APPLICATION OF THE SPARE APPROXIMATE INVERSE PRECONDITIONER TO SHALLOW WATER PROBLEM

GUANGHUI WANG¹, XUESHUN SHEN¹, FENGFENG CHEN¹ AND ZHONGZHEN JI²

Abstract. To improve the computation speed in numerically solving weather equations, we investigate the use of sparse approximate inverse preconditioners for numerically solving shallow water equations. This is a fast algorithm for solving large scale linear equations. Some strategies for determining the nonzero of an approximate inverse are described in this paper. As an example, we use the GMRES iterative algorithm to solve the finite difference equations of shallow equations and analyze the results that are obtained with preconditioning and without preconditioning, respectively. It is shown that the computation speed is greatly improved after we use the preconditioning method. In addition, this preconditioning algorithm is simple and parallelizable. Therefore, the algorithm is potential in the applications of weather equations.

Key Words. Preconditioning, Shallow water equations, Sparse approximation inverse, GMRES iterative algorithm.

1. Introduction

Atmospheric motions are described by coupled nonlinear partial differential equations. The equations not only describe the dynamical processes such as fluid motion, but also the physical processes such as atmospheric radiation, chemistry, etc. (cf.[1]). Because few analytical solution exist, in order for solution to be obtained, the equation are usually discretized and solved numerically. The early primitive equation models, based on explicit time-difference schemes, were computationally very inefficient, requiring a large expenditure of computer time for a given level of accuracy. This resulted from the Courant-Friedrichs-Lewy (CFL) condition for the high frequency motion, which required the use of time step that were enough small. this results in an algebraic system of equations with a large and sparse matrix.

\[ Ax = b, \]

where \( A \in \mathbb{R}^{n \times n} \) is nonsingular, \( x \in \mathbb{R}^n \), and \( b \in \mathbb{R}^n \). The obtained linear system is aimed to be solved with as small computational effort and memory demand as possible. For really large problems \( (n > 500000) \), the only way to achieve this is to use an optimal, robust, preconditioned, iterative solution method. Below, the particular meaning of the terminology is explicitly stated (cf. [4]).

(i) Robustness means that the iterative solver converges independently of the parameters of the underlying problem (such as the Poisson number in elasticity problems and the viscosity in fluid dynamics).
(ii) For the iterative method to be optimal, its rate of convergence, i.e. the number of iterations required for the method to converge, must be independent of the size of $A$.

(iii) Furthermore, in order to handle large scale applications, the iterative solution method should work fast in terms of CPU-time. To achieve this, the iterative solution method must be numerically efficient (few arithmetic operations per unknown).

Generally speaking, there are two approaches to precondition solving system. One popular approach in applications involving PDEs is to design specialized algorithms that are optimal (or nearly so) for a narrow class of problems. It is a physics-based preconditioning method (cf., for example, [21]). This application-specific approach can be very successful, but it may require complete knowledge of the problem at hand, including the original (continuous) equations, the domain of integration, the boundary conditions, details of the discretization, and so forth. By making use of available information about the analysis, geometry, and physics of the problem, very effective preconditioners can be developed. As pointed out in [12], the method of diffusion synthetic acceleration (cf. [2],[18],[20]), which is widely used in the transport community, can be regarded as a physics-based preconditioner for the transport equation. Also, multi-grid preconditioners are often of this kind (see [16],[21] for a recent example).

