Piecewise Linear Solution Paths for Parametric Piecewise Quadratic Programs with Application to Direct Weight Optimization

Jacob Roll
Division of Automatic Control
E-mail: roll@isy.liu.se

28th August 2007

Report no.: LiTH-ISY-R-2816

Address:
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

WWW: http://www.control.isy.liu.se
Abstract
Recently, pathfollowing algorithms for parametric optimization problems with piecewise linear solution paths have been developed within the field of regularized regression. This paper presents a generalization of these algorithms to a wider class of problems, namely a class of parametric piecewise quadratic programs and related problems. It is shown that the approach can be applied to the nonparametric system identification method Direct Weight Optimization (DWO) and be used to enhance the computational efficiency of this method. The most important design parameter in the DWO method is a parameter ($\lambda$) controlling the bias-variance trade-off, and the use of parametric optimization with piecewise linear solution paths means that the DWO estimates can be efficiently computed for all values of $\lambda$ simultaneously. This allows for designing computationally attractive adaptive bandwidth selection algorithms. One such algorithm for DWO is proposed and demonstrated in two examples.

Keywords: System identification, Non-parametric identification, Parametric optimization, Pathfollowing algorithms
Piecewise Linear Solution Paths for Parametric
Piecewise Quadratic Programs with Application
to Direct Weight Optimization

Jacob Roll
2007-08-28

Abstract

Recently, pathfollowing algorithms for parametric optimization problems with piecewise linear solution paths have been developed within the field of regularized regression. This paper presents a generalization of these algorithms to a wider class of problems, namely a class of parametric piecewise quadratic programs and related problems. It is shown that the approach can be applied to the nonparametric system identification method Direct Weight Optimization (DWO) and be used to enhance the computational efficiency of this method. The most important design parameter in the DWO method is a parameter ($\lambda$) controlling the bias-variance trade-off, and the use of parametric optimization with piecewise linear solution paths means that the DWO estimates can be efficiently computed for all values of $\lambda$ simultaneously. This allows for designing computationally attractive adaptive bandwidth selection algorithms. One such algorithm for DWO is proposed and demonstrated in two examples.

1 Introduction

In many applications, one encounters optimization problems involving a trade-off between two terms to optimize, i.e., problems of the type

$$\min_{x \in P} L(x) + \lambda J(x)$$

where $\lambda$ is a design parameter controlling the trade-off, and $P$ is the feasible region. The problem (1) is a parametric optimization problem (Guddat et al., 1990), or can also be viewed as a special case of multiobjective optimization (Boyd and Vandenberghe, 2004). Examples include bias-variance trade-off issues, regularization (where the size of the penalty is to be estimated) etc.

In recent years, results in explicit model predictive control has led to a growing interest in multiparametric linear and quadratic programming, where it has been shown that the solutions to different classes of problems are piecewise affine functions of the parameters (see, e.g., Bemporad et al., 2002; Borrelli, 2003; Tøndel et al., 2003).

On the other hand, examples of (single-)parametric optimization problems in the form (1), with solutions that are piecewise affine functions of $\lambda$, can be found in the field of regularized regression. In the paper by Efron et al. (2004),
the authors present a new estimation method, least angle regression (LARS), and show that both the solutions to both LARS and LASSO (Tibshirani, 1996) can be efficiently computed for all values of λ simultaneously. As pointed out in (Rosset and Zhu, 2004, 2007), the key to these algorithms is that the solution paths (i.e., the optimal solutions x to the parametric optimization problem as a function of λ) are piecewise linear as λ varies from 0 to ∞. Similar results have recently also been shown for the related m-garrote method and grouped versions of all these methods (Yuan and Lin, 2006). In all these cases, having a single-parametric optimization problem allows for developing path-following algorithms that exploit the piecewise linearity to efficiently find and represent the solution path.

This paper presents a generalization of the framework of path-following algorithms for piecewise linear solution paths in (Efron et al., 2004; Rosset and Zhu, 2007; Yuan and Lin, 2006), and extends the problem class to a broad class of optimization problems. For the case of quadratic plus piecewise affine cost functions, an algorithm with explicit expressions for computation of the solution path is given.

A particular example of parametric problems in the form (1) occurs in Direct Weight Optimization (Roll, 2003; Roll et al., 2005a,b), which is a nonparametric identification/function estimation method. DWO computes pointwise function estimates, given data \( \{y(t), \varphi(t)\}_{t=1}^{N} \) from

\[
y(t) = f_0(\varphi(t)) + e(t)
\]

where \( f_0 \) is the unknown function to be estimated, \( f_0 : \mathbb{R}^n \to \mathbb{R} \), and \( e(t) \) are white noise terms. The idea of making pointwise function estimates has also appeared under names such as Model on Demand, lazy learning and least commitment learning (see, e.g., Atkeson et al., 1997a,b; Bontempi et al., 1999; Stenman, 1999, and references therein).

In order to estimate \( f_0(\varphi^*) \) for a given point \( \varphi^* \), the idea of DWO is to use an affine estimator \( \hat{f}_N(\varphi^*) \)

\[
\hat{f}_N(\varphi^*) = w_0 + \sum_{t=1}^{N} w_t y(t)
\]

and to select the weights \( (w_0, w) \), \( w = (w_1, \ldots, w_N) \) of the estimator by convex optimization. Assuming that \( f_0 \) belongs to some function class \( \mathcal{F} \), the weights can be determined by minimizing a convex upper bound on the maximum mean-squared error (MSE): \(^1\)

\[
mMSE(\varphi^*, w_0, w) = \sup_{f_0 \in \mathcal{F}} \mathbb{E} \left[ \left( f_0(\varphi^*) - \hat{f}_N(\varphi^*) \right)^2 \right] \{\varphi(t)\}_{t=1}^{N}
\]

The resulting minimization problem is convex and can be written in the following abstract form:

\[
\min_{w \in D} \lambda U_2(w) + V(w)
\]

\(^1\)Note that (4) is always convex in \((w_0, w)\), regardless of how the function class \( \mathcal{F} \) is chosen, so in principle we could minimize the maximum MSE directly. However, for many function classes, (4) is difficult to compute, and we have to find an upper bound instead.
where $w$ is a weight vector needed for computing the function estimate, $U_2$ is basically an upper bound on the squared bias, and $V$ the variance term. The design parameter $\lambda$ determines the trade-off between the flexibility of the function class and the noise variance (see Section 3 for more details).

The computed estimate will of course depend on the choice of $\lambda$ controlling the bias-variance trade-off. A method for selecting $\lambda$ for the case when the noise variance is known was given in (Juditsky et al., 2004). It could also be chosen by using cross-validation or some other criterion (Härdle, 1990; Stenman, 1999). For all these methods, one needs to compute the DWO estimates for several different parameter values, which makes it desirable to be able to efficiently compute the entire solution path.

We will show that the developed pathfollowing algorithm can be applied to the DWO approach. This means that we can simultaneously compute the DWO values from (5) for all choices of $\lambda$, which would mean a great gain in computational efficiency. A cross-validation-based algorithm for selection of $\lambda$ in DWO is also proposed.

The paper is organized as follows: Section 2 considers some specific problem classes for which the solution paths are piecewise linear, while Section 3 proposes how this property can be exploited in the DWO approach.

### 2 Piecewise linear solution paths

In this section, we will consider some specific classes of optimization problems of the type (1), which will be shown to have piecewise linear solution paths.

