An algorithm for optimization with disjoint linear constraints and its application for predicting rain

Tijana Janjić[*], Yvonne Ruckstuhl[*] and Philippe L. Toint[†]

September 11, 2019

Abstract

A specialized algorithm for quadratic optimization (QO, or, formerly, QP) with disjoint linear constraints is presented. In the considered class of problems, a subset of variables are subject to linear equality constraints, while variables in a different subset are constrained to remain in a convex set. The proposed algorithm exploits the structure by combining steps in the nullspace of the equality constraint's matrix with projections onto the convex set. The algorithm is motivated by application in weather forecasting. Numerical results on a simple model designed for predicting rain show that the algorithm is an improvement on current practice and that it reduces the computational burden compared to a more general interior point QO method. In particular, if constraints are disjoint and the rank of the set of linear equality constraints is small, further reduction in computational costs can be achieved, making it possible to apply this algorithm in high dimensional weather forecasting problems.

1 Introduction

We consider the problem

$$\min_{x,y} \mathcal{J}(x,y) \stackrel{\text{def}}{=} (g_x^T, g_y^T) \begin{pmatrix} x \\ y \end{pmatrix} + \frac{1}{2}(x^T y^T) \begin{pmatrix} P_{xx} & P_{xy} \\ P_{xy}^T & P_{yy} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{1}$$

subject to

$$Ax = b \quad \text{and} \quad y \leq \ell \tag{2}$$

where x and g_x belong to $\Re^n$, y and g_y to $\Re^p$, P_{xx} is an $n \times n$ symmetric real matrix, P_{yy} a $p \times p$ symmetric real matrix, P_{xy} a $n \times p$ real matrix, A an $m \times n$ real matrix with $m \leq n$ of rank m, $b \in \Re^m$ and the inequality is understood component-wise. It is easy to extend our discussion to more general bound constraints where one requires $\ell \leq y \leq u$ for some $\ell, u \in \Re^p$ with $\ell \leq u$, both of them possibly having infinite components. We refer to a problem of the type (1)-(2) as having *disjoint constraints* in the sense that the two sets of constraints of (2) involve disjoint sets of variables. In what follows, we focus on the convex case, and assume that

$$P \stackrel{\text{def}}{=} \begin{pmatrix} P_{xx} & P_{xy} \\ P_{xy}^T & P_{yy} \end{pmatrix}$$

[1]Ludwig-Maximilians University Munich, Theresienstrasse 37, 80333 München, Germany. Email: tijana.pfander@lmu.de,

[2]Namur Centre for Complex Systems,, University of Namur, 61, rue de Bruxelles, B-5000, Namur, Belgium. Email: philippe.toint@unamur.be.

is positive definite.

Our motivation for considering problem (1)–(2) arises from data assimilation in weather forecasting at the convective scale [16]. These forecast are produced by running a complicated numerical fluid-dynamics model in the future, starting from an initial condition. This initial condition is however only very partially known and is typically obtained by fitting available observational data with short-range model simulation using an optimization algorithm, a technique known as data assimilation. For convective scale applications, high resolution numerical models are used that are very sensitive to the proper specification of the initial conditions, i.e. predictions of the future can significantly differ even if the differences in the initial conditions are small [22].

Typical data assimilation algorithms produce the initial condition of the model by minimizing, perhaps iteratively in a Gauss-Newton framework, an objective function of type (1). Minimization is usually performed every hour using the new measurements of the atmosphere. The result of the minimization is a correction to a prediction of the numerical model for a given time. In this application, the vector $z = (x, y)$ to be estimated consists of variables describing the state of the atmosphere at a given time (such as pressure, temperature, wind direction and speed, $\dots$) in all grid points of the numerical model. Its size $n + p$ ranges between 10^6 and 10^9, resulting in truly large-scale problems. The vector y usually describes different water phases such as rain, graupel (i.e. soft hail) and snow at all grid points. The physical nature of these variables implies that they have to be non-negative. In this case, p is approximately one third of n, which remains very large.

In current practice, many data assimilation methods do not preserve the non-negativity of y. Either these variables are kept constant if the minimization algorithm attempts to make them negative, or their optimized values are simply projected onto the positive orthant. These techniques clearly interfere with the optimality of the computed result and the quality of the resulting prediction, especially given the ill-conditioning of the problem. This was demonstrated by [19] to the weather forecasting community using a simple example where preserving non-negativity of the y variables (the second part of (2)) as well as the total mass (the first part of (2)) during the minimization was shown to be beneficial. Further, [24] extended this observation to show that similar conclusions hold for a nonlinear multivariable model designed to test convective scale data assimilation applications. These papers use an active set or interior point quadratic optimization (QO) algorithm for solving the constrained minimization as implemented in matlab [11, 12] and python [1]. Although these results convincingly illustrate the benefits of including the constraints in the minimization, the QO algorithm has turned out to be difficult to implement in practice for applications such as weather forecasting at the convective scale. This is primarily due to the size of vectors z and y and the frequency of their estimation that is usually less than an hour. Both significantly limit the number of iterations of minimization algorithm. Further, often for weather forecasting at the convective scale, not only one but rather ensemble of predictions are produced in order to correctly specify, for example, uncertainty of rain at a particular location, even further increasing the computational considerations. It is the purpose of this short paper to show that alternative methods do exist and are significantly less expensive in computer time, thereby making a practical application affordable. While these considerations are based on data assimilation for weather forecasting, we note that the methods discussed here are also applicable to similar contexts in a wide variety of problems including chemistry, ecosystems and ocean data assimilation [3, 27, 28, 4], to mention a few.

