(\mathbf{x}) = 0\} \cup \{\nabla h_j(\mathbf{x}) : h_j(\mathbf{x}) = 0\}$  of the active constraints are linearly independent at  $\mathbf{x}(\rho)$  for  $\rho > 0$ , then the coefficients  $s_i(\rho)$  and  $t_j(\rho)$  are unique and continuous near  $\rho$  as well.*

*Proof* The special case of a least squares objective with affine constraints appears in our previous paper [8]. The following proof applies to general convex programming. In accord with Proposition 2, we assume that either  $f(\mathbf{x})$  is strictly convex and coercive or restrict our attention to the open interval  $(0, \infty)$ . Consider a subinterval  $[a, b]$  and fix a point  $\mathbf{x}$  in the common domain of the functions  $\mathcal{E}_\rho(\mathbf{y})$ . The coercivity of  $\mathcal{E}_a(\mathbf{y})$  and the inequalities

$$\mathcal{E}_a[\mathbf{x}(\rho)] \leq \mathcal{E}_\rho[\mathbf{x}(\rho)] \leq \mathcal{E}_\rho(\mathbf{x}) \leq \mathcal{E}_b(\mathbf{x})$$

demonstrate that the solution vector  $\mathbf{x}(\rho)$  is bounded over  $[a, b]$ . To prove continuity, suppose that it fails for a given  $\rho \in [a, b]$ . Then there exists an  $\epsilon > 0$  and a sequence  $\rho_n$  tending to  $\rho$  such  $\|\mathbf{x}(\rho_n) - \mathbf{x}(\rho)\|_2 \geq \epsilon$  for all  $n$ . Since  $\mathbf{x}(\rho_n)$  is bounded, we can pass to a subsequence if necessary and assume that  $\mathbf{x}(\rho_n)$  converges to some point  $\mathbf{y}$ . Taking limits in the inequality  $\mathcal{E}_{\rho_n}[\mathbf{x}(\rho_n)] \leq \mathcal{E}_{\rho_n}(\mathbf{x})$  demonstrates that  $\mathcal{E}_\rho(\mathbf{y}) \leq \mathcal{E}_\rho(\mathbf{x})$  for all  $\mathbf{x}$ . Because  $\mathbf{x}(\rho)$  is unique, we reach the contradictory conclusions  $\|\mathbf{y} - \mathbf{x}(\rho)\|_2 \geq \epsilon$  and  $\mathbf{y} = \mathbf{x}(\rho)$ .

Verification of the second claim is deferred to permit further discussion of path following. The claim says that an active constraint ( $g_i(\mathbf{x}) = 0$  or  $h_j(\mathbf{x}) = 0$ ) remains active until its coefficient hits an endpoint of its subdifferential. Because the solution path is, in fact, piecewise smooth, one can follow the coefficient path by numerically solving an ordinary differential equation (ODE).

Our path following algorithm works segment-by-segment. Along the path we keep track of the following index sets

$$\begin{aligned} \mathcal{N}_E &= \{i : g_i(\mathbf{x}) < 0\} & \mathcal{N}_I &= \{j : h_j(\mathbf{x}) < 0\} \\ \mathcal{Z}_E &= \{i : g_i(\mathbf{x}) = 0\} & \mathcal{Z}_I &= \{j : h_j(\mathbf{x}) = 0\} \\ \mathcal{P}_E &= \{i : g_i(\mathbf{x}) > 0\} & \mathcal{P}_I &= \{j : h_j(\mathbf{x}) > 0\} \end{aligned} \tag{6}$$

determined by the signs of the constraint functions. For the sake of simplicity, assume that at the beginning of the current segment  $s_i$  does not equal  $-1$  or  $1$  when  $i \in \mathcal{Z}_E$  and  $t_j$  does not equal  $0$  or  $1$  when  $j \in \mathcal{Z}_I$ . In other words, the coefficients of the active constraints occur on the interior of their subdifferentials. Let us show in this circumstance that the solution path can be extended in a smooth fashion. Our plan of attack is to reparameterize by the Lagrange multipliers for the active constraints. Thus, set  $\lambda_i = \rho s_i$  for  $i \in \mathcal{Z}_E$  and  $\omega_j = \rho t_j$  for  $j \in \mathcal{Z}_I$ . The multipliers satisfy  $-\rho < \lambda_i < \rho$  and  $0 < \omega_j < \rho$ . The stationarity condition now reads

$$\begin{aligned} \mathbf{0} = & \nabla f(\mathbf{x}) - \rho \sum_{i \in \mathcal{N}_E} \nabla g_i(\mathbf{x}) + \rho \sum_{i \in \mathcal{P}_E} \nabla g_i(\mathbf{x}) + \rho \sum_{j \in \mathcal{P}_I} \nabla h_j(\mathbf{x}) \\ & + \sum_{i \in \mathcal{Z}_E} \lambda_i \nabla g_i(\mathbf{x}) + \sum_{j \in \mathcal{Z}_I} \omega_j \nabla h_j(\mathbf{x}). \end{aligned}$$

To this we concatenate the constraint equations  $0 = g_i(\mathbf{x})$  for  $i \in \mathcal{Z}_E$  and  $0 = h_j(\mathbf{x})$  for  $j \in \mathcal{Z}_I$ .

For convenience now define

$$U_{\mathcal{Z}}(\mathbf{x}) = \begin{bmatrix} dg_{\mathcal{Z}_E}(\mathbf{x}) \\ dh_{\mathcal{Z}_I}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{u}_{\bar{\mathcal{Z}}}(\mathbf{x}) = - \sum_{i \in \mathcal{N}_E} \nabla g_i(\mathbf{x}) + \sum_{i \in \mathcal{P}_E} \nabla g_i(\mathbf{x}) + \sum_{j \in \mathcal{P}_I} \nabla h_j(\mathbf{x}).$$

In this notation the stationarity equation can be recast as

$$\mathbf{0} = \nabla f(\mathbf{x}) + \rho \mathbf{u}_{\bar{\mathcal{Z}}}(\mathbf{x}) + U_{\mathcal{Z}}^t(\mathbf{x}) \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\omega} \end{bmatrix}.$$

Under the assumption that the matrix  $U_{\mathcal{Z}}(\mathbf{x})$  has full row rank, one can solve for the Lagrange multipliers in the form

$$\begin{bmatrix} \boldsymbol{\lambda}_{\mathcal{Z}_E} \\ \boldsymbol{\omega}_{\mathcal{Z}_I} \end{bmatrix} = -[U_{\mathcal{Z}}(\mathbf{x})U_{\mathcal{Z}}^t(\mathbf{x})]^{-1}U_{\mathcal{Z}}(\mathbf{x})[\nabla f(\mathbf{x}) + \rho \mathbf{u}_{\bar{\mathcal{Z}}}(\mathbf{x})]. \tag{7}$$

Hence, the multipliers are unique. Continuity of the multipliers is a consequence of the continuity of the solution vector  $\mathbf{x}(\rho)$  and all functions in sight on the right-hand side of Eq. (7). This observation completes the proof of Proposition 4.

Collectively the stationarity and active constraint equations can be written as the vector equation  $\mathbf{0} = k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho)$ . To solve for  $\mathbf{x}$ ,  $\boldsymbol{\lambda}$  and  $\boldsymbol{\omega}$  in terms of  $\rho$ , we apply the implicit function theorem [18, 19]. This requires calculating the differential of  $k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho)$  with respect to the underlying dependent variables  $\mathbf{x}$ ,  $\boldsymbol{\lambda}$ , and  $\boldsymbol{\omega}$  and the independent variable  $\rho$ . Because the equality constraints are affine, a brief calculation gives

$$\begin{aligned} \partial_{\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}} k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho) &= \begin{bmatrix} d^2 f(\mathbf{x}) + \rho \sum_{j \in \mathcal{P}_1} d^2 h_j(\mathbf{x}) + \sum_{j \in \mathcal{Z}_1} \omega_j d^2 h_j(\mathbf{x}) & \mathbf{U}_{\mathcal{Z}}^t(\mathbf{x}) \\ \mathbf{U}_{\mathcal{Z}}(\mathbf{x}) & \mathbf{0} \end{bmatrix} \\ \partial_\rho k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho) &= \begin{pmatrix} \mathbf{u}_{\bar{\mathcal{Z}}}(\mathbf{x}) \\ \mathbf{0} \end{pmatrix}. \end{aligned}$$

The matrix  $\partial_{\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}} k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho)$  is nonsingular when its upper-left block is positive definite and its lower-left block has full row rank [20, Proposition 1.3.2]. Given that it is nonsingular, the implicit function theorem applies, and we can in principle solve for  $\mathbf{x}$ ,  $\boldsymbol{\lambda}$  and  $\boldsymbol{\omega}$  in terms of  $\rho$ . More importantly, the implicit function theorem supplies the derivative

$$\frac{d}{d\rho} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda}_{\mathcal{Z}_E} \\ \boldsymbol{\omega}_{\mathcal{Z}_1} \end{bmatrix} = -\partial_{\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}} k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho)^{-1} \partial_\rho k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho), \tag{8}$$

which is the key to path following. We summarize our findings in the next proposition. A similar proof appears in our application paper [21], where we implement path following in a convex clustering context.

