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

Raju Ram

August 24, 2017

Deltares Enabling Delta Life | COSSE | TU Delft Delft University of Technology

Groundwater Management | Masters Program | Numerical Analysis, Applied Mathematics

Agenda

1. Problem Description
   - Groundwater Flow
2. Proposed solution
3. Results
4. Conclusions and Recommendations

Hydrology Background

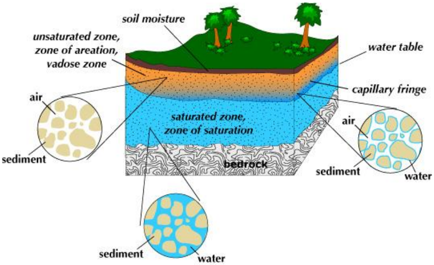
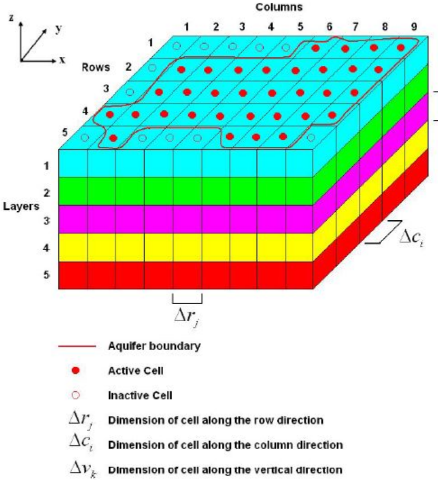


Figure: What is under the earth's surface

- 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

- 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):

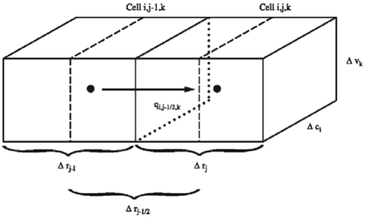
$$q_{(i,j-\frac{1}{2})} = CC_{(i,j-\frac{1}{2})}(h_{i,j-1} - h_{i,j})$$

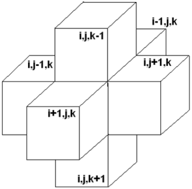
- Continuity equation:

$$\sum_{n=1}^{N} q_{i,j,n} = S_s \Delta V \frac{\Delta h}{\Delta t}$$

- For $N = 6$, above becomes

$$q_{left} + q_{right} + q_{up} + q_{down} + q_{top} + q_{bottom} = S_s \Delta V \frac{\Delta h}{\Delta t}$$

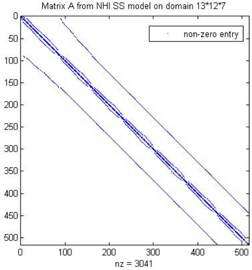




System of Equations

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

- System of equations of form $A\underline{u} = \underline{f}$.
- 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

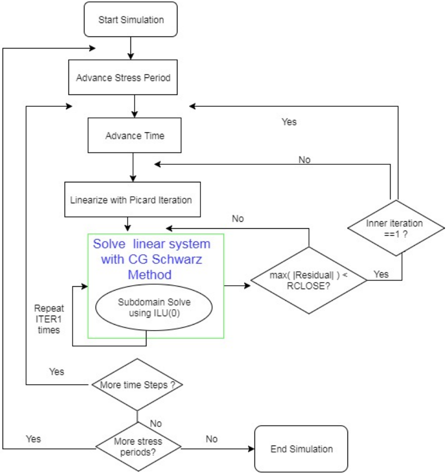


Figure: Grounder water simulation flowchart

Block Jacobi Preconditioner M

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

$$M^{-1}A\underline{u} = M^{-1}\underline{f}.$$

- For 2 subdomains:

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

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

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

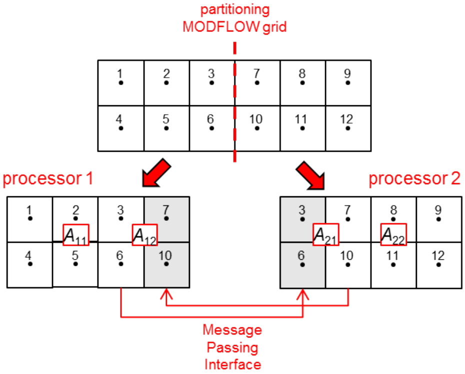


Figure: Partitioning of grid using 2 processors in MODFLOW

Nederlands Hydrologisch Instrumentarium (NHI)

- 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.