#### 2.1 Piecewise quadratic plus piecewise affine cost function

First, we will consider a class of optimization problems in the form (1) where $J(x)$ is piecewise linear and $L(x)$ is a piecewise quadratic function. A general piecewise linear convex function can be written (Boyd and Vandenberghe, 2004)

$$J(x) = \max_k \{c_k^T x + d_k\}$$

(6)

$L(x)$ is supposed to be strictly convex and in the following form

$$L(x) = \frac{1}{2} x^T Q_i x + f_i^T x + r_i \quad \text{if } x \in \mathcal{X}_i$$

(7)

where the polyhedral regions $\mathcal{X}_i = \{x \mid \tilde{H}_i x \preceq \tilde{q}_i\}, i \in \mathcal{I}$ (here $\preceq$ denotes componentwise inequalities), form a partition of the $x$ space (for simplicity, we let the regions be closed sets, which means that they will intersect at the boundaries). Furthermore, we assume that for each $\lambda \geq 0$, problem (1) has a unique, finite optimal solution.

We can now show the following lemma.

**Lemma 1.** The problem

$$\min_x \lambda \max_k \{c_k^T x + d_k\} + L(x)$$

(8)

subject to $Ax = b$

$\tilde{A}x \preceq \tilde{b}$
with \( L(x) \) given by (7) has a piecewise linear solution path, i.e., the optimal \( x \in \mathbb{R}^n \) is a piecewise affine function of \( \lambda \in [0, \infty) \).

**Proof.** It is easy to see that the optimum of (8), which is unique and finite for given \( \lambda \) according to the assumtions, changes continuously with \( \lambda \).

Now, we can partition the feasible set into a number of relatively open polyhedra together with a number of points (the corners of the polyhedra), denoted \( P_j \) (i.e., either \( P_j = \text{relint}(P_j) \) or \( P_j \) is a single point; for the definition of relative interior, see Boyd and Vandenberghe (2004)), such that on \( P_j \), the cost function of (1) equals

\[
\lambda (c^T_k x + d_{kj}) + \frac{1}{2} x^T Q_{ij} x + f^T_{ij} x + r_{ij}
\]

Let the affine hull of \( P_j \) (Boyd and Vandenberghe, 2004) be described by

\[
\text{aff}(P_j) = \{ x \mid \tilde{A}_j x = \tilde{b}_j \}
\]

where \( \tilde{A}_j \) has full row rank.

Assume that the solution to (8) for a given \( \lambda \) lies in \( P_j \). Then, since this solution is either in the relative interior of \( P_j \) or the only point of \( P_j \), it is also the solution to

\[
\begin{align*}
\min_x & \quad \lambda (c^T_k x + d_{kj}) + \frac{1}{2} x^T Q_{ij} x + f^T_{ij} x + r_{ij} \\
\text{subj. to} & \quad \tilde{A}_j x = \tilde{b}_j
\end{align*}
\]  

(9)

But the solution to this problem can be computed as

\[
x = Q^{-1}_{ij} \left( (\tilde{A}^T_j (\tilde{A}_j Q^{-1}_{ij} \tilde{A}^T_j)^{-1} \tilde{A}_j Q^{-1}_{ij} - I) (f_{ij} + c_k \lambda) + \tilde{A}^T_j (\tilde{A}_j Q^{-1}_{ij} \tilde{A}^T_j)^{-1} \tilde{b}_j \right)
\]

(see Appendix A). Here, \( x \) is linear in \( \lambda \). This means that the solution to (8) must consist of a number of such linear pieces, one piece for every \( P_j \) that the solution path passes through, and hence, the solution path is piecewise linear.

\( \square \)

**Remark 1.** The strict convexity condition for \( L(x) \) can be relaxed. It is sufficient that \( L(x) \) is strictly convex in a neighborhood of each point on the solution path, and convex elsewhere.

Having shown Lemma 1, let us try to outline an algorithm that computes the entire solution path. For simplicity, we assume that \( A \) has full row rank. We can rewrite the problem by introducing slack variables according to

\[
\begin{align*}
\min_{x,s} & \quad \lambda s + L(x) \\
\text{subj. to} & \quad s \geq c^T_k x + d_k \\
& \quad Ax = b \\
& \quad \tilde{A}x \preceq \tilde{b}
\end{align*}
\]

(11)

The Lagrangian function of (11) becomes

\[
\begin{align*}
\mathcal{L}(s, x; \mu, \mu^A, \mu^A) &= \lambda s + L(x) \\
& \quad - \sum_{k=1}^m \mu_k (s - c^T_k x - d_k) - \mu^A^T (\tilde{b} - \tilde{A}x) - \mu^A^T (b - Ax)
\end{align*}
\]

(12)
Using a version of the Karush-Kuhn-Tucker (KKT) conditions (Rockafellar, 1970, Cor. 28.3.1) we can see that \((x, s)\) is the optimal solution to (11) if and only if the following conditions are satisfied for some subgradient \(\nu\) of \(L(x)\):

\[
\begin{align*}
\nu + \sum_{k=1}^{m} \mu_k c_k + \bar{A}^T \mu^A &+ A^T \mu^A = 0 \quad (13a) \\
\lambda - \sum_{k=1}^{m} \mu_k & = 0 \quad (13b) \\
s &\geq c_k^T x + d_k \quad (13c) \\
\bar{A} x & \preceq b \\nA x & = b \quad (13d) \\
\mu_k (s - c_k^T x - d_k) & = 0 \quad (13f) \\
\mu_j^A (\bar{b}_j - \bar{A}^j x) & = 0 \quad (13g) \\
\mu_k &\geq 0, \quad \mu_j^A \preceq 0 \quad (13h)
\end{align*}
\]

The subgradient \(\nu\) can be written as a convex combination of linear expressions:

\[
\nu = \sum_{i \in T^a} \alpha_i (Q_i x + f_i) \quad (14a)
\]

where

\[
T^a = \{ i \mid x \in X_i \}, \quad \sum_{i \in T^a} \alpha_i = 1, \quad \alpha_i \geq 0 \quad (14b)
\]

The KKT conditions have a solution that is unique in \((x, s)\), but not necessarily in \((\mu, \mu^A, \mu^L, \mu^A)\). Hence, we should keep the number of active constraints at a minimum, to ensure that we get a unique solution.

Denote the different sets of active constraints by \(K^a\) (for \(\mu_k\)) and \(J^a\) (for \(\mu_j^A\)). If we assume that the solution for the current \(\lambda\) is at a linear piece (not at a knot) in the interior of a \(X_i\) region, then (13a) and (13b) can be written as

\[
\begin{align*}
Q_i x + \sum_{k \in K^a} \mu_k c_k + \sum_{j \in J^a} \mu_j^A \bar{A}^j + A^T \mu^A & = -f_i \quad (15a) \\
\sum_{k \in K^a} \mu_k & = \lambda \quad (15b)
\end{align*}
\]

Let us introduce the following notation: Let

\[
\begin{align*}
K^a & = \{ k_1, k_2, \ldots, k_n \} \\
J^a & = \{ j_1, j_2, \ldots, j_m \} \\
\mu_k^a & = (\mu_{k_1}, \ldots, \mu_{k_n})^T \\
\hat{\mu}_j^a & = (\mu_{j_1}^A, \ldots, \mu_{j_m}^A)^T \\
C^a & = (c_{k_1}, \ldots, c_{k_n})^T \\
\bar{A}^j & = \begin{pmatrix} \bar{A}_{j_1} \\ \vdots \\ \bar{A}_{j_m} \end{pmatrix}
\end{align*}
\]
\[ d^{K_a} = (d_{k_1}, \ldots, d_{k_n})^T \]
\[ b^{J_a} = (b_{j_1}, \ldots, b_{j_n})^T \]

If we combine (15), (13f), (13g) and (13e), to obtain the solution we then need to solve

\[
\begin{bmatrix}
Q_i & 0 & C_{K_a}^T & \tilde{A}_{J_a}^T & A^T \\
0 & 0 & \mathbf{1}_{n^c} & 0 & 0 \\
C_{K_a} & \mathbf{1}_{n^c} & 0 & 0 & 0 \\
A_{J_a} & 0 & 0 & 0 & 0 \\
A & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\lambda \\
\mu_{K_a} \\
\mu_{J_a} \\
\mu_{A}
\end{bmatrix}
= 
\begin{bmatrix}
-f_i \\
-s \\
-d_{K_a} \\
b_{J_a} \\
b
\end{bmatrix}
\tag{16}
\]

It now follows from Lemma 2 of Appendix B that if

\[
\begin{bmatrix}
C_{K_a} & \mathbf{1}_{n^c} \\
A_{J_a} & 0 \\
A & 0
\end{bmatrix}
\tag{17}
\]

has full row rank, then the solution to (16) is unique.

