Hybrid techniques in the solution of large scale problems

Iain S. Duff
iain.duff@stfc.ac.uk

STFC Rutherford Appleton Laboratory
Oxfordshire, UK.

and

CERFACS, Toulouse, France

Homepage: http://www.numerical.rl.ac.uk/people/isd/isd.html
In this talk, we discuss work mainly on projects at CERFACS

The main people involved in this work are:

Mario Arioli, Luc Giraud, Serge Gratton, Azzam Haidar, Xavier Pinel, Jean-Christophe Rioual, Xavier Vasseur
Task is to solve

$$Ax = b$$

where the dimension of $A$ may be $10^6$ or greater.

In our case $A$ is normally from the discretization of a three-dimensional PDE.
Outline

- Direct methods
- Hybrid methods
- Static pivoting
- Domain decomposition
- Helmholtz equation in geophysics
### Direct methods

<table>
<thead>
<tr>
<th>Grid dimensions</th>
<th>Matrix order</th>
<th>Work to factorize</th>
<th>Factor storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k \times k$</td>
<td>$k^2$</td>
<td>$k^3$</td>
<td>$k^2 \log k$</td>
</tr>
<tr>
<td>$k \times k \times k$</td>
<td>$k^3$</td>
<td>$k^6$</td>
<td>$k^4$</td>
</tr>
</tbody>
</table>

$\mathcal{O}$ complexity of direct method on 2D and 3D grids.
Hybrid methods

COMBINING DIRECT AND ITERATIVE METHODS
(can be thought of as sophisticated preconditioning)

Multigrid
Using direct method as coarse grid solver.

Domain Decomposition
Using direct method on local subdomains and “direct” preconditioner on interface.

Block Iterative Methods
Direct solver on sub-blocks.

Partial factorization as preconditioner

Factorization of nearby problem as a preconditioner
A sparse direct method normally consists of three phases

- Analysis (determine ordering and data structures)
- Numerical factorization \( A \rightarrow LDL^T \)
- Solution phase (obtain solution using sparse triangular solves)

When the matrix is positive definite this works well but in the indefinite case subsequent numerical pivoting may mean that the initial analysis is not respected.
Static Pivoting

The default action for general matrices is to use some form of threshold pivoting in the numerical factorization phase.

An alternative is to use **Static Pivoting**, by replacing potentially small pivots $p_k$ by

\[ p_k + \tau \]

and maintaining the same pivoting strategy as advocated in the analysis.

This is even more important in the case of parallel implementation where static data structures are often preferred
Static Pivoting

Several codes use (or have an option for) this device:

- SuperLU (Demmel and Li)
- PARDISO (Gärtner and Schenk)
- MA57 (Duff and Pralet)
- MUMPS (Amestoy, Duff, L’Excellent, and Koster)

mumps.ensee.fr  mumps@cerfacs.fr
Static Pivoting

We thus have factorized

\[ A + E = LDL^T \]

where \(|E| \leq \tau I\)

The four codes then have an Iterative Refinement option

The problem is that this sometimes does not converge
Choosing $\tau$

Increase $\tau \implies$ increase stability of decomposition

Decrease $\tau \implies$ better approximation of the original matrix, reduces $||E||$
Choosing $\tau$

Increase $\tau \implies$ increase stability of decomposition

Decrease $\tau \implies$ better approximation of the original matrix, reduces $||E||$

Trade-off

- $\approx 1 \implies$ huge error $||E||$.
- $\approx \varepsilon \implies$ big growth in preconditioning matrix

