An Incomplete LU-Based Family of Preconditioners for Numerical Resolution of a Shallow Water System Using a Duality Method—Applications

M. CASTRO, J. MACÍAS AND C. PARÉS
Dpto de Análisis Matemático Universidad de Málaga
Campus de Teatinos s/n, 29080 Málaga, Spain
grouponamate.unam.es

(Received and accepted April 2000)

Communicated by R. Glowinski

Abstract—In this paper, we present a family of preconditioners well adapted to the solution of linear problems that arise from a particular discretization of shallow water equations in the flux form. The formulation of the shallow water equations used here is discretized in time using the method of characteristics and the Euler implicit method, and solved by a duality technique with automatic choice of parameters. The space discretization is performed using the first-order Raviart-Thomas finite element. The family of preconditioners designed for solving the linear problems that appear at each time iteration greatly improves convergence and significantly reduces the CPU time needed to solve them © 2001 Elsevier Science Ltd All rights reserved

Keywords—Preconditioners, Linear systems, Shallow water equations, Duality methods, FEM

1. INTRODUCTION

The aim of this paper is to describe a family of preconditioners that have proved to be very efficient at improving the convergence and reducing the computing time needed to solve a given numerical algorithm. The interest in the design of such preconditioners arises in the development of a numerical model for oceanographical applications. The convergence problems that may appear when solving the linear systems in such applications led us to analyse the nature of these problems and try to build a preconditioner (or a family of preconditioners) well adapted to this nature. The numerical model we talk about considered a multilayer shallow water approximation of the fluid (see [1]). In order to solve numerically the nonlinear problems appearing at each time iteration, an adaptation of the duality method proposed by Bermúdez and Moreno in [2] was used. In [1,3], a variant of this method which improves the convergence of the algorithm is proposed. To further improve the convergence of the algorithm, a new preconditioner was designed and used. In this paper, we will describe this family of incomplete LU-based preconditioners. In
Section 2, we introduce the problem that motivated us. Section 3 is devoted to the description of the preconditioners and in Section 4, we illustrate the behaviour of the preconditioner with two numerical examples.

### 2. SETTING THE PROBLEM: THE SHALLOW WATER MODEL

Let us consider the following formulation of the one-layer shallow water equations in flux form (as in [4])

\[
\frac{\partial Q}{\partial t} + \nabla \cdot (\bar{u} \otimes Q) + \frac{g}{2} \nabla (\eta^2 + 2\eta H) = g\eta \nabla H + F, \\
\frac{\partial \eta}{\partial t} + \nabla \cdot Q = 0,
\]

where \( \eta \) is the elevation above a reference level (the mean sea level, for example), \( H \) is the depth from the reference level to the bottom of the basin, and \( h = H + \eta \) is the thickness of the layer, \( \bar{u} \) is the average velocity through the depth, \( Q = h\bar{u} \) the flux, \( g \) the gravity, and \( F \) represents external forces (Coriolis, wind, and bottom effects) Here, \((0, T)\) is the time interval where the flow is studied and \( \Omega \) represents the horizontal projection of the volume occupied by water, whose boundary will be represented by \( \partial \Omega \) To solve this set of equations, suitable initial and boundary conditions must also be considered For sake of simplicity of the presentation, let \( Q \ n = 0 \) at \( \partial \Omega \) and \( F = 0 \) As the ideas behind the development of preconditioners do not need to introduce the multilayer model, we will keep the formulation (1) of the one-layer model (the reader is addressed to [1] for the multilayer formulation)

#### Numerical Resolution

The discretisation in time of the equations is performed by using a method of characteristics [5] for the convective term in the momentum equation, and an implicit Euler method in the mass conservation equation In order to perform a finite element discretisation of the resulting equations, we formulate them as a variational problem

\[
\text{find } (Q, \theta) \text{ such that } \forall z \in V = \left\{ v \in (L^2(\Omega))^2 \mid \nabla \cdot v \in L^2(\Omega), \ v \ n = 0 \text{ at } \partial \Omega \right\},
\]

\[
\frac{1}{\Delta t} \int_\Omega Q \ z \ dx - \frac{g}{2} \int_\Omega \theta \nabla \cdot z \ dx - g \int_\Omega (\eta_0 - \Delta t \nabla \cdot Q) \nabla H \ z \ dx = (F, z),
\]

