High-order time integration Leap-Frog schemes combined with a discontinuous Galerkin method for the solution of the Maxwell equations

Dmitry V. Ponomarev

▶ **To cite this version:**

Dmitry V. Ponomarev. High-order time integration Leap-Frog schemes combined with a discontinuous Galerkin method for the solution of the Maxwell equations. [Research Report] RR-7067, INRIA. 2009, pp.91. inria-00424560

HAL Id: inria-00424560
https://inria.hal.science/inria-00424560

Submitted on 16 Oct 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

High-order time integration Leap-Frog schemes combined with a discontinuous Galerkin method for the solution of the Maxwell equations

Dmitry V. Ponomarev

N° 7067

Septembre 2009

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--7067--FR+ENG

High-order time integration Leap-Frog schemes combined with a discontinuous Galerkin method for the solution of the Maxwell equations

Dmitry V. Ponomarev*

Thème NUM — Systèmes numériques

Projet NACHOS

Rapport de recherche n° 7067 — Septembre 2009 — 90 pages

Abstract: In this report, after pedagogical mathematical insight into basic notions of numerical analysis for differential equations, more specific Discontinuous Galerkin (DG) method is introduced. Afterwards, the DG method is combined with a fourth-order staggered Leap-Frog (LF4) scheme to be applied to the solution of the Maxwell equations wave-propagation problem. Stability analysis of the resulting scheme is performed and some peculiarities related with the choice of basis functions in the DG method are stressed.

Key-words: High-order time integration schemes, Discontinuous Galerkin method, staggered Leap-Frog scheme, Maxwell equations.

* Dm.V.Ponomarev@gmail.com

Unité de recherche INRIA Sophia Antipolis

2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)

Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Schémas saute-mouton d'ordre élevé combinée à une méthode Galerkin discontinue pour la résolution numérique des équations de Maxwell

Résumé : Dans ce rapport, aprés un aperçu pédagogique des notions de base de l'analyse numérique des équations différentielles, on étudie plus précisément une méthode Galerkin discontinue combinée à un schéma saute-mouton d'ordre 4 pour la résolution des équations de Maxwell. On réalise une analyse de stabilité du schéma résultant et on souligne quelques particularités liées au choix des fonctions de base dans la méthode Galerkin discontinue.

Mots-clés : Schémas d'intégration en temps d'ordre élevé, méthode Galerkin discontinue, schéma saute-mouton équations de Maxwell.

1 Introduction

This work has been conducted under the framework of first year Master internship and encompasses both educational and research aspects of the selected topic in numerical analysis.

In the first part of the present paper, basic definitions and ideas of numerical methods will be covered by considering and analyzing finite difference methods with few important examples given.

Next, another numerical technique, so-called Discontinuous Galerkin method, will be introduced and illustrated on a simple advection equation problem.

This method being very flexible and with help of appropriate finite difference scheme gives opportunity to gain desired accuracy in solving time-dependent problems.

In the last section, application of the Discontnuous Galerkin method combined with particular finite difference scheme, staggered Leap-Frog of the fourth order, to an electromagnetic wave propagation problem governed by Maxwell's equations will be given and stability study will be stressed.

2 Approximation of ODEs and PDEs with finite differences

Although the majority of problems are mathematically formulated in PDE form, it is reasonable to start with considering an ODE problem, not just because of its simplicity, but also due to the fact that it is useful for solving PDEs. For example, application of semi-discretized methods for solving a PDE yields a set of ODEs: in a time-dependent PDE problem we do discretization in space at every time step and thus end up with ODEs in time (this is so-called method of lines). Therefore, it is essential to introduce some basic concepts and methods for numerical solution of an ODE. Due to the fact that a high-order ODE is equivalent to the system of the first order ODEs, the most crucial is to consider the following ODE problem:

$$\begin{cases} \dfrac{dy}{dt} \equiv y' = f(t, y),\ t > 0, \\ y(0) = y_0, \end{cases} \tag{1}$$

where $y(t)$, $f(t, y)$ can be either functions or vector-functions.

Later on, in all numerical methods we intend to discuss in the current work we will use the following notations: k stands for the time step, y_n are approximations of the solution at $t_n = nk$ (that is $y_n \approx y(t_n)$ for all integer n starting from 0) and $f_n = f(t_n, y_n)$.

2.1 Runge-Kutta methods

To introduce such a powerful instrument as Runge-Kutta methods, we start with rewriting the ODE of (1) in the integral form:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt. \tag{2}$$

One can see that use of the midpoint formula for integration leads to $y_{n+1} = y_n + kf\left(t_n + \frac{k}{2}, y\left(t_n + \frac{k}{2}\right)\right)$ where $y\left(t_n + \frac{k}{2}\right)$ can be evaluated using just Euler method: $y\left(t_n + \frac{k}{2}\right) = y_n + \frac{k}{2}f_n$ (which preserves here the second order approximation of the midpoint formula due to the multiplication of f by k). This particular second-order method referred as RK2 gave inspiration to formulate the general idea of what is called Runge-Kutta methods.

All the Runge-Kutta methods are one-step methods (which means that to find value y_{n+1} one needs to know just the value at the previous time step y_n), however, within one time step we have internal stages.

General s-stage Runge-Kutta method reads:

$$\begin{cases} Y_i = y_n + k\sum_{j=1}^{s} a_{ij} f(t_n + c_i k, Y_j),\ 1 \le i \le s, \\ y_{n+1} = y_n + k\sum_{i=1}^{s} b_i f(t_n + c_i k, Y_i), \end{cases}$$

where the coefficients a_{ij}, b_i, c_i can be found from some consistency conditions (we refer, for example, to [3]) which in general become extremely cumbersome with growth of number of stages s.

If the matrix a_{ij} is lower diagonal, then a Runge-Kutta method is explicit. Order of accuracy of Runge-Kutta methods is usually less than number of stages s and equal to it

just for a couple of methods, one of them is the classical fourth-order method RK4. Since the computational difficulty imposes restrictions on usage of higher-order Runge-Kutta methods, the RK4 method, having also good stability characteristics, is the most commonly used and sometimes is even referred as just the Runge-Kutta method.

2.2 Linear multistep methods

A linear s-step method is generally given in the following form:

$$\sum_{j=0}^{s} \alpha_j y_{n+1-j} = k \sum_{j=0}^{s} \beta_j f_{n+1-j},$$

where, by convention, we set $\alpha_0 = 1$.

If $\beta_0 = 0$, then the method is explicit and we can write:

$$y_{n+1} = \sum_{j=1}^{s} \left(-\alpha_j y_{n+1-j} + k\beta_j f_{n+1-j} \right).$$

Adams methods are based on using an interpolating polynomial to approximate $f(t, y)$ in the integral form (2) and easily perform integration. If interpolating polynomial is driven through the points t_n, t_{n-1}, ..., t_{n-s+1} , the methods are explicit and they form **Adams-Bashforth family**. In case we make interpolating polynomial additionally pass through t_{n+1}, we derive **Adams-Moulton family** of methods which are obviously implicit (since the right-hand side involves y_{n+1}).

Another commonly used family of methods, **BDF (Backward Differentiation Formula methods)**, also employs polynomial interpolation through the points t_{n+1}, t_n, t_{n-1}, ..., t_{n-s+1} but for approximation of $y(t)$, not $f(t, y)$. Once it is approximated, we compute

the derivative and plug it directly to the ODE of (1) where in the right-hand side we take $f(t, y) = f(t_{n+1}, y_{n+1})$. After that it remains to solve this for y_{n+1}.

However, when one intends to use a linear multistep method, usually a Runge-Kutta method is still needed to get initial steps in order to start the multistep method.

2.3 Stability, consistency, convergence

To proceed with the notion of stability, we introduce a numerical scheme operator $\mathbb{N}_\pi$ such that mesh function $y_\pi(t)$ corresponding to the exact solution (i.e. $y_\pi(t_n) = y_n$) satisfies equation $\mathbb{N}_\pi y_\pi(t_n) = 0$ for all $n = 0, \ldots, N$.