Conventional wisdom is to choose

$$\tau = \mathcal{O}(\sqrt{\varepsilon})$$
### Static pivoting

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Order</th>
<th>Entries</th>
<th>Number delayed</th>
<th>Number tiny</th>
<th>Factorization time seconds</th>
<th>Size of the factors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>delayed num</td>
<td>tiny num</td>
<td>delayed static</td>
<td>tiny static</td>
</tr>
<tr>
<td>BRAINPC2</td>
<td>27607</td>
<td>96601</td>
<td>14267</td>
<td>12932</td>
<td>0.18</td>
<td>0.11</td>
</tr>
<tr>
<td>BRATU3D</td>
<td>27792</td>
<td>88627</td>
<td>90052</td>
<td>8429</td>
<td>34.2</td>
<td>9.24</td>
</tr>
<tr>
<td>CONT-201</td>
<td>80595</td>
<td>239596</td>
<td>71296</td>
<td>27470</td>
<td>5.51</td>
<td>1.94</td>
</tr>
<tr>
<td>CONT-300</td>
<td>180895</td>
<td>562496</td>
<td>183306</td>
<td>67864</td>
<td>21.1</td>
<td>6.08</td>
</tr>
<tr>
<td>cvxqp3</td>
<td>17500</td>
<td>62481</td>
<td>30519</td>
<td>6277</td>
<td>9.73</td>
<td>3.08</td>
</tr>
<tr>
<td>DTOC</td>
<td>24993</td>
<td>34986</td>
<td>29478</td>
<td>9790</td>
<td>29.1</td>
<td>0.41</td>
</tr>
<tr>
<td>mario001</td>
<td>38434</td>
<td>114643</td>
<td>15463</td>
<td>10305</td>
<td>0.28</td>
<td>0.23</td>
</tr>
<tr>
<td>NCVXQP1</td>
<td>12111</td>
<td>40537</td>
<td>12463</td>
<td>3619</td>
<td>2.69</td>
<td>1.29</td>
</tr>
<tr>
<td>NCVXQP5</td>
<td>62500</td>
<td>237483</td>
<td>16703</td>
<td>8402</td>
<td>25.7</td>
<td>23.0</td>
</tr>
<tr>
<td>NCVXQP7</td>
<td>87500</td>
<td>312481</td>
<td>195973</td>
<td>31043</td>
<td>195.</td>
<td>71.6</td>
</tr>
<tr>
<td>SIT100</td>
<td>10262</td>
<td>34094</td>
<td>2710</td>
<td>1388</td>
<td>0.13</td>
<td>0.11</td>
</tr>
<tr>
<td>stokes128</td>
<td>49666</td>
<td>295938</td>
<td>18056</td>
<td>12738</td>
<td>1.14</td>
<td>1.06</td>
</tr>
<tr>
<td>stokes64</td>
<td>12546</td>
<td>74242</td>
<td>4292</td>
<td>3106</td>
<td>0.33</td>
<td>0.29</td>
</tr>
</tbody>
</table>

## Component-wise backward error

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Num pivoting strategy</th>
<th>Static pivoting strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>it. 0</td>
<td>it. 1</td>
</tr>
<tr>
<td>BRAINPC2</td>
<td>1.6e-15</td>
<td>1.0e-15</td>
</tr>
<tr>
<td>BRATU3D</td>
<td>2.0e-09</td>
<td>1.7e-16</td>
</tr>
<tr>
<td>CONT-201</td>
<td>8.8e-11</td>
<td>1.6e-16</td>
</tr>
<tr>
<td>CONT-300</td>
<td>7.6e-11</td>
<td>1.9e-16</td>
</tr>
<tr>
<td>cvxq3</td>
<td>5.2e-11</td>
<td>2.7e-16</td>
</tr>
<tr>
<td>DTOC</td>
<td>2.1e-16</td>
<td>2.7e-20</td>
</tr>
<tr>
<td>mario001</td>
<td>6.3e-15</td>
<td>1.3e-16</td>
</tr>
<tr>
<td>NCVXQP1</td>
<td>4.6e-14</td>
<td>1.7e-17</td>
</tr>
<tr>
<td>NCVXQP5</td>
<td>2.0e-11</td>
<td>2.0e-16</td>
</tr>
<tr>
<td>NCVXQP7</td>
<td>9.6e-10</td>
<td>2.2e-16</td>
</tr>
<tr>
<td>SIT100</td>
<td>4.4e-15</td>
<td>1.4e-16</td>
</tr>
<tr>
<td>stokes128</td>
<td>1.1e-14</td>
<td>5.5e-16</td>
</tr>
<tr>
<td>stokes64</td>
<td>4.3e-15</td>
<td>1.5e-15</td>
</tr>
</tbody>
</table>
Arioli, Duff and Gratton have shown that using FGMRES rather than iterative refinement results in a backward stable method that converges for really quite poor factorizations of $A$. 
Numerical experiments