In Section 2, we present an algorithm which exploits the fact that the constraints are disjoint. Section 3 then discusses the results obtained when applying this algorithm on a known convective-scale example due to [30]. In Section 4, we modify the algorithm given in Section 2 to exploit further properties of our problem, namely that the equality constraints of (2) are of (very) low rank, leading to further computational savings. A discussion and some perspectives are presented in Section 5.

2 The algorithm

For solving problem (1)–(2), we propose an active-set algorithm whose feature is to maintain feasibility with respect to the linear equality constraints on x at all iterations, while at the same time using classical projection techniques [7, Chapter 12] to enforce feasibility of the y.

If v is a non-negative vector in $\Re^p$ and $\mathcal{A} \subseteq \{1, \ldots, p\}$, we denote by $v^{\mathcal{A}}$ the vector v reduced to its active component, that is

$$v^{\mathcal{A}} = \begin{cases} [v]_i & \text{if } \; i \in \mathcal{A} \\ \ell & \text{otherwise,} \end{cases}$$

where $[v]_i$ denotes the i-th component of v. Similarly, $M^{\mathcal{A}}$ is the matrix M reduced to its active columns (and rows, if it is symmetric).

Our algorithm is stated as Algorithm 2.1 on this page.

Algorithm 2.1: QO algorithm for disjoint constraints

Step 0: Initialization. A feasible starting point (x_0, y_0) is given (i.e., $Ax_0 = b$, $y_0 \geq \ell$), as well as an accuracy threshold $\epsilon > 0$. Set $k = 0$.

Step 1: Active-set update.

$$\mathcal{A}_k \overset{\text{def}}{=} \{i \in \{1, \ldots, p\} \mid [y_k]_i = \ell \; \text{ and } \; \nabla_y \mathcal{J}(x_k, y_k) > 0\} \tag{1}$$

$$\mathcal{A}_k^c \overset{\text{def}}{=} \{i \in \{1, \ldots, p\} \mid i \notin \mathcal{A}_k\} \tag{2}$$

Step 2: Termination test. Terminate if $\|[\nabla_y \mathcal{J}(x_k, y_k)]^{\mathcal{A}_k^c}\| \leq \epsilon$.

Step 3: Search direction computation. Solve

$$\begin{pmatrix} P_{xx} & P_{xy}^{\mathcal{A}_k^c} & A^T \\ (P_{xy}^{\mathcal{A}_k^c})^T & P_{yy}^{\mathcal{A}_k^c} & 0 \\ A & 0 & 0 \end{pmatrix} \begin{pmatrix} s_k \\ v_k^{\mathcal{A}_k^c} \\ w_k \end{pmatrix} = - \begin{pmatrix} \nabla_x^{\mathcal{A}_k^c} \mathcal{J}(x_k, y_k) \\ \nabla_y^{\mathcal{A}_k^c} \mathcal{J}(x_k, y_k) \\ 0 \end{pmatrix} \tag{3}$$

Step 4: Projected search. Determine $\alpha > 0$ such that (x_{k+1}, y_{k+1}) is the first minimizer of $\mathcal{J}\left(x_k + \alpha s_k, \max\left[y_k + \alpha v_k, \ell\right]\right)$, where v_k is obtained from $v_k^{\mathcal{A}_k^c}$ by setting $[v_k]_i = 0$ for $i \in \mathcal{A}_k$.

If not available on the onset, a feasible point can be computed by solving a linear least-squares problem for x_0 and choosing any $y_0 \geq \ell$.

The third line of (3) imposes that $As_k = 0$. It is important that this equation be satisfied to high precision if exact feasibility with respect to the linear equality constraint is to be preserved. We refer the reader to [14] for a discussion of this point. With this caveat, the system (3) can be solved using a Krylov solver like MINRES or GMRES (see [25] for a description of these methods), or by a "constrained preconditioned" conjugate gradient method (see [13, 14]). If this is the case, any preconditioner must also be reduced (in its y part) to the subset of currently active variables $\mathcal{A}_k$. If dimension and sparsity of P allows (which is typically not the case in weather forecasting), a stable factorization can also be used to solve (3) accurately.

3 Numerical experiments

In order to illustrate the behaviour of the algorithm in our context, we use the modified shallow water model of [30]. This model has been used for testing different data assimilation algorithms in [17, 24]. The model is based on the shallow water (or Saint Venant) equations, which have been used for a long time in testing both numerical discretization schemes [26, 2, 9, 29, 20] as well as data assimilation algorithms [5, 31, 32]. As the name suggests, in [30] the shallow water equations have been altered in order to mimic key aspects of convection. To that end, a third variable rain r was introduced in addition to the velocity (or wind) u and water height level h fields. The one-dimensional modified shallow water model consists of following equations:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial(\phi + \gamma^2 r)}{\partial x} = \beta_u + D_u\frac{\partial^2 u}{\partial x^2}, \tag{1}$$

with

$$\phi = \begin{cases} \phi_c & \text{if} \quad h > h_c \\ gh & \text{otherwise,} \end{cases} \tag{2}$$

$$\frac{\partial r}{\partial t} + u\frac{\partial r}{\partial x} = D_r\frac{\partial^2 r}{\partial x^2} - \eta r - \begin{cases} \delta\frac{\partial u}{\partial x}, & \text{if } h > h_r \text{ and } \frac{\partial u}{\partial x} < 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

$$\frac{\partial h}{\partial t} + \frac{\partial(uh)}{\partial x} = D_h\frac{\partial^2 h}{\partial x^2}. \tag{4}$$