**Proposition 5** *Suppose the surrogate function  $\mathcal{E}_\rho(\mathbf{y})$  is strictly convex and coercive. If at the point  $\mathbf{x}(\rho_0)$  the matrix  $\partial_{\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}} k(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \rho)$  is nonsingular and the coefficient of each active constraints occurs on the interior of its subdifferential, then the solution path  $\mathbf{x}(\rho)$  and Lagrange multipliers  $\boldsymbol{\lambda}(\rho)$  and  $\boldsymbol{\omega}(\rho)$  satisfy the differential Eq. (8) in the vicinity of  $\mathbf{x}(\rho_0)$ .*

In practice one traces the solution path along the current time segment until either an inactive constraint becomes active or the coefficient of an active constraint hits the boundary of its subdifferential. The earliest hitting time or escape time over all constraints determines the duration of the current segment. When the hitting time for an inactive constraint occurs first, we move the constraint to the appropriate active set  $\mathcal{Z}_E$  or  $\mathcal{Z}_1$  and keep the other constraints in place. Similarly, when the escape time for an active constraint occurs first, we move the constraint to the appropriate inactive set and keep the other constraints in place. In the second scenario, if  $s_i$  hits the value  $-1$ , then we move  $i$  to  $\mathcal{N}_E$ ; If  $s_i$  hits the value  $1$ , then we move  $i$  to  $\mathcal{P}_E$ . Similar comments apply when a coefficient  $t_j$  hits  $0$  or  $1$ . Once this move is executed, we commence path following along the new segment. Path following continues until for sufficiently large  $\rho$ , the sets  $\mathcal{N}_E$ ,  $\mathcal{P}_E$ , and  $\mathcal{P}_1$  are exhausted,  $\mathbf{u}_{\bar{\mathcal{Z}}} = \mathbf{0}$ , and the solution vector  $\mathbf{x}(\rho)$  stabilizes.

Algorithm 1 summarizes the above path following strategy. It makes the convenient assumptions that there are no active constraints ( $\mathcal{Z}_E = \mathcal{Z}_1 = \emptyset$ ) at the unconstrained solution  $\mathbf{x}(0)$  and that the hitting and/or escape times do not occur simultaneously for any  $\rho > 0$ . Our previous paper [8] suggests remedies in the very rare situations where these assumptions fail.

Path following simplifies considerably in two special cases. Consider convex quadratic programming with objective function  $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^t \mathbf{A} \mathbf{x} + \mathbf{b}^t \mathbf{x}$  and equality constraints  $\mathbf{V} \mathbf{x} = \mathbf{d}$  and inequality constraints  $\mathbf{W} \mathbf{x} \leq \mathbf{e}$ , where  $\mathbf{A}$  is positive semi-definite. The exact penalized objective function becomes

```

1 Initialize:  $\rho = 0, \mathbf{x}(0) = \arg \min f(\mathbf{x})$ , assume  $\mathcal{Z}_E = \mathcal{Z}_I = \emptyset$  and initialize sets  $\mathcal{N}_E, \mathcal{P}_E, \mathcal{N}_I$  and  $\mathcal{P}_I$ 
  according to the prescriptions in Eq. (6)
2 repeat
3   Solve ODE
4     
$$\frac{d}{d\rho} \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda}_{\mathcal{Z}_E} \\ \boldsymbol{\omega}_{\mathcal{Z}_I} \end{pmatrix} = - \left( d^2 f(\mathbf{x}) + \rho \sum_{j \in \mathcal{P}_I} d^2 h_j(\mathbf{x}) + \sum_{j \in \mathcal{Z}_I} \omega_j d^2 h_j(\mathbf{x}) \right) \mathbf{U}_{\mathcal{Z}}^t(\mathbf{x})^{-1} \begin{pmatrix} \mathbf{u}_{\mathcal{Z}}(\mathbf{x}) \\ \mathbf{0} \end{pmatrix}$$

5   until (i)  $g_i(\mathbf{x}) = 0$  for some  $i \in \mathcal{P}_E \cup \mathcal{N}_E$ , (ii)  $h_j(\mathbf{x}) = 0$  for some  $j \in \mathcal{P}_I$ , (iii)  $\lambda_i \in \{\rho, -\rho\}$  for
     some  $i \in \mathcal{Z}_E$ , or (iv)  $w_j \in \{0, \rho\}$  for some  $j \in \mathcal{Z}_I$ 
6   if events (i) or (iii) then
7     | Update sets  $\mathcal{N}_E, \mathcal{Z}_E, \mathcal{P}_E$ 
8   else if events (ii) or (iv) then
9     | Update sets  $\mathcal{N}_I, \mathcal{Z}_I, \mathcal{P}_I$ 
10  end
11 until  $\mathcal{N}_E = \mathcal{P}_E = \mathcal{P}_I = \emptyset$ 

```

**Algorithm 1:** Exact path algorithm for convex programming.

$$\mathcal{E}_\rho(\mathbf{x}) = \frac{1}{2} \mathbf{x}^t \mathbf{A} \mathbf{x} + \mathbf{b}^t \mathbf{x} + \rho \sum_{i=1}^s |v_i^t \mathbf{x} - d_i| + \rho \sum_{j=1}^t (\mathbf{w}_j^t \mathbf{x} - e_j)_+.$$

Since both the equality and inequality constraints are affine, their second derivatives vanish. Both  $\mathbf{U}_{\mathcal{Z}}$  and  $\mathbf{u}_{\mathcal{Z}}$  are constant on the current path segment, and the path  $\mathbf{x}(\rho)$  satisfies

$$\frac{d}{d\rho} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda}_{\mathcal{Z}_E} \\ \boldsymbol{\omega}_{\mathcal{Z}_I} \end{bmatrix} = - \begin{pmatrix} \mathbf{A} & \mathbf{U}_{\mathcal{Z}}^t \\ \mathbf{U}_{\mathcal{Z}} & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{u}_{\mathcal{Z}} \\ \mathbf{0} \end{pmatrix}. \tag{9}$$

This implies that the solution path  $\mathbf{x}(\rho)$  is piecewise linear [8]. When  $\mathbf{V}$  is the identity matrix,  $\mathbf{d} = \mathbf{0}$ , and there are no inequality constraints, Algorithm 1 reduces to the well-known path algorithm for lasso regression [15,16]. Tibshirani and Taylor [17] construct a piecewise-linear path algorithm for the generalized lasso problem with a general  $\mathbf{V}$  and  $\mathbf{d} = \mathbf{0}$ . Their approach seems to circumvent the need for linear independence of the active constraints.

On the next rung on the ladder of generality are convex programs with affine constraints. For the exact surrogate

$$\mathcal{E}_\rho(\mathbf{x}) = f(\mathbf{x}) + \rho \sum_{i=1}^s |v_i^t \mathbf{x} - d_i| + \rho \sum_{j=1}^t (\mathbf{w}_j^t \mathbf{x} - e_j)_+,$$

the matrix  $\mathbf{U}_{\mathcal{Z}}$  and vector  $\mathbf{u}_{\mathcal{Z}}$  are still constant along a path segment. The relevant differential equation becomes

$$\frac{d}{d\rho} \begin{bmatrix} \mathbf{x} \\ \lambda_{\mathcal{Z}_E} \\ \omega_{\mathcal{Z}_I} \end{bmatrix} = - \begin{pmatrix} d^2 f(\mathbf{x}) \mathbf{U}_{\mathcal{Z}}^t \\ \mathbf{U}_{\mathcal{Z}} \\ \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{u}_{\mathcal{Z}} \\ \mathbf{0} \end{pmatrix}. \tag{10}$$

There are two approaches for computing the right-hand side of Eq. (10). When  $\mathbf{A} = d^2 f(\mathbf{x})$  is positive definite and  $\mathbf{B} = \mathbf{U}_{\mathcal{Z}}$  has full row rank, the relevant inverse amounts to

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^t \\ \mathbf{B} & \mathbf{0} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B}^t [\mathbf{B} \mathbf{A}^{-1} \mathbf{B}^t]^{-1} \mathbf{B} \mathbf{A}^{-1} & \mathbf{A}^{-1} \mathbf{B}^t [\mathbf{B} \mathbf{A}^{-1} \mathbf{B}^t]^{-1} \\ [\mathbf{B} \mathbf{A}^{-1} \mathbf{B}^t]^{-1} \mathbf{B} \mathbf{A}^{-1} & -[\mathbf{B} \mathbf{A}^{-1} \mathbf{B}^t]^{-1} \end{pmatrix}.$$

The numerical cost of computing the inverse scales as  $O(n^3) + O(|\mathcal{Z}|^3)$ . When  $d^2 f(\mathbf{x})$  is a constant, the inverse is computed once. Sequentially updating it for different active sets  $\mathcal{Z}$  is then conveniently organized around the sweep operator of computational statistics [8]. For a general convex function  $f(\mathbf{x})$ , every time  $\mathbf{x}$  changes, the inverse must be recomputed. This burden plus the cost of computing the entries of  $d^2 f(\mathbf{x})$  slow the path algorithm for general convex problems. At the same time, Sect. 4 demonstrates how a greater level of generality opens up many new applications. See our recent paper [21] on convex clustering for yet another application of path following.

