Shraddha Raghuram
5211972
ShraddhaRaghuram@student.tudelft.nl

Assignment 4: Numerical Solution of Time-Dependent Problems

2022-10-12

1 Upper Bound on the Maximum Eigenvalue

While determining the stability condition for a time-integration method one is often interested in the “largest” eigenvalue of some matrix A. Sometimes this eigenvalue is given in a closed form. However, more often, for example, when the coefficient function is inhomogeneous, the largest eigenvalue cannot be obtained analytically. In such cases it is possible to estimate the location of the eigenvalues of A approximately using the following Lemma.
Lemma (Gerschgorin): Let$A \in \mathbb{C}^{n\times n}$ be an arbitrary square matrix with elements $a_{r,m}$. Then all eigenvalues of A are contained in the union of n discs $\mathbb{S}_r$, r = 1, . . . , n:

$$\mathbb{S}_r = \{z \in \mathbb{C}; |z - a_{r,r}| \leq \sum_{m\neq r}^{n} |a_{r,m}|\} \tag{1}$$

In other words, the Gerschgorin’s disc Sk represents a circle in the complex plane. The center of $\mathbb{S}_k$ is $a_{k,k}$, i.e., the complex number equal to the diagonal element of the k-th row of A. The radius of $\mathbb{S}_k$ is equal to the sum of the absolute values of all off-diagonal elements of the k-th row of A. There are as many Gerschgorin discs as there are rows in A and all eigenvalues of A are located somewhere inside the union of all these discs.

1. Let A be the matrix obtained by the Finite-Volume discretization of the two-dimensional Laplace operator -$\nabla \cdot (k\nabla ....)$ with an inhomogeneous coefficient function k(x, y) and Dirichlet boundary conditions. Use Assignment 3 to write down the diagonal and off-diagonal elements along some row of the matrix A.
The diagonal elements of the matrix A are:

$$\frac{k_{r-0.5h_x,m}}{h_x^2} + \frac{k_{r,m-0.5h_y}}{h_y^2} + \frac{k_{r+0.5h_x,m}}{h_x^2} + \frac{k_{r,m+0.5h_y}}{h_y^2}$$

The off diagonal elements are as follows, respectively for offsets equal to -1, 1, $N_x - 1$ and $-N_x + 1$:

$$-\frac{k_{r-0.5h_x,m}}{h_x^2}$$

$$-\frac{k_{r+0.5h_x,m}}{h_x^2}$$

$$-\frac{k_{r,m-0.5h_y}}{h_y^2}$$

$$-\frac{k_{r,m+0.5h_y}}{h_y^2}$$

2. Use the fact that A is symmetric and positive and Gerschgorin’s Lemma to determine an upper bound on the largest eigenvalue of A.
Using the Gerschgorin’s Lemma, we find that

$$|z - a_{r,r}| \leq \sum_{m\neq r}^{n} |a_{r,m}|$$

Since the matrix A is symmetric and positive, the eigenvalues are real and positive. The possible values of $\sum_{m\neq r}^{n} |a_{r,m}|$ are $\frac{k_{r+0.5h_x,m}}{h_x^2} + \frac{k_{r,m-0.5h_y}}{h_y^2}$,
$\frac{k_{r-0.5h_x,m}}{h_x^2} + \frac{k_{r+0.5h_x,m}}{h_x^2} + \frac{k_{r,m-0.5h_y}}{h_y^2}$,
$\frac{k_{r,m+0.5h_y}}{h_y^2} + \frac{k_{r-0.5h_x,m}}{h_x^2} + \frac{k_{r+0.5h_x,m}}{h_x^2} + \frac{k_{r,m-0.5h_y}}{h_y^2}$,
$\frac{k_{r,m+0.5h_y}}{h_y^2} + \frac{k_{r-0.5h_x,m}}{h_x^2} + \frac{k_{r+0.5h_x,m}}{h_x^2}$,

and $\frac{k_{r,m+0.5h_y}}{h_y^2} + \frac{k_{r-0.5h_x,m}}{h_x^2}$.
Removing the modulus symbol from the LHS of the inequality,