Then we can define **stability (0-stability)** in the following way (see [1]): if there exist positive constants k_0, K such that for any mesh functions x_π and z_π for $k \leq k_0$ one has:

$$|x_n - z_n| \leq K\{|x_0 - z_0| + \max_{1 \leq j \leq N} |\mathbb{N}_\pi x_\pi(t_j) - \mathbb{N}_\pi z_\pi(t_j)|\},$$

for all $1 \leq n \leq N$, then the method with operator $\mathbb{N}_\pi$ is called 0-stable. In other words, stability ensures that the numerical solution obtained by numerical method corresponding to N_π does not blow up.

One can note that application of a numerical method operator to the exact solution computed at one of the points t_n gives local truncation error:

$$\mathbb{N}_\pi y(t_n) = d_n.$$

If we assume:

$$\max_n |d_n| = O(k^p),$$

for all problems with sufficiently smooth solutions, then the method is said to have the **order of accuracy** p.

In case $p \geq 1$, a method is called **consistent**.

Assume a method to be consistent of order p and stable, then:

$$|y_n - y(t_n)| \leq K \max_n |d_n| = O(k^p),$$

so the method is **convergent** of order p. That is to say that consistency and stability imply convergence.

One of the practical way to study stability of an ODE method is to consider its application to the test equation:

$$y' = \lambda y. \tag{3}$$

Obviously, to avoid solution for this equation to be unbounded, one has to have non-positive real part of λ:

$$\Re e\lambda \leq 0.$$

In a similar way, for a numerical method applied to (3) we define a region of the z-complex plane (denoting $z = k\lambda$) where:

$$|y_{n+1}| \leq |y_n|,$$

for all $n = 0, 1, 2, \ldots$, this region we call the **region of absolute stability.**

If the region of absolute stability of a method contains the entire left half-plain of z, we call such method **A-stable**.

We define the **stability function** $R(z)$ in a way that:

$$y_{n+1} = R(z)y_n, \tag{4}$$

$$y_n = R(z)^n y_0. \tag{5}$$

Then the region of absolute stability corresponds to:

$$|R(z)| \leq 1. \tag{6}$$

For some methods (for example, trapezoidal or midpoint), in spite of $|R(z)| < 1$ for finite z, we might have $\lim_{z \to -\infty} |R(z)| = 1$ that is not very good characteristic of the method unless the time step k is very small: because solutions for smaller λ (i.e. greater $|\lambda|$ which are referred as higher modes) are damped less than bigger ones (which are lower modes) that contradicts to behavior that exact solution of (3) exhibits with respect to change of parameter λ. This leads to the definition of another type of stability: a method is called **L-stable** or having **stiff decay** if its stability function satisfies the following condition:

$$\lim_{|z| \to \infty} |R(z)| = 0. \tag{7}$$

Generally, we define **stiffness** in the following way (although there is no proper unique definition, here we refer to [1]) - we call an ODE problem stiff if the absolute stability condition for an explicit Runge-Kutta method impose higher restriction on step size than it is needed for achieving desired accuracy.

If we have a linear system of ODEs or linear multistep method, then $R(z)$ will be a matrix and in the conditions (6), (7) instead of modulus we should write spectral radius of this matrix. This will be demonstrated further on examples (see the next section).

Talking about absolute stability, it makes sense to write down explicitly stability function for Runge-Kutta methods (due to their high-importance). A general Runge-Kutta method for the test equation (3) reads:

$$\begin{cases} Y_i = y_n + z\sum_{j=1}^{s} a_{ij}Y_j, \\ y_{n+1} = y_n + z\sum_{j=1}^{s} b_j Y_j. \end{cases}$$

Rewriting this is in matrix form we have:

$$\begin{cases} Y = y_n + zAY \quad \Rightarrow Y = (I - zA)^{-1}y_n, \\ y_{n+1} = y_n + zb^T Y = (1 + zb^T(I - zA)^{-1}\mathbf{1})y_n. \end{cases}$$

Therefore, the stability function for Runge-Kutta methods is given by:

$$R(z) = 1 + zb^T(I - zA)^{-1}\mathbf{1},$$

where $\mathbf{1} = (1, \ldots, 1)^T$ is the vector having dimension s.

Another essential tool to study stability of a numerical scheme is to use **Fourier analysis**.

Given an explicit scheme in general form:

$$y_j^{n+1} = \sum_{m=-l}^{r} b_m y_{j+m}^n, \tag{8}$$

where $y_j^n \approx y(t_n, x_j)$ are approximations of solution on an uniform grid with time step k and step in space h.

We apply this scheme to the constant coefficient PDE problem with periodic boundary conditions.

Due to periodicity of the problem, the solution can be expanded in the Fourier series:

$$y(x,t) = \sum_{j=-\infty}^{\infty} \alpha_j(t) e^{\frac{2\pi i j x}{L}},$$

with the coefficients determined by:

$$\alpha_j(t) = \frac{1}{L} \int_0^L y(\xi,t) e^{-\frac{2\pi i j \xi}{L}} d\xi.$$

Recalling Parseval's equality:

$$\|y(x,t)\|_{L_2} = \sum_{j=-\infty}^{\infty} |\alpha_j(t)|^2,$$

we conclude that we can study stability by analyzing behavior in time of the coefficients $\alpha_j(t)$. Therefore, using (8) we compute:

$$\alpha_j(t_{n+1}) = \alpha_j(t_n + k) = \frac{1}{L} \int_0^L \underbrace{y(\xi, t_n + k)}_{= \sum_{m=-l}^{r} b_m y(\xi + mh, t_n)} \exp\left(-\frac{2\pi i j \xi}{L}\right) d\xi.$$

By means of substitution $\tilde{\xi} = \xi + mh$, the last expression transforms into:

$$\alpha_j(t_{n+1}) = \sum_{m=-l}^{r} \frac{b_m \exp\left(\frac{2\pi i m h}{L}\right)}{L} \int_{mh}^{L+mh} y(\tilde{\xi}, t_n) \exp\left(-\frac{2\pi i j \tilde{\xi}}{L}\right) d\tilde{\xi}.$$

Now we develop the integral:

$$\int_{mh}^{L+mh} \ldots = \int_0^L \ldots + \underbrace{\left[\int_L^{L+mh} \ldots - \int_0^{mh} \ldots\right]}_{=0},$$

using periodicity of the function under integral sign that results in vanishing of the term in the square brackets.

Finally, we arrive at:

$$\alpha_j(t_{n+1}) = \sum_{m=-l}^{r} b_m \exp\left(\frac{2\pi imh}{L}\right) \underbrace{\frac{1}{L}\int_0^L y(\tilde{\xi}, t_n) \exp\left(-\frac{2\pi ij\tilde{\xi}}{L}\right) d\tilde{\xi}}_{=\alpha_j(t_n)}.$$

Hence we have established a link between all the coefficients $\alpha_j(t_{n+1})$ and $\alpha_j(t_n)$:

$$\alpha_j(t_{n+1}) = g(\zeta)\alpha_j(t_n),$$

where:

$$g(\zeta) = \sum_{m=-l}^{r} b_m e^{im\zeta} \ , \ \zeta = \frac{2\pi h}{L}.$$

One can see that the so-called **amplification factor** (or **amplification matrix** in case of a linear system of PDEs or a linear multistep in time method) $g(\zeta)$ has the same meaning as the stability function $R(z)$ introduced above.

In a similar way to (6), for absolute stability we require:

$$|g(\zeta)| \leq 1. \tag{9}$$

In case of a linear system of PDEs or a linear multistep time method, we impose the same condition, known as **von Neumann condition,** on spectral radius of amplification matrix:

$$\rho(g(\zeta)) \leq 1. \tag{10}$$

However, one should be careful in the situation when $\rho(g(\zeta)) = 1$, namely, if $g(\zeta)$ has multiple eigenvalues that might cause instability.

