Fast solvers and practical implementation in PDE-constrained optimization

Andrew T. Barker

LLNL-PRES-693611. This work was performed under the auspices of the U.S. Department of Energy under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC
Solvers for fully coupled time-dependent optimization problems. (with M. Stoll)

$H^1$ regularization in PDE-constrained optimization. (with T. Rees and M. Stoll)

Optimization of Navier slip boundary conditions. (with H. Antil and S. W. Walker)

Algebraic multigrid optimization solvers. (with A. Draganescu)
PART ONE

Solvers for fully coupled time-dependent optimization problems

Joint work with:
▶ M. Stoll, Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg.
We are interested in finding a state $y$ and a control $u$ to minimize the functional

$$J(y, u) = \int_0^T \|y - \bar{y}\|^2 \, dt + \frac{\omega}{2} \int_0^T \|u\|^2 \, dt$$

subject to some time–dependent partial differential equation

$$\mathcal{L}(y) = u$$

where $\bar{y}$ is a given desired state.
Examples

- Heat equation: \( \mathcal{L} = \frac{\partial}{\partial t} - \Delta \)

- Convection–diffusion: \( \mathcal{L} = \frac{\partial}{\partial t} - \epsilon \Delta + \beta \cdot \nabla \)

- Stokes: \( \mathcal{L} \approx \left( \begin{array}{c} \frac{\partial}{\partial t} \\ 0 \end{array} \right) + \left( \begin{array}{cc} -\Delta & \nabla \\ \nabla \cdot & 0 \end{array} \right) \)

(with appropriate boundary conditions)
All-at-once approach

\[-\frac{\partial \lambda}{\partial t} - A^* \lambda = -(y - \bar{y}) \]  \hspace{1cm} \text{(with final condition)}
\[
\frac{\partial y}{\partial t} - Ay = \frac{1}{\omega} \lambda \]  \hspace{1cm} \text{(with initial condition)}

Taking liberties with boundary conditions and regularity in space and time, we can (formally) obtain the normal equations

\[-\omega \frac{\partial^2 y}{\partial t^2} + (\omega A + \omega A^*) \frac{\partial y}{\partial t} + (\omega A^* A + 1)y = \bar{y} \]  \hspace{1cm} \text{(with boundary conditions in time)}
All-at-once approach

We have been solving this problem with an “all-at-once” approach. If $y$ is a vector that represents a spatial state, then we will solve a system for the unknown block vector

$$[y_1, y_2, y_3, \ldots, u_1, u_2, u_3, \ldots, \lambda_1, \lambda_2, \lambda_3]$$

So

number of unknowns $= 3 \times$ (spatial degrees of freedom)
$\times$ (number of timesteps)
## Memory use and parallel computing

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N_T$</th>
<th>$N_x$</th>
<th>iterations</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.11e+08</td>
<td>256</td>
<td>274 625</td>
<td>13</td>
<td>3962</td>
</tr>
<tr>
<td>4.22e+08</td>
<td>512</td>
<td></td>
<td>15</td>
<td>6355</td>
</tr>
<tr>
<td>8.44e+08</td>
<td>1024</td>
<td></td>
<td>16</td>
<td>10879</td>
</tr>
<tr>
<td>1.69e+09</td>
<td>2048</td>
<td></td>
<td>19</td>
<td>21890</td>
</tr>
</tbody>
</table>
We want to minimize

\[ J(y, u) = \int_0^T \| y - \bar{y} \|_{L^2(\Omega)}^2 \, dt + \frac{\omega}{2} \int_0^T \| u \|_{L^2(\Omega)}^2 \, dt \]

subject to the heat equation

\[ \frac{\partial y}{\partial t} - \Delta y = u \]

with some boundary conditions.
We discretize in space with quadrilateral finite elements.
We discretize the time derivative with backward Euler.
We integrate the functional in time with the trapezoid rule.
We follow the discretize then optimize approach.

