CHALMERS
FINITE ELEMENT CENTER

PREPRINT 2001–14

A hybrid method for the wave equation

Larisa Beilina, Klas Samuelsson, Krister Åhlander

Chalmers Finite Element Center
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg Sweden 2001

CHALMERS FINITE ELEMENT CENTER

Preprint 2001–14

A hybrid method for the wave equation

Larisa Beilina, Klas Samuelsson, Krister Åhlander

CHALMERS

Chalmers Finite Element Center
Chalmers University of Technology
SE–412 96 Göteborg Sweden
Göteborg, July 2001

A hybrid method for the wave equation
Larisa Beilina, Klas Samuelsson, Krister Åhlander
NO 2001–14
ISSN 1404–4382

Chalmers Finite Element Center
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone: +46 (0)31 772 1000
Fax: +46 (0)31 772 3595
`www.phi.chalmers.se`

Printed in Sweden
Chalmers University of Technology
Göteborg, Sweden 2001

A HYBRID METHOD FOR THE WAVE EQUATION

LARISA BEILINA, KLAS SAMUELSSON, AND KRISTER ÅHLANDER

Abstract. Hybrid finite element/finite difference simulation of the wave equation is studied. The simulation method is hybrid in the sense that different numerical methods, finite elements and finite differences, are used in different subdomains. The purpose is to combine the flexibility of finite elements with the efficiency of finite differences.

The construction of proper geometry discretisations is important for the hybrid approach. A decomposition of the computational domain is described, which yields simple communication between structured and unstructured subdomains.

An explicit hybrid method for the wave equation is constructed where the explicit finite difference schemes and finite element schemes coincide for structured subdomains. These schemes are used in the hybrid approach, keeping finite differences on the structured subdomains and applying finite elements on the unstructured domains. As a consequence of the discretisation strategy, the resulting hybrid scheme can be regarded as a pure finite element scheme. Any numerical difficulties such as instabilities at the interfaces are thus avoided.

The feasibility of the hybrid approach is illustrated by numerous wave equation simulations in two and three space dimensions. In particular, the approach can easily be used for implementing absorbing boundary conditions.

The efficiency of different approaches is a key issue of the current study. For our test cases, the hybrid approach is about 5 times faster than a corresponding highly optimised finite element method. It is concluded that the hybrid approach may be an important tool to reduce the execution time and memory requirement for this kind of large scale computations.

1. Introduction

When simulating partial differential equations in three space dimensions, it is important to use efficient implementation strategies, in particular with respect to memory usage. This motivates our research on *hybrid* methods, which combine the flexibility of the finite element method with the efficiency of the finite difference method. Finite differences are used where the geometry is simple, and finite elements are used where the geometry is more complex. Hybrid approaches has also been studied in [4, 5].

Date: 2nd July 2001.

Larisa Beilina, Department of Mathematics, Chalmers University of Technology, S–412 96 Göteborg, Sweden, *email*: larisa@math.chalmers.se

Klas Samuelsson, Department of Mathematics, Chalmers University of Technology, S–412 96 Göteborg, Sweden, *email*: klassam@math.chalmers.se

Krister Åhlander, Department of Scientific Computing, Uppsala University, Uppsala, Sweden. Currently at the Department of Informatics, University of Bergen, Bergen, Norway, *email*: krister@tdb.uu.se.

In order to evaluate our hybrid approach, we simulate the wave equation in two and three dimensions. The computational domains for this kind of problems often exhibit large regions where the geometry is simple, and small regions where the geometry is complex. This suggests that a hybrid approach might be beneficial. In this paper, we discretise the scalar wave equation in time with explicit methods. Both absorbing boundary conditions and Dirichlet boundary conditions are used.

The finite difference stencil is constructed by applying the finite element method with node based quadrature rules on a structured Cartesian grid which cells have regularly subdivided into triangles and tetrahedra, in two and three space dimensions, respectively. By this construction will the proposed hybrid method and the finite element method give identical results, which is validated in numerical experiments of our implementation.

A main concern for hybrid methods is the possible instability of the method at interfaces between regions with different methods. In our case this issue is resolved by the following observation: Since the hybrid computations are identical to the finite element method, the stability of the hybrid method is determined by the stability of the underlying finite element method.

Another important research topic for our project is to implement appropriate software for the hybrid method. We present a set of C++ classes, developed in order to handle space discretisations that consist of both structured, Cartesian, grids, to be used with finite differences, and unstructured grids, for usage with finite elements. Techniques supporting the necessary communication between subdomains are also presented.

When comparing our hybrid approach with a pure finite element method, our results show that the hybrid method is faster, in particular for problems where the memory demands are high.

We conclude that our hybrid approach is especially useful for large problems where the computational domain consists of big, simple, regions, where Cartesian grids can be used, together with relatively small regions, where the geometry is more complex and unstructured grids have to be used.

2. THE MATHEMATICAL MODEL

The model problem is the wave equation

$$\begin{aligned}
\frac{\partial^2 u}{\partial t^2} &= \nabla \cdot (a^2 \nabla u) + f, \qquad x \in \Omega \subset \mathbf{R}^3,\ t > 0, && (2.1)\\
u(x,0) &= 0, \quad x \in \Omega, && (2.2)\\
\frac{\partial}{\partial t} u(x,0) &= 0, \quad x \in \Omega, && (2.3)\\
u &= 0, \quad x \in \Gamma,\ t > 0 && (2.4)
\end{aligned}$$

where $u(x,t)$ is unknown, a is the wave velocity, x ranges over the points of the space domain, t over the time, f is the source function, and Γ denotes the boundary of the domain Ω. We can rewrite this second order equation as a system of first order equations

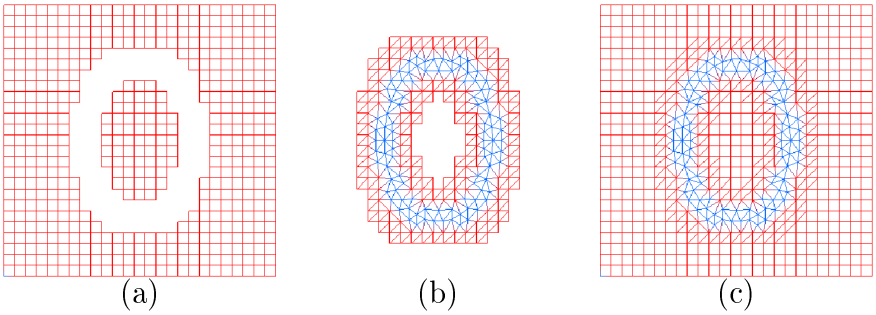


FIGURE 1. Domain decomposition. The hybrid mesh (c) is a combination of the structured mesh (a) and the unstructured mesh (b) with a thin overlapping of structured elements. The unstructured grid is constructed so that the grid contains edges approximating an ellipse.

in time using the substitution $u_1 = \frac{\partial}{\partial t} u$ and $u_2 = u$, thus obtaining

$$\frac{\partial}{\partial t} u_1 - \nabla \cdot (a^2 \nabla u_2) = f, \quad x \in \Omega, \tag{2.5}$$

$$\frac{\partial}{\partial t} u_2 = u_1, \quad x \in \Omega, \tag{2.6}$$

$$u_1(x,0) = u_2(x,0) = 0, \quad x \in \Omega. \tag{2.7}$$

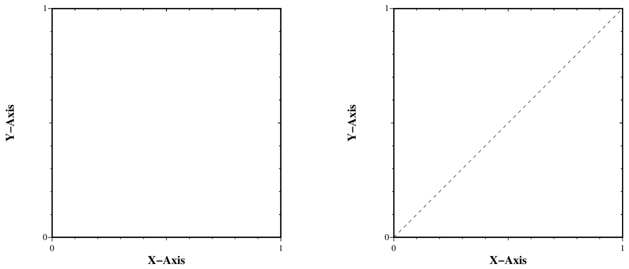
In many wave equation applications, only a small part of the computational domain Ω is complex enough to motivate a more complex unstructured discretisation, whereas quite large regions of the computational domain are sufficiently discretised with simple, Cartesian grids. For our exposition, our model domain consists of two regions, Ω_{FEM} and Ω_{FDM}, not necessarily simply connected. In the relatively small Ω_{FEM} domain, we assume that an unstructured discretisation is appropriate. In the Ω_{FDM} domain, we assume that a structured, Cartesian, grid is suitable. Fig. 1 illustrates the principle in two dimensions. Our three-dimensional geometries are built up similarly. The FEM grid is generated such that the thin overlapping domain consists of simplexes obtained by splitting the structured cells as described in Fig. 2. In the interior part of the FEM grid the discretisation is allowed to be truly unstructured.

In most of our test cases, we have used Dirichlet boundary conditions. We have also used an absorbing boundary condition, taken from [6]. At a boundary Γ, we then use $\frac{\partial}{\partial t} u - \frac{\partial}{\partial n} u|_\Gamma = 0$, where $\partial / \partial n$ is the normal derivative.

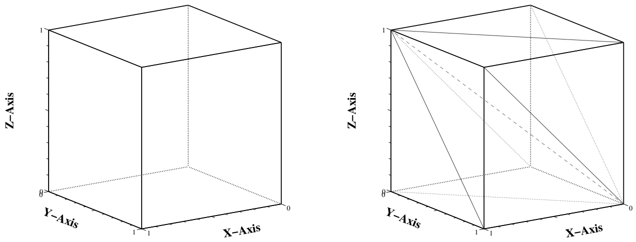
3. THE NUMERICAL METHOD

With a hybrid discretisation of the computational domain as described in the previous section, we are now in a position to formulate our hybrid algorithm. We observe that the interior nodes of the computational domain belong to either of the following sets:

ω_o: : Nodes interior to Ω_{FDM} and boundary nodes to Ω_{FEM},

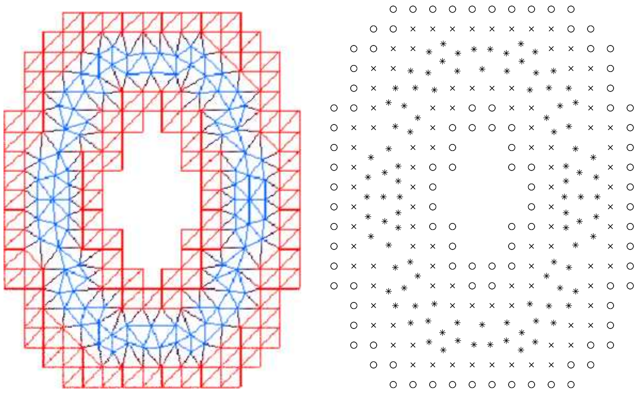


(a) A quadrilateral, split into two triangles.