Another preconditioning approach is based directly on linear algebra equations, which is referred as algebraic preconditioning method. It is well known that the rate of convergence of iterative methods for solving (1) is strongly influenced by the spectral properties of $A$. Preconditioning amounts to transforming the original system into one having the same solution but more favorable spectral properties, such as a clustering of the eigenvalues around 1. A preconditioner is a matrix that can be used to accomplish such a transformation. If $M$ is a nonsingular matrix which approximates $A^{-1}$ ($M \approx A^{-1}$), the transformed linear system

$$MAx = Mb.$$  

will have the same solution as system (1) but the convergence rate of iterative methods applied to (2) may be much higher. Problem (2) is preconditioned from the left, but right preconditioning is also possible. Preconditioning on the right leads to the transformed linear system

$$AMy = b.$$  

In order to construct efficient preconditioner $M$, preconditioning techniques based on spare approximate inverses have been vigorously developed in recent years (cf. [5]-[10],[13]-[15],[17],[19],[23]). The common idea underlying this class of algorithm is that a sparse matrix $M \approx A^{-1}$ is explicitly computed and used as a preconditioner for Krylov subspace methods (cf. [3]) for the solution of equations (1). The main advantage of this approach is that the preconditioning operation can easily be implemented in parallel, since it consists of matrix-vector products. Furthermore, the construction and application of preconditioners of the approximate inverse type tend to be immune from such numerical difficulties as pivot breakdowns and instability. Approximate inverse techniques rely on the assumption that for a given sparse matrix $A$, it is possible to find a sparse matrix $M$ which is a good approximation, in some sense, of $A^{-1}$. Since the inverse of a sparse matrix is usually dense, this means that for a given irreducible sparsity pattern, it is always possible to assign numerical value to the non-zeros in such a way that all entries of the inverse will be non-zero. Nevertheless, it is often the case that many of the entries in inverse
of a spare matrix are small in absolute value, thus making the approximation of $A^{-1}$ with a spare matrix possible. In this paper, we use the method of [6], [22] to compute a spare matrix $M \approx A^{-1}$ as the solution of the constrained minimization problem, the sparsity pattern is the banded matrix provided by [15]. Finally, we apply this preconditioning to shallow water problem, finding that it is very efficient.

The remainder of this paper is organized as follows. In Section 2, we construct the preconditioner based on the idea of spare approximate inverse techniques for linear systems. In Sections 3 we apply these precondition techniques to the linear system arising from the finite difference approximation of a shallow water problem, numerical results with and without preconditioning are also reported. A few concluding remarks are given in the last section.

2. Preconditioning based on spare approximate inverse

Spare approximate inverse methods have been proposed for preconditioning spare linear systems [11], [14]. Our aim is to construct an approximate inverse $M$ to $A$ and consider applying the iterative solver to the preconditioned system

$$AMy = b, \quad M \approx A^{-1}, \quad x = My.$$  

Set $t_s$, be the execution time per step without preconditioning, and $T_p$ the total execution time with preconditioning. Roughly speaking

$$T_p = t_1 + n_p \times (t_p + t_m),$$

where $T_1$ is the time spend initially to compute the approximate inverse $M$, $t_m$ the time necessary to compute $Mv$ for some vector $v$ and $n_p$ the number of iterations. In the same way, the total time without preconditioning might be expressed by

$$T_s = n_s \times t_s.$$  

The “art of preconditioning” consists now in finding a matrix $M$ which minimizes the ratio $T_p/T_s$.

From the expression for $T_p$ we can immediately deduce what kind of conditions $M$ must satisfy to be effective: (i) The crucial time which needs to be minimized is of course $t_m$, since it occurs at each iteration. (ii) Since $n_p$ depends heavily on the closeness of $M$ to $A^{-1}$. Hence, if the number of iterations $n_p$ is reduced to $\bar{n}_p$, the total execution time will be lowered to $t_1 + \bar{n}_p \times (t_s + t_m)$. However, $t_m$ might increase for a more sophisticated approximate inverse. When $M$ is a banded matrix, we compute $Mv$ extremely rapidly. Thus it seems natural to require that $M$ be a matrix with, say, $2p + 1$ diagonals, $p > 0$, so that we may keep $t_m$ very small relatively to the time spent for compute $Av$.

The closeness might be measures in some norm $\| \cdot \|$, so that we need to find an $M$ which minimizes

$$\|AM - I\|.$$  

In general, this problem is even harder than solving $Ax = b$. In order to reduce the computation of constructing $M$, we may specify a spare pattern of $M$, and the columns of $M$, denoted by $m_j$, can be computed independently and in parallel. Once such an $M$ has been constructed, we can solve the preconditioned linear system (4).