For the physically minded reader, we now briefly describe the meaning of the various involved quantities. Here, g is the gravity constant and h_c represents the admissible level of free convection. When this threshold is reached, the geopotential ϕ is set to a lower constant value ϕ_c. The parameters D_u, D_r, D_h are the corresponding diffusion constants, $\gamma = \sqrt{gh_0}$ is the gravity wave speed for the absolute fluid layer h_0 ($h_0 < h_c$). The small stochastic Gaussian forcing β_u is added at random locations to the velocity, in order to trigger perturbations and hence convection. Note that this implies that the location of convection is mostly random. The parameter δ is the production rate for rain and η is its removal rate. When h reaches the rain threshold h_r ($h_r > h_c$), rain is 'produced' by adding rain water mass to the potential, leading to a decrease of the water level and of buoyancy.

In our numerical implementation of the model, the one dimensional domain, representing 125 km is discretized with 250 points using standard second-order centered differences on a staggered grid. To compute evolution of the dynamical system, the time variable is discretized into time steps of 5 seconds. The model conserves mass, so the spatial integral over h is constant in time and the rain r cannot be negative. The other model parameters are given by

$$h_0 = 90\,m, \quad h_c = 90.02\,m, \quad h_r = 90.4\,m, \quad D_u = D_h = 25000\,m^2s^{-1}, \quad D_r = 200\,m^2s^{-1},$$
$$\phi_c = 899.77\,m^2s^{-2}, \quad \eta = 2.5\cdot 10^{-4}\,s^{-1}, \text{ and } \delta = 1/300.$$

The Gaussian stochastic forcing β_u has a half width of 4 grid points and an amplitude of 0.002 m/s. The fields produced by running this model with three random initial conditions are illustrated in Figure 1 after 60 model time steps (which is equivalent to five minutes in real time). As illustrated in Figure 1 position of clouds (height field) and rain are quite different after only 60 model time steps, mimicking fast changing convective storms whose intermittency is one of the challenges of data assimilation on convective scale. In practice, the availability of the radar data every 5-15 minutes would preferably be used to recover correct position and intensities of storms, while longer predictions (48 hours) would be issued routinely every 6 hours starting from improved initial condition.

To illustrate Algorithm 2.1 we perform a twin experiment, where we consider a model run to be the true state $z = \left(u^T, h^T, r^T\right)^T$, which we call the nature run. A vector of synthetic observations $z^{obs} \in \Re^o$ is then created by randomly perturbing the nature run such that $z^{obs} = Hz + \epsilon$, where H is the $o \times n$ matrix that determines the location of the observations and $\epsilon \in \Re^o$ is a random noise whose components depend on the observed variable and is computed as follows. For observations of wind and height field, a Gaussian observation noise is added to the wind u and height h fields with zero mean and standard deviations 0.001 m/s and 0.02 m, respectively. A lognormal noise is added to the rain field with parameters -8 and 1.8, yielding a very small observation bias of 0.000825 and standard deviation of 0.00185. For this choice of parameters, the observation error for each field is approximately 10% of the maximum deviation from the field mean. The prior state estimate $\tilde{z} = (\tilde{u}^T, \tilde{h}^T, \tilde{r}^T)^T$ is taken to be equal to nature run value at a random, much later time.

Given an estimate $\tilde{z}$ and observations z^{obs} of the true state of the atmosphere, we minimize a quadratic cost function based on the error covariance matrix of the state estimate B and the observations R respectively, in order to find an improved estimate $z^* = \left(u^{*T}, h^{*T}, r^{*T}\right)^T$ of the true state. We constrain the mass of h^* such that $e^T h^* = e^T \tilde{h}$ and $r^* \geq 0$. Specifically the minimization problem to be solved is [23, 8]:

$$\min_z \mathcal{J}(z) \overset{\text{def}}{=} \frac{1}{2}(z-\tilde{z})^T B^{-1}(z-\tilde{z}) + \frac{1}{2}(Hz - z^{obs})^T R^{-1}(Hz - z^{obs}) \tag{5}$$

subject to

$$e^T h = e^T \tilde{h} \quad \text{and} \quad r \geq 0. \tag{6}$$

Therefore, in our setup $n = 750$, $p = 250$, $x = (u^T, h^T)^T$ and $y = r$, $P = B^{-1} + H^T R^{-1} H$ and $g_x = -B^{-1}\tilde{z} - H^T R^{-1} z^{obs}$. A natural feasible starting point is $(x_0, y_0) = \left((\tilde{u}^T, \tilde{h}^T)^T, \tilde{r}\right)$. We estimate B as a sample covariance from the ensemble of 1000 model simulations that start from different initial conditions in which correlations that are 10 grids points apart are set to zero. The observation error covariance matrix R is taken to be diagonal with values on diagonal corresponding to variances of distributions used for generating observation error vector ϵ. We use an LU decomposition with pivoting to solve (3) accurately.