$$z - a_{r,r} \leq \sum_{m \neq r}^{n} |a_{r,m}|$$

$$\implies z \leq \sum_{m \neq r}^{n} |a_{r,m}| + a_{r,r}$$

and

$$a_{r,r} - z \leq \sum_{m \neq r}^{n} |a_{r,m}|$$

$$\implies z \geq a_{r,r} - \sum_{m \neq r}^{n} |a_{r,m}|$$

So, the largest value of the upper bound of z is the largest value of $\sum_{m \neq r}^{n} |a_{r,m}| + a_{r,r}$. This is $2a_{r,r}$. Hence, the upper bound of the eigenvalues is $2(\frac{k_{r-0.5h_x,m}}{h_x^2} + \frac{k_{r,m-0.5h_y}}{h_y^2} + \frac{k_{r+0.5h_x,m}}{h_x^2} + \frac{k_{r,m+0.5h_y}}{h_y^2})$.

2 Diffusion Equation

Consider the following problem:

$$\frac{\partial u}{\partial t} - \nabla \cdot (k \nabla u) = f, t \in (0, T]$$

$$u(x, y, t) = 0, (x, y) \in \partial\Omega$$

$$u(x, y, 0) = 0, (x, y) \in \Omega$$

$$f(x, y) = e^{-\alpha(x-x_1)^2 - \alpha(y-y_1)^2} + e^{-\alpha(x-x_2)^2 - \alpha(y-y_2)^2} + e^{-\alpha(x-x_3)^2 - \alpha(y-y_3)^2} + e^{-\alpha(x-x_4)^2 - \alpha(y-y_4)^2} \tag{2}$$

$$k(x, y) = \begin{cases} 0.1 \text{ if } x < L_x/2, y < L_y/2; \\ 0.4 \text{ if } x < L_x/2, y \geq L_y/2; \\ 0.7 \text{ if } x \geq L_x/2, y \geq L_y/2; \\ 1.0 \text{ if } x \geq L_x/2, y < L_y/2; \end{cases}$$

where

- $\overline{\Omega} = [0, L_x] \times [0, L_y]$ is a rectangle with $L_x = 10, L_y = 5$
- $\alpha = 40$
- $(x_1, y_1) = (0.25L_x, 0.25L_y)$
- $(x_2, y_2) = (0.25L_x, 0.75L_y)$
- $(x_3, y_3) = (0.75L_x, 0.75L_y)$
- $(x_4, y_4) = (0.75L_x, 0.25L_y)$

2.1 Spatial Discretization and Time Integration

After applying the Finite Volume Method to discretize the problem in space it reduces to the following system of ordinary differential equations:

$$\mathbf{u}' = -A\mathbf{u} + \mathbf{f}, \tag{3}$$

where A is the matrix containing the FV approximation of the negative Laplacian operator derived in Assignment 3.

1. Write down the recurrence formulas (iteration formulas) of the Forward-Euler and Trapezoidal Methods for your problem, denoting the solution at t_k as $\mathbf{u}^k$.
   Forward-Euler Method:
   Single point approximation of the integral at the starting point:
$$\int_{t_0}^{t_0+h} \mathbf{f}(t, \mathbf{u}(t))\, dt \approx h\mathbf{f}(t_0, \mathbf{u}(t_0))$$
   Explicit one-step prediction:
$$\mathbf{u}(t_0 + h) \approx \mathbf{u}(t_0) + h\mathbf{f}(t_0, \mathbf{u}(t_0))$$
   Hence the Forward-Euler iterations are as follows:
$$\mathbf{u}(t_k + h) \approx \mathbf{u}(t_k) + h\mathbf{f}(t_0, \mathbf{u}(t_0)), k = 0, 1, ...$$
$$\implies \mathbf{u}(t_k + h) \approx \mathbf{u}^k + h\mathbf{f}(t_k, \mathbf{u}^k)$$
   Trapezoidal method:
   Two-point approximation of the integral:
$$\int_{t_k}^{t_k+1} \mathbf{f}(t, \mathbf{u}(t))\, dt = \frac{1}{2}h[\mathbf{f}(t_k, \mathbf{u}(t_k)) + \mathbf{f}(t_{k+1}, \mathbf{u}(t_{k+1}))]$$
   Hence, the implicit trapezoidal iterations are as follows:
$$\mathbf{u}(t_{k+1}) \approx \mathbf{u}(t_k) + \frac{1}{2}h[\mathbf{f}(t_k, \mathbf{u}(t_k)) + \mathbf{f}(t_{k+1}, \mathbf{u}(t_{k+1}))], k = 0, 1, ...$$
$$\mathbf{u}^{k+1} \approx \mathbf{u}^k + \frac{1}{2}h[\mathbf{f}(t_k, \mathbf{u}^k) + \mathbf{f}(t_{k+1}, \mathbf{u}^{k+1})]$$

