Parallel Deflated CG Method to Simulate Groundwater Flow in a Layered Grid

Raju Ram

August 24, 2017
1. Problem Description
   - Groundwater Flow

2. Proposed solution

3. Results

4. Conclusions and Recommendations
Hydrology Background

Figure: What is under the earth’s surface

- About 98% of the earth’s available fresh water is present beneath the earth’s surface in soil pore spaces, called groundwater.
- Hydraulic head calculates measurement of liquid pressure is groundwater.
- Darcy’s law defines the movement of water in the subsurface.
MODFLOW software developed by U.S Geological Survey is used to simulate groundwater flow.

- Cell centered finite volume discretization: Domain is divided into rectangular boxes called cells.
- Geometries of underlying countries are not rectangular, MODFLOW computes head only at active cells (red).
Groundwater Flow Equation

\[
\frac{\partial}{\partial x} \left( K_{xx} \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left( K_{yy} \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left( K_{zz} \frac{\partial h}{\partial z} \right) + W = S_s \frac{\partial h}{\partial t}
\]

where,

\( K_{xx}, K_{yy} \) and \( K_{zz} \) are hydraulic conductivities along the x, y, and z coordinate axes \((LT^{-1})\).

\( W \) is volumetric flux per unit volume representing sources and sinks of water \((T^{-1})\).

\( S_s \) is specific storage of porous material \((L^{-1})\).

\( h \) is Hydraulic head \((L)\).
Finite Volume Discretization

- Flow from cell \((i, j - 1, k)\) into cell \((i, j, k)\):
  \[
  q_{i,j-1/2} = CC_{i,j-1/2}(h_{i,j-1} - h_{i,j})
  \]
- Continuity equation:
  \[
  \sum_{n=1}^{N} q_{i,j,n} = S_s \Delta V \frac{\Delta h}{\Delta t}
  \]
- For \(N = 6\), above becomes
  \[
  q_{left} + q_{right} + q_{up} + q_{down} + q_{top} + q_{bottom} = S_s \Delta V \frac{\Delta h}{\Delta t}
  \]
System of Equations

\[ CV_{(i,j,k-\frac{1}{2})} h(i,j,k-1) + CR_{(i-\frac{1}{2},j,k)} h(i-1,j,k) + CC_{(i,j-\frac{1}{2},k)} h(i,j-1,k) + H_c h(i,j,k) + CC_{(i,j+\frac{1}{2},k)} h(i,j+1,k) + CR_{(i+\frac{1}{2},j,k)} h(i+1,j,k) + CV_{(i,j,k+\frac{1}{2})} h(i,j,k+1) = RHS_{(i,j,k)} \]

- System of equations of form \( Au = f \).
- \( H_c \) depends on \( h(i,j,k) \): system of equations becomes non-linear.
- Picard iteration is used to make the system linear.
Simulation Flowchart

Figure: Grounder water simulation flowchart
Problem Description

Proposed solution

Results

Conclusions and Recommendations

Block Jacobi Preconditioner $M$

- Preconditioned Conjugate Gradient (PCG) in Parallel Krylov Solver (PKS) solves:

$$M^{-1}Au = M^{-1}f.$$

- For 2 subdomains:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}.$$

- $A_{11}u_1 = f_1 - A_{12}u_2$

- $A_{22}u_2 = f_2 - A_{21}u_1$

**Figure**: Partitioning of grid using 2 processors in MODFLOW
MODFLOW: 3D
Groundwater flow using 7 layers.
Numerical experiments for Steady state (SS) model, Stress loop and time loop is fixed.
Consider outer Picard iteration and inner PCG iteration.
Vary cell size: 250 m, 100 m, 50 m.
Problem statement

- PCG iterations increase with increasing number of subdomains in PKS, due to decoupling in global information.
- Goal of this masters project is to gain wall clock time by reducing the iteration increase.
So far we covered ...

- Hydrological background behind the problem.
- Finite Volume Discretization.
- Preconditioner.
- Problem statement.
Summary: Problem Description

- So far we covered ...
  - Hydrological background behind the problem.
  - Finite Volume Discretization.
  - Preconditioner.
  - Problem statement.

- Next ...
  - Deflation Preconditioner
Our approach: Deflation

- Eigenvectors with small eigenvalues hampers the PCG convergence.