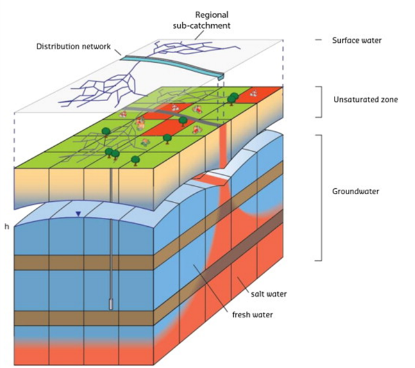
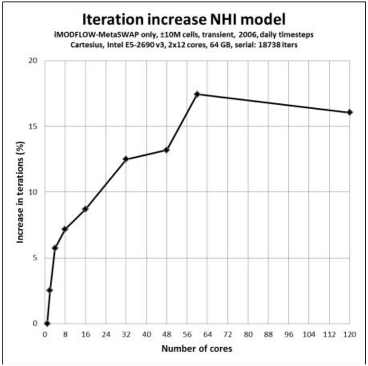


Fig. 1.
The water domains covered by the five hydrological models in NHI.

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.



Summary: Problem Description

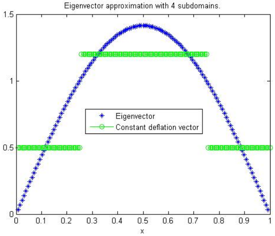
- So far we covered ...
  - Hydrological background behind the problem.
  - Finite Volume Discretization.
  - Preconditioner.
  - Problem statement.

Summary: Problem Description

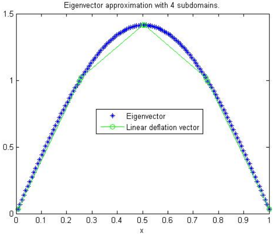
- So far we covered ...
  - Hydrological background behind the problem.
  - Finite Volume Discretization.
  - Preconditioner.
  - Problem statement.
- Next ...
  - Deflation Preconditioner

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: $P_1 A\tilde{u} = P_1 f$.

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

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

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: $P_1 A\tilde{u} = P_1 f$.

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

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

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

$$\text{solve } Eq_1 = Z^T r^{(0)}, E = Z^T AZ, \text{ sparse LU to decompose } E$$
$$\tilde{r}^{(0)} = r^{(0)} - AZq_1$$

- Deflation runtime phase: DPCG mat-vec prod:

$$Ax = r^{(0)} \xrightarrow{Deflation} P_1 A\tilde{x} = P_1 r^{(0)}$$
$$\text{solve } Eq_3^{(k)} = Z^T v^{(k)}$$
$$P_1 v^{(k)} = v^{(k)} - AZq_3^{(k)}$$

- Deflation post processing phase:

$$\text{Solve for } q_2 : Eq_2 = Z^T A\tilde{x}$$
$$\text{Solution correction: } u = Z(q_1 - q_2) + \tilde{x} + u^{(0)}$$