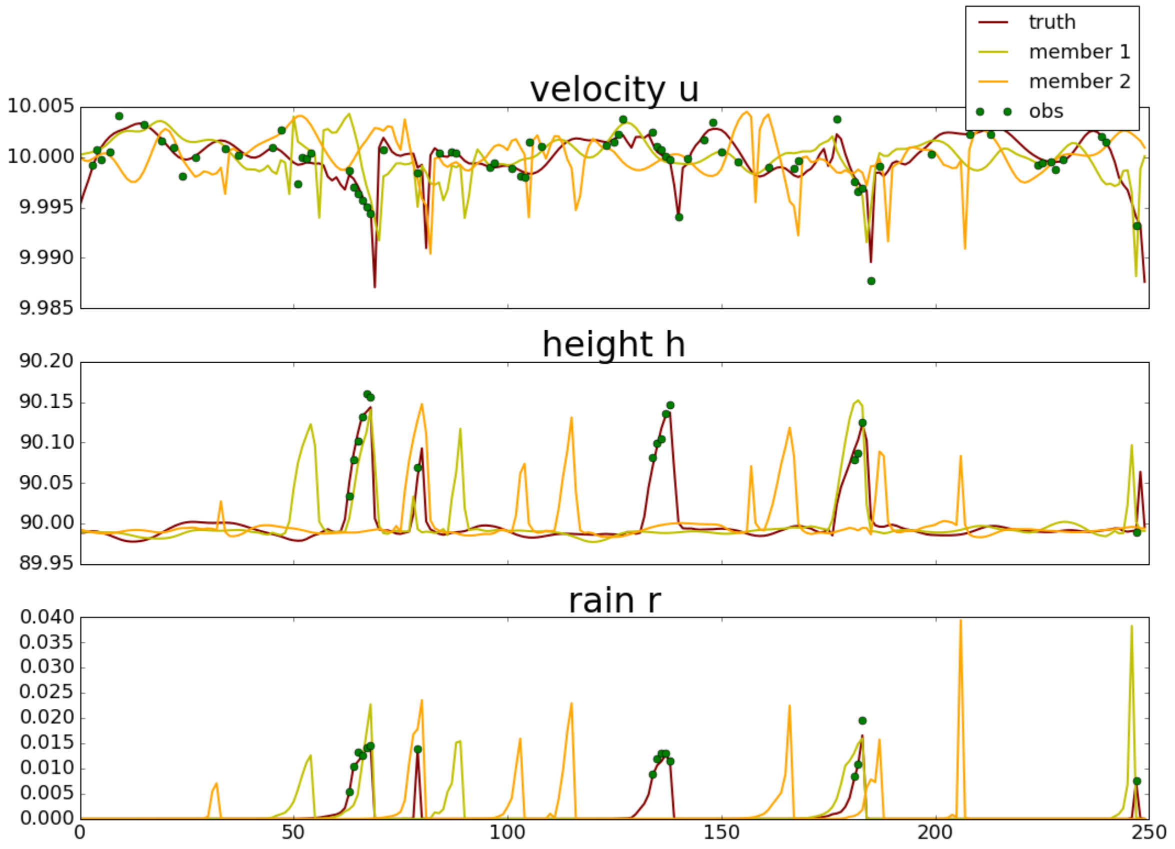

Figure 1: Fields produced by running the modified shallow water model with three different initial conditions after 60 model time steps. One of the experiments is chosen as nature run (red line). Observations (green circles) are simulated at locations where it rains in the nature run (truth) plus a random noise for all fields and, in addition for the u field, an extra 25% of observations are simulated at other locations. Noise is Gaussian for the h and u fields with zero mean and standard deviations of 0.001 m/s and 0.02 m, respectively. Noise is lognormal for the r field, yielding a very small observation bias of 0.000825 and standard deviation of 0.00185.

