Improved GdfidL with Generalized Diagonal Fillings and Reduced Memory and CPU Requirements

W. Bruns*, Technische Universität Berlin, EN-2, Berlin, Germany

Abstract

A new version of the Finite Difference code GdfidL implements generalized diagonal fillings for the discretization of the material distribution. The new algorithm allows more than 72 different types of material filling for each cell, whereas the common diagonal filling only allows 7 types. With the improved material filling, the discretization error for realistic geometries is reduced by a factor of ten. The improved meshing is implemented both in the resonant solver and the time domain solver. Computed frequencies for some simple geometries are given to show the reduction in discretization error. GdfidL's organisation of the computational volume as a linked list has already reduced the resource requirements to about the half of other codes. A new organisation of the linked list reduces the requirements for time domain computations again by a factor of 0.7.

1 INTRODUCTION

We want to compute electromagnetic fields. We want them accurate, we don't want to wait too long for them and we want to compute them within the memory limits of our computers. How to do this?

The first part deals with the accuracy. The next parts introduce a simple scheme to speed up both the FDTD-algorithm and the curl-curl-operator which is applied to find eigenfrequencies and fields.

2 ACCURACY

We can distinguish two kinds of error: The error in the approximation of the differential equation by the difference equation, and the approximation of the boundary conditions by the discretized material fillings. In order to have a small total error one has to have both errors at about the same size. When an (almost) homogeneous mesh is used, the error in the solution is proportional to the square of the mesh spacing. → We want to keep the error in the material filling at about the same size. There is little gained when the boundary conditions are discretized "perfectly" (except the mesh might look better). Near the boundaries, though, a "perfect"-mesh will be much better than what I present here.

How large is the error associated with the approximation of the differential equation by difference equations? For example, the error in a discretized rectangular cavity originates only from the error in the approximation of the fields, not from the approximation of the boundaries: Figure 1 shows the computed field in such a rectangular geometry
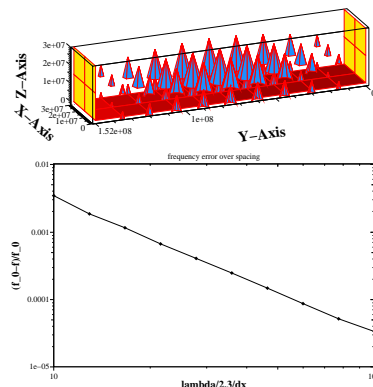
* bruns@tetibm2.ee.TU-Berlin.DE

Figure 1: Above: Field in a simple rectangular resonator, Below: The error in the computed frequency. The error is proportional to Δ^2.
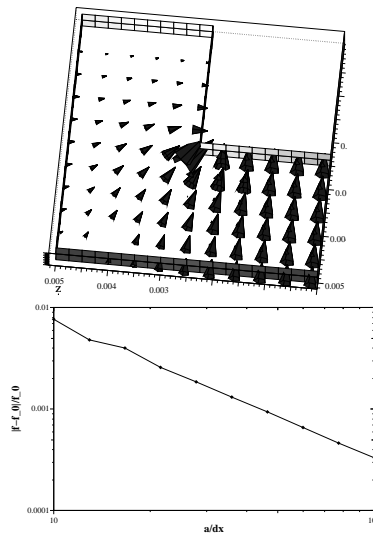

Figure 2: Above: Resonatorfield with singularity, Below: The error in the computed frequency. The error is proportional to $\Delta^{4/3}$.

and the error in the computed frequency. The frequency error is of second order, ie. when decreasing the stepsize by a factor of 10, the error is decreased by a factor of 100. This is the best possible error with standard Finite Differences.

In contrast to this, when one computes the field in a resonator with sharp edges of angle 90 degrees, the error is only of order 4/3, since the basis functions of the finite difference approach cannot model the field singularity well. Figure 2 shows such a geometry and the error in the computed frequency.

2.1 Improved mesh filling

Material-fillings are boundary conditions for the differential equations. The simpler finite difference programs approximate these boundary conditions by using a material filling where every grid cell is assumed to be homogeneously filled with a single material. This is the "staircase" approximation. A better approximation of the boundary conditions can be achieved with prismatic cells, as e.g. the MAFIA [3] group of codes uses them.