2.4 Stability on examples

2.4.1 Heat equation

We consider the Dirichlet problem for the constant coefficient heat equation:

$$\begin{cases} y_t = a y_{xx}, \;\; t > 0,\, 0 < x < L, \\ y(0,t) = y(L,t) = 0, \;\; t \geq 0, \\ y(x,0) = y^0(x), \;\; 0 \leq x \leq L, \end{cases} \tag{11}$$

and do semi-discretization choosing uniform mesh in space: $x_0 = 0$, $x_1 = h$, $x_2 = 2h$, ..., $x_{N+1} = L$ where $h = \dfrac{L}{N+1}$.

Thus, with second-order accuracy in space we have:

$$\begin{cases} (y_t)_j = a\dfrac{y_{j-1} - 2y_j + y_{j+1}}{h^2}, \;\; j = 1, \ldots, N, \\ y_0 = y_{N+1} = 0, \end{cases}$$

where $y_j \equiv y_j(t) \approx y(x_j, t)$ (for $j = 0, \ldots, N+1$), that is approximation of the solution in x_i at given time t.

We can rewrite the same in the vector form:

$$\begin{cases} \mathbf{y}_t = A\mathbf{y}, \\ y_0 = y_{N+1} = 0, \end{cases}$$

where:

$$A = \frac{a}{h^2}\begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ 0 & \dots & \dots & \dots & 0 \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix},$$

is a symmetric negative definite (as it will be clear later from its spectrum) matrix and $\mathbf{y} = (y_1, y_2, \dots, y_{N-1}, y_N)^T$ is the vector of the unknowns.

Motivated by the exact solution to the Dirichlet eigenvalue problem:

$$\begin{cases} y'' = \lambda y, \;\; 0 < x < L, \\ y(0) = y(L) = 0, \end{cases}$$

we can make the following guess for eigenvectors of the matrix A:

$$\mathbf{v}_l^{(A)} = \begin{pmatrix} \sin\left(\frac{\pi l h}{L}\right) \\ \sin\left(\frac{2\pi l h}{L}\right) \\ \dots \\ \sin\left(\frac{(N-1)\pi l h}{L}\right) \\ \sin\left(\frac{N\pi l h}{L}\right) \end{pmatrix}, \; l = 1, \dots, N.$$

Indeed, by utilizing the well-known trigonometric formulas, we check:

$$-2\sin\left(\frac{\pi l h}{L}\right) + \sin\left(\frac{2\pi l h}{L}\right) = -2\sin\left(\frac{\pi l h}{L}\right) + 2\sin\left(\frac{\pi l h}{L}\right)\cos\left(\frac{\pi l h}{L}\right) =$$
$$= -2\sin\left(\frac{\pi l h}{L}\right)\left(1 - \cos\left(\frac{\pi l h}{L}\right)\right) = -4\sin\left(\frac{\pi l h}{L}\right)\sin^2\left(\frac{\pi l h}{2L}\right),$$

$$\sin\left((j-1)\frac{\pi lh}{L}\right) - 2\sin\left(j\frac{\pi lh}{L}\right) + \sin\left((j+1)\frac{\pi lh}{L}\right) =$$
$$= 2\sin\left(j\frac{\pi lh}{L}\right)\cos\left(\frac{\pi lh}{L}\right) - 2\sin\left(j\frac{\pi lh}{L}\right) = -4\sin\left(\frac{j\pi lh}{L}\right)\sin^2\left(\frac{\pi lh}{2L}\right),$$

$$\sin\left(\frac{(N-1)\pi lh}{L}\right) - 2\sin\left(\frac{N\pi lh}{L}\right) = \sin\left(\frac{(N-1)\pi lh}{L}\right) - 2\sin\left(\frac{N\pi lh}{L}\right) +$$
$$+\underbrace{\sin\left(\frac{(N+1)\pi lh}{L}\right)}_{=0} = -4\sin\left(\frac{N\pi lh}{L}\right)\sin^2\left(\frac{\pi lh}{2L}\right).$$

Thus we find out that $\mathbf{v}_l^{(A)}$ satisfies:

$$A\mathbf{v}_l^{(A)} = \lambda_l^{(A)}\mathbf{v}_l^{(A)},$$

with:

$$\lambda_l^{(A)} = -\frac{4a}{h^2}\sin^2\left(\frac{\pi lh}{2L}\right),\ l = 1, \ldots, N.$$

Now, we study stability for both the forward (explicit) and the backward (implicit) Euler schemes for the discretization in time.

For the **Forward Euler** scheme we have

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{k} = A\mathbf{u}^n \qquad \Rightarrow \qquad \mathbf{u}^{n+1} = B^{FE}\mathbf{u}^n,$$

where $B^{FE} = kA + I$ and I is the identity matrix.

Note that $\left(B^{FE}\right)^T B^{FE} = B^{FE}\left(B^{FE}\right)^T$ (due to the symmetry of the matrix A) and hence B^{FE} is a normal matrix. But operator norm of a normal matrix B (with respect to L_2 vector norm) is bounded by its spectral radius. Indeed, because of the fact that a normal matrix can be reduced to a diagonal D by an orthogonal transformation P (that is to say,

$B = P^T DP$), we have:

$$\begin{aligned} ||B|| &= \sup_{||x||=1} ||Bx|| = \sup_{||x||=1} |(Bx, Bx)|^{1/2} = \sup_{||x||=1} |(P^T DPx, P^T DPx)|^{1/2} = \\ &= \sup_{||x||=1} |(DTx, \underbrace{PP^T}_{=I} DPx)|^{1/2} = \sup_{||x||=1} |(x, P^T \underbrace{D^T D}_{=diag(\lambda^2)} Px)|^{1/2} = \\ &= |\lambda|_{max} \cdot |(x, \underbrace{P^T IP}_{=I} x)|^{1/2} = |\lambda|_{max} = \rho(B), \end{aligned}$$

where the supremum is attained at the normalized eigenvector corresponding to an eigenvalue with the maximal modulus.

Thus, to check stability it rests to find spectrum of the matrix B.

Turning back to our particular case, the eigenvalues of matrix B^{FE}, obviously, are:

$$\lambda_l^{(B^{FE})} = 1 + k\lambda_l^{(A)} = 1 - \frac{4ak}{h^2} \sin^2\left(\frac{\pi lh}{2L}\right),\ l = 1, \ldots., N,$$

and its spectral radius is:

$$\rho(B^{FE}) = \max_l |\lambda_l^{(B^{FE})}| = \left|1 - \frac{4ak}{h^2} \sin^2\left(\frac{\pi N}{2(N+1)}\right)\right| \approx \left|1 - \frac{4ak}{h^2}\right|.$$

Using analogy with (6) we can say that the scheme is absolute stable if:

$$\rho(B^{FE}) \leq 1.$$

This condition gives $|1 + k\lambda_l^{(A)}| \leq 1 \ \Rightarrow 1 - \frac{4ak}{h^2} \geq -1 \ \Rightarrow \frac{ak}{h^2} \leq \frac{1}{2}$. Therefore:

$$k \leq \frac{h^2}{2a}. \tag{12}$$

Since the absolute stability region is just the interior of the unit disk, the method is not A-stable, although conditionally stable.

Now we are moving to the **Backward Euler** scheme:

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{k} = A\mathbf{u}^{n+1} \qquad \Rightarrow \qquad \mathbf{u}^{n+1} = B^{BE}\mathbf{u}^n,$$

where $B^{BE} = (I - kA)^{-1}$.

In order to have boundedness by spectral radius, again we need to ensure that the matrix B^{BE} is normal, that is to check that:

$$(I-kA)^{-1}(I-kA^T)^{-1} = (I-kA^T)^{-1}(I-kA)^{-1}.$$

First notice that:

$$(I-kA)(I-kA^T) = I - kA - kA^T + k^2\underbrace{AA^T}_{=A^TA} = (I-kA^T)(I-kA).$$

Then taking inverse of both sides yields the desired result.

