Chapter 8

Reconstruction of Velocity Wind Fields from Horizontal Data by Projection Methods

L. Héctor Juárez¹, María Luisa Sandoval¹, Jorge López²

Abstract

For several meteorological problems and a large number of applications, the knowledge of a realistic wind field over a region is required. A successful method to generate adjusted wind fields from horizontal data is based on the mass conservation equation. This method leads to the solution of an elliptic problem for the multiplier (associated to the mass conservation constraint). However this models require a careful selection of boundary conditions and robust numerical methods to find the complete wind field. In this article we continue with a previous study in which we introduced two ways to solve the problem. Both methods can be considered as projection methods: an orthogonal $L_2$ projection method, and a reformulation of the problem as a saddle–point problem. For the elliptic problem, we consider a different approach to deal with the truncated boundary. For the saddle point problem, we introduce a preconditioned conjugate gradient algorithm, where the preconditioner is the elliptic problem associated to the first approach.

**keywords:** Mass conservation, elliptic problem, saddle-point problem, constrained minimization, preconditioned conjugate gradient.

¹Departamentos de Matemáticas, Universidad Autónoma Metropolitana Iztapalapa, A. P. 55-534, C.P. 09340, D. F., México, hect@xanum.uam.mx, mlss@xanum.uam.mx
²División de Ciencias Básicas, Universidad Juárez Autónoma de Tabasco.
Reconstruction of Velocity Wind Fields from Horizontal Data by Projection Methods

8.1 Introduction

In meteorology there are many problems and applications where the knowledge of a velocity vector field over a finite region is required. Some examples include dispersion of air pollutants in the atmosphere [1], [2], realization of wind maps for the design of different urban and general projects [3], among many others. Moreover, velocity fields are also required inputs for air quality models in meteorology [4]. In practice, however only limited horizontal wind field measurements are available, and the calculation of the vertical motion is of fundamental importance. Several models, methods and strategies, with various levels of complexity, have been proposed to address this problem. Diagnostic wind models require available interpolated data (generated by measurements from meteorological stations) to generate wind fields that satisfy some physical constraints. For instance, to assure mass conservation, a simplified steady state version of the continuity equation is imposed and the resulting model is then called a mass–consistent model. Mass–consistent models are attractive because of their simplicity and because they are easy and economical to operate. A review of these models is available in [5] and [6].

In this work we continue with the study, initiated in a recent article [7], of a variational mass–consistent model which is based on the original formulation by Sasaki [8]. This approach is a valuable tool for air quality applications and there have been several developments over the last decades [1], [3], [5], [6], [9], [10], [11], [12], [13]. The variational method proposed by Sasaki, uses the continuity equation in the form

\[ \nabla \cdot u = 0, \]  

(1)

where \( u \) is the wind velocity vector field that we want to recover on a given domain \( \Omega \) from the initial data \( u^0 \). This initial wind field \( u^0 \) is obtained by interpolation of an initial observed wind field, after the elimination of possible outliers. The vertical component of \( u^0 \) is taken as zero because meteorological stations usually do not measure this component. The usual approach to recover \( u \), [5], is based on the minimization of a functional \( L \) defined by

\[
L(u, \lambda) = \frac{1}{2} \int_{\Omega} \{ S (u - u^0) \cdot (u - u^0) + \lambda \nabla \cdot u \} \, dV.
\]

(2)

The new function \( \lambda \) is a Lagrange multiplier, and \( S \) is a diagonal matrix with weighting parameters \( \alpha_i^2, \ i = 1, 2, 3 \), called Gaussian precision moduli. The Euler–Lagrange equations associated to the Lagrangian (2) are:

\[
u = u^0 + S^{-1} \nabla \lambda, \]

(3)

\[
\lambda \delta u \cdot n = 0.
\]

(4)

From (1) and (3) we get the elliptic equation for the multiplier \( \lambda \):

\[ -\nabla \cdot (S^{-1} \nabla \lambda) = \nabla \cdot u^0. \]

(5)

After complementing this equation with convenient boundary conditions, and solving for the Lagrange multiplier \( \lambda \), the velocity field \( u \) is recovered using equation (3). There are two critical and important practical issues which have not been studied carefully by meteorologist:

- There is no general accepted criterion for choosing or estimating the values of parameters \( \alpha_i \) in matrix \( S \).
8.1 Introduction

There is no general consensus about the appropriate boundary conditions for the elliptic equation (5) to compute \( \lambda \).

Actually, we believe that these two issues are related and that the introduction of a matrix \( S \), which is different from the identity matrix, is a trick or an artifice to compensate the choice of inconsistent boundary conditions on the artificial boundary of the computational truncated domain.

