Efficiency of a Hybrid Method for the Wave Equation

Larisa Beilina* Klas Samuelsson* Krister Åhlander§

20th December 2000

Abstract

A hybrid method for simulating partial differential methods is studied. The 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 feasibility of the hybrid approach is illustrated by the wave equation in two and three space dimensions.

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

It is shown how to construct explicit finite difference schemes and finite element schemes that 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 discretization 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 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 optimized 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 [5, 6].

*Finite Element Centre, Chalmers University of Technology, Göteborg, Sweden, Email: {larisa,klassam}@math.chalmers.se.

§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 discretize the scalar wave equation in time with explicit methods.

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 discretizations 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

$$\frac{\partial^2 u}{\partial t^2} = \nabla \cdot (a^2 \nabla u) + f, \qquad x \in \Omega \subset \mathbf{R}^n,\ t > 0, \tag{1}$$

where $u(x,t)$ is unknown, a is the wave velocity, x ranges over the points of the space domain, t is time, f is a given source function, $n = 2, 3$ is the number of space dimensions, and Ω denotes a bounded domain. Appropriate initial and boundary conditions are assumed. In [3], we also study discretizations of the wave equation when it is rewritten as a system of first order equations. This is achieved with the substitutions $u_1 = \frac{\partial}{\partial t} u$ and $u_2 = u$.

In many wave equation applications, only a small part of the computational domain Ω is complex enough to motivate a more complex unstructured discretization, whereas quite large regions of the computational domain are sufficiently discretized 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

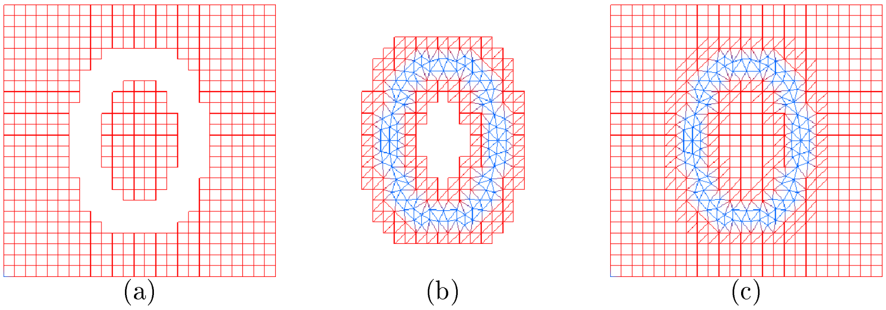


Figure 1: Domain decomposition. The hybrid mesh (c) is a combination of the structured mesh (a) and the unstructured mesh (b) with a thin overlapping of structured elements. The unstructured grid is constructed so that the grid contains edges approximating an ellipse.

discretization is appropriate. In the Ω_{FDM} domain, we assume that a structured Cartesian grid is suitable. Fig. 1 illustrates the principle in two dimensions. The used 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 discretization 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 [7]. At the 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 discretization 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},

$\omega_\times$: Nodes interior to Ω_{FEM} and boundary nodes to Ω_{FDM},

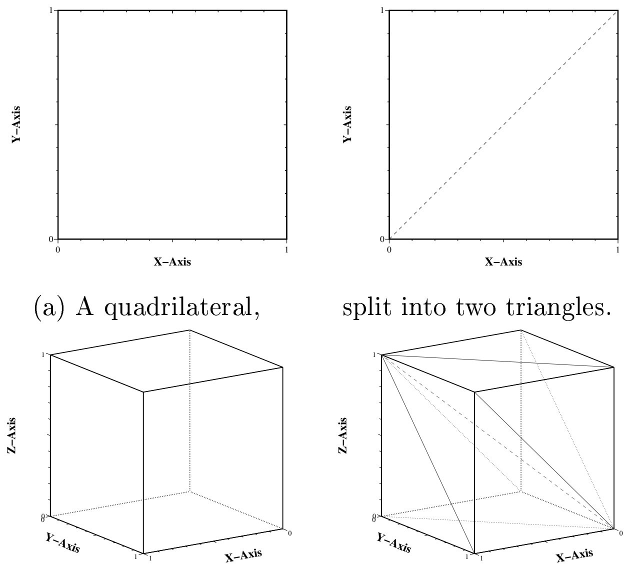
ω_* : Nodes interior to Ω_{FEM} and not contained in Ω_{FDM},