The filling with prismatic cells can be generalized. Since the finite difference coefficients for a field component depend only on the material in the immediate vicinity of the edge where the component is defined on, one can work easily with a mesh-filling that is constructed by a boolean combination of prismatic fillings. Figure 3 shows some of the possible discretized material distributions. A similiar mesh filling is mentioned in [2]. Figure 4 sketches the procedure to evaluate the FD-coefficient for some field component.
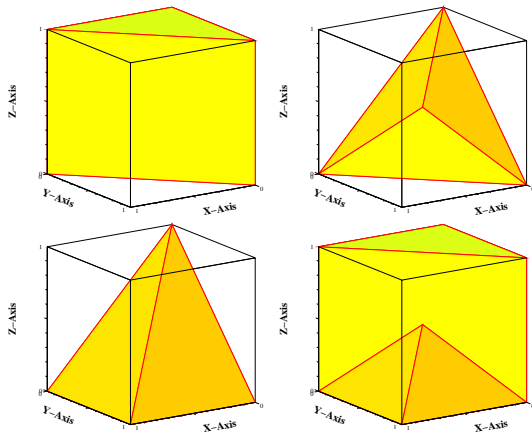

Figure 3: Some examples of the possible inhomogeneous fillings of a cell. Upper left: a prism. lower left: Intersection of two prisms. Upper right: Intersection of three prisms. lower right: Union of "upper left" and "lower left". The prism in the upper left can be oriented in 2 x 3 different kinds in a cell, the other three material fillings are possible in 4 x 3 x 2 different orientations.
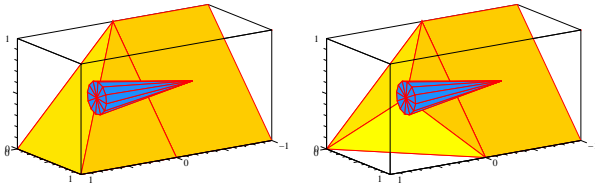

Figure 4: A magnetic field component touching some material boundaries. To evaluate the FD-coefficient for this component, one has to evaluate the effectice permeability along the path where the component is defined on. The Finite Difference coefficient for *this* component is the same in both cases.
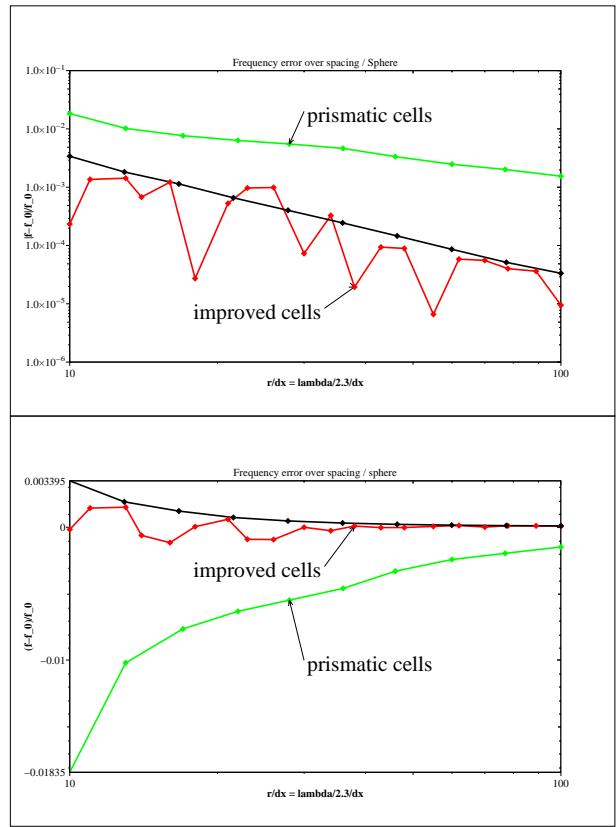

Figure 5: Error in the computed frequency of the lowest mode in a sphere. Above: absolute values in a double logarithmic scale, below: signed values in single logarithmic scale.

