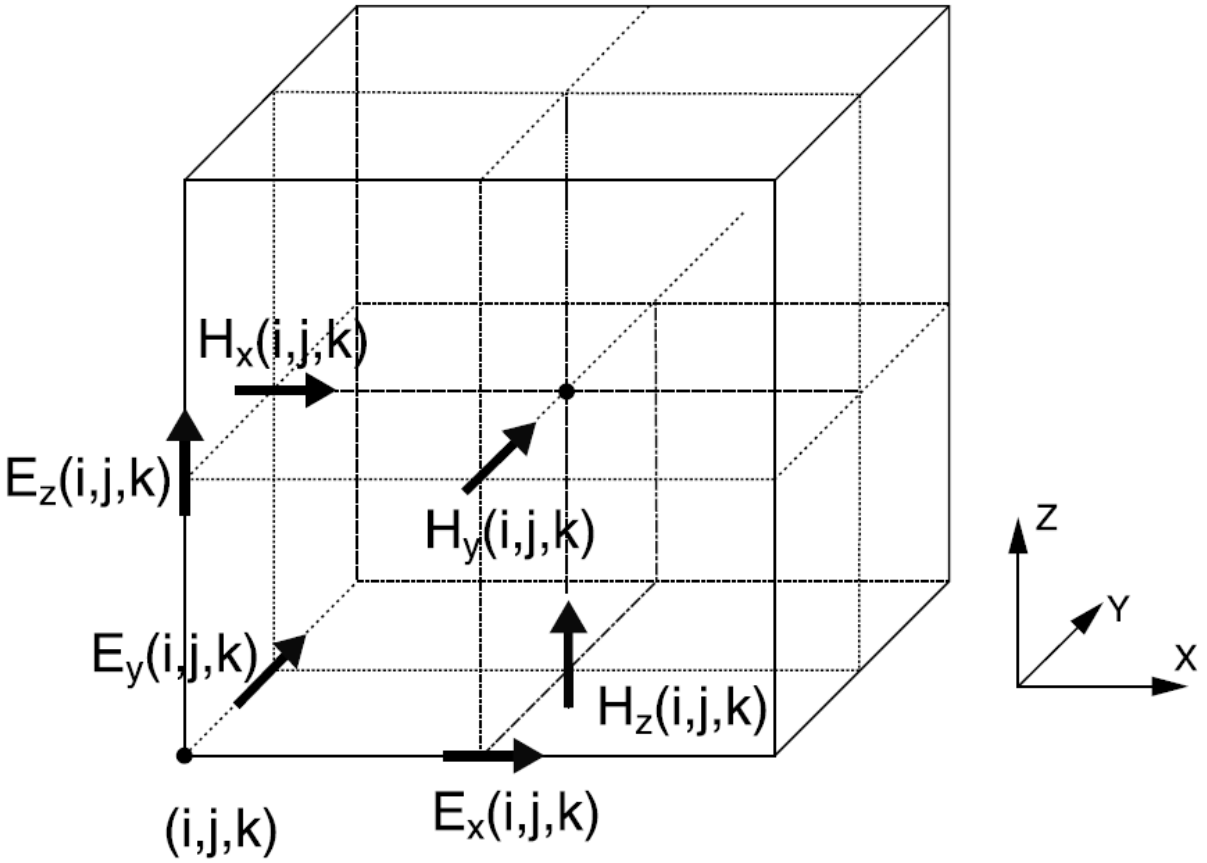
Программная реализация метода конечных разностей во временной области для графических процессоров с использованием OpenCL

Нагорный Н. А., 4 курс

Научный руководитель: Кретов П. А.

Воронежский государственный университет
Физический факультет, кафедра электроники