(a) Constant deflation vectors

(b) Linear deflation vectors

- We approximate the eigenvectors with constant deflation vectors (CDPCG) and linear deflation vectors (LDPCG).
- Columns of deflation matrix $Z$ are deflation vectors.
Basic Idea Behind Deflation Preconditioner

a) Used to remove influence of \( k \) small eigenvalues. Condition number reduces to \( \frac{\lambda_n}{\lambda_{k+1}} \) from \( \frac{\lambda_n}{\lambda_1} \).
Basic Idea Behind Deflation Preconditioner

a) Used to remove influence of $k$ small eigenvalues. Condition number reduces to $\frac{\lambda_n}{\lambda_{k+1}}$ from $\frac{\lambda_n}{\lambda_1}$.

b) We define projector

\[ P_1 = I - AZE^{-1}Z^T, \quad P_2 = I - ZE^{-1}Z^TA \]
Basic Idea Behind Deflation Preconditioner

a) Used to remove influence of \( k \) small eigenvalues. Condition number reduces to \( \frac{\lambda_n}{\lambda_{k+1}} \) from \( \frac{\lambda_n}{\lambda_1} \).

b) We define projector
   \[
   P_1 = I - AZE^{-1}Z^T, \quad P_2 = I - ZE^{-1}Z^T A
   \]

c) Solve for deflated system: \( P_1 A\tilde{u} = P_1 f \).
Basic Idea Behind Deflation Preconditioner

a) Used to remove influence of $k$ small eigenvalues. Condition number reduces to $\frac{\lambda_n}{\lambda_{k+1}}$ from $\frac{\lambda_n}{\lambda_1}$.

b) We define projector

$$P_1 = I - AZE^{-1}Z^T, \quad P_2 = I - ZE^{-1}Z^T A$$

c) Solve for deflated system: $P_1A\tilde{u} = P_1f$.

d) $u = (I - P_2)u + P_2u$, 
Basic Idea Behind Deflation Preconditioner

a) Used to remove influence of $k$ small eigenvalues. Condition number reduces to $\frac{\lambda_n}{\lambda_{k+1}}$ from $\frac{\lambda_n}{\lambda_1}$.

b) We define projector
$$ P_1 = I - AZE^{-1}Z^T, \quad P_2 = I - ZE^{-1}Z^T A $$

c) Solve for deflated system: $P_1 A \tilde{u} = P_1 f$.

d) $u = (I - P_2) u + P_2 u$,

e) $(I - P_2) u$ in d) becomes $ZE^{-1}Z^T f$. 
Basic Idea Behind Deflation Preconditioner

a) Used to remove influence of $k$ small eigenvalues. Condition number reduces to $\frac{\lambda_n}{\lambda_{k+1}}$ from $\frac{\lambda_n}{\lambda_1}$.

b) We define projector

$$P_1 = I - AZE^{-1}Z^T, \quad P_2 = I - ZE^{-1}Z^T A$$


c) Solve for deflated system: $P_1A\tilde{u} = P_1f$.

d) $u = (I - P_2)u + P_2 u$,

e) $(I - P_2)u$ in d) becomes $ZE^{-1}Z^T f$.

f) $P_2 u = P_2 \tilde{u}$, substitute $\tilde{u}$ from c) in d) to obtain $u$. 
What to add in PCG to make it DPCG?

- **Deflation pre processing phase: residual update**

  solve $Eq_1 = Z^T r^{(0)}$, $E = Z^T AZ$, sparse LU to decompose $E$

  \[ \tilde{r}^{(0)} = r^{(0)} - AZq_1 \]

- **Deflation runtime phase: DPCG mat-vec prod:**

  \[ Ax = r^{(0)} \xrightarrow{\text{Deflation}} P_1 A\tilde{x} = P_1 r^{(0)} \]

  solve $Eq_3^{(k)} = Z^T v^{(k)}$

  \[ P_1 v^{(k)} = v^{(k)} - AZq_3^{(k)} \]

- **Deflation post processing phase:**

  Solve for $q_2 : Eq_2 = Z^T A\tilde{x}$

  Solution correction: $u = Z(q_1 - q_2) + \tilde{x} + u^{(0)}$
Deflated PCG Algorithm