Before we go further, we first introduce some terminology in order to formulate the problem more precisely in mathematical terms. Let \( \Omega \) be an open, simply connected and bounded region in \( \mathbb{R}^d \) (\( d = 2 \) or \( 3 \)) with Lipschitz boundary \( \Gamma = \partial \Omega \) decomposed as \( \Gamma = \Gamma_N \cup \Gamma_D \), where \( \Gamma_N \) is the part of the boundary associated to the surface terrain (topography) and \( \Gamma_D \) is the rest of the boundary, as shown in Figure 8.1. Vector \( \mathbf{n} \) is a unit outer normal vector at each point on the boundary. Given an initial vector field \( \mathbf{u}^0 \) in \( \Omega \), we want to compute a solenoidal vector function \( \mathbf{u} \) – called an adjusted field – as close to \( \mathbf{u}^0 \) as possible, in a least squares sense, such that \( \mathbf{u} \cdot \mathbf{n} = 0 \) on \( \Gamma_N \). Thus the idea is to project \( \mathbf{u}^0 \) into the space of divergence free vector functions. To formulate the problem in mathematical terms we introduce the following vector function spaces

\[
\begin{align*}
\mathbf{L}_2(\Omega) &= L_2(\Omega)^d, \text{ with } d = 2 \text{ or } 3, \quad (6) \\
\mathbf{H}(\text{div}; \Omega) &= \{ \mathbf{v} \in \mathbf{L}_2(\Omega) : \nabla \cdot \mathbf{v} \in L_2(\Omega) \}, \quad (7) \\
\mathbf{V} &= \{ \mathbf{v} \in \mathbf{H}(\text{div}; \Omega) : \nabla \cdot \mathbf{v} = 0 \text{ and } \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \Gamma_N \}. \quad (8)
\end{align*}
\]

Space \( \mathbf{V} \) is equipped with the norm \( ||\cdot||_{S,\Omega} \) associated to the inner product

\[
\langle \mathbf{u}, \mathbf{v} \rangle = \int_{\Omega} (S\mathbf{u}) \cdot \mathbf{v} \, dx, \quad (9)
\]

where \( \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{d} v_i w_i \) is the usual scalar product in \( \mathbb{R}^d \), and \( S = S(x) \in \mathbb{R}^{d \times d} \) is a symmetric positive definite matrix (usually diagonal). On \( \mathbf{V} \) we introduce the following convex quadratic functional:

\[
J(\mathbf{v}) = \frac{1}{2} \| \mathbf{v} - \mathbf{u}^0 \|_{S,\Omega}^2 = \frac{1}{2} \int_{\Omega} (S(\mathbf{v} - \mathbf{u}^0)) \cdot (\mathbf{v} - \mathbf{u}^0) \, dx. \quad (10)
\]
Therefore, the problem to generate the adjusted wind field \( u \) from a given initial field \( u^0 \) can be stated as follows:

Given \( u^0 \in H(div; \Omega) \), find \( u \in V \) such that \( J(u) \leq J(v), \forall v \in V \). \hfill (11)

A necessary and sufficient condition for \( J \) to have a minimizer \( u \in V \) is that

\[
\int_{\Omega} (S(u - u^0)) \cdot v \, dx = 0, \forall v \in V.
\hfill (12)
\]

Furthermore, the minimizer \( u \) is unique and it is given by equation (3), where \( \lambda \in H^1(\Omega) \) is the solution of the elliptic equation (5) (see [7] for the details).

Concerning the boundary conditions for the multiplier \( \lambda \) in the elliptic problem (5), the most common choices (among the meteorological community) are (a) \( \lambda = 0 \) on open or “flow through” boundaries, and (b) \( \partial \lambda / \partial n = 0 \) on closed or “no flow through” boundaries [5]. However, these boundary conditions are not physically nor mathematically justified, also they produce solutions which are poorly adjusted near the boundaries, and therefore they degrade the solutions, sometimes by several orders of magnitude. This is demonstrated in [7], where the solution obtained by solving (5), with different boundary conditions, is compared with the saddle–point formulation of the problem where the multiplier \( \lambda \) is not forced to take boundary values on the boundary. The saddle–point formulation is obtained from the Lagrangian (see equation 2)

\[
L(v, q) = J(v) + \langle q, \nabla \cdot v \rangle = \frac{1}{2} \int_{\Omega} (S(v - u^0)) \cdot (v - u^0) \, dx + \int_{\Omega} q \nabla \cdot v \, dx,
\hfill (13)
\]

defined on \( V_N \times L_2(\Omega) \), where \( V_N \) is the space of vector functions defined by

\[
V_N = \{ v \in H(div; \Omega) : v \cdot n = 0 \text{ on } \Gamma_N \}.
\hfill (14)
\]