To compute what happens for a small change in \( \lambda \), we can solve

\[
\begin{bmatrix}
Q_i & 0 & C_{K_a}^T & \tilde{A}_{J_a}^T & A^T \\
0 & 0 & \mathbf{1}_{n^c} & 0 & 0 \\
C_{K_a} & \mathbf{1}_{n^c} & 0 & 0 & 0 \\
A_{J_a} & 0 & 0 & 0 & 0 \\
A & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x^T \\
\frac{\partial x}{\partial \lambda} \\
\frac{\partial s}{\partial \lambda} \\
\frac{\partial \mu_{K_a}}{\partial \lambda} \\
\frac{\partial \mu_{J_a}}{\partial \lambda} \\
\frac{\partial \mu_{A}}{\partial \lambda}
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
1 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\tag{18}
\]

If \( x \) belongs to an intersection between a number of regions \( \bigcap_{i \in T^c} X_i \), we need to solve

\[
\begin{bmatrix}
\sum_{i \in T^c} \alpha_i Q_i & 0 & C_{K_a}^T & \tilde{A}_{J_a}^T & A^T \\
0 & 0 & \mathbf{1}_{n^c} & 0 & 0 \\
C_{K_a} & \mathbf{1}_{n^c} & 0 & 0 & 0 \\
A_{J_a} & 0 & 0 & 0 & 0 \\
A & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\lambda \\
\mu_{K_a} \\
\mu_{J_a} \\
\mu_{A}
\end{bmatrix}
= 
\begin{bmatrix}
-s \\
-d_{K_a} \\
b_{J_a} \\
b
\end{bmatrix}
\tag{19}
\]

instead of (16). Since \( \alpha_i \) are unknown, this is not a system of linear equations, but as we will see, it can still be handled using mostly linear techniques. Let

\[
H_{T^c} x = q_{T^c}
\tag{20}
\]

be a minimal number of constraints that restrict \( x \) to \( \text{aff} \left( \bigcap_{i \in T^c} X_i \right) \) (taking into account also the last three block rows of (19)). What we need to find is a solution to the combined problem (19) and (20).

Extend \( H_{T^c} \) to a square, non-singular matrix according to

\[
\begin{bmatrix}
H_{T^c} \\
H_{\perp T^c}
\end{bmatrix}
\in \mathbb{R}^{n \times n}, \quad H_{T^c} H_{\perp T^c}^T = 0
\]

Given a particular solution \( x^* \) to (20), the general solution can be written as

\[
x = x^* + H_{\perp T^c}^T \beta
\]

6
Inserting this into the first block row of (19) and multiplying from left by \( H_{Ia}^\perp \) gives
\[
\sum_{i \in I^a} \alpha_i H_{Ia}^\perp \left( Q_i(x^* + H_{Ia}^\perp T \beta) + f_i \right)
+ H_{Ia}^\perp C_{K^a}^T \mu_{K^a} + H_{Ia}^\perp \bar{A}_{fa}^T \mu_{Ja}^A + H_{Ia}^\perp A^T \mu^A = 0
\]
Now, since \( L(x) \) is continuous, the gradients of all the quadratic functions with indices in \( I^a \) have the same component along the common boundary of the regions \( X_i, i \in I^a \). Hence, the first sum is independent of \( \alpha \), and we can choose any index \( l \in I^a \) and replace the sum according to
\[
\sum_{i \in I^a} \alpha_i H_{Ia}^\perp \left( Q_i(x^* + H_{Ia}^\perp T \beta) + f_i \right) = H_{Ia}^\perp \left( Q_l(x^* + H_{Ia}^\perp T \beta) + f_l \right)
\]
Hence, we can solve
\[
\begin{pmatrix}
H_{Ia}^\perp Q_l H_{Ia}^\perp T & 0 & H_{Ia}^\perp C_{K^a}^T & H_{Ia}^\perp \bar{A}_{fa}^T & H_{Ia}^\perp A^T \\
0 & 0 & 1_{n^c}^T & 0 & 0 \\
C_{K^a} H_{Ia}^\perp T & 0 & 0 & 0 & 0 \\
\bar{A}_{fa} H_{Ia}^\perp T & 0 & 0 & 0 & 0 \\
AH_{Ia}^\perp T & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
\beta \\
-s \\
\mu_{K^a} \\
\mu_{Ja}^A \\
\mu^A
\end{pmatrix}
= \begin{pmatrix}
-H_{Ia}^\perp(Q_lx^* + f_l)
\end{pmatrix}
\]
which, due to the minimality of (20), is still uniquely solvable. The solution can then be inserted into the first block row of (19) to solve for \( \alpha \).

As before, to compute what happens for a small change in \( \lambda \), we can solve
\[
\begin{pmatrix}
H_{Ia}^\perp Q_l H_{Ia}^\perp T & 0 & H_{Ia}^\perp C_{K^a}^T & H_{Ia}^\perp \bar{A}_{fa}^T & H_{Ia}^\perp A^T \\
0 & 0 & 1_{n^c}^T & 0 & 0 \\
C_{K^a} H_{Ia}^\perp T & 0 & 0 & 0 & 0 \\
\bar{A}_{fa} H_{Ia}^\perp T & 0 & 0 & 0 & 0 \\
AH_{Ia}^\perp T & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
\frac{\partial \beta}{\partial \lambda} \\
\frac{\partial s}{\partial \lambda} \\
\frac{\partial \mu_{K^a}}{\partial \lambda} \\
\frac{\partial \mu_{Ja}^A}{\partial \lambda} \\
\frac{\partial \mu^A}{\partial \lambda}
\end{pmatrix}
= \begin{pmatrix}
0 \\
1 \\
0 \\
0 \\
0
\end{pmatrix}
\]
To start the algorithm, we first have to compute the optimal solution for \( \lambda = 0 \) (let us denote this solution by \( x_0 \)). This is an ordinary convex optimization problem. After this, we select a maximal subset of the constraints that are active at \( x_0 \), such that (17) has full row rank. (However, note that in \( K^a \), it is sufficient to include only one index \( k \) to start with.)

The algorithm can now be described as follows:

**Algorithm 2.1.** Given: A parametric optimization problem of the type (8).

1. Set \( \lambda = 0 \).
2. Compute the solution \( x_0 \) to (8).

7
3. Let \( S = \{ (\lambda, x_{\lambda}) \} = \{ (0, x_0) \} \).

4. Let \( K^a = \{ k \} \) for some \( k \in \arg \max_k \{ c_k^T x_{\lambda} + d_k \} \), and let \( J^a, I^a \) be maximal subsets of indices for which corresponding constraints are active at \( x_0 \), such that (17) has full row rank. Compute \( H_{T^a} \) and \( H_{\bar{f}_x} \). (If \( I^a \) has only one member, \( H_{T^a} \) will be empty, and we can let \( H_{\bar{f}_x} = I \).)

5. Compute (22) to get \( s, \mu_{K^a}, \mu_{J^a}, \) and \( \mu^A \).

6. Compute the directions given by (23).

7. Find the minimal \( \delta \lambda \geq 0 \) such that one of the following conditions are satisfied:

   (a) \( s + \frac{\partial s}{\partial \lambda} \delta \lambda = c_k^T (x_{\lambda} + \frac{\partial x}{\partial \lambda} \delta \lambda) + d_k \) and \( \frac{\partial s}{\partial \lambda} < c_k^T \frac{\partial x}{\partial \lambda} \) for some \( k \notin K^a \). Then move the corresponding \( k \) to \( K^a \).