(b) A hexahedron, split into six tetrahedra.

FIGURE 2. In the overlapping domain the finite element grid is created by splitting the structured cells into simplexes as depicted in (a) and (b) for 2D and 3D, respectively.



(a) (b)

FIGURE 3. Coupling of FEM and FDM. The nodes of the unstructured FEM grid of (a) is shown in (b), where rings and crosses are nodes which are shared between the FEM and FDM grids. The remaining nodes are marked with stars. The ring nodes are interior to the FDM grid, while the nodes crosses are interior to the FEM grid. At each time iteration, the FDM solution values at ring nodes are copied to the corresponding FEM solution values. At the same time at cross nodes the FEM solution values are copied to the FDM solution values.

$\omega_\times$: : Nodes interior to Ω_{FEM} and boundary nodes to Ω_{FDM},
ω_*: : Nodes interior to Ω_{FEM} and not contained in Ω_{FDM},
ω_D: : Nodes interior to Ω_{FDM} and not contained in Ω_{FEM}.

Fig. 3 illustrates the situation for a two-dimensional domain where some nodes are confined to an ellipse, which requires an unstructured discretisation. The exterior and the interior of the ellipse may use a structured discretisation. For clarity, nodes belonging to Ω_D are not shown.

In our algorithm, we store nodes belonging to ω_o and $\Omega_\times$ twice, both as nodes belonging to Ω_{FEM} and Ω_{FDM}. For explicit time stepping schemes, the main loop of the simulation can be sketched:

For every time step
(1) Update the solution in the interior of Ω_{FDM}, i.e. at nodes ω_D and ω_o using FDM
(2) Update the solution in the interior of Ω_{FEM}, i.e. at nodes ω_* and $\omega_\times$ using FEM
(3) Copy values at nodes $\omega_\times$ from Ω_{FEM} to Ω_{FDM}
(4) Copy values at nodes ω_o from Ω_{FDM} to Ω_{FEM}

The FEM and the FDM schemes that we use are well-known. For the sake of completeness, we present them below, and we also point out that we can regard the FDM sheme as a reformulation of the FEM scheme for a structured grid. Therefore, our hybrid approach can be analysed as a pure FEM scheme.

3.1. **Finite element formulation.** In the Ω_{FEM} domain where an unstructured grid is assumed, we use FEM. To formulate the finite element method for problem (2.1) we use the standard Galerkin finite element method with linear elements in space and a centralised finite difference approximation for the second order time derivative. We introduce finite element space V_h for u, consisting of standard piecewise linear continuous functions on a mesh and satisfying Dirichlet boundary conditions. Let V_h^0 denote the corresponding finite element spaces satisfying homogeneous Dirichlet boundary conditions. The finite element method now reads: find u_h in V_h such that

$$(\frac{u_h^{k+1} - 2u_h^k + u_h^{k-1}}{\tau^2}, v) + (a^2 \nabla u_h^k, \nabla v) = (f^k, v), \quad \forall v \in V_h^0. \tag{3.1}$$

This produces the system of linear equations

$$M\mathbf{u}^{k+1} \;=\; \tau^2 F^k + 2M\mathbf{u}^k - \tau^2 K\mathbf{u}^k - M\mathbf{u}^{k-1}, \tag{3.2}$$

with proper initial and boundary conditions. Here, M is the mass matrix, K is the stiffness matrix depending on a possible varying wave velocity a, $k = 1, 2, 3\ldots$ denotes the time level, F^k is the load vector, $\mathbf{u}$ is the unknown discrete field values of u, and τ is the time step.

The explicit formulas for the entries in system (3.2) at the element level can be given as:

$$M^e_{i,j} = (\varphi_i, \varphi_j)_e, \tag{3.3}$$

$$K^e_{i,j} = (\nabla\varphi_i, \nabla\varphi_j)_e, \tag{3.4}$$

$$F^e_j = (f, \varphi_j)_e, \tag{3.5}$$

$$\text{where} \quad (a,b)_e = \int_{\Omega_e} a\ b \quad d\Omega_e, \tag{3.6}$$

where Ω_e is domain of the element e.

The matrix M^e is the contribution from element e to the global assembled matrix M, K^e is the similar contribution to global assembled matrix K, F^e is the contribution from element e to the assembled source vector F.

To obtain an explicit scheme we approximate M with the lumped mass matrix M^L, the diagonal approximation obtained by taking the row sum of M, see e.g. [8]. By multiplying (3.2) with $(M^L)^{-1}$ we obtain an efficient explicit formulation:

$$\mathbf{u}^{k+1} = \tau^2 (M^L)^{-1} F^k + 2\mathbf{u}^k - \tau^2 (M^L)^{-1} K \mathbf{u}^k - \mathbf{u}^{k-1}, \tag{3.7}$$

where matrix M^L is the approximation of the global mass matrix M by

$$M^L_{i,j} = \begin{cases} \sum_n M_{i,n} & , \quad i = j, \\ 0 & , \quad i \neq j, \end{cases} \tag{3.8}$$

that is the diagonal elements of the matrix M^L are the row-sums of M. In Section 4.1 we describe how the term $\tau^2 (M^L)^{-1} F^k$ can be efficiently implemented without using the standard assembly procedure at each time step.

To formulate the finite element method for system (2.5 -2.7) we use the standard Galerkin finite element method in space and the forward finite difference approximation to the first order time derivative. We introduce finite element spaces V_h, W_h for u_1, u_2, consisting of standard piecewise linear continuous functions on a mesh and satisfying Dirichlet boundary conditions. Let V_h^0, W_h^0 denote the corresponding finite element spaces satisfying homogeneous Dirichlet boundary conditions. The finite element method now reads: find (u_{1h}, u_{2h}) in $(V_h \times W_h)$ such that

$$(\frac{u_{1h}^{k+1} - u_{1h}^k}{\tau}, v) + (a^2 \nabla u_{2h}^k, \nabla v) = (f^k, v), \quad \forall v \in V_h^0, \tag{3.9}$$

$$(\frac{u_{2h}^{k+1} - u_{2h}^k}{\tau}, w) = (u_{1h}^k, w), \quad \forall w \in W_h^0. \tag{3.10}$$

This produces the system of linear equations for model (2.5–2.7) at each time step :

$$M\mathbf{u}_1^{k+1} = (F^k - a^2 K \mathbf{u}_2^k)\tau + M\mathbf{u}_1^k, \tag{3.11}$$

$$M\mathbf{u}_2^k = M\mathbf{u}_1^k \tau + M\mathbf{u}_2^k, \tag{3.12}$$

$$\mathbf{u}_1^0 \mid_\Gamma = 0 \tag{3.13}$$

$$\mathbf{u}_2^0 \mid_\Gamma = 0 \tag{3.14}$$

In these equations, M and K are the same matrices, as in the system (3.2), k denotes the time level, $\mathbf{u}_1$ and $\mathbf{u}_2$ are the unknown discrete field values of u_1 and u_2, τ is the time step size, Γ is the boundary of the *inner* region.

To obtain an explicit scheme we approximate M by M^L and multiply the first of the system equations by $(M^L)^{-1}$ so that the system can be rewritten in the more efficient form:

$$\mathbf{u}_1^{k+1} = ((M^L)^{-1}F^k - a^2(M^L)^{-1}K\mathbf{u}_2^k)\tau + \mathbf{u}_1^k, \tag{3.15}$$

$$\mathbf{u}_2^{k+1} = \mathbf{u}_1^k\tau + \mathbf{u}_2^k. \tag{3.16}$$

The disadvantage with explicit schemes is of course that we must choose small time steps to respect a CFL criterion:

$$\tau \leq \frac{h}{ac} \ , \tag{3.17}$$

where h is the minimal local mesh size of the elements, and c is a constant.

3.2. **Finite difference formulation.** In the Ω_{FDM} domain, we use FDM. The FDM stencil can be derived via the FEM schemes presented in the previous section, when applied to a structured Cartesian grid. For problem (2.1) we obtain

$$u_{i,j,k}^{l+1} = \tau^2(f_{i,j,k}^l + a^2\Delta u_{i,j,k}^l) + 2u_{i,j,k}^l - u_{i,j,k}^{l-1}, \tag{3.18}$$

where $u_{i,j,k}^l$ is the solution on time iteration l at point (i,j,k), $f_{i,j,k}^l$ is the source function, τ is the time step, and $\Delta v_{i,j,k}^l$ is the discrete Laplacian. In three dimensions, we get the standard seven-point stencil:

$$\begin{aligned} \Delta v_{i,j,k}^l &= \frac{v_{i+1,j,k}^l - 2v_{i,j,k}^l + v_{i-1,j,k}^l}{dx^2} + \frac{v_{i,j+1,k}^l - 2v_{i,j,k}^l + v_{i,j-1,k}^l}{dy^2} + \\ &\quad \frac{v_{i,j,k+1}^l - 2v_{i,j,k}^l + v_{i,j,k-1}^l}{dz^2}, \end{aligned} \tag{3.19}$$

where dx, dy, and dz are the steps of the discrete finite difference meshes in the directions x, y, z, respectively.