A point \((u, \lambda) \in V_N \times L_2(\Omega)\) is a stationary point of the Lagrangian (13) if and only if it is solution of the saddle–point problem:

\[
\int_{\Omega} S u \cdot v \, dx + \int_{\Omega} \lambda \nabla \cdot v \, dx = \int_{\Omega} S u^0 \cdot v \, dx, \quad \forall v \in V_N,
\hfill (15)
\]

\[
\int_{\Omega} q \nabla \cdot u \, dx = 0, \quad \forall q \in L_2(\Omega),
\hfill (16)
\]

where \( \lambda \) need not to satisfy boundary conditions. The solution \( u \) is the minimizer of \( J \), but now it has to be obtained from the enlarged space \( V_N \) where divergence free is not satisfied. Instead, condition (1) is relaxed by introducing a Lagrange multiplier \( \lambda \), so that \( u \) must satisfy the weaker condition (16). This saddle–point problem is solved by an iterative conjugate gradient algorithm introduced in [7]. This algorithm does not degrade the solution near artificial boundaries, (vertical and top boundaries), and produce accurate numerical solutions which satisfy the constraint (1) almost to the desired tolerance, in a weak sense. However, to obtain a good accuracy about one thousand iterations are required.
In this work we mainly discuss another way to deal with the boundary conditions for the elliptic problem on truncated boundaries. On the other hand, we introduce a preconditioned conjugate gradient algorithm in order to accelerate convergence of the iterative method for the saddle-point problem. The organization of this article is as follows: in Section 2 we consider the elliptic problem (5), and we propose a new way to deal with the boundary conditions on the truncated boundary. In Section 3 we demonstrate that the elliptic problem (5) is actually an optimal preconditioner of the conjugate gradient algorithm introduced in [7]. Then, we present a preconditioned conjugate gradient algorithm with a mixed finite element discretization. We still working on the computer implementation of this preconditioned algorithm. However, we can anticipate that the number of iterations should be reduced from one thousand to about twenty, as it occurs when we solve problems in fluid mechanics. Finally, in Section 4 we establish some concluding remarks.

8.2 Elliptic problem: a different approach

8.2.1 Formulation of the elliptic problem

Given that \( u \) is of the form (3) and it satisfies \( \nabla \cdot u = 0 \) and \( u \cdot n = 0 \) on \( \Gamma_N \), then the Lagrange multiplier \( \lambda \) satisfies the following equations

\begin{align}
- \nabla \cdot (S^{-1} \nabla \lambda) &= \nabla \cdot u^0 \text{ in } \Omega, \quad (17) \\
-(S^{-1} \nabla \lambda) \cdot n &= u^0 \cdot n \text{ on } \Gamma_N. \quad (18)
\end{align}

Observe that the boundary condition (18) is imposed only on the physical boundary \( \Gamma_N \). We do not impose an explicit boundary condition on the artificial truncated boundary \( \Gamma_D \), since the bounded domain \( \Omega \) is arbitrary and it could be smaller or larger, depending of available information such as \( u^0 \). Instead, we enforce that the flux across the entire boundary be zero in order to guarantee mass conservation, i.e.

\[
0 = \int_{\Gamma} u \cdot n \, d\Gamma = \int_{\Gamma} (u^0 + S^{-1} \nabla \lambda) \cdot n \, d\Gamma.
\]

Now, this condition and (18) imply

\[
\int_{\Gamma_D} (u^0 + S^{-1} \nabla \lambda) \cdot n \, d\Gamma = 0. \quad (19)
\]

Then, the physical condition (19) must be enforced in some way when computing the solution of the elliptic problem. Equations (17)–(19) imply the identity

\[
\int_{\Omega} \nabla \cdot u^0 \, dx - \int_{\Gamma} u^0 \cdot n \, d\Gamma = 0.
\]

Actually this is also the compatibility condition associated to the Poisson–Neumann–like problem (17)–(19). Therefore, the problem has a unique solution \( \lambda \), up to a constant, in \( H^1(\Omega) \). If we define

\[
\Lambda = \left\{ q \in H^1(\Omega) \left| \int_{\Omega} q \, dx = 0 \right. \right\},
\]

\[
\quad (20)
\]
then, the variational formulation of the problem is: Given \( u^0 \in L^2(\Omega) \), find \( \lambda \in \Lambda \) such that

\[
\int_\Omega (S^{-1} \nabla \lambda) \cdot \nabla q \, dx = - \int_\Omega u^0 \cdot \nabla q \, dx + \int_{\Gamma_D} q (u^0 + S^{-1} \nabla \lambda) \cdot n \, d\Gamma, \quad \forall q \in \Lambda.
\]

(21)

Observe that if \( q \) were a constant function in \( \Omega \), we would recover (19).