\[
\theta(x) \in G_x(\eta_0 - \Delta t \nabla \cdot Q), \quad \text{a.e. } x \in \Omega,
\]

where \( G_x \) is the multivariate operator defined by

\[
G_x(\eta) = \begin{cases} 
\eta^2 + 2\eta H(x), & \text{if } \eta + H > 0, \\
(-\infty, -H^2], & \text{if } \eta + H = 0,
\end{cases}
\]

and \( \eta_0 \) would be \( \eta \) at time \( \Delta t n \), which is assumed to be known, and \( (F, z) \) is the term given by the method of characteristics Once solved, the solution at time \( \Delta t(n + 1) \) is given by \( (Q, \eta = \eta_0 - \Delta t \nabla \cdot Q) \) The problem to be solved can be written as

\[
a(Q, z) = \frac{g}{2} \int_\Omega p \nabla \cdot z \ dx + (L, z), \quad p = G_{x,\lambda}^w(\eta_0 - \Delta t \nabla \cdot Q + \lambda p),
\]

\[
a(Q, z) = \frac{1}{\Delta t} \int_\Omega Q \ z \ dx + \frac{\Delta t g}{2} \int_\Omega \nabla \cdot \int_\Omega \nabla Q(z) \ dx + \Delta t \int_\Omega \nabla Q(\nabla H \ z) \ dx,
\]

\[
(L, z) = (F, z) + g \int_\Omega \eta_0(\nabla H \ z) \ dx + \frac{g w}{2} \int_\Omega \eta_0(\nabla \cdot z) \ dx,
\]

where \( G_{x,\lambda}^w \) is the Yosida approximation of the operator \( G_x^w = G_x - wI \) (see [2])
Problem (4) suggests a point fixed algorithm with $\lambda w \leq 1/2$ for the operator $G_{x,\lambda}^{w,\ldots}$ to be Lipschitz. The convergence of $Q^n$ to $Q$ is assured, at least in norm $L^2$ if $\lambda w = 1/2$. The good performance of the method strongly depends on a right choice of the parameters $(\lambda, w)$ and, in general, this choice is not easy to perform, as $\lambda$ and $w$ have no relationship with the physics of the problem. In this case, optimal values of $\lambda$ and $w$ can be computed (see [1,3]).

$$\lambda_x = \frac{1}{4 (\eta_0(x) + H(x))}, \quad \forall x \in \Omega \text{ and } w_x = \frac{1}{2\lambda_x}$$

To discretise problem (4) in space, Raviart-Thomas mixed finite elements [6] have been used. Let $\Omega_h$ be a triangulated mesh of the bidimensional domain $\Omega$, we define $V_h$, the space of vector functions which are polynomials of degree one on each triangle, discontinuous on the edges of the elements but whose normal components are continuous and constant on them and $M_h$ the space of piecewise constant functions.

3. RESOLUTION OF THE LINEAR SYSTEMS—PRECONDITIONERS

The numerical scheme used here to discretise the shallow water equations needs, at each time step $t_n$, the resolution of a fixed-point problem. At each iteration of the fixed-point algorithm, we have to solve a linear system under the form, $A^n Q_h = b$, where $A^n$ is the matrix appearing in the space discretisation of the bilinear form defined in (4). Matrices $A^n$ have the following properties $A^n Q_h = \alpha_n \|Q_h\|_2, \forall Q_h \in V_h$, $A^n = \tilde{A} + \tilde{A}^n(\omega_x^{n,h})$, $n \geq 0$, where $\tilde{A}$ is time independent and $\tilde{A}^n(\omega_x^{n,h})$ depends on $\omega_x^{n,h}$. They are not necessarily symmetric and their zero-structure is the classical of matrices arising from a finite element discretisation. In general, their conditioning numbers, $K(A^n)$, are high and depend on $\omega^n_x$. This entails an additional difficulty if a generalised conjugate gradient method is going to be used. Therefore, a good choice of the preconditioner matrices $C^n$ is needed.