To specify the matrix $M$, write $\mathcal{N} = \{1, 2, \cdots, n\}$, let $\mathcal{S}$ be a given set of $(i, j)$ with $(i, j) \in \mathcal{N}$, i.e., a subset of $\mathcal{N} \times \mathcal{N}$, and $\mathcal{G}_s$ be the space of all $n \times n$ matrices that have entries in positions indexed by $\mathcal{S}$, and zeros outside $\mathcal{S}$. For each column index $j = 1, \cdots, n$, define its set of row indices $\mathcal{S}_j = \{i : (i, j) \in \mathcal{S}\}$ and let its vector space $\mathcal{G}_s$, contain all vectors that have entries in positions indexed by $\mathcal{S}_j$. Then the
approximate inverse $M$ is calculated by solving the least squares (LS) problem (cf. [6],[22])

$$\min_{M \in \mathcal{G}_s} \|AM - I\|_F^2 = \sum_{j=1}^n \min_{m_j \in \mathcal{V}_j} \|Am_j - e_j\|_2^2,$$

where $M = \{m_1, \cdots, m_n\}$ and $I = [e_1, \cdots, e_n]$. In theory, such an LS problem can be posed in any norm, but the Frobenius norm leads to an easier solution. The full problem of finding $M$ is reduced to $n$ standard LS problems.

The suitable specification of $\mathcal{S}$ will then become essential. For PDEs solved by domain-type discretization methods (mainly finite difference method in numerical weather prediction), $\mathcal{S}$ is usually chosen so that $M$ is of some spare structure, e.g., a banded matrix. Here we select the inverse of $A$ can be of the following specific sparsity pattern with $2p + 1 = 5$ (cf. [15])

$$M = \begin{pmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
* & * & * & * & * \\
* & * & * & * & * 
\end{pmatrix}.$$ 

Thus $\mathcal{S}$ will represent all non-zero positions in above spare structure. We now consider the solution of the least squares problem for finding the right preconditioner. Since matrix $M = [m_1, m_2, \cdots, m_n]$ is consisted of column vectors, for each column $j$, the least squares problem is to solve

$$\begin{pmatrix}
A_{1,j_1} & A_{1,j_2} & A_{1,j_3} \\
\vdots & \vdots & \vdots \\
A_{j_1,j_1} & A_{j_1,j_2} & A_{j_1,j_3} \\
A_{j_2,j_1} & A_{j_2,j_2} & A_{j_2,j_3} \\
A_{j_3,j_1} & A_{j_3,j_2} & A_{j_3,j_3}
\end{pmatrix} \begin{pmatrix}
M_{j_1,j} \\
M_{j_2,j} \\
M_{j_3,j}
\end{pmatrix} = \begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix},$$

where $j_2 = j_1 = j - 1, j_3 = j + 1$ for $j = 2, \ldots, n; j_1 = n, j_3 = 2$ for $j = 1$, and $j_1 = n - 1, j_3 = 1$ for $j = n$ due to the choice of $\mathcal{S}$ and the wrap-around nature of $M$. We solve the least squares problem by the following algorithm.

**Algorithm 1** (Approximate inversion for $x = My$ (cf. [15]))

(a) For each column $i = 1, \cdots, n$, set $i_1, i_2, i_3$ and form the $n \times 3$ sub-matrix $W$ as above. Set $c = e_i$, the $i$th unit vector. Then solve an $n \times 3$ LS system $Wc = e$ by the QR method. Set $p_i = c$ (the $i$th column of $M$).

(b) Evaluate $x = My$.

3. Application of the spare approximate inverse techniques to a shallow water problem

In this section we will detail how a simple preconditioner is incorporated into the iterative solving procedure as well as illustrate the detail of spare approximate inverse techniques on the one-dimensional shallow water equations, and in this we will show results from the shallow water model.
3.1. A shallow water model.