### 8.2.2 Finite element approximation

We solve problem (21) by the finite element method. Let \( \mathcal{T}_h \) be a finite element triangulation of \( \Omega \subset \mathbb{R}^2 \), where \( h \) is the space discretization step [14]. Denote by \( P_1 \) the space of polynomials of degree less or equal than 1. Then, spaces \( L^2(\Omega) \) and \( \Lambda \), are approximated by the following finite dimensional spaces

\[
L_h = \{ v_h \in C^0(\Omega) \times C^0(\Omega) : v_h \big|_T \in P_1 \times P_1, \forall T \in \mathcal{T}_h \}, \tag{22}
\]

\[
\Lambda_h = \{ q \in C^0(\Omega) : q \big|_T \in P_1 \text{ and } \int_\Omega q \, dx = 0, \forall T \in \mathcal{T}_h \}, \tag{23}
\]

respectively. Thus, the discrete version of (21) is: Given \( u^0_h \in L_h \), find \( \lambda_h \in \Lambda_h \) such that

\[
\int_\Omega (S^{-1} \nabla \lambda_h) \cdot \nabla q \, dx = - \int_\Omega u^0_h \cdot \nabla q \, dx + \int_{\Gamma_D} q (u^0_h + S^{-1} \nabla \lambda_h) \cdot n \, d\Gamma, \quad \forall q \in \Lambda_h, \tag{24}
\]

where \( u^0_h \in L_h \) is the interpolant of the initial velocity field, \( u^0 \). Observe that the above discrete variational problem produces a nonsymmetric algebraic problem, because the boundary integral on the right hand side has the unknown \( \lambda_h \). We do not want to deal with this nonsymmetric problem, since this would require about twice the memory than the one required for the symmetric case. So, in order to keep the problem symmetric we consider two choices:

1. **Ghost Nodes.** The finite dimensional space \( \Lambda_h \) is generated from the set of piecewise linear “hat” functions \( \phi_i = \phi_i(x) \) (up to a constant) defined by

\[
\phi_i(x_j) = \delta_{ij},
\]

where \( \{x_i\}_{i=1}^N \) is the set of \( N \) nodes associated to the triangulation \( \mathcal{T}_h \) of \( \Omega \). If \( x_i \) is an interior vertex, then the boundary integral in (24) with \( q = \phi_i \) is zero, and

\[
\int_\Omega (S^{-1} \nabla \lambda_h) \cdot \nabla \phi_i \, dx = - \int_\Omega u^0_h \cdot \nabla \phi_i \, dx.
\]

(25)

So, if we introduce a layer of ghost nodes around and beyond \( \Gamma_D \), and we define \( \lambda_h = 0 \) or \( \partial \lambda_h / \partial n = 0 \) on those ghost nodes, then we only have to solve the symmetric algebraic system associated to (25). At the end we discard the solution on the ghost nodes and we only keep the solution values on the actual nodes. Actually this is a well known way to deal with PDE in domains with truncated boundaries.
2. **Iteration.** The solution obtained by the introduction of ghost nodes may be improved by the following iteration: We denote the initial solution for \( \lambda^0_h \), and then for each \( k \geq 1 \), we get \( \lambda^k_h \) solving the following problem

\[
\int_{\Omega} (S^{-1} \nabla \lambda^k_h) \cdot \nabla q \, dx = -\int_{\Omega} u^0_h \cdot \nabla q \, dx + \int_{\Gamma_D} q (u^0_h + S^{-1} \nabla \lambda^{k-1}_h) \cdot n \, d\Gamma, \quad \forall \, q \in \Lambda_h.
\]

We hope that about two iterations are sufficient. Once \( \lambda_h \) is computed, the numerical approximation \( u_h \) to the adjusted field \( u \) is calculated using the weak version of (3).

### 8.2.3 Numerical example

We consider the two dimensional solenoidal vector field \( u(x, y) = (x, -y) \) defined in \( \Omega = (1, 2) \times (0, 1) \), which satisfies the conditions \( \nabla \cdot u = 0 \) in \( \Omega \) and \( u \cdot n = 0 \) on \( \Gamma_N \). Assuming that we have the horizontal component \( u^0(x, z) = (x, 0) \) as the initial wind field, we want to see how much we can recover of the vertical component of \( u \). For this calculation we divide \( \Omega \) into a triangular mesh of size \( 80 \times 80 \), and we choose the values \( \alpha_1 = 1 \) and \( \alpha_3 = 0.001 \) for the Gaussian precision modula. We apply the previous algorithm, incorporating two layers of ghost nodes, and without doing iterations.