Сетка И

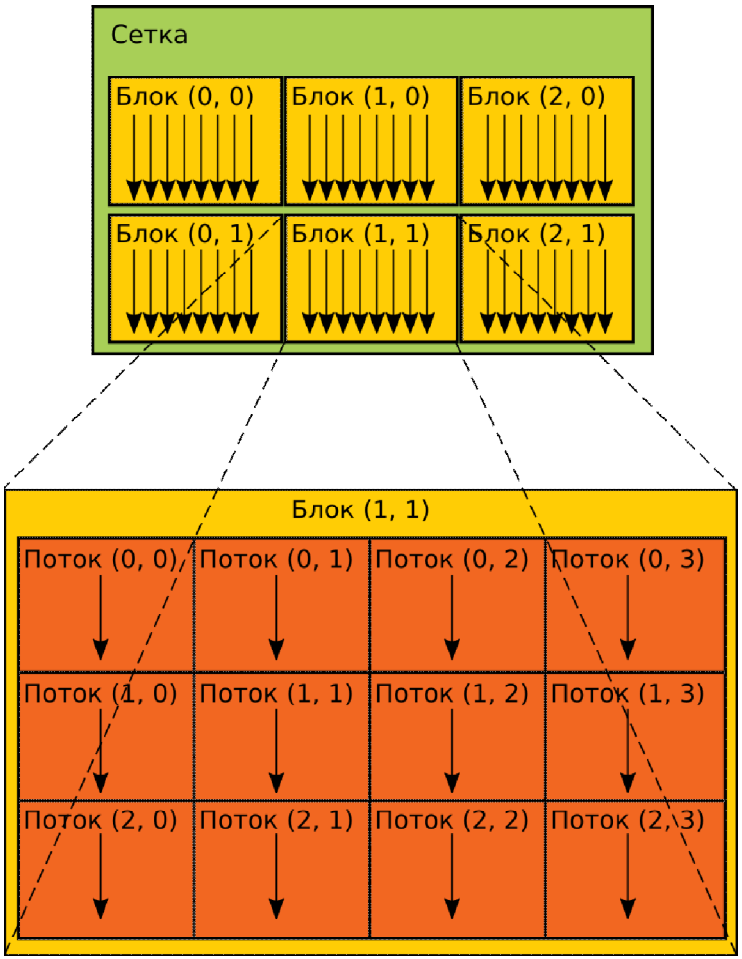


Базовые уравнения FDTD

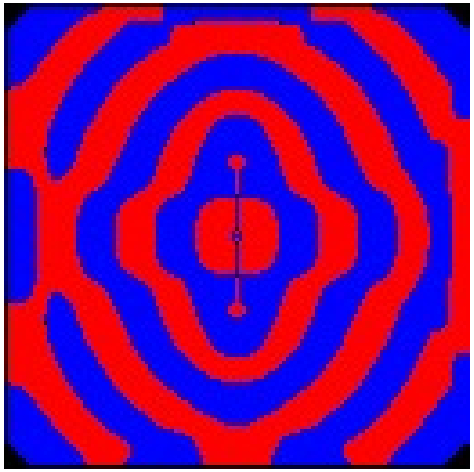
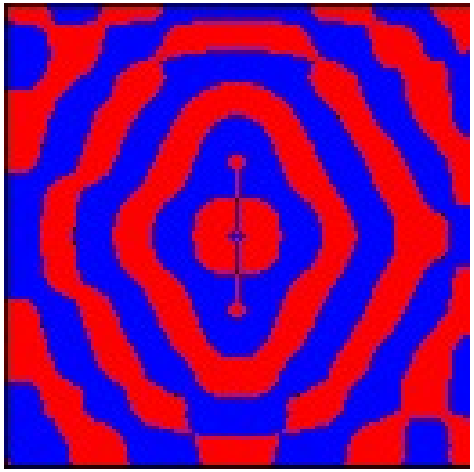
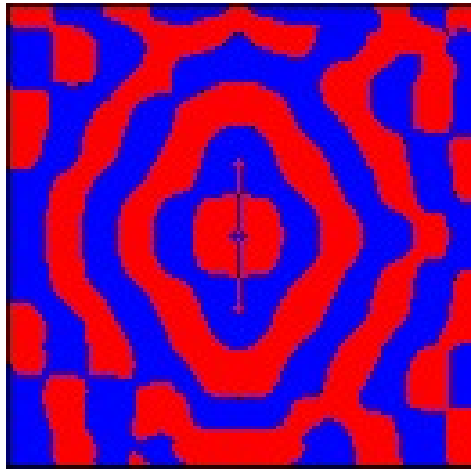
Для компонент магнитного поля:

- $$H_x|_{i,j,k}^{n+1/2} = H_x|_{i,j,k}^{n-1/2} - \frac{\frac{\Delta t}{\mu\mu_0}}{1+\frac{\sigma_H \Delta t}{2\mu\mu_0}}\left[\frac{E_z|_{i,j+1,k}^{n} - E_z|_{i,j,k}^{n}}{\Delta y} - \frac{E_y|_{i,j,k+1}^{n} - E_y|_{i,j,k}^{n}}{\Delta z}\right],$$
- $$H_y|_{i,j,k}^{n+1/2} = H_y|_{i,j,k}^{n-1/2} - \frac{\frac{\Delta t}{\mu\mu_0}}{1+\frac{\sigma_H \Delta t}{2\mu\mu_0}}\left[\frac{E_x|_{i,j,k+1}^{n} - E_x|_{i,j,k}^{n}}{\Delta z} - \frac{E_z|_{i+1,j,k}^{n} - E_z|_{i,j,k}^{n}}{\Delta x}\right],$$
- $$H_z|_{i,j,k}^{n+1/2} = H_z|_{i,j,k}^{n-1/2} - \frac{\frac{\Delta t}{\mu\mu_0}}{1+\frac{\sigma_H \Delta t}{2\mu\mu_0}}\left[\frac{E_y|_{i+1,j,k}^{n} - E_y|_{i,j,k}^{n}}{\Delta x} - \frac{E_x|_{i,j+1,k}^{n} - E_x|_{i,j,k}^{n}}{\Delta y}\right].$$