Deflated PCG Algorithm

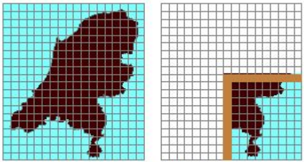
Algorithm 1 Deflated PCG Algorithm

procedure DPCG($A, f, u^{(0)}, tol, k_{max}, M, Z$)
  $r^{(0)} = f - Au^{(0)}$, k=1   Once   ▷ Initialization
  if (deflation) **then**   ▷ Deflation pre-processing phase
    $\tilde{u}^{(0)} = u^{(0)}$
    $u^{(0)} = 0$
    Decompose $Z^T AZ$ ($d \times$ LC, GC) $= \tilde{L}\tilde{U}$ ▷ d=3 for NHI model in LDPCG
    solve $\tilde{L}\tilde{q}_1 = Z^T r^{(0)}$ (GC); $\tilde{U}q_1 = \tilde{q}_1$
    $r^{(0)} = r^{(0)} - AZq_1$
  end if
  while ($k < k_{max}$ **and** $\|r^{(k-1)}\| > tol$) **do**
    $z^{(k-1)} = M^{-1}r^{(k-1)}$ ▷ Preconditioning with Additive Schwarz
    if $k = 1$ **then**
      $p^{(1)} = z^{(0)}$
    else
      $\beta_k = \frac{(r^{(k-1)})^T z^{(k-1)}}{(r^{(k-2)})^T z^{(k-2)}}$

      $p^{(k)} = z^{(k-1)} + \beta_k p^{(k-1)}$ ▷ Search direction
    end if
    $v^{(k)} = Ap^{(k)}$ ITER1 times
    if (deflation) **then**   ▷ Deflation run time phase
      solve $\tilde{L}\tilde{q}_3^{(k)} = Z^T v^{(k)}$ (GC); $\tilde{U}q_3^{(k)} = \tilde{q}_3^{(k)}$
      $v^{(k)} = v^{(k)} - AZq_3^{(k)}$
    end if
    $\alpha_k = \frac{(r^{(k-1)})^T z^{(k-1)}}{(p^{(k)})^T v^{(k)}}$

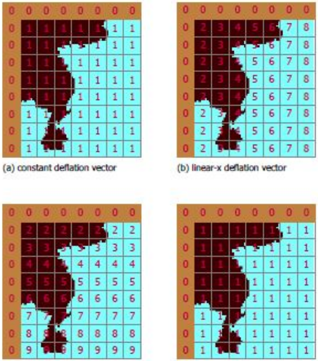
    $u^{(k)} = u^{(k-1)} + \alpha_k p_k$ ▷ Iterate update
    $r^{(k)} = r^{(k-1)} - \alpha_k v^{(k)}$ ▷ Residual update
    $k = k + 1$
  end while   Once
  k = k -1
  if (deflation) **then**   ▷ Deflation post-processing phase
    solve $\tilde{L}\tilde{q}_2 = Z^T Au^{(k)}$ (LC, GC); $\tilde{U}q_2 = \tilde{q}_2$
    $u^{(k)} = u^{(k)} + \tilde{u}^{(0)} + Z(q_1 - q_2)$
  end if
  return $u^{(k)}$ ▷ The converged solution
end procedure

Choosing Deflation Vectors



(a) Layer L_1 in the domain of Netherlands (b) One subdomain from layer L_1 in a)

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



(a) constant deflation vector (b) linear-x deflation vector (c) linear-y deflation vector (d) linear-z deflation vector

Figure: Deflation vectors: a) in CDPCG and a)-d) in LDPCG

Summary: Proposed Solution

- We discussed Deflation algorithm.