Figure 8.2 shows the exact and computed velocity fields, which are indistinguishable at a first look. To measure the global difference between the exact field \( u \) and the computed adjusted field \( u_h \) we use the following formula for the relative error

\[
e_r = \frac{||u - u_h||_2}{||u||_2},
\]

For the present example we obtain \( e_r = 2.1 \times 10^{-5} \). We also computed a mean value of the divergence of \( u_h \), defined as

\[
mdiv = \text{mean}_x \{ \nabla \cdot u_h(x_i) \mid x_i \text{ is an interior vertex of the computational mesh} \}
\]

where the point–wise divergence is computed in a weak sense. More precisely, if \( \phi_i \) is the piecewise linear base function associated to node \( x_i \), the weak divergence at the interior node \( x_i \) is defined as

\[
\nabla \cdot u_h(x_i) = -\int_{\Omega} u_h \cdot \nabla \phi_i \, dx.
\]

The weak divergence of the computed solution for the present example is \( mdiv = 1.6 \times 10^{-6} \).

We compare this solution with the one obtained in reference [7], where two different sets of boundary conditions on \( \Gamma_D \) were considered. For reference, from now on we denote by \( BCE \) the boundary conditions considered in this work. Thus the different sets of boundary conditions for comparison are:

- **BC1:** \(-S^{-1} \nabla \lambda \cdot n = u^0 \cdot n \) on \( \Gamma_N \), \( \lambda = 0 \) on \( \Gamma_D \),
- **BC2:** \(-S^{-1} \nabla \lambda \cdot n = u^0 \cdot n \) on \( \Gamma_N \), \(-S^{-1} \nabla \lambda \cdot n = 0 \) on \( \Gamma_V \), \( \lambda = 0 \) on \( \Gamma_T \),
- **BCE:** \(-S^{-1} \nabla \lambda \cdot n = u^0 \cdot n \) on \( \Gamma_N \), with two layers of ghost nodes on \( \Gamma_D \).
where $\Gamma_D$ was divided in two parts: the vertical artificial boundary $\Gamma_V$, and the top artificial boundary $\Gamma_T$. Table 8.1 shows how boundary conditions degrade numerical calculations. It is observed that the solution improves each time the Dirichlet boundary condition $\lambda = 0$ is applied to a smaller section of the non-physical boundary. This is not surprising, since this boundary condition introduces a large artificial gradient when calculating the term $\nabla \lambda$ in order to get $u = u^0 + S^{-1} \nabla \lambda$ at the corresponding boundary nodes.

<table>
<thead>
<tr>
<th>Case</th>
<th>$e_r$</th>
<th>$mdiv$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BC1$</td>
<td>$1.9 \times 10^{-2}$</td>
<td>$4.1 \times 10^{-2}$</td>
</tr>
<tr>
<td>$BC2$</td>
<td>$4.0 \times 10^{-4}$</td>
<td>$1.8 \times 10^{-2}$</td>
</tr>
<tr>
<td>$BCE$</td>
<td>$2.1 \times 10^{-5}$</td>
<td>$1.6 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

Table 8.1: Comparison of numerical solutions obtained with different sets of boundary conditions.

### 8.3 Preconditioned conjugate gradient algorithm

#### 8.3.1 An operator for the Lagrange multiplier

There are some effective numerical techniques to solve saddle-point problems like the problem (15)–(16). Here, we shall adapt a numerical technique, introduced by Glowinski [15], to solve generalized Stokes problems. This approach has proven to provide accurate solutions efficiently.
when it is applied to problems in fluid mechanics. We proceed as follows: Assume that \((u, \lambda)\) is solution of problem (15)–(16) with
\[
u = u^0 + u_\lambda, \tag{30}\]
where the initial velocity is tangent to \(\Gamma_N\). Then \(u_\lambda \in V_N\) solves
\[
\int_\Omega (S u_\lambda) \cdot v \, dx = - \int_\Omega \lambda \nabla \cdot v \, dx \quad \forall v \in V_N. \tag{31}\]
Since \(\nabla \cdot u = 0\), then from (30) it follows that \(-\nabla \cdot u_\lambda = \nabla \cdot u^0\), and this equation can be expressed in the functional form
\[
A \lambda = \nabla \cdot u^0, \tag{32}\]
where \(A : L_2(\Omega) \to L_2(\Omega)\) is an operator defined by
\[
Aq = -\nabla \cdot u^0, \tag{33}\]
with \(u_q \in V_N\) the solution of
\[
\int_\Omega (S u_q) \cdot v \, dx = - \int_\Omega q \nabla \cdot v \, dx \quad \forall v \in V_N. \tag{34}\]
Operator \(A\) is linear, and taking \(v = u_q'\), then (33) and (34) imply
\[
\int_\Omega (A q') \, q \, dx = - \int_\Omega q \nabla \cdot u_q' \, dx = \int_\Omega (S u_q) \cdot u_q' \, dx.
\]
The above properties imply that the operator \(A\) is self-adjoint and strongly elliptic. Therefore, equation (32) can be solved by the conjugate gradient algorithm:

1. **Initialization:** \(\lambda^0\) given, \(g^0 = A \lambda^0 = \nabla \cdot u^0, \quad d^0 = -g^0\).

2. **Descent:** For \(k \geq 0\), assuming we know \(\lambda^k, g^k, d^k\), find \(\lambda^{k+1}, g^{k+1}, d^{k+1}\) by

\[
\lambda^{k+1} = \lambda^k + \alpha_k d^k \quad \text{where} \quad \alpha_k = \langle g^k, g^k \rangle / \langle d^k, A d^k \rangle,
\]
\[
g^{k+1} = g^k + \alpha_k A d^k.
\]

3. **Test of convergence and new conjugate direction:**

If \(\langle g^k, g^k \rangle \leq \epsilon \langle g^0, g^0 \rangle\), take \(\lambda = \lambda^{k+1}\) and stop.

If not, compute
\[
d^{k+1} = -g^{k+1} + \beta_k d^k \quad \text{with} \quad \beta_k = \frac{\langle g^{k+1}, g^{k+1} \rangle}{\langle g^k, g^k \rangle}.
\]

Do \(k = k + 1\) and return to 2.

The nice properties of this iterative algorithm are discussed in [7], and it is shown how it produces excellent computational results when using a mixed finite element method to compute \(A d^k\) at each iteration. Actually most of the computational cost is due to this calculation which involves the solution of the integral equation (34) with \(q = d^k\). For the two dimensional problem, considered in the previous section, the obtained solution satisfies \(e_r = 5.9 \times 10^{-4}\) and \(m_{div} = -5.3 \times 10^{-12}\), after 1214 iterations.
8.3.2 Preconditioned conjugate gradient

The conjugate gradient algorithm is very effective in reducing the average divergence of the wind velocity field. However, the number of iterations required for convergence is more than one thousand. So, in order to obtain a more efficient iterative algorithm we need to reduce the number of iterations, and one way to do that is introducing a good preconditioner.

Let \( B : L_2(\Omega) \to L_2(\Omega) \) be an operator defined by

\[
Bq = \phi_q,
\]

where \( \phi_q \) solves the problem

\[
\int_{\Omega} (S^{-1}\nabla \phi_q) \cdot \nabla \psi \, dx = \int_{\Omega} q \, \psi \, dx \quad \forall \psi \in H^1(\Omega).
\]

The operator \( B \) is self-adjoint, positive definite, and satisfies \( ABq = q \) for every \( q \in L^2(\Omega) \). Therefore, we can take \( B^{-1} \) as a preconditioner for \( A \). Then, the preconditioned conjugate gradient algorithm to solve problem (15)–(16) is as follows:

1. Initialization: \( \lambda^0 \) given, \( g^0 = A\lambda^0 - \nabla \cdot u^0 \), \( \tilde{g}^0 = Bg^0 \), \( d^0 = -\tilde{g}^0 \).
2. Descent: For \( k \geq 0 \), assuming we know \( \lambda^k, g^k, \tilde{g}^k, d^k \), find \( \lambda^{k+1}, g^{k+1}, \tilde{g}^{k+1}, d^{k+1} \) by
   \[
   \lambda^{k+1} = \lambda^k + \alpha_k d^k \quad \text{where} \quad \alpha_k = \langle g^k, d^k \rangle / \langle d^k, Ad^k \rangle,
   \]
   \[
   g^{k+1} = g^k + \alpha_k A d^k,
   \]
   \[
   \tilde{g}^{k+1} = \tilde{g}^k + \alpha_k B(A d^k).
   \]
3. Test of convergence and new conjugate direction:
   If \( \langle g^k, \tilde{g}^k \rangle \leq \epsilon \langle g^0, \tilde{g}^0 \rangle \), take \( \lambda = \lambda^{k+1} \) and stop.
   If not, compute
   \[
   d^{k+1} = -\tilde{g}^{k+1} + \beta_k d^k \quad \text{with} \quad \beta_k = \frac{\langle g^{k+1}, \tilde{g}^{k+1} \rangle}{\langle g^k, \tilde{g}^k \rangle}.
   \]
   Do \( k = k + 1 \) and return to 2.

