Computational Continuum Physics
Problem set 1

Finite Difference Methods

Chalmers University of Technology

Viktor Lilja and Nico Guth

May 17, 2023

Problem 1: Explicit FDM with 5-point stencil for the heat equation

See solution in appendix

Problem 2: FDTD modification without numerical dispersion along grid axis.

See solution in appendix

Problem 3: 1D FDTD method for temporal evolution of an electromagnetic pulse

The dynamics of electromagnetic fields, which vary only over the x-direction, are described by the Maxwell equations

$$\frac{\partial E_y}{\partial t} = -\frac{\partial B_z}{\partial x} \tag{1}$$

$$\frac{\partial B_z}{\partial t} = -\frac{\partial E_y}{\partial x}, \tag{2}$$

if units are assumed where the speed of light is $c = 1$. The FDTD method simulates the time evolution of the EM-field using finite differences with staggered grids. The B-field is shifted by half a step in both time and space relative E-field using the definitions

$$E_m^n = E_y \left(x = x_0 + mh, t = t_0 + n\tau\right)$$
$$B_m^n = B_z \left(x = x_0 + \left(m + \frac{1}{2}\right) h, t = t_0 + \left(n + \frac{1}{2}\right) \tau\right)$$

where h and τ are the grid spacings in space and time respectively. Using central differences, eq. (1) and eq. (2) are then approximated by

$$\frac{E_m^{n+1} - E_m^n}{\tau} = -\frac{B_m^n - B_{m-1}^n}{h}$$
$$\frac{B_m^{n+1} - B_m^n}{\tau} = -\frac{E_{m+1}^{n+1} - E_m^{n+1}}{h},$$

where it is implied that first the simulation step is calculated for the E-field using

$$E_m^{n+1} = E_m^n - \frac{\tau}{h}\left(B_m^n - B_{m-1}^n\right) \tag{3}$$

and then the B-field is calculated through

$$B_m^{n+1} = B_m^n - \frac{\tau}{h}\left(E_{m+1}^{n+1} - E_m^{n+1}\right). \tag{4}$$

An analysis similar to that in Problem 2 shows that the method is stable if $\sigma = \tau/h \leq 1$, and dispersion is eliminated if $\sigma = \tau/h = 1$.

Simulated using eq. (3) and eq. (4) is an electromagnetic pulse in a 1D space $x \in [-1, 1]$ in the timespan $t \in [0, 2.5]$. We used $M = 500$ space steps and $\sigma = \tau/h = 1.0$, implying $N = 625$ time steps. The simulation needs an initial condition for both E_y and B_z. Chosen is a rightward traveling sinusoidal wave pulse of two periods

$$E_y(x, t = 0) = \begin{cases} \sin(20\pi x) & \text{for } |x| < 0.1 \\ 0 & \text{for } |x| \geq 0.1 \end{cases}.$$

For a rightward traveling wave, the B field has to be shifted in space by $c\tau/2$ since there is a half timestep shift between E and B

$$B_z(x, t = \tau/2) = \begin{cases} \sin(20\pi x + \tau/2) & \text{for } |x + \tau/2| < 0.1 \\ 0 & \text{for } |x + \tau/2| \geq 0.1 \end{cases}.$$

Since our discretization has an E-node at both ends of the simulated domain, we need to specify boundary conditions for the E-field at

$$E_y(x = -1, t = n\tau) = E_0^n \qquad E_y(x = 1, t = n\tau) = E_{M-1}^n \tag{5}$$

where M is the number of grid points for the E-field and the indices start at 0. Demonstrated here are three different boundary conditions for this grid setup.

The first case, for which two time steps are shown in fig. 1, is periodic boundary conditions. The simulation cell is assumed to be repeated infinitely many times in both x-directions, and therefore the imposed the boundary conditions are

$$E_0^{n+1} = E_{M-1}^{n+1} = E_0^n - \frac{\tau}{h}\left(B_0^n - B_{M-2}^n\right). \tag{6}$$

The simulation shows that the wave, which is travelling in the positive x-direction with the speed of light $c = 1$, is at the same x coordinate at times $t = 0.5\,\mathrm{a\,u}$ and $t = 2.5\,\mathrm{a\,u}$. Note that the shape of the wave packet stays constant over time. If we had not used $\sigma = 1$, due to numerical dispersion, high frequency components of the wave would “bleed out” on the left side of the wave.