Hence it remains to find spectrum and estimate spectral radius of the matrix B^{BE}:

$$\lambda_l^{(B^{BE})} = \frac{1}{1-k\lambda_l^{(A)}} = \frac{1}{1+\dfrac{4ak}{h^2}\sin^2\left(\dfrac{\pi lh}{2L}\right)},\ l = 1, \ldots., N,$$

$$\rho(B^{BE}) = \frac{1}{1+\dfrac{4ak}{h^2}\sin^2\left(\dfrac{\pi N}{2(N+1)}\right)} \approx \frac{1}{1+\dfrac{4ak}{h^2}} < 1.$$

We can note that since the absolute stability region includes the whole negative ($k\lambda^{(A)}$) half-plane, this method is A-stable.

Generalization (in a similar way as we did with absolute stability a few lines before) of (7) yields imposing condition:

$$\lim_{k\lambda^{(A)}\to-\infty} \rho(B^{BE}) = 0,$$

if one wants to have L-stability. Evidently, here, higher harmonics are well damped. Thus, for the Backward Euler scheme we have both A- and L-stability.

Note that here discussing stability we were all the time talking about the absolute stability whereas there is a weaker definition of stability that just requires solution to have less than an exponential growth (this corresponds to the well-posedness of the differential equation problem). However, since the exact solution in our case does not grow in time (due to the maximum principle that is valid for the heat equation), general stability criterion in the first order of time-step k coincides with the absolute stability criterion we used.

2.4.2 Wave equation

Since the wave equation is exactly the aim of our study, we develop this subsection in more details.

Consider 1D wave equation problem with constant velocity c:

$$\begin{cases} y_{tt} = c^2 y_{xx}, \;\; 0 < x < L, \;\; t > 0, \\ y(x,0) = \phi(x), \;\; 0 \le x \le L, \\ y_t(x,0) = \psi(x), \;\; 0 \le x \le L, \\ y(0,t) = y(L,t) = a(t), \;\; t \ge 0. \end{cases} \tag{13}$$

Sometimes it can be more convenient to find an appropriate numerical method if the wave equation in (26) (which is a second-order PDE) is written as a symmetric system of two first-order PDEs.

Indeed, a simple substitution:

$$\begin{cases} u = cy_x, \\ v = y_t, \end{cases} \tag{14}$$

yields an equivalent to the original wave equation system:

$$\begin{cases} u_t = cv_x, \\ v_t = cu_x. \end{cases} \tag{15}$$

By means of straightforward differentiating initial and boundary conditions of (13) using (14) we obtain initial and boundary conditions for the equivalent equations (15):

$$\begin{cases} u(x,0) = c\phi'(x), \\ v(x,0) = \psi(x), \\ v(0,t) = v(L,t) = a'(t), \\ u_x(0,t) = u_x(L,t) = \dfrac{1}{c}a''(t). \end{cases} \tag{16}$$

Here we also utilized equations (15) to get the last couple of conditions (namely, the boundary conditions on u_x), however, the new problem we obtained is overdetermined and we will see in the demonstration at the end of this section that one of these conditions is redundant.

Forgetting about boundary conditions for a while, the system (15) can be written in the matrix form:

$$\mathbf{U}_t = \begin{pmatrix} 0 & c \\ c & 0 \end{pmatrix} \mathbf{U}_x, \tag{17}$$

where $\mathbf{U} = (u, v)^T$.

Performing discretization in space (in a same way for u and v):

$$\begin{cases} u_t = Av, \\ v_t = Au, \end{cases}$$

we have in the matrix form:

$$\tilde{\mathbf{U}}_t = \underbrace{\begin{pmatrix} 0 & cA \\ cA & 0 \end{pmatrix}}_{\equiv C_U} \tilde{\mathbf{U}}, \tag{18}$$

where: $\tilde{\mathbf{U}} = (u_1, \ldots, u_N, v_1, \ldots, v_N)^T$ is extended vector, A is $(N \times N)$ dimensional discretization matrix.

The block matrix C_U having dimension $(2N \times 2N)$ can be factorized as follows:

$$C_U = \underbrace{\begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}}_{\equiv P} \begin{pmatrix} cA & 0 \\ 0 & -cA \end{pmatrix} \underbrace{\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}}_{=P^T},$$

where P is an orthogonal matrix of similarity transformation (so $P^T = P^{-1}$).

Performing substitution of C_U in (18) and multiplying both sides of the equation by P^T, we obtain:

$$\tilde{\mathbf{V}}_t = \underbrace{\begin{pmatrix} cA & 0 \\ 0 & -cA \end{pmatrix}}_{\equiv C_V} \tilde{\mathbf{V}}, \tag{19}$$

where:

$$\tilde{\mathbf{V}} = P^T \tilde{\mathbf{U}}. \tag{20}$$

After discretization in time we arrive at:

$$\tilde{\mathbf{V}}^{n+1} = B_V \tilde{\mathbf{V}}^n. \tag{21}$$

In general, stability of the original problem (18) may not follow from stability of the block diagonalized problem (19) or the fully discretized problem (21). However, if matrix C_U is normal it can be reduced to block diagonal form by an orthogonal transformation P, and here it is exactly the case, hence we have:

$$||\tilde{\mathbf{U}}(t)|| \leq \underbrace{||P|| \cdot ||P^T||}_{=1} ||\tilde{\mathbf{U}}(0)||,$$

provided that all eigenvalues of matrix C_V have non-positive real part, that is, $\max(\Re e\lambda) \leq 0$. That means that stability of (19) implies stability of the original problem (18).

Moreover, in our situation, since we are lucky to have the matrix C_U in very special form, the similarity transformation matrix P is not dependent on h and thereby (due to (20)) stability conditions for $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ even for the fully discretized problem are equivalent. Here, as in the case of the heat equation problem considered above, talking about stability we all the time imply absolute stability, this is due to the fact that exact solution of the wave equation problem in the bounded domain is not growing in time.

Finally, we come to particular schemes and we start with the scheme named **Forward Time Centered Space (FTCS)** which approximates (17) in the following way:

$$\frac{\mathbf{U}_j^{n+1} - \mathbf{U}_j^n}{k} = \begin{pmatrix} 0 & c \\ c & 0 \end{pmatrix} \frac{\mathbf{U}_{j+1}^n - \mathbf{U}_{j-1}^n}{2h}.$$

Without loss of generality, let us first focus on the space discretization operator A (and therefore we omit writing indices for time steps for a while) applied to u (the procedure with v goes absolutely the same way) and find its spectrum.

In order to estimate it, we impose particular boundary conditions $u_0 = 1$, $u_{N+1} = 1$ that are completely artificial but serve our purpose[1]. Then:

$$\frac{u_{j+1} - u_{j-1}}{2h} = \lambda u_j, \quad j = 2, \ldots, 2N-1,$$

$$\frac{u_2 - 1}{2h} = \lambda u_1,$$

$$\frac{1 - u_{2N-1}}{2h} = \lambda u_{2N}.$$

This formulation can be written in the matrix form:

$$\frac{1}{2h}\begin{pmatrix} 0 & 1 & 0 & \ldots & 0 & 0 & 0 \\ -1 & 0 & 1 & \ldots & 0 & 0 & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & 0 & \ldots & -1 & 0 & 1 \\ 0 & 0 & 0 & \ldots & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \ldots \\ u_j \\ \ldots \\ u_{N-1} \\ u_N \end{pmatrix} + \frac{1}{2h}\begin{pmatrix} -1 \\ 0 \\ \ldots \\ 0 \\ \ldots \\ 0 \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ \ldots \\ u_j \\ \ldots \\ u_{N-1} \\ u_N \end{pmatrix}.$$

Like in the case of heat equation, here again, keeping in mind the solution for the continuous analogue of the problem:

[1] Though, generally discretization of boundary conditions may turn a stable scheme into unstable one, but here, as it will be clear after the calculations, the scheme happens to be unconditionally unstable, and to show instability it is enough to show that the scheme is unstable just for some specific choice of boundary conditions.

$$\begin{cases} u' = \lambda u, \;\; 0 < x < L, \\ u(0) = u(L) = 1. \end{cases}$$