The previous algorithm can be expressed in more detail when we take into account equations (33)–(34), for the definition of operator \( A \), and equations (35)–(36) for the definition of operator \( B \). Then, the detailed conjugate gradient algorithm with preconditioning is as follows:

**Initialization**

1. Given \( \lambda^0 \in L_2(\Omega) \), solve for \( u^0_{\lambda} \in V_N \)
   \[
   \int_{\Omega} (S u^0_{\lambda}) \cdot v \, dx = -\int_{\Omega} \lambda^0 \nabla \cdot v \, dx, \forall v \in V_N.
   \]
2. Let \( g^0 = \nabla \cdot (u^0_0 + u^0) \).

3. Solve for \( \phi^0 \in H^1(\Omega)/\mathbb{R} \)

\[
\int_{\Omega} (S^{-1} \nabla \phi^0) \cdot \nabla \psi \, dx = \int_{\Omega} g^0 \psi \, dx, \ \forall \psi \in H^1(\Omega).
\]

4. Let \( \hat{g}^0 = \phi^0, d^0 = \hat{g}^0 \).

Descent

For \( m \geq 0 \), assuming \( \lambda^m, g^m, d^m, u^m \) are known, compute \( \lambda^{m+1}, g^{m+1}, d^{m+1} \) and \( u^{m+1} \), using the following steps:

5. Solve for \( \bar{u}^m \in V_N \)

\[
\int_{\Omega} (S \bar{u}^m) \cdot v \, dx = - \int_{\Omega} d^m \nabla \cdot v^m \, dx, \ \forall v \in V_N.
\]

6. Let \( \bar{g}^m = \nabla \cdot \bar{u}^m \).

7. Solve for \( \phi^m \in H^1(\Omega)/\mathbb{R} \)

\[
\int_{\Omega} (S^{-1} \nabla \phi^m) \cdot \nabla \psi \, dx = \int_{\Omega} \bar{g}^m \psi \, dx, \ \forall \psi \in H^1(\Omega).
\]

8. Let \( \alpha_m = \int_{\Omega} g^m \hat{g}^m \, dx \big/ \int_{\Omega} \bar{g}^m d^m \, dx \).

9. Set

\[
\lambda^{m+1} = \lambda^m - \alpha_m d^m, \\
u^{m+1} = u^m - \alpha_m \bar{u}^m, \\
g^{m+1} = g^m - \alpha_m \bar{g}^m, \\
\hat{g}^{m+1} = g^m - \alpha_m \phi^m.
\]

Test of convergence and new descent direction

If \( \int_{\Omega} g^{m+1} \hat{g}^{m+1} \, dx \big/ \int_{\Omega} g^0 \hat{g}^0 \, dx < \varepsilon \), then do \( \lambda = \lambda^{m+1}, u = u^{m+1} \) and stop. Otherwise, do the following

10. Compute \( \beta_m = \int_{\Omega} g^{m+1} \bar{g}^{m+1} \, dx \big/ \int_{\Omega} g^m \bar{g}^m \, dx \).

11. Set \( d^{m+1} = \hat{g}^{m+1} + \beta_m d^m \).

12. Do \( m = m + 1 \) and return to 5.

Note that in this algorithm, \( u \) and \( \lambda \) are computed simultaneously. Additional steps in this algorithm with respect to the conjugate gradient algorithm without preconditioning are mainly steps 3 and 7. Then, the additional cost of this algorithm, at each iteration, is the solution of the elliptic problem in step 7. However, this additional cost is offset by two nice properties: a) the preconditioning reduces dramatically the number of iterations (in CFD, the average number of iterations is between 10 and 20); b) there is a significant reduction of degrees of freedom in the discrete version of the elliptic problem in steps 3 and 7. We shall clarify this last point after having discretized the algorithm by the following mixed finite element method.
8.3.3 Discretization by a mixed finite element method

To approximate the functions belonging to the spaces \( V_N \) and \( L_2(\Omega) \), we make use of the Bercovier–Pironneau finite element approximation \([16]\). This a stable mixed method where the vector functions on \( V_N \), such as \( u_0^\lambda \), \( u_m^\lambda \) and \( \bar{u}_m^\lambda \), are approximated by continuous piecewise linear polynomials on a fine triangulation \( T_h \) of \( \Omega \). On the other hand, scalar functions on \( L_2(\Omega) \), such as \( \lambda_m \), \( g_m \), \( \bar{g}_m \), \( \hat{g}_m \), \( d_m \), are also approximated with piecewise linear polynomials, but this time on a triangulation twice as coarse, \( T_{2h} \) of \( \Omega \). The fine triangulation \( T_h \) is obtained from the coarse triangulation through a regular subdivision of each triangle \( T \in T_{2h} \), as shown in Figure 8.3. Then, the function spaces \( V_N \) and \( L_2(\Omega) \) are approximated by the following finite dimensional subspaces