2. Derive the stability condition for the FE method with the FVM discretization and inhomogeneous k(x, y), using the bound on the largest eigenvalue of A obtained above.
   The stability condition of the Forward Euler Method can be derived as follows: After FVM discretization, the diffusion equation becomes
$$\mathbf{u}' = -A\mathbf{u}, \mathbf{u}(t_0) = \mathbf{u}_0$$
   The Forward-Euler Method with the time step h produces iterations k=0,1,....
$$\mathbf{u}_h(t_k) = \mathbf{u}_h(t_{k-1}) - hA\mathbf{u}_h(t_{k-1}) = [I - hA]\mathbf{u}_h(t_{k-1}) = [I - hA]^k\mathbf{u}_0$$
   Let A be diagonalizable: $A = V\Lambda V^{-1}$ with eigenvalues $\lambda_j, j = 1, ..., n$.
$$\mathbf{u}(t_k) = Ve^{\Lambda t_k}V^{-1}\mathbf{u}_0 = \sum_{j=1}^{n} b_j(0)e^{-\lambda_j t_k}\mathbf{v}_j \text{(method of lines)}$$
$$\mathbf{u}_h(t_k) = [I - hA]^k\mathbf{u}_0 = V[I - h\Lambda]^kV^{-1}\mathbf{u}_0 = \sum_{j=1}^{n} b_j(0)(1 - h\lambda_j)^k\mathbf{v}_j$$
   Hence, for $\lambda_j > 0, e^{-\lambda_j t_k}$ always decays with t_k, but $|1 - h\lambda_j|^k$ may be growing with k if h is too large. The Forward-Euler method is not always stable.
   Deriving the condition for the stability of the Forward-Euler method:
$$|1 - h\lambda_j| < 1$$
$$-1 < 1 - h\lambda_j < 1$$
$$-2 < -h\lambda_j < 0$$
$$0 < h\lambda_j < 2$$
$$0 < h < \frac{2}{\lambda_j}, j = 1, ..., n$$
   It is common to denote the time step as $h = \Delta t$:
$$0 < \Delta t < \frac{2}{max_j\lambda_j}$$

2.2 Implementation and Numerical Experiments

Save and submit the code for this diffusion problem in the file Assignment-4-diffusion.py. Re-use the Python functions from Assignment 3 as much as possible. Here and in what follows, use the doubly- uniform grid with $N_x = 200$ and $N_y = 100$.

1. Create the figures of the source and coefficient functions and insert them in your report.
   Done in Jupyter Notebook.