We search the eigenvectors in the form:

$$u_l^{(A)} = \begin{pmatrix} \exp\left(\dfrac{2\pi i l h}{L}\right) \\ \exp\left(\dfrac{4\pi i l h}{L}\right) \\ \cdots \\ \exp\left(\dfrac{2\pi i j l h}{L}\right) \\ \cdots \\ \exp\left(\dfrac{2(N-1)\pi i l h}{L}\right) \\ \exp\left(\dfrac{2N\pi i l h}{L}\right) \end{pmatrix}, \; l = 1, \ldots., N.$$

Plugging this into the matrix form above and using the Euler's formulas for simplifying:

$$\exp\left(\frac{4\pi i l h}{L}\right) - \underbrace{\exp(i0)}_{=1} = 2i \sin\left(\frac{2\pi l h}{L}\right) \exp\left(\frac{2\pi i l h}{L}\right),$$

$$\exp\left(\frac{2\pi i (j+1) l h}{L}\right) - \exp\left(\frac{2\pi i (j-1) l h}{L}\right) = 2i \sin\left(\frac{2\pi l h}{L}\right) \exp\left(\frac{2\pi i j l h}{L}\right),$$

$$-\exp\left(\frac{2(N-1)\pi i l h}{L}\right) + \underbrace{\exp\left(\frac{2(N+1)\pi i l h}{L}\right)}_{=\cos(2\pi l)=1} = 2i \sin\left(\frac{2\pi l h}{L}\right) \exp\left(\frac{2N\pi i l h}{L}\right),$$

we conclude that $u_l^{(A)}$ are truly the eigenvectors that correspond to the eigenvalues

$$\lambda_l^{(A)} = \frac{i}{h} \sin\left(\frac{2\pi l h}{L}\right), \quad l = 1, \ldots, N.$$

Then for the block diagonalized matrix C_V (that has the same eigenvalues as C_U) we obtain:

$$\lambda_l^{(C_V)} = \pm \frac{ic}{h} \sin\left(\frac{2\pi l h}{L}\right), \quad l = 1, \ldots, N,$$

and it means that we have $2N$ eigenvalues in total, but only N of them are distinct (that is to say, each eigenvalue is of multiplicity 2).

Let us proceed with discretization in time:

$$\frac{\tilde{\mathbf{V}}^{n+1} - \tilde{\mathbf{V}}^n}{k} = C_V \tilde{\mathbf{V}}^n \qquad \Rightarrow \qquad \tilde{\mathbf{V}}^{n+1} = \underbrace{(kC_V + I)}_{=B_V} \tilde{\mathbf{V}}^n.$$

It follows that:

$$\lambda^{(B_V)} = k\lambda^{(C_V)} + 1,$$

and:

$$\rho^{(B_V)} = \sqrt{\frac{k^2 c^2}{h^2} + 1} > 1.$$

Hence we conclude that FTCS method is unconditionally unstable (no matter what time and space steps we take).

Now, after the breakdown of the previous scheme, we try to apply another method, the so-called **Leap-Frog (LF2)** scheme, which gives the second order approximation of the solution in both space and time:

$$\frac{y_j^{n+1} - 2y_j^n + y_j^{n-1}}{k^2} = c^2 \frac{y_{j+1}^n - 2y_j^n + y_{j-1}^n}{h^2}. \tag{22}$$

Unlike for the previous scheme we will check stability in a different way demonstrating an alternative approach.

Since we have periodic boundary conditions, we can use Fourier analysis discussed in the previous section, that is, we find amplification matrix and impose the condition on its spectral radius. Alternatively, since the original problem allows separation of variables and for spatial part Fourier analysis can be applied, we can search for a discrete solution in the

form:

$$y_j^n = G^n e^{i\xi x_j} = G^n e^{i\xi jh} = G^n e^{ij\zeta},$$

where we denote $\zeta = \xi h$.

In order to have the absolute stability, G (which is usually referred as the growth factor) must satisfy condition:

$$|G| \leq 1.$$

Plugging this into the scheme (22), we obtain the equation for G:

$$\begin{aligned}
&\left(G^{n+1} - 2G^n + G^{n-1}\right) e^{ij\zeta} = \frac{c^2k^2}{h^2} G^n e^{ij\zeta} \left(e^{i\zeta} - 2 + e^{-i\zeta}\right) \quad \Rightarrow \\
&\Rightarrow \quad G^2 - 2G + 1 = G\frac{2c^2k^2}{h^2}(\cos\xi - 1) \quad \Rightarrow \\
&\Rightarrow \quad G^2 - 2\left(1 - \frac{2c^2k^2}{h^2}\sin^2(\xi/2)\right) G + 1 = 0
\end{aligned}$$

.

Hence we have two roots - solutions for the growth factor:

$$G_{1,2} = \alpha \pm \sqrt{\alpha^2 - 1},$$

where we denote $\alpha = 1 - \dfrac{2c^2k^2}{h^2}\sin^2(\xi/2)$.

First, it is easy to see that if $|\alpha| > 1$, then for at least one of the root $|G| > 1$, which leads to instability.

Now, assume $|\alpha| \leq 1$. Obviously, we have 2 complex conjugated roots and thus:

$$|G_{1,2}| = \alpha^2 + (1 - \alpha)^2 \leq 1.$$

This is fulfilled automatically due to our assumption $|\alpha| \leq 1$. It means $\alpha \geq -1$ (since, evidently, $\alpha \leq 1$) which results in the well-known **Courant-Friedrichs-Lewy (CFL) con-**

dition:

$$k \leq \frac{h}{c}. \tag{23}$$

We should stress that this condition imposes much less limitation (the restriction is just linear in h) on a time step than the one for the heat equation problem (12) (where the restriction on a time step was quadratic in h) and thereby makes way for an explicit scheme.

Now we want to apply the LF2 scheme to discretize (15) instead of tackling original wave equation.

In order to do that, let us consider the following numerical scheme:

$$\begin{cases} \dfrac{u_j^{n+1} - u_j^{n-1}}{2k} = c\dfrac{v_{j+1}^n - v_{j-1}^n}{2h}, \\ \dfrac{v_j^{n+1} - v_j^{n-1}}{2k} = c\dfrac{u_{j+1}^n - u_{j-1}^n}{2h}. \end{cases} \tag{24}$$

One can easily see that in this scheme, values (in both time and space) on one hand side are computed in between of the points used on the other and vice versa, therefore the symmetry guarantees the second order of accuracy in time and space. However, as we are going to show now, more natural way to preserve the symmetry (even within a single step) is using so-called staggered grid as it follows. Assume we prescribe values of u in time points in usual way but for space points we define its values on a dual grid that we denote with half-integer indices; for v everything is the other way round - we use regular mesh in space and half-integer time steps.

This staggered grid is illustrated on Fig. 1.

The numerical scheme on this mesh reads:

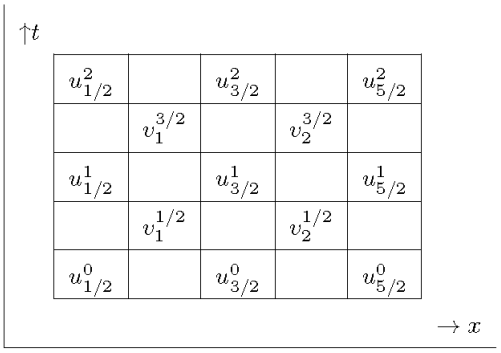

Fig. 1: Staggered grid pattern

$$\begin{cases} \dfrac{u_{j+1/2}^{n+1} - u_{j+1/2}^{n}}{k} = c\dfrac{v_{j+1}^{n+1/2} - v_{j}^{n+1/2}}{h}, \\ \dfrac{v_{j+1}^{n+3/2} - v_{j+1}^{n+1/2}}{k} = c\dfrac{u_{j+3/2}^{n+1} - u_{j+1/2}^{n+1}}{h}. \end{cases} \tag{25}$$

This is a particular case (for full discretization in space and time) of the scheme usually referred as StaggeredLF2.