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 discretization. The exterior and the interior of the ellipse may use a structured discretization. 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:

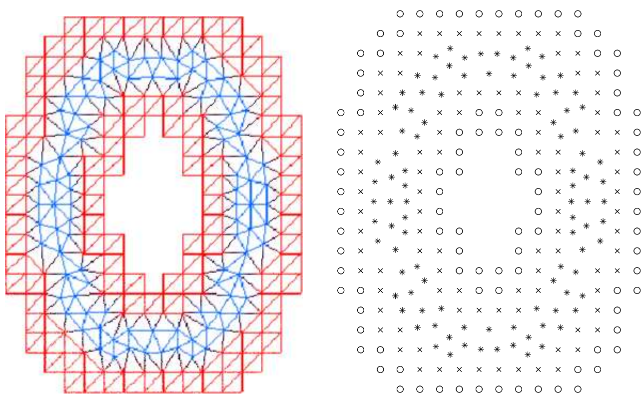
For every time step



(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, whereas 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.

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 (1) we use the standard Galerkin finite element method with linear tetrahedral elements in space and a centralised finite difference approximation for the second order time derivative. After discretization, see [3, 8, 9] for more details, the following system of linear equations is derived:

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

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, $\mathbf{u}^k$ is the unknown discrete field values of u at k time level, and τ is the time step.

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. [9]. Multiplying (2) with $(M^L)^{-1}$, we obtain an efficient explicit formulation:

$$\mathbf{u}^{k+1} \quad = \quad \tau^2 (M^L)^{-1} F^k + 2\mathbf{u}^k - \tau^2 (M^L)^{-1} K\mathbf{u}^k - \mathbf{u}^{k-1}. \tag{3}$$

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.

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

$$\tau \leq \frac{h}{ac} \quad , \tag{4}$$

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 (1) we obtain

$$u^{l+1}_{i,j,k} = \tau^2 (f^l_{i,j,k} + a^2 \Delta u^l_{i,j,k}) + 2u^l_{i,j,k} - u^{l-1}_{i,j,k}, \tag{5}$$

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

$$\begin{aligned}\Delta v^l_{i,j,k} \quad &= \quad \frac{v^l_{i+1,j,k} - 2v^l_{i,j,k} + v^l_{i-1,j,k}}{dx^2} + \frac{v^l_{i,j+1,k} - 2v^l_{i,j,k} + v^l_{i,j-1,k}}{dy^2} + \\ &\quad \frac{v^l_{i,j,k+1} - 2v^l_{i,j,k} + v^l_{i,j,k-1}}{dz^2}, \end{aligned} \tag{6}$$

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

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 (5) is performed by looping over the index sets and applying the finite difference molecule. See [3] 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 (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);     // initialize with grids
                                       // for FEM and FDM
dt = p.initTime(t);                    // get time step and init time
p.initFDM();                           // initialize FDM
p.initFEM();                           // initialize 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 initialization, 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 optimizations

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 optimize the implementation of the involved methods. Here, we comment on some optimizations.

First, observe that in (3) 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 initialization step. A similar optimization is used for FDM.

Regarding the computation of the load vector F^k in (3), a standard assembly computation using Gaussian quadrature is expensive. By instead using nodal quadrature, the load vector will be $F^k = M_L\mathbf{f}^k$, 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

$$\mathbf{u}^{k+1} = \tau^2\mathbf{f}^k + A\mathbf{u}^k - \mathbf{u}^{k-1}. \tag{7}$$

Note that the FDM scheme (5) coincides with (7) in regions where the grid is structured.

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. See [3] for more details on the optimization issues.

Regarding the finite difference region, we use Cartesian grids which allocate memory also in the Ω_{FEM} region, but the actual calculations are only performed in the Ω_{FDM} region. The advantage is that we can use direct addressing over all the domain. For our test domains, we found that this strategy gave the best performance. See [3] for more details. Compare also with [1, 4] which are dealing with optimizing C++ code and studying the effect of the cache sizes of the computer, respectively.

5 Numerical examples

We illustrate the use of the hybrid method in several examples. In all examples we simulate model problem (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:

$$f_1(x, t; x_0) = \begin{cases} 10^3 \sin^2 \pi t & \text{if } 0 \leq t \leq 0.1 \text{ and } |x - x_0| < 0.1, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

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 Application examples

In the first example we solve problem (1) with Dirichlet boundary conditions. We use (3) to implement FEM and (5) to implement FDM. We choose the source function f_1 with $x_0 = (0.5, 0.5)$, the centre of the Ω_{FEM} domain, and we use time step $\tau \approx 0.000097$, computed from criterion (4) with the constant $c = 0.1$. We plot the hybrid method solution in Fig. 4.