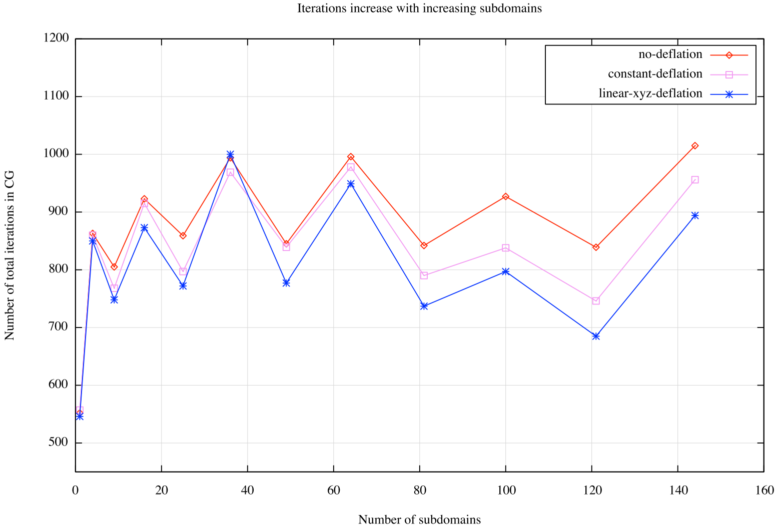
Summary: Proposed Solution

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

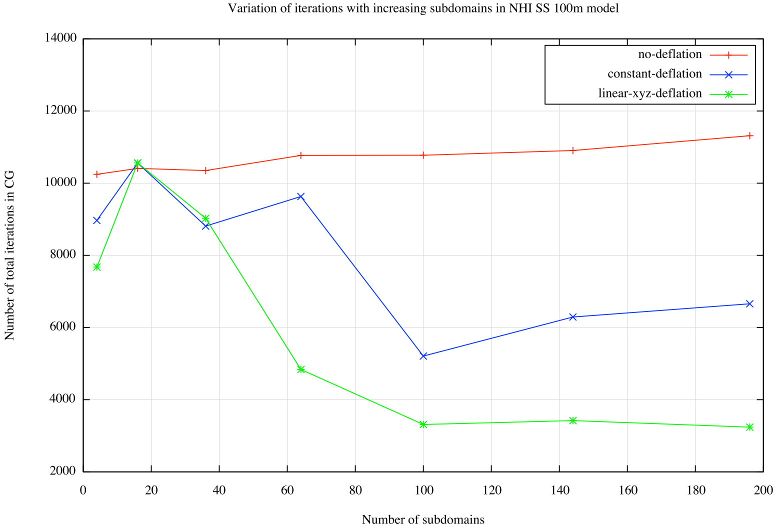
Summary: Proposed Solution

- 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.

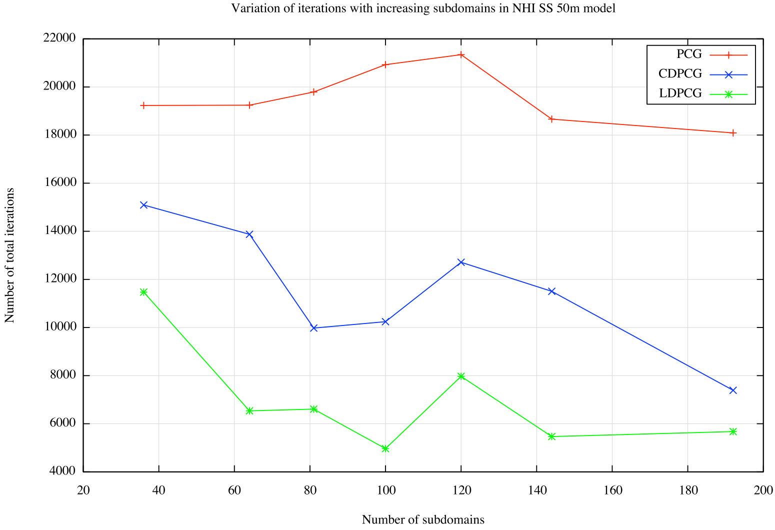
250 m, Two Layer iMOD Unit Case Iterations



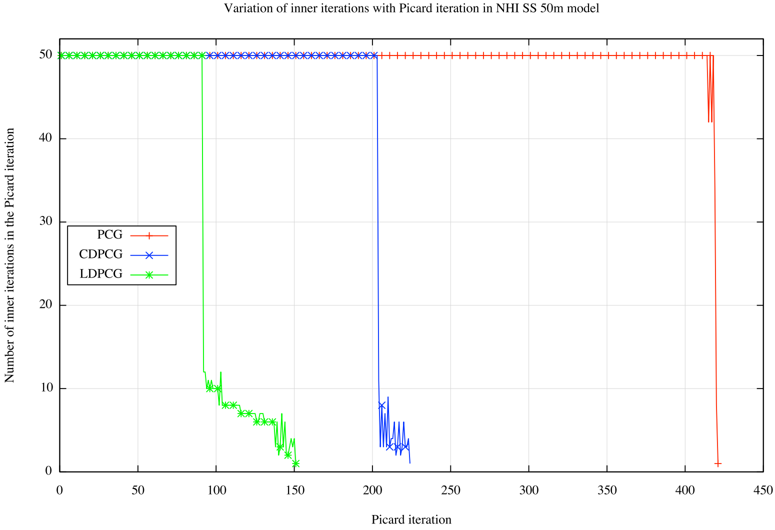
NHI 100m Cellsize: Iteration Improvement