2.2 Examples, Effect of the new meshing

In order to show the effect of the generalized prismatic filling, figure 5 shows the computed resonance frequency in a sphere as a function of the mesh-spacing. For comparison, the results for prismatic filling and the optimal quadratic behaviour is plotted also. The error with the improved filling is about as low as the optimal quadratic behaviour. If the boundary conditions, ie. the materials would have been discretized perfectly, the result would not be much better.

A more interesting geometry is a reentrant cavity. Figure 6 shows the computed resonance frequency in such a geometry as a function of the mesh-spacing. For comparison, the results for prismatic filling is plotted also. Here the frequency error is not of second order. One can blame this on the small radius of the nose, where the field behaviour is almost singular. The frequency error is comparable to the error as shown in figure 2.
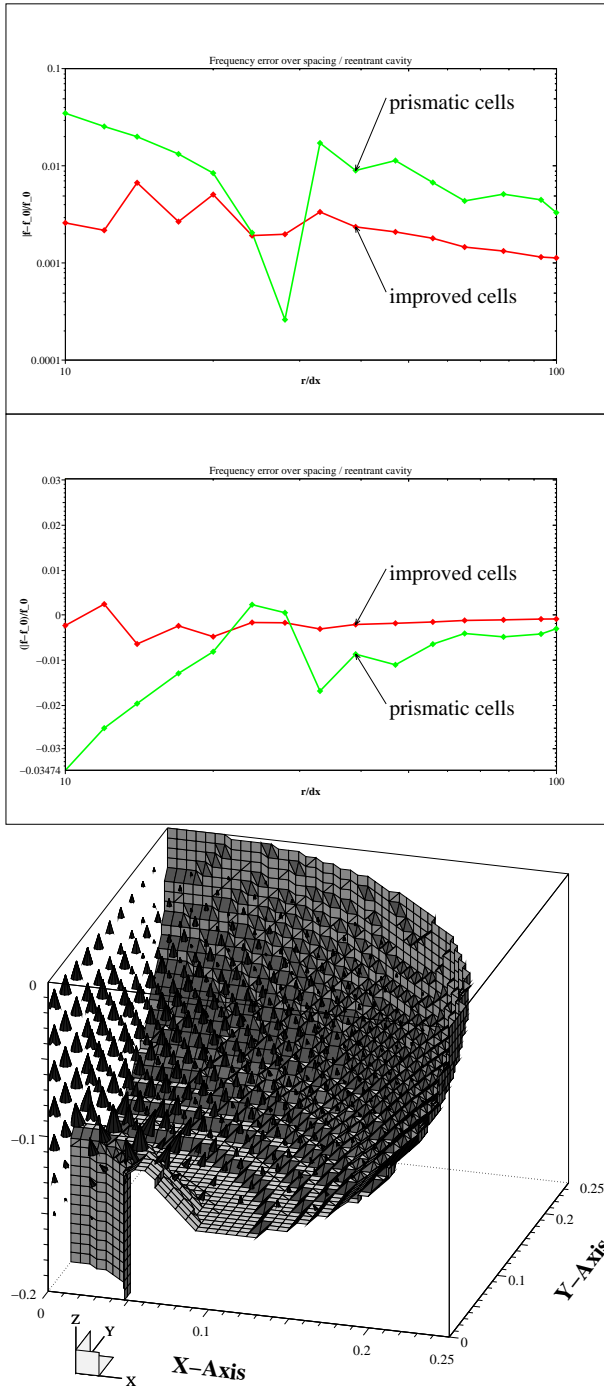
Figure 6: Error in the computed frequency of the first mode in a reentrant cavity. Above: absolute values in a double logarithmic scale, Center: signed errors versus planes / radius in logarithmic scale. Below: The geometry as discretized with 40 meshplanes / radius.

3   FASTER FD-IMPLEMENTATION

In addition to the commonly known FD-optimization that computes with $\int H ds$ and $\int E ds$ instead of H and E [1] two algorithmic improvements have been found and implemented.

3.1   *Single sweep through memory*

Normally the H- and E-Update in FDTD-codes are performed as follows:

```
do
    Update all H-components,
    Update all H-components,
enddo
```

The above algorithm in every timestep reads all the E components to update all H-components, then reads all the H-components to update all E components. → In every timestep, all field-components are touched twice, but with a long CPU-time distance between.