   (b) \( \mu_k + \frac{\partial \mu_k}{\partial \lambda} \delta \lambda = 0 \) and \( \frac{\partial \mu_k}{\partial \lambda} < 0 \) for some \( k \in K^a \). Then remove the corresponding \( k \) from \( K^a \).

   (c) \( A_j (x_{\lambda} + \frac{\partial x}{\partial \lambda} \delta \lambda) = b_j \) and \( A_j \frac{\partial x}{\partial \lambda} > 0 \) for some \( j \notin J^a \). Then move the corresponding \( j \) to \( J^a \).

   (d) \( \mu^A_j + \frac{\partial \mu^A_j}{\partial \lambda} \delta \lambda = 0 \) and \( \frac{\partial \mu^A_j}{\partial \lambda} < 0 \) for some \( j \in J^a \). Then remove the corresponding \( j \) from \( J^a \).

   (e) \( x_{\lambda} + \frac{\partial x}{\partial \lambda} \delta \lambda \in X_l \) for some \( l \notin I^a \), and \( x_{\lambda} + \frac{\partial x}{\partial \lambda} (\delta \lambda + \epsilon) \notin X_l \) for some \( \epsilon > 0 \) and \( i \in I^a \). Then include \( l \) in \( I^a \).

   (f) There would not be a subgradient \( \nu \) satisfying (14) (with \( x \) replaced by \( x_{\lambda} + \frac{\partial x}{\partial \lambda} \delta \lambda \)) if increasing \( \delta \lambda \) further. Then remove one \( i \) with corresponding \( \alpha_i = 0 \) from \( I^a \).

Add \( \delta \lambda \) to \( \lambda \), update \( x_{\lambda}, s, \mu_{K^a}, \mu_{J^a}, \) and \( \mu^A \), and recompute \( H_{T^a} \) and \( H_{\bar{f}_x} \). If there is no \( \delta \lambda \geq 0 \) for which the conditions are satisfied, set \( \lambda = \infty \).

8. Add the new pair \((\lambda, x_{\lambda})\) to \( S \); \( S := \{ S, (\lambda, x_{\lambda}) \} \).

9. If \( \lambda = \infty \), stop. Otherwise, go to 6.

When the algorithm has finished, \( S \) will contain the knots of the piecewise linear solution path, and any solution to (8) can be obtained by linear interpolation between two neighboring knots.

Note that only linear techniques are needed to find the solution path, except for step 7f, where \( \alpha_i \) is a nonlinear function of \( \lambda \). This step can be handled by simple line search. It can also be shown that all the changes of step 7 will lead to full row rank of (17).

Remark 2. Without restriction, we can assume non-degeneracy in the sense that each term \( c_k^T x + d_k \) in (6) is the maximal term on a set of positive measure, and that two terms are equal only on a set of measure zero. This may simplify implementations for computing the solution path. However, this is not necessary for Lemma 1 to hold.
2.2 Quadratic plus piecewise affine cost function

Now let us consider the class of optimization problems with positive definite quadratic \( L(x) \) and a \( J(x) \) which is a sum of absolute values of affine functions. We will also allow linear equality constraints. In other words, the problems we will consider can be written as

\[
\min_z \lambda \sum_{k=1}^{m} |h_k^T z + g_k| + \frac{1}{2} z^T Q z + p^T z \tag{24}
\]

subject to \( Az = b \)

with \( Q > 0 \). Provided that the problem is feasible, we can use the constraints to eliminate variables, to get an equivalent problem in the form

\[
\min_{\tilde{z}} \lambda \sum_{k=1}^{m} |\tilde{h}_k^T \tilde{z} + \tilde{g}_k| + \frac{1}{2} \tilde{z}^T \tilde{Q} \tilde{z} + \tilde{p}^T \tilde{z} \tag{25}
\]

We can now make the variable substitution \( x = \tilde{Q}^{1/2} \tilde{z} + \tilde{Q}^{-1/2} \tilde{p} \). Hence, it turns out that it is sufficient to consider problems of the type

\[
\min_x \lambda \sum_{k=1}^{m} |c_k^T x + d_k| + \frac{1}{2} x^T x \tag{26}
\]

in more detail. This is a special case of (8), and hence (26) has a piecewise linear solution path.

We will now derive explicit expressions for the directions of the linear parts of the solution path. For simplicity, we will assume that the problem is non-degenerate in the sense that for all \( x \), the vectors of the set \( C_0^x = \{c_k^T x + d_k = 0\} \) are linearly independent.

Introducing slack variables, we can rewrite (26) as

\[
\min_{x,s} \lambda \sum_{k=1}^{m} s_k + \frac{1}{2} x^T x \tag{27}
\]

subject to \( s_k \geq c_k^T x + d_k \)

\( s_k \geq -c_k^T x - d_k \)

The Lagrangian function of (27) becomes

\[
\mathcal{L}(s,x;\mu) = \lambda \sum_{k=1}^{m} s_k + \frac{1}{2} x^T x - \sum_{k=1}^{m} \mu_k^+ (s_k - c_k^T x - d_k) - \sum_{k=1}^{m} \mu_k^- (s_k + c_k^T x + d_k) \]

We can now write the KKT conditions as

\[
x + \sum_{k=1}^{m} c_k \mu_k^+ - \sum_{k=1}^{m} c_k \mu_k^- = 0 \tag{28a}
\]

\[
\lambda - \mu_k^+ - \mu_k^- = 0 \tag{28b}
\]

\[
\mu_k^+ (s_k - c_k^T x - d_k) = 0 \tag{28c}
\]

\[
\mu_k^- (s_k + c_k^T x + d_k) = 0 \tag{28d}
\]

\[
\mu_k^\pm \geq 0 \tag{28e}
\]
Define the sets
\[ K^+ = \{ k : c_k^T x + d_k > 0 \} \]
\[ K^- = \{ k : c_k^T x + d_k < 0 \} \]
\[ K^0 = \{ k : c_k^T x + d_k = 0 \} \] (29)

For \( k \in K^+ \), we get \( \mu_k^+ = \lambda \) and \( \mu_k^- = 0 \) from (28b) and (28d). Similarly, for \( k \in K^- \) we obtain \( \mu_k^+ = 0 \) and \( \mu_k^- = \lambda \). Using (28a) this implies that
\[ x + \lambda \sum_{k \in K^+} c_k - \lambda \sum_{k \in K^-} c_k + \sum_{k \in K^0} c_k(2\mu_k^+ - \lambda) = 0 \] (30)

Since we would like to consider the linear parts of the solution path, we can assume \( 0 < \mu_k^+ < \lambda \) in the last sum. Let us now add \( \delta \lambda \) to \( \lambda \). The solution \((x, s)\) to (28) will then change accordingly by \((\delta x, \delta s)\), so that if \( \delta \lambda \) is small enough we have
\[ x + \delta x + (\lambda + \delta \lambda) \left( \sum_{k \in K^+} c_k - \sum_{k \in K^-} c_k \right) + \sum_{k \in K^0} c_k (2(\mu_k^+ + \delta \mu_k^+) - (\lambda + \delta \lambda)) = 0 \]
which together with (30) implies that
\[ \delta x + \delta \lambda \left( \sum_{k \in K^+} c_k - \sum_{k \in K^-} c_k \right) + \sum_{k \in K^0} c_k (2\delta \mu_k^+ - \delta \lambda) = 0 \] (31)

If we introduce the notation \( K^0 = \{ k_1, \ldots, k_n^0 \} \) and
\[ C^0 = \begin{pmatrix} c_{k_1}^T \\ \vdots \\ c_{k_n^0}^T \end{pmatrix}, \quad \delta M^+ = \begin{pmatrix} \delta \mu_{k_1}^+ \\ \vdots \\ \delta \mu_{k_n^0}^+ \end{pmatrix} \]
and similarly for \( C^+ \) and \( C^- \), we can write (31) as
\[ \delta x + \left( C^+ C^0 + C^- C^0 \right) \delta \lambda + C^0 \left( 2\delta M^+ - 1_{n^0} \delta \lambda \right) = 0 \] (32)