For our demonstration of spare approximate inverse approach we have chosen a simple model problem that solves the one-dimensional shallow-water equations in flux form with no bottom topography (cf. [24]).

We have added additional forcing terms that lead to a closed-form analytical solution. The equations are

\[
\begin{align*}
\frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} &= f_h, \\
\frac{\partial (uh)}{\partial t} + \frac{\partial (uh^2)}{\partial x} &= -gh\frac{\partial h}{\partial x} + uf_h + hf_u,
\end{align*}
\]

where \( h \) is the height of the fluid, \( u \) is the velocity of the fluid, \( g \) is the acceleration due to gravity, and

\[
\begin{align*}
f_h &= (a_h \omega - a_u h_0 k) \sin(kx + \omega t) + a_h a_u \sin(2kx + \omega t), \\
f_u &= (a_h g k - a_u \omega) \sin(kx + \omega t) - 0.5a_u^2 k \sin(2kx + \omega t),
\end{align*}
\]

and equation (5) have the analytic solutions

\[
u = u_a \cos(kx + \omega t), \quad h = h_0 - a_u \cos(kx + \omega t),
\]

where \( h_0 \) is the average height of the initial wave, \( \omega \) is the frequency of the wave, and \( k \) is the wave-number. Note, the analytical solution are used as initial conditions (set \( t = 0 \) ) and periodic boundary conditions are employed in this simple model.

3.2. Semi-implicity difference scheme and its solution.

In order to numerically solve differential equations (5), we obtain the following difference equations

\[
\begin{align*}
\frac{h_i^{n+1} - h_i^n}{\Delta t} + \frac{(uh)_{i+1}^{n+1} - (uh)_{i-1}^{n+1}}{2\Delta x} &= f_h^{n+1}, \\
\frac{(uh)^{n+1}_i - (uh)^n_i}{\Delta t} + \frac{(u^2 h)_{i+1}^{n+1} - (u^2 h)_{i-1}^{n+1}}{2\Delta x} &= -gh_i^{n+1} h_{i+1}^{n+1} - h_{i-1}^{n+1} + u^n h_i^{n+1}.
\end{align*}
\]

We have chosen the following parameters for our simple model. \( g = 1.0\text{m}s^{-2} \), \( h_0 = 1.0m \), \( a_h = 0.01m \), \( a_u = 0.01m\text{s}^{-1} \), \( k = 2m^{-1} \), and \( \omega = 0.0s^{-1} \), \( x \in [0, 2\pi] \), \( t \in [0, 2\pi] \).

Evidently, (7) is a linear system, which can be written as \( Ax = b \), for the unknown \( h_i^{n+1} \) and \( (uh)^n_i \). We will use the following restart \( GMRES \) method to solve it.

**Algorithm 2:** \( GMRES(m) \) (restart \( GMRES \) iterative method (cf.[25]) )

(i) Start: Choose \( x_0 \) and compute \( r_0 = b - Ax_0 \) and \( v_1 = r_0 / \| r_0 \| \).

(ii) Iterate: for \( j = 1, 2, \ldots, m \) do:

\[
\begin{align*}
h_{i,j} &= (Av_j, v_i), \quad i = 1, 2, \ldots, j, \\
\hat{v}_{j+1} &= Av_j - \sum_{i=1}^{j} h_{i,j} v_i, \\
\hat{h}_{j+1,j} &= \| \hat{v}_{j+1} \|, \\
v_{j+1} &= \hat{v}_{j+1} / \hat{h}_{j+1,j}.
\end{align*}
\]

(iii) Form the approximate solution:

\[
x_m = x_0 + V_m y_m \text{ minimizes } \| b - \hat{H} m y \|, \quad y \in R^m.
\]

(iv) Restart:
Compute \( r_m = b - Ax_m \); if satisfied then stop
else compute \( x_0 := x_m \), \( v_1 := r_m/\|r_m\| \) and go to (ii)

