

A fast Fourier transform based direct solver for the Helmholtz problem

Jari Toivanen Monika Wolfmayr



AANMPDE 11
Särkisaari, August 6, 2018

Content

- 1 Model problem
- 2 Discretization
- 3 Fast solver
 - in the 2d case
 - in the 3d case
- 4 Numerical results
- 5 Conclusions and outlook

Model problem

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a d -dimensional rectangular domain. The pressure field satisfies the **Helmholtz** partial differential equation

$$-\Delta u - \omega^2 u = f \quad \text{in } \Omega, \quad (1)$$

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

where ω denotes the wave number. The boundary $\Gamma = \partial\Omega = \Gamma_N \cup \Gamma_B$ is decomposed into Neumann boundary condition (BC) Γ_N and (first-order) **absorbing BC (ABC)** Γ_B :

$$\mathcal{B}u = \nabla u \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_N, \quad (3)$$

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

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

$$a(u, v) = \int_{\Omega} f v \, d\mathbf{x} \quad \forall v \in V, \quad (5)$$

where

$$a(u, v) = \int_{\Omega} (\nabla u \cdot \nabla v - \omega^2 uv) \, d\mathbf{x} - i\omega \int_{\partial\Omega} uv \, ds. \quad (6)$$

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

$$\mathbf{A} \mathbf{u} = \mathbf{f}, \quad (7)$$

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

Discretization

For the two-dimensional (2d) case, the matrix \mathbf{A} is given by

$$\mathbf{A} = (\mathbf{K}_1 - \omega^2 \mathbf{M}_1) \otimes \mathbf{M}_2 + \mathbf{M}_1 \otimes \mathbf{K}_2,$$

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

$$\mathbf{A} = (\mathbf{K}_1 - \omega^2 \mathbf{M}_1) \otimes \mathbf{M}_2 \otimes \mathbf{M}_3 + \mathbf{M}_1 \otimes (\mathbf{K}_2 \otimes \mathbf{M}_3 + \mathbf{M}_2 \otimes \mathbf{K}_3),$$

where \mathbf{K}_j and \mathbf{M}_j are one-dimensional (1d) stiffness and mass matrices, respectively, in the x_j -direction with possible modifications on or near the boundaries due to the ABC.

Discretization

\mathbf{K}_j and \mathbf{M}_j are computed by 1d numerical quadrature on $[0, 1]$:

$$\mathbf{K}_j = \frac{1}{h_j} \begin{pmatrix} k_{1,1} & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & k_{n_j, n_j} \end{pmatrix}, \quad \mathbf{M}_j = \frac{h_j}{6} \begin{pmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 2 \end{pmatrix}$$

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 $k_{1,1} = k_{n_j, n_j} = 1$.

\mathbf{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

Fast solver - idea

The main idea for solving the problem $\mathbf{A}\mathbf{u} = \mathbf{f}$ is to consider an **auxiliary problem** $\mathbf{B}\mathbf{v} = \mathbf{f}$, where the system matrix \mathbf{B} is derived by changing the ABCs to **periodic** ones.

The key is that we can solve the modified (periodic) problem $\mathbf{B}\mathbf{v} = \mathbf{f}$ now by using the FFT method, which is not possible for the original problem $\mathbf{A}\mathbf{u} = \mathbf{f}$.

The problem $\mathbf{A}\mathbf{u} = \mathbf{f}$ can be solved applying the following steps:

1. Solve $\mathbf{B}\mathbf{v} = \mathbf{f}$.
2. Solve $\mathbf{A}\mathbf{w} = \mathbf{f} - \mathbf{A}\mathbf{v} = \mathbf{B}\mathbf{v} - \mathbf{A}\mathbf{v} = (\mathbf{B} - \mathbf{A})\mathbf{v}$, $\mathbf{u} = \mathbf{v} + \mathbf{w}$.
3. Solve $\mathbf{B}\mathbf{u} = \mathbf{f} + (\mathbf{B} - \mathbf{A})(\mathbf{v} + \mathbf{w})$.