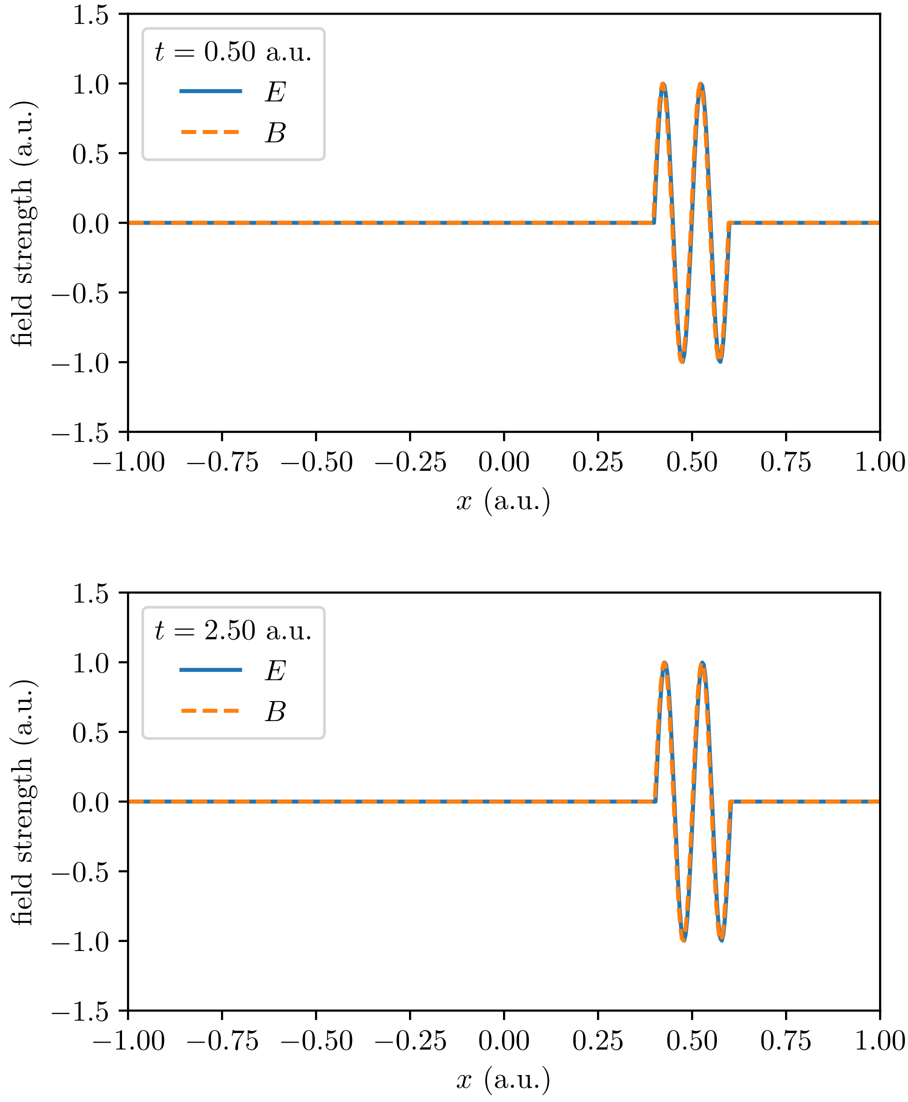


Figure 1: Simulated EM wave packet using periodic boundary conditions. The packet travels to the right and loops around once.

The second case, for which two time steps are shown in fig. 2, is reflective boundary conditions. These boundary conditions model a perfect conductor at the left and right boundary, which reflects the wave without any loss. In the figure, we see that at $t = 2.50$, the pulse has been reflected once and is travelling to the left.

In an ideal electrical conductor, any E-field induced by an external field will be cancelled immediately. Therefore, the E-field must vanish, and the boundary conditions translate to

$$E_0^n = 0 \qquad\qquad E_{M-1}^n = 0\,. \tag{7}$$

To model ideal *magnetic* conductors, we could use a finite difference approximation to set the space derivative of E to zero on the boundaries.

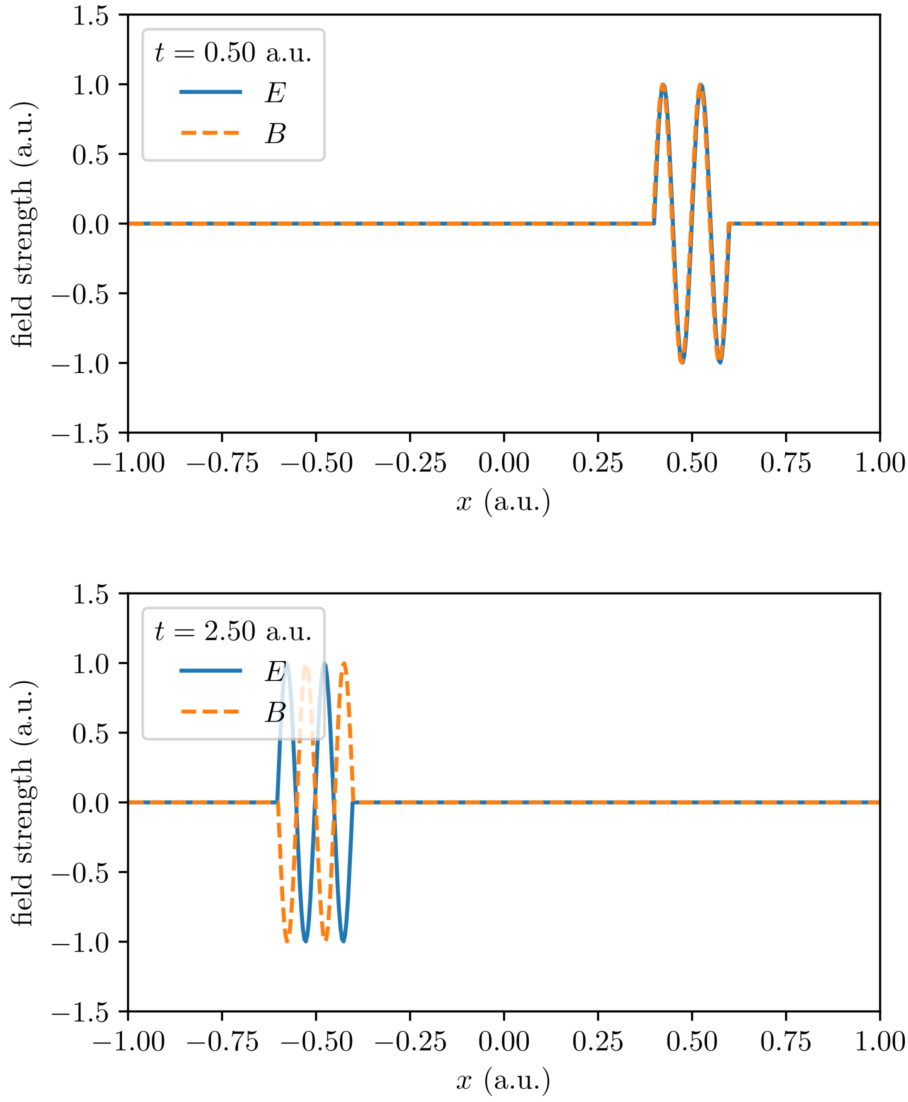


Figure 2: Simulated EM wave packet using reflective boundary conditions. The packet travels to the right, is reflected, and then travels to the left.

The third case, for which two time steps are shown in fig. 3, is an antenna at $x = 0$ emitting a continuous signal

$$E(x = 0, t) = \sin(20t) \tag{8}$$

in empty space. Empty space is modelled by boundaries left and right which are perfect absorbers, so that there is no reflection back into the simulation cell. Perfect absorption is implemented using Mur's boundary conditions

$$E_0^n = E_1^{n-1} + \frac{\tau - h}{\tau + h}\left(E_1^n - E_0^n\right) \tag{9}$$

$$E_{M-1}^n = E_{M-2}^{n-1} + \frac{\tau - h}{\tau + h}\left(E_{M-2}^n - E_{M-1}^n\right), \tag{10}$$

derived in [1]. In short, these can be obtained from finite difference approximations of the advection equations

$$\frac{\partial E_y}{\partial x} - \frac{\partial E_y}{\partial t} = 0$$
$$\frac{\partial E_y}{\partial x} + \frac{\partial E_y}{\partial t} = 0,$$

which only allow for solutions where waves travel outwards on the left and right boundaries respectively. Another way to model absorbing boundaries, which we did not implement, is to generalize the finite difference method to lossy media. One could then add strongly dampening regions to the left and right of the simulation region to attenuate the waves.