Following [26], we ran the simulations at successively finer resolution while maintaining a fixed Courant number \( a_1(\Delta t/\Delta x) \) with each successive \( \Delta t \) and \( \Delta x \) being reduced by a factor of 2. For each simulation, we calculate the truncation error

\[
\text{error}(N_t, N_x) = \left( \sum_{j=1}^{N_t} \sum_{i=1}^{N_x} (\psi(t_j, x_i) - \psi^j_i)^2 \right)^{1/2},
\]

where \( N_t = T/\Delta t \), \( N_x = L/\Delta x \) with \( T = 2\pi(s) \) and \( L = 2\pi(m) \), and \( \psi(x_i, t_j), \psi^j_i \) are the analytical and numerical solution, respectively, at the point \((x_i, t_j)\).

### 3.3. Preconditioning and comparison of results.

In this section, we shall explain how to construct preconditioner, and compare the results before and after preconditioned.

We rewrite semi-implicit difference equations (7)

\[
\begin{aligned}
  h_i^{n+1} + kP_{i+1}^{n+1} - kP_{i-1}^{n+1} &= h_i^n + \Delta t f_{hi}^{n+1}, \\
  P_i^{n+1} + gh_i^{n+1}h_i^{n+1} - gh_i^{n+1}h_i^{n-1} &= d_i^n,
\end{aligned}
\]

where

\[
P_i^n = (uh)_i^n, \quad d_i^n = -k(u_{i+1}^nP_{i+1}^n - u_{i-1}^nP_{i-1}^n) + \Delta t u_i^n f_{hi}^{n+1},
\]

\[
k = \Delta t/(2\Delta x).
\]

We shall firstly solve \( h_i^{n+1} \) and \( P_i^{n+1} \), then obtain \( u_i^{n+1} \) by \( P = uh \). Evidently, the coefficient matrix of linear equations (9) is a non-symmetry matrix. We use 9 non-zero entries bandwidth as spare pattern \( M \), then the non-zero entries of pre-conditioner matrix can be decided using algorithm 1. We solve the preconditioned linear equations \( AMy = b \) using algorithm 2 for \( y \); here iteration matrix is \( AM \). Further, we get \( x \) by \( x = My \). In this paper, we assume that the restart parameter \( m = 10 \).

In order to demonstrate the effectiveness of spare approximate inverse method, we firstly show that our algorithm for solving equations (7) is stable. In Figure 1, the larger the number of halvings of the grid and time increase is, the smaller the truncation error becomes. Here we use the logarithm of truncation error and halvings of the grid and time describe the relation that the error was decreased with net are fined. This confirm that our numerical method is stable. In Figure 2, we have a comparison about average number of Krylov iterations according to iterative convergence error with preconditioning and without preconditioning. Evidently, the increment of average number is small with iterative convergence error decrease after linear system is preconditioned.

In TABLE 1, there are the comparison about iterative number and total CPU time with the increments of number of halvings of the grid and time. The total CPU time saved about 47 percent after preconditioned. If time grids are fixed and spatial grids are increased in TABLE 2, we find that total CPU time was saved more, specially, when \( N_x = 1000 \), 95 percent CPU time was saved. Because we use banded matrix as spare pattern, the changes of CPU time and iterative number with \( 2p+1 \) are described in TABLE 3. On the one hand, the larger \( 2p+1 \), the closer preconditioner \( M \) to \( A^{-1} \), therefore, the fewer Krylov iterative number. On the other hand, constructing a preconditioner with large \( 2p+1 \) need more time. So we must take a suitable \( 2p+1 \) such that total CPU time is minimal. From TABLE 3,
Total CPU time is minimal when $2p+1 = 9$ under $N_x = 100$ and $N_t = 20$ case, and more experimental results have confirmed the conclusion, such as $N_x = 50, N_t = 10$; $N_x = 150, N_t = 30$; $N_x = 250, N_t = 50$; $N_x = 300, N_t = 60$; · · ·, etc.