Fast solver - the auxiliary problem

In case of periodic BCs in x_1 -direction, the matrices \mathbf{K}_1 and \mathbf{M}_1 change to \mathbf{K}_1^B and \mathbf{M}_1^B and are given by

$$\mathbf{K}_1^B = \frac{1}{h_1} \begin{pmatrix} 2 & -1 & & -1 \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ -1 & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad \mathbf{M}_1^B = \frac{h_1}{6} \begin{pmatrix} 4 & 1 & & & 1 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ 1 & & & & 1 & 4 \end{pmatrix},$$

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

$$\mathbf{B} = (\mathbf{K}_1^B - \omega^2 \mathbf{M}_1^B) \otimes \mathbf{M}_2 + \mathbf{M}_1^B \otimes \mathbf{K}_2 \quad (2d),$$

$$\mathbf{B} = (\mathbf{K}_1^B - \omega^2 \mathbf{M}_1^B) \otimes \mathbf{M}_2 \otimes \mathbf{M}_3 + \mathbf{M}_1^B \otimes (\mathbf{K}_2 \otimes \mathbf{M}_3 + \mathbf{M}_2 \otimes \mathbf{K}_3) \quad (3d).$$

Fast solver

After a suitable permutation \mathbf{A} and \mathbf{B} have the block forms

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{bb} & \mathbf{A}_{br} \\ \mathbf{A}_{rb} & \mathbf{A}_{rr} \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} \mathbf{B}_{bb} & \mathbf{A}_{br} \\ \mathbf{A}_{rb} & \mathbf{A}_{rr} \end{pmatrix}, \quad (8)$$

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

$$\mathbf{B} - \mathbf{A} = \begin{pmatrix} \mathbf{B}_{bb} - \mathbf{A}_{bb} & 0 \\ 0 & 0 \end{pmatrix} \quad (9)$$

and only the matrix

$$\mathbf{C}_{bb} = \mathbf{B}_{bb} - \mathbf{A}_{bb} \quad (10)$$

has to be saved for the application of the fast solver.

Fast solver - some ideas behind applying the partial solution method

The eigenvectors given by the generalized eigenvalue problems

$$\mathbf{K}_1 \mathbf{V}_1 = \mathbf{M}_1 \mathbf{V}_1 \mathbf{\Lambda}_1^A \quad \text{and} \quad \mathbf{K}_1^B \mathbf{W}_1 = \mathbf{M}_1^B \mathbf{W}_1 \mathbf{\Lambda}_1^B \quad (11)$$

diagonalize \mathbf{K}_1 , \mathbf{K}_1^B and \mathbf{M}_1 , \mathbf{M}_1^B .

The matrices $\mathbf{\Lambda}_1^A$ and $\mathbf{\Lambda}_1^B$ contain the eigenvalues as diagonal entries and the matrices \mathbf{V}_1 and \mathbf{W}_1 contain the corresponding eigenvectors as their columns. The eigenvectors have to be normalized in order to apply the **partial solution method**:

$$\mathbf{V}_1^T \mathbf{M}_1 \mathbf{V}_1 = \mathbf{I}_1 \quad \text{and} \quad \mathbf{V}_1^T \mathbf{K}_1 \mathbf{V}_1 = \mathbf{\Lambda}_1^A, \quad (12)$$

$$\mathbf{W}_1^T \mathbf{M}_1^B \mathbf{W}_1 = \mathbf{I}_1 \quad \text{and} \quad \mathbf{W}_1^T \mathbf{K}_1^B \mathbf{W}_1 = \mathbf{\Lambda}_1^B. \quad (13)$$

\mathbf{I}_1 denotes the identity matrix of length n_1 , whereas \mathbf{I}_j and \mathbf{I}_{jk} denote the identity matrices of lengths n_j and $n_j \times n_k$.

The eigenvalue problems (11) have to be solved only once during the solution process – in the initialization.