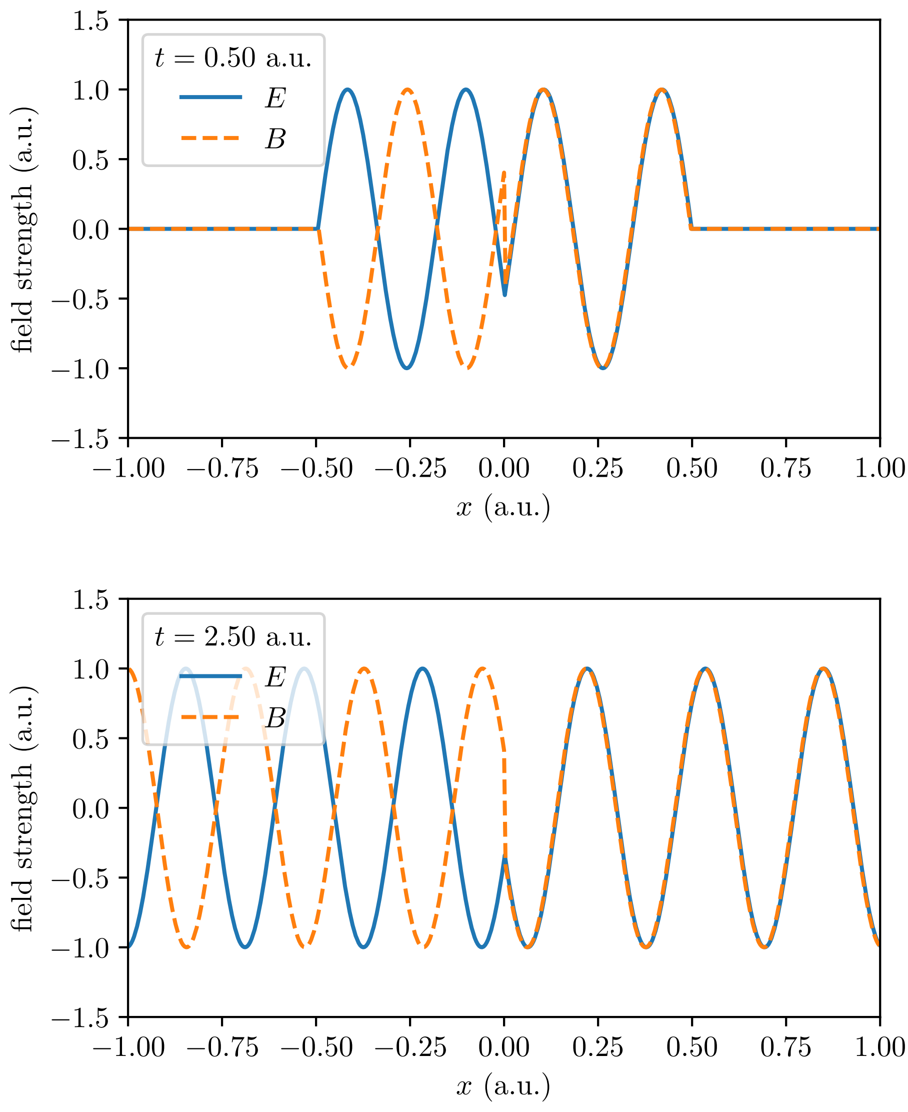


Figure 3: Simulated emission of an antenna in empty space. The antenna continuously generates waves at $x = 0$, travelling outwards. When the wavefronts reach the boundaries, they are absorbed.

References

[1] Department of Electrical and University of Utah Computer Engineering. *The Finite-Difference TimeDomain Method (FDTD)*. Online lecture notes. 2012. URL: `https://my.ece.utah.edu/~ece6340/LECTURES/lecture%2014/FDTD.pdf`.

Appendix: C++ code for problem 3

```cpp
#include <iostream>
#include <math.h>
#include <fstream>

using namespace std;

int main()
{

    double sigma = 1.0;  // Courant number

    // Space discretization
    double x_min = -1.0;
    double x_max = 1.0;
    const int M = 500; // Number of grid points for E
    double h = (x_max - x_min) / M;

    // Time discretization
    double t_min = 0.0;
    double t_max = 2.5;
    double tau   = sigma * h; // Magic time step
    const int N  = ceil((t_max - t_min) / tau); // number of timesteps


    // For storing snapshots of fields
    int frame_count  = 25;
    int frame_stride = N / frame_count;

    // check if stability condition (sigma = tau/h < 1) is true
    if (tau/h > 1) {
        cerr << "ERROR! Method is instable: tau = " << tau << " h = " << h << "\n";
        exit(1);
    }

    // For storing fields
    double E[M], B[M-1], E_old[M], B_old[M-1];

    // linspaces for the x values
    double x_E[M], x_B[M-1];
    for (int m=0; m<M; m++) {
        x_E[m] = x_min + m*h;
        x_B[m] = x_min + (m+0.5)*h;
    }