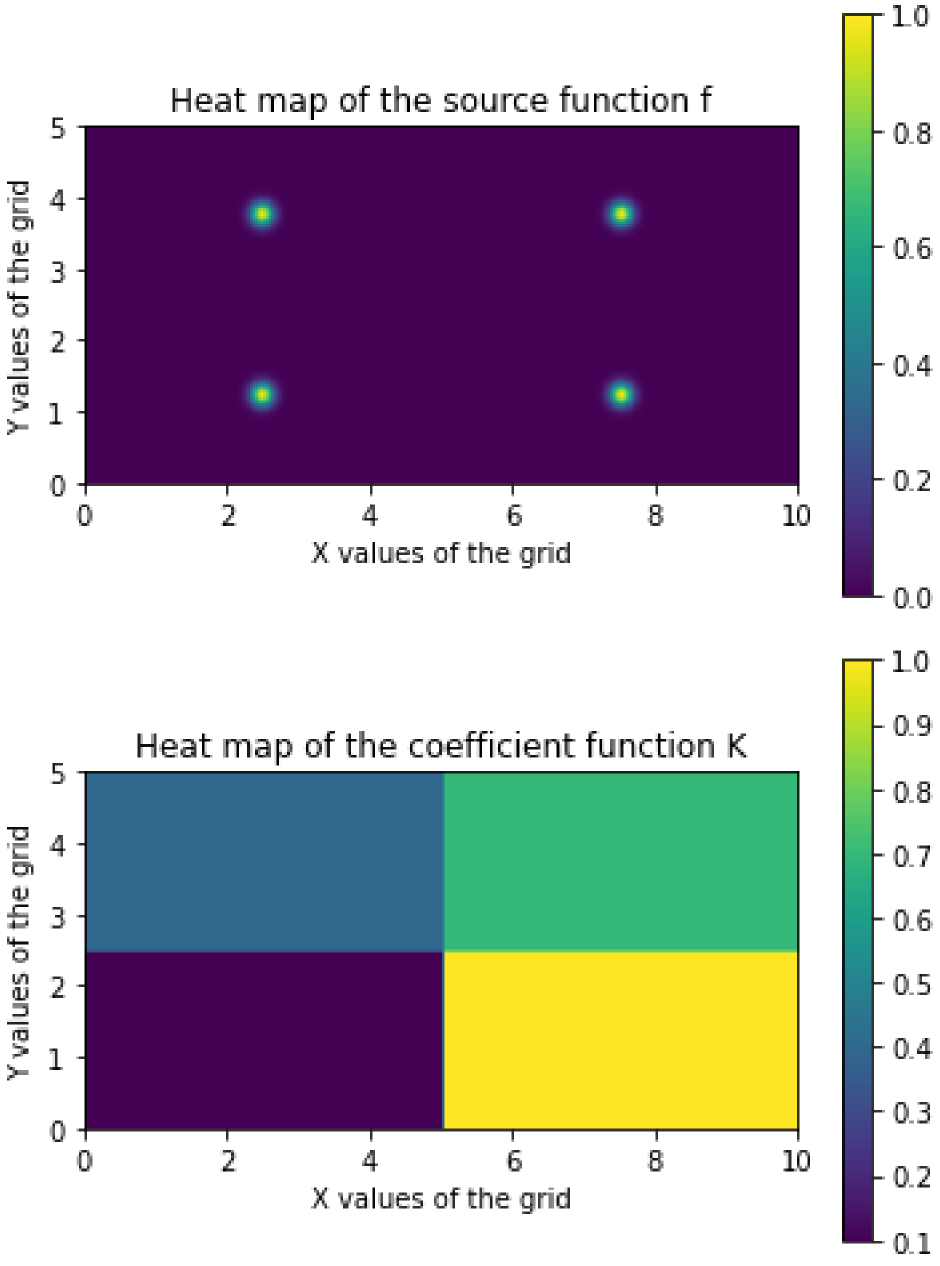

Figure 1: Plot from Matplotlib for 1.

2. Define two new Python functions that implement the Forward-Euler and the Trapezoidal Methods, respectively:
   Input:
   - uStart – lexicographic vector of the solution (initial condition) at t =tStart
   - tStart and tEnd – start and end times of the simulation
   - Nt – number of time intervals between tStart and tEnd

   Output:

- uEnd – lexicographic vector of the solution at t =tEnd computed with either the Forward-Euler or the Trapezoidal Method

You may either consider the matrix of the negative FD Laplacian to be globally defined as a sparse matrix A through A = create2DLFVM(...,coeffK), as well as the identity matrix I, defined through I = sp.eye(...), or pass these matrices as additional arguments to these functions.
Use the sparse linear solver from the scipy module wherever necessary.
Done in Jupyter Notebook.

3. Given T = 1, use the previously derived stability condition to compute the minimal number of time steps N_t that the Forward-Euler Method must take to compute the solution of the diffusion problem $\mathbf{u}_{FE}$ at t = T . The stable time step and the corresponding number of time steps N_t should be printed out by your code and presented in your report.
Done in Jupyter Notebook.
The stable time step is (0.0006253709302216115+0j).
The stable number of time steps is (1599.0509818638866+0j).