In many applications  $f(\mathbf{x})$  is convex but not necessarily strictly convex. By Proposition 2,  $\mathcal{E}_\rho$  is also convex for all  $\rho > 0$ . Furthermore,  $\mathcal{E}_\rho$  may admit a unique global minimum for large  $\rho$ . In this case the proof of Proposition 4 still applies, and the solution path is continuous. One can circumvent problems in inverting  $d^2 f(\mathbf{x})$  by reparameterizing [4]. For the sake of simplicity, suppose that all of the constraints are affine and that  $\mathbf{U}_{\mathcal{Z}}$  has full row rank. The set of points  $\mathbf{x}$  satisfying the active constraints can be written as  $\mathbf{x} = \mathbf{w} + \mathbf{Y}\mathbf{y}$ , where  $\mathbf{w}$  is a particular solution,  $\mathbf{y}$  is free to vary, and the columns of  $\mathbf{Y} \in \mathbb{R}^{n \times (n-|\mathcal{Z}|)}$  span the null space of  $\mathbf{U}_{\mathcal{Z}}$  and hence are orthogonal to the rows of  $\mathbf{U}_{\mathcal{Z}}$ . Under the null space reparameterization,  $\frac{d\mathbf{x}}{d\rho} = \mathbf{Y} \frac{d\mathbf{y}}{d\rho}$ . Furthermore,

$$\begin{aligned} \mathbf{Y}^t d^2 f(\mathbf{x}) \mathbf{Y} &= d_y^2 f(\mathbf{w} + \mathbf{Y}\mathbf{y}) \\ \mathbf{Y}^t \mathbf{u}_{\mathcal{Z}}(\mathbf{x}) &= \nabla_y \left[ -\rho \sum_{i \in \mathcal{N}_E} g_i(\mathbf{w} + \mathbf{Y}\mathbf{y}) \right. \\ &\quad \left. + \rho \sum_{i \in \mathcal{P}_E} g_i(\mathbf{w} + \mathbf{Y}\mathbf{y}) + \rho \sum_{j \in \mathcal{P}_I} h_j(\mathbf{w} + \mathbf{Y}\mathbf{y}) \right]. \end{aligned}$$

It follows that Eq. (10) becomes

$$\begin{aligned} \frac{d}{d\rho} \mathbf{y} &= -[\mathbf{Y}^t d^2 f(\mathbf{x}) \mathbf{Y}]^{-1} \mathbf{Y}^t \mathbf{u}_{\mathcal{Z}} \\ \frac{d}{d\rho} \mathbf{x} &= -\mathbf{Y} [\mathbf{Y}^t d^2 f(\mathbf{x}) \mathbf{Y}]^{-1} \mathbf{Y}^t \mathbf{u}_{\mathcal{Z}}. \end{aligned} \tag{11}$$

Differentiating Eq. (7) gives the multiplier derivatives

$$\frac{d}{d\rho} \begin{bmatrix} \lambda_{\mathcal{Z}_E} \\ \omega_{\mathcal{Z}_I} \end{bmatrix} = -(\mathbf{U}_{\mathcal{Z}} \mathbf{U}_{\mathcal{Z}}^t)^{-1} \mathbf{U}_{\mathcal{Z}} \left( d^2 f(\mathbf{x}) \frac{d\mathbf{x}}{d\rho} + \mathbf{u}_{\mathcal{Z}} \right). \tag{12}$$

The obvious advantage of using Eq. (11) is that the matrix  $\mathbf{Y}^t d^2 f(\mathbf{x}) \mathbf{Y}$  can be non-singular when  $d^2 f(\mathbf{x})$  is singular. The computational cost of evaluating the right-hand sides of Eqs. (11) and (12) is  $O([n - |\mathcal{Z}|]^3) + O(|\mathcal{Z}|^3)$ . When  $n - |\mathcal{Z}|$  and  $|\mathcal{Z}|$  are small compared to  $n$ , this is an improvement over the cost  $O(n^3) + O(|\mathcal{Z}|^3)$  of computing the right-hand side of Eq. (8). Balanced against this gain is the requirement of finding a basis of the null space of  $\mathbf{U}_{\mathcal{Z}}$ . Fortunately, the matrix  $\mathbf{Y}$  is constant over each path segment and in practice can be computed by taking the QR decomposition of the active constraint matrix  $\mathbf{U}_{\mathcal{Z}}$ . At each kink of the solution path, either one constraint enters  $\mathcal{Z}$  or one leaves. Therefore,  $\mathbf{Y}$  can be sequentially computed by standard updating and downdating formulas [4,22]. Which ODE (8) or (11) is preferable depends on the specific application. When the loss function  $f(\mathbf{x})$  is not strictly convex, for example when the number of parameters exceeds the number of cases in regression, path following requires the ODE (11). Interested readers are referred to the book [4] for a more extended discussion of range-space versus null-space optimization methods.

For a general convex program, one can employ Euler’s update

$$\begin{bmatrix} \mathbf{x}(\rho + \Delta\rho) \\ \lambda(\rho + \Delta\rho) \\ \omega(\rho + \Delta\rho) \end{bmatrix} = \begin{bmatrix} \mathbf{x}(\rho) \\ \lambda(\rho) \\ \omega(\rho) \end{bmatrix} + \Delta\rho \frac{d}{d\rho} \begin{bmatrix} \mathbf{x}(\rho) \\ \lambda(\rho) \\ \omega(\rho) \end{bmatrix}$$

to advance the solution of the ODE (8). Euler’s formula may be inaccurate for  $\Delta\rho$  large. One can correct it by fixing  $\rho$  and performing one step of Newton’s method to re-connect with the solution path. This amounts to replacing the position-multiplier vector by

$$\begin{bmatrix} \mathbf{x} \\ \lambda \\ \omega \end{bmatrix} - \partial_{\mathbf{x}, \lambda, \omega} k(\mathbf{x}, \lambda, \omega, \rho)^{-1} k(\mathbf{x}, \lambda, \omega, \rho).$$

In practice, it is certainly easier and probably safer to rely on ODE packages such as the ODE45 function in MATLAB to advance the solution of the ODE.

### 4 Examples of path following

Our examples are intended to illuminate the mechanics of path following and showcase its versatility. As we emphasized in the introduction, we forgo comparisons with other methods. Comparisons depend heavily on programming details and problem choices, so a premature study might well be misleading.

#### Example 1 Projection onto the Feasible Region

Finding a feasible point is the initial stage in many convex programs. Dykstra’s algorithm [23,24] was designed precisely to solve the problem of projecting an exterior

point onto the intersection of a finite number of closed convex sets. The projection problem also yields to our generic path following algorithm. Consider the toy example of projecting a point  $\mathbf{b} \in \mathbb{R}^2$  onto the intersection of the closed unit ball and the closed half space  $x_1 \geq 0$  [18]. This is equivalent to solving

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) = \frac{1}{2} \|\mathbf{x} - \mathbf{b}\|^2 \\ \text{subject to} \quad & h_1(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|^2 - \frac{1}{2} \leq 0, \quad h_2(\mathbf{x}) = -x_1 \leq 0. \end{aligned}$$

The relevant gradients and second differentials are

$$\begin{aligned} \nabla f(\mathbf{x}) &= \mathbf{x} - \mathbf{b}, \quad \nabla h_1(\mathbf{x}) = \mathbf{x}, \quad \nabla h_2(\mathbf{x}) = -\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ d^2 f(\mathbf{x}) &= d^2 h_1(\mathbf{x}) = \mathbf{I}_2, \quad d^2 h_2(\mathbf{x}) = \mathbf{0}. \end{aligned}$$

Path following starts from the unconstrained solution  $\mathbf{x}(0) = \mathbf{b}$ ; the direction of movement is determined by formula (8). For  $\mathbf{x} \in \{\mathbf{x} : \|\mathbf{x}\|^2 > 1, x_1 > 0\}$ , the path

$$\frac{d}{d\rho} \mathbf{x} = -[(1 + \rho)\mathbf{I}_2]^{-1} \mathbf{x} = -\frac{1}{1 + \rho} \mathbf{x}$$

heads toward the origin. For  $\mathbf{x} \in \{\mathbf{x} : |x_2| > 1, x_1 = 0\}$ , the path

$$\frac{d}{d\rho} \begin{pmatrix} \mathbf{x} \\ \omega_2 \end{pmatrix} = -\begin{pmatrix} 1 + \rho & 0 & -1 \\ 0 & 1 + \rho & 0 \\ -1 & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = -\frac{1}{1 + \rho} \begin{pmatrix} 0 \\ x_2 \\ 0 \end{pmatrix}$$

also heads toward the origin. For  $\mathbf{x} \in \{\mathbf{x} : \|\mathbf{x}\|^2 > 1, x_1 < 0\}$ , the path

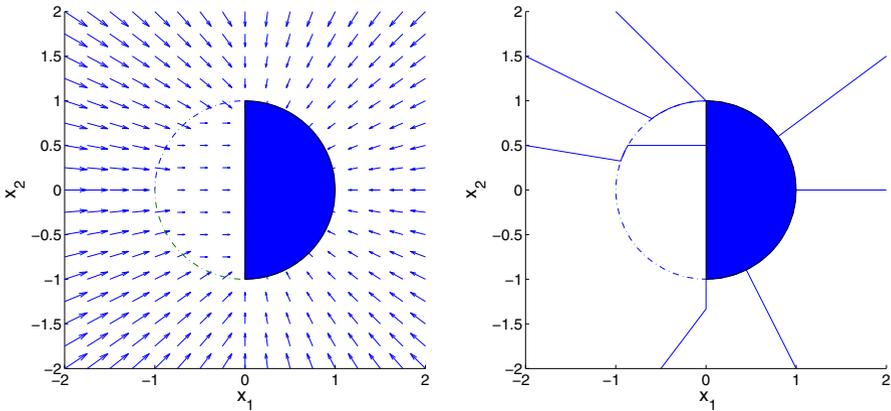
$$\frac{d}{d\rho} \mathbf{x} = -[(1 + \rho)\mathbf{I}_2]^{-1} \begin{pmatrix} x_1 - 1 \\ x_2 \end{pmatrix} = -\frac{1}{1 + \rho} \begin{pmatrix} x_1 - 1 \\ x_2 \end{pmatrix}.$$

heads toward the point  $(1, 0)^t$ . For  $\mathbf{x} \in \{\mathbf{x} : \|\mathbf{x}\|^2 = 1, x_1 < 0\}$ , the path

$$\frac{d}{d\rho} \begin{pmatrix} \mathbf{x} \\ \omega_1 \end{pmatrix} = -\begin{pmatrix} 1 + \omega_1 & 0 & x_1 \\ 0 & 1 + \omega_1 & x_2 \\ x_1 & x_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{x_2^2}{1 + \omega_1} \\ \frac{x_1 x_2}{1 + \omega_1} \\ -x_1 \end{pmatrix}$$

is tangent to the circle. Finally, for  $\mathbf{x} \in \{\mathbf{x} : \|\mathbf{x}\|^2 < 1, x_1 < 0\}$ , the path

$$\frac{d}{d\rho} \mathbf{x} = -\mathbf{I}_2^{-1} \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