```

```cpp
    // initial conditions:
    for (int m=0; m<M; m++) {

        // Calculate the shift in the initial condition for the B field for the first halfstep.
        // We are simulating the wave equation with an initial state traveling to the right
        // (E, B > 0 --> k = E x B > 0), so the shift is given by:
        double c = 1.0;
        double x_shift = c * tau / 2;

        // Gaussian wave packet:
        //E[m] = abs(x_E[m])<0.3 ? exp(-x_E[m]*x_E[m]/(2*0.1*0.1))*sin(20 * M_PI * x_E[m]) : 0;
        //B[m] = abs(x_B[m]+x_shift)<0.3 ? exp(-x_B[m]*(x_B[m]+x_shift)/(2*0.1*0.1))*sin(20 * M_PI * (x_B[m]+x_shift)) : 0;

        // Sine wave packet
        //E[m] = abs(x_E[m])<0.1 ? sin(20 * M_PI * x_E[m]) : 0;
        //B[m] = abs(x_B[m]+x_shift)<0.1 ? sin(20 * M_PI * (x_B[m]+x_shift)) : 0;

        // Nothing
        E[m] = 0;
        B[m] = 0;
    }

    // write header and initial conditions to the output file
    ofstream file = ofstream("data/output.csv", ios::trunc);
    file << "# n, t, E[0:(M-1)], B[0:(M-1)]\n";
    file << 0 << ", " << t_min;
    for (int m=0; m<M; m++) {
        file << ", " << E[m];
    }
    for (int m=0; m<M-1; m++) {
        file << ", " << B[m];
    }
    file << "\n";
    file.close();
    std::cout << "\rSimulating... " << 1 << "/" << N;

    for (int n=1; n<N; n++) {

        // copy arrays
        for (int m=0; m<M; m++)   E_old[m] = E[m];
        for (int m=0; m<M-1; m++) B_old[m] = B[m];

        // FD step B
        for (int m=0; m<M-1; m++)
            B[m] = B_old[m] - (tau/h) * (E_old[m+1] - E_old[m]);

        // FD step E
        for (int m=1; m<M-1; m++)
            E[m] = E_old[m] - (tau/h) * (B[m] - B[m-1]);

        // Boundary conditions on the left
```

```
        //E[0] = E_old[1] + (tau-h)/(tau + h) * (E[1]-E[0]);    // absorbing
        E[0] = 0;                                              // reflective
        //E[0] = E_old[0] - (tau/h) * (B[0] - B[M-2]);          // periodic
        //E[0] = 0.5 * sin(10 * n * tau);                       // antenna

        // Boundary conditions on the right
        //E[M-1] = E_old[M-2] + (tau-h)/(tau+h) * (E[M-2]-E[M-1]);  //
    absorbing
        E[M-1] = 0;                                                 // reflective
        //E[M-1] = E[0];                                              // periodic
        //E[M-1] = 0.5 * sin(10 * n * tau);                           // antenna

        // antenna in the middle
        //E[M/2] = sin(20 * n * tau);

        // save the current timestep in regular intervals
        if (((n+1) % frame_stride == 0) || (n==(N-1))) {
            file = ofstream("data/output.csv", ios::app);
            file << n << ", " << t_min+n*tau;
            for (int m=0; m<M; m++) {
                file << ", " << E[m];
            }
            for (int m=0; m<M-1; m++) {
                file << ", " << B[m];
            }
            file << "\n";
            file.close();
            std::cout << "\rSimulating... " << n+1 << "/" << N;
        }
    }
    std::cout << "\n";

    // write metadata to a file for plotting
    file = ofstream("data/output_constants.json", ios::trunc);
    if (file.is_open())
    {
        file << "{\n";
        file << "\"M\": " << M << ",\n";
        file << "\"x_min\": " << x_min << ",\n";
        file << "\"x_max\": " << x_max << ",\n";
        file << "\"dx\": " << h << ",\n";
        file << "\"N\": " << N << ",\n";
        file << "\"t_min\": " << t_min << ",\n";
        file << "\"t_max\": " << t_max << ",\n";
        file << "\"dt\": " << tau << "\n";
        file << "}\n";
    }
    file.close();

    std::cout << "Done.\n";
}
```

Appendix: Python code for problem 3

```
# %%
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import json

# For latex interpretation of the figures
plt.rcParams.update({
    "text.usetex": True,
    "font.family": "Computer Modern",
    "font.size": 11.0, # 11pt is fontsize of captions in the report
})

# %%
print("Reading in data...")

# read in the constants
with open("data/output_constants.json", "r") as f:
    consts = json.load(f)
M = consts["M"]
x_min = consts["x_min"]
x_max = consts["x_max"]
h = consts["dx"]

# read in the simulation data
data = np.genfromtxt("data/output.csv", delimiter=",")
n = data[:,0]
t = data[:,1]
E = data[:,2:2+M]
B = data[:,2+M:2+2*M]

x_E = np.linspace(x_min,       x_min + (M-1.0)*h, M)
x_B = np.linspace(x_min+0.5*h, x_min + (M-0.5)*h, M)

frame_count = E.shape[0]

# %%
print("Plot individual frames...")
for n_i in [0, frame_count//5, frame_count//5*2, frame_count//2, frame_count
    -1]:
    plt.figure(figsize=(5,3)) # textwidth in report is 6.6 inches
    plt.plot(x_E, E[n_i,:], linestyle="-", label=f"$E$")
    plt.plot(x_B, B[n_i,:], linestyle="--", label=f"$B$")
    plt.xlim(-1,1)
    plt.ylim(-1.5,1.5)
    plt.xlabel("$x$ (a.u.)")
    plt.ylabel("field strength (a.u.)")
    plt.legend(loc="upper left", title=f"$t = {t[n_i]:.2f}$ a.u.")
    plt.tight_layout()
    plt.savefig(f"figures/frame_{n_i:03d}_t_{t[n_i]:.2f}.pdf")


# %%
print("Plotting animation...")

# dont use latex
plt.rcParams.update({
```

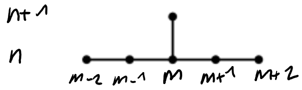
```python
    "text.usetex": False
})

fig = plt.figure()
ax = plt.axes(xlim=(x_min, x_max), ylim=(-1.5, 1.5))
ax.set_xlabel("x")
line_E, = ax.plot([], [], lw=2, label="E")
line_B, = ax.plot([], [], lw=2, label="B")
ax.legend(loc="upper right")


def init():
    line_E.set_data([], [])
    line_B.set_data([], [])
    return line_E, line_B

def animate(frame_idx):
    if (frame_idx % 10 == 0) or (frame_idx == frame_count-1):
        print(f"\rRendering... ({100*((frame_idx+1)/frame_count):.1f}%)", end="")
    ax.set_title(f"t = {t[frame_idx]:.2f}")
    line_E.set_data(x_E, E[frame_idx,:])
    line_B.set_data(x_E, B[frame_idx,:])
    return line_E, line_B

anim = animation.FuncAnimation(
  fig, animate, init_func=init,
  frames=frame_count, interval=20, blit=True)

anim.save('figures/movie.gif', writer='imagemagick')
print("\nRendering done!")
```

Problem 1

06 April 2023 10:35

Problem 1. Explicit FDM with 5-point stencil for the heat equation. (10 points)

For one-dimensional heat equation obtain an expression that describes the explicit FDM with the highest possible accuracy for the stencil shown below. Determine the approximation order and analyze the stability of the obtained FDM.