NHI 50m Cellsize: Iteration Improvement



NHI 50m Cellsize: Inner Iteration in Each Picard Iteration



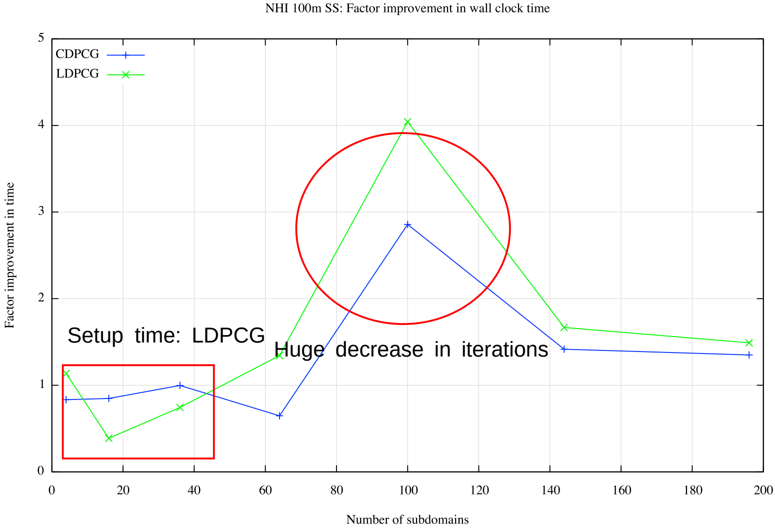
Overview of Results: 100 Subdomains

	PCG	CDPCG		LDPCG		LDPCG SU
Cell size	Iters	Iters	SU	Iters	SU	vs CDPCG SU
250	2527	1768	**1.43**	1496	**1.69**	1.18
100	10775	5209	**2.07**	3313	**3.25**	1.57
50	20927	10244	**2.04**	4966	**4.21**	2.06

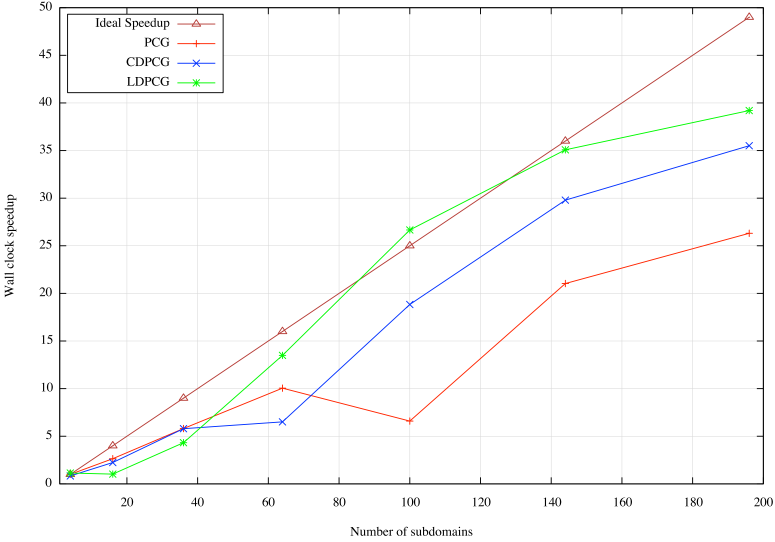
Table: Speed up in iterations (Iters) for NHI SS model with 100 subdomains, SU stands for speed up.

- Performance of LDPCG improves for higher resolution odels.

Improvement in Wall Clock Time: NHI SS 100m



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



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.

Conclusions

- Deflation preconditioner (using linear deflation vectors) has potential to achieve speed up in a wall clock time by factor **4**.

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.

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.

Recommendations

- Investigate the serial solver convergence: by changing the maximum number of inner iterations, checking accuracy of ILU(0) subdomain solve.

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.

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 ?