4. Compute the numerical approximation $u_1(x, y)$ of u(x, y, T) using the Forward-Euler method with the number of time steps between 0 and T = 1 set to Nt as determined by the stability bound in the previous step. Plot the result and insert it in your report.
Done in Jupyter Notebook.

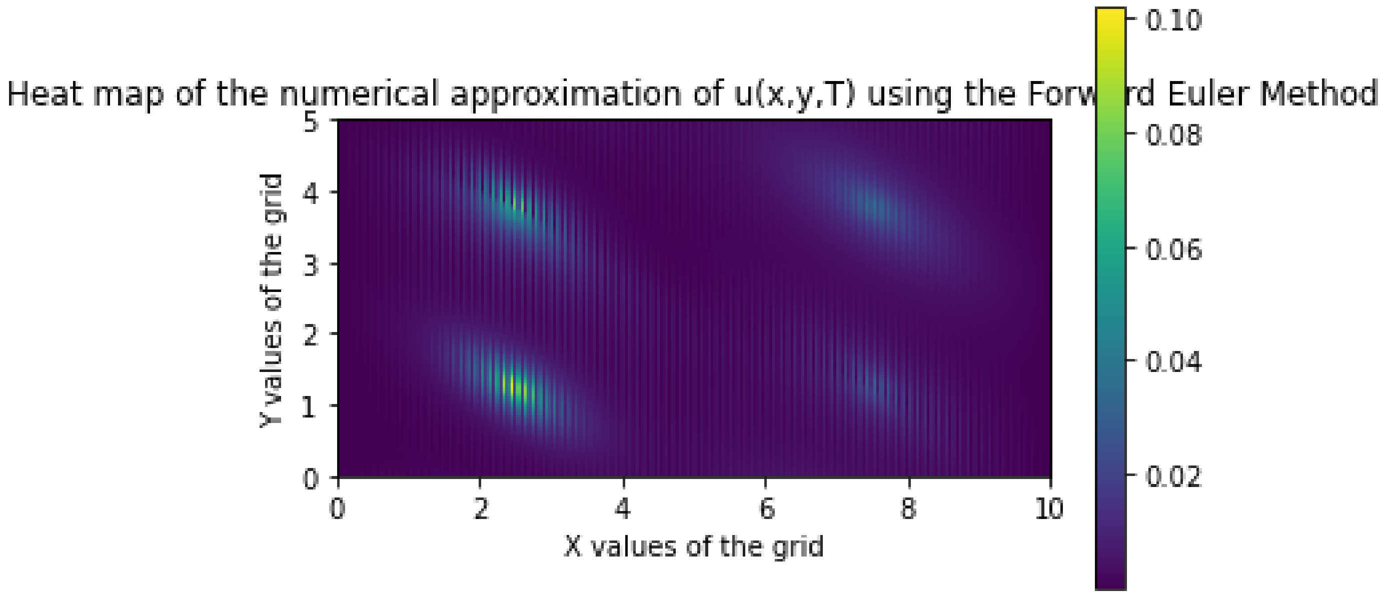

Figure 2: Plot from Matplotlib for 4.

5. Use the time module to measure the execution time of the FE Method in seconds. Print out this time and present it in your report.
Done in Jupyter Notebook.
The total amount of execution time of the FE method in seconds is 0.47228169441223145 s.

6. Compute the numerical approximation $u_2(x, y)$ of u(x, y, T) using the Forward-Euler method with the number of time steps between 0 and T = 1 set to $2N_t$, where N_t was determined by the stability bound.
Done in Jupyter Notebook.

7. Compute 8 numerical approximations $\tilde{u_i}(x, y)$ of u(x, y, T) using the Trapezoidal method with the number of time steps N_i between 0 and T = 1 set to 2^{i-1}, i = 1, 2, 3, 4, 5, 6, 7, 8. Plot the solution $\tilde{u_8}(x, y)$ and insert the figure in your report.
Done in Jupyter Notebook.

8. Compute and plot the difference $u_1(x, y) - \tilde{u_8}(x, y)$. Insert the figure in your report.
Done in Jupyter Notebook.