Restarted GMRES vs. FGMRES on CONT-201 test example: $\tau = 10^{-8}$
Domain decomposition

Two PhD theses at CERFACS

Jean-Christophe Rioual
Solving linear systems for semiconductor device simulations on parallel distributed computers
CERFACS report: TH/PA/02/49

and

Azzam Haidar
On the parallel scalability of hybrid linear solvers for large 3D problems
CERFACS report: TH/PA/08/93

www.cerfacs.fr/algor/
Domain decomposition

Matrix representation is:

\[
\begin{pmatrix}
A_{11} & A_{1\Gamma} \\
A_{22} & A_{2\Gamma} \\
A_{33} & A_{3\Gamma} \\
A_{44} & A_{4\Gamma}
\end{pmatrix}
\]

Schur complement is:

\[
A_{\Gamma\Gamma} - \sum_{i=1}^{4} A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}
\]
Domain Decomposition

\[
\begin{pmatrix}
A_{ii} & A_{i\Gamma} \\
A_{\Gamma i} & A_{\Gamma\Gamma}^{(i)}
\end{pmatrix}
\]

where

- \(A_{ii}\) : is the local subproblem,
- \(A_{i\Gamma}\) : is the boundary of the local problem, and
- \(A_{\Gamma\Gamma}^{(i)}\) : is the contribution to the stiffness matrix entries from variables on the artificial interface (\(\Gamma_i\)) around the \(i\)th subregion.

resulting in a contribution to the Schur complement of

\[
S^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma},
\]

called a local Schur (complement).
Hybrid approach

We will use a direct method on the subproblems $A_{ii}$ and an iterative one (perhaps) on the Schur complement.

MUMPS is used as the direct code.
Non-overlapping Domain Decomposition

Algebraic Additive Schwarz preconditioner \[ \text{[ L.Carvalho, L.Giraud, G.Meunant - 01]} \]

\[
S = \sum_{i=1}^{N} R_{\Gamma_i}^{T} S^{(i)} R_{\Gamma_i}
\]

\[
S = \begin{pmatrix}
\ddots & & & \\
S_{kk} & S_{k\ell} & & \\
S_{k\ell} & S_{\ell\ell} & S_{\ell m} & \\
S_{m\ell} & S_{m m} & S_{m n} & \\
S_{n m} & S_{n n} & & \\
\end{pmatrix} \quad \Rightarrow \quad M = \begin{pmatrix}
\ddots & & & & & \\
S_{kk} & S_{k\ell} & & & & -1 \\
S_{k\ell} & S_{\ell\ell} & S_{\ell m} & & & -1 \\
& S_{m\ell} & S_{m m} & S_{m n} & & \\
& & S_{n m} & S_{n n} & & \\
\end{pmatrix}
\]

\[
M = \sum_{i=1}^{N} R_{\Gamma_i}^{T} (\bar{S}^{(i)})^{-1} R_{\Gamma_i}
\]

where \( \bar{S}^{(i)} \) is obtained from \( S^{(i)} \)

\[
S^{(i)} = \begin{pmatrix}
S_{kk}^{(i)} & S_{k\ell}^{(i)} \\
S_{k\ell}^{(i)} & S_{\ell\ell}^{(i)}
\end{pmatrix}
\]