**Fig. 1** Projection to the positive half disk. *Left* Derivatives at  $\rho = 0$  for projection onto the half disc. *Right* Projection trajectories from various initial points

heads toward the  $x_2$ -axis. The left panel of Fig. 1 plots the vector field  $\frac{d}{d\rho}\mathbf{x}$  at the time  $\rho = 0$ . The right panel shows the solution path for projection from the points  $(-2, 0.5)^t$ ,  $(-2, 1.5)^t$ ,  $(-1, 2)^t$ ,  $(2, 1.5)^t$ ,  $(2, 0)^t$ ,  $(1, 2)^t$ , and  $(-0.5, -2)^t$  onto the feasible region. In projecting the point  $\mathbf{b} = (-1, 2)^t$  onto  $(0, 1)^t$ , the ODE45 solver of MATLAB evaluates derivatives at 19 different time points. Dykstra’s algorithm by comparison takes about 30 iterations to converge [18].

*Example 2* Nonnegative Least Squares (NNLS) and Nonnegative Matrix Factorization (NNMF)

Non-negative matrix factorization (NNMF) is an alternative to principle component analysis and is useful in modeling, compressing, and interpreting nonnegative data such as observational counts and images. The articles [25–27] discuss in detail estimation algorithms and statistical applications of NNMF. The basic idea is to approximate an  $m \times n$  data matrix  $\mathbf{X} = (x_{ij})$  with nonnegative entries by a product  $\mathbf{V}\mathbf{W}$  of two low rank matrices  $\mathbf{V} = (v_{ik})$  and  $\mathbf{W} = (w_{kj})$  with nonnegative entries. Here  $\mathbf{V}$  and  $\mathbf{W}$  are  $m \times r$  and  $r \times n$  respectively, with  $r \ll \min\{m, n\}$ . One version of NNMF minimizes the criterion

$$f(\mathbf{V}, \mathbf{W}) = \|\mathbf{X} - \mathbf{V}\mathbf{W}\|_F^2 = \sum_i \sum_j \left(x_{ij} - \sum_k v_{ik}w_{kj}\right)^2, \tag{13}$$

where  $\|\cdot\|_F$  denotes the Frobenius norm. In a typical imaging problem,  $m$  (number of images) might range from  $10^3$  to  $10^4$ ,  $n$  (number of pixels per image) might surpass  $10^4$ , and a rank  $r = 50$  approximation might adequately capture  $\mathbf{X}$ .

Minimization of the objective function (13) is nontrivial because it is not jointly convex in  $\mathbf{V}$  and  $\mathbf{W}$ . Multiple local minima are possible. The well-known multiplicative algorithm [26,27] enjoys the descent property, but it is not guaranteed to converge to even a local minimum [25]. An alternative algorithm that exhibits better convergence

is alternating least squares (ALS). In updating  $W$  with  $V$  fixed, ALS solves the  $n$  separated nonnegative least square (NLS) problems

$$\min_{w_j} \|x_j - V w_j\|_2^2 \quad \text{subject to } w_j \geq 0, \tag{14}$$

where  $x_j$  and  $w_j$  denote the  $j$ -th columns of the corresponding matrices. Similarly, in updating  $V$  with  $W$  fixed, ALS solves  $m$  separated NNLS problems. The unconstrained solution  $W(0) = (V^T V)^{-1} V^T X$  of  $W$  for fixed  $V$  requires just one QR decomposition of  $V$  or one Cholesky decomposition of  $V^T V$ . The exact path algorithm for solving the subproblem problem (14) commences with  $W(0)$ . If  $W(\rho)$  stabilizes with just a few zeros, then the path algorithm ends quickly and is extremely efficient. For a NNLS problem, the path is piecewise linear, and one can straightforwardly project the path to the next hitting or escape time using the sweep operator [8]. Figure 2 shows a typical piecewise linear path for a problem with  $r = 50$  predictors. Each projection to the next event requires  $2r^2$  flops. The number of path segments (events) roughly scales as the number of negative components in the unconstrained solution.

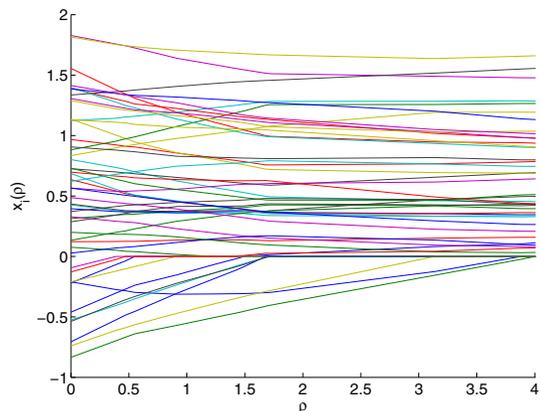
*Example 3* Quadratically Constrained Quadratic Programming (QCQP)

Example 1 is a special case of quadratically constrained quadratic programming (QCQP). In convex QCQP [1, Section4.4], one minimizes a convex quadratic function over an intersection of ellipsoids and affine subspaces. Mathematically, this amounts to the problem

$$\begin{aligned} \text{minimize} \quad & f(x) = \frac{1}{2}x^T P_0 x + b_0^T x + c_0 \\ \text{subject to} \quad & g_i(x) = a_i^T x - d_i = 0, \quad i = 1, \dots, r \\ & h_j(x) = \frac{1}{2}x^T P_j x + b_j^T x + c_j \leq 0, \quad j = 1, \dots, s, \end{aligned}$$

where  $P_0$  is a positive definite matrix and the  $P_j$  are positive semidefinite matrices. Our algorithm starts with the unconstrained minimum  $x(0) = -P_0^{-1}b_0$  and proceeds along the path determined by the derivative

**Fig. 2** Piecewise linear paths of the regression coefficients for a NNLS problem with 50 predictors



$$\frac{d}{d\rho} \begin{bmatrix} \mathbf{x} \\ \lambda_{\mathcal{Z}_E} \\ \omega_{\mathcal{Z}_1} \end{bmatrix} = - \left( \mathbf{P}_0 + \rho \sum_{j \in \mathcal{P}_1} \mathbf{P}_j + \sum_{j \in \mathcal{Z}_1} \omega_j \mathbf{P}_j \mathbf{U}_{\mathcal{Z}}^t(\mathbf{x}) \right)^{-1} \begin{pmatrix} \mathbf{u}_{\mathcal{Z}}(\mathbf{x}) \\ \mathbf{0} \end{pmatrix},$$

where  $\mathbf{U}_{\mathcal{Z}}(\mathbf{x})$  has rows  $\mathbf{a}_i^t$  for  $i \in \mathcal{Z}_E$  and  $(\mathbf{P}_j \mathbf{x} + \mathbf{b}_j)^t$  for  $j \in \mathcal{Z}_1$ , and

$$\mathbf{u}_{\mathcal{Z}}(\mathbf{x}) = - \sum_{i \in \mathcal{N}_E} \mathbf{a}_i + \sum_{i \in \mathcal{P}_E} \mathbf{a}_i + \sum_{i \in \mathcal{P}_1} (\mathbf{P}_j \mathbf{x} + \mathbf{b}_j).$$

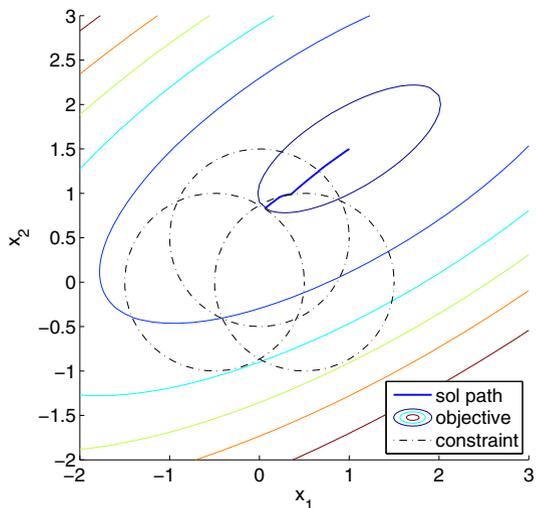
Affine inequality constraints can be accommodated by setting one or more of the  $\mathbf{P}_j$  equal to  $\mathbf{0}$ .

As a numerical illustration, consider the bivariate problem

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) = \frac{1}{2}x_1^2 + x_2^2 - x_1x_2 + \frac{1}{2}x_1 - 2x_2 \\ \text{subject to} \quad & h_1(\mathbf{x}) = \left(x_1 - \frac{1}{2}\right)^2 + x_2^2 - 1 \leq 0 \\ & h_2(\mathbf{x}) = \left(x_1 + \frac{1}{2}\right)^2 + x_2^2 - 1 \leq 0 \\ & h_3(\mathbf{x}) = x_1^2 + \left(x_2 - \frac{1}{2}\right)^2 - 1 \leq 0. \end{aligned} \tag{15}$$

Here the feasible region is given by the intersection of three disks with centers  $(0.5, 0)^t$ ,  $(-0.5, 0)^t$ , and  $(0, 0.5)^t$ , respectively, and a common radius of 1. Figure 3 displays the solution trajectory. Starting from the unconstrained minimum  $\mathbf{x}(0) = (1, 1.5)^t$ , it hits, slides along, and exits two circles before its journey ends at the constrained minimum  $(0.059, 0.829)^t$ . The ODE45 solver of MATLAB evaluates derivatives at 72 time points along the path.