1D heat equation: $\frac{\partial u}{\partial t} - c^2 \frac{\partial^2 u}{\partial x^2} = f(x,t)$

general FDM described by this stencil:

$$(u_m^{n+1} - u_m^n)/\tau = c^2(\alpha_{+2} u^n_{m+2} + \alpha_{+1} u^n_{m+1} + \alpha_0 u^n_m + \alpha_{-1} u^n_{m-1} + \alpha_{-2} u^n_{m-2})/h^2$$

where α_i are constants independent of τ, h and c.

we assume symmetry: $\alpha_{-2} = \alpha_{+2} = \alpha_2$, $\alpha_{-1} = \alpha_{+1} = \alpha_1$

$$\Rightarrow (u_m^{n+1} - u_m^n)/\tau - c^2(\alpha_2 [u^n_{m+2} + u^n_{m-2}] + \alpha_1 [u^n_{m+1} + u^n_{m-1}] + \alpha_0 u^n_m)/h^2 = f^n_m \qquad (*)$$

<u>approximation order analysis</u>

evaluate residual $r_{\tau h} = f^n_m - (Lu)^n_m$ where $(Lu)^n_m$ is the left side of $(*)$

Taylor expansions in space: (with $u_m^{(i)} = \left(\frac{\partial^i u}{\partial x^i}\right)_m$ and $u^n_{(i)} = \left(\frac{\partial^i u}{\partial t^i}\right)^n$)

$$u_{m\pm 1} = u_m^{(0)} \pm h u_m^{(1)} + \frac{1}{2} h^2 u_m^{(2)} \pm \frac{1}{6} h^3 u_m^{(3)} + \frac{1}{4!} h^4 u_m^{(4)} \pm \frac{1}{5!} h^5 u_m^{(5)} + \frac{1}{6!} h^6 u_m^{(6)} + O(h^7)$$

$$u_{m\pm 2} = u_m^{(0)} \pm 2h u_m^{(1)} + 2h^2 u_m^{(2)} \pm h^3 u_m^{(3)} + \frac{2^4}{4!} h^4 u_m^{(4)} \pm \frac{2^5}{5!} h^5 u_m^{(5)} + \frac{2^6}{6!} h^6 u_m^{(6)} + O(h^7)$$

$$u^{n+1} = u^n_{(0)} + \tau u^n_{(1)} + \frac{1}{2}\tau^2 u^n_{(2)} + O(\tau^3)$$

$\Rightarrow$ if inserted into $(*)$, all odd terms in space cancel, all even terms combine.

$$\Rightarrow r_{\tau h} = f^n_m - (\cancel{u^n_m} + \tau u^n_{(1)m} + \frac{1}{2}\tau^2 u^n_{(2)m} + O(\tau^3) - \cancel{u^n_m})/\tau$$
$$+ c^2 (2\alpha_2 [u^n_m + 2h^2 u^{(2)n}_m + \frac{2^4}{4!} h^4 u^{(4)n}_m + \frac{2^6}{6!} h^6 u^{(6)n}_m + O(h^7)]$$
$$+ 2\alpha_1 [u^n_m + \frac{1}{2} h^2 u^{(2)n}_m + \frac{1}{4!} h^4 u^{(4)n}_m + \frac{1}{6!} h^6 u^{(6)n}_m + O(h^7)] + \alpha_0 u^n_m)/h^2$$

$\Rightarrow$ The u^n_m term must cancel $\Rightarrow$ $\boxed{2\alpha_2 + 2\alpha_1 + \alpha_0 \overset{!}{=} 0 \quad (\mathrm{I})}$

$\Rightarrow$ The f^n_m term must cancel by using the PDE $u^n_{(1)m} - c^2 u^{(2)n}_m = f^n_m$

$\Rightarrow$ $\boxed{4\alpha_2 + \alpha_1 \overset{!}{=} 1 \quad (\mathrm{II})}$

$\Rightarrow$ In order to specify $\alpha_0, \alpha_1, \alpha_2$ we need 3 equations but so far we only have (I) and (II)

$\Rightarrow$ we can get a higher approximation order by cancelling the $u^{(4)n}_m$ terms

$\Rightarrow$ $2\frac{2^4}{4!}\alpha_2 + 2\frac{1}{4!}\alpha_1 = 0 \iff \boxed{16\alpha_2 + \alpha_1 = 0 \quad (\mathrm{III})}$

This leads to:

$$r_{\tau h} = -\frac{1}{2}\tau u^n_{(2)m} + O(\tau^2) + c^2 (2\alpha_2 \frac{2^6}{6!} h^4 u^{(6)n}_m + 2\alpha_1 \frac{1}{6!} h^4 u^{(6)n}_m + O(h^5))$$

$\Rightarrow$ $\boxed{\text{approximation order } r_{\tau h} \sim O(\tau, h^4)}$

now specify $\alpha_0, \alpha_1, \alpha_2$:

$(II) \Leftrightarrow \alpha_1 = 1 - 4\alpha_2$ into $(III) \Rightarrow 16\alpha_2 + 1 - 4\alpha_2 = 0 \Leftrightarrow \boxed{\alpha_2 = -\frac{1}{12}}$

$\Rightarrow \alpha_1 = 1 - 4(-\frac{1}{12}) \Leftrightarrow \boxed{\alpha_1 = \frac{4}{3}}$ into $(I) \Rightarrow 2(-\frac{1}{12}) + 2(\frac{4}{3}) + \alpha_0 = 0 \Leftrightarrow \boxed{\alpha_0 = -\frac{5}{2}}$

$$\Rightarrow \text{FDM: } (u_m^{n+1} - u_m^n)/\tau - c^2\left(-\frac{1}{12}\left[u_{m+2}^n + u_{m-2}^n\right] + \frac{4}{3}\left[u_{m+1}^n + u_{m-1}^n\right] - \frac{5}{2}u_m^n\right)/h^2 = f_m^n$$

$$\Leftrightarrow \boxed{\frac{1}{\tau}(u_m^{n+1} - u_m^n) - \frac{c^2}{12h^2}\left(-u_{m+2}^n + 16u_{m+1}^n - 30u_m^n + 16u_{m-1}^n - u_{m-2}^n\right) = f_m^n \quad (**)}$$

Von Neumann stability analysis

To analyse the numerical stability, analyse the error of the numerical to the exact solution

$\varepsilon_m^n = u_m^n - y_m^n$ since the FDM equation is linear the equation is the same for ε_m^n instead of u_m^n.