At the same time, for \( j \in K^0 \) it must hold that
\[ c_j^T (x + \delta x) + d_j = 0 \quad \Rightarrow \quad c_j^T \delta x = 0 \]
Hence, multiplying (33) by \( C^0 \) yields
\[ C^0 \left( C^+ C^0 + C^- C^0 \right) \delta \lambda + C^0 C^0 \left( 2\delta M^+ - 1_{n^0} \delta \lambda \right) = 0 \] (34)

Since \( C^0 C^0 \) is invertible, we get
\[ \frac{\delta M^+}{\delta \lambda} = \frac{1}{2} \left( C^0 C^0 \right)^{-1} C^0 \left( -C^+ C^0 + C^- C^0 + C^0 C^0 \right) \] (35)
Inserting this into (33) results in

$$\frac{\delta x}{\delta \lambda} = \left( I - C^0T (C^0C^0T)^{-1} C^0 \right) \left( -C^+T 1_n + C^-T 1_n - C^0T 1_n^o \right)$$ (36)

Note that since this expression is locally constant, the solution \( x \) will locally change linearly as \( \lambda \) changes. Just as for the problem class considered in Section 2.1, this means that when computing the solution path, we only need to store the solutions and values of \( \lambda \) for the knots of the solution path, as the values in between can be obtained afterwards by simple linear interpolation.

We can now give an algorithm for finding the solution path to a problem in the form (26). For the algorithm, we will need the rates of change in value of the terms \( c_k^T x + d_k \) as \( \delta x \) is added to \( x \):

$$\frac{\partial (c_k^T x)}{\partial \lambda} = c_k^T \left( I - C^0T (C^0C^0T)^{-1} C^0 \right) \left( -C^+T 1_n + C^-T 1_n - C^0T 1_n^o \right)$$ (37)

**Algorithm 2.2.** Given: An optimization problem of the type (26).

1. Set \( \lambda = 0 \), \( x_\lambda = 0 \), and \( S = \{(\lambda, x_\lambda)\} \).
2. Compute the sets \( K^+, K^- \) and \( K^0 \), as defined in (29).
3. Compute the values of \( \mu^+_k \) for all \( k \) from the KKT conditions.
4. Compute the directions given by (35) and (36).
5. Find the minimal \( \delta \lambda \geq 0 \) such that one of the following conditions are satisfied:
   
   (a) \( c_k^T x + d_k = 0 \), and the right hand side of (37) is negative, for some \( k \in K^+ \). Then move the corresponding \( k \) from \( K^+ \) to \( K^0 \).
   (b) \( c_k^T x + d_k = 0 \), and the right hand side of (37) is positive, for some \( k \in K^- \). Then move the corresponding \( k \) from \( K^- \) to \( K^0 \).
   (c) \( \mu^+_k = \lambda \) and the corresponding element of the right hand side of (35) is positive for some \( k \in K^0 \). Then move the corresponding \( k \) from \( K^0 \) to \( K^+ \).
   (d) \( \mu^-_k = 0 \) and the corresponding element of the right hand side of (35) is negative for some \( k \in K^0 \). Then move the corresponding \( k \) from \( K^0 \) to \( K^- \).

Add \( \delta \lambda \) to \( \lambda \). If there is no \( \delta \lambda \geq 0 \) for which the conditions are satisfied, set \( \lambda = \infty \).
6. Add the new pair \( (\lambda, x_\lambda) \) to \( S \); \( S := \{S, (\lambda, x_\lambda)\} \).
7. If \( \lambda = \infty \), stop. Otherwise, go to 4.
2.3 Related solution paths for different problems

Apart for the classes described in the previous sections, there are several other problem classes that have piecewise linear solution paths. In fact, starting from one problem, we can derive a family of problems having the same solution path. This can be seen from the following observation (cf. Boyd and Vandenberghe (2004, Exercise 4.51)).

**Observation 1.** Suppose that $L : \mathcal{D}(L) \subseteq \mathbb{R}^n \rightarrow \mathcal{R}(L) \subseteq \mathbb{R}$ and $J : \mathcal{D}(L) \rightarrow \mathcal{R}(J) \subseteq \mathbb{R}$ are convex functions defined on the same convex domain $\mathcal{D}(L)$, that

$$\min_{\lambda} L(x) + \lambda J(x)$$

has a well-defined solution path\(^2\) for $\lambda \in [0, \infty]$, and that $f_1 : \mathcal{R}(L) \rightarrow \mathbb{R}$ and $f_2 : \mathcal{R}(J) \rightarrow \mathbb{R}$ are strictly increasing functions. Then the solution path of

$$\min f_1(L(x)) + \mu f_2(J(x))$$

for $\mu \in [0, \infty]$ is a subset of the solution path of (38).

**Proof.** It is trivial to see that $\lambda = \mu = 0$ give the same optima. Now if the observation is false, there is a $\mu_0 > 0$ such that $x_{\mu_0}$ is a solution to (39) with $\mu = \mu_0$, but that there is no $\lambda$ such that $x_{\lambda}$ is a solution to (38).

From (39) we get

$$J(x_{\mu_0}) = \min_{x : L(x) = L(x_{\mu_0})} J(x)$$

(40)

Hence, if $x_{\mu_0}$ does not belong to the solution path of (38), there is no $x_\lambda$ on this solution path with $L(x_\lambda) = L(x_{\mu_0})$. This means that for some value $\lambda_0$, there must be (at least) two solutions, $x_1$ and $x_2$, to (38), satisfying $L(x_1) < L(x_{\mu_0}) < L(x_2)$, for large enough $\lambda$, the solutions $x_\lambda$ must satisfy $L(x_\lambda) \geq L(x_{\mu_0})$.

Furthermore, it must hold that

$$L(x_1) + \lambda_0 J(x_1) = L(x_2) + \lambda_0 J(x_2) < L(x_{\mu_0}) + \lambda_0 J(x_{\mu_0})$$

Due to the convexity of $L$ and $J$, there is a $\nu \in (0,1)$ such that $L(\nu x_1 + (1 - \nu)x_2) = L(x_{\mu_0})$ and

$$L(\nu x_1 + (1 - \nu)x_2) + \lambda_0 J(\nu x_1 + (1 - \nu)x_2) = L(x_2) + \lambda_0 J(x_2)$$

$$< L(x_{\mu_0}) + \lambda_0 J(x_{\mu_0})$$

which implies that $J(\nu x_1 + (1 - \nu)x_2) < J(x_{\mu_0})$. But this contradicts (40), and the observation follows.