To improve the convergence speed of the method, we initially use what we have have called an LU(0) preconditioner matrix. This is an LU incomplete factorization with the same zero-structure of matrices $A^n$ (see [7]). Although this preconditioner appears efficient in numerical simulations performed with small time steps, and constant $\omega^n_x$, the computing time increases considerably with increasing time steps and variations of $\omega^n_x$, which may even lead to problems of convergence. Therefore, an analysis of the linear systems obtained was needed in order to undertake this difficulty.

A Brief Analysis of the Numerical Problem

A general study of the linear problems appearing in previous sections is difficult due to the fact that the bilinear form (4) is not symmetric. Here, we present a simplification of the original problem where the function $H(x,y)$ is supposed to be constant in $\Omega$. Under this assumption, the bilinear form (4) is symmetric. Nevertheless, numerical experience shows (see Section 4) that even if $H(x,y)$ is not constant, the same behaviour is observed.

If $H(x,y)$ is constant, the bilinear form (4) can be rewritten as

$$a_h^n(Q_h^n, z_h) = \sum_{K \in \Omega_h} \frac{1}{\Delta t} \int_K Q_h^n z_h \, dx + \frac{\Delta t}{2} \int_K \omega_x^{n,h} \nabla \cdot Q_h^n \nabla z_h \, dx, \quad \forall z_h \in V_h$$

From the previous expression, we obtain

$$c^n_0 a_h^n(z_h, z_h) \leq a_h^{n+1}(z_h, z_h) \leq c^n_0 a_h^n(z_h, z_h),$$

where $c^n_0 = 1 + \max_{x \in \Omega_h} |\omega_x^{n+1} - \omega_x^n|$, and $c^n_1 = 1 - \max_{x \in \Omega_h} |\omega_x^{n+1} - \omega_x^n|$.
As $w_{x,h} = 2(H_h + \eta_h^n)$, $\max_{x \in \Omega_h} |\omega_{x,h}^{n+1} - \omega_{x,h}^n|$ is controlled by $2 \max_{x \in \Omega_h} |\eta_h^n - \eta_h^{n+1}|$, therefore, on the assumption that fluctuations in $\eta$ are small (i.e., $\eta_h^n$ is close to $\eta_h^{n+1}$), the difference $|\omega_{x,h}^{n+1} - \omega_{x,h}^n|$ is small, $\forall x \in \Omega$.

As $A^n$ is symmetric, $K_2(A^n) = \rho^n_{\max}/\rho^n_{\min}$, where $\rho^n_{\max}$ and $\rho^n_{\min}$ are, respectively, the greatest and smallest eigenvalues of $A^n$. If we assume that $\max_{x \in \Omega_h} |\omega_{x,h}^{n+1} - \omega_{x,h}^n| < 1$, then (6) gives

$$K_2 (A^{n+1}) \leq \frac{c_0^n}{c_1^n} K_2 (A^n),$$

where $c_0^n/c_1^n$ is a constant depending on the fluctuation of $\omega_{x,h}^n$ and $\omega_{x,h}^{n+1}$.

**Observation 1** Note that, if $\max_{x \in \Omega_h} |\eta_h^{n+1} - \eta_h^n|$ is not small enough, $c_1^n$ is no longer smaller than one. Nevertheless, in the (oceanographical) problem that motivates our study, the $\eta$-fluctuations are small.

**LU($k$) Preconditioner Matrices**

Let us consider the following definition to denote the preconditioners introduced here, depending on a parameter $k$ ($0 \leq k \leq 100$).

**Definition 1** Let $C$ and $A$ be square matrices of the same order. We call $C$ an $LU(k)$-type precondition matrix of $A$, if $C$ is an incomplete LU factorization of $A$ with the following two properties:

1. It has, at least, the same nonzero structure of matrix $A$.
2. The $k%$ of the nonzero coefficients of the complete LU factorization are kept.

**Observation 2** Note that an $LU(0)$ precondition matrix has the same zero-structure as matrix $A$. This is also true for all $LU(k)$-type precondition matrices with $k < k_0$, for a given $k_0$ depending on the number of nonzero coefficients of the complete LU factorised matrix.

The most delicate part in the process of building an $LU(k)$-type preconditioner matrix is the computation of its zero-structure, not all the coefficients and diagonals have the same “weight” in the complete LU decomposition. The structure of the discretised problem described in previous sections allows us to develop a well-adapted strategy to determine the zero-structure of the preconditioner matrices. We develop this strategy in the next paragraph.