Now express ε_m^n as the Fourier series $\varepsilon_m^n = \sum_k \lambda^n \exp(ikmh)$, the same argument holds for this sum,

so it is enough to insert a single Fourier mode into the homogeneous $(f=0)$ FDM and the FDM is stable if $|\lambda| \leq 1$ for all k

$\rightarrow$ insert $\lambda^n e_m$ into $(**)$ with the shorthand $e_m = \exp(ikmh)$

$$\Rightarrow \frac{1}{\tau}(\lambda^{n+1}e_m - \lambda^n e_m) - \frac{c^2}{12h^2}\left(-\lambda^n e_{m+2} + 16\lambda^n e_{m+1} - 30\lambda^n e_m + 16\lambda^n e_{m-1} - \lambda^n e_{m-2}\right) = 0$$

$$\Leftrightarrow \lambda - 1 = \frac{c^2\tau}{12h^2}\left(-e_{+2} + 16e_{+1} - 30 + 16e_{-1} - e_{-2}\right) \quad \text{use} \quad e_m + e_{-m} = 2\cos(kmh)$$

$$\Leftrightarrow \lambda = 1 + \frac{c^2\tau}{12h^2}\left(-2\cos(2kh) + 32\cos(kh) - 30\right) \quad \text{use} \quad 1 - \cos(x) = 2\sin^2(x/2)$$

$$\Leftrightarrow \lambda = 1 + \frac{c^2\tau}{12h^2}\left(4\sin^2(kh) - 64\sin^2(kh/2)\right)$$

$$|\lambda| \leq 1 \Leftrightarrow -2 \leq \frac{c^2\tau}{12h^2}\left(4\sin^2(kh) - 64\sin^2(kh/2)\right) \leq 0$$

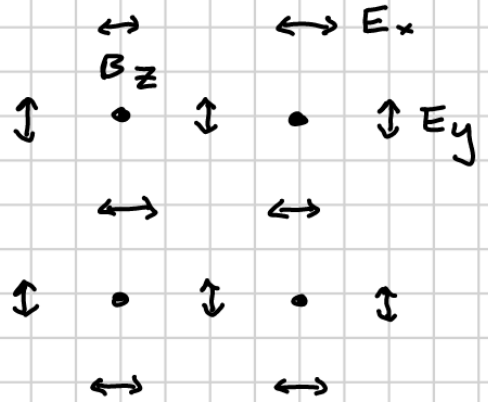
$$\Leftrightarrow -6\frac{h^2}{c^2\tau} \leq \sin^2(kh) - 16\sin^2(kh/2) \leq 0 \quad \text{must hold for all } k$$

but we also know $-16 \leq \sin^2(kh) - 16\sin^2(kh/2) \leq 0$

$$\Rightarrow -6\frac{h^2}{c^2\tau} \leq -16 \Leftrightarrow \boxed{\frac{c^2\tau}{h^2} \leq \frac{3}{8} \quad \text{stability condition for } (**)}$$

Problem 2

2D FDFT for EM-field (TE):



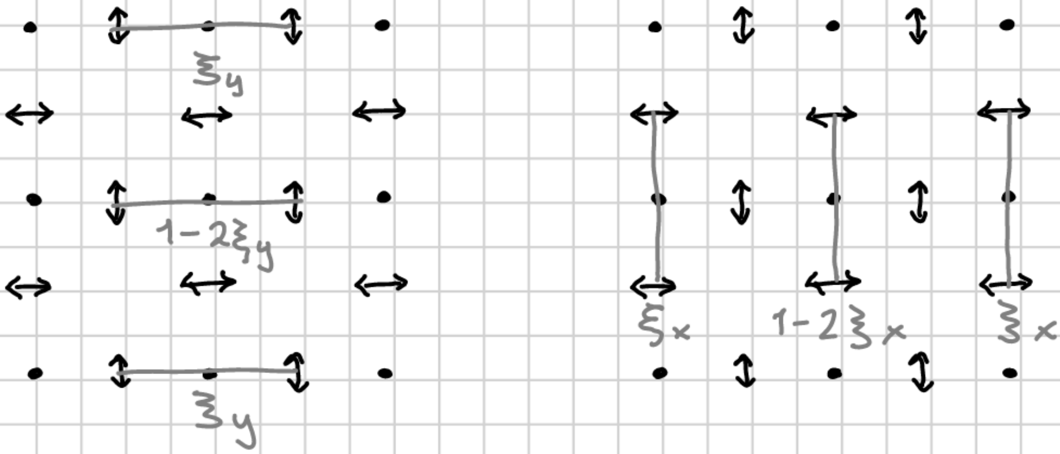
$$\begin{cases} \dfrac{\partial B_z}{\partial t} = -c\left(\dfrac{\partial E_y}{\partial x} - \dfrac{\partial E_x}{\partial y}\right) \\ \dfrac{\partial E_x}{\partial t} = c\,\dfrac{\partial B_z}{\partial y} \\ \dfrac{\partial E_y}{\partial t} = -c\,\dfrac{\partial B_z}{\partial x} \end{cases}$$

We shift the E-grid in time and space, so the FDM is given by

$$\begin{cases} \left.\dfrac{\partial B_z}{\partial t}\right|^{n+\frac{1}{2}}_{m_x m_y} = -c\left(\left.\dfrac{\partial E_x}{\partial y}\right|^{n+\frac{1}{2}}_{m_x m_y} + \left.\dfrac{\partial E_y}{\partial x}\right|^{n+\frac{1}{2}}_{m_x m_y}\right) \\ \left.\dfrac{\partial E_x}{\partial t}\right|^{n}_{m_x(m_y+\frac{1}{2})} = c\left.\dfrac{\partial B_z}{\partial z}\right|^{n}_{m_x(m_y+\frac{1}{2})} \\ \left.\dfrac{\partial E_y}{\partial t}\right|^{n}_{(m_x+\frac{1}{2})m_y} = -c\left.\dfrac{\partial B_z}{\partial x}\right|^{n}_{(m_x+\frac{1}{2})m_y} \end{cases}$$

Where we use standard stencils for all derivatives except $\partial E_x/\partial y$ and $\partial E_y/\partial x$.