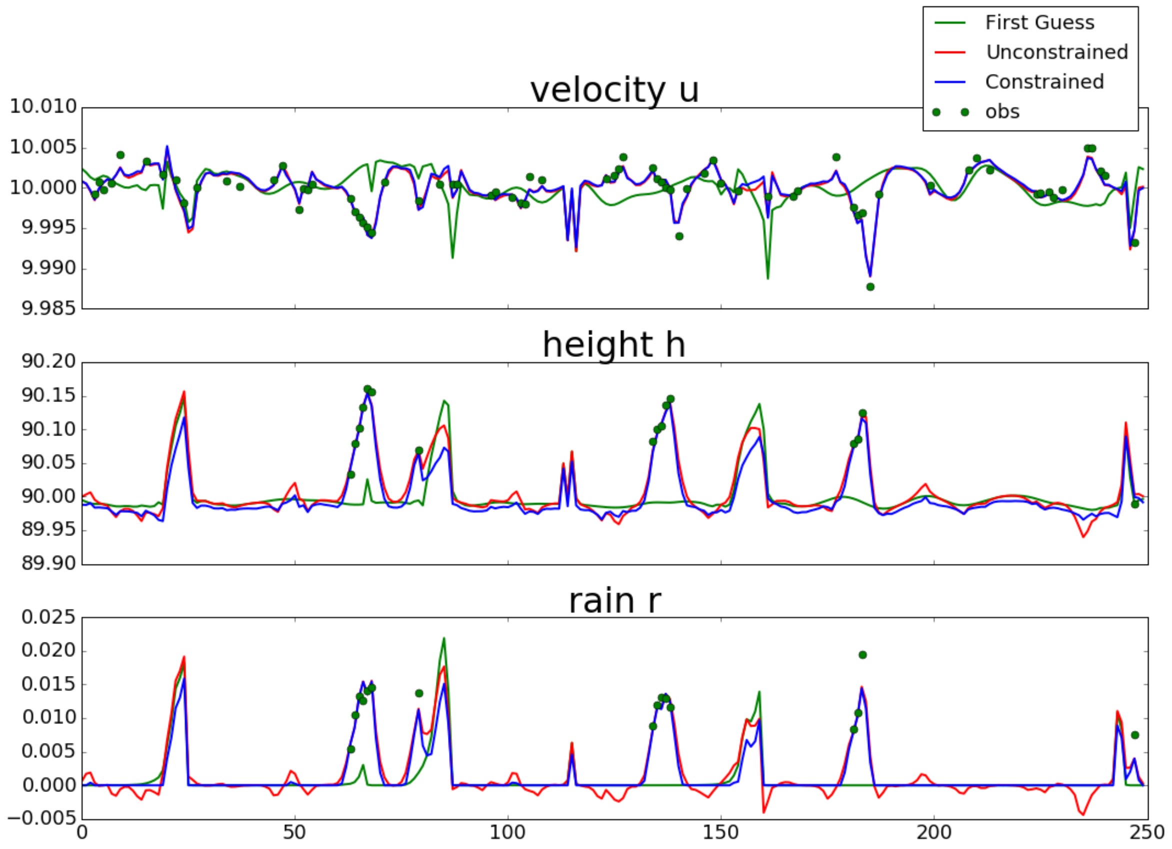

Figure 2: Results of minimization for unconstrained (red) and constrained (blue) problems based on prior estimate (green line) and observations (green circles).

Results for both constrained and unconstrained minimizations are illustrated in Figure 2. Constrained minimization produces a slightly smaller root mean square errors (RMSE) than unconstrained minimization. In addition, the value of rain is positive in all grid points. Although differences in RMSE between constrained and unconstrained minimization are small after one assimilation cycle, in [24] was shown that errors of unconstrained minimization will accumulate over time leading to large errors in total mass and total rain after repeating data assimilation 250 times, i.e. in less than one day. Table 1 illustrates the performance of the algorithm.

	$J(z)$	$\|\mathcal{A}_k^c\|$	$\|[\nabla_{y_i}\mathcal{J}(x_k,y_k)]^{\mathcal{A}_k^c}\|$	α_k
1	-1.799206e+03	133	4.811187e+02	1.
2	-1.804615e+03	157	8.878054e+01	1.
3	-1.805238e+03	160	1.084530e+01	1.
4	-1.805271e+03	162	2.373267e-01	0.9992
5	-1.805271e+03	162	4.156212e-12	1.

Table 1: Illustration of performance of the Algorithm 2.1.

As illustrated in Table 1, Algorithm 2.1 converges in only five iterations on this example. If a more general interior point method like the CVXOPT package [1] is applied for minimization of this problem, the number of iterations required is typically between ten and twenty.

4 The projected algorithm

Note that the matrix A in our previous example has a very simple form ($A = (0_u^T, e_h^T, 0_r^T)$) and is of size $1 \times n$. It obviously has rank one. We may then easily project the problem into the nullspace of A by defining $Z = I - A^T A/h^2$, the projection onto this nullspace, and applying the change of variable $x = Z\tilde{x}$ for $\tilde{x} \in \Re^{n-1}$, which leads to the problem

$$\min_{\tilde{x},y} \tilde{\mathcal{J}}(\tilde{x}, y) \overset{\text{def}}{=} ((Z^T g_x)^T, g_y^T) \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} + \frac{1}{2}(\tilde{x}^T y^T) \begin{pmatrix} Z^T P_{\tilde{x}x} Z & Z^T P_{xy} \\ P_{xy}^T Z & P_{yy} \end{pmatrix} \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} \tag{1}$$

subject to

$$y \geq \ell. \tag{2}$$

Problem (1)-(2) is now a bound-constrained quadratic problem, to which standard techniques can be applied, including for large-scale instances (see [6, 10, 21], for example). A typical such method applies the Conjugate Gradients (CG) [18] to minimize the quadratic in the current face, that is the subspace spanned by the inactive variables at the current iterate, restarting the procedure as needed when new bound constraints become active during the calculation and a new (lower dimensional) face must be explored. Note that the first face contains the negative gradient of the inactive variables, and the first step of CG performs a (in this case projected) line search along this direction, yielding what is known as the generalized Cauchy point. Methods differ essentially by their face changing mechanisms but insist that constraints active at the Cauchy point are not made inactive during the rest of the restarted CG steps. A simple version of the resulting algorithm (based on [6]) is now stated as Algorithm 4.1 on the next page.

In many cases, the efficient application of the CG algorithm requires preconditioning. We refer the reader to [15] for a discussion of suitable strategies in the context of data assimilation. It is also known that Algorithm 4.1 could be implemented without Step 4 if mere convergence is wanted, but that performing conjugate gradient iterations as suggested in [6], very often significantly reduces number of outer iterations. This was also observed for our test problem. If subproblems in Step 4 are solved accurately, Algorithm 4.1 requires three outer iterations, as illustrated in Table 2. If the number of conjugate gradient iterations per outer iteration is fixed a priori (a standard practice in weather forecasting), the number of outer iterations increases, and could reach twenty, but the cost of each outer iteration decreases. The behavior of the algorithm with the number of CG iterations fixed a priori to 25, 50, 400 and 800 is illustrated in Table 3. For the computational consideration, we also impose an additional stopping criteria to Algorithm 4.1: the algorithm is stopped either when it has converged or when the number of faces reduces to one. Note that the latter criteria is met only if the CG iterations for minimising equation (5) did not encounter any bounds prescribed by (6), which would suggest that the current guess of the active set is fairly accurate, though not guaranteed to be exact. Fixing the total number of CG iterations per outer iteration limits number of CG restarts during one major iteration and reduces accuracy as well as cost. For example, for a fixed number of 25 CG iterations, the solutions obtained by Algorithm 2.1 and

Algorithm 4.1: Projected QO algorithm for disjoint constraints

Step 0: Initialization. A feasible starting point $(\tilde{x}_0, y_0)$ is given (i.e. $y_0 \geq \ell$), as well as an accuracy threshold $\epsilon > 0$. Compute the projection Z onto the null space of A and set $k = 0$.