(Stoll and Wathen 2010)
All-at-once approach

The optimality system can be written

\[
\begin{pmatrix}
\tau M^{1/2} & 0 & -K^T \\
0 & \omega \tau M^{1/2} & \tau M \\
-K & \tau M & 0
\end{pmatrix}
\begin{pmatrix}
y \\
u \\
\lambda
\end{pmatrix}
=
\begin{pmatrix}
\tau M^{1/2} \bar{y} \\
0 \\
d
\end{pmatrix}
\]
Some of the pieces

$$\mathcal{K} = \begin{pmatrix} M + \tau K & -M & M + \tau K \\ -M & M + \tau K & -M \\ & \ddots & \ddots \\ & -M & M + \tau K \end{pmatrix}$$
Some of the pieces

\[ M = \begin{pmatrix} M & M & \cdots & M \\ M & M & \cdots & M \\ \vdots & \vdots & \ddots & \vdots \\ M & M & \cdots & M \end{pmatrix} \]
All-at-once approach

The optimality system can be written

\[
\begin{pmatrix}
\tau M_{1/2} & 0 & -K^T \\
0 & \omega \tau M_{1/2} & \tau M \\
-K & \tau M & 0
\end{pmatrix}
\begin{pmatrix}
y \\
u \\
\lambda
\end{pmatrix}
= 
\begin{pmatrix}
\tau M_{1/2} \bar{y} \\
0 \\
d
\end{pmatrix}
\]

This system is symmetric but indefinite and in serial it is well-suited to preconditioned Minres.
Schur complement preconditioning

\[
\begin{pmatrix}
A & B^T \\
B & 0
\end{pmatrix}
= 
\begin{pmatrix}
\mathcal{M}_{1/2} & 0 & -\mathcal{K}^T \\
0 & \omega \tau \mathcal{M}_{1/2} & \tau \mathcal{M} \\
-\mathcal{K} & \tau \mathcal{M} & 0
\end{pmatrix}
\]
Schur complement preconditioning

\[
P = \begin{pmatrix} A & 0 \\ 0 & BA^{-1}B^T \end{pmatrix} = \begin{pmatrix} \tau M_{1/2} & \omega \tau M_{1/2} \\ \omega \tau M_{1/2} & S \end{pmatrix}
\]

where

\[
S = KM^{-1}K^T + \frac{\tau}{\omega}MM_u^{-1}M^T
\]
Schur complement preconditioning

True Schur complement:

\[ S = \mathcal{K} \mathcal{M}^{-1} \mathcal{K}^T + \frac{T}{\omega} \mathcal{M} \mathcal{M}_u^{-1} \mathcal{M}^T \]

[Dollar, Rees, Wathen 2010] (our heat and Stokes problems)

[\hat{S} = -\mathcal{K} \mathcal{M}^{-1} \mathcal{K}^T]

[Pearson, Wathen 2010] (our convection–diffusion problem)

[\hat{S} = \left( \mathcal{K} + \frac{1}{\sqrt{\omega}} \mathcal{M} \right) \mathcal{M}^{-1} \left( \mathcal{K}^T + \frac{1}{\sqrt{\omega}} \mathcal{M} \right) ]
We need to approximate the inverse of the Schur complement $BA^{-1}B^T \approx \mathcal{K}M^{-1}\mathcal{K}^T$. Recall

$$\mathcal{K} = \begin{pmatrix}
M + \tau K & M + \tau K \\
-M & -M & M + \tau K \\
& & & \ddots
\end{pmatrix}$$

We can do a sweep through time with the help of an algebraic multigrid preconditioner for $M + \tau K$.
(Rees et. al. 2010, Stoll and Wathen 2010)
Schur complement preconditioning

\[ \mathcal{K} = \begin{pmatrix} M + \tau K & M + \tau K \\ -M & -M & M + \tau K \\ & & \ddots \end{pmatrix} \]

- This works fine, but it does not parallelize well.
- Our problems are too big to solve in serial.
Schwarz preconditioning

\[ T_1 \quad \delta \quad T_2 \]
Schwarz preconditioning

If $A$ is the large system matrix, then we define a one-level additive Schwarz preconditioner

$$B^{-1} = \sum_{j=1}^{N_p} I_j A_j^{-1} I_j^T$$

where $I_j$ extends a function on a sub-timedomain to the whole timedomain, and $A_j$ represents an optimization problem with zero initial and final condition on the sub-timedomain.
Local subproblems

\[ A_j = \begin{pmatrix} A_j & B_j^T \\ B_j & 0 \end{pmatrix} \]

Within the Schwarz preconditioner, we approximate the inverse of \( A_j \) using Minres, preconditioned by using the diagonal Schur complement approach discussed earlier.
Other approaches to this problem