3.3. **Absorbing boundary conditions.** We have also simulated a variation of the problem (2.1–2.4) with Dirichlet boundary condition replaced by the absorbing boundary condition. It means, that this boundary conditions approximate the solution on the boundaries. We use the following boundary condition taken from [6] :

$$\left.\frac{\partial}{\partial t}u - \frac{\partial}{\partial x}u\right|_{x=0} = 0. \tag{3.20}$$

We are using forward finite difference approximation in the middle point of the condition (3.20), which gives a numerical approximation of higher order than ordinary (backward or forward) approximation. For example, for the left boundary of the *outer* domain we obtain:

$$\frac{u_{i,j,k}^{l+1} - u_{i,j,k}^l}{dt} + \frac{u_{i+1,j,k}^{l+1} - u_{i+1,j,k}^l}{dt} - \frac{u_{i+1,j,k}^l - u_{i,j,k}^l}{dx} - \frac{u_{i+1,j,k}^{l+1} - u_{i,j,k}^{l+1}}{dx} = 0, \tag{3.21}$$

which can be transformed to

$$u_{i,j,k}^{l+1} = u_{i+1,j,k}^{l} + u_{i,j,k}^{l} \frac{dx - dt}{dx + dt} - u_{i+1,j,k}^{l+1} \frac{dx - dt}{dx + dt}. \tag{3.22}$$

For other boundaries of the *outer* domain we find analogous boundary conditions.

4. Implementational issues

We have chosen C++ as the implementation language. It allows us to implement the problem and the algorithms on a high level of abstraction without much loss of efficiency. We have implemented important notions such as grid, boundary, operator, and grid function as C++ classes. For FEM, we have reused Kraftwerk, an existing in-house FEM framework. The software package PETSc [2] is used for matrix vector computations. For FDM, new classes were developed, called ABCD. They are specialised for Cartesian grids with cavities. The cavities may be filled with unstructured grids. The ABCD grid class represents its nodes in a memory efficient way, by only storing index sets of contiguous nodes along one axis. The matrix multiplication described by (3.18) is performed by looping over the index sets and applying the finite difference molecule. See Appendix for more information on Kraftwerk and ABCD. Here, we report only on the hybrid layer of the implementation.

We sketch the use of the hybrid method on problem (2.1). In order to have a flexible implementation, we introduce an auxiliary class `WaveEqProblem`, which are configured either to use FDM in the whole region (using a structured grid), or to use FEM in the whole region (using a hybrid or a structured grid), or to use the hybrid method. The main program which uses Kraftwerk for the unstructured regions and ABCD for structured regions can then be written in this schematic form (Compare with the algorithm in Section 3):

```
WaveEqProblem p(femGrid, fdmGrid);     // initialise with grids
                                       // for FEM and FDM
dt = p.initTime(t);                    // get time step and init time
p.initFDM();                           // initialise FDM
p.initFEM();                           // initialise FEM
p.initExchangeFEM();                   // make data structures
p.initExchangeFDM();                   // for overlap exchanges

for(k = 0; k < noTimeSteps; k++)       // main loop
{
  p.solveFDM(t);                       // solve one time step with FDM
  p.solveFEM(t);                       // solve one time step with FEM
  p.applyExchange();                   // perform overlap exchanges
  p.applySwap();                       // swap solutions vectors
  t += dt;                             // increment time
}
```

Apart from initialisation, the essential work is done for each time step in `p.solveFDM(t)` and `p.solveFEM(t)` where the source function f is evaluated and matrix vector multiplications are performed.

4.1. **Notes on optimisations.** One of the central issues of our project has been to study the efficiency of the hybrid method, and it is therefore important for a just evaluation to optimise the implementation of the involved methods. Here, we comment on some optimisations.

First, we observe that in (3.7) we can write $2\mathbf{u}^k - \tau^2 (M^L)^{-1} K \mathbf{u}^k = A\mathbf{u}^k$, where $A = 2I - \tau^2 (M^L)^{-1} K$ is a sparse matrix, and I the identity matrix. If the timestep τ is constant, A is independent of time and can therefore be computed in the initialisation step. A similar optimisation is used for FDM.

Regarding the computation of the load vector F^k in (3.7), a standard assembly computation using Gaussian quadrature is expensive. By instead using nodal quadrature, the load vector will be $F^k = (M_L)^{-1}\mathbf{f}^k$, where the vector $\mathbf{f}^k$ is the source function evaluated at the nodes of the grid at time t_k. Alternatively, the source function can be approximated by the finite element function obtained by interpolating the source function at the nodes. In this case, exact quadrature yields that $F^k = M\mathbf{f}^k$.

Thus, the FEM computation of the solution $\mathbf{u}^{k+1}$ becomes

$$\mathbf{u}^{k+1} = \tau^2 \mathbf{f}^k + A\mathbf{u}^k - \mathbf{u}^{k-1}. \tag{4.1}$$

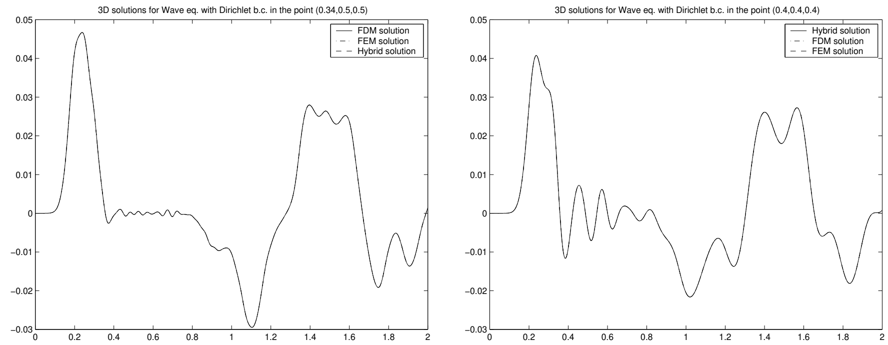
Note that the FDM scheme (3.18) coincides with (4.1) in regions where the grid is structured, see [9].

In the interior of a structured FEM grid with constant wave velocity a, the nonzero components of the sparse stiffness matrix K will reduce to the FDM five and seven point stencils, for 2D and 3D, respectively. This property can be used to replace the sparse matrix A with a matrix where the zero components are eliminated. The size of the reduced matrix is in 2D 5/7 times the size of the original matrix. In 3D the corresponding factor is 7/15, which is quite significant. A matrix vector product with the new matrix has a correspondingly smaller cost. However, in general for an unstructured grid the zero elements in the stiffness matrix will not occur and hence the described matrix reduction is not possible.

Regarding the finite difference region, three different strategies were considered:

1. Allocate memory all over Ω and compute all over Ω.
2. Allocate memory all over Ω but compute only in Ω_{FDM}.
3. Allocate memory only in Ω_{FDM} and compute only in Ω_{FDM}.

If the Ω_{FEM} region is very small relative to Ω, the first strategy may be advantageous. Even though some unnecessary computations would be carried out, the overhead of administering the loop boundaries would be avoided. If, on the other hand, the Ω_{FEM} region is very large relative to Ω, the third strategy may be advantageous, because no unnecessary memory is allocated. The disadvantage with the third strategy is a more complicated communication pattern, with negative effect on the performance. However, for the main problems of our present study, we found that strategy two was most advantageous. This strategy uses direct addressing all over the domain and avoids unnecessary computations. Compare also with [1, 3] which are dealing with optimizing C++ code and studying the effect of the cache sizes of the computer, respectively.



(a) (b)

Figure 4. Solutions for three dimensional wave equation at one point. We use one source function in the *inner* domain. In the graphs (a) - (b) we present the FEM, FDM and hybrid solutions, which coincide at both points. We used the same structured mesh to test all these three methods.

5. Numerical examples

We illustrate the use of the hybrid method in several examples. In all examples we simulate model problem (2.1) in domain $\Omega = [0,1]^n, n = 2,3$, with homogeneous Dirichlet boundary conditions, and with initial conditions $u = \frac{\partial u}{\partial t} = 0$. The domain Ω is decomposed into the two domains Ω_{FEM} and Ω_{FDM} with two overlapping layers of nodes. The inner domain is $\Omega_{FEM} = [0.4, 0.6]^n, n = 2,3$ in two and three dimensions respectively. In the Ω_{FDM} domain we apply FDM with homogeneous Dirichlet boundary conditions. The space mesh for FEM in two dimensional examples in the Ω_{FEM} domain is unstructured and consists of triangles. In three dimensions we use tetrahedra for the unstructured grid. The wave is in our examples driven by source functions which generate pulses at different points:

$$f_1(x, x_0) = \begin{cases} 10^3 \sin^2 \pi t & \text{if } 0 \le t \le 0.1 \text{ and } |x - x_0| < 0.1, \\ 0 & \text{otherwise;} \end{cases} \tag{5.1}$$

$$f_2(x, x_0) = \begin{cases} \sin(10^3 \sin(40\pi t)) e^{-10 \cdot (x - x_0)^2} & \text{if } |x - x_0| < 0.05, \\ 0 & \text{otherwise;} \end{cases} \tag{5.2}$$

$$f_3(x, x_0) = \begin{cases} \sin(10^3 \sin(40\pi t)) e^{-10 \cdot (x - x_0)^2} & \text{if } |x - x_0| < 0.05, \\ 0 & \text{otherwise;} \end{cases} \tag{5.3}$$

First, we present a few model applications in two and three dimensions. Next, we present performance measurements, where we compare the efficiency of the hybrid method with FDM and FEM.

5.1. **Two-dimensional examples.** In the first example we solve the problem (2.5–2.7) with Dirichlet boundary conditions on the *outer* domain boundary. We have chosen the explicit scheme (3.7) to implement the FEM and the explicit scheme (3.18) to implement FDM. We choose source function f_1 located at the centre of the *inner* domain and with time step $\tau \approx 0.000097$, which we compute from criterion (3.17) with the constant $c = 0.1$. This time step makes the explicit scheme stable. We plot the hybrid method solutions in Fig. 5. The solutions in the point with coordinates $(0.34, 0.5)$ plotted in Fig. 14-a and the solutions in the same point, but with source functions f_2 and f_3 shown in Fig. 14-b and Fig. 14-c, respectively.

The same example but with absorbing boundary conditions on the outer boundary we show in Fig. 7. The solution at the point with coordinates $(0.34, 0.5)$ plotted in Fig. 14-d.

We perform the same test but now the source function is located in the structured *outer* domain. We compute with time step $\tau \approx 0.00019$, which we get from criterion (3.17) with the constant $c = 0.2$. We present result of the computation for hybrid method in Fig. 6.