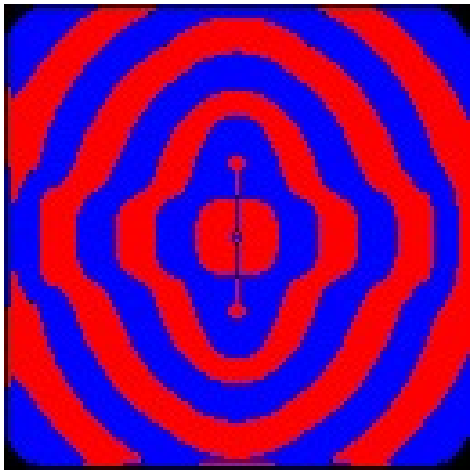
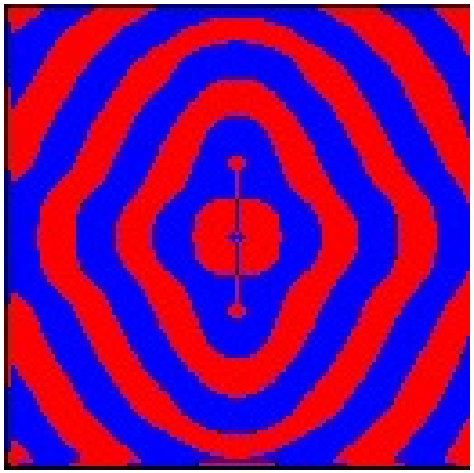
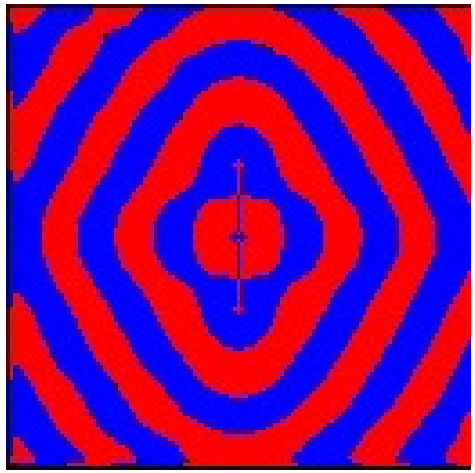
Потоки выполнения CUDA