**Fig. 3** Trajectory of the exact penalty path algorithm for a QCQP problem (15). The solid lines are the contours of the objective function  $f(\mathbf{x})$ . The dashed lines are the contours of the constraint functions  $h_j(\mathbf{x})$



*Example 4* Geometric Programming

As a branch of convex optimization theory, geometric programming stands just behind linear and quadratic programming in importance [28–31]. It has applications in chemical equilibrium problems [32], structural mechanics [29], digit circuit design [33], maximum likelihood estimation [34], stochastic processes [35], and a host of other subjects [28,29]. Geometric programming deals with posynomials, which are functions of the form

$$f(\mathbf{x}) = \sum_{\alpha \in S} c_{\alpha} \prod_{i=1}^n x_i^{\alpha_i} = \sum_{\alpha \in S} c_{\alpha} e^{\alpha^t \mathbf{y}} = f(\mathbf{y}). \tag{16}$$

In the left-hand definition of this equivalent pair of definitions, the index set  $S \subset \mathbb{R}^n$  is finite, and all coefficients  $c_{\alpha}$  and all components  $x_1, \dots, x_n$  of the argument  $\mathbf{x}$  of  $f(\mathbf{x})$  are positive. The possibly fractional powers  $\alpha_i$  corresponding to a particular  $\alpha$  may be positive, negative, or zero. For instance,  $x_1^{-1} + 2x_1^3 x_2^{-2}$  is a posynomial on  $\mathbb{R}^2$ . In geometric programming, one minimizes a posynomial  $f(\mathbf{x})$  subject to posynomial inequality constraints of the form  $h_j(\mathbf{x}) \leq 1$  for  $1 \leq j \leq s$ . In some versions of geometric programming, equality constraints of monomial type are permitted [28]. The right-hand definition in Eq. (16) invokes the exponential reparameterization  $x_i = e^{y_i}$ . This simple transformation has the advantage of rendering a geometric program convex. In fact, any posynomial  $f(\mathbf{y})$  in the exponential parameterization is log-convex and therefore convex. The concise representations

$$\nabla f(\mathbf{y}) = \sum_{\alpha \in S} c_{\alpha} e^{\alpha^t \mathbf{y}} \alpha, \quad d^2 f(\mathbf{y}) = \sum_{\alpha \in S} c_{\alpha} e^{\alpha^t \mathbf{y}} \alpha \alpha^t$$

of the gradient and the second differential are helpful in both theory and computation.

Without loss of generality, one can repose geometric programming as

$$\begin{aligned} &\text{minimize} && \ln f(\mathbf{y}) \\ &\text{subject to} && \ln g_i(\mathbf{y}) = 0, \quad 1 \leq i \leq r \\ &&& \ln h_j(\mathbf{y}) \leq 0, \quad 1 \leq j \leq s, \end{aligned} \tag{17}$$

where  $f(\mathbf{y})$  and the  $h_j(\mathbf{y})$  are posynomials and the equality constraints  $\ln g_i(\mathbf{y})$  are affine. In this exponential parameterization setting, it is easy to state necessary and sufficient conditions for strict convexity and coerciveness.

**Proposition 6** *The objective function  $f(\mathbf{y})$  in the geometric program (17) is strictly convex if and only if the subspace spanned by the vectors  $\{\alpha\}_{\alpha \in S}$  is all of  $\mathbb{R}^n$ ;  $f(\mathbf{y})$  is coercive if and only if the polar cone  $\{\mathbf{z} : \mathbf{z}^t \alpha \leq 0 \text{ for all } \alpha \in S\}$  reduces to the origin  $\mathbf{0}$ . Equivalently,  $f(\mathbf{y})$  is coercive if the origin  $\mathbf{0}$  belongs to the interior of the convex hull of the set  $S$ .*

*Proof* These claims are proved in detail in our paper [36].

According to Propositions 1 and 4, the strict convexity and coerciveness of  $f(\mathbf{y})$  guarantee the uniqueness and continuity of the solution path in  $\mathbf{y}$ . This in turn implies the uniqueness and continuity of the solution path in the original parameter vector  $\mathbf{x}$ . The path directions are related by the chain rule

$$\frac{d}{d\rho} x_i(\rho) = \frac{dx_i}{dy_i} \frac{d}{d\rho} y_i(\rho) = x_i \frac{d}{d\rho} y_i(\rho).$$

As a concrete example, consider the problem

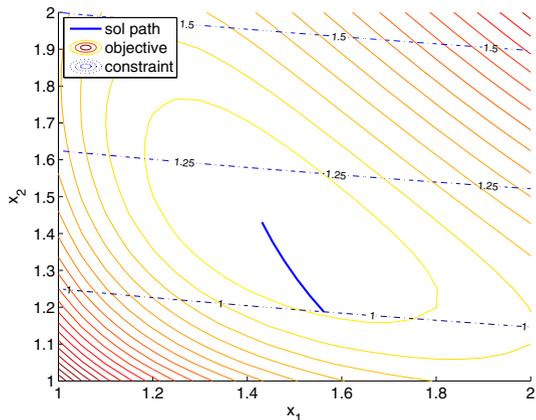
$$\begin{aligned} &\text{minimize} && x_1^{-3} + 3x_1^{-1}x_2^{-2} + x_1x_2 && (18) \\ &\text{subject to} && \frac{1}{6}x_1^{1/2} + \frac{2}{3}x_2 \leq 1, \quad x_1 > 0, \quad x_2 > 0. \end{aligned}$$

It is easy to check that the vectors  $\{(-3, 0)^t, (-1, -2)^t, (1, 1)^t\}$  span  $\mathbb{R}^2$  and generate a convex hull strictly containing the origin  $\mathbf{0}$ . Therefore,  $f(\mathbf{y})$  is strictly convex and coercive. It achieves its unconstrained minimum at the point  $\mathbf{x}(0) = (\sqrt[3]{6}, \sqrt[3]{6})^t$ , or equivalently  $\mathbf{y}(0) = (\ln 6/5, \ln 6/5)^t$ . To solve the constrained minimization problem, we follow the path dictated by the revised geometric program (17). Figure 4 plots the trajectory from the unconstrained solution to the constrained solution in the original  $\mathbf{x}$  variables. The solid lines in the figure represent the contours of the objective function  $f(\mathbf{x})$ , and the dashed lines represent the contours of the constraint function  $h(\mathbf{x})$ . The ODE45 solver of MATLAB evaluates derivatives at seven time points along the path.

*Example 5 Semidefinite Programming (SDP)*

The linear semidefinite programming problem [37] consists in minimizing the trace function  $\mathbf{X} \mapsto \text{tr}(\mathbf{C}\mathbf{X})$  over the cone of positive semidefinite matrices  $S_+^n$  subject to the linear constraints  $\text{tr}(\mathbf{A}_i\mathbf{X}) = b_i$  for  $1 \leq i \leq p$ . Here  $\mathbf{C}$  and the  $\mathbf{A}_i$  are assumed symmetric. According to Sylvester’s criterion, the constraint  $\mathbf{X} \in S_+^n$  involves a complicated system of inequalities involving nonconvex functions. One way of cutting through this morass is to focus on the minimum eigenvalue  $v_1(\mathbf{X})$  of  $\mathbf{X}$ .

**Fig. 4** Trajectory of the exact penalty path algorithm for the geometric programming problem (18). The solid lines are the contours of the objective function  $f(\mathbf{x})$ . The dashed lines are the contours of the constraint function  $h(\mathbf{x})$  at levels 1, 1.25, and 1.5



Because the function  $-v_1(\mathbf{X})$  is convex, one can enforce positive semidefiniteness by requiring  $-v_1(\mathbf{X}) \leq 0$ . Thus, the linear semidefinite programming problem is a convex program in the standard functional form.