local Schur

\[
\sum_{\ell \in \text{adj}} S_{\ell\ell}^{(i)}
\]

local assembled Schur

\[
\sum_{i \in \text{adj}} S_{kk}^{(i)}
\]

\[
\sum_{i \in \text{adj}} S_{k\ell}^{(i)}
\]

\[
\sum_{i \in \text{adj}} S_{\ell\ell}^{(i)}
\]
The main difference lies in the interface problem (Schur complement). In 2D the interface/interior ratio is small while in 3D there are severe problems in computing and storing the preconditioner.

Therefore, we must seek a cheaper alternative.

Two main ideas (used by Giraud and Haidar)

Sparsify the preconditioner

Set $s_{kl} = 0$ if $s_{kl} < \xi(|s_{kk}| + |s_{ll}|)$

Use 32-bit arithmetic
Runs on System X

3D heterogeneous diffusion problem with $43 \times 10^6$ on 1000 processors

Graphs show effect of sparsification

Even though more iterations are required, the sparsified versions are faster as the time per iteration and preconditioner setup require less time
3D heterogeneous diffusion problem with $43 \times 10^6$ on 1000 processors

Graphs show effect of using mixed precision

Although the number of iterations slightly increases, the mixed approach is fastest down to a level commensurate with the problem
3D heterogeneous diffusion problem with size varying from $5.3 \times 10^6$ to $74 \times 10^6$ degrees of freedom

There is good scalability although the number of iterations grows with the number of subdomains

Two ways to overcome this problem are:
- Coarse grid correction
- Two-level parallelism
Use as many degrees of freedom in the coarse space as subdomains

Work of Carvalho, Giraud, Le Tallec (2001)
3D heterogeneous convection-diffusion problem with $27 \times 10^6$ on 1000 processors

Graphs show effect of sparsification

Even though more iterations are required, the sparsified versions are faster as the time per iteration and preconditioner setup require less time.

Roughly the same as for the pure diffusion problem
Industrial Problem

- Structural mechanics problem from Samtech (Pralet)
- Aircraft fuselage
- 6.5 million degrees of freedom
Fuselage problem of 6.5 million dof mapped on 16 processors

 Runs on IBM JS21 at CERFACS

 The sparse preconditioner setup is four times faster than the dense one (19.5 v.s. 89 seconds)

 In term of global computing time, the sparse algorithm is about twice as fast

 The accuracy of the hybrid solver is comparable to that of the direct solver
Scalability of Fuselage Problem

- Fixed problem size: increasing the # of subdomains → an increase in the # of iterations
- Attractive speedups can be observed
- The sparsified variant is the most efficient
Two levels of parallelism on Fuselage

<table>
<thead>
<tr>
<th># total processors</th>
<th>Algo</th>
<th># subdomains</th>
<th># processors/subdomain</th>
<th># iter</th>
<th>iterative loop time</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 processors</td>
<td>1-level parallel</td>
<td>16</td>
<td>1</td>
<td>147</td>
<td>77.9</td>
</tr>
<tr>
<td></td>
<td>2-level parallel</td>
<td>8</td>
<td>2</td>
<td>98</td>
<td>51.4</td>
</tr>
<tr>
<td>32 processors</td>
<td>1-level parallel</td>
<td>32</td>
<td>1</td>
<td>176</td>
<td>58.1</td>
</tr>
<tr>
<td></td>
<td>2-level parallel</td>
<td>16</td>
<td>2</td>
<td>147</td>
<td>44.8</td>
</tr>
<tr>
<td></td>
<td>2-level parallel</td>
<td>8</td>
<td>4</td>
<td>98</td>
<td>32.5</td>
</tr>
<tr>
<td>64 processors</td>
<td>1-level parallel</td>
<td>64</td>
<td>1</td>
<td>226</td>
<td>54.2</td>
</tr>
<tr>
<td></td>
<td>2-level parallel</td>
<td>32</td>
<td>2</td>
<td>176</td>
<td>40.1</td>
</tr>
<tr>
<td></td>
<td>2-level parallel</td>
<td>16</td>
<td>4</td>
<td>147</td>
<td>31.3</td>
</tr>
<tr>
<td></td>
<td>2-level parallel</td>
<td>8</td>
<td>8</td>
<td>98</td>
<td>27.4</td>
</tr>
</tbody>
</table>