- (Yang, Prudencio, Cai 2012) do parallel domain decomposition in space.
- (Heinkenschloss 2005) breaks time into subdomains, but focus is not on parallel computation.
- (Du et. al. 2013) uses the Parareal method to precondition (only) the forward problem.
One–level Schwarz preconditioner

\[ B^{-1} = \sum_j I_j A_j^{-1} I_j^T \]

- This preconditioner is in general indefinite.
- Minres requires a symmetric and positive definite preconditioner, so will instead use the quasi-minimal residual method (QMR).
Symmetric, flexible QMR

- There is a special case of QMR with a symmetric matrix and a symmetric (but not necessarily positive definite!) preconditioner which reduces the cost to be comparable to Minres. (Freund and Nachtigal 1994)

- Another variation of QMR (flexible QMR) allows for the preconditioner to change (a little) at each iteration. (Szyld and Vogel 2001)

- We will use both of these variations together.
Our approach in one slide

- Solve the global optimization system with flexible, symmetric QMR.
- The global preconditioner is a one–level additive Schwarz preconditioner in time:
  - Divide the time domain into a bunch of slices, give each slice to a processor.
  - Solve each subdomain problem with Minres.
  - Precondition the local subdomain problems with the Schur complement approach.
Numerical results

- Running on a standard Linux cluster with 24 GB of memory per node.
- Using deal.II and MPI.
- Key issue is memory use—we use 4 processes per node even though each node has 12 cores.
Numerical results

Slice of a 3D solution with $\omega = 10^{-6}$, desired state on the left, achieved solution on the right.
Numerical results

Slice of a 3D solution with $\omega = 10^{-6}$, control is shown.

[scale is from 0 to 2.01e+4]
Large problems running on 64 processors (16 nodes), overlap of 2 time steps, and $\omega = 10^{-4}$:

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N_T$</th>
<th>$N_x$</th>
<th>iterations</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.11e+08</td>
<td>256</td>
<td>274 625</td>
<td>13</td>
<td>3962</td>
</tr>
<tr>
<td>4.22e+08</td>
<td>512</td>
<td></td>
<td>15</td>
<td>6355</td>
</tr>
<tr>
<td>8.44e+08</td>
<td>1024</td>
<td></td>
<td>16</td>
<td>10879</td>
</tr>
<tr>
<td>1.69e+09</td>
<td>2048</td>
<td></td>
<td>19</td>
<td>21890</td>
</tr>
</tbody>
</table>
## Strong scaling

With 274625 spatial degrees of freedom and 256 timesteps, overlap of 2 timesteps, $\omega = 10^{-4}$:

<table>
<thead>
<tr>
<th>cores</th>
<th>iterations</th>
<th>time (sec)</th>
<th>time/iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>10904.2</td>
<td>1363.0</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>6152.1</td>
<td>769.0</td>
</tr>
<tr>
<td>32</td>
<td>9</td>
<td>4035.8</td>
<td>448.4</td>
</tr>
<tr>
<td>64</td>
<td>13</td>
<td>3956.2</td>
<td>304.3</td>
</tr>
</tbody>
</table>
Weak scaling

With 274625 spatial degrees of freedom and overlap of 2:

<table>
<thead>
<tr>
<th>$N_T$</th>
<th>cores</th>
<th>iterations</th>
<th>total time (sec)</th>
<th>time/iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>4</td>
<td>6</td>
<td>4096.2</td>
<td>682.7</td>
</tr>
<tr>
<td>128</td>
<td>8</td>
<td>7</td>
<td>4850.3</td>
<td>692.9</td>
</tr>
<tr>
<td>256</td>
<td>16</td>
<td>9</td>
<td>5532.8</td>
<td>614.8</td>
</tr>
<tr>
<td>512</td>
<td>32</td>
<td>17</td>
<td>9952.0</td>
<td>585.4</td>
</tr>
</tbody>
</table>
Stokes equations

We want to minimize

\[ J(y, u) = \int_0^T \| y - \bar{y} \|^2_{L^2(\Omega)} \, dt + \omega \int_0^T \| u \|^2_{L^2(\Omega)} \, dt \]

subject to the Stokes system

\[ \frac{\partial y}{\partial t} - \nu \Delta y + \nabla p = u \]
\[ \nabla \cdot y = 0, \]

with some boundary conditions.
Stokes equations
Stokes problem—strong scaling

In two dimensions, with 37507 spatial degrees of freedom, $N_T = 256$ timesteps, overlap of 2, and $\omega = 10^{-4}$.