Динамика изменения поля

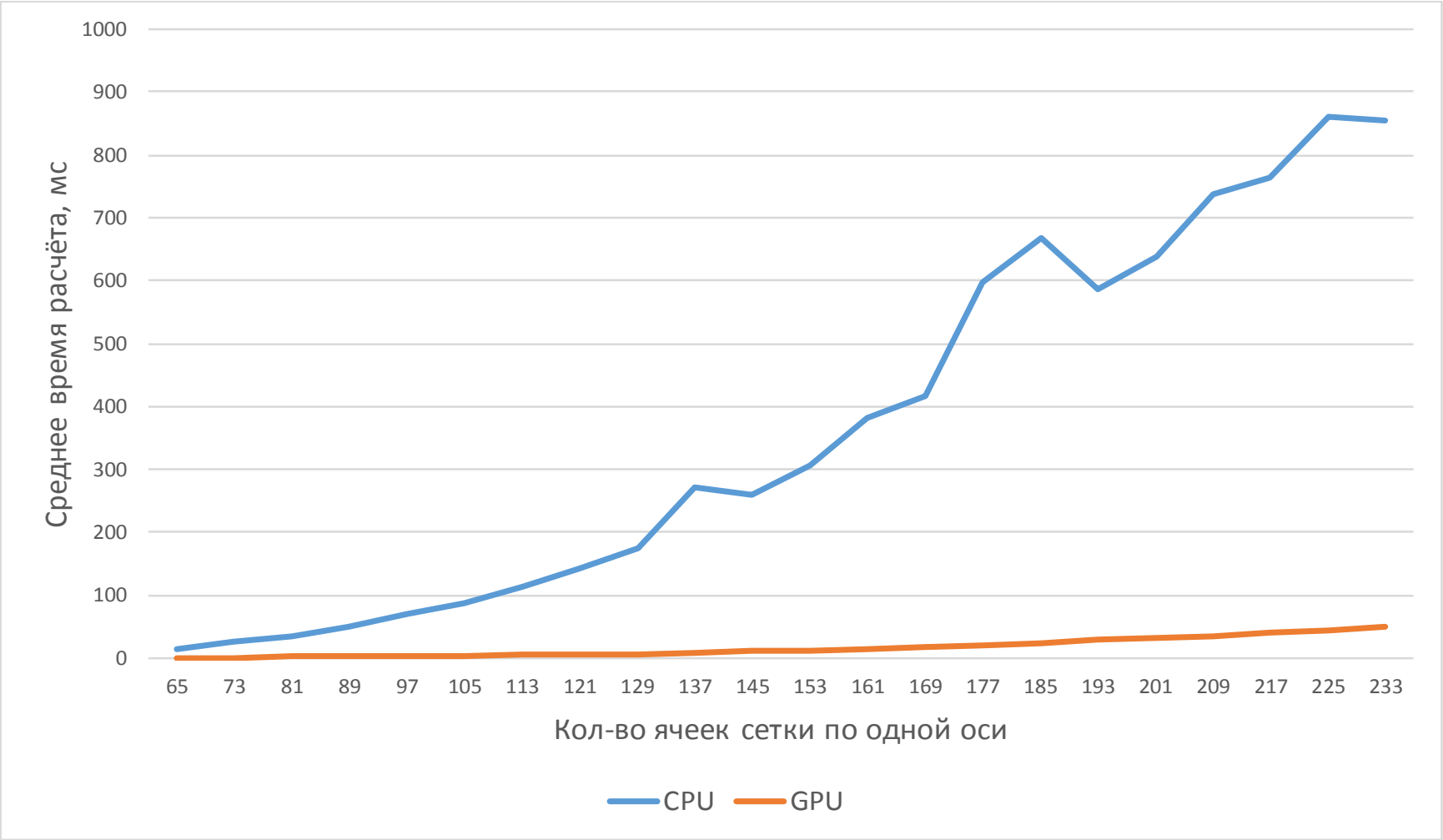




Условия идеального отражения

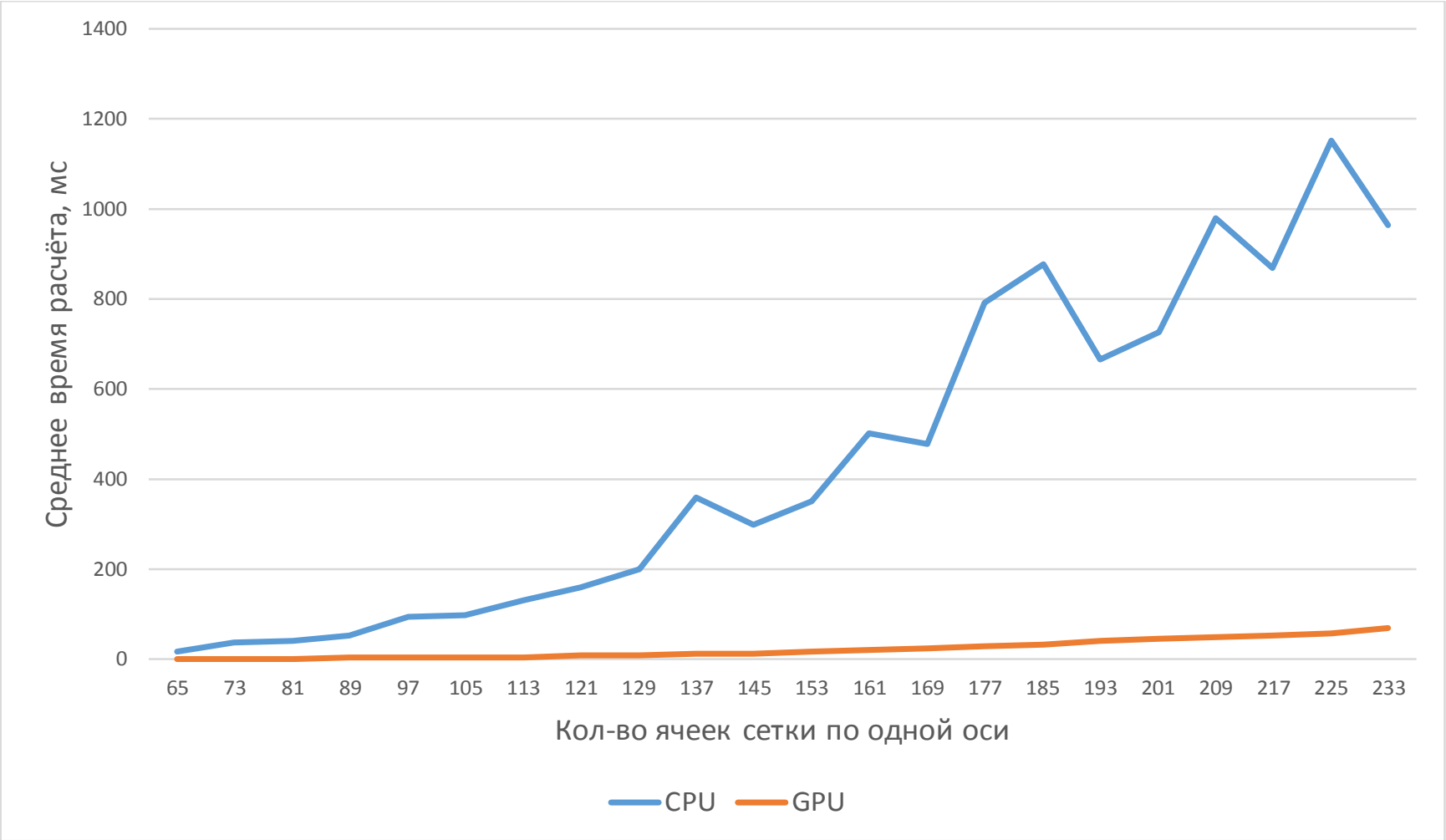




PML

Зависимость среднего времени расчёта компонент вектора H во всех ячейках в конкретный момент времени от размеров сетки



Зависимость среднего времени расчёта компонент вектора E во всех ячейках в конкретный момент времени от размеров сетки



Код расчёта одной компоненты вектора H во всех точках поля в конкретный момент времени на CPU

```
int nx = grid.Hx.getCountX();
int ny = grid.Hx.getCountY();
int nz = grid.Hx.getCountZ();

float delta_x = grid.delta_x;
float delta_y = grid.delta_y;
float delta_z = grid.delta_z;

for(int ix = 0; ix < nx-1; ix++)
for(int iy = 0; iy < ny-1; iy++)
for(int iz = 0; iz < nz-1; iz++) {
    float& curHx      = grid.Hx          .at(ix,   iy,   iz);
    float  curD_Hx    = grid.D_Hx        .at(ix,   iy,   iz);
    float  curEz1     = grid.Ez          .at(ix,   iy+1, iz);
    float  curEz0     = grid.Ez          .at(ix,   iy,   iz);
    float  curEy1     = grid.Ey          .at(ix,   iy,   iz+1);
    float  curEy0     = grid.Ey          .at(ix,   iy,   iz);

    curHx -= curD_Hx * ((curEz1 - curEz0) / delta_y -