9. Compute the RMS differences: $RMS(\mathbf{u}_1, \tilde{\mathbf{u}}_i)$ and $RMS(\mathbf{u}_2, \tilde{\mathbf{u}}_i)$, i=1,....,8, where $\mathbf{u}_1, \mathbf{u}_2$ and $\tilde{\mathbf{u}}_i$ are the lexicographically ordered vectors of the numerical solutions $u_1(x,y), u_2(x,y)$ and $\tilde{u}(x,y)$, obtained previously. Consider performing these calculations in the same loop where you compute $\tilde{u}_i(x,y)$. Plot both RMS curves in the same graph using plt.semilogy() as functions of the number of steps Ni of the Trapezoidal method and insert it in your report.
   Done in Jupyter Notebook.

10. Which numerical solution is closer to the exact solution:
    (a) $\tilde{u_6}(x,y)$ or $\tilde{u_8}(x,y)$
    (b) $u_1(x,y)$ or $\tilde{u_8}(x,y)$
    (c) $u_1(x,y)$ or $\tilde{u_6}(x,y)$

    Provide argumentation in each case.

3 Wave Equation

Consider the following problem:

$$\frac{\partial u}{\partial t} - \nabla \cdot (k\nabla u) = f, (x,y) \in \Omega, t \in (0,T]$$

$$u(x,y,t) = 0, (x,y) \in \partial\Omega$$

$$u(x,y,0) = 0, (x,y) \in \Omega$$

$$f(x,y) = sin(\omega t) * (e^{-\alpha(x-x_1)^2-\alpha(y-y_1)^2} + e^{-\alpha(x-x_2)^2-\alpha(y-y_2)^2} + e^{-\alpha(x-x_3)^2-\alpha(y-y_3)^2} + e^{-\alpha(x-x_4)^2-\alpha(y-y_4)^2}) \tag{4}$$

$$k(x,y) = \begin{cases} 0.1 \text{ if } x < L_x/2, y < L_y/2; \\ 0.4 \text{ if } x < L_x/2, y \geq L_y/2; \\ 0.7 \text{ if } x \geq L_x/2, y \geq L_y/2; \\ 1.0 \text{ if } x \geq L_x/2, y < L_y/2; \end{cases}$$

where

- $\overline{\Omega} = [0, L_x] \times [0, L_y]$ is a rectangle with $L_x = 10, L_y = 5$
- $\alpha = 40$
- $(x_1, y_1) = (0.25L_x, 0.25L_y)$
- $(x_2, y_2) = (0.25L_x, 0.75L_y)$
- $(x_3, y_3) = (0.75L_x, 0.75L_y)$
- $(x_4, y_4) = (0.75L_x, 0.25L_y)$
- $\omega = 4\pi$

3.1 Spatial Discretization and Time Integration

After applying the Finite Volume Method to discretize the problem in space it reduces to the following system of ordinary differential equations:

$$\mathbf{u}'' = -A\mathbf{u} + \mathbf{f} \tag{5}$$

where A is the matrix containing the FV approximation of the negative Laplacian operator derived in Assignment 3.

1. Write down the three-step recurrence formula (iteration formula) for the problem (5), denoting the solution at t_k as $\mathbf{u}_k$. Provide an expression for $\mathbf{u}^1$ as well.
   Central difference time discretization:

$$\frac{\partial^2 u}{\partial t^2} = \frac{u^{k+1}(\mathbf{x}) - 2u^k(\mathbf{x}) + u^{k-1}(\mathbf{x})}{h_t^2} + \mathcal{O}(h_t^2)$$

Explicit time-stepping scheme:

$$\frac{u^{k+1}(\mathbf{x}) - 2u^k(\mathbf{x}) + u^{k-1}(\mathbf{x})}{c^2 h_t^2} = \Delta u^k(\mathbf{x}) + f^k(\mathbf{x})$$
$$u^{k+1}(\mathbf{x}) = 2u^k(\mathbf{x}) - u^{k-1}(\mathbf{x}) + (ch_t)^2 \Delta u^k(\mathbf{x}) + (ch_t)^2 f^k(\mathbf{x})$$

This needs both $u^k(\mathbf{x})$ and $u^{k-1}(\mathbf{x})$ to run.
Approximation of initial conditions:
To start the iterations we need $u^1(\mathbf{x})$ and $u^0(\mathbf{x})$. The initial conditions are:

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \frac{\partial u(\mathbf{x}, t)}{\partial t}|_{t=0} = v_0(\mathbf{x}), \mathbf{x} \in \Omega$$

Hence, $u^0(\mathbf{x}) = u_0(\mathbf{x})$. To approximate $u^1(\mathbf{x})$ consider the Taylor expansion

$$u^1(\mathbf{x}) = u^0(\mathbf{x}) + \frac{\partial u(\mathbf{x}, t)}{\partial t}|_{t=0} h_t + \frac{1}{2}\frac{\partial^2 u(\mathbf{x}, t)}{\partial t^2}|_{t=0} h_t^2 + \mathcal{O}(h_t^3)$$

Since one-sided FD approximation

$$\frac{u^1(\mathbf{x}) - u^0(\mathbf{x})}{h_t} = \frac{\partial u(\mathbf{x}, t)}{\partial t}|_{t=0} + \mathcal{O}(h_t)$$

gives only $\mathcal{O}(h_t)$ approximation of the initial condition, we need to retain one more term in the approximation of $u^1(\mathbf{x})$.
$\mathcal{O}(h_t^2)$ approximation of initial conditions:

$$u^1(\mathbf{x}) = u^0(\mathbf{x}) + \frac{\partial u(\mathbf{x}, t)}{\partial t}|_{t=0} h_t + \frac{1}{2}\frac{\partial^2 u(\mathbf{x}, t)}{\partial t^2}|_{t=0} h_t^2 + \mathcal{O}(h_t^3) = u_0(\mathbf{x}) + h_t v_0(\mathbf{x}) + \frac{(ch_t)^2}{2}[\Delta u_0(\mathbf{x}) + f^0(\mathbf{x})] + \mathcal{O}(h_t^3)$$

where we have used the wave equation at t=0:

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}|_{t=0} = \Delta u(\mathbf{x}, 0) + f(\mathbf{x}, 0)$$

Hence, given $u_0(\mathbf{x}), v_0(\mathbf{x})$, and $f^k(\mathbf{x})$, the time-stepping iterations are:

$$u^0(\mathbf{x}) = u_0(\mathbf{x})$$
$$u^1(\mathbf{x}) = u_0(\mathbf{x}) + h_t v_0(\mathbf{x}) + \frac{(ch_t)^2}{2}[\Delta u_0(\mathbf{x}) + f^0(\mathbf{x})]$$
$$u^{k+1}(\mathbf{x}) = 2u^k(\mathbf{x}) - u^{k-1}(\mathbf{x}) + (ch_t)^2[\Delta u^k(\mathbf{x}) + f^k(\mathbf{x})]$$

Time-stepping with FDM/FVM Laplacian:
After discretization in space:$-\Delta u^k(\mathbf{x}) -> A\mathbf{u}^k$:
$\mathbf{u}^0, \mathbf{w}^0$ - given
$\mathbf{u}^1 = \mathbf{u}^0 + h_t \mathbf{w}_0 + \frac{(ch_t)^2}{2}[-A\mathbf{u}^0 + \mathbf{f}^0]$
$\mathbf{u}^{k+1} = \mathbf{2}u^k - \mathbf{u}^{k-1} + (ch_t)^2[-A\mathbf{u}^k + \mathbf{f}^k]$

2. Derive the CFL stability condition for the three-step recurrence with the FVM discretization, inhomogeneous k(x, y), and a zero source f (x, y, t) = 0.
Time-stepping with FDM Laplacian:
We consider the iterations:

$$\mathbf{u}^{k+1} = \mathbf{2}u^k - \mathbf{u}^{k-1} + (ch_t)^2[-A\mathbf{u}^k + \mathbf{f}^k]$$

Assuming $A = V\Lambda V^{-1}$ exists, let $\mathbf{u}^k = V\mathbf{b}^k$, same as in the method of lines. Then we obtain iterations for $\mathbf{b}^k$:

$$\mathbf{b}^{k+1} = 2\mathbf{b}^k - \mathbf{b}^{k-1} - (ch_t)^2\Lambda\mathbf{b}^k, \mathbf{b}^k \in \mathbb{R}^n$$
$$b_j^{k+1} = 2b_j^k - b_j^{k-1} - (ch_t)^2\lambda_j b_j^k, j = 1, ..., n$$