Algorithm 1 Deflated PCG Algorithm

1: \textbf{procedure} DPCG\((A, f, u^{(0)}, tol, k_{\text{max}}, M, Z)\)
2: \(r^{(0)} = f - Au^{(0)}, \quad k=1\) \hspace{1cm} \text{Once} \hspace{1cm} \text{\Rightarrow Initialization}
3: \textbf{if } (\text{deflation}) \text{ then} \hspace{2cm} \text{\Rightarrow Deflation pre-processing phase}
4: \quad \tilde{u}^{(0)} = u^{(0)}
5: \quad u^{(0)} = 0
6: \quad \text{Decompose } Z^T A Z (d \times LC, GC) = \tilde{L} \tilde{U} \hspace{1cm} \Rightarrow d=3 \text{ for NHI model in LDPCG}
7: \quad \textbf{solve} \quad \tilde{L} \tilde{q}_1 = Z^T r^{(0)} (GC); \quad \tilde{U} q_1 = \tilde{q}_1
8: \quad r^{(0)} = r^{(0)} - AZ q_1
9: \textbf{end if}
10: \textbf{while } (k < k_{\text{max}} \textbf{ and } ||r^{(k-1)}|| \textbf{ > tol}) \textbf{ do}
11: \quad z^{(k-1)} = M^{-1} r^{(k-1)} \hspace{1cm} \Rightarrow \text{Preconditioning with Additive Schwarz}
12: \quad \textbf{if } k = 1 \textbf{ then}
13: \quad \quad \quad \beta^{(1)} = Z^{(1)} x^{(k-1)} \quad \beta^{(2)} = Z^{(2)} x^{(k-2)}
14: \quad \quad \quad p^{(1)} = z^{(k-1)} + \beta^{(1)} b^{(k-1)} \hspace{1cm} \Rightarrow \text{Search direction}
15: \quad \textbf{else}
16: \quad \quad \quad \beta^{(k)} = (r^{(k-1)})^T x^{(k-1)}
17: \quad \quad \quad p^{(k)} = z^{(k-1)} + \beta^{(k)} b^{(k-1)}
18: \quad \textbf{end if}
19: \quad v^{(k)} = A p^{(k)} \hspace{1cm} \text{ITER1 times}
20: \quad \textbf{if } (\text{deflation}) \text{ then} \hspace{1cm} \Rightarrow \text{Deflation run time phase}
21: \quad \quad \textbf{solve} \quad \tilde{L} q_3^{(k)} = Z^T v^{(k)} (GC); \quad \tilde{U} q_3^{(k)} = q_3^{(k)}
22: \quad \quad v^{(k)} = v^{(k)} - AZ q_3^{(k)}
23: \quad \textbf{end if}
24: \quad a_k = \frac{r^{(k-1)} x^{(k-1)}}{p^{(k)} x^{(k)}}
25: \quad u^{(k)} = u^{(k-1)} + \alpha_k p_k \hspace{1cm} \Rightarrow \text{Iterate update}
26: \quad r^{(k)} = r^{(k-1)} - a_k v^{(k)} \hspace{1cm} \Rightarrow \text{Residual update}
27: \quad k = k + 1
28: \textbf{end while}
29: \quad k = k - 1 \hspace{1cm} \text{Once}
30: \quad \textbf{if } (\text{deflation}) \text{ then} \hspace{1cm} \Rightarrow \text{Deflation post-processing phase}
31: \quad \quad \textbf{solve} \quad \tilde{L} q_2^{(k)} = Z^T A u^{(k)} (LC, GC); \quad \tilde{U} q_2^{(k)} = \tilde{q}_2
32: \quad \quad u^{(k)} = u^{(k)} + \tilde{u}^{(0)} + Z (q_1 - q_2)
33: \quad \textbf{end if}
34: \quad \textbf{return} \quad u^{(k)} \hspace{1cm} \Rightarrow \text{The converged solution}
35: \textbf{end procedure}
Choosing Deflation Vectors

- Extraction of one subdomain from the Netherlands domain.
- The brown layer denote ghost layer cells.