Step 1: Active-set update. Define

$$\mathcal{A}_k \stackrel{\text{def}}{=} \{i \in \{1, \dots, p\} \mid [y_k]_i = \ell\} \quad \text{and} \quad \mathcal{A}_k^c \stackrel{\text{def}}{=} \{1, \dots, p\} \setminus \mathcal{A}_k. \tag{3}$$

Step 2: Termination test. Terminate if the following conditions hold:

- $\|[\nabla_y \tilde{\mathcal{J}}(\tilde{x}_k, y_k)]^{\mathcal{A}_k^c}\| \leq \epsilon$
- $\|\nabla_{\tilde{x}} \tilde{\mathcal{J}}(\tilde{x}_k, y_k)\| \leq \epsilon$
- $\nabla_{y_i} \tilde{\mathcal{J}}(\tilde{x}_k, y_k) \geq 0 \quad \text{for} \quad i \in \mathcal{A}_k.$

Step 3: Find the Cauchy point and determine its active set. Determine $\alpha > 0$ such that $(\tilde{x}_k^c, y_k^c)$ is the first minimizer of

$$\tilde{\mathcal{J}}\left(x_k - \alpha \nabla_{\tilde{x}} \tilde{\mathcal{J}}(\tilde{x}_k, y_k), \max\left[y_k - \alpha \nabla_y \tilde{\mathcal{J}}(\tilde{x}_k, y_k), \ell\right]\right).$$

Set

$$\mathcal{A}_{k,C} \stackrel{\text{def}}{=} \{i \in \{1, \dots, p\} \mid [y_k^c]_i = \ell\}, \quad \text{and} \quad \mathcal{C}_k \stackrel{\text{def}}{=} \{1, \dots, p\} \setminus \mathcal{A}_{k,C}. \tag{4}$$

Step 4: Minimization beyond the Cauchy point. Apply the CG algorithm to find an approximate minimizer $(\tilde{x}_{k+1}, y_{k+1}^{\mathcal{C}_k})$ of

$$((Z^T g_x)^T, g_{y^{\mathcal{C}_k}}^T) \begin{pmatrix} \tilde{x} \\ y^{\mathcal{C}_k} \end{pmatrix} + \frac{1}{2}(\tilde{x}^T y^{\mathcal{C}_k,T}) \begin{pmatrix} Z^T P_{xx} Z & Z^T P_{xy^{\mathcal{C}_k}} \\ P_{xy^{\mathcal{C}_k}}^T Z & P_{y^{\mathcal{C}_k} y^{\mathcal{C}_k}} \end{pmatrix} \begin{pmatrix} \tilde{x} \\ y^{\mathcal{C}_k} \end{pmatrix} \tag{5}$$

subject to

$$y^{\mathcal{C}_k} \geq \ell^{\mathcal{C}_k}. \tag{6}$$

Terminate the CG once one (or more) bound(s) of indices $j_1, \dots, j_s$ are encountered, after a maximum number of iterations or once it has converged. If CG was terminated because bounds were encountered, restart it after redefining $\mathcal{C}_k = \mathcal{C}_k \setminus \{j_1, \dots, j_s\}$. Repeat this process until the size of $\mathcal{C}_k$ does not decrease anymore.

Algorithm 4.1 only coincide to two significant digits, while if 800 CG iterations are allowed, they share eleven significant digits. When allowing the number of CG iterations to increase from 25 to 800, the total number of iterations performed increases from 300 to 2400 and reaches 2472 in case where no limit is set while the cost increases by 75% for no-limit case. While these number are encouraging, they also indicate that more attention must be given to preconditioning.

	$J(z)$	$\|\mathcal{A}_k^c\|$	$\|[\nabla_y \mathcal{J}(x_k, y_k)]^{\mathcal{A}_k^c}\|$	α_k	CG its	faces	$\|z_k - z^*\|$
1	-1.794983e+03	191	6.762874e-09	6.921e-07	884	58	8.521e-02
2	-1.804782e+03	167	7.144318e-09	7.204e-06	825	9	2.193e-02
3	-1.805271e+03	162	3.945761e-09	1.012e-05	763	6	1.572e-12

Table 2: Illustration of performance of the Algorithm 4.1. In this table, “CG its” stands for the total number of CG iterations at major iteration k and “faces” is the number of explored faces at iteration k. To illustrate the accuracy, the difference is calculated between result of each major iteration z_k to z^* an end solution of Algorithm 2.1 on page 3.

5 Conclusion

We have presented two projection algorithms which exploit the disjoint nature of constraints typically occurring in weather forecasting applications. While projection methods may be inefficient when the combinatorial aspect of selecting the correct active bounds dominate and many faces need to be explored at each major iteration (in which case the interior-point algorithms perform better), they do perform well compared to the interior-point algorithms when the gradient quickly provides a good identification of the active constraints. This appears to be the case in our (representative) application.

The first of our methods, Algorithm 2.1, is more efficient than an interior point approach on a representative example, but still requires solving the KKT system (3), which is impractical in weather forecasting applications due to problem size and frequency of solution. By contrast, Algorithm 4.1 exploits the low rank of the linear equality constraints and uses a well-known iterative approach to compute a possibly approximate solution while ensuring satisfaction of the constraint. If the size of the problem is such that the conjugate gradient algorithm is allowed to converge, the number of outer iterations required by Algorithm 4.1 is smaller or comparable to that required by Algorithm 2.1. If the number of conjugate gradient iterations is limited from the start (as is often the case in weather forecasting applications), the number of outer iterations typically increases and finding the optimal equilibrium between accuracy and cost then depends on the problem at hand. A further advantage of Algorithm 4.1 is that its applies the conjugate gradient to a subproblem whose size is significantly smaller than that of the KKT system (3) (remember that $p \approx n/3$).