- Reduce the number of subdomains $\implies$ reduce the number of iterations
- Though the subdomain size increases, the time for the iterative loop decreases because:
  - The number of iterations decreases
  - Each subdomain is handled in parallel
Quite recently, Azzam Haidar has been experimenting with matrices which are given as a sparse algebraic data structure without any information about the original problem or the grid. We now show his results from two industrial problems: AMENDE and AUDI, the first from CEA-CESTA and the second from the PARASOL project.

Their characteristics are:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Application</th>
<th>order</th>
<th>number entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amende</td>
<td>Electromagnetics</td>
<td>6,994,683</td>
<td>58,477,383</td>
</tr>
<tr>
<td>Audi</td>
<td>Structures</td>
<td>943,695</td>
<td>39,297,771</td>
</tr>
</tbody>
</table>
Amende problem of 6.99M dof mapped on 32 processors

- Sparse algorithm is twice as fast
- Global sparse conditioner performs well
- Accuracy of hybrid solver is comparable with direct solver
Amende problem of 6.99M dof mapped on 128 processors
Sparse algorithm is similar to dense
Dense preconditioner works well because local Schurs are small
Global sparse conditioner is good numerically but slower
Audi problem of 0.9M dof mapped on 16 processors

For very small $\xi$ convergence only marginally affected but memory savings are substantial

For larger $\xi$ memory is reduced but convergence is poor

Sparsified versions require more iterations but are faster

Accuracy of hybrid solver is comparable with direct solver
Helmholtz Equation in Geophysics

Work with

Serge Gratton
Xavier Vasseur
and
Xavier Pinel

at CERFACS

Helmholtz problem

- Helmholtz equation in the frequency domain:

\[-\Delta u - \frac{\omega^2}{v^2} u = g \quad \text{in} \quad \Omega\]

- with radiation boundary conditions \([k = \frac{\omega}{v}: \text{wavenumber}]:\)

\[\frac{\partial u}{\partial n} - i k u = 0 \quad \text{or} \quad \frac{\partial u}{\partial n} - i k u - \frac{i}{2k} \frac{\partial^2 u}{\partial^2 \tau} = 0 \quad \text{on} \quad \delta \Omega\]

- or with Perfectly Matched Layer (PML) [Berenger, 1994]

Notation:

\(\omega = 2\pi f\) is the angular frequency, \(v\) the velocity of the wave, \(u\) the pressure of the wave, \(g\) represents the source term.
Helmholtz problem with PML formulation

- $\Omega$ is divided into two sets: $\Omega_I$ and $\Omega_{PML}$
- PDE with variable coefficients must now be solved:

$$\begin{cases} 
-\omega^2 u - \frac{1}{\xi_x(x)} \frac{\partial}{\partial x} \left( \frac{1}{\xi_x(x)} \frac{\partial u}{\partial x} \right) - \frac{1}{\xi_y(y)} \frac{\partial}{\partial y} \left( \frac{1}{\xi_y(y)} \frac{\partial u}{\partial y} \right) - \frac{1}{\xi_z(z)} \frac{\partial}{\partial z} \left( \frac{1}{\xi_z(z)} \frac{\partial u}{\partial z} \right) = g \\
\quad u = 0 \text{ on } \delta\Omega = \delta\Omega_{PML}
\end{cases}$$

- Variable complex-valued coefficients only in $\Omega_{PML}$:

$$\xi_d(\delta) = 1 \quad \text{in } \Omega_I \quad \text{and} \quad \xi_d(\delta) = 1 + \frac{i \eta_d(\delta)}{\omega} \quad \text{in } \Omega_{PML}$$

for $d = x, y, z$ and where $\eta_d$ is called a PML function.