If we substitute here midpoint approximation of (14):

$$\begin{cases} u_{j+1/2}^{n} = c\dfrac{y_{j+1}^{n} - y_{j}^{n}}{h}, \\ v_{j}^{n+1/2} = \dfrac{y_{j}^{n+1} - y_{j}^{n}}{k}, \end{cases}$$

the first equation in (25) turns out to be trivially satisfied and the second gives:

$$\frac{y_j^{n+2} - 2y_j^{n+1} + y_j^n}{k^2} = c^2\frac{y_{j+1}^{n+1} - 2y_j^{n+1} + y_{j-1}^{n+1}}{h^2},$$

which is exactly (after re-indexing in time $(n+1) \to n$) the LF2 scheme introduced above for the original wave equation. Thus, the equivalence of (22) and (25) is now shown.

Demonstration of the StaggeredLF2 scheme

Let us illustrate stable and unstable behaviors of the StaggeredLF2 scheme on a particular example of the wave equation problem:

$$\begin{cases} y_{tt} = c^2 y_{xx}, \;\; 0 < x < L, \;\; t > 0, \\ y(x,0) = \sin\left(\dfrac{\pi x}{L}\right), \;\; 0 \leq x \leq L, \\ y_t(x,0) = 0, \;\; 0 \leq x \leq L, \\ y(0,t) = y(L,t) = 0, \;\; t \geq 0. \end{cases} \tag{26}$$

The problem obviously has the analytical solution:

$$y(x,t) = \frac{1}{2}\left(\sin\left(\frac{\pi(x+ct)}{L}\right) + \sin\left(\frac{\pi(x-ct)}{L}\right)\right). \tag{27}$$

As it was described before, introducing new variables (14), we can transform the wave equation problem (13) into a system of first order PDEs (15) with the corresponding initial and boundary conditions (16). Applying this to our particular case (26), we arrive at:

$$\begin{cases} u_t = c v_x, \\ v_t = c u_x, \\ u(x,0) = \dfrac{\pi c}{L}\cos\left(\dfrac{\pi x}{L}\right), \\ v(x,0) = 0, \\ v(0,t) = v(L,t) = 0, \\ u_x(0,t) = u_x(L,t) = 0. \end{cases} \tag{28}$$

Exact solution of (28) follows directly from (27) and (14), and it is given by:

$$\begin{cases} u(x,t) = \dfrac{\pi c}{2L}\left(\cos\left(\dfrac{\pi(x+ct)}{L}\right) + \cos\left(\dfrac{\pi(x-ct)}{L}\right)\right), \\ v(x,t) = \dfrac{\pi c}{2L}\left(\cos\left(\dfrac{\pi(x+ct)}{L}\right) - \cos\left(\dfrac{\pi(x-ct)}{L}\right)\right). \end{cases} \tag{29}$$

Now we move to numerical solution of (28). The StaggeredLF2 scheme (25) gives:

$$\begin{aligned} u_{j+1/2}^{n+1} &= u_{j+1/2}^{n} + \frac{kc}{h}\left(v_{j+1}^{n+1/2} - v_{j}^{n+1/2}\right), \qquad j = 0, \ldots, N, \\ & \qquad\qquad\qquad\qquad\qquad\qquad n = 0, \ldots, M, \end{aligned}$$

$$\begin{aligned} v_{j+1}^{n+3/2} &= v_{j+1}^{n+1/2} + \frac{kc}{h}\left(u_{j+3/2}^{n+1} - u_{j+1/2}^{n+1}\right) = \\ &= v_{j+1}^{n+1/2} + \frac{kc}{h}\left(u_{j+3/2}^{n} - u_{j+1/2}^{n} + \right. \\ &\left. + \frac{kc}{h}\left(v_{j+2}^{n+1/2} - 2v_{j+1}^{n+1/2} + v_{j}^{n+1/2}\right)\right), \qquad j = 0, \ldots, N-1, \\ & \qquad\qquad\qquad\qquad\qquad\qquad n = 0, \ldots, M. \end{aligned} \tag{30}$$

The second expression is more convenient to write replacing $(j+1) \to j$, namely:

$$\begin{aligned} v_{j}^{n+3/2} &= v_{j}^{n+1/2} + \frac{kc}{h}\left(u_{j+1/2}^{n} - u_{j-1/2}^{n} + \right. \\ &\left. + \frac{kc}{h}\left(v_{j+1}^{n+1/2} - 2v_{j}^{n+1/2} + v_{j-1}^{n+1/2}\right)\right), \qquad j = 1, \ldots, N, \\ & \qquad\qquad\qquad\qquad\qquad\qquad n = 0, \ldots, M. \end{aligned} \tag{31}$$

The initial and boundary conditions read:

$$\begin{aligned} u_{j+1/2}^{0} &= \frac{\pi c}{L}\cos\left(\frac{\pi x_{j+1/2}}{L}\right), \quad j = 0, \ldots, N+1, \\ v_{j}^{1/2} &= 0, \qquad\qquad j = 0, \ldots, N+1, \\ v_{0}^{n+1/2} &= 0, \qquad\qquad n = 0, \ldots, M+1, \\ v_{N+1}^{n+1/2} &= 0, \qquad\qquad n = 0, \ldots, M+1, \\ u_{N+3/2}^{n} &= u_{N+1/2}^{n}, \qquad n = 0, \ldots, M+1, \end{aligned} \tag{32}$$

where the last expression is a consequence of the second order approximation of u_x at the boundary of the staggered grid.

We cannot impose a similar condition on the other boundary, since we do not have value $u^n_{-1/2}$ in order to discretize it symmetrically (and therefore preserve the second order accuracy), however this condition is not needed (we have already mentioned redundancy of boundary conditions when were formulating (16)), because the first formula in (30), that is valid for $j = 0$, can be utilized:

$$u^{n+1}_{1/2} = u^n_{1/2} + \frac{kc}{h}\left(v^{n+1/2}_1 - \underbrace{v^{n+1/2}_0}_{=0}\right) = u^n_{1/2} + \frac{kc}{h}v^{n+1/2}_1, \tag{33}$$

and thereby this value at the boundary is transmitted step by step from the initial one at $t = 0$.

The formulas above (30), (31), (32), (33) allow us to perform calculation at all space and time points.

Once all values of $u^n_{j+1/2}$ and $v^{n+1/2}_j$ are computed, we can get back to solution of the original wave equation problem (26) simply by integrating one of the expressions (14) using midpoint rule (to preserve the second order accuracy):

$$\begin{gathered} y^n_j = y(x_j, t_n) = y(0, t_n) + \frac{1}{c}\int_0^{x_j} u(\xi, t_n)d\xi \approx y^n_0 + \frac{h}{c}\sum_{l=0}^{j-1} u^n_{l+1/2} \\ (j = 0, \ldots, N+1, \quad n = 0, \ldots, M+1)\,, \end{gathered} \tag{34}$$

or:

$$\begin{gathered} y^n_j = y(x_j, t_n) = y(x, 0) + \int_0^{t_n} v(x_j, \theta)d\theta \approx y^0_j + k\sum_{l=0}^{n} v^{l+1/2}_j \\ (j = 0, \ldots, N+1, \quad n = 0, \ldots, M+1). \end{gathered} \tag{35}$$

Each of these two formulas has its own advantages and drawbacks. The first of them (34) does not require storage of the solution at all previous times, hence we can benefit

from StaggeredLF2 being an explicit method, whereas the second one (35) employing time integration gives better accuracy since usually (due to stability condition) we have more temporal points than spatial.

On Fig. 2 we show a comparison of the numerical solutions with the exact ones at different time stations for the following values of numerical parameters: length of physical (spatial) domain $L = 10$, velocity $c = 1.5$, total time of integration $T = 10$, number of space intervals $N + 1 = 50$, number of time intervals $M + 1 = 100$.

The same is shown on Fig. 3 but with the number of time intervals $M + 1 = 50$ yielding violation of the CFL condition (23) that results in the instability in time illustrated below.