The observations made in this note are encouraging (and have already spurred some interest from the weather forecasting operational centers), but the authors are aware that adapting the proposed method(s) to a real production environment remains a significant task, as preconditioning and the details of the face changing mechanism will need thought and fine tuning.

References

[1] M. S. Andersen, J. Dahl, and L. Vandenberghe. Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Mathematical Programming, Series C*, 2:167–201, 2010.

[2] A. Arakawa and V. R. Lamb. A Potential Enstrophy and Energy conserving scheme for the Shallow Water Equations. *Monthly Weather Review*, 109, 18–36, 1980.

[3] L. Bertino, G. Evensen, and H. Wackernagel. Sequential data assimilation techniques in oceanography. *International Statistical Reviews*, 71:223–241, 2003.

[4] M. Bühner, A. Caya, L. Pogson, T. Carrieres, and P. Pestieau. A new environment Canada regional ice analysis system. *Atmosphere-Ocean*, 51(1):18–34, 2013.

[5] S. E. Cohn, and D. F. Parrish. The Behavior of Forecast Error Covariances for a Kalman Filter in Two Dimensions. *Monthly Weather Review*, 119(8): 1757–1785, 1991.

[6] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*. Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[7] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000.

[8] P. Courtier, E. Andersson, W. Heckley, J. Pailleux, D. Vasiljevic, M. Hamrud, A. Hollingsworth, F. Rabier, and M. Fisher. The ECMWF implementation of three-dimensional variational assimilation (3D-Var). I: Formulation. *Quarterly Journal of the Royal Meteorological Society*, 127, 1783–1807, 1998.

[9] M. J. P. Cullen, T. Davies, M. H. Mawson, J. A. James, and S. C. Coulter. An Overview of Numerical Methods for the Next Generation U.K. NWP and Climate Mode1. *Atmosphere-Ocean*, 35, 425–444,1997.

[10] Z. Dostál. Box constrained quadratic programming with proportioning and projections. *SIAM Journal on Optimization*, 7(3):871–887, 1997.

[11] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, UK, 1981.

[12] P. E. Gill, W. Murray, M. A. Saunders and M. H. Wright. *Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints. ACM Trans. Math. Software*, 10:282–298, 1984.

[13] N. I. M. Gould, M. E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4):1375–1394, 2001.

[14] N. I. M. Gould and Ph. L. Toint. An iterative working-set method for large-scale non-convex quadratic programming. *Applied Numerical Mathematics*, 43(1–2):109–128, 2002.

[15] S. Gratton, S. Gürol, and Ph. L. Toint. Preconditioning and globalizing conjugate gradients in dual space for quadratically penalized nonlinear-least squares problems. *Computational Optimization and Applications*, 54(1):1–25, 2013.

[16] N. Gustafsson, T. Janjić, C. Schraff, D. Leuenberger, M. Weissman, H. Reich, P. Brousseau, T. Montmerle, E. Wattrelot, A. Bucánek, M. Mile, R. Hamdi, M. Lindskog, J. Barkmeijer, M. Dahlbom, B. Macpherson, S. Ballard, G. Inverarity, J. Carley, C. Alexander, D. Dowell, S. Liu, Y. Ikuta, and T. Fujita. Survey of data assimilation methods for convective-scale numerical weather prediction at operational centres. *Quarterly Journal of the Royal Meteorological Society*, 144(713):1218–1256, 2018.

[17] M. Haslehner, T. Janjić, and G. C. Craig. Testing particle filters on simple convective-scale models. Part 2: A modified shallow-water model. *Quarterly Journal of the Royal Meteorological Society*, 142(697):1628–1646, 2016.

[18] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of the National Bureau of Standards*, 49:409–436, 1952.

[19] T. Janjić, D. McLaughlin, S. E. Cohn, and M. Verlaan. Conservation of mass and preservation of positivity with ensemble-type Kalman filter algorithms. *Monthly Weather Review*, 142:755–773, 2014.

[20] G. S. Ketefian, and M. Z. Jacobson. A mass, energy, vorticity, and potential enstrophy conserving lateral fluid-land boundary scheme for the shallow water equations. *Journal of Computational Physics*, 228,1–32, 2009.

[21] C. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

[22] E. N. Lorenz. The predictability of a flow which possesses many scales of motion. *Tellus*, XXI/3:289–307, 1969.

[23] A. C. Lorenc. A global three-dimensional multivariate statistical interpolation scheme. *Monthly Weather Review*, 109: 701–721, 1981.

[24] Y. Ruckstuhl and T. Janjić. Parameter and state estimation with ensemble Kalman filter based approaches for convective scale data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 144(712): 826–841, 2018.

[25] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, Boston, USA, 1996.

[26] R. Sadourny. The dynamics of finite-difference models of the shallow-water equations. *Journal of the Atmospheric Sciences*, 1, 119–143,1975.

[27] E. Simon and L. Bertino. Application of the Gaussian anamorphosis to assimilation in a 3-D coupled physical-ecosystem model of the North Atlantic with the EnKF: A twin experiment. *Ocean Science*, 5:495–510, 2009.