The 2d case

Fast solver (2d) - Reformulation of \mathbf{A} and \mathbf{B}

These conditions also lead to a convenient representation for the inverses of the system matrices \mathbf{A} and \mathbf{B} . Using (13) as follows

$$\begin{aligned}\mathbf{B} &= (\mathbf{W}_1^{-T} \boldsymbol{\Lambda}_1^B \mathbf{W}_1^{-1} - \omega^2 \mathbf{W}_1^{-T} \mathbf{I}_1 \mathbf{W}_1^{-1}) \otimes \mathbf{M}_2 + (\mathbf{W}_1^{-T} \mathbf{I}_1 \mathbf{W}_1^{-1}) \otimes \mathbf{K}_2 \\ &= (\mathbf{W}_1^{-T} (\boldsymbol{\Lambda}_1^B - \omega^2 \mathbf{I}_1) \mathbf{W}_1^{-1}) \otimes \mathbf{M}_2 + (\mathbf{W}_1^{-T} \mathbf{I}_1 \mathbf{W}_1^{-1}) \otimes \mathbf{K}_2 \\ &= (\mathbf{W}_1^{-T} \otimes \mathbf{I}_2) ((\boldsymbol{\Lambda}_1^B - \omega^2 \mathbf{I}_1) \otimes \mathbf{M}_2 + \mathbf{I}_1 \otimes \mathbf{K}_2) (\mathbf{W}_1^{-1} \otimes \mathbf{I}_2),\end{aligned}$$

the inverse of \mathbf{B} can be represented by

$$\begin{aligned}\mathbf{B}^{-1} &= (\mathbf{W}_1 \otimes \mathbf{I}_2) \mathbf{H}_B^{-1} (\mathbf{W}_1^T \otimes \mathbf{I}_2), \\ \mathbf{H}_B &= (\boldsymbol{\Lambda}_1^B - \omega^2 \mathbf{I}_1) \otimes \mathbf{M}_2 + \mathbf{I}_1 \otimes \mathbf{K}_2.\end{aligned}\tag{14}$$

Similarly using (12), we obtain

$$\begin{aligned}\mathbf{A}^{-1} &= (\mathbf{V}_1 \otimes \mathbf{I}_2) \mathbf{H}_A^{-1} (\mathbf{V}_1^T \otimes \mathbf{I}_2), \\ \mathbf{H}_A &= (\boldsymbol{\Lambda}_1^A - \omega^2 \mathbf{I}_1) \otimes \mathbf{M}_2 + \mathbf{I}_1 \otimes \mathbf{K}_2.\end{aligned}\tag{15}$$



Fast solver (2d) - LU decomposition

The LU decomposition of \mathbf{H}_B and \mathbf{H}_A are computed as follows

$$\mathbf{H}_B = \mathbf{L}_B \mathbf{U}_B \quad \text{and} \quad \mathbf{H}_A = \mathbf{L}_A \mathbf{U}_A. \quad (16)$$

Then the linear system

$$\mathbf{H}_B \mathbf{y} = \mathbf{L}_B \mathbf{U}_B \mathbf{y} = \mathbf{r} \quad (17)$$

is solved by solving the respective two subproblems

$$\mathbf{L}_B \mathbf{z} = \mathbf{r} \quad \text{and} \quad \mathbf{U}_B \mathbf{y} = \mathbf{z} \quad (18)$$

consecutively in the application of the fast solver (analogously for \mathbf{H}_A). The vector \mathbf{r} denotes some right hand side which is in the different steps of the solver also different.

The structure of \mathbf{H}_B and \mathbf{H}_A is essential for the fast application of the solver: The diagonal blocks are tridiagonal which makes the LU decomposition fast.

Computational complexity is optimal $\mathcal{O}(N)$.

Fast solver (2d) - Step 1

Solve the auxiliary problem

$$\mathbf{B}\mathbf{v} = \mathbf{B} \begin{pmatrix} \mathbf{v}_b \\ \mathbf{v}_r \end{pmatrix} = \mathbf{f}, \quad (19)$$

but compute only \mathbf{v}_b and not \mathbf{v}_r .

(a) Compute the Fourier transformation $\hat{\mathbf{f}}$ of \mathbf{f} using FFT (which corresponds to $\hat{\mathbf{f}} = (\mathbf{W}_1^T \otimes \mathbf{I}_2) \mathbf{f}$), where its coefficients \hat{f}_k are given as follows

$$\hat{f}_k = \sum_{l=1}^N e^{-2\pi i \frac{(l-1)(k-1)}{N}} f_l \quad \forall k = 1, \dots, N, \quad (20)$$

and save it, since it will be needed in Step 3 as well.