- PML function [Operto et al., 2004]

$$\eta_d(\delta) = c_{PML} \cos\left(\frac{\pi}{2L_{PML}}\delta\right) \quad \text{in } \Omega_{PML}$$

where $L_{PML}$ is the width of the PML and $c_{PML}$ is a real positive number.
Discretized problem

- $\Omega$ is always box shaped
- Second-order finite difference discretization methods on non-uniform grids
- Seven-point discretization in three dimensions

- Accuracy requirement for second order discretization: $k h \leq \frac{\pi}{6}$ for 12 points per wavelength
- This leads to a large complex sparse linear system (symmetric in case of radiation boundary conditions)
State of the art solution schemes

- **Sparse multifrontal direct methods:**
  - Very robust but requires too much storage for large-scale problems

- **Multigrid methods:**
  - Multigrid as a solver on the original Helmholtz problem [Elman et al, 2001].
  - **Geometric** multigrid preconditioner on a complex shifted Helmholtz operator [Erlangga, Oosterlee, Vuik, 2006].
We use a **two-level grid** to avoid both smoothing and coarse grid correction difficulties and simultaneously to benefit from the robustness and computational efficiency of modern sparse direct solvers.

We thus use a **direct method** on the nearby problem from a not too coarse grid from **multigrid applied to the original Helmholtz equation**.

Multigrid is **not** a convergent method but acts as a preconditioner for the original (unshifted) Helmholtz operator.

Eigenspectrum of $AC^{-1}$ is clustered around 1 with the isolated eigenvalues captured using **Krylov subspace methods**.
Numerical results

Constant wavenumber: Runs on the CERFACS IBM JS21

<table>
<thead>
<tr>
<th>k</th>
<th>Grid</th>
<th>It</th>
<th>Time (s) Fac.</th>
<th>Mem. (Mb) Fac.</th>
<th>Proc</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>$64^3$</td>
<td>10</td>
<td>3.94</td>
<td>529</td>
<td>2</td>
</tr>
<tr>
<td>45</td>
<td>$96^3$</td>
<td>11</td>
<td>33.24</td>
<td>3323</td>
<td>3</td>
</tr>
<tr>
<td>60</td>
<td>$128^3$</td>
<td>12</td>
<td>73.38</td>
<td>11359</td>
<td>16</td>
</tr>
<tr>
<td>90</td>
<td>$192^3$</td>
<td>13</td>
<td>696.21</td>
<td>62970</td>
<td>32</td>
</tr>
</tbody>
</table>

- **Smoother**: Gauss-Seidel
- **Direct method**: MUMPS
- **Robustness** of the two-grid approach with respect to the wavenumber $k$
Where are the challenges?

Heterogeneous velocity field: $13.5 \times 10 \times 15 \, km^3$, $f = 10 \, Hz$, $h = 12.5\,m$.

- Problem size of $1.16 \times 10^9$ unknowns to be solved for multiple sources (around 500 to 1000 in practice)!
- Indefinite complex-valued problem known as difficult for iterative methods!
Geometric two-grid preconditioner

Two-grid preconditioner

- One cycle of a two-grid method is used as a preconditioner
- Krylov "smoother" as in [Elman, 2001] and [Adams, 2007]: preconditioned \texttt{GMRES(2)}
- Trilinear interpolation and adjoint as restriction
- \texttt{GMRES(m)} as coarse grid solver to solve \textit{only approximately} the coarse grid systems: preconditioned \texttt{GMRES(10)}

Outer Krylov subspace method

- Flexible GMRES [Saad, 1993]: \texttt{FGMRES(5)}
Geometric two-grid preconditioner