**Example 1.** Note that the solution paths do not need to be identical, as the following example shows: Let

$L(x) = x^2$, \quad $J(x) = (1 - x)^2$,

$f_1(y) = y$, \quad $f_2(y) = \begin{cases} y & y \leq \frac{1}{4} \\ y + \frac{1}{12} & y > \frac{1}{4} \end{cases}$

\(^2\)By well-defined solution path we here mean that for all values of $\lambda$, there is at least one minimum to the problem, and that all minimum points are finite.
Then the solution to
\[
\min_x L(x) + \lambda J(x) = \min_x (1 + \lambda)x^2 - 2\lambda x + \lambda
\]
\[
= \min_x (1 + \lambda)(x - \frac{\lambda}{1 + \lambda})^2 + \frac{\lambda}{1 + \lambda}
\]
is given by \(x = \frac{\lambda}{1 + \lambda}\), which runs continuously from 0 towards 1 as \(\lambda \to \infty\). On the other hand
\[
\min_x f_1(L(x)) + \mu f_2(J(x)) = \min_x \begin{cases} (1 + \mu)(x - \frac{\mu}{1 + \mu})^2 + \frac{\mu}{1 + \mu} & x \geq \frac{1}{2} \\ (1 + \mu)(x - \frac{\mu}{1 + \mu})^2 + \frac{\mu}{1 + \mu} + \mu & x < \frac{1}{2} \end{cases}
\]
For \(\mu \geq 1\) we get the same solution as before, since the optimum then satisfies \(x \geq \frac{1}{2}\). However, for \(\mu \in [0,1)\), letting \(x = \frac{1}{2}\) could give a smaller value than minimizing the second expression above. Hence, the minimum is given by
\[
\min \left\{ \frac{1}{4}(1 + \mu), \frac{\mu}{1 + \mu} + \frac{\mu}{12} \right\}
\]
where the first term is obtained by setting \(x = \frac{1}{2}\), and the second one by setting \(x = \frac{\mu}{1 + \mu}\). Simple computations now give the overall solution
\[
x = \begin{cases} \frac{\mu}{1 + \mu} & 0 \leq \mu \leq \frac{1}{4} \text{ or } \mu > 1 \\ \frac{1}{2} & \text{otherwise} \end{cases}
\]
Remark 3. If \(f_1(L(x))\) and \(f_2(J(x))\) are convex, the solution paths are identical. This can be seen by applying Observation 1 to \(f_1(L(x)), f_2(J(x)), f_1^{-1}, \) and \(f_2^{-1}\).

Example 2. If we do not assume convexity of \(L(x)\) and \(J(x)\), Observation 1 does not necessarily hold. For instance, let
\[
L(x) = \begin{cases} -x & x \leq 0 \\ -2x & 0 < x \leq 1 \\ 2x - 4 & 1 < x \end{cases} \quad J(x) = \begin{cases} -2x - 4 & x \leq -1 \\ 2x & -1 < x \leq 0 \\ x & 0 < x \end{cases}
\]

Figure 1: Cost functions for Example 2.
(see Figure 1(a)) and let

\[
f_1(y) = f_2(y) = \begin{cases} y & y \leq 0 \\ 2y & y > 0 \end{cases}
\]

Then the solution path for (38) is

\[
x_\lambda = \begin{cases} 1 & \lambda < 1 \\ \{-1, 1\} & \lambda = 1 \\ -1 & \lambda > 1 \end{cases}
\]

while the solution path for (39) becomes

\[
x_\mu = \begin{cases} 1 & \mu < 0.5 \\ [0, 1] & \mu = 0.5 \\ 0 & 0.5 < \mu < 2 \\ [-1, 0] & \mu = 2 \\ -1 & \mu > 2 \end{cases}
\]

Figure 1(b) shows the cost functions of (38) and (39) when \( \mu = \lambda = 1 \).

Where \( f_1, f_2, J \) and \( L \) are differentiable, the relationship between \( \mu \) and \( \lambda \) can be established as follows (for simplicity, we only consider the case of Remark 3): For an optimal point \( x_\mu \) on the solution path, for some \( \lambda \) it holds that

\[
\nabla L(x_\mu) + \lambda \nabla J(x_\mu) = 0
\]

\[
= f_1'(L(x_\mu)) \nabla L(x_\mu) + \mu f'_2(J(x_\mu)) \nabla J(x_\mu)
\]

If \( \nabla J(x_\mu) \neq 0 \), this yields

\[
\mu = \lambda \frac{f_1'(L(x_\mu))}{f'_2(J(x_\mu))}
\]

(41)

If \( \nabla J(x_\mu) = 0 \), then \( \nabla L(x_\mu) = 0 \), and \( x_\mu \) will be a minimum point for all \( \lambda \) and \( \mu \).

3 Direct weight optimization

The following section will outline how the piecewise linear solution path algorithm can be applied to Direct Weight Optimization (DWO; Roll (2003); Roll et al. (2005a,b)). DWO is a nonparametric method for system identification and function estimation, which for some function classes and under some assumptions has been shown to give optimal pointwise function estimates (in the sense of minimizing the maximum MSE; Nazin et al. (2006)).

Assume that we are given data \( \{y(t), \varphi(t)\}_{t=1}^N \) generated from

\[
y(t) = f_0(\varphi(t)) + e(t)
\]

(42)

where \( f_0 \) is an unknown function, \( f_0 : \mathbb{R}^n \rightarrow \mathbb{R} \), and \( e(t) \) are zero-mean, i.i.d. random variables with variance \( \sigma^2 \), independent of \( \varphi(\tau) \) for all \( \tau \). In this paper, we will assume that \( f_0 \) belongs to a function class \( \mathcal{F} \), whose members can locally be approximately described by a given basis function expansion, with a known upper bound on the approximation error. More precisely, \( \mathcal{F} \) is defined as follows:
Definition 1. Let \( F = F(D, D_\theta, F, M) \) be the set of all functions \( f \), for which there, for each \( \varphi_0 \in D \), exists a \( \theta^0(\varphi_0) \in D_\theta \), such that

\[
\left| f(\varphi) - \theta^{0T}(\varphi_0)F(\varphi) \right| \leq M(\varphi, \varphi_0) \quad \forall \varphi \in D
\]

Here, \( F(\cdot) \) is a vector of given basis functions, \( \theta^{0T}(\varphi_0)F(\varphi) \) is a local (unknown) approximation of \( f(\varphi) \) around \( \varphi_0 \), and \( M(\varphi, \varphi_0) \) is a given upper bound on the approximation error. Figure 2 illustrates the definition for a case when \( \theta^{0T}(\varphi_0)F(\varphi) \) is linear and the bound \( M(\varphi, \varphi_0) \) is quadratic. Examples of function classes that can be formulated in this way include the class of functions with Lipschitz continuous gradients (with a given Lipschitz constant \( L \)). Also systems with both stochastic and unknown-but-bounded noise terms can be handled within the DWO framework. For more details and examples of function classes covered by Definition 1, see e.g., (Roll et al., 2005b).

Figure 2: Illustration of Definition 1: The true function \( f(\varphi) \) (thick line), the local approximation \( \theta^{0T}(\varphi_0)F(\varphi) \) (thin line), and the bounds \( \theta^{0T}(\varphi_0)F(\varphi) \pm M(\varphi, \varphi_0) \) (dashed).

Now, given this information, how would we estimate \( f_0(\varphi^*) \) for a given point \( \varphi^* \)? The idea behind DWO is to estimate \( f_0(\varphi^*) \) by postulating that the estimate should be affine in \( y(t) \), i.e.,

\[
\hat{f}_N(\varphi^*) = w_0 + \sum_{t=1}^{N} w_t y(t)
\]

and determine the weights \( (w_0, w), w = (w_1, \ldots, w_N) \) by minimizing an upper bound on the maximum MSE (4), i.e.,

\[
mMSE(\varphi^*, w_0, w) = \sup_{f_0 \in F} E \left[ \left( f_0(\varphi^*) - \hat{f}_N(\varphi^*) \right)^2 \right| \{ \varphi(t) \}_{t=1}^{N}
\]

\[
= \sup_{f_0 \in F} E \left[ \left( f_0(\varphi^*) - w_0 - \sum_{t=1}^{N} w_t (f_0(\varphi(t)) + e(t)) \right)^2 \right| \{ \varphi(t) \}_{t=1}^{N}
\]

\[\text{This assumption is not very restrictive. For instance, note that any least-squares estimation with fixed basis functions will lead to an estimate that is affine in } y(t).\]
where we have replaced \( \hat{f}_N(\varphi^*) \) in (4) by using (42) and (44).