Proposed stencil for computing spatial derivatives of E:



$$
\begin{aligned}
\left.\frac{\partial E_y}{\partial x}\right|^{n+1/2}_{m_x y_x} \sim\ & \xi_y \frac{E_y|^{n+1/2}_{(m_x+\frac{1}{2})(m_y+1)} - E_y|^{n+1/2}_{(m_x-\frac{1}{2})(m_y+1)}}{h_x} + \\
& + (1-2\xi_y)\frac{E_y|^{n+1/2}_{(m_x+\frac{1}{2})m_y} - E_y|^{n+1/2}_{(m_x-\frac{1}{2})m_y}}{h_x} + \\
& + \xi_y \frac{E_y|^{n+1/2}_{(m_x+\frac{1}{2})(m_y-1)} - E_y|^{n+1/2}_{(m_x-\frac{1}{2})(m_y-1)}}{h_x}
\end{aligned}
$$

(and similarily for $\partial E_x/\partial y$)

For dispersion and stability analysis we examine plane wave modes:

$$B_z = B_{z0}(t)e^{i(-k_x x - k_y y)}$$
$$E_y = E_{y0}(t)e^{i(-k_x x - k_y y)}$$
$$E_x = E_{x0}(t)e^{i(-k_x x - k_y y)}$$

For these, the improved stencil will give us derivative approximation:

$$\left.\frac{\partial E_y}{\partial x}\right|_{m_x m_y}^{n+1/2} \approx \frac{e^{-ik_x h_x/2} - e^{ik_x h_x/2}}{h_x} \times$$
$$\times \left(\xi_y e^{-ik_y h_y} + (1-2\xi_y) + \xi_y e^{ik_y h_y} \right) \times$$
$$\times E_{y0}((n+1/2)\tau)e^{i(-k_x m_x h_x - k_y m_y h_y)} =$$
$$= -\frac{2i}{h_x}\sin\left(\frac{k_x h_x}{2}\right)\left[1 - 2\xi_y(1-\cos k_y h_y)\right] E_y\Big|_{m_x m_y}^{n+1/2}$$
$$= -\frac{2i}{h_x}\sin\left(\frac{k_x h_x}{2}\right)\underbrace{\left[1 - 4\xi_y \sin^2\left(\frac{k_y h_y}{2}\right)\right]}_{\alpha_y} E_y\Big|_{m_x m_y}^{n+1/2} =$$
$$= -\frac{2i}{h_x}\sin\left(\frac{k_x h_x}{2}\right)\alpha_y E_y\Big|_{m_x m_y}^{n+1/2}$$

And similarily for $\partial E_x/\partial y$.

Dispersion analysis

Analyze monochromatic plane wave:

$$y\Big|_{m_x m_y}^{n} = y_0 e^{i(\omega n\tau - k_x m_x h_x - k_y m_y h_y)}$$

for $y = B_z, E_x, E_y$. The time derivative stencil for B_z becomes

$$\frac{\partial B_z}{\partial t}\Big|_{m_x m_y}^{n+\frac{1}{2}} \,\hat{=}\, \frac{B_z\big|_{m_x m_y}^{n+1} - B_z\big|_{m_x m_y}^{n}}{\tau} =$$

$$\frac{e^{i\omega\tau/2} - e^{-i\omega\tau/2}}{\tau} B_z\Big|_{m_x m_y}^{n+1/2} = \frac{2i}{\tau}\sin\left(\frac{\omega\tau}{2}\right) B_z\Big|_{m_x m_y}^{n+1/2}$$

Inserting the stencil approximations in the FDM, we get

$$\begin{cases} \sin\left(\frac{\omega\tau}{2}\right) B_z = -c\tau\left(-\frac{1}{h_x}\sin\left(\frac{k_x h_x}{2}\right)\alpha_y E_y + \frac{1}{h_y}\sin\left(\frac{k_y h_y}{2}\right)\alpha_x E_x\right) \\ \sin\left(\frac{\omega\tau}{2}\right) E_x = c\tau\left(-\frac{1}{h_y}\sin\left(\frac{k_y h_y}{2}\right) B_z\right) \\ \sin\left(\frac{\omega\tau}{2}\right) E_y = -c\tau\left(-\frac{1}{h_x}\sin\left(\frac{k_x h_x}{2}\right) B_z\right) \end{cases}$$

Where all fields are implicitly evaluated at $\Big|_{m_x m_y}^{n+1/2}$ (we can multiply by a phase on both sides to shift in x and y).

Substitute the 2:nd and 3:rd equation into the 1:st to get

$$\sin^2\left(\frac{\omega\tau}{2}\right) = c^2\tau^2\left(\frac{1}{h_x^2}\sin^2\left(\frac{k_x h_x}{2}\right)\alpha_y + \frac{1}{h_y^2}\sin^2\left(\frac{k_y h_y}{2}\right)\alpha_x\right)$$

For a mode propagating in the x-direction, $k_y = 0 \Rightarrow \alpha_y = 1$. The dispersion relation is then

$$\sin^2\left(\frac{\omega\tau}{2}\right) = \frac{c^2\tau^2}{h_x^2}\sin^2\left(\frac{k_x h_x}{2}\right)$$

If we set $c\tau/h_x = 1$:

$$\frac{\omega\tau}{2} = \pm\frac{k_x h_x}{2} \iff \boxed{\omega = \pm c k_x}$$

this is perfect (no) dispersion along x! By symmetry, we can also set $h_y = h_x = c\tau$ to also get perfect dispersion along y.

It remains to find out how to select ξ_x and ξ_y in order to make the scheme stable for all k_x, k_y.