- Stopping criterion: \( \frac{\|\bar{r}(it)\|_2}{\|\bar{r}(0)\|_2} \leq 10^{-1} \) with a maximum of 100 iterations of GMRES(10) for the coarse grid

- Stopping criterion: \( \frac{\|r(it)\|_2}{\|r(0)\|_2} \leq 10^{-6} \) with zero initial guess

Three-dimensional benchmark problems

- Both homogeneous and heterogeneous velocity fields
- PML formulation with 15 points on each side of the domain
- Experiments performed on BG/L and BG/P
Homogeneous velocity field on BG/P

Weak scalability experiments [fixed local problem size per core]

<table>
<thead>
<tr>
<th>Grid</th>
<th># Cores</th>
<th>Time (s)</th>
<th>It</th>
<th>Time/It</th>
<th>Mem (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>1024</td>
<td>1687</td>
<td>58</td>
<td>29.08</td>
<td>170</td>
</tr>
<tr>
<td>2048</td>
<td>8192</td>
<td>3718</td>
<td>127</td>
<td>29.28</td>
<td>1362</td>
</tr>
<tr>
<td>4096</td>
<td>65536</td>
<td>9634</td>
<td>327</td>
<td>29.46</td>
<td>10892</td>
</tr>
</tbody>
</table>

- Computations performed in single precision arithmetic
- Velocity is homogeneous and equal to $1500 \text{ m s}^{-1}$
- The wavenumber $k$ is variable ($k \cdot h = \pi / 6$)
- Number of iterations (It) increases linearly with $k$
- The time per iteration is nearly constant
- Memory required (Mem) is increased by a factor of 8 as expected
- A sparse indefinite linear system of more than 68 billion unknowns has been solved
Homogeneous velocity field on BG/P

Strong scalability experiments [fixed global problem size]

<table>
<thead>
<tr>
<th>1/h</th>
<th>Grid</th>
<th># Cores</th>
<th>Time (s)</th>
<th>It</th>
<th>Time/It</th>
<th>Mem (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048</td>
<td>2048³</td>
<td>4096</td>
<td>7706</td>
<td>128</td>
<td>60.20</td>
<td>1341</td>
</tr>
<tr>
<td>2048</td>
<td>2048³</td>
<td>8192</td>
<td>3719</td>
<td>127</td>
<td>29.28</td>
<td>1361</td>
</tr>
<tr>
<td>2048</td>
<td>2048³</td>
<td>16384</td>
<td>1773</td>
<td>128</td>
<td>13.85</td>
<td>1382</td>
</tr>
<tr>
<td>2048</td>
<td>2048³</td>
<td>32768</td>
<td>798</td>
<td>129</td>
<td>6.19</td>
<td>1404</td>
</tr>
</tbody>
</table>

- Computations performed in single precision arithmetic
- Velocity is homogeneous and equal to $1500 \, m \, s^{-1}$
- The wavenumber $k$ is now fixed: $k \, h = \pi / 6$
- Number of iterations (It) is almost independent of the number of cores
- The time per iteration is divided by a factor of 2 as expected [factor greater than 2 due to cache effects]
Heterogeneous velocity field on BG/L

Experiments on BG/L (13.5 × 10 × 15 km³ domain).

<table>
<thead>
<tr>
<th>Grid</th>
<th>h (m)</th>
<th>f (Hz)</th>
<th>Processors</th>
<th>It</th>
<th>T (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>295 × 227 × 327</td>
<td>50</td>
<td>2.5</td>
<td>16</td>
<td>39</td>
<td>25</td>
</tr>
<tr>
<td>567 × 431 × 639</td>
<td>25</td>
<td>5.0</td>
<td>128</td>
<td>83</td>
<td>47</td>
</tr>
<tr>
<td>1119 × 831 × 1247</td>
<td>12.5</td>
<td>10.0</td>
<td>1024</td>
<td>205</td>
<td>107</td>
</tr>
</tbody>
</table>

- Computations performed in double precision arithmetic
- Minimum and maximum velocity are 1500 m s⁻¹ and 6000 m s⁻¹
- Number of iterations increases still linearly with the frequency
Heterogeneous velocity on BG/P IDRIS

Experiments on BG/P (SEG/EAGE Overthrust domain $20 \times 20 \times 5 \ km^3$).