It simplifies matters enormously to assume that  $v_1(\mathbf{X})$  has multiplicity 1. Let  $\mathbf{u}$  be the unique, up to sign, unit eigenvector corresponding to  $v_1(\mathbf{X})$ . The matrix  $\mathbf{X}$  is parameterized by the entries of its lower triangle. With these conventions, the following formulas

$$\begin{aligned}
 -\frac{\partial}{\partial x_{ij}} v_1(\mathbf{X}) &= -\mathbf{u}^t \frac{\partial}{\partial x_{ij}} \mathbf{X} \mathbf{u} & (19) \\
 -\frac{\partial^2}{\partial x_{ij} \partial x_{kl}} v_1(\mathbf{X}) &= -\mathbf{u}^t \frac{\partial}{\partial x_{ij}} \mathbf{X} (\nu_1 \mathbf{I} - \mathbf{X})^- \frac{\partial}{\partial x_{kl}} \mathbf{X} \mathbf{u} \\
 &\quad -\mathbf{u}^t \frac{\partial}{\partial x_{kl}} \mathbf{X} (\nu_1 \mathbf{I} - \mathbf{X})^- \frac{\partial}{\partial x_{ij}} \mathbf{X} \mathbf{u} \\
 &= -2\mathbf{u}^t \frac{\partial}{\partial x_{ij}} \mathbf{X} (\nu_1 \mathbf{I} - \mathbf{X})^- \frac{\partial}{\partial x_{kl}} \mathbf{X} \mathbf{u} & (20)
 \end{aligned}$$

for the first and second partial derivatives of  $-v_1(\mathbf{X})$  are well known [19]. Here the matrix  $(\nu_1 \mathbf{I} - \mathbf{X})^-$  is the Moore-Penrose inverse of  $\nu_1 \mathbf{I} - \mathbf{X}$ . The partial derivative of  $\mathbf{X}$  with respect to its lower triangular entry  $x_{ij}$  equals  $\mathbf{E}_{ij} + 1_{\{i \neq j\}} \mathbf{E}_{ji}$ , where  $\mathbf{E}_{ij}$  is the matrix consisting of all 0's excepts for a 1 in position  $(i, j)$ . Note that  $\mathbf{u}^t \mathbf{E}_{ij} = u_i \mathbf{e}_j^t$ , and  $\mathbf{E}_{kl} \mathbf{u} = u_l \mathbf{e}_k$  for the standard unit vectors  $\mathbf{e}_j$  and  $\mathbf{e}_k$ . The second partial derivatives of  $\mathbf{X}$  vanish. The Moore-Penrose inverse is most easily expressed in terms of the spectral decomposition of  $\mathbf{X}$ . If we denote the  $i$ th eigenvalue of  $\mathbf{X}$  by  $v_i$  and the corresponding  $i$ th unit eigenvector by  $\mathbf{u}_i$ , then we have

$$(\mathbf{X} - \nu_1 \mathbf{I})^- = \sum_{i>1} \frac{1}{v_i - \nu_1} \mathbf{u}_i \mathbf{u}_i^t.$$

Finally, the formulas

$$\begin{aligned}
 \text{tr}(\mathbf{A}_i \mathbf{X}) - b_i &= \sum_k (\mathbf{A}_i)_{kk} x_{kk} + 2 \sum_k \sum_{l<k} (\mathbf{A}_i)_{kl} x_{kl} - b_i \\
 \frac{\partial}{\partial x_{kl}} [\text{tr}(\mathbf{A}_i \mathbf{X}) - b_i] &= (\mathbf{A}_i)_{kl} + 1_{\{k \neq l\}} (\mathbf{A}_i)_{lk}
 \end{aligned}$$

express the linear constraints and their partial derivatives in terms of the lower triangular entries of  $\mathbf{X}$ .

Initiating path following is problematic because  $\text{tr}(\mathbf{C}\mathbf{X})$  has minimum  $-\infty$ . A good strategy is to amend the surrogate function  $\mathcal{E}_\rho(\mathbf{x})$  by adding the term  $\frac{\epsilon(\rho)}{2} \|\mathbf{X}\|_F^2$ , where  $\epsilon(\rho)$  is a smooth positive function that decreases to 0. Taking  $\epsilon(\rho) = e^{-c\rho}$  for  $c$  positive works well in practice. The new surrogate function  $\text{tr}(\mathbf{C}\mathbf{X}) + \frac{\epsilon(\rho)}{2} \|\mathbf{X}\|_F^2$  is strictly convex and possesses a unique minimum for all  $\rho \geq 0$ . In view of the identities

$\|X\|_F^2 = \sum_i \sum_j x_{ij}^2$  and  $\text{tr}(CX) = \sum_i \sum_j c_{ij}x_{ij}$  for  $X = (x_{ij})$  and  $C = (c_{ij})$ , the initial condition  $X(0) = -\epsilon(0)^{-1}C$  is straightforward to deduce.

Path following must be modified to accommodate the new surrogate function. In the notation of [19], let  $x = v(X)$  be the  $\frac{1}{2}n(n+1)$  vector obtained from  $\text{vec}(X)$  by eliminating all supradiagonal entries, and let  $D$  be the  $n^2 \times \frac{1}{2}n(n+1)$  duplication matrix satisfying  $\text{vec}(X) = Dx$ . Applying the chain rule to the obvious identities  $\|X\|_F^2 = xD^tDx$  and  $\text{tr}(CX) = \text{vec}(C)^tDx$ , one can extend the derivation of Proposition 5 and prove that

$$\begin{aligned} & \frac{d}{d\rho} \begin{bmatrix} x \\ \lambda_{Z_E} \\ \omega_{Z_T} \end{bmatrix} \\ &= - \left[ \begin{array}{c} \epsilon(\rho)D^tD - \omega_{Z_T}d^2v_1(x)1_{\{v_1(X)=0\}} \\ U_Z \\ \mathbf{0} \end{array} \right]^{-1} \\ & \times \left( \begin{array}{c} \frac{d\epsilon(\rho)}{d\rho}D^tDx - \sum_{i \in N_E} D^t \text{vec}(A_i) + \sum_{i \in P_E} D^t \text{vec}(A_i) - \nabla v_1(x)1_{\{v_1(X)<0\}} \\ \mathbf{0} \end{array} \right). \end{aligned}$$

Path following proceeds until all constraints are satisfied and  $\epsilon(\rho)$  is negligible.

For didactic purposes, considering the problem of minimizing  $\text{tr}(CX)$  subject to

$$\text{tr}(A_1X) = 1, \quad \text{tr}(A_2X) = 2, \quad \text{and} \quad X \in S_+^2,$$

where

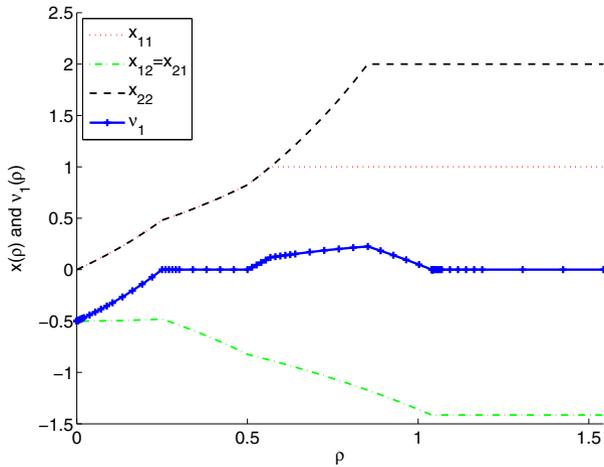
$$C = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \quad A_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Figure 5 displays the solution paths of the entries  $x_{ij}$  of  $X$  and the minimum eigenvalue  $v_1$ . Here we use  $\epsilon(\rho) = e^{-\rho}$ . The path starts with  $X(0) = -C$ , hits, slides along, and exits various constraints, and ends at the constrained solution  $\begin{pmatrix} 1 & -\sqrt{2} \\ -\sqrt{2} & 2 \end{pmatrix}$ .

*Example 6 Image Denoising*

Image analysis is another fertile field for path following. Here we explore how to restore or enhance images by removing noise. This example differs from previous examples in that the fully constrained solution is trivial. The solution path itself is the object of interest. Suppose that  $w = (w_{ij}) \in \mathbb{R}^{m \times n}$  represents the recorded gray levels across a 2D array of pixels from a noisy image with true gray levels  $u = (u_{ij})$ . The well-known denoising model of Rudin-Osher-Fatemi (ROF) [38] minimizes the total variation regularized least squares criterion

$$\begin{aligned} & \frac{1}{2} \|w - u\|_2^2 + \rho \text{TV}(u) \\ &= \frac{1}{2} \sum_{ij} (w_{ij} - u_{ij})^2 + \rho \sum_{i,j} \sqrt{(u_{i+1,j} - u_{ij})^2 + (u_{i,j+1} - u_{ij})^2}. \end{aligned} \tag{21}$$



**Fig. 5** Solution path of a semidefinite programming example

The total variation penalty serves to smooth the reconstructed image and preserve its edges. A similar effect can be achieved by replacing the isotropic penalty  $TV(\mathbf{u})$  by the anisotropic penalty

$$TV_1(\mathbf{u}) = \sum_{i,j} (|u_{i+1,j} - u_{ij}| + |u_{i,j+1} - u_{ij}|). \tag{22}$$

In this example we focus on path following for the anisotropic penalty and a more general convex loss function  $f(\mathbf{u})$ . The objective function is now

$$f(\mathbf{u}) + \rho \|D\mathbf{u}\|_1. \tag{23}$$

For instance, the amended loss function  $f(\mathbf{u}) = \frac{1}{2} \|\mathbf{w} - \mathbf{K}\mathbf{u}\|_2^2$  with a Gaussian or motion blurring matrix  $\mathbf{K}$  is appropriate in many imaging problems. Poisson count data are relevant to image reconstruction in X-ray and positron tomography [20] and to image denoising in certain circumstances [39]. With Poisson noise, the least squares criterion is replaced by a negative loglikelihood. The difference matrix  $\mathbf{D}$  captures the  $\ell_1$  penalty (22). Note that the matrices  $\mathbf{w}$  and  $\mathbf{u}$  are now viewed as vectors. For an  $m \times n$  2D image, the difference matrix  $\mathbf{D}$  has  $2mn - m - n$  rows (penalties) and  $mn$  columns (pixels). This matrix is very sparse, with just  $2(2mn - m - n)$  nonzero entries equal to  $\pm 1$ . When  $m$  and  $n$  are both at least 2,  $\mathbf{D}$  has more rows than columns and a reduced column rank of  $mn - 1$ .