<table>
<thead>
<tr>
<th>cores</th>
<th>iterations</th>
<th>time (sec)</th>
<th>time/iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>6</td>
<td>9580.1</td>
<td>1596.7</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>5480.9</td>
<td>913.5</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>3435.3</td>
<td>572.6</td>
</tr>
<tr>
<td>64</td>
<td>10</td>
<td>3206.3</td>
<td>320.6</td>
</tr>
</tbody>
</table>
In two dimensions, with 37507 spatial degrees of freedom, $N_T = 256$ timesteps, overlap of 2, and $\omega = 10^{-4}$.

<table>
<thead>
<tr>
<th>$N_T$</th>
<th>cores</th>
<th>iterations</th>
<th>time (sec)</th>
<th>time/iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>2</td>
<td>4</td>
<td>2208.1</td>
<td>552.0</td>
</tr>
<tr>
<td>65</td>
<td>4</td>
<td>4</td>
<td>3125.3</td>
<td>781.3</td>
</tr>
<tr>
<td>129</td>
<td>8</td>
<td>5</td>
<td>4331.2</td>
<td>866.2</td>
</tr>
<tr>
<td>257</td>
<td>16</td>
<td>6</td>
<td>5541.2</td>
<td>923.5</td>
</tr>
<tr>
<td>513</td>
<td>32</td>
<td>11</td>
<td>11321.4</td>
<td>1029.2</td>
</tr>
<tr>
<td>1025</td>
<td>64</td>
<td>18</td>
<td>17496.6</td>
<td>972.0</td>
</tr>
</tbody>
</table>
Convection–diffusion
Strong scaling, convection–diffusion

\[ \epsilon = 0.1, \omega = 10^{-3}, \text{32768 spatial degrees of freedom, overlap of 2 timesteps:} \]

<table>
<thead>
<tr>
<th>cores</th>
<th>iterations</th>
<th>time (sec)</th>
<th>time/iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>5577.2</td>
<td>929.5</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>2946.2</td>
<td>491.0</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>1663.9</td>
<td>277.3</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>985.7</td>
<td>164.3</td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>915.1</td>
<td>114.4</td>
</tr>
</tbody>
</table>
Weak scaling, convection–diffusion

\[ \epsilon = 0.1, \omega = 10^{-3}, \text{32768 spatial degrees of freedom, overlap of 2 timesteps:} \]

<table>
<thead>
<tr>
<th>(N_T)</th>
<th>cores</th>
<th>iterations</th>
<th>time (sec)</th>
<th>time/iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>2</td>
<td>4</td>
<td>1542.3</td>
<td>385.6</td>
</tr>
<tr>
<td>127</td>
<td>4</td>
<td>5</td>
<td>2570.2</td>
<td>514.0</td>
</tr>
<tr>
<td>255</td>
<td>8</td>
<td>6</td>
<td>2974.8</td>
<td>495.8</td>
</tr>
<tr>
<td>511</td>
<td>16</td>
<td>7</td>
<td>3333.1</td>
<td>476.2</td>
</tr>
<tr>
<td>1023</td>
<td>32</td>
<td>9</td>
<td>4099.9</td>
<td>455.5</td>
</tr>
<tr>
<td>2047</td>
<td>64</td>
<td>13</td>
<td>5760.5</td>
<td>443.1</td>
</tr>
</tbody>
</table>
Weak scaling, convection–diffusion

Time per iteration.
Robustness with $\omega$ for convection–diffusion

32768 spatial degrees of freedom, 32 cores, $\epsilon = 0.1$.

<table>
<thead>
<tr>
<th>$N_T$</th>
<th>$10^{-2}$</th>
<th>$10^{-3}$</th>
<th>$10^{-4}$</th>
<th>$10^{-6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>127</td>
<td>11</td>
<td>7</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>255</td>
<td>12</td>
<td>8</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>511</td>
<td>14</td>
<td>10</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>1023</td>
<td>14</td>
<td>11</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
Interplay of $\omega$ and $\epsilon$ for convection–diffusion

$N_T = 127$, 32768 spatial degrees of freedom, running on 32 cores.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$\omega$</th>
<th>0.1</th>
<th>0.01</th>
<th>$10^{-3}$</th>
<th>$10^{-4}$</th>
<th>$10^{-6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>33</td>
<td>11</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>18</td>
<td>14</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>32</td>
<td>13</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
PART TWO