The second example we present a computation of wave propagation in the inhomogeneous *inner* domain composed of different material types having different wave velocities (the coefficients a in (2.2)). We tested with different coefficient $a^2 = 4, 2.25, 6.25$ inside the unstructured sub-domain, and $a^2 = 1$ outside it.

We assume absorbing boundary conditions at the all boundaries of the structured domain.

We show results for the hybrid method in Fig. 8. We can clearly see difference in the wave speed between two materials.

In the third example we present a plane wave propagation. We define a plane wave on the left boundary of the *outer* domain as the function

$$f(x,t) \mid_{x=0} = 0.1 \sin(25\ t - \pi/2) + 0.1 \tag{5.4}$$

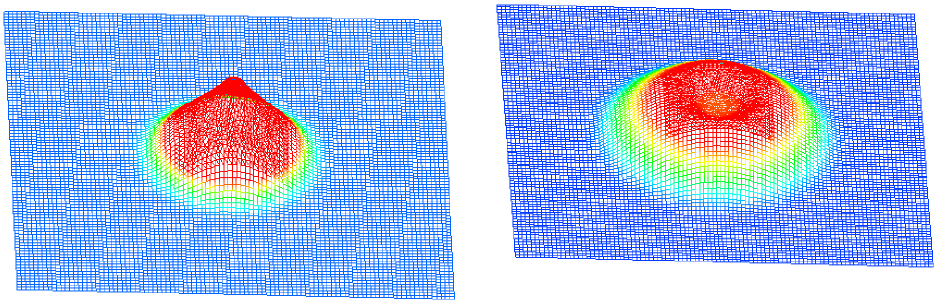
and solve with the hybrid method with the time step $\tau \approx 0.001$. We present results in Fig. 9.

In the fourth example we present the hybrid method with two source functions : one located in the unstructured *inner* domain and another in the structured *outer* domain. We use explicit schemes (3.7),(3.18) and apply absorbing boundary conditions on the *outer* domain. We compute with time step $\tau \approx 0.00097$, which we get from criterion (3.17) with the constant $c = 0.1$. We present result of the computations for the hybrid method in Fig. 10.

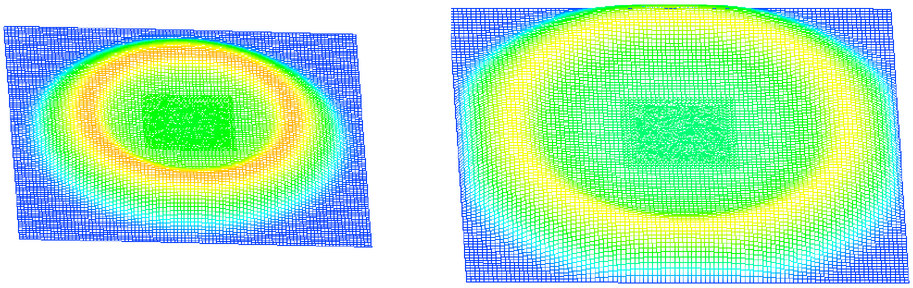
5.2. **Three-dimensional examples.** We demonstrate the use of the hybrid method on the domain $\Omega = [0, 1]^3$. For the numerical simulations we have chosen a finite element mesh with element size 0.04 and perform computations in the time interval $[0, 2]$.

In the first example we are solving the model problem (2.5–2.7) with Dirichlet boundary conditions and with the source function f_1 located at the centre of the three dimensional domain. We show the hybrid method solutions in Fig. 12 and solution at the point of the domain with coordinates $(0.4, 0.4, 0.4)$ in Fig. 4-a and in the point with coordinates $(0.34, 0.5, 0.5)$ in Fig. 4-b.

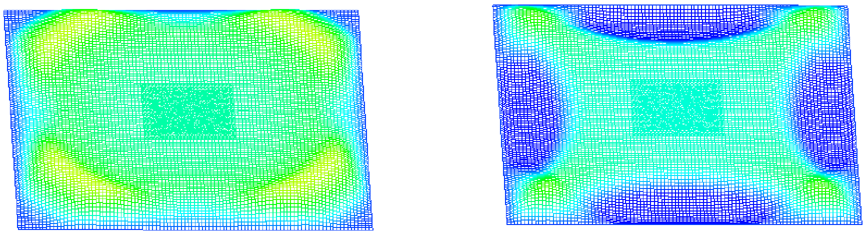
The second example is solution of model problem (2.5–2.7) with Dirichlet boundary conditions and with two source functions. One of them, located in the *inner* domain, and another, located in the *outer* domain. The solution of this problem we present in Fig. 13.



a) time is 0.3884 b) time is 0.4855

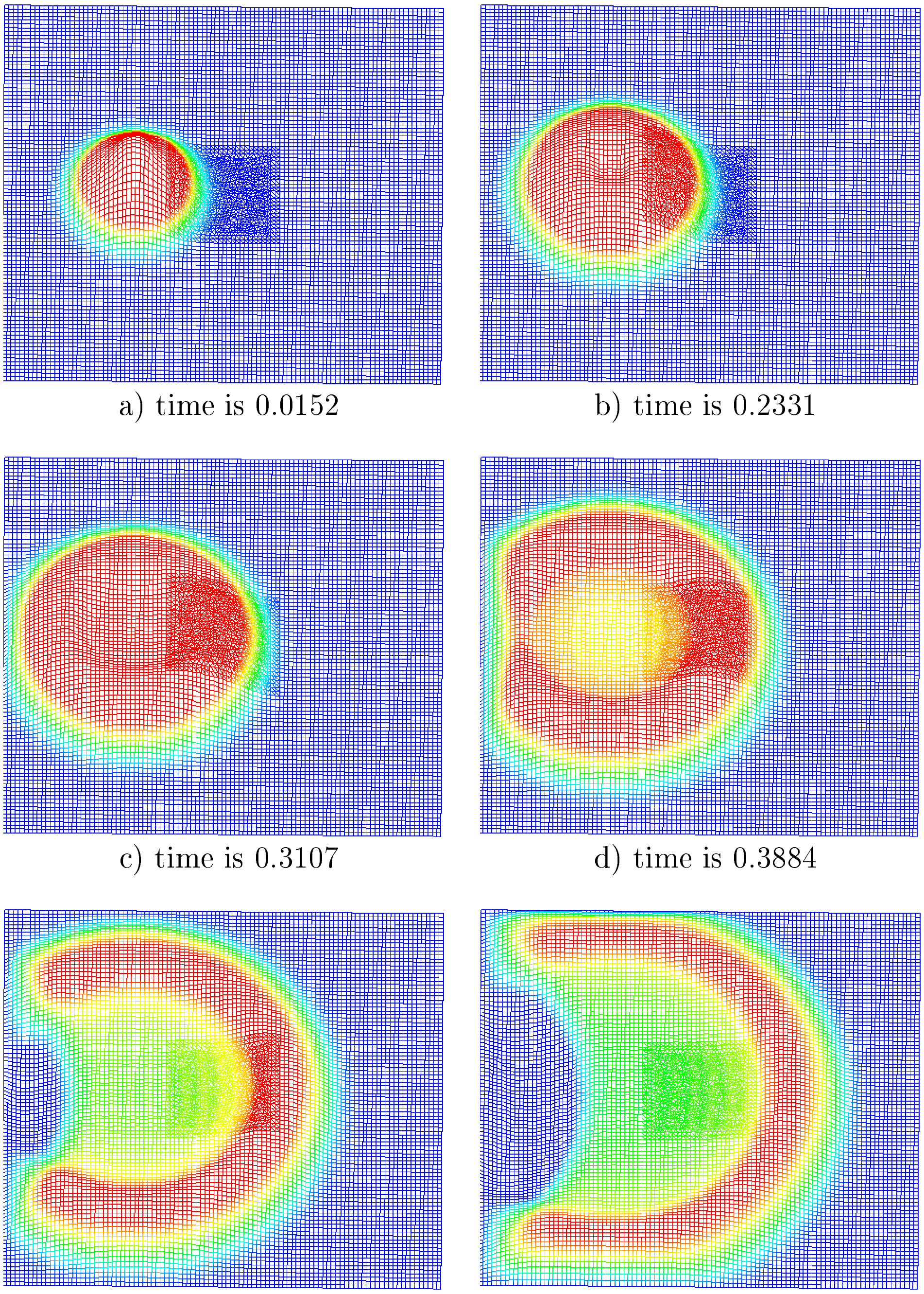


c) time is 0.5827 d) time is 0.6798



e) time is 0.7769 f) time is 0.8740

FIGURE 5. Example of the hybrid method for the two-dimensional wave equation with Dirichlet boundary conditions. We choose a source function located at the centre of the unstructured domain. We can see the effect of the Dirichlet boundary conditions in the graphs (c) and (d). The values of the solution are represented both by the high of the graph and by the gray scale.


a) time is 0.0152 b) time is 0.2331

c) time is 0.3107 d) time is 0.3884

e) time is 0.4661 f) time is 0.5438

Figure 6. Example of the hybrid method for the two-dimensional wave equation with Dirichlet boundary conditions. We choose a source function located in the *outer* domain at point (0.5, 0.3). We can see how smoothly passes the solution through the *inner* domain in the graphs (a)–(d). We show how the Dirichlet boundary conditions works in the graphs (e) and (f). The values of the solution are represented both by the high of the graph and by the gray scale.