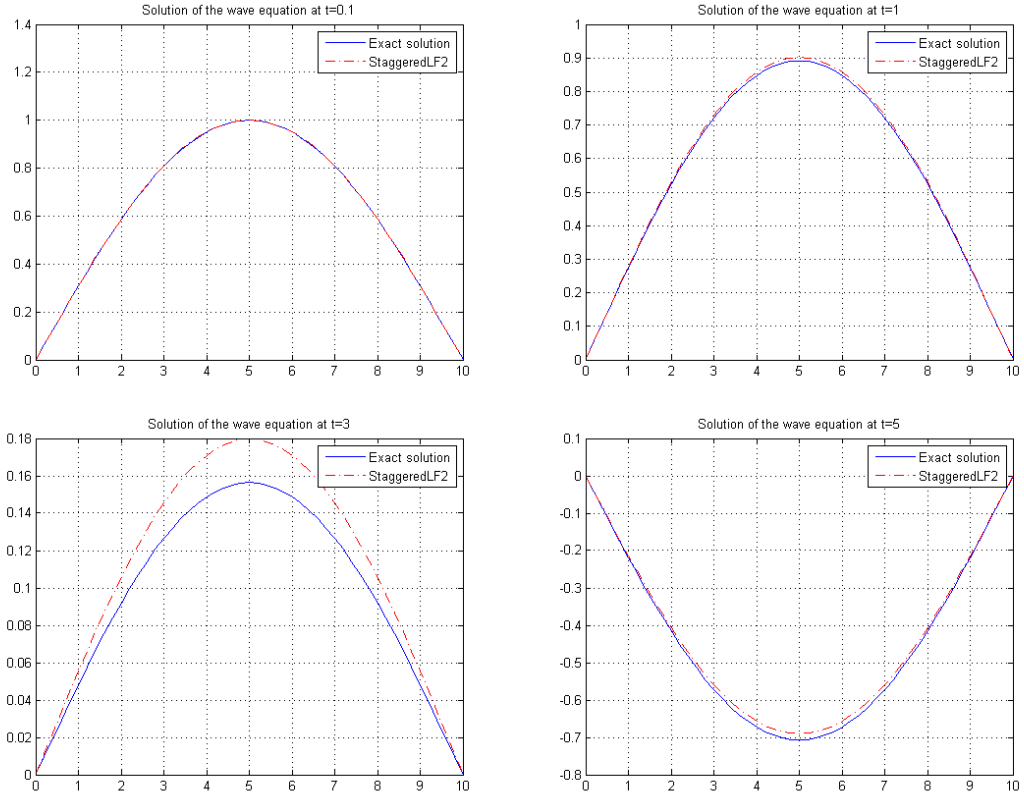

Fig. 2: Numerical solution for the wave equation with the CFL condition held

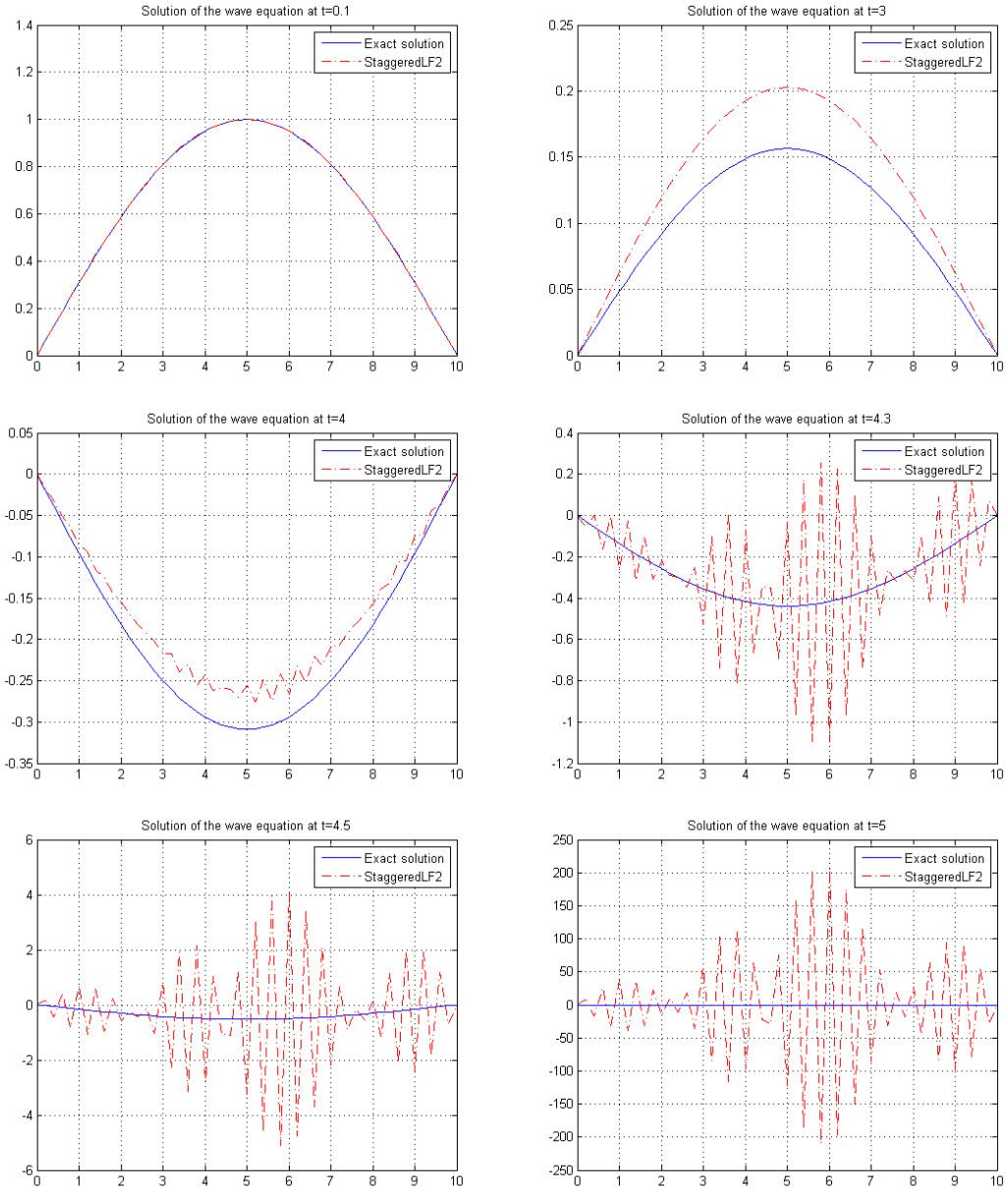

Fig. 3: Numerical solution for the wave equation with the CFL condition violated

3 Discontinuous Galerkin method

Besides the finite differences method, there are other numerical methods that are widely used for discretization in space, such as the finite volume method and the finite element method. The latter is mainly used in elliptic and parabolic problems (i.e. for problems without particular space directions dictating by equation and thereby allowing use of symmetric basis functions to expand solution). The finite volume method is similar to the finite differences method, but based on integral form of equations and therefore being perfectly suitable for problems with discontinuities, is designed for hyperbolic problems but generally having as disadvantage inability to give high-order approximation on unstructured grid.

Therefore we would like to find a wise mixture of the two methods mentioned and this leads us to the so-called Discontinuous Galerkin (DG) method.

We intend to introduce the DG method by considering homogeneous one-dimensional advection equation:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \tag{36}$$

with linear flux $f(u) = cu$.

We look for a solution to this equation on an spatial interval $\Omega = [0,\, L]$ performing partitioning of the whole interval into non-overlapping elements $\Omega = \bigcup_{k=1}^{K} D^k$ as well as discretization within an each element $D^k = [x_1^k,\, x_{N_p}^k]$. Note that $x_1^1 = 0,\ x_1^k = x_{N_p}^{k-1},\ x_{N_p}^k = x_1^{k+1},\ x_{N_p}^K = L$ where number of elements is K and number of grid points within one element is N_p and size of an element is $h^k = x_{N_p}^k - x_1^k$.

Illustration of this partitioning is given on Fig. 4.