For sufficiently large  $\rho$ , the minimum of the objective functions (21) reduces to a constant vector (blank image) equal to the average value  $\bar{w}$  of the  $w_{ij}$ . The goal of image denoising is to find a  $\rho$  such that the recovered image is judged satisfactory by visual inspection or other more quantitative criteria. Notable computational advances in solving this problem include Chambolle’s algorithm [40] and split Bregman iteration [41]. These methods minimize the objective functions (21) and (23) for a fixed value of

$\rho$ . The web site of UCLA's Computational and Applied Math Group summarizes the most recent progress in this area. In reality, outer iterations are almost always required to tune the parameter  $\rho$ . Path following is an attractive option because it provides the whole solution path at about the same computational cost as recovering the solution for an individual  $\rho$ .

Although it is tempting to minimize the criterion (23) by path following, the regularization matrix  $\mathbf{D}$  has linearly dependent rows and deficient rank. Because the assumptions of Proposition 4 are violated, the multipliers  $\lambda_E$  of the active constraints in Eqs. (7) and (9) are not uniquely determined. One can intuitively understand the difficulty by considering a square with four pixels. Whenever any three constraints are active, the fourth is automatically active as well. This constraint redundancy can be remedied by reparameterizing the model in terms of neighboring pixel differences  $\mathbf{x} = \mathbf{D}\mathbf{u}$ . Unfortunately, the rank deficiency of  $\mathbf{D}$  is also an issue. Adding the same constant to all of the components of  $\mathbf{u}$  yields exactly the same  $\mathbf{x}$ . To circumvent this problem, we simply append a bottom row to  $\mathbf{D}$  with all entries 0 except for a 1 in the last position. If  $\mathbf{V}$  is the amended version of  $\mathbf{D}$ , then  $\mathbf{V}$  has full column rank, and the vector  $\mathbf{x} = \mathbf{V}\mathbf{u}$  uniquely determines the image. Indeed, one can solve for  $\mathbf{x}$  in the form  $\mathbf{u} = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{x}$ . The bottom entry of  $\mathbf{x}$  is obviously the gray level of the last pixel of the image.

Despite the presence of the inverse of the huge  $mn \times mn$  matrix  $\mathbf{V}^T\mathbf{V}$ , the transformation  $\mathbf{u} = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{x}$  is not as daunting as it appears. First of all, multiplication by the sparse matrix  $\mathbf{V}^T$  is trivial. More importantly, the matrix  $\mathbf{V}^T\mathbf{V}$  is symmetric, banded, and extremely sparse. To count its nonzero entries, note that except for diagonal entries, these entries occur in the same positions as the nonzero entries of the adjacency matrix of a corresponding graph with  $2mn - m - n$  edges and  $mn$  nodes. Because an adjacency matrix has twice as many nonzero entries as edges, the matrix  $\mathbf{V}^T\mathbf{V}$  has at most  $2(2mn - m - n) + mn = 5mn - 2m - 2n$  nonzero entries. These occur within a band of width  $\min\{m, n\}$  along the main diagonal, depending on whether we stack columns or concatenate rows. The most convenient way to solve equations of the kind  $\mathbf{V}^T\mathbf{V}\mathbf{a} = \mathbf{b}$  is to extract the Cholesky decomposition  $\mathbf{L}$  of  $\mathbf{V}^T\mathbf{V}$  and execute forward and backward substitution. Although extraction of  $\mathbf{L}$  is cheap for banded matrices, it is even cheaper for banded matrices with just a handful of nonzero entries per row. In our experience, the computational complexity of extracting  $\mathbf{L}$  scales linearly in the product  $mn$ . Since  $\mathbf{L}$  itself is sparse, forward and backward substitution are also very cheap. For instance with a  $256 \times 256$  image, MATLAB computes  $\mathbf{L}$  (a  $65536 \times 65536$  matrix) in 0.26 seconds on a laptop;  $\mathbf{L}$  contains just 1,971,395 nonzero entries. The sparsity of  $\mathbf{L}$  suggests that it be computed once and stored in compressed format for all images of a given size. Many of its nonzero entries are close to zero. Thus, a fairly light truncation of the non-diagonal entries of  $\mathbf{L}$  gives an even sparser matrix realizing nearly the same transformation. Figure 6 displays the sparsity pattern of the matrix  $\mathbf{V}^T\mathbf{V}$  and its permuted Cholesky factor  $\mathbf{L}$  for  $64 \times 64$  images. Images of other sizes show similar sparsity patterns.

The problem of minimizing the objective function  $\frac{1}{2}\|\mathbf{w} - \mathbf{K}\mathbf{u}\|_2^2 + \rho\|\mathbf{D}\mathbf{u}\|_1$  in the transformed variable  $\mathbf{x}$  turns out to coincide with lasso penalized regression, for which an efficient path algorithm is known [15, 16]. Let us sketch how path following works in the more general case. The objective function is  $f(\mathbf{B}\mathbf{x}) + \rho\|\mathbf{x}\|_1$ , where

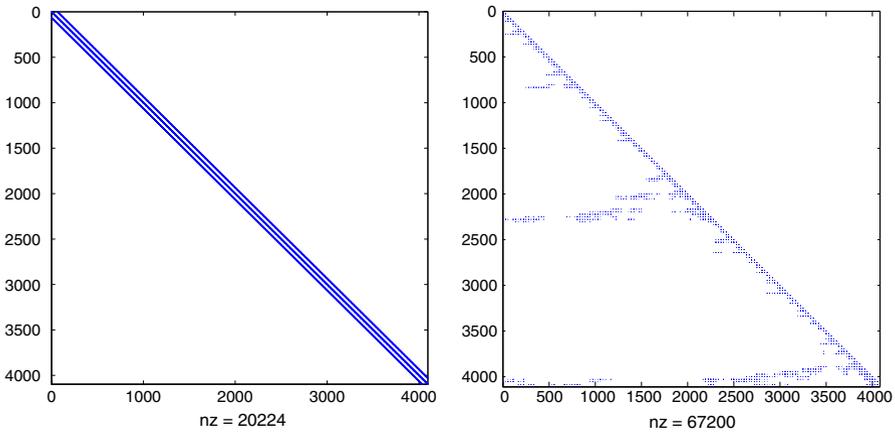


Fig. 6 Sparsity patterns of  $V^T V$  and its Cholesky decomposition  $L$  for 64-by-64 images

$B = (V^T V)^{-1} V^T$  and  $x_-$  denotes the vector  $x$  with its last entry deleted. The penalty contributions correspond to affine equality constraints in constrained minimization. In path following, the penalty constant  $\rho$  starts large and moves downward. The initial image is flat with gray level determined by taking  $x_- = \mathbf{0}$  and adjusting the last entry of  $x$  to minimize  $f(Bx)$ . Call this point  $x_\infty$ . The first escape time occurs at  $\rho_{\max} = \max_j |(B^T \nabla f(Bx_\infty))_j|$ . At this juncture path following begins in earnest. Under the  $x$  parameterization, the loss function has gradient  $B^T \nabla f(Bx)$  and second differential  $B^T d^2 f(Bx) B$ . Because  $f(Bx)$  is not strictly convex, our previous reparameterization from  $x$  to  $y$  variables is needed. Based on Eq. (11), the path ODEs reduce to

$$\begin{aligned} \frac{d}{d\rho} x_{\bar{z}} &= -(B^T_{\bar{z}} d^2 f(Bx) B_{\bar{z}})^{-1} \text{sgn}(x_{\bar{z}}), & \frac{d}{d\rho} x_Z &= \mathbf{0}, \\ \frac{d}{d\rho} \lambda_Z &= -B^T_{\bar{z}} d^2 f(Bx) B_{\bar{z}} \frac{d}{d\rho} x_{\bar{z}}. \end{aligned} \tag{24}$$

Observe that the updates of Eq. (11) drastically simplify because the rows of the active constraint matrix  $U_Z$  and the columns of its null space matrix  $Y$  are populated by standard Euclidean unit vectors. Furthermore, for the ROF model of image denoising,  $d^2 f(Bx)$  is a diagonal matrix. Alternatively, one can derive the ODEs (24) from first principles by implicitly differentiating the stationary conditions. Path following solves the coupled ODEs (24) segment by segment.

For a quadratic loss function, the second differential is constant, and the solution path is piecewise linear. Thus no ODE solving is involved. With a blurring matrix  $K$ , the second differential is  $B^T d^2 f(Bx) B = B^T K^T K B$ . After each path extension, the path directions (24) yield the next event time  $\rho_j$  at which a nonzero component  $x_j$  hits zero, or a multiplier  $\lambda_j$  of a zero component  $x_j$  hits  $\rho$  or  $-\rho$ . The path is then extended to the closest of these event times. In deblurring or denoising, the inverse of  $B^T_{\bar{z}} K^T K B_{\bar{z}}$  is best computed via a QR decomposition of  $B_{\bar{z}} K$ . At each kink in the path,  $B_{\bar{z}} K$  changes by adding or deleting a column of  $B K$ . As we mentioned



**Fig. 7** A noisy image and snapshots along the regularization path

earlier, it is straightforward to update or downdate the QR decomposition [22]. In the original ROF model, traversing one time segment requires about  $O(p)$  operations for  $p = mn$  total pixels. The whole process ends when  $T$  differences  $x_j$  becomes nonzero. In practice, a large value of  $T$  recovers too grainy an image, so  $T$  is typically much smaller than  $p$ . The total cost of computing the solution path is approximately  $O(Kp)$ , where  $K$  is the number of traversed kinks. Empirically  $K$  is on the order of  $O(T)$  for our examples.