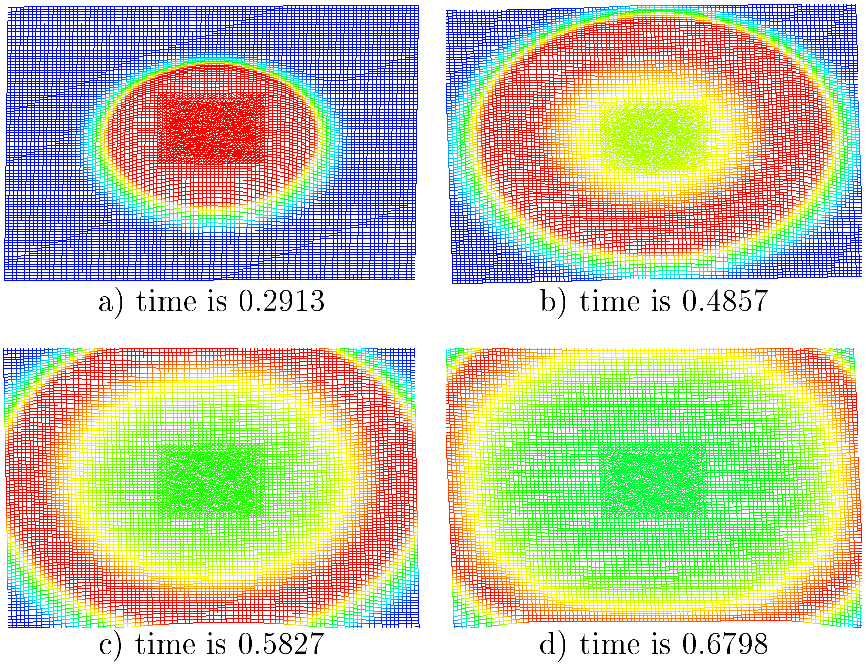
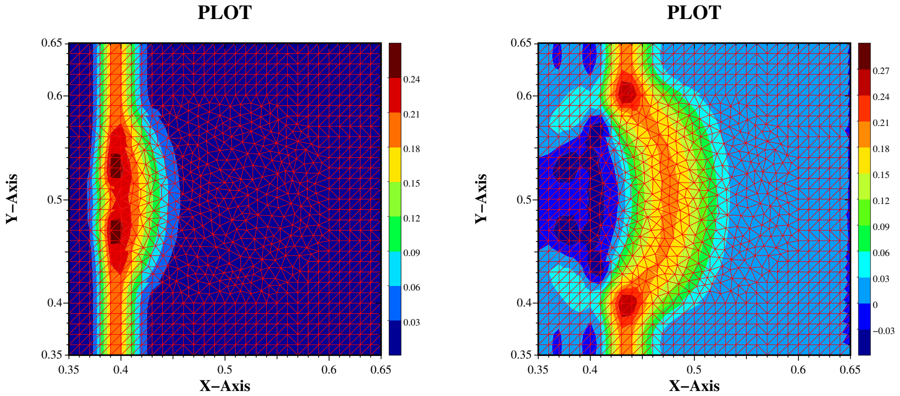
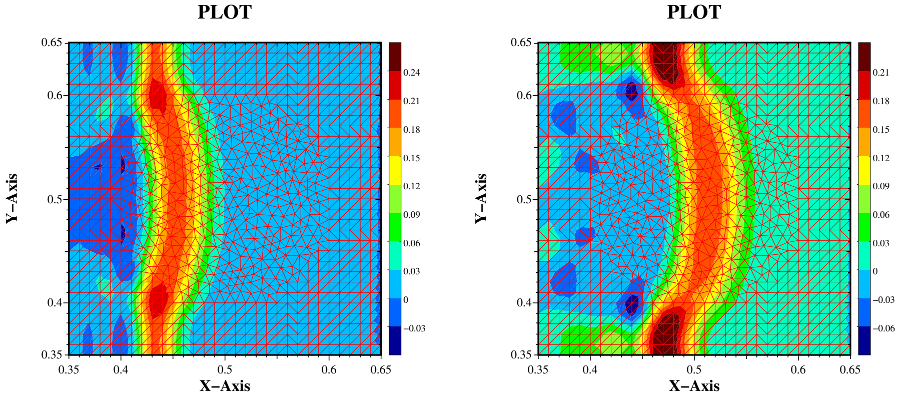

a) time is 0.2913
b) time is 0.4857
c) time is 0.5827
d) time is 0.6798

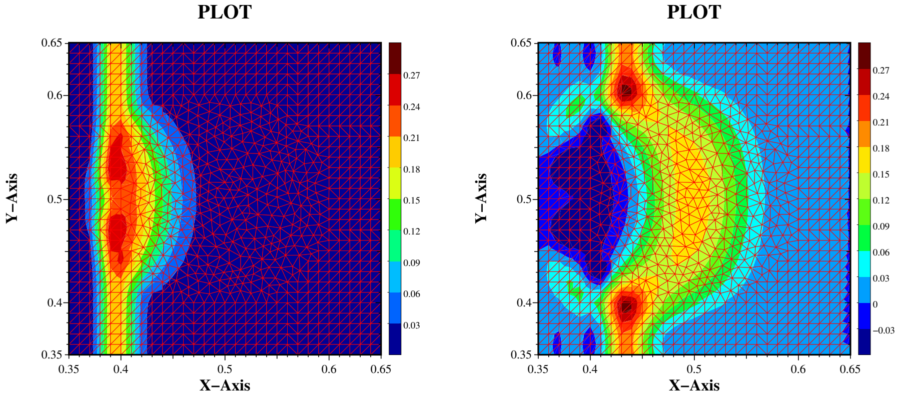
FIGURE 7. Example of the hybrid method for the two dimensional wave equation with absorbing boundary conditions. The source function is located in the unstructured domain. We apply Engquist-Majda absorbing boundary condition on the outer boundary of the structured domain. The graphs (c) and (d) show the effect of this condition.



a) time is 0.12 b) time is 0.16

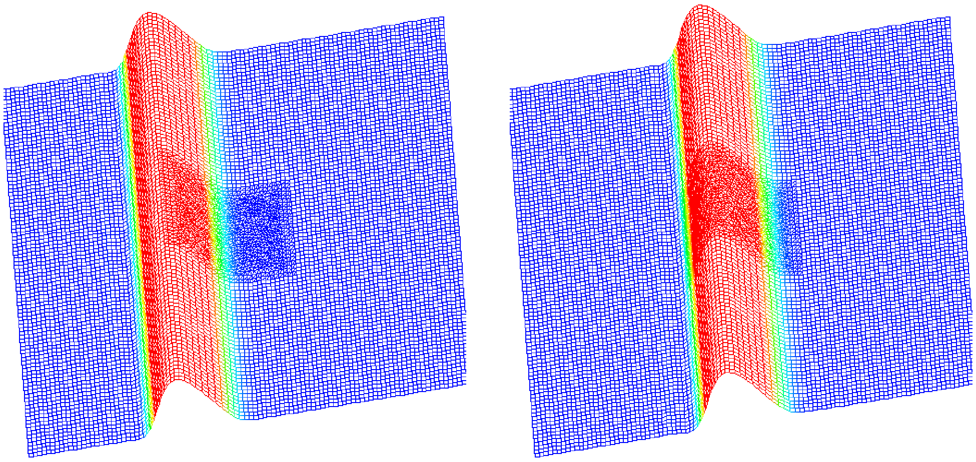


c) time is 0.16 d) time is 0.2

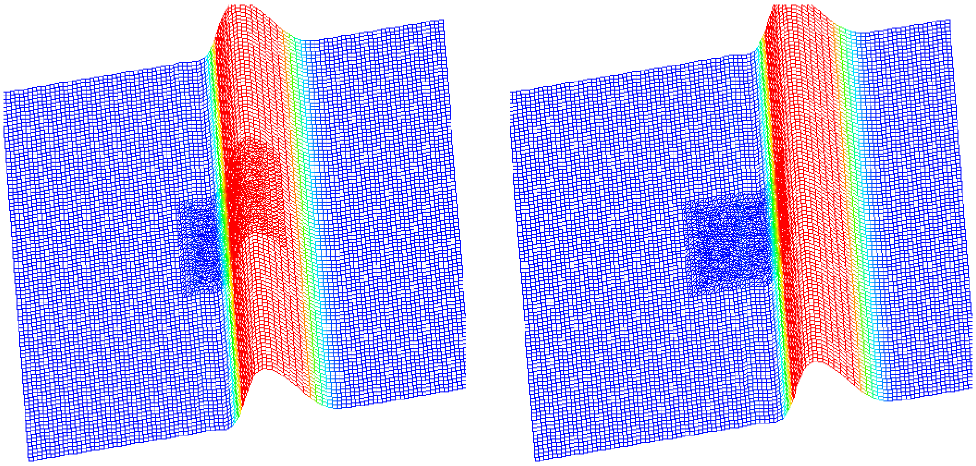


e) time is 0.12 f) time is 0.16

FIGURE 8. Example of the hybrid method for two dimensional wave equation in the inhomogeneous *inner* domain composed of different material types having different wave velocities (the coefficients a in (2.2)). This coefficient is taken as $a^2 = 4$ at the a) - b), $a^2 = 2.25$ at the c) - d), $a^2 = 6.25$ at the e) - f) inside the unstructured subdomain, and $a^2 = 1$ outside it. We can clearly see in the graphs (a)–(c) that the wave propagates faster inside the unstructured domain, than outside it.

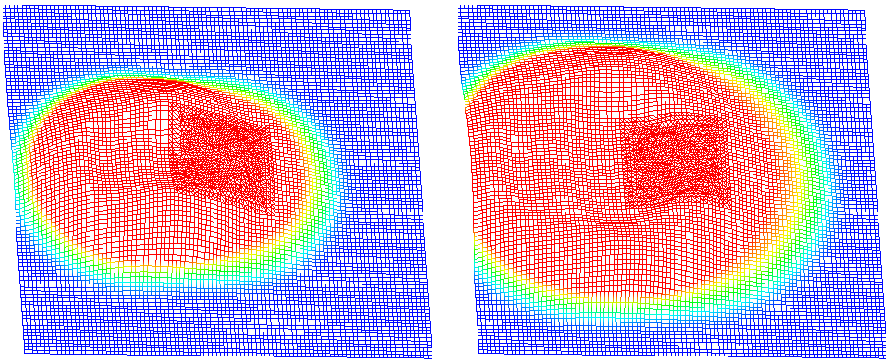


time is 0.5 time is 0.6

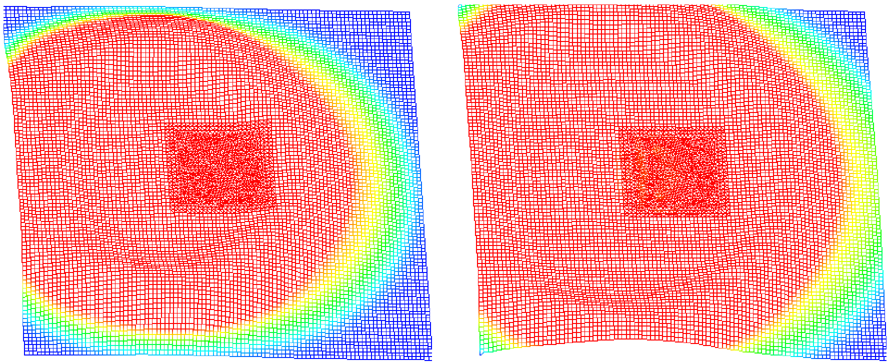


time is 0.8 time is 0.9

FIGURE 9. A plane wave. We define a plane wave on the left boundary of the *outer* domain and solve the wave equation with the hybrid method.