<table>
<thead>
<tr>
<th>Grid</th>
<th>h (m)</th>
<th>f (Hz)</th>
<th>Processors</th>
<th>It</th>
<th>T (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$863 \times 863 \times 231$</td>
<td>24.21</td>
<td>7.5</td>
<td>64</td>
<td>37</td>
<td>2678</td>
</tr>
<tr>
<td>$1690 \times 1690 \times 426$</td>
<td>12.11</td>
<td>15.0</td>
<td>512</td>
<td>102</td>
<td>6362</td>
</tr>
<tr>
<td>$3356 \times 3356 \times 829$</td>
<td>6.05</td>
<td>30.0</td>
<td>4096</td>
<td>490</td>
<td>28601</td>
</tr>
</tbody>
</table>

- Computations performed in double precision arithmetic
- Minimum and maximum velocity are $2200 \ m \ s^{-1}$ and $6000 \ m \ s^{-1}$
- Number of iterations no longer increases linearly with the frequency
Heterogeneous velocity on BG/P IDRIS

Experiments on BG/P (SEG/EAGE salt domain $8 \times 8 \times 4 \ km^3$ domain).

<table>
<thead>
<tr>
<th>Grid</th>
<th>h (m)</th>
<th>f (Hz)</th>
<th>Processors</th>
<th>It</th>
<th>T (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$671 \times 671 \times 351$</td>
<td>12.500</td>
<td>10</td>
<td>64</td>
<td>43</td>
<td>2797</td>
</tr>
<tr>
<td>$1311 \times 1311 \times 671$</td>
<td>6.250</td>
<td>20</td>
<td>512</td>
<td>101</td>
<td>6117</td>
</tr>
<tr>
<td>$2597 \times 2597 \times 1317$</td>
<td>3.125</td>
<td>40</td>
<td>4096</td>
<td>283</td>
<td>16492</td>
</tr>
</tbody>
</table>

- Computations performed in double precision arithmetic
- Minimum and maximum velocity are $1500 \ m \ s^{-1}$ and $4400 \ m \ s^{-1}$
- Number of iterations no longer increases linearly with the frequency
Conclusions

We can solve really large, realistic and computationally challenging problems in important application areas.

A range of techniques involving both sparse direct and a range of sparse iterative solvers is required including hybrid methods.
Conclusions

THANK YOU
for your attention
Heterogeneous velocity field on BG/L

$13.5 \times 10 \times 15 \ km^3$, $f = 2.5 \ Hz$

- Problem size of $2.19 \times 10^7$ unknowns
Heterogeneous velocity field on BG/L

$13.5 \times 10 \times 15 \ km^3$, $f = 5 \ Hz$

Problem size of $1.56 \times 10^8$ unknowns
Heterogeneous velocity field on BG/L

\[ 13.5 \times 10 \times 15 \text{ km}^3, \ f = 10 \text{ Hz} \]

Problem size of \( 1.16 \times 10^9 \) unknowns
SEG/EAGE Overthrust velocity field on BG/P

$20 \times 20 \times 5 \, km^3$

SEG/EAGE Overthrust velocity field (20km x 20km x 5km)

Velocity Field

7.5Hz

15Hz

10Hz
SEG/EAGE Salt velocity field on BG/P

\[ 8 \times 8 \times 4 \, km^3 \]

SEG/EAGE Salt Dome velocity field (8kmx8kmx4km)

Velocity Field

10Hz

20Hz

15Hz