Figure 7 illustrates denoising of a  $112 \times 91$  image of a lighthouse. The corrupted image appears in the top-left corner of the figure. The  $p = 10,192$  pixels generate 20,182 transformed variables. It takes our MATLAB script about one minute of desktop computing time to traverse  $K = 2,500$  segments along the regularization path from  $\rho = 87.9881$  (blank image) to  $\rho = 0.5206$  (a nearly optimal image), where  $T = 2148$  differences become nonzero. In the process, the lighthouse clearly emerges from the fog of oversmoothing. Figure 7 displays selected snapshots along the regularization path. We emphasize that path following based on Eq. (23) reveals the entire path for the interval  $[0.5206, 87.9881]$  of  $\rho$  values. In practice, one can accelerate path following by starting from a  $\rho$  nearer to the ultimate destination.

## 5 Discussion

Our path following algorithm for constrained convex optimization builds on but differs from the tradition of path following in homotopy methods [12] and interior point programming [1]. The paths encountered in the exact penalty method introduce the novelty of piecewise differentiability, which can be effectively handled by tracking the Lagrange multipliers. Computational statisticians deserve credit for exploring this difficult terrain [8, 13, 15, 16, 42, 43]. To our knowledge we are the first to make the connection to exact penalty methods.

Our algorithm enjoys the dual advantages of simplicity and generality. Given the rich numerical resources of MATLAB, it is straightforward to solve the required ODEs segment by segment. Regardless of whether path following is faster or slower than existing optimization methods, it supplies the whole solution path. In regularized estimation, this level of detail offers unprecedented insight into how penalties and predictors interact. Our example on image denoising is a case in point.

In quadratic programming with affine equality and inequality constraints, the solution path is piecewise linear [8]. This permits path following to take large steps. Furthermore, each step can be implemented very efficiently by the sweep operator of computational statistics. Despite the loss of these advantages in more complicated examples, the real culprit in path-following deceleration in many applications is an excessive number of constraints to be navigated. Our image denoising example suffers from this defect. On the positive side of the ledger, in nonconvex problems path following may well prove to be more reliable than competing methods in separating global from local minima [44].

Various extensions of path following are in order. First, the current algorithm commences from the unconstrained solution. Our development relies on the strict convexity and coerciveness of the objective function to ensure a unique starting point. In principle, path initiation should work for any problem with a unique unconstrained minimum. Similarly, path continuation should be possible whenever the interior solution is well defined and piecewise smooth. As the image denoising example suggests, reparametrization can play an important role in correcting defects in strict convexity. Another possibility is to amend the surrogate function  $\mathcal{E}_\rho(\mathbf{x})$ . In our semidefinite programming example, we add the term  $e^{-c\rho} \|\mathbf{X}\|_{\mathbb{F}}^2$  to enforce strict convexity and coerciveness. A similar tactic obviously works in other examples.

A second generalization is to expand the list of penalty functions. For instance, Euclidean penalties of the form  $\|\mathbf{M}\mathbf{x} + \mathbf{a}\|_2$  are useful in grouping parameters in statistical problems. It should be straightforward to extend path following to include such penalties. A third generalization is to remove convexity restrictions altogether. As we have noted, the exact penalty method applies equally to nonconvex programming. Path following in this setting is nontrivial since the solution path is no longer necessarily continuous. This poses a real challenge, and it is unclear to us whether one can construct a theory as satisfying as that standing behind modern interior point methods. We invite the optimization community to tackle this broader issue. In the meantime, we are happy to share our MATLAB code with interested researchers.

**Acknowledgments** Research supported in part by National Science Foundation Grant DMS-1310319 and National Institutes of Health Grants GM53275, MH59490, HG006139 and GM105785.

## References

1. Boyd, S., Vandenberghe, L.: *Convex Optimization*. Cambridge University Press, Cambridge (2004)
2. Forsgren, A., Gill, P.E., Wright, M.H.: Interior methods for nonlinear optimization. *SIAM Rev.* **44**, 525–597 (2002)
3. Luenberger, D.G., Ye, Y.: *Linear and Nonlinear Programming*. International Series in Operations Research & Management Science, 3rd edn, p. 116. Springer, New York (2008)
4. Nocedal, J., Wright, S.J.: *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, 2nd edn. Springer, New York (2006)
5. Ruszczyński, A.: *Nonlinear Optim.* Princeton University Press, Princeton (2006)
6. Zangwill, W.I.: Non-linear programming via penalty functions. *Manag. Sci.* **13**(5), 344–358 (1967)
7. Hestenes, M.R.: *Optimization Theory: The Finite Dimensional Case*. Wiley-Interscience (Wiley), New York (1975)
8. Zhou, H., Lange, K.: A path algorithm for constrained estimation. *J. Comput. Gr. Stat.* **22**, 261–283 (2013)
9. Cottle, R.W., Pang, J.-S., Stone, R.E.: *The Linear Complementarity Problem*. Computer Science and Scientific Computing. Academic Press Inc., Boston (1992)
10. Watson, L.T.: Numerical linear algebra aspects of globally convergent homotopy methods. *SIAM Rev.* **28**(4), 529–545 (1986)
11. Watson, L.T.: Theory of globally convergent probability-one homotopies for nonlinear programming. *SIAM J. Optim.*, **11**(3), 761–780, electronic (2000/2001)
12. Zangwill, W.I., Garcia, C.B.: *Pathways to Solutions, Fixed Points, and Equilibria*. Prentice-Hall Series in Computational Mathematics. Prentice-Hall, New Jersey (1981)
13. Zhou, H., Wu, Y.: A generic path algorithm for regularized statistical estimation. *J. Am. Statist. Assoc.* **109**(506), 686–699 (2014)
14. Bertsekas, D.P.: *Convex Analysis and Optimization*. Athena Scientific, Belmont. With Angelia Nedić and Asuman E. Ozdaglar (2003)
15. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. *Ann. Stat.* **32**(2), 407–499 (2004). (with discussion, and a rejoinder by the authors)
16. Osborne, M.R., Presnell, B., Turlach, B.A.: A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.* **20**(3), 389–403 (2000)
17. Tibshirani, R.J., Taylor, J.: The solution path of the generalized lasso. *Ann. Stat.* **39**(3), 1335–1371 (2011)
18. Lange, K.: *Optimization*. Springer Texts in Statistics. Springer, New York (2004)
19. Magnus, J.R., Neudecker, H.: *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley Series in Probability and Statistics. Wiley, Chichester (1999)
20. Lange, K.: *Numerical Analysis for Statisticians*. Statistics and Computing, 2nd edn. Springer, New York (2010)
21. Chi, E., Lange, K.: Splitting methods for convex clustering. *J. Comput. Gr. Stat.* (in press) (2014)
22. Lawson, C.L., Hanson, R.J.: *Solving least squares problems*. Classics in Applied Mathematics, Society for Industrial Mathematics, new ed., (1987)
23. Dykstra, R.L.: An algorithm for restricted least squares regression. *J. Am. Stat. Assoc.* **78**(384), 837–842 (1983)
24. Deutsch, F.: *Best Approximation in Inner Product Spaces*. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, 7. Springer, New York (2001)
25. Berry, M.W., Browne, M., Langville, A.N., Puaça, V.P., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. *Comput. Statist. Data Anal.* **52**(1), 155–173 (2007)
26. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. *Nature* **401**, 788–791 (1999)
27. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization, in NIPS, pp. 556–562, MIT Press (2001)
28. Boyd, S., Kim, S.-J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. *Optim. Eng.* **8**(1), 67–127 (2007)

29. Ecker, J.G.: Geometric programming: methods, computations and applications. *SIAM Rev.* **22**(3), 338–362 (1980)
30. Peressini, A.L., Sullivan, F.E., Uhl Jr, J.J.: *The Mathematics of Nonlinear Programming*. Undergraduate Texts in Mathematics. Springer, New York (1988)
31. Peterson, E.L.: Geometric programming. *SIAM Rev.* **18**(1), 1–51 (1976)
32. Passy, U., Wilde, D.J.: A geometric programming algorithm for solving chemical equilibrium problems. *SIAM J. Appl. Math.* **16**, 363–373 (1968)
33. Boyd, S.P., Kim, S.-J., Patil, D.D., Horowitz, M.A.: Digital circuit optimization via geometric programming. *Oper. Res.* **53**, 899–932 (2005)
34. Mazumdar, M., Jefferson, T.R.: Maximum likelihood estimates for multinomial probabilities via geometric programming. *Biometrika* **70**(1), 257–261 (1983)
35. Feigin, P.D., Passy, U.: The geometric programming dual to the extinction probability problem in simple branching processes. *Ann. Probab.* **9**(3), 498–503 (1981)
36. Lange, K., Zhou, H.: MM algorithms for geometric and signomial programming. *Math. Program. Ser. A* **143**, 339–356 (2014)
37. Vandenberghe, L., Boyd, S.: Semidefinite programming. *SIAM Rev.* **38**(1), 49–95 (1996)
38. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenom.* **60**(1–4), 259–268 (1992)
39. Le, T., Chartrand, R., Asaki, T.J.: A variational approach to reconstructing images corrupted by Poisson noise. *J. Math. Imaging Vis.* **27**, 257–263 (2007)
40. Chambolle, A.: An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.* **20**, 89–97 (2004)
41. Goldstein, T., Osher, S.: The split Bregman method for  $l_1$ -regularized problems. *SIAM J. Img. Sci.* **2**, 323–343 (2009)
42. Zhou, H., Armagan, A., Dunson, D.: Path following and empirical Bayes model selection for sparse regressions. [arXiv:1201.3528](https://arxiv.org/abs/1201.3528) (2012)
43. Xiao, W., Wu, Y., Zhou, H.: ConvexLAR: an extension of least angle regression. *J. Comput. Gr. Stat.* Vol. (in press) (2015)
44. Zhou, H., Lange, K.: On the bumpy road to the dominant mode. *Scand. J. Stat.* **37**(4), 612–631 (2010)