Let $b_j^k = (b_j)^k$, i.e., b_j to the power k. Dividing by $(b_j)^{k-1}$ and solving the quadratic equation for b_j:

$$(b_j)^2 - [2 - (ch_t)^2\lambda_j]b_j + 1 = 0$$
$$b_j = \frac{1}{2}[2 - (ch_t)^2\lambda_j \pm \sqrt{(2 - (ch_t)^2\lambda_j)^2 - 4}$$

Condition on exact b_j:

- In the exact method-of-lines solution $|b_j(t_k)|$ are oscillating in time. There is no exponential decay/growth.
- Hence, in the numerical time stepping we need to have $|b_j| = 1$, so that $|(b_j)^k| = |b_j|^k = 1^k = 1$, since otherwise we get exponential decay/growth of $(b_j)^k$ with k
- This is only possible if in the formula for b_j we set:

$$(ch_t)^2\lambda_j = 4$$

Numerical dispersion:
Hence, ideally, we would like to have

$$\frac{(ch_t)^2}{4}\lambda_j = 1$$

However, e.g., in the 1D case, we have

$$\lambda_j \frac{4}{h_x^2}\sin^2(\frac{\pi j}{2N}),$$

and it is not possible to choose h_t and h_x in such a way that $(\frac{ch_t}{h_x})^2\sin^2(\frac{\pi j}{2N}) = 1$, for all j=1,...,n;

- Therefore,the numerical time-evolution coefficients b_j^k of different eigenvectors $\mathbf{v}_j$ will never be perfectly "synchronized"- numerical dispersion.

CFL stability condition:
At least we can avoid divergence over time by requiring:

$$|b_j| \leq 1$$

$$\frac{(ch_t)^2}{4}\lambda_j \leq 1$$

This is a general condition. Example, in 1D case:

$$\frac{(ch_t)^2}{4}\frac{4}{h_x^2}\sin^2(\frac{\pi j}{2N}) \leq 1$$

$$\frac{(ch_t)^2}{h_x^2}\sin^2(\frac{\pi j}{2N}) \leq 1$$

$$\frac{(ch_t)^2}{h_x^2} \leq 1$$

$$0 \leq \frac{ch_t}{h_x^2} \leq 1$$

This is the Courant-Friedrichs-Lewy (CFL) stability condition in 1D, and ch_t/h_x is called the Courant number.

3.2 Implementation and Numerical Experiments

Save and submit the code for this wavefield problem in the file Assignment-4-wave.py. Re-use the Python functions from Assignment 3 and the previous Section as much as possible. Here and in what follows, use the doubly-uniform grid with $N_x = 200$ and $N_y = 100$.

1. Define the Python function that implements one iteration of the three- step method:
   Input:
   - u0Start – lexicographic vector of the solution (initial condition) at t =tStart
   - u1Start – lexicographic vector of the solution (initial condition) at t =tStart+dt
   - dt – time step

   Output:
   - uEnd – lexicographic vector of the solution at t =tStart+2*dt

   Done in Jupyter Notebook.

2. What is the maximal time-step size for your problem according to your previously derived CFL bound? Given T = 4, what is the minimal number of time steps N_t that the three-step method must take to compute the solution u of the wave equation at t = T ?
   Done in Jupyter Notebook.
   The stable time step is 0.0025000000000000005.
   The stable number of time steps is 1599.9999999999998.

3. Make a movie of wavefield motion for $t \in [0, 4]$. Provided your function stepWave() works as expected, you may use the following piece of code. Show the result to your TA.
   Done in Jupyter Notebook.

4. Insert a figure of the numerical approximation of u(x, y, T) for T = 4 in your report.
   Done in Jupyter Notebook.

5. Explain why the movie and the figure make physical sense.

NB: I couldn't make my code up to the mark for the exercises towards the end, because most of my code for the exercises in the beginning took ages to run, creating memoryerrors and all sorts of issues, which took almost two entire exercise classes to resolve. Since it took so long to run some of the code in the beginning, I did not have enough time to run the code towards the end and to debug the errors in it, despite having worked on this all week.