A Fast Fourier transform based direct solver for the Helmholtz problem

Jari Toivanen      Monika Wolfmayr

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

AANMPDE 2017
Palaiochora, Crete, Oct 2, 2017
Content

1. Model problem
2. Discretization
3. Fast solver
4. Numerical results
5. Conclusions und outlook
Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a $d$-dimensional rectangular domain. The pressure field satisfies the **Helmholtz** partial differential equation

$$-\Delta u - \omega^2 u = f \quad \text{in } \Omega,$$

$$\mathcal{B}u = 0 \quad \text{on } \Gamma,$$  \hspace{1cm} (1, 2)

where $\omega$ denotes the wave number. The boundary

$\Gamma = \partial \Omega = \Gamma_N \cup \Gamma_B$ is decomposed into Neumann boundary condition (BC) $\Gamma_N$ and (first-order) absorbing BC (ABC) $\Gamma_B$:

$$\mathcal{B}u = \nabla u \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_N,$$

$$\mathcal{B}u = \nabla u \cdot \mathbf{n} - i\omega u = 0 \quad \text{on } \Gamma_B,$$  \hspace{1cm} (3, 4)

where $\mathbf{n}$ denotes the outward normal to the boundary. Equation (2) is an approximation for the Sommerfeld radiation condition.
Discretization

Weak formulation for the Helmholtz problem (1)–(2): Find $u \in V = H^1(\Omega)$ such that

$$a(u, v) = \int_{\Omega} f v \, dx \quad \forall v \in V,$$

(5)

where

$$a(u, v) = \int_{\Omega} (\nabla u \cdot \nabla v - \omega^2 uv) \, dx - i\omega \int_{\partial\Omega} uv \, ds.$$

(6)

Discretizing (5) by bilinear or trilinear finite elements on an orthogonal mesh leads to a system of linear equations given by

$$Au = f,$$

(7)

where the matrix $A$ has a separable tensor product form. The mesh will be equidistant in each direction $x_j$. 

M. Wolfmayr  
FFT based direct solver for the Helmholtz problem  4 / 19
Discretization

For the two-dimensional (2d) case, the matrix $A$ is given by

$$A = (K_1 - \omega^2 M_1) \otimes M_2 + M_1 \otimes K_2,$$

whereas in three dimensions (3d) it is given by

$$A = (K_1 - \omega^2 M_1) \otimes M_2 \otimes M_3 + M_1 \otimes (K_2 \otimes M_3 + M_2 \otimes K_3),$$

where $K_j$ and $M_j$ are one-dimensional stiffness and mass matrices, respectively, in the $x_j$-direction with possible modifications on or near the boundaries due to the ABC.
Discretization

\( K_j \) and \( M_j \) are computed by 1d numerical quadrature on \([0, 1]\):

\[
K_j = \frac{1}{h_j} \begin{pmatrix}
  k_{1,1} & -1 & -1 \\
  -1 & 2 & -1 \\
  -1 & 2 & -1 \\
  \vdots & \ddots & \ddots \\
  -1 & 2 & k_{n_j,n_j}
\end{pmatrix}, \quad M_j = \frac{h_j}{6} \begin{pmatrix}
  2 & 1 & 1 \\
  1 & 4 & 1 \\
  \vdots & \ddots & \ddots \\
  1 & 4 & 1 \\
  1 & 2 & 1
\end{pmatrix}
\]

where the first and last entries are including the corresponding BCs. ABCs (4) yield the entries \( k_{1,1} = k_{n_j,n_j} = 1 - i\omega h_j \), whereas Neumann BCs lead to \( k_{1,1} = k_{n_j,n_j} = 1 \).

\( M_j \) is the same for both Neumann and (first-order) ABCs.

Let the ABCs be given in direction of \( x_1 \) for both (opposite) sides.
Fast solver - idea

The main idea for solving the problem $Au = f$ is to consider an auxiliary problem $Bv = f$, where the system matrix $B$ is derived by changing the ABCs to periodic ones. The key is that we can solve the modified (periodic) problem $Bv = f$ now by using the FFT method, which is not possible for the original problem $Au = f$.

The problem $Au = f$ can be solved applying the following steps:
1. Solve $Bv = f$.
2. Solve $Aw = f - Av = Bv - Av = (B - A)v$, $u = v + w$.
3. Solve $Bu = f + (B - A)(v + w)$. 