For the particular function class given by Definition 1, we can give an upper bound on the maximal MSE that leads to the following minimization (see Roll et al., 2005b):

\[
\min_w \left( \sum_{t=1}^{N} |w_t| M(\varphi(t), \varphi^*) + M(\varphi^*, \varphi^*) \right)^2 + \sigma^2 \sum_{t=1}^{N} w_t^2 \\
\text{subj. to } \sum_{t=1}^{N} w_t F(\varphi(t)) - F(\varphi^*) = 0
\]

(46)

This is a typical bias-variance trade-off problem, where the first term of (46) is an upper bound on the squared bias of \( \hat{f}_N(\varphi^*) \), and the second term is the variance. The problem is convex, and the solution depends on the size of \( \sigma^2 \) compared to \( M \). In many cases, both \( \sigma^2 \) and \( M \) are unknown, and have to be estimated along with the function. For simplicity, we may assume that at least the shape of the function \( M \) is given, so that \( M(\varphi_1, \varphi_2) = M_0(\varphi_1, \varphi_2) \) where \( M \) is a known function. Then, (46) may be rewritten as

\[
\min_w \lambda \left( \sum_{t=1}^{N} m_t |w_t| + m_0 \right)^2 + \sum_{t=1}^{N} w_t^2 \\
\text{subj. to } \sum_{t=1}^{N} w_t F(\varphi(t)) - F(\varphi^*) = 0
\]

(47)

where we have introduced the notation \( m_0 = m(\varphi^*, \varphi^*) \), \( m_t = m(\varphi(t), \varphi^*) \), and \( \lambda = L^2/\sigma^2 \).

The design parameter \( \lambda \) can be interpreted as controlling the trade-off between the flexibility of the function class and the noise variance. A small \( \lambda \) will put effort on minimizing the variance term (by making more weights non-zero) at the expense of the bias error, while a large \( \lambda \) will decrease the bias error (by making the estimates more local — the exact properties depend on the shapes of \( F(\varphi) \) and \( m(\varphi_1, \varphi_2) \)). To select a good value for \( \lambda \), problem (47) needs to be solved for various values of \( \lambda \), and the solutions can be evaluated, e.g., by cross-validation (Härdle, 1990; Stenman, 1999). For known \( \sigma^2 \), one method for selecting \( \lambda \) was proposed in (Juditsky et al., 2004).

3.1 Solution path for DWO

Now we are ready to show that the DWO problem (47) has a piecewise linear solution path. Using Observation 1 and Remark 3, we can see that (47) has the same solution path as

\[
\min_w \lambda \sum_{t=1}^{N} m_t |w_t| + \sum_{t=1}^{N} w_t^2 \\
\text{subj. to } \sum_{t=1}^{N} w_t F(\varphi(t)) - F(\varphi^*) = 0
\]

(48)
But this problem is in the same form as (24), and hence has a piecewise linear solution path. Therefore, we can use Algorithm 2.2 to compute it, which leads to a significant gain in computational complexity.

3.2 An algorithm for DWO with unknown $\lambda$

With the algorithm for finding the solution path for a given DWO problem, the next step is to choose which $\lambda = L^2/\sigma^2$ to use for the final function estimate. Here we will take a cross-validation approach (for other alternatives, see (Roll, 2003, Section 7.2), where the problem is discussed to some extent under the term “adaptive bandwidth selection”, and (Juditsky et al., 2004)).

Assuming that a set of experimental data is collected, split it into two sets, called the estimation data set $S_e$ and calibration data set $S_c$. Now, the algorithm contains an offline and an online part:

**Algorithm 3.1. Computation of DWO solution with estimated $\lambda$, using a calibration data set.**

*Offline part:*

1. For each point $\varphi(i)$ in the calibration data set, compute the DWO solution paths $w^{(i)}(\lambda)$ and the corresponding function estimates $\hat{f}(\varphi(i), \lambda)$ using the estimation data set.

*Online part:*

1. Given a regression vector $\varphi^*$, for which the function value $f(\varphi^*)$ is to be estimated, choose the $k$ nearest neighbours of $\varphi^*$ from the calibration data set. Denote the set $S^*_k \subset S_c$.

2. Choose $\hat{\lambda}$ as

$$\hat{\lambda} = \arg \min_{\lambda} \sum_{j: \varphi(j) \in S^*_k} \left( y(j) - \hat{f}(\varphi(j), \lambda) \right)^2$$

(we will call the function on the right hand side the $k$-nn cost function).

3. Compute the DWO solution $\hat{f}(\varphi^*, \hat{\lambda})$ using the estimation data set.

An advantage with this scheme is that the computation of the DWO solution paths for the calibration data can be done offline, once and for all. Hence, in the online part, we only need to find the $k$ nearest neighbors and then minimize a cost function in one variable. In practice, the value of each term of the cost function can be computed for a number of representative $\lambda$ values already during the offline part, which makes step 2 in the online part very fast.

Having found an appropriate $\hat{\lambda}$, all that remains is to find the DWO function estimate $\hat{f}(\varphi^*, \hat{\lambda})$ for that particular $\hat{\lambda}$, which is a problem that can be efficiently solved, either through the pathfollowing algorithm, or with the help of some other convex optimization algorithm (Boyd and Vandenberghe, 2004).

The choice of $k$ is a trade-off between relying on enough calibration data points, and at the same time keeping the $k$-nn cost function local enough. This can be compared to choosing the bandwidth for pilot estimates in, e.g., plug-in methods (Stenman, 1999). Preliminary experiments indicate that the quality of the function estimates is fairly robust to the choice of $k$, but this is something that should be further investigated.
3.3 Examples

Let us illustrate Algorithm 3.1 with two examples.

![Data from the system (49).](image)

**Example 3.** First, consider the system

\[
\begin{align*}
y(t) &= f_1(\varphi(t)) + e(t) \\
f_1(\varphi(t)) &= 0.5 + 0.4\varphi_1(t) - 0.2\varphi_2(t) + \varphi_1^2(t) - \varphi_2^2(t)
\end{align*}
\]

where \(e(t)\) is a white noise term with variance 0.1. An estimation data set of 500 points were generated, where \(\varphi(t)\) was drawn from an \(N(0, I)\) distribution. The estimation data set is shown in Figure 3(a). Also, a similar calibration data set was generated.

Suppose now that we would like to estimate \(f_1(\varphi^*)\) for \(\varphi^* = (0.5, 0.5)^T\). To do this, the algorithm in Section 3.2 with \(k = 30\) was used. Figure 3(b) shows the calibration data set with the 30 selected nearest neighbours highlighted. Running the algorithm gives the \(k\)-nn cost function shown in Figure 4(a), and the function estimates as a function of \(\lambda\) are plotted in Figure 4(b). We get \(\hat{\lambda} = 1.3461\), which leads to one of the best function estimates in Figure 4(b). The resulting weights \(w(\hat{\lambda})\) are shown in Figure 4(c).

We can make some observations from the example. Choosing a value of \(\lambda\) means that we decide what trade-off between bias and variance to make. For the system (49), the overall bias does not increase that much when increasing the number of nonzero weights, since the effect of the positive second derivative in one direction is neutralised by the effect of the negative second derivative in another direction. This gives us a wider range of values of \(\lambda\) with approximately the same \(k\)-nn cost. This is not a major problem, since what we are primarily after is not \(\lambda\) in itself, but a good function value estimate \(\hat{f}(\varphi^*, \hat{\lambda})\).
Figure 4: Result of running the DWO solution path algorithm on the system (49).
To confirm the argument above, let us instead consider the system

\[
y(t) = f_2(\varphi(t)) + e(t)
\]

\[
f_2(\varphi(t)) = 0.5 + 0.4\varphi_1(t) - 0.2\varphi_2(t) + \varphi_1^2(t) + \varphi_2^2(t)
\]

and estimate \( f_2(\varphi^*) \) using the same experimental setup as before, with estimation and calibration data sets containing exactly the same regression vectors. For this system, the Hessian is positive definite everywhere, which means that the bias will increase with a decreasing \( \lambda \) (i.e., an increasing number of nonzero weights).