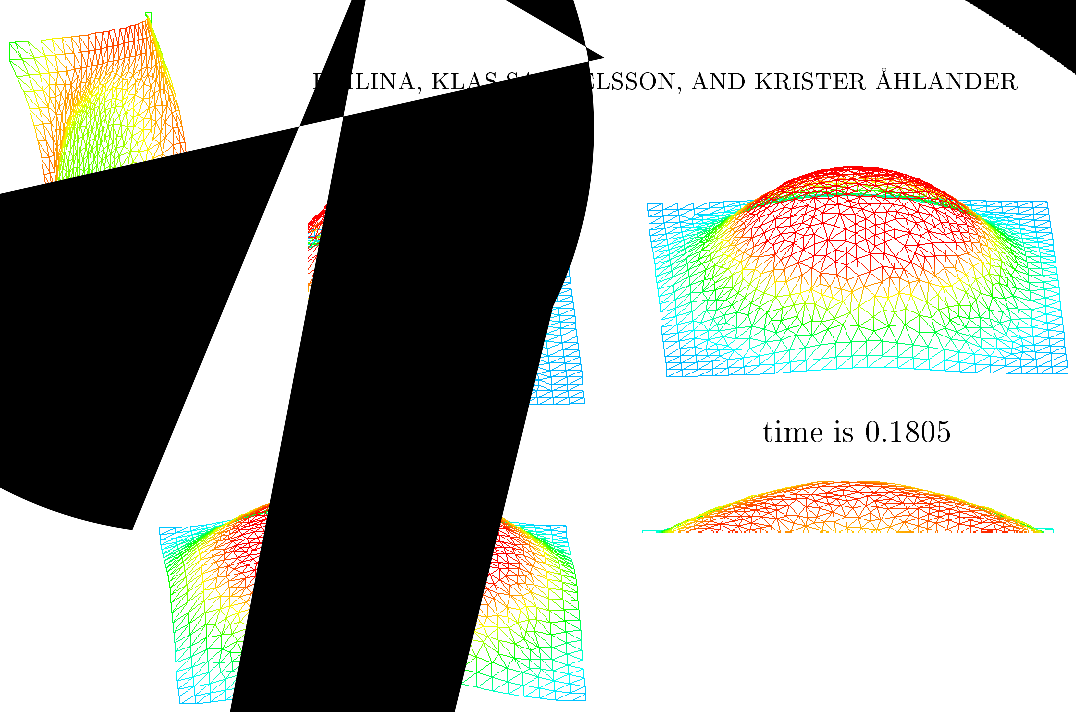


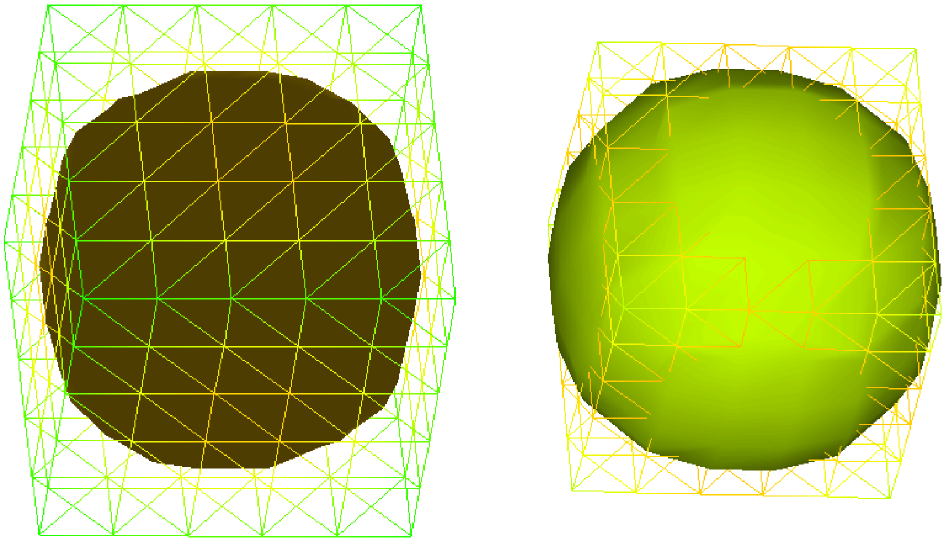
a) time is 0.2913 b) time is 0.3884



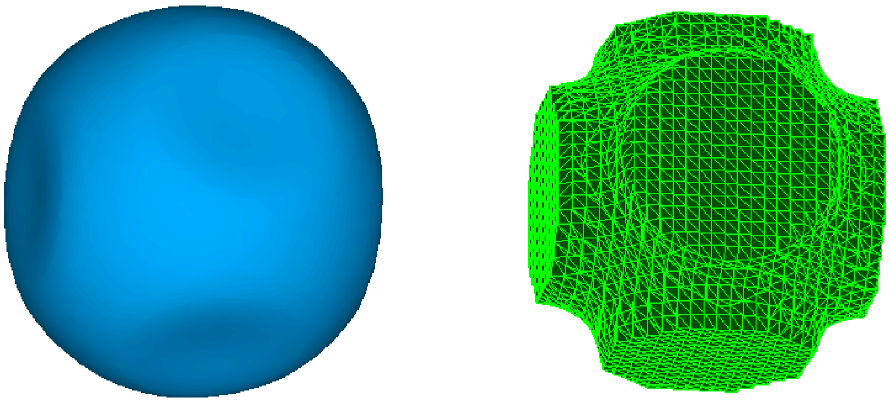
c) time is 0.4855 d) time is 0.5827

FIGURE 10. Two pulses. Example of the hybrid method for the two dimensional wave equation. We start with one pulse in the *inner* and one in the *outer* domains. We apply absorbing boundary conditions on the boundary of the structured domain, as can be observed on (b)–(d).


time is 0.1805

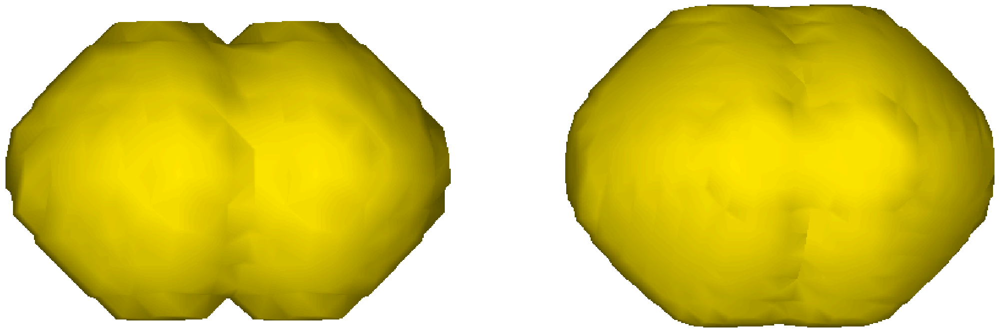


a) time is 0.105 b) time is 0.140

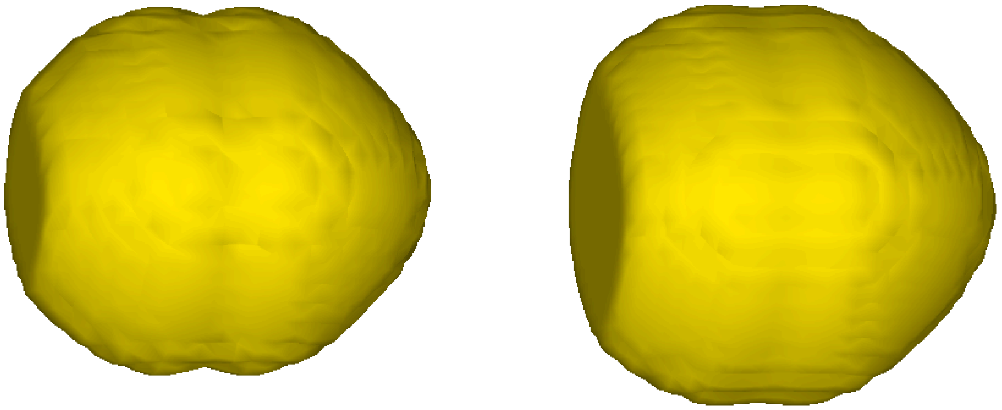


c) time is 0.175 d) time is 0.210

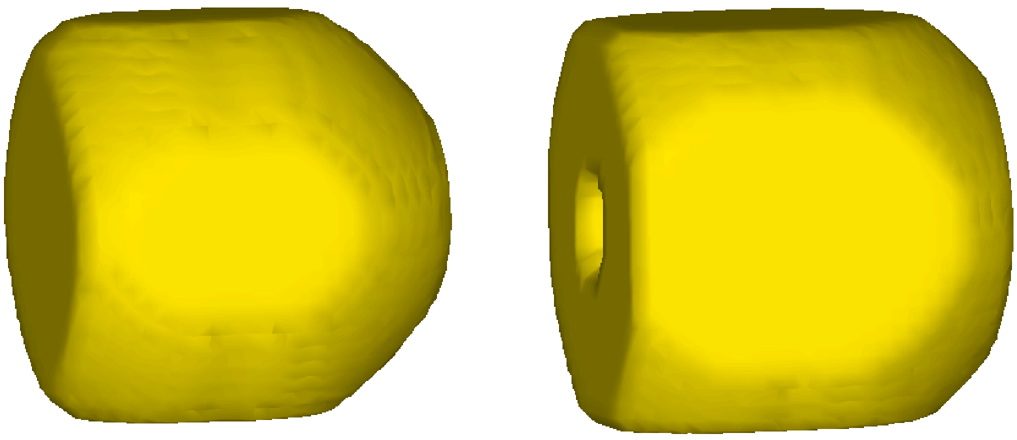
FIGURE 12. Example of the hybrid method for the three dimensional wave equation. We apply Dirichlet boundary conditions on the *outer* domain. We test the hybrid method with one source function located in the centre of the *inner* domain, see graph (a) and (b). We can see in the graph (b) that the finite element solution goes through the *inner* domain into *outer* domain. We show the effect of the Dirichlet boundary condition in the graphs (c)–(d).