                        (curEy1 - curEy0) / delta_z);
}
```

Код расчёта одной компоненты вектора H во всех точках поля в конкретный момент времени на GPU

```
__kernel void calcH(int nx, int ny, int nz, float delta_x, float delta_y, float delta_z,
    __global float *Ex,
    __global float *Ey,
    __global float *Ez,
    __global float *Hx,
    __global float *Hy,
    __global float *Hz,
    __global float *D_Hx,
    __global float *D_Hy,
    __global float *D_Hz) {

    int ix = get_global_id(0);
    int iy = get_global_id(1);
    int iz = get_global_id(2);

    int idx = ix * ny * nz + iy * nz + iz;

    int idx010 = ix * ny * nz + (iy + 1) * nz + iz;
    int idx001 = ix * ny * nz + iy * nz + (iz + 1);
    int idx100 = (ix + 1) * ny * nz + iy * nz + iz;

    Hx[idx] -= D_Hx[idx] * ((Ez[idx010] - Ez[idx]) / delta_y -
                            (Ey[idx001] - Ey[idx]) / delta_z);

    Hy[idx] -= D_Hy[idx] * ((Ex[idx001] - Ex[idx]) / delta_z -
                            (Ez[idx100] - Ez[idx]) / delta_x);

    Hz[idx] -= D_Hz[idx] * ((Ey[idx100] - Ey[idx]) / delta_x -
                            (Ex[idx010] - Ex[idx]) / delta_y);
}
```