(b) Next, we apply the LU decomposition (17) with the right-hand side $\hat{\mathbf{f}}$

$$\mathbf{L}_B \mathbf{z}_1 = \hat{\mathbf{f}} \quad \text{and} \quad \mathbf{U}_B \tilde{\mathbf{z}}_1 = \mathbf{z}_1. \quad (21)$$

Fast solver (2d) - Step 1

(c) Performing the inverse FFT on the resulting vector $\tilde{\mathbf{z}}_1$ would provide both \mathbf{v}_b and \mathbf{v}_r , which would correspond to $\mathbf{v} = (\mathbf{W}_1 \otimes \mathbf{I}_2) \tilde{\mathbf{z}}_1$. Since we only need \mathbf{v}_b , instead of that, we multiply the vector $\tilde{\mathbf{z}}_1$ by the matrix \mathbf{W}_1^b (the eigenvectors of \mathbf{W}_1 which correspond only to the boundary Γ_B): leading to

$$\mathbf{v}_b = (\mathbf{W}_1^b \otimes \mathbf{I}_2) \tilde{\mathbf{z}}_1, \quad (22)$$

which altogether resembles the representation (14) for \mathbf{B}^{-1} .
Computational complexity of step 1 is $\mathcal{O}(N \log N)$.

Fast solver (2d) - Step 2

Introduce an additional vector \mathbf{w} given by $\mathbf{w} = \mathbf{u} - \mathbf{v}$, and solve the problem

$$\mathbf{A}\mathbf{w} = \mathbf{A} \begin{pmatrix} \mathbf{w}_b \\ \mathbf{w}_r \end{pmatrix} = (\mathbf{B} - \mathbf{A})\mathbf{v} = \begin{pmatrix} \mathbf{C}_{bb}\mathbf{v}_b \\ 0 \end{pmatrix}, \quad (23)$$

since

$$\mathbf{A}\mathbf{w} = \mathbf{A}\mathbf{u} - \mathbf{A}\mathbf{v} = \mathbf{f} - \mathbf{A}\mathbf{v} = \mathbf{B}\mathbf{v} - \mathbf{A}\mathbf{v}, \quad (24)$$

but compute only \mathbf{w}_b and not \mathbf{w}_r again:

$$(a) \quad \mathbf{g}_b = (\mathbf{V}_1^{bT} \otimes \mathbf{I}_2) \mathbf{C}_{bb} \mathbf{v}_b, \quad (25)$$

$$(b) \quad \mathbf{L}_A \mathbf{z}_2 = \mathbf{g}_b \quad \text{and} \quad \mathbf{U}_A \tilde{\mathbf{z}}_2 = \mathbf{z}_2, \quad (26)$$

$$(c) \quad \mathbf{w}_b = (\mathbf{V}_1^b \otimes \mathbf{I}_2) \tilde{\mathbf{z}}_2. \quad (27)$$

Computational complexity of step 2 is optimal $\mathcal{O}(N)$.

Fast solver (2d) - Step 3

Solve now the problem

$$\begin{aligned} \mathbf{B}\mathbf{u} &= \mathbf{f} + (\mathbf{B} - \mathbf{A})(\mathbf{v} + \mathbf{w}) \\ &= \mathbf{f} + \begin{pmatrix} \mathbf{C}_{bb}(\mathbf{v}_b + \mathbf{w}_b) \\ 0 \end{pmatrix} \end{aligned} \quad (28)$$

due to $\mathbf{B}\mathbf{u} = \mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{u} - \mathbf{A}\mathbf{u}$.

(a) Use the Fourier transformation $\hat{\mathbf{f}}$ of \mathbf{f} from step 1. Only need the Fourier transformation of the second term computed by

$$\mathbf{h}_b = (\mathbf{W}_1^{bT} \otimes \mathbf{I}_2) \mathbf{C}_{bb}(\mathbf{v}_b + \mathbf{w}_b) \quad (29)$$

leading to the Fourier transformation of the entire right-hand side of equation (28) denoted by $\hat{\mathbf{f}} + \hat{\mathbf{h}}$.