Figure 1. The dependence of the measure of the truncation error on the number of halvings of the grid and time increments.

Figure 2. Average number of Krylov iterations as a function the logarithm of the iterative convergence error for shallow problem simulations. Star “∗” denote the result without preconditioning, and circle “◦” the result with preconditioning.

4. Concluding remarks

Preconditioning based on spare approximate inverse can lead to very successful parallel or sequential iterative procedures for solving general spare linear system. In this paper we examined a class of preconditioning methods for spare linear system arising from finite difference discretization of shallow water problem, the results have shown that the computation speed is greatly improved after we use the preconditioning method and this preconditioning algorithm is simple and parallelizable.
Table 1. Comparison of total iterative number and the total CPU time before and after preconditioned

<table>
<thead>
<tr>
<th>$N_x \times N_t$</th>
<th>no-preconditioned</th>
<th>preconditioned</th>
</tr>
</thead>
<tbody>
<tr>
<td>$60 \times 10$</td>
<td>320</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>0.19</td>
<td>0.22</td>
</tr>
<tr>
<td>$120 \times 20$</td>
<td>640</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>1.64</td>
<td>1.35</td>
</tr>
<tr>
<td>$240 \times 40$</td>
<td>1280</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>19.13</td>
<td>10.44</td>
</tr>
<tr>
<td>$480 \times 80$</td>
<td>2560</td>
<td>320</td>
</tr>
<tr>
<td></td>
<td>183.92</td>
<td>96.82</td>
</tr>
<tr>
<td>$960 \times 160$</td>
<td>5120</td>
<td>640</td>
</tr>
<tr>
<td></td>
<td>1902.72</td>
<td>986.38</td>
</tr>
<tr>
<td>$1000 \times 170$</td>
<td>5270</td>
<td>680</td>
</tr>
<tr>
<td></td>
<td>2175.8</td>
<td>1146.69</td>
</tr>
</tbody>
</table>

Table 2. Comparison of Krylov iterative numbers at average time-step and the total CPU time before and after preconditioned

<table>
<thead>
<tr>
<th>$N_x(N_t=30)$</th>
<th>no-preconditioned</th>
<th>preconditioned</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>aver. num.</td>
<td>cpu time</td>
</tr>
<tr>
<td>$60$</td>
<td>10</td>
<td>0.18</td>
</tr>
<tr>
<td>$120$</td>
<td>19</td>
<td>1.46</td>
</tr>
<tr>
<td>$240$</td>
<td>51</td>
<td>23.7</td>
</tr>
<tr>
<td>$480$</td>
<td>181</td>
<td>395.5</td>
</tr>
<tr>
<td>$960$</td>
<td>746</td>
<td>8404</td>
</tr>
<tr>
<td>$1000$</td>
<td>812</td>
<td>9987.25</td>
</tr>
</tbody>
</table>

Table 3. Total CPU time and iterative number per time step with $p = 1, 2, 3, 4, 5, 6, 7, 8$ when $N_x = 100, N_t = 20$

<table>
<thead>
<tr>
<th>$2p+1$</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>0.71</td>
<td>0.76</td>
<td>0.57</td>
<td>0.5</td>
<td>0.57</td>
<td>0.66</td>
<td>0.67</td>
<td>0.73</td>
</tr>
<tr>
<td>iterat. num.</td>
<td>17</td>
<td>16</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Acknowledgments

We would like to thank professor Bin Wang for his interest and help.

References

APPLICATION OF SAI METHOD TO SHALLOW WATER PROBLEM


1. State Key Laboratory of Severe Weather (LaSW), Chinese Academy of Meteorological Sciences, Beijing 100081, China
E-mail: wangggh@cams.cma.gov.cn, shenxs@cams.cma.gov.cn, chenff@cams.cma.gov.cn

2. State Key Laboratory of Numerical Modeling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG), Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing 100029, China
E-mail: jzhz@lasg.iap.ac.cn