time is 0.105 time is 0.140

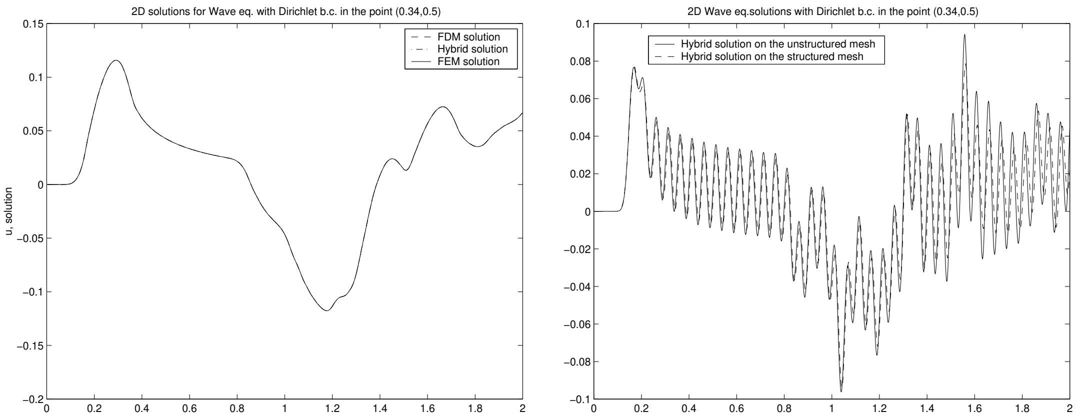


time is 0.175 time is 0.210

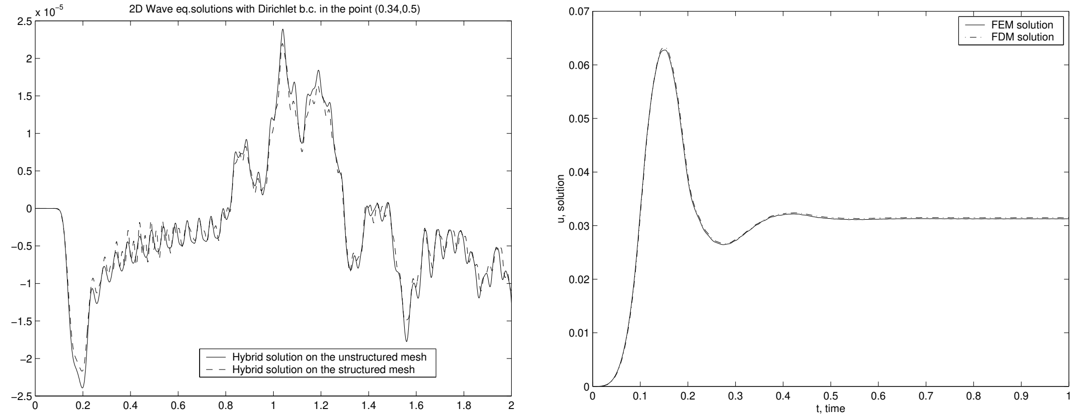


time is 0.245 time is 0.275

FIGURE 13. Solutions of the three dimensional wave equation with Dirichlet boundary conditions, two pulses. We define one pulse in the *inner* domain, where we apply FEM, and another in the *outer* domain, where we apply FDM.



(a) with Dirichlet b.c. (b) with Dirichlet b.c.



(c) with Dirichlet b.c. (d) with absorbing b.c.

FIGURE 14. Solutions for the two dimensional wave equation at one point (0.34, 05). In the graph (a) we present the FEM, FDM and hybrid solutions. As we have explained in section (4.1), the three solutions are to be identical on the whole time range. In the graphs (b)–(c) we present two dimensional wave equation solution at one point (0.34, 05). We solved the problem by the hybrid method first on the unstructured and then on the structured mesh. We show the solution of the two dimensional wave equation with absorbing boundary conditions found by the hybrid method, at one point, in the graph (d).

5.3. **Performance comparisons.** We investigate the performance of the different methods by computing, with each method, the wave equation on structured grids, and measuring the cpu time per node and per iteration.

The size of the used computational grids are shown in Table 1. The performance tests were performed on a Sun workstation with free memory size 773Mb and 2048Mb real memory.

Table 2 and Table 3 present efficiency results, in terms of cpu time per node and iteration. The fractions FEM/Hybrid and FEM/FDM are also presented in the tables. We note that, for two dimensions, the fraction FEM/Hybrid ≈ 3.2 and the fraction FEM/FDM ≈ 3.7. In our three-dimensional tests, the corresponding fractions have increased. Here, the fraction FEM/Hybrid ≈ 4.4, and the fraction FEM/FDM is around 6.7.

The tables show that the fractions increase with the size of the grid. This can be explained by cache effects, since the required memory of the FEM sparse matrix is much larger than the corresponding FDM difference molecule. Another effect of importance is that the nodes at the boundary is making up a smaller part of the total number of nodes. For the hybrid method, the relative cost associated with computing the solution in the overlap region with both methods and exchanging solution values, decreases as the grid sizes increase, compare with Table 1.

5.4. **Remarks on the performance comparisons.** For our test cases, the source function evaluations in the simulations are a minor part of the execution time, since f_1 is nonzero only in a small fraction of the time steps. Since the source function evaluations are, by Section 4.1, identical for FEM and FDM methods, would an expensive source function evaluation results in a decreased fraction FEM/FDM.

The experiments are made on structured grids for which it was possible to use reduced FEM matrices as was described in Section 4.1. The ratios FEM/FDM in Tables 2 and 3 are essentially due to the difference in execution time of multiplying a row of a sparse matrix compared to applying the corresponding finite difference molecule. Numerical experiments (not presented here) indicate that the execution time for a sparse matrix multiplication increases linearly with the number of coefficients per row. Therefore, experiments performed with unreduced FEM matrices would result in an increase of the fraction FEM/FDM, approximately with the factors 7/5 and 15/7, for 2D and 3D simulations, respectively. The factors come from the matrix reduction sizes of Section 4.1. A corresponding increase would occur for the fraction FEM/Hybrid.

5.5. **Memory consumption.** An important issue, which in this paper has not been studied in detail, is the memory consumption of the FEM version versus the FDM and hybrid implementations. The FDM implementation is here advantageous in two respects. Firstly, the FDM grid representation uses much less memory than the corresponding unstructured FEM grid, and secondly a finite difference molecule is used instead of a memory consuming sparse matrix. These two advantages for the hybrid method regarding the memory consumption is probably, for many applications, more important than the speed-up presented above.

size of the mesh elements, h	number of nodes in Ω_{FEM}	number of nodes in Ω	number of nodes in the overlapping layers	spatial dimension
0.0025	6561	160801	3192	2
0.005	1681	40401	1592	2
0.01	441	10201	152	2
0.02	121	2601	72	2
0.01	9261	1030301	4348	3
0.02	1331	132651	988	3
0.04	216	17576	208	3

Table 1. Meshes for the performance test.

h	Hybrid	FEM	FDM	FEM/Hybrid	FEM/FDM
0.0025	9.70385e-7	3.25877e-6	8.11587e-7	3.3582	4.0153
0.005	9.25238e-7	3.05127e-6	8.09583e-7	3.2978	3.7689
0.01	9.27085e-7	2.81574e-6	8.09961e-7	3.0372	3.4764
0.02	9.31629e-7	2.69945e-6	7.90749e-7	2.8976	3.4138

Table 2. Performance for the 2D wave equation

h	Hybrid	FEM	FDM	FEM/Hybrid	FEM/FDM
0.01	4.84596e-6	2.21148e-5	3.05454e-6	4.5636	7.2400
0.02	4.88609e-6	2.15223e-5	3.28249e-6	4.4048	6.55670
0.04	4.73657e-6	2.02856e-5	3.17052e-6	4.2828	6.39819

Table 3. Performance for the 3D wave equation

6. Conclusions

We have presented explicit hybrid methods for the scalar wave equation. The hybrid approach can be derived from FEM, which is rewritten as a FDM method in parts of the domain where the discretisation is Cartesian. Thus, we combine the efficiency of FDM with the flexibility of FEM. Moreover, since our algorithm is equivalent to a pure FEM approach, stability results from FEM are valid, and no interface instabilities are introduced.

Regarding the implementation, we emphasize that object-oriented abstractions made it easy to develop our application as independent modules.

Numerical examples are presented, validating that the hybrid approach is equivalent to a pure FEM algorithm, and on a structured grid also to a pure FDM simulation. We also present examples illustrating the capabilities of our approach in 2D and 3D. Our examples use both Dirichlet boundary conditions and absorbing boundary conditions.

One of the most important questions of this study is whether the hybrid approach really gives better performance, compared to pure FEM. Our test cases for structured grids indicates that this is the case. In our examples, the hybrid method is 2.9–4.6 faster, than a highly optimized pure finite element version. The relative efficiency of the hybrid method is somewhat better for larger problems and when the unstructured part of the domain becomes relatively smaller. Both these effects are typical for 3D problems.

We conclude that the hybrid approach is advantageous, in particular for memory demanding problems where large parts of the computational domain may be discretised by uniform Cartesian grids, while unstructured grids are more suitable in small regions of the domain. Our present study has been concerned with explicit time stepping methods. Further investigations to extend the hybrid method to implicit methods is an area of future research.

References

[1] E. Acklam, A. Jacobsen and H.P. Langtangen. Optimizing C++ code for explicit finite difference schemes. Oslo Scientific Computing Archive, Report 1998-4.