$H^1$ regularization in PDE-constrained optimization

Joint work with:
- T. Rees, Rutherford Appleton Laboratory, Oxford.
- M. Stoll, Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg.
Now we minimize

\[ J(y, u) = \int_0^T \| y - \bar{y} \|_{L^2}^2 \, dt + \frac{\omega}{2} \int_0^T \| u \|_{H^1}^2 \, dt \]

subject to the heat equation.
All at once system

\[
\begin{pmatrix}
\tau M & 0 & -K^T \\
0 & \omega \tau (M_u + K_u) & \tau M \\
-K & \tau M & 0
\end{pmatrix}
\begin{pmatrix}
y \\
u \\
\lambda
\end{pmatrix}
=
\begin{pmatrix}
\tau M \bar{y} \\
0 \\
d
\end{pmatrix}
\]

(being a bit sloppy with some timestep scaling)
The true Schur complement is

$$S = \frac{1}{\tau} \mathcal{K} \mathcal{M}^{-1} \mathcal{K} + \frac{\tau}{\omega} \mathcal{M} (\mathcal{M}_u + \mathcal{K}_u)^{-1} \mathcal{M}^T$$

For large $\omega$ we can just use the first term

$$\hat{S} = \frac{1}{\tau} \mathcal{K} \mathcal{M}^{-1} \mathcal{K}$$

But this is not robust with respect to $\omega$. 
Schur complement preconditioning

Following the ideas of (Pearson, Stoll, Wathen 2012). The true Schur complement is

\[ S = \frac{1}{\tau} K M^{-1} K + \frac{\tau}{\omega} \mathcal{M} (\mathcal{M} u + K u)^{-1} \mathcal{M}^T \]

So we will look for matrices \( \mathcal{X}_1, \mathcal{X}_2 \) so that

\[ \hat{S} = \frac{1}{\tau} (K + \mathcal{X}_1) M^{-1} (K + \mathcal{X}_2) \]

is close to \( S \).
Schur complement preconditioning

Skipping all the details, we end up with

\[
\hat{S} = \frac{1}{\tau} \left( \mathcal{K} + \frac{\tau}{\sqrt{\omega}} \mathcal{M}(\mathcal{M}_u + \mathcal{K}_u)^{-1}\mathcal{M} \right) \mathcal{M}^{-1} \left( \mathcal{K} + \frac{\tau}{\sqrt{\omega}} \mathcal{M} \right)^T
\]

We have to approximate the inverse of this.
A few results

Iteration counts for $H^1$ regularized heat equation optimization:

<table>
<thead>
<tr>
<th>dofs</th>
<th>$\omega = 10^{-2}$</th>
<th>$\omega = 10^{-4}$</th>
<th>$\omega = 10^{-6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1089</td>
<td>13</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>4225</td>
<td>13</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td>16641</td>
<td>15</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>66049</td>
<td>17</td>
<td>20</td>
<td>31</td>
</tr>
<tr>
<td>263169</td>
<td>19</td>
<td>22</td>
<td>34</td>
</tr>
</tbody>
</table>
PART THREE

Optimization of Navier slip boundary conditions

Joint work with:
- H. Antil, George Mason University.
- S. W. Walker, Louisiana State University.
Superhydrophobic materials

Among many applications is *drag reduction*.

They are nanoscale materials and are extremely expensive to produce.
Flow past superhydrophobic materials can be modeled with a Navier boundary condition.

\[ u \cdot \nu = 0 \]

\[ \frac{1}{\gamma} Tu + T\sigma(u, p)\nu = 0 \]

These kind of boundary conditions are fairly well studied mathematically, but very little has been done computationally.