In order to validate the hybrid method, we have studied the solution at a specific point over time when using the different methods. The solution is plotted in Fig. 5. Similarly, for computations on a structured three dimensional grid, computations using the three methods, FEM, FDM, and hybrid, give identical results.

In Fig. 6, we present a computation where two pulses f_1 are generated, one located in the center of the domain and one located at $x_0 = (0.5, 0.3, 0.5)$, in the Ω_{FDM} domain.

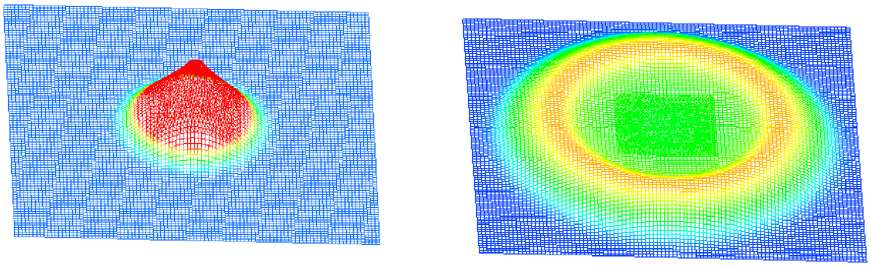
5.2 Performance comparisons

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



a) time is 0.3884 b) time is 0.5827

Figure 4: Example of the hybrid method for the two-dimensional wave equation with a pulse f_1 generated at $x_0 = (0.5, 0.5)$.

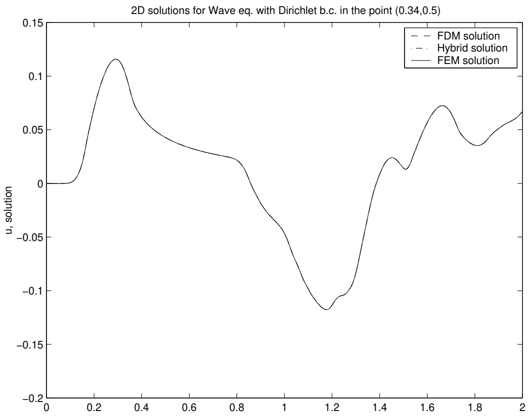


Figure 5: Solutions for the two dimensional wave equation at point (0.34, 0.5). The three solutions of the FEM, FDM and hybrid solutions are identical on the whole time range. This is due to that the FEM grid is structured.

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.

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.

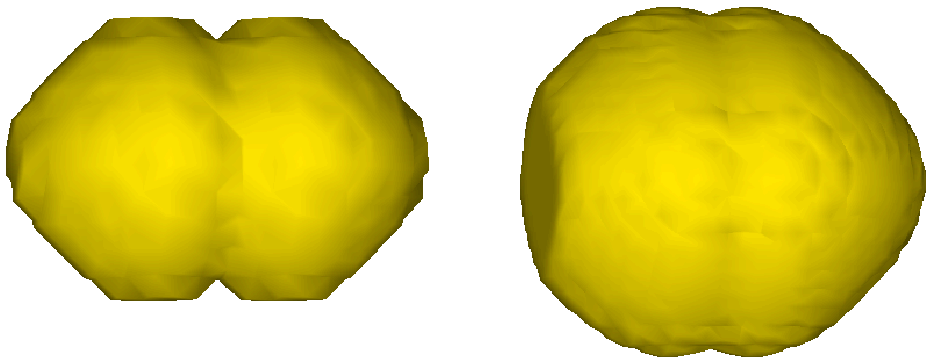
5.3 Remarks on the performance comparisons

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.

5.4 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.



time is 0.105 time is 0.175

Figure 6: Solutions of the three dimensional wave equation, using two source pulses.

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 discretization 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.

One of the most important questions of this study is whether the hybrid approach really gives better performance, compared to pure FEM. Our tests for structured grids indicate 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 discretized 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] L. Beilina, K. Samuelsson, K. Åhlander. A hybrid method for the wave equation. Technical report under preparation, Chalmers Finite Element Center (2000).

[4] 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)

[5] 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).

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

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

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

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