**Figure:** Deflation vectors: a) in CDPCG and a)-d) in LDPCG
We discussed Deflation algorithm.
Summary: Proposed Solution

- We discussed Deflation algorithm.
- Choosing deflation vectors in NHI Steady State (SS) model.
Summary: Proposed Solution

- We discussed Deflation algorithm.
- Choosing deflation vectors in NHI Steady State (SS) model.
- What next?: Numerical results for various models.
  - cell size: 250 m, two layer iMOD unit case.
  - cell size: 100 m, seven layer NHI SS model.
  - cell size: 50 m, seven layer NHI SS model.
250 m, Two Layer iMOD Unit Case Iterations

Iterations increase with increasing subdomains

- no-deflation
- constant-deflation
- linear-xyz-deflation

Number of total iterations in CG vs Number of subdomains
NHI 100m Cellsize: Iteration Improvement

Variation of iterations with increasing subdomains in NHI SS 100m model

- no-deflation
- constant-deflation
- linear-xyz-deflation

Number of total iterations in CG

Number of subdomains

Results
NHI 50m Cellsize: Iteration Improvement

Variation of iterations with increasing subdomains in NHI SS 50m model

- PCG
- CDPCG
- LDPCG
NHI 50m Cellsize: Inner Iteration in Each Picard Iteration

Variation of inner iterations with Picard iteration in NHI SS 50m model
Overview of Results: 100 Subdomains

<table>
<thead>
<tr>
<th>Cell size</th>
<th>PCG</th>
<th>CDPCG</th>
<th>LDPCG</th>
<th>LDPCG SU vs CDPCG SU</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>2527</td>
<td>1768</td>
<td>1.43</td>
<td>1.18</td>
</tr>
<tr>
<td>100</td>
<td>10775</td>
<td>5209</td>
<td>2.07</td>
<td>1.57</td>
</tr>
<tr>
<td>50</td>
<td>20927</td>
<td>10244</td>
<td>2.04</td>
<td>2.06</td>
</tr>
</tbody>
</table>

Table: Speed up in iterations (Iters) for NHI SS model with 100 subdomains, SU stands for speed up.

- Performance of LDPCG improves for higher resolution models.
Improvement in Wall Clock Time: NHI SS 100m

NHI 100m SS: Factor improvement in wall clock time

Setup time: LDPCG

Huge decrease in iterations

CDPCG
LDPCG
Speed up in NHI SS 100m: 4 subdomains as a reference

![Graph showing speedup comparison](image-url)
Challenges

- LDPCG method (especially the construction of $E$) is difficult to implement.
- Load imbalance issue due to active cell of ghost layer arises, even after using Recursive Coordinate Bisection (RCB) domain decomposition.
Deflation preconditioner (using linear deflation vectors) has potential to achieve speed up in a wall clock time by factor 4.
Deflation preconditioner (using linear deflation vectors) has potential to achieve speed up in a wall clock time by factor 4.

The wall clock improvement is obtained due to huge decrease in iterations.
Conclusions

- Deflation preconditioner (using linear deflation vectors) has potential to achieve speed up in a wall clock time by factor 4.
- The wall clock improvement is obtained due to huge decrease in iterations.
- Linear deflation vectors seems to be the optimal choice in the deflation preconditioner.
Investigate the serial solver convergence: by changing the maximum number of inner iterations, checking accuracy of ILU(0) subdomain solve.
Investigate the serial solver convergence: by changing the maximum number of inner iterations, checking accuracy of ILU(0) subdomain solve.

Reduce the local communication in constructing $AZ$ with linear deflation vectors.
Recommendations

- Investigate the serial solver convergence: by changing the maximum number of inner iterations, checking accuracy of ILU(0) subdomain solve.
- Reduce the local communication in constructing AZ with linear deflation vectors.
- Investigate the load imbalance in PCG and deflated PCG.
Recommendations

- Investigate the serial solver convergence: by changing the maximum number of inner iterations, checking accuracy of ILU(0) subdomain solve.
- Reduce the local communication in constructing AZ with linear deflation vectors.
- Investigate the load imbalance in PCG and deflated PCG.
- Check Deflation performance in NHI transient simulation.
Recommendations

- Investigate the serial solver convergence: by changing the maximum number of inner iterations, checking accuracy of ILU(0) subdomain solve.
- Reduce the local communication in constructing $AZ$ with linear deflation vectors.
- Investigate the load imbalance in PCG and deflated PCG.
- Check Deflation performance in NHI transient simulation.
- Implement deflation in other Deltaras packages such as SEAWAT (used for fresh salt groundwater computation).
References

- Jarno Verkaik, First Applications of the New Parallel Krylov Solver for MODFLOW on a National and Global Scale.
- PKS Workshop, iMOD Delft software days (DSD), 14 June 2017, Deltares
Questions/Feedback ?