\[
V_{N_h} = \{ v_h \in C^0(\overline{\Omega})^2 : v_h|_T \in P_1 \times P_1, \forall T \in T_h, v_h \cdot n = 0 \text{ on } \Gamma_N \}, \quad (37)
\]

and

\[
L_{2h} = \{ q_h \in C^0(\overline{\Omega}) : q_h|_T \in P_1, \forall T \in T_{2h} \}, \quad (38)
\]

respectively. We apply the mixed method described above, particularly in steps 1 and 5, as well as in the weak version of the steps 2 and 6 of the PCG–Algorithm.

Concerning the elliptic problems in steps 3 and 7, they are approximated over the coarse triangulation \( T_{2h} \). Scalar functions on \( H^1(\Omega) \), such as \( \phi^0 \) and \( \phi^m \) are approximated by continuous piecewise linear polynomials on each of the triangles of \( T_{2h} \). Then, \( H^1(\Omega) \) is approximated by means of the finite dimensional space

\[
H_{2h}^1 = \{ q_h \in C^0(\overline{\Omega}) : q_h|_T \in P_1, \forall T \in T_{2h} \}. \quad (39)
\]

Finally, scalar functions, such as \( g^0 \), \( \bar{g}^m \) are approximated by functions of \( L_{2h} \) defined in (38), as we have mentioned before.

Note that since \( u \) is obtained on the fine mesh, its resolution is the same as that obtained with the traditional algorithm. Also, if the trapezoidal rule is applied to calculate the integrals on the left hand side in steps 1 and 5, we obtain a system of algebraic equations with diagonal matrix, and the cost to solve them is only a vector multiplication. Then, the additional cost of the PCG–Algorithm
8.4 Concluding remarks

Compared to the cost of the CG–Algorithm is the solution of these elliptic problems in steps 3 and 7, but these problems are solved in a mesh twice as coarse. So, in a two-dimensional problem, the number of degrees of freedom (number of unknowns) in the resulting algebraic problem is about four times less than the number of degrees of freedom obtained when solving the elliptic problem with the traditional method described in Section 2. In a three-dimensional problem the number of degrees of freedom is about eight times less. According to this, the PCG–Algorithm algorithm saves memory on the matrix storage, compared with the EE–algorithm algorithm. This matrix can be pre-calculated before starting to iterate because it remains constant throughout the calculation process. Regarding the numerical calculations with this algorithm, work is under development and we hope to report numerical results soon.

8.4 Concluding remarks

Table 1 shows that boundary conditions can significantly affect numerical solutions, and that they may degrade the solution to a greater or lesser degree, depending of how we treat artificial truncated boundaries. We have proposed two additional ways of dealing with this problem: “ghost nodes”, and an iterative method, which we should explore more deeply. We think that the choice of appropriate boundary conditions is more crucial than the choice of parameters in the matrix $S$ to obtain good solutions. A bad choice of boundary conditions on non physical boundaries, such as the Dirichlet boundary condition $\lambda = 0$, produces poor results due to the introduction of spurious high gradients by the term $S^{-1}\nabla \lambda$ in formula (3), particularly on each node near the corresponding part of the boundary. The best result we can expect with this type of boundary conditions is to obtain a numerical solution with a weak divergence of the order of $10^{-2}$ and an overall relative accuracy of the same order.

Concerning the second approach, it is based on the iterative conjugate gradient algorithm applied to the functional equation obtained from the saddle point problem. In previous work we have shown that this method gives very good results. However, the number of iterations for convergence, in some problems, is typically on the order of several hundred, and sometimes around a thousand. This slow convergence motivated us to find a good preconditioner to accelerate convergence of the iterative method. It turned out that the preconditioner is an elliptic operator, which involves solving a Poisson problem. This preconditioner was derived following the idea of Cahouet and Chabard [17]. The extra work introduced by the preconditioner in the conjugate gradient method, is mainly the solution of elliptic problem in each iteration. However, this elliptic problem is solved in a coarse mesh, and it is four times smaller than the elliptic problem for the multiplier $\lambda$. Again, we need to consider appropriate boundary conditions to solve these elliptic problems. In short, we want to combine the preconditioned conjugate gradient with an efficient elliptic solver, but without degrading the numerical solutions. Clearly, the boundary conditions for the elliptic operator are important again, and we continue working on this problem.
Bibliography