![k-nn cost function](a)

![Function value estimates](b)

![Weights corresponding to \( \hat{\lambda} \)](c)

Figure 5: Result of running the DWO solution path algorithm on the system (50).

The resulting \( k \)-nn cost function, function estimates and weights are presented in Figure 5. Note that the differences in the \( k \)-nn cost function are much larger than for the previous system, which allows us to more unambiguously determine the value of \( \hat{\lambda} \).
Example 4. Consider the system

\[
y(t) = f(\varphi(t)) + e(t) \\
f(\varphi(t)) = 1 + \sin(6 \arctan(\varphi_1(t)\varphi_2(t)))
\]  

(51)

where \(e(t)\) is a white noise term with variance 0.1. An estimation data set of 500 points were generated, where \(\varphi(t) \in \mathbb{R}^2\) was drawn from a uniform distribution on \([-2.5, 2.5]^2\). A similar calibration data set was also generated, as well as a test set of 100 data points.

The function values were estimated for the test set using the proposed algorithm. For comparison, a sigmoidal neural network model was identified. Several models of different complexity were identified using the estimation data set, and the one performing best on the calibration data set, as well as on the test set, had 25 neurons. The results are shown in Table 1, where it can be observed that the performance of the proposed DWO algorithm is about 5\% better than the best neural network found.

<table>
<thead>
<tr>
<th>Estimation method</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWO with (\lambda)</td>
<td>0.1436</td>
</tr>
<tr>
<td>Neural network (25 neurons)</td>
<td>0.1517</td>
</tr>
</tbody>
</table>

Table 1: Performance of DWO solution path algorithm on a test set, measured as \(\frac{1}{n} \sum_j (f(\varphi^*(j)) - \hat{f}(\varphi^*(j)))^2\).

4 Conclusions

This paper has extended the use of piecewise linear solution paths suggested for LARS and LASSO in (Efron et al., 2004) to a more general setting, and proposed to use it for computation of DWO estimates. The benefit of exploiting the piecewise linear solution paths are that we can efficiently compute all solutions to a parametric optimization problem, such as the DWO problem for unknown bias-variance trade-off.

A cross-validation-based method for selecting the DWO weights was also proposed, and showed good performance in examples. There are many possible alternatives to this approach, including modified versions of Akaike’s criteria (Akaike, 1973), Mallows’ \(C_p\) criterion (Cleveland and Devlin, 1988; Mallows, 1973) etc. See also (Härdle, 1990; Stenman, 1999) and references therein. However, regardless of what method is used for choosing an appropriate \(\lambda\) value for the bias-variance trade-off, one can benefit from the reduced computational complexity obtained by simultaneously computing the solutions for all \(\lambda\) values.

A topic for further studies is to enhance the computational complexity of Algorithms 2.1 and 2.2 by taking advantage of the specific problem structures. For instance, to further increase the efficiency of Algorithm 2.2, one could consider to compute the expressions (35) and (36) recursively, similarly to what is done in, for instance, the RLS algorithm. Also numerical issues should be studied in more detail. Furthermore, it would be interesting to compute the solutions in the other direction, i.e., for \(\lambda\) starting at \(\infty\) and decreasing to 0. These could all be topics for further research.
References


**A Solution to the restricted parametric optimization problem (9).**

In this section, the solution to

\[
\min_x \lambda(c_k^T x + d_{kj}) + \frac{1}{2} x^T Q_{ij} x + f_{ij}^T x + r_{ij}
\]

subj. to \( \tilde{A}_j x = \tilde{b}_j \)

will be computed.

The Lagrangian of (52) is

\[
\mathcal{L}(x; \mu) = \lambda(c_k^T x + d_{kj}) + \frac{1}{2} x^T Q_{ij} x + f_{ij}^T x + r_{ij} - \mu^T (\tilde{A}_j x - \tilde{b}_j)
\]

Differentiating with respect to \( x \) and setting to zero yields

\[
0 = c_k x + Q_{ij} x + f_{ij} - A_j^T \mu
\]

\( \iff \) \( x = Q_{ij}^{-1} (A_j^T \mu - f_{ij} - c_k x) \)

Inserting this into \( \tilde{A}_j x = \tilde{b}_j \) gives

\[
\tilde{A}_j Q_{ij}^{-1} (A_j^T \mu - f_{ij} - c_k x) = \tilde{b}_j
\]

\( \iff \) \( \mu = (\tilde{A}_j Q_{ij}^{-1} A_j^T)^{-1} \left( \tilde{A}_j Q_{ij}^{-1} (f_{ij} + c_k x) + \tilde{b}_j \right) \)
and finally
\[ x = Q_i^{-1} \left( (\tilde{A}_j^T (\tilde{A}_j Q_i^{-1} \tilde{A}_j)^{-1} \tilde{A}_j Q_i^{-1} - I) (f_{ij} + c_{kj} \lambda) + \tilde{A}_j^T (\tilde{A}_j Q_i^{-1} \tilde{A}_j)^{-1} \tilde{b}_j \right) \]

\section*{B Uniqueness of (16)}

\textbf{Lemma 2.} Assume that \( Q \in \mathbb{R}^{m \times m}, A \in \mathbb{R}^{n \times m}, B \in \mathbb{R}^{p \times m}, \) that \( Q = Q^T > 0, \) and that \( (A_1^T B_0^T) \) has full row rank. Then
\[
\begin{pmatrix}
A & 1_n \\
B & 0
\end{pmatrix}
\]

is nonsingular.

\textbf{Proof.} It is equivalent to show that
\[
\begin{pmatrix}
Q & 0 & A^T & B^T \\
0 & 0 & 1_n^T & 0 \\
A & 1_n & 0 & 0 \\
B & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
w
\end{pmatrix}
= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\]

(53)

has a unique solution. To begin with, we can solve for \( x \) to get
\[ x = -Q^{-1}A^T z - Q^{-1}B^T w \]

and
\[
\begin{pmatrix}
0 & 1_n^T & 0 \\
-1_n & 0 & 0 \\
0 & -BQ^{-1}A^T & -BQ^{-1}B^T
\end{pmatrix}
\begin{pmatrix}
y \\
z \\
w
\end{pmatrix}
= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
\]

If we multiply from the left by \( (y \ z^T \ w^T) \), this yields
\[
0 = y1_n^T z + z^T 1_n y + (z^T \ w^T) \begin{pmatrix}
-AQ^{-1}A^T & -AQ^{-1}B^T \\
-BQ^{-1}A^T & -BQ^{-1}B^T
\end{pmatrix}
\begin{pmatrix}
z \\
w
\end{pmatrix}
\]

since \( 1_n^T z = 0 \) by the second block row of (53). Hence, since \( Q > 0, \)
\[
(z^T \ w^T) \begin{pmatrix}
A & 1_n \\
B & 0
\end{pmatrix} = 0
\]

Due to the full rank assumption, it follows that \( (z^T \ w^T) = 0. \) It now directly follows that also \( x = 0, \) and from the third block row of (53) we finally get \( y = 0. \) \qed
### Abstract

Recently, pathfollowing algorithms for parametric optimization problems with piecewise linear solution paths have been developed within the field of regularized regression. This paper presents a generalization of these algorithms to a wider class of problems, namely a class of parametric piecewise quadratic programs and related problems. It is shown that the approach can be applied to the nonparametric system identification method Direct Weight Optimization (DWO) and be used to enhance the computational efficiency of this method. The most important design parameter in the DWO method is a parameter ($\lambda$) controlling the bias-variance trade-off, and the use of parametric optimization with piecewise linear solution paths means that the DWO estimates can be efficiently computed for all values of $\lambda$ simultaneously. This allows for designing computationally attractive adaptive bandwidth selection algorithms. One such algorithm for DWO is proposed and demonstrated in two examples.

### Keywords

- System identification
- Non-parametric identification
- Parametric optimization
- Pathfollowing algorithms