[28] E. Simon and L. Bertino. Gaussian anamorphosis extension of the DEnKF for combined state and parameter estimation: application to a 1D ocean ecosystem model. *Journal of Marine Systems*, 89:1–18, 2012.

[29] Sommer, M. and Névir, P. A conservative scheme for the shallow-water system on a staggered geodesic grid based on a Nambu representation. *Quarterly Journal of the Royal Meteorological Society*, 135, 485–494, 2009.

[30] M. Würsch and G. C Craig. A simple dynamical model of cumulus convection for data assimilation research. *Meteorologische Zeitschrift*, 23:483–490, 2014.

[31] Y. Zeng, and T. Janjić. Study of conservation laws with the Local Ensemble Transform Kalman Filter. *Quarterly Journal of the Royal Meteorological Society*, 142 (699): 2359–2372, 2016.

[32] Y. Zeng, T. Janjić, Y. Ruckstuhl, and M. Verlaan. Ensemble-type Kalman filter algorithm conserving mass, total energy and enstrophy. *Quarterly Journal of the Royal Meteorological Society*, 143(708): 2902–2914, 2017.

a)	$J(z)$	$\|\mathcal{A}_k^c\|$	$\|[\nabla_y\mathcal{J}(x_k,y_k)]^{\mathcal{A}_k^c}\|$	α_k	CG its	faces	$\|z_k - z^*\|$
1	-1.655016e+03	140	3.574082e+03	6.921e-07	25	24	2.780e-01
2	-1.791893e+03	164	1.546087e+03	1.411e-06	25	24	2.153e-01
3	-1.803668e+03	170	5.819331e+02	8.576e-07	25	15	1.787e-01
4	-1.804859e+03	168	4.755341e+02	9.515e-07	25	7	1.535e-01
5	-1.805179e+03	167	7.550745e+02	7.056e-07	25	7	1.033e-01
6	-1.805260e+03	164	1.648924e+02	6.691e-07	25	3	8.738e-02
7	-1.805270e+03	165	1.098210e+02	9.226e-07	25	4	5.906e-02
8	-1.755016e+03	164	1.129395e+02	7.539e-07	25	4	4.440e-02
9	-1.791893e+03	163	2.762388e+02	7.185e-07	25	3	2.991e-02
10	-1.803668e+03	162	3.200715e+01	6.718e-07	25	8	2.7e-02
11	-1.804859e+03	162	5.363492e+01	8.069e-06	25	4	1.682e-02
12	-1.805179e+03	163	1.593856e+01	7.072e-07	25	2	8.455e-03
13	-1.805260e+03	163	1.339967e+01	6.14e-07	25	2	5.098e-03
14	-1.805270e+03	163	8.526599e+00	6.484e-07	25	2	4.336e-03
15	-1.755016e+03	163	6.887983e+00	6.516e-07	25	2	3.584e-03
16	-1.791893e+03	163	3.851198e+00	6.514e-07	25	2	2.823e-03
17	-1.803668e+03	163	3.183931e+00	7.135e-07	25	2	2.201e-03
18	-1.804859e+03	163	2.726162e+00	7.205e-07	25	2	1.602e-03
19	-1.805179e+03	162	6.857200e+00	9.396e-07	25	1	5.692e-04

b)	$J(z)$	$\|\mathcal{A}_k^c\|$	$\|[\nabla_y\mathcal{J}(x_k,y_k)]^{\mathcal{A}_k^c}\|$	α_k	CG its	faces	$\|z_k - z^*\|$
1	-1.755016e+03	172	8.588824e+02	6.921e-07	50	41	2.292e-01
2	-1.791893e+03	172	9.320186e+02	9.247e-07	50	22	1.571e-01
3	-1.803668e+03	168	1.219848e+02	5.457e-07	50	10	6.399e-02
4	-1.804859e+03	163	2.176264e+02	2.483e-06	50	7	3.290e-02
5	-1.805179e+03	164	5.157843e+01	6.450e-07	50	4	1.802e-02
6	-1.805260e+03	162	2.105820e+01	6.865e-07	50	2	5.707e-03
7	-1.805270e+03	162	1.431093e+01	9.323e-07	50	1	1.158e-03

c)	$J(z)$	$\|\mathcal{A}_k^c\|$	$\|[\nabla_y\mathcal{J}(x_k,y_k)]^{\mathcal{A}_k^c}\|$	α_k	CG its	faces	$\|z_k - z^*\|$
1	-1.794983e+03	191	3.271074e-01	6.921e-07	400	58	8.522e-02
2	-1.804782e+03	167	7.592096e-03	7.204e-06	400	9	2.193e-02
3	-1.805271e+03	162	2.548443e-03	1.012e-05	400	6	2.849e-07
4	-1.805271e+03	162	2.351854e-08	5.207e-07	400	1	4.587e-12

d)	$J(z)$	$\|\mathcal{A}_k^c\|$	$\|[\nabla_y\mathcal{J}(x_k,y_k)]^{\mathcal{A}_k^c}\|$	α_k	CG its	faces	$\|z_k - z^*\|$
1	-1.794983e+03	191	2.929874e-07	6.921e-07	800	58	8.521e-02
2	-1.804782e+03	167	3.247265e-08	7.204e-06	800	9	2.193e-02
3	-1.805271e+03	162	5.351619e-09	1.012e-05	762	6	1.583e-12

Table 3: Illustration of performance of the Algorithm 4.1 when the maximum number of CG iterations per major iteration is fixed to a) 25, b) 50, c) 400 and d) 800 respectively. The notation follows that of Table 2.