M. Wolfmayr  FFT based direct solver for the Helmholtz problem  7 / 19
In case of periodic BCs in $x_1$-direction, the matrices $K_1$ and $M_1$ change to $K_1^B$ and $M_1^B$ and are given by

$$K_1^B = \frac{1}{h_1} \begin{pmatrix} 2 & -1 & -1 & \cdots & -1 \\ -1 & 2 & -1 & \cdots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & 2 & -1 & 2 \end{pmatrix}, \quad M_1^B = \frac{h_1}{6} \begin{pmatrix} 4 & 1 & 1 & \cdots & 1 \\ 1 & 4 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 4 & 1 & 4 \end{pmatrix},$$

which means that the BCs on the two opposite $x_1$-boundaries have been changed to be of periodic type. The matrix $B$ is given by

$$B = (K_1^B - \omega^2 M_1^B) \otimes M_2 + M_1^B \otimes K_2 \quad (2d),$$

$$B = (K_1^B - \omega^2 M_1^B) \otimes M_2 \otimes M_3 + M_1^B \otimes (K_2 \otimes M_3 + M_2 \otimes K_3) \quad (3d).$$
After a suitable permutation $A$ and $B$ have the block forms

$$A = \begin{pmatrix} A_{bb} & A_{br} \\ A_{rb} & A_{rr} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_{bb} & A_{br} \\ A_{rb} & A_{rr} \end{pmatrix},$$

the subscripts $b$ and $r$ correspond to the nodes on the $\Gamma_B$ boundary and to the rest of the nodes, respectively. Note that the matrix $B - A$ has the structure

$$B - A = \begin{pmatrix} B_{bb} - A_{bb} & 0 \\ 0 & 0 \end{pmatrix}.$$
The eigenvectors given by the generalized eigenvalue problems

\[ K_1 V = M_1 V \Lambda_A \quad \text{and} \quad K_1^B W = M_1^B W \Lambda_B \quad (10) \]

diagonalize \( K_1, K_1^B \) and \( M_1, M_1^B \). The matrices \( \Lambda_A \) and \( \Lambda_B \) contain the eigenvalues as diagonal entries and the matrices \( V \) and \( W \) contain the corresponding eigenvectors as their columns. The eigenvectors have to form a basis in \( \mathbb{C}^{n_1} \) in order to apply the partial solution method:

\[ V^T M_1 V = I_1 \quad \text{and} \quad V^T K_1 V = \Lambda_A, \quad (11) \]
\[ W^T M_1^B W = I_1 \quad \text{and} \quad W^T K_1^B W = \Lambda_B. \quad (12) \]

\( I_1 \) denotes the identity matrix of length \( n_1 \), whereas \( I_j \) and \( I_{jk} \) denote the identity matrices of lengths \( n_j \) and \( n_j \times n_k \). The eigenvalue problems (10) have to be solved only once during the solution process – in the initialization.
Fast solver

These conditions also lead to a convenient representation for the inverses of the system matrices $A$ and $B$. We obtain for the system matrix $B$ the following representation:

\[ B^{-1} = (W \otimes I_2) H_B^{-1}(W^T \otimes I_2), \]
\[ H_B = (\Lambda_B - \omega^2 I_1) \otimes M_2 + I_1 \otimes K_2 \]

for 2d as well as

\[ B^{-1} = (W \otimes I_{23}) H_B^{-1}(W^T \otimes I_{23}), \]
\[ H_B = ((\Lambda_B - \omega^2 I_1) \otimes M_2 + I_1 \otimes K_2) \otimes M_3 + I_1 \otimes M_2 \otimes K_3 \]

for 3d.

The representation for $A$ goes completely analogously.
Fast solver - efficient computation for the inverses of $H_B$ and $H_A$

2d: Compute the LU decomposition of $LU = H_B$. Instead of directly computing the inverse of $H_B$, solve the linear system

$$H_By = LUy = r$$

by solving the problems

$$Lz = r \quad \text{and} \quad Uy = z \quad \text{(13)}$$

iteratively (with $r$ being some right-hand side).

3d: Instead of solving the problems (13) with the block diagonal matrices $L$ and $U$, split them into $n_1$ subproblems of size $n_2 \times n_3$ corresponding to the respective diagonal blocks. Then solve the independent $n_1$ subproblems in the same iterative scheme (13) leading to a faster implementation for the 3d problem.
Fast solver - Step 1

Compute the Fourier transformation $\hat{f}$ of $f$ using FFT and save it, since it will be needed in Step 3 as well. Solve the auxiliary problem

$$Bv = B \begin{pmatrix} v_b \\ v_r \end{pmatrix} = f,$$  \hspace{1cm} (14)

but compute only $v_b$ and not $v_r$. Performing the inverse Fourier transformation would provide both $v_b$ and $v_r$. Instead of that, it is efficient to solve problem $H_B y = LUy = \hat{f}$ and then multiply the resulting vector by the eigenvectors of $W$ which correspond to the boundary $\Gamma_B$.