Careful inspection of the FDTD-update reveals that it is possible to update the H components of a cell, and immediately after that the E-components can be updated, since they are no longer needed to update other H-components. → The second use of the field components will be much faster on cache-based computers, since the field components will already be in the cache. The old E-components can be overwritten, if the mesh is traversed e.g. in the natural order. E.g. for lossfree problems it might look like:

```
REAL, DIMENSION(1:3,0:nx+1,0:ny+1,0:nz+1) :: &
    Eds, Hds, dsoEpsA, dsoMueA
DO iz= 1, nz, 1
    DO iy= 1, ny, 1
        DO ix= 1, nx, 1
Hds(1,ix,iy,iz)= Hds(1,ix,iy,iz) - dt*dsoMueA(1,ix,iy,iz) &
    * ( Eds(2,ix ,iy ,iz )-Eds(2,ix ,iy ,iz+1) &
    + Eds(3,ix ,iy ,iz )-Eds(3,ix ,iy+1,iz ) )
Hds(2,ix,iy,iz)= Hds(2,ix,iy,iz) - dt*dsoMueA(2,ix,iy,iz) &
    * (-(Eds(1,ix ,iy ,iz )-Eds(1,ix ,iy ,iz+1)) &
    + Eds(3,ix ,iy ,iz )-Eds(3,ix+1,iy ,iz ) )
Hds(3,ix,iy,iz)= Hds(3,ix,iy,iz) - dt*dsoMueA(3,ix,iy,iz) &
    * ( Eds(1,ix ,iy ,iz )-Eds(1,ix ,iy+1,iz )&
    -(Eds(2,ix ,iy ,iz )-Eds(2,ix+1,iy ,iz )))
!!
!! Now we can update the Eds-components of the cell since
!! they are no longer needed for the remaining H-updates.
!! All used H-components have already been updated:
!!
Eds(1,ix,iy,iz)= Eds(1,ix,iy,iz) + dt*dsoEpsA(1,ix,iy,iz) &
    * (-(Hds(2,ix ,iy ,iz ) - Hds(2,ix ,iy ,iz-1)) &
    + Hds(3,ix ,iy ,iz ) - Hds(3,ix ,iy-1,iz ) )
Eds(2,ix,iy,iz)= Eds(2,ix,iy,iz) + dt*dsoEpsA(2,ix,iy,iz) &
    * ( Hds(1,ix ,iy ,iz ) - Hds(1,ix ,iy ,iz-1) &
    -(Hds(3,ix ,iy ,iz ) - Hds(3,ix-1,iy ,iz )))
Eds(3,ix,iy,iz)= Eds(3,ix,iy,iz) + dt*dsoEpsA(3,ix,iy,iz) &
    * (-(Hds(1,ix ,iy ,iz ) - Hds(1,ix ,iy-1,iz )) &
    + Hds(2,ix ,iy ,iz ) - Hds(2,ix-1,iy ,iz ) )
        ENDDO
    ENDDO
ENDDO
```

A similiar optimization can be applied in the discretized curl-curl operator that is used for eigenvalue computation. These optimizations save about 30 % CPU-time on typical desktop computers.

3.2 Better grid representation

The Finite Difference algorithm can be easily implemented as a computer code if one represents the fields as 4-D arrays. An example how simple the FDTD-update procedure might look was shown in the previous section. This approach has the disadvantage that memory (and CPU) is also used for cells that are totally inside perfect electric or magnetic materials. For many realistic geometries, the volume occupied by electric material is half of the total discretized volume or more.

One wants to use some scheme to only deal with the cells that really have a nonzero field.

One possible scheme could be the use of linked lists, where every field carrying cell carries the information about its neighbours. This needs 6 indices per field cell in addition to the 12 floating point words for the field components and their coefficients. This approach was used in the former "GdfidL" [4] [5].