**Nonzero Structure of the LU($k$)-Type Preconditioner Matrices**

Keeping in mind (7), it stands to reason that if $C^n$ is a good preconditioner for $A^n$, it will also be a good preconditioner for $A^{n+1}$, so it is reasonable to say that the structure of nonzero coefficients remains unchanged during the numerical integration. Using this principle, only the zero-structure of the initial preconditioner $C^0$ has to be computed and $C^n$, $\forall n \in N$, inherits this structure.

The procedure followed to determine the structure of the $LU(k)$-type preconditioner matrix, $C^0$ for $k > 0$ can be split into the following steps:

1. Compute the full LU factorization of matrix $A^0$ at the initial time $t = 0$.
2. For each diagonal, $d_i$, of the matrix $L$, (respectively, $U$) compute the norm

$$\|d_i\| = \sqrt{\sum_{l=1}^{m-1} a_{l+1,i}^2/m - i},$$

where $m$ represents the number of degrees of freedom and the index $i$ gives the $i^{th}$ diagonal ($i = 0$ for the principal diagonal).
3 Compute the $k$-pivots $L_{\text{min}}$ and $U_{\text{min}}$ so that approximately the $k\%$ of the absolute value of the coefficients of $L$ (respectively, $U$) are greater than $L_{\text{min}}$ (respectively, $U_{\text{min}}$).

4 We impose on the preconditioner matrix $C^0$ to have, at least, the same nonzero structure of matrix $A^0$. Note that the structure of $A^n$ is the same for all $n \geq 0$, because it only depends on mesh connectivities.

5 The former minimal structure is enriched, if necessary, up to the $k\%$ of the nonzero elements of $L$, and $U$ is reached. This is done, from the diagonals with greatest norms up to the diagonals with smaller norms, by the elimination of the coefficients of $L$ (respectively, $U$), whose absolute values are smaller than $L_{\text{min}}$ (respectively, $U_{\text{min}}$).

Once the zero-structure is defined for $C^0$, and, consequently, for $C^n$, $\forall n > 0$, their coefficients must be calculated. At this stage, an usual algorithm for computing an incomplete LU factorization is used (see [7,8] for further details).

In practice, we can avoid computing the coefficients of $C^n$ at each time iteration. In fact, as $A^n \approx A^{n+1}$ and $K_2(A^n) \approx K_2(A^{n+1})$, their preconditioners, $C^n$ and $C^{n+1}$, respectively, are practically the same, therefore, $C^n$ is used instead of $C^{n+1}$. The coefficients of the preconditioner matrix are computed only if the fluctuation between matrices is large. This strategy is very efficient for any time step and does not present convergence problems (see Tables 1 and 2 for numerical results).

### Table 1 Performance of the preconditioners Test 1 Constant depth

<table>
<thead>
<tr>
<th>$k$</th>
<th>CPU Time</th>
<th>Iterat</th>
<th>$E_f(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7603</td>
<td>105-100</td>
<td>1.00</td>
</tr>
<tr>
<td>15</td>
<td>3640</td>
<td>38-35</td>
<td>1.08</td>
</tr>
<tr>
<td>20</td>
<td>2456</td>
<td>25-22</td>
<td>1.38</td>
</tr>
<tr>
<td>30</td>
<td>1587</td>
<td>15-12</td>
<td>1.33</td>
</tr>
<tr>
<td>40</td>
<td>1397</td>
<td>10-8</td>
<td>1.14</td>
</tr>
<tr>
<td>50</td>
<td>1246</td>
<td>8-6</td>
<td>1.02</td>
</tr>
<tr>
<td>60</td>
<td>1184</td>
<td>6-5</td>
<td>0.90</td>
</tr>
<tr>
<td>70</td>
<td>1143</td>
<td>4-3</td>
<td>0.78</td>
</tr>
<tr>
<td>80</td>
<td>1124</td>
<td>3</td>
<td>0.69</td>
</tr>
<tr>
<td>90</td>
<td>1117</td>
<td>3-2</td>
<td>0.61</td>
</tr>
</tbody>
</table>