We aim to apply PDE-constrained optimization to determine if using these materials can be economically beneficial, and if so, where to use these materials and how much to use.
Optimization problem

We aim to minimize

\[ J(u, p, \gamma) = \text{drag}(u, p) + \omega \text{cost}(\gamma) \]

subject to

steady-state Stokes equations

\[ u \cdot \nu = 0 \]
\[ \frac{1}{\gamma} Tu + T\sigma(u, p)\nu = 0 \]
Regularization parameter $\omega = 0.01$
Regularization parameter $\omega = 0.01$
Regularization parameter $\omega = 1.0e + 4$
Regularization parameter $\omega = 1.0e + 4$
Algebraic multigrid for PDE-constrained optimization

Joint work with:
- A. Draganescu, University of Maryland, Baltimore County.
The problem

Given a desired state $\bar{y}$, find a (distributed) control $u$ to minimize the functional

$$J = \frac{1}{2} \| y - \bar{y} \|_Y^2 + \frac{\omega}{2} \| u \|_U^2$$

subject to the constraint

$$Ay = M_{yu}u,$$

where $A$ is an (elliptic) spatial operator on the state space.
The reduced space problem

The solution operator for the state equation is $\mathcal{K} = A^{-1} M_y u$. The reduced space problem is then

$$G u = (\mathcal{K}^* \mathcal{K} + \omega I) u = \mathcal{K}^* \bar{y},$$

Where

$$\mathcal{K}^* = M_u^{-1} \mathcal{K}^T M_y$$

We are interested in preconditioning the Hessian $G$. 
We would like to form a coarse Hessian $G_H$.
We cannot apply algebraic multigrid directly to $G$, because we don’t want to form it.
We follow the geometric multigrid of (Draganescu and Dupont, 2008).
AMG for the Hessian

Given an AMG prolongator $P$, we can coarsen:

- $A_H = P^T A P$
- $M_H = P^T M P$
- $\mathcal{K}_H = A_H^{-1} M_H$
- $G_H = (\mathcal{K}_H^* \mathcal{K} + \omega I)$

And define a two-level preconditioner by

$$M_h^{-1} = P (G_H)^{-1} \Pi + \frac{1}{\omega} (I - P \Pi),$$

where

$$\Pi = M_H^{-1} P^T M.$$
We use a multilevel (W-cycle based) version of this algorithm. The discretization is standard low order Lagrange finite elements. The AMG is Hypre BoomerAMG with default parameters. We cheat by using (almost) the same space for state and control, so we can reuse mass matrices. We solve the Poisson problem on a 3D cube with an unstructured mesh on 8 processors.
Iteration counts for unpreconditioned CG for the Hessian

<table>
<thead>
<tr>
<th>refine</th>
<th>$\omega$</th>
<th>0.0001</th>
<th>0.01</th>
<th>1.0</th>
<th>100.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>60</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>62</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>57</td>
<td>66</td>
<td>67</td>
<td>67</td>
<td></td>
</tr>
</tbody>
</table>
Iteration counts for CG preconditioned with our multilevel preconditioner

<table>
<thead>
<tr>
<th>refine</th>
<th>( \omega )</th>
<th>0.0001</th>
<th>0.01</th>
<th>1.0</th>
<th>100.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
## Results

Solve times for unpreconditioned CG

<table>
<thead>
<tr>
<th>refine</th>
<th>$\omega$</th>
<th>0.0001</th>
<th>0.01</th>
<th>1.0</th>
<th>100.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.4565</td>
<td>1.3022</td>
<td>1.2594</td>
<td>1.2654</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8.3637</td>
<td>8.0912</td>
<td>8.2210</td>
<td>8.0970</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>98.7817</td>
<td>119.7712</td>
<td>105.5309</td>
<td>119.5120</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1114.9886</td>
<td>1526.1242</td>
<td>1513.2152</td>
<td>1325.6787</td>
<td></td>
</tr>
</tbody>
</table>
Results

Solve times for CG preconditioned with our multilevel preconditioner

<table>
<thead>
<tr>
<th>refine</th>
<th>$\omega$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0001</td>
<td>0.01</td>
<td>1.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2</td>
<td>0.4549</td>
<td>0.1830</td>
<td>0.1019</td>
<td>0.1042</td>
</tr>
<tr>
<td>3</td>
<td>2.4991</td>
<td>0.9040</td>
<td>0.5052</td>
<td>0.5559</td>
</tr>
<tr>
<td>4</td>
<td>27.7249</td>
<td>9.6749</td>
<td>5.0136</td>
<td>5.1048</td>
</tr>
<tr>
<td>5</td>
<td>287.4749</td>
<td>104.0656</td>
<td>59.8052</td>
<td>58.5670</td>
</tr>
</tbody>
</table>
Conclusions

Solvers in PDE-constrained optimization are important for
- Large 3D problems
- Time-dependent problems
- Robustness with respect to parameters

Solvers for the forward problem are not enough. For realistic optimization problems you need more.