[2] S. Balay, W. Gropp, L-C. McInnes, B. Smith. PETSc user manual, http://www.mcs.anl.gov/petsc

[3] C. Douglas, J. Hu, M. Kowarschik, U. Rüde and C. Weiss. Cache optimization for structured and unstructured grid multigrid. ETNA volume 10, pp.21-40, (2000)

[4] F. Edelvik, U. Andersson, and G. Ledfelt. Hybrid finite volume - finite difference solver for the Maxwell equations. In AP2000 Millennium Conference on Antennas & Propagation, Davos, Switzerland, (2000).

[5] F. Edelvik and G. Ledfelt. Explicit hybrid time domain solver for the Maxwell equations in 3D. J. Sci. Comput., 2000.

[6] B. Engquist, A. Majda. Absorbing boundary conditions for the numerical simulation of waves, Math. Comp., Volume 31, number 139, p.629–651, (1977).

[7] K. Eriksson, D. Estep and C. Johnson. Computational Differential Equations. Studentlitteratur, Lund, (1996).

[8] T. J. R. Hughes. The Finite Element Method. Prentice Hall, (1987).

[9] S. C. Brenner, L. R. Scott. The Mathematical theory of finite element methods. Springer-Verlag, (1994).

APPENDIX A. DESCRIPTION OF THE MAIN KRAFTWERK CLASSES

Base class	methods	description
`GridB`		grid base class which represents the geometry of the domain.
	`getNoSpaceDim`	get space dimensions
	`getNoElms`	get number of elements in the grid
	`getNoNodes`	get number nodes in grid
	`getMaxNoNodesInElm`	get max number of the nodes in element
	`getMaterialType`	get material type of the elements in grid
	`setMaterialType`	set type of the material
	`getCoor`	get coordinate of the node n with index d
	`putCoor`	put coordinate of the node n with index d
	`loc2glob`	get global node number of the element e and local node number i
	`putLoc2glob`	put global node number in element
	`oneElementTypeInGrid`	type bool
	`getElementType`	get element type for element e
	`getNoNodesInElm`	get number of nodes in element
	`getNoNodesForElmType`	get number nodes in element for different element types
	`print`	print to file
	`scan`	scan from file
	`getNeighbor`	neighbour information about elements in grid
	`getElmFromNodes`	get element from two nodes with global numbers $n1$ and $n2$
	`getNoNodesForElmType`	get nodes number from type of the element
`NeighborFE`		represents the neighbour information in a finite element grid.
`SparseDS`		defines the data structure and a public interface for general sparse matrix storage.
`Mapper`		implements iso-parametric mapping for each elements in the finite element grid into a centre of coordinates, compute the Jacobian of the mapping and local assembled integral over elements depending of the quadrature rule
	`mapIsoparametric`	make iso-parametric mapping for element e and quadrature rule q

Base class	subclass, methods	description
`QuadRule`		abstract base class which determines the number of the Gauss points and their coordinates
	`getGPntCoord`	get coordinates of the Gauss points
	`QuadRuleTet4PntGauss`	It computes the integral for tetrahedron with 4 Gauss points.
	`QuadRuleTri1PntGauss`	It computes the integral for triangles with 1 Gauss point.
	`QuadRuleTri3PntGauss`	It computes the integral for triangles with 3 Gauss points
`Equation`		abstract base class which computes the local assembled matrix for each element of the grid.
	`integrateJacobian`	computes the local assembled matrix for main equation without a right hand side.
	`integrateResidual`	computes the local assembled vector for right hand side of the equation.
	`EquationLaplace2D`	represents the Laplace operator
	`EquationMass`	represents the mass equation
	`EquationWaveCDE`	represents the wave equation as a system of the two equations for implicit scheme
	`EquationWavefnimpl`	represents the right hand side for wave equation for implicit scheme
	`EquationWavefnexpl`	represents the right hand side for wave equation for explicit scheme
`Element`		base class which initialise the type of the elements, the numbers of the basis functions, values for basis functions and its derivatives, the global numbers of the nodes for elements of the grid.
	`getCoord`	get coordinates for node
	`loc2glob`	get global node number from local
	`glob2loc`	get local node number from global
	`getNoNodes`	get number of the nodes in element
	`getElmNo`	get number of the element
	`getMaterialType`	get material type for element
	`setNoGPnts`	set number of Gauss points for element
	`getNoGPnts`	get number of Gauss points for element
	`setGPntCoord`	set Gauss points coordinates
	`gPntCoord`	get Gauss points coordinates
	`setDetJac`	set determinant of Jacobian for element
	`detJac`	determinant of Jacobian
`ElementTetLin`		sets the basis functions and its derivatives for tetrahedron
`ElementTriLin`		sets the basis functions and its derivatives for triangle
`ElementQuadLin`		sets the basis functions and its derivatives for square

Appendix B. Description of the ABCD classes

Base class	subclass	description
`SDGeometry`		represent a structured, equidistant grid for FDM.
`SDIndexes`		represent indexes for a subdomain in `SDGeometry`. We can represent our domain and boundaries with help of `SDIndexes`.
	`SDMaskIndexes`	produces indexes for various curvilinear domains or boundary depending on the codes of the nodes.
`SDOperator`		make all operations with FDM: from discretisation of the PDE to output of the results. The `SDOperator` contains an association to an `SDIndexes` object and contain subclasses:
	`DplusDminusOp`	supplies DplusDminus-stencil in the prescribed direction
	`DplusDminusVec2Op`	supplies DplusDminus-stencil in the prescribed direction
	`LaplacianOp`	supplies five-point-stencil in 2d and seven-point-stencil in 3d
	`DirichletOp`	supplies Dirichlet BCs. The value is a constant.
	`AssignmentOp`	assigns a value to y.
	`AssignFunctionOp`	assigns values to y as a function of the coordinates
	`AssignTimeFunctionOp`	assigns values to y as a function of space and time
	`ComputeTimeDerivative`	compute time derivative for y using a formula $(u^{k-2} + 3u^k - 4 * u^{k-1})/2dt$
	`ApplyFunctionOp`	adds values of the two functions in points of the FD grid
	`OutputOp`	outputs for y
	`AVSOutputOp`	outputs the grid and the value of y
	`DifferenceCheckOp`	is a simple operator that compares x[IX] with y[IY]. The indexes IX and IY are supplied to the constructor.
	`WaveEqInteriorexpl1`	to compute the wave equation as system, use explicit scheme, solve first equation of system
	`WaveEqInteriorexpl2`	to compute the wave equation as system, use explicit scheme, solve second equation of system

Chalmers Finite Element Center Preprints

2000–01	*Adaptive Finite Element Methods for the Unsteady Maxwell's Equations* Johan Hoffman
2000–02	*A Multi-Adaptive ODE-Solver* Anders Logg
2000–03	*Multi-Adaptive Error Control for ODEs* Anders Logg
2000–04	*Dynamic Computational Subgrid Modeling* (Licentiate Thesis) Johan Hoffman
2000–05	*Least-Squares Finite Element Methods for Electromagnetic Applications* (Licentiate Thesis) Rickard Bergström
2000–06	*Discontinuous Galerkin Methods for Incompressible and Nearly Incompressible Elasticity by Nitsche's Method* Peter Hansbo and Mats G. Larson
2000–07	*A Discountinuous Galerkin Method for the Plate Equation* Peter Hansbo and Mats G. Larson
2000–08	*Conservation Properties for the Continuous and Discontinuous Galerkin Methods* Mats G. Larson and A. Jonas Niklasson
2000–09	*Discontinuous Galerkin and the Crouzeix-Raviart element: Application to elasticity* Peter Hansbo and Mats G. Larson
2000–10	*Pointwise A Posteriori Error Analysis for an Adaptive Penalty Finite Element Method for the Obstacle Problem* Donald A. French, Stig Larson and Ricardo H. Nochetto
2000–11	*Global and Localised A Posteriori Error Analysis in the Maximum Norm for Finite Element Approximations of a Convection-Diffusion Problem* Mats Boman
2000–12	*A Posteriori Error Analysis in the Maximum Norm for a Penalty Finite Element Method for the Time-Dependent Obstacle Problem* Mats Boman
2000–13	*A Posteriori Error Analysis in the Maximum Norm for Finite Element Approximations of a Time-Dependent Convection-Diffusion Problem* Mats Boman
2001–01	*A Simple Nonconforming Bilinear Element for the Elasticity Problem* Peter Hansbo and Mats G. Larson
2001–02	*The $\mathcal{LL}^*$ Finite Element Method and Multigrid for the Magnetostatic Problem* Rickard Bergström, Mats G. Larson, and Klas Samuelsson
2001–03	*The Fokker-Planck Operator as an Asymptotic Limit in Anisotropic Media* Mohammad Asadzadeh
2001–04	*A Posteriori Error Estimation of Functionals in Elliptic Problems: Experiments* Mats G. Larson and A. Jonas Niklasson

2001–05	*A Note on Energy Conservation for Hamiltonian Systems Using Continuous Time Finite Elements* Peter Hansbo
2001–06	*Stationary Level Set Method for Modelling Sharp Interfaces in Groundwater Flow* Nahidh Sharif and Nils-Erik Wiberg
2001–07	*Integration methods for the calculation of the magnetostatic field due to coils* Marzia Fontana
2001–08	*Adaptive finite element computation of 3D magnetostatic problems in potential formulation* Marzia Fontana
2001–09	*Multi-Adaptive Galerkin Methods for ODEs I: Theory & Algorithms* Anders Logg
2001–10	*Multi-Adaptive Galerkin Methods for ODEs II: Applications* Anders Logg
2001–11	*Energy norm a posteriori error estimation for discontinuous Galerkin methods* Roland Becker, Peter Hansbo, and Mats G. Larson
2001–12	*Analysis of a family of discontinuous Galerkin methods for elliptic problems: the one dimensional case* Mats G. Larson and A. Jonas Niklasson
2001–13	*Analysis of a nonsymmetric discontinuous Galerkin method for elliptic problems: stability and energy error estimates* Mats G. Larson and A. Jonas Niklasson
2001–14	*A hybrid method for the wave equation* Larisa Beilina, Klas Samuelsson, Krister Åhlander

These preprints can be obtained from

`www.phi.chalmers.se/preprints`