Stability analysis

To assess possibility of growing mode, set

$$y\big|^n_{m_x m_y} = \lambda^n e^{i(-k_x h_x m_x - k_y h_y m_y)}$$

where $y = B_z, E_x, E_y$. Time derivative stencil for e.g. B_z becomes

$$\frac{\partial B_z}{\partial t}\bigg|^{n+1/2}_{m_x m_y} = \frac{\lambda^{1/2} - \lambda^{-1/2}}{\tau} B_z\bigg|^{n+1/2}_{m_x m_y}$$

Insert into FDM to get

$$\begin{cases} \dfrac{\lambda^{1/2}-\lambda^{-1/2}}{\tau} B_z = -c\left(-\dfrac{2i}{h_x}\sin\left(\dfrac{k_x h_x}{2}\right)\alpha_y E_y + \dfrac{2i}{h_y}\sin\left(\dfrac{k_y h_y}{2}\right)\alpha_x E_x\right) \\ \dfrac{\lambda^{1/2}-\lambda^{-1/2}}{\tau} E_x = c\left(-\dfrac{2i}{h_y}\sin\left(\dfrac{k_y h_y}{2}\right) B_z\right) \\ \dfrac{\lambda^{1/2}-\lambda^{-1/2}}{\tau} E_y = -c\left(-\dfrac{2i}{h_x}\sin\left(\dfrac{k_x h_x}{2}\right) B_z\right) \end{cases}$$

where again i shifted eq. 2 and 3. Substitute second and third eq. into first to get

$$\frac{(\lambda-1)^2}{\tau^2} = -c^2\left(\frac{4}{h_x^2}\sin^2\left(\frac{k_x h_x}{2}\right)\alpha_y + \frac{4}{h_y^2}\sin^2\left(\frac{k_y h_y}{2}\right)\alpha_x\right)\lambda$$

This can be written on the form

$$\lambda^2 - (2-\beta)\lambda + 1 = 0 \qquad (*)$$

where

$$\beta = 4c^2\tau^2\left[\frac{\alpha_y}{h_x^2}\sin^2\left(\frac{k_x h_x}{2}\right) + \frac{\alpha_x}{h_y^2}\sin^2\left(\frac{k_y h_y}{2}\right)\right] =$$

$$= 4c^2\tau^2\left[\frac{1}{h_x^2}\sin^2\left(\frac{k_x h_x}{2}\right) + \frac{1}{h_y^2}\sin\left(\frac{k_y h_y}{2}\right) - \right.$$

$$\left. -4\left(\frac{\xi_y}{h_x^2} + \frac{\xi_x}{h_y^2}\right)\sin^2\left(\frac{h_x k_x}{2}\right)\sin^2\left(\frac{h_y k_y}{2}\right)\right].$$

When the solutions λ_1 λ_2 to $(*)$ are smaller than one in magnitude, the scheme is stable.

$$\lambda^2 - (2-\beta)\lambda + 1 = (\lambda - \lambda_1)(\lambda - \lambda_2) = 0$$

$$\lambda_{1,2} = \frac{2-\beta}{2} \pm i\sqrt{1^2 - \left(\frac{2-\beta}{2}\right)^2}$$

$\lambda_1\lambda_2 = 1 \Rightarrow$ λ_1 and λ_2 must be equal or imaginary for $|\lambda_1|, |\lambda_2| \leq 1$.

In that case

$$|\lambda_{1,2}|^2 = \left(\frac{2-\beta}{2}\right)^2 + \left(-\left(\frac{2-\beta}{2}\right)^2 - 1\right) = 1 \leq 1$$

so $\lambda_{1,2}$ imaginary $\Leftrightarrow$ scheme stable.

We have imaginary solutions when

$$1^2 - \left(\frac{2-\beta}{2}\right)^2 \geq 0 \quad \Leftrightarrow \quad (2-\beta)^2 \leq 2^2$$

$$\Leftrightarrow \quad |2-\beta| \leq 2 \quad \Leftrightarrow \quad 0 \leq \beta \leq 4$$

We are interested in the dispersion free case $c\tau/h_x = 1$.

If $h_x = h_y = c\tau$ and $2\xi = (\xi_x + \xi_y)$, β reduces to

$$\beta = 4\left(s_x + s_y - 8\xi s_x s_y\right)$$

where $s_i = \sin^2\left(\frac{k_i h_i}{2}\right) \in [0,1]$, $i = x, y$

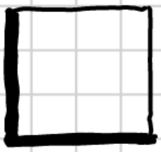
The scheme is then stable iff

$$0 \leq f(s_x, s_y) \leq 1, \quad f = s_x + s_y - 8\xi s_x s_y$$

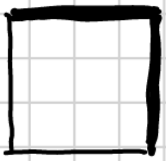
for all $s_x, s_y \in [0,1]^2$.

Finding bounds for $f(s_x, s_y)$:

Along edges:

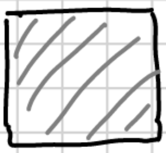


$$f(0, s_y) = s_y \in [0, 1]$$



$$f(1, s_y) = 1 + s_y - 8\xi s_y = $$
$$= 1 + (1 - 8\xi) s_y \in [2 - 8\xi, 1]$$

Inside domain:



$$\nabla f = \begin{pmatrix} 1 - 8\xi s_y \\ 1 - 8\xi s_x \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$\Rightarrow$ extemum at $\begin{pmatrix} s_x \\ s_y \end{pmatrix} = \begin{pmatrix} 1/8\xi \\ 1/8\xi \end{pmatrix}$

$$f\left(\frac{1}{8\xi}, \frac{1}{8\xi}\right) = \frac{1}{8\xi} + \frac{1}{8\xi} - 8\xi \frac{1}{8\xi} \frac{1}{8\xi} = \frac{1}{8\xi}$$

For positive ξ:

$$\min(0, 2 - 8\xi) \leq f(s_x, s_y) \leq \max\left(1, \frac{1}{8\xi}\right)$$

Stability condition is satisfied if

$$\begin{cases} 2 - 8\xi \geq 0 \\ 1/8\xi \leq 1 \end{cases} \Leftrightarrow \boxed{\frac{1}{8} \leq \xi \leq \frac{1}{4}}$$

Summary

If $h_x = h_y = c\tau$, there is no dispersion of waves along the x and y axes.

Such a scheme is stable if

$$\frac{1}{4} \le \xi_1 + \xi_2 \le \frac{1}{2} .$$