Recall, e.g., in 2d, $B^{-1} = (W \otimes I_2) H_B^{-1} (W^T \otimes I_2)$. 
Introduce an additional vector \( \mathbf{w} \) given by \( \mathbf{w} = \mathbf{u} - \mathbf{v} \), and solve the problem

\[
A\mathbf{w} = A\begin{pmatrix} \mathbf{w}_b \\ \mathbf{w}_r \end{pmatrix} = (B - A)\mathbf{v} = \begin{pmatrix} (B_{bb} - A_{bb}) \mathbf{v}_b \\ 0 \end{pmatrix},
\]

(15)
since

\[
A\mathbf{w} = A\mathbf{u} - A\mathbf{v} = f - A\mathbf{v} = B\mathbf{v} - A\mathbf{v},
\]

(16)
but compute only \( \mathbf{w}_b \) and not \( \mathbf{w}_r \). Problem (15) can be solved efficiently using the diagonalization with multiplications by the eigenvector matrices \( \mathbf{V}^T \) and \( \mathbf{V} \) of \( A \) corresponding to the boundary \( \Gamma_B \).
Solve now the problem

\[ Bu = f + (B - A)(v + w) \]

\[ = f + \begin{pmatrix} (B_{bb} - A_{bb})(v_b + w_b) \\ 0 \end{pmatrix} \]  \hspace{1cm} (17)

due to \( Bu = Au + Bu - Au \). Use the Fourier transformation \( \hat{f} \) of \( f \) from Step 1. The Fourier transformation of the second term

\[ g = (B - A)(v + w) = \begin{pmatrix} (B_{bb} - A_{bb})(v_b + w_b) \\ 0 \end{pmatrix} \]  \hspace{1cm} (18)

can be performed efficiently by multiplying it by \( W^T \) as the term (18) is sparse again.

Finally, solve \( H_B y = LU y = \hat{f} + \hat{g} \) and then perform the inverse FFT on the resulting vector leading to the solution \( u \) of the original problem \( Au = f \).
Numerical results

The numerical experiments have been computed in Matlab.

$$\Omega = [0, 1]^d, \omega = 2\pi, \text{uniform meshes wrt each } x_j, \text{ step size } h = 1/n$$

Right-hand side is chosen as 0.01 for the first $n$ entries and 1 for all the other entries.

We compare the CPU times in seconds for computing the solution by applying Matlab’s backslash, the fast solver presented as well as second version of it, where in step 1 and 3 we use the FFT instead of the multiplications by $W$. We present the times for the computations in the initialization process as well.
Numerical results (2d)

CPU times in seconds:

<table>
<thead>
<tr>
<th>$n_j$</th>
<th>65</th>
<th>129</th>
<th>257</th>
<th>513</th>
<th>1025</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>0.11</td>
<td>0.26</td>
<td>0.82</td>
<td>3.93</td>
<td>22.92</td>
</tr>
<tr>
<td>Matlab’s backslash</td>
<td>0.04</td>
<td>0.16</td>
<td>0.52</td>
<td>3.17</td>
<td>13.11</td>
</tr>
<tr>
<td>Fast Solver</td>
<td>0.02</td>
<td>0.05</td>
<td>0.13</td>
<td>0.47</td>
<td>2.09</td>
</tr>
<tr>
<td>Fast Solver version 2</td>
<td>0.01</td>
<td>0.03</td>
<td>0.07</td>
<td>0.37</td>
<td>1.74</td>
</tr>
</tbody>
</table>
Numerical results (3d)

CPU times in seconds:

<table>
<thead>
<tr>
<th>$n_j$</th>
<th>9</th>
<th>17</th>
<th>33</th>
<th>65</th>
<th>129</th>
<th>257</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>0.09</td>
<td>0.13</td>
<td>0.32</td>
<td>1.95</td>
<td>15.32</td>
<td>152.28</td>
</tr>
<tr>
<td>Matlab’s backslash</td>
<td>0.06</td>
<td>0.31</td>
<td>7.33</td>
<td>671.77</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Fast Solver</td>
<td>0.06</td>
<td>0.13</td>
<td>0.53</td>
<td>4.64</td>
<td>63.94</td>
<td>631.39</td>
</tr>
<tr>
<td>Fast Solver version 2</td>
<td>0.08</td>
<td>0.12</td>
<td>0.51</td>
<td>4.45</td>
<td>61.41</td>
<td>647.46</td>
</tr>
</tbody>
</table>
Conclusions and outlook

Efficient numerical method employing FFT combined with a fast direct solver for the Helmholtz problem with ABCs. Solving the Helmholtz equation is in general difficult or impossible to solve efficiently with most numerical methods.


Outlook:

- More complicated domains and data
- Combine this solver with domain decomposition methods for layered media
- Parallel implementation

Ευχαριστώ!