The improved "GdfidL" has a grid organization that needs only a single index per cell, but also does not compute and store field components of cells that are totally surrounded by perfect electric or magnetic materials. The fields itself are stored in 2D-arrays, where the second index is the number of the cell. An INTEGER array "NrofCell" is used to extract the topology information. The FDTD-update with the single index per cell might look like:

```
REAL,DIMENSION(1:3,0:*) :: &
    Eds, Hds, dsoEpsA, dsoMueA
INTEGER, DIMENSION(0:nx+1,0:ny+1,0:nz+1) :: &
    NrofCell
DO iz= 1, nz, 1
    DO iy= 1, ny, 1
        DO ix= 1, nx, 1
i= NrofCell(ix,iy,iz)
IF (i .LT. 1) CYCLE !! skip when no field possible
!!
!! indices of neighbour cells in positive directions
!!
        ipx= NrofCell(ix+1,iy ,iz )
        ipy= NrofCell(ix ,iy+1,iz )
        ipz= NrofCell(ix ,iy ,iz+1)
Hds(1,i)= Hds(1,i) - dt*dsoMueA(1,i) &
    * ( Eds(2,i)-Eds(2,ipz) + Eds(3,i)-Eds(3,ipy) )
Hds(2,i)= Hds(2,i) - dt*dsoMueA(2,l) &
    * (-(Eds(1,i)-Eds(1,ipz)) + Eds(3,i)-Eds(3,ipx) )
Hds(3,i)= Hds(3,i) - dt*dsoMueA(3,i) &
    * ( Eds(1,i)-Eds(1,ipy) -(Eds(2,i)-Eds(2,ipx)))
!!
!! indices of neighbour cells in negative directions
!!
imx= NrofCell(ix-1,iy ,iz )
imy= NrofCell(ix ,iy-1,iz )
imz= NrofCell(ix ,iy ,iz-1)
Eds(1,i)= Eds(1,i) + dt*dsoEpsA(1,i) &
    * (-(Hds(2,i) - Hds(2,imz)) + Hds(3,i) - Hds(3,imy) )
Eds(2,i)= Eds(2,i) + dt*dsoEpsA(2,l) &
    * ( Hds(1,i) - Hds(1,imz) -(Hds(3,i) - Hds(3,imx)))
Eds(3,i)= Eds(3,i) + dt*dsoEpsA(3,i) &
    * (-(Hds(1,i) - Hds(1,imy)) + Hds(2,i) - Hds(2,imx) )
        ENDDO
    ENDDO
ENDDO
```

4 CONCLUSION

An improved mesh filling has been implemented that reduces the frequency error by a factor of ten as compared to a prismatic filling. The frequency error from the boundary approximation is now in the range of the error due to the discretization of the differential equation itself. The time consuming field updates have been further optimized for modern cache based computers, giving a speed improvement of about 30% over the former GdfidL. The actual GdfidL is now typically 10 times as fast as a conventional FD-program. The memory consumption of GdfidL has been reduced again by a factor of 0.7. The memory consumption is typically only a quarter of that of conventional FD programs.

5 REFERENCES

[1] Th. Weiland, "Ein Verfahren zur Berechnung von Wirbelströmen in massiven, dreidimensionalen, beliebig geformten Eisenkörpern", *etz Archiv*, H. 9 (1979), pp. 263-267

[2] W. Müller, J. Krueger, A. Jacobus. R. Winz, T. Weiland, H. Euler, U. Hamm, W. R. Novender, "Numerical Solution of 2- or 3-Dimensional Nonlinear Field Problems by Means of the Computer Program PROFI", *Archiv für Elektrotechnik 65* (1982) pp. 299-307

[3] F. Ebeling, R. Klatt, F. Krawczyk, E. Lawinsky, T. Weiland, S. G. Wipf, B. Steffen, T. Barts, M. J. Browman, R. K. Cooper, H. Deaven, G. Rodenz, "The 3-D MAFIA group of electromagnetic codes", *IEEE Transactions on Magnetics*, vol. 25, pp. 2962-2964, July 1989

[4] W. Bruns, "GdfidL: A Finite Difference Program for Arbitrarily Small Perturbations in Rectangular Geometries", *IEEE Transactions on Magnetics*, vol. 32, no. 3, May 1996, pp. 1453-1456

[5] W. Bruns, "GdfidL: A Finite Difference Program with Reduced Memory and CPU Usage", *Proceedings of the PAC-97, Vancouver*, vol. 2, pp. 2651-2653, http://www.triumf.ca/pac97/papers/pdf/9P118.PDF