Fast solver (2d) - Step 3

$$(b) \quad \mathbf{L}_B \mathbf{z}_3 = \hat{\mathbf{f}} + \hat{\mathbf{h}} \quad \text{and} \quad \mathbf{U}_B \tilde{\mathbf{z}}_3 = \mathbf{z}_3. \quad (30)$$

(c) In the last step all resulting components are needed (not only the ones corresponding to Γ_B) to obtain the solution \mathbf{u} by applying the inverse Fourier transformation on $\tilde{\mathbf{z}}_3$, where its coefficients \hat{u}_k are given as follows

$$\hat{u}_k = \frac{1}{N} \sum_{l=1}^N e^{2\pi i \frac{(l-1)(k-1)}{N}} \tilde{z}_{3,l} \quad \forall k = 1, \dots, N. \quad (31)$$

The computation of the inverse Fourier transformation corresponds to the multiplication

$$\mathbf{u} = (\mathbf{W}_1 \otimes \mathbf{I}_2) \tilde{\mathbf{z}}_3. \quad (32)$$

Computational complexity of step 3 is $\mathcal{O}(N \log N)$.

The 3d case

Fast solver (3d) - Reformulation of \mathbf{A} and \mathbf{B}

$$\mathbf{B} = (\mathbf{K}_1^B - \omega^2 \mathbf{M}_1^B) \otimes \mathbf{M}_2 \otimes \mathbf{M}_3 + \mathbf{M}_1^B \otimes (\mathbf{K}_2 \otimes \mathbf{M}_3 + \mathbf{M}_2 \otimes \mathbf{K}_3) \quad (3d).$$

Analogously as in the 2d case

$$\mathbf{A}^{-1} = (\mathbf{V}_1 \otimes \mathbf{I}_{23}) \mathbf{H}_A^{-1} (\mathbf{V}_1^T \otimes \mathbf{I}_{23}) \quad (33)$$

and

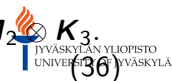
$$\mathbf{B}^{-1} = (\mathbf{W}_1 \otimes \mathbf{I}_{23}) \mathbf{H}_B^{-1} (\mathbf{W}_1^T \otimes \mathbf{I}_{23}), \quad (34)$$

where

$$\mathbf{H}_A = ((\mathbf{\Lambda}_1^A - \omega^2 \mathbf{I}_1) \otimes \mathbf{M}_2 + \mathbf{I}_1 \otimes \mathbf{K}_2) \otimes \mathbf{M}_3 + \mathbf{I}_1 \otimes \mathbf{M}_2 \otimes \mathbf{K}_3 \quad (35)$$

and

$$\mathbf{H}_B = ((\mathbf{\Lambda}_1^B - \omega^2 \mathbf{I}_1) \otimes \mathbf{M}_2 + \mathbf{I}_1 \otimes \mathbf{K}_2) \otimes \mathbf{M}_3 + \mathbf{I}_1 \otimes \mathbf{M}_2 \otimes \mathbf{K}_3. \quad (36)$$



Fast solver (3d) - Applying the 2d fast solver instead of LU decomposition

Since the LU decomposition is slow for very large problems in 3d, the linear systems with the block diagonal matrices \mathbf{H}_B and \mathbf{H}_A are efficiently implemented by applying the 2d fast solver n_1 times for 2d subproblems of size $n_2 \times n_3$ including the computation of the partial solution method in x_2 -direction for

$$\mathbf{A}_{A,l} = (\mathbf{K}_2 - \underbrace{(\omega^2 - \mathbf{\Lambda}_{1,l}^A)}_{=:p_A} \mathbf{M}_2) \otimes \mathbf{M}_3 + \mathbf{M}_2 \otimes \mathbf{K}_3 \quad (37)$$

$$\mathbf{A}_{B,l} = (\mathbf{K}_2 - \underbrace{(\omega^2 - \mathbf{\Lambda}_{1,l}^B)}_{=:p_B} \mathbf{M}_2) \otimes \mathbf{M}_3 + \mathbf{M}_2 \otimes \mathbf{K}_3, \quad (38)$$

where $l = 1, \dots, n_1$, solving the generalized eigenvalue problems

$$\mathbf{K}_2 \mathbf{V}_2 = \mathbf{M}_2 \mathbf{V}_2 \mathbf{\Lambda}_2^A \quad \text{and} \quad \mathbf{K}_2^B \mathbf{W}_2 = \mathbf{M}_2^B \mathbf{W}_2 \mathbf{\Lambda}_2^B \quad (39)$$