### Table 2 Performance of the preconditioners Test 2 Channel with variable depth

<table>
<thead>
<tr>
<th>$k$</th>
<th>CPU Time</th>
<th>Iterat</th>
<th>$E_f(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11902</td>
<td>110-105</td>
<td>1.00</td>
</tr>
<tr>
<td>15</td>
<td>3819</td>
<td>30-20</td>
<td>1.68</td>
</tr>
<tr>
<td>20</td>
<td>3258</td>
<td>25-23</td>
<td>1.57</td>
</tr>
<tr>
<td>30</td>
<td>2403</td>
<td>15-13</td>
<td>1.35</td>
</tr>
<tr>
<td>40</td>
<td>1932</td>
<td>10-8</td>
<td>1.29</td>
</tr>
<tr>
<td>50</td>
<td>1662</td>
<td>7-6</td>
<td>1.18</td>
</tr>
<tr>
<td>60</td>
<td>1523</td>
<td>5-4</td>
<td>1.07</td>
</tr>
<tr>
<td>70</td>
<td>1503</td>
<td>4-3</td>
<td>0.93</td>
</tr>
<tr>
<td>80</td>
<td>1469</td>
<td>4-3</td>
<td>0.82</td>
</tr>
<tr>
<td>90</td>
<td>1488</td>
<td>3-2</td>
<td>0.72</td>
</tr>
</tbody>
</table>

### 4. NUMERICAL EXAMPLES

This section presents a comparison between the different preconditioners obtained by varying the value of the parameter $k$. The computational domain is a 64 km long ($x_1$), 21 km wide ($x_2$) channel. A $1\text{ Sv}$ ($1\text{ Sv} = 10^6 \text{ m}^3\text{s}^{-1}$) input flow is imposed at the edge of the channel located at $x_1 = 0$. The same output is imposed at the other end of the channel, located at $x_1 = 64$. At the channel walls, normal components of the flow equal to zero are considered. As initial conditions, the water is taken to be at rest and with no elevation at the free surface.

In the first example, the depth of the channel is taken to be constant and equal to 100 m. In the second example, the depth of the channel is given by the following sinusoidal function $H(x_1,x_2) = 90 \cos(x_1/4000 + \pi/2) + 100$.

Problem (1) has been solved in the domain described above until reaching a stationary state. The mesh considered for this comparison has 4855 nodes. The time step is $\Delta t = 120\text{ s}$, and the stationary state is reached after 800 iterations.

Table 1 summarizes some of the data referring to Test 1 channel with constant depth. The total CPU time spent in a dual-PentiumII 333 MHz (under Linux system), depending on the values of the parameter $k$ in Table 1 displayed, as well as the mean number of iterations of the iterative method used to solve the linear system (iterat).
We can see that the reduction in computing time and number of iterations is larger for values of \( k \) between 0 and 30. In order to measure the performance of the LU\((k)\) preconditioner matrices, we have introduced the following definition for efficiency:

\[
E_f(k) = \frac{\text{CPU-time LU}(0)/\text{CPU-time LU}(k)}{\text{nonzeros LU}(k)/\text{nonzeros LU}(0)}
\]  

(8)

In other words, efficiency \( E_f(k) \) gives the ratio between the average CPU-time spent and the average of nonzero coefficients (or used memory) related to the standard incomplete LU factorization. Note that \( E_f(0) = 1 \). Table 1 shows the evolution of the efficiency, \( E_f(k) \), as a function of \( k \). Note that the greatest values for the efficiency are concentrated around \( k = 20 \) and \( k = 40 \). For these parameters, the efficiency is greater than 1.

Table 2 gathers the data from the numerical experiment performed in the channel with variable depth Test 2. Note that if \( k = 0 \), total CPU time rises up to 11902 units, while value \( k = 15 \) reduces computing time down to 3819 units. Again, there can be observed a larger reduction of CPU time for \( k \) values between 15 and 40. Note that in this example, the largest values for \( E_f(k) \) are also concentrated between \( k = 15 \) and \( k = 40 \).

5. CONCLUSIONS

We have shown that the LU\((k)\)-type preconditioner considered in this paper improves convergence speed in the resolution of linear problems, and, consequently, the CPU time spent in solving shallow-water problems. The same strategy can be used in problems where matrices are time dependent and their conditioning numbers are practically the same.

REFERENCES