Computational complexity is now $\mathcal{O}(N \log N)$.

Numerical results

Numerical results

The numerical experiments have been computed in MATLAB 9.3, R2017b, on a laptop with Intel(R) Core(TM) i5-6267U CPU @ 2.90GHz processor and 16 GB 2133 MHz LPDDR3 memory.

$\Omega = [0, 1]^d$, $\omega = 2\pi$, uniform meshes wrt each x_j , step size $h_j = 1/n_j$

Right-hand side is chosen as 0.01 for the first n_1 entries and 1 for all the other entries

We **compare** the CPU times in seconds for computing the solution by applying **Matlab's backslash** and the **fast solver** presented. In our case, Matlab's backslash uses the sparse direct solver UMFPACK for computing the solution of the sparse linear systems. We present the times for the computations in the initialization process as well.

Numerical results (2d)

CPU times in seconds for different values of $n = n_1 = n_2$:

n	65	129	257	513	1025	2049
Initialization	0.07	0.09	0.31	1.74	13.38	118.99
Matlab's backslash	0.04	0.14	0.50	2.64	12.42	93.32
Fast solver	0.01	0.02	0.06	0.23	1.05	4.58

Largest numerical experiments in 2d have been computed for $n = 2049$ with 4 198 401 unknowns.

Numerical results (2d)

CPU times in seconds for different combinations of values for n_1 and n_2 :

	n_1	65	65	2049	2049
	n_2	65	2049	65	2049
Initialization		0.07	0.17	110.83	118.99
Matlab's backslash		0.04	0.98	1.07	93.32
Fast solver		0.01	0.10	0.12	4.58

Numerical results (3d)

CPU times in seconds for different values of $n = n_1 = n_2 = n_3$:

n	9	17	33	65	129	257	513
Initialization	0.07	0.07	0.08	0.45	0.41	2.16	17.33
Matlab's backslash	0.02	0.28	6.39	572.91	–	–	–
Fast solver	0.09	0.12	0.21	1.03	8.43	69.55	672.27

Largest numerical experiments in 3d have been computed for $n = 513$ with 135 005 697 unknowns.

Numerical results (3d)

CPU times in seconds for different combinations of values for n_1 , n_2 and n_3 :




n_1	9	9	9	513	9	513	513	513
n_2	9	513	9	9	513	9	513	513
n_3	9	9	513	9	513	513	9	513
Initialization	0.07	0.36	0.12	1.91	2.17	2.20	2.40	17.33
Matlab's backslash	0.02	1.06	0.96	0.90	–	–	–	–
Fast solver	0.09	0.22	0.25	0.62	10.95	8.00	7.01	672.27

The larger the size of the problem in 3d the more efficient the FFT based fast direct solver.

For large n , applying Matlab's `lu` on the subproblems as well as Matlab's `kron` are most time consuming.

⇒ Can be expected that e.g. C++ implementation would be essentially faster.

References

-  T. Rossi and J. Toivanen, *A nonstandard cyclic reduction method, its variants and stability*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 628–645.
-  E. Heikkola, T. Rossi, and J. Toivanen, *Fast direct solution of the Helmholtz equation with a perfectly matched layer or an absorbing boundary condition*, Internat. J. Numer. Methods Engrg., 57 (2003), pp. 2007–2025.
-  J. Toivanen and M. Wolfmayr, *A Fast Fourier Transform based direct solver for the Helmholtz problem*, 2018.

Conclusions and outlook

Conclusions:

- **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:

- Implementation in C++
- Application also to other problems
- ...

Thank you!