In the DG method we follow the idea of the finite element method, but we search for local (not global as in the case of FEM) approximation of solution $u_h(x,t)$ in D^k as an expansion on some basis of functions $\left\{\psi_n^k(x)\right\}_{n=1}^{N_p}$ that we assume here to be chosen from the space

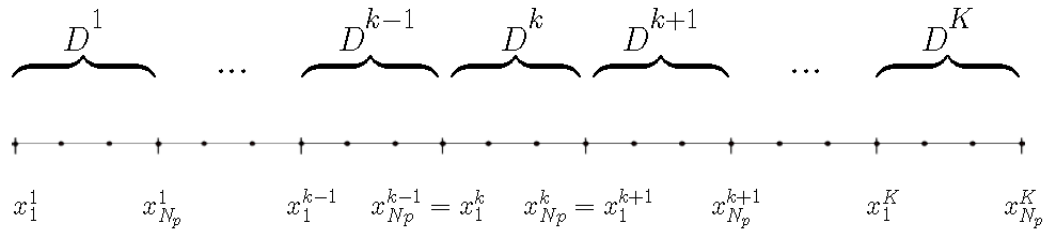

Fig. 4: Domain partitioning for the DG method

$C^\infty(D^k)$:

$$u_h^k(x,t) = \sum_{n=1}^{N_p} \hat{u}_n^k(t)\psi_n^k(x). \tag{37}$$

Then the global solution can be approximated as:

$$u(x,t) \approx u_h(x,t) = \bigoplus_{k=1}^{K} u_h^k(x,t).$$

Since we replace original infinite dimensional space with finite dimensional approximation space that is spanned by $\left\{\psi_n^k(x)\right\}_{n=1}^{N_p}$, the local approximation of solution $u_h^k(x,t)$ does not exactly satisfy the original equation (36), and this yields notion of the local residual:

$$R_h^k(x,t) = \frac{\partial u_h^k}{\partial t} + \frac{\partial f(u_h^k)}{\partial x}. \tag{38}$$

We want this local residual to be orthogonal to a test function from the space that, according to the Galerkin approach, we choose to be the same as the approximation space defined above. Due to independency of basis functions, it results in orthogonality of the residual (38) to all the functions $\left\{\psi_n^k(x)\right\}_{n=1}^{N_p}$:

$$\int_{D^k} R_h^k(x,t)\psi_n^k(x)dx = 0, \tag{39}$$

for $n = 1, \ldots, N_p$.

After plugging (38) into (39) we can perform integration by parts (since, as we assumed, our basis functions $\left\{\psi_n^k(x)\right\}_{n=1}^{N_p}$ are smooth on D^k):

$$\int_{D^k} \left(\frac{\partial u_h^k}{\partial t} \psi_n^k - c u_h^k \frac{\partial \psi_n^k}{\partial x} \right) dx = - \left[c u_h^k \psi_n^k \right] \Big|_{x_1^k}^{x_{N_p}^k}. \tag{40}$$

If we considered just an isolated element D^k, (40) would give us N_p equations allowing to determine the expansion coefficients $\hat{u}_n^k(t)$ (for $n = 1, \ldots, N_p$ and fixed k) of the local solution (43). However, due to locality of definition of our approximation space, we have discontinuities of the solution $u_h(x,t)$ at every interface between elements, and it gives rise to a question regarding which value of u_h^k to take at each element boundary. Therefore, in general we can simply rewrite (43):

$$\int_{D^k} \left(\frac{\partial u_h^k}{\partial t} \psi_n^k - c u_h^k \frac{\partial \psi_n^k}{\partial x} \right) dx = - \left[f^\star \psi_n^k \right] \Big|_{x_1^k}^{x_{N_p}^k}, \tag{41}$$

introducing the "numerical flux" $f^\star = (cu)^\star = cu_h^\star$ as a smart combination of flux values on the common boundary of every adjacent elements to approximate the real flux $f = cu$ through this boundary. For instance, on the right boundary of D^k the numerical flux is some function of $u_h^k(x_{N_p}^k)$ and $u_h^{k+1}(x_1^{k+1})$: $f^\star|_{x_{N_p}^k} = f^\star(u_h^k, u_h^{k+1})$ that must be chosen in a way not to cause instability of the whole method (study of stability will be considered few paragraphs later) and obviously be consistent (that is to satisfy $f^\star(u_h^k, u_h^k) = cu_h^k$ and $f^\star(u_h^{k+1}, u_h^{k+1}) = cu_h^{k+1}$).

Once the numerical flux is chosen, we can use (41), that is referred as weak formulation, to obtain all the expansion coefficients $\hat{u}_n^k(t)$ for all elements and thereby recover globally approximated solution $u_h(x,t)$.

Having introduced the numerical flux, we can perform integration by parts again and hence transform (41) back to the original form:

$$\int_{D^k} R_h^k(x,t)\psi_n^k(x)dx = \left[\left(cu_h^k - f^\star\right)\psi_n^k\right]\Big|_{x_1^k}^{x_{N_p}^k}, \tag{42}$$

which is called strong formulation and it gives a way to pose the problem with basis functions that are non-smooth or even discontinuous inside an element.

The question that still remains is how exactly to choose the numerical flux. Since the crucial property of a numerical method is stability, we will be looking for the simplest, linear, numerical flux in a way for the method to be stable.

We are going to use the energy method for stability analysis. In order to do that, it is convenient to choose the Lagrange polynomials as basis functions (that is so-called nodal approach). Then for local solution approximation in an element D^k we have:

$$u_h^k(x,t) = \sum_{j=1}^{N_p} u_h^k(x_j,t) l_j^k(x), \tag{43}$$

where $l_i^k(x) = \prod_{\substack{j=1 \\ (j\neq i)}}^{N_p} \frac{x - x_j^k}{x_i^k - x_j^k}$ is the Lagrange interpolation polynomial.

The strong formulation (44) reads:

$$\int_{D^k} \left(\frac{\partial u_h^k}{\partial t} + \frac{\partial f(u_h^k)}{\partial x}\right) l_i^k(x)dx = \left[\left(cu_h^k - f^\star\right) l_i^k(x)\right]\Big|_{x_1^k}^{x_{N_p}^k}, \tag{44}$$

for $i = 1, \ldots, N_p$.

Plugging (43) into the left-hand side of (44), we arrive at:

$$\sum_{j=1}^{N_p} \frac{du_h^k(x_j,t)}{dt} \underbrace{\int_{D^k} l_i^k(x) l_j^k(x)dx}_{\equiv M_{ij}^k} + \sum_{j=1}^{N_p} cu_h^k(x_j,t) \underbrace{\int_{D^k} \frac{dl_j^k(x)}{dx} l_i^k(x)dx}_{\equiv S_{ij}^k} = \left[\left(cu_h^k - f^\star\right) l_i^k(x)\right]\Big|_{x_1^k}^{x_{N_p}^k}.$$

This can be written in the vector form:

$$M^k \frac{d}{dt}\mathbf{u_h^k} + S^k(c\mathbf{u_h^k}) = \left[\left(cu_h^k - f^\star\right)\mathbf{l^k}\right]|_{x_1^k}^{x_{N_p}^k}, \tag{45}$$

where $\mathbf{u_h^k} = \left(u_h^k(x_1,t),\, \ldots,\, u_h^k(x_{N_P},t)\right)^T$, $\mathbf{l^k} = \left(l_1^k(x),\, \ldots,\, l_{N_p}^k(x)\right)^T$ and M^k, S^k introduced above are local mass and stiffness matrices, respectively.

Multiplying (45) by $\left(\mathbf{u_h^k}\right)^T$, we have the following:

- the first term yields:

$$\sum_{j=1}^{N_p} u_h^k(x_j,t) \sum_{i=1}^{N_p} \frac{du_h^k(x_i,t)}{dt} \int_{D^k} l_i^k(x) l_j^k(x) dx =$$
$$= \int_{D^k} \underbrace{\sum_{j=1}^{N_p} u_h^k(x_j,t) l_j^k(x)}_{=u_h^k(x,t)} \underbrace{\sum_{i=1}^{N_p} \frac{du_h^k(x_i,t)}{dt} l_i^k(x)}_{=\frac{\partial u_h^k(x,t)}{\partial t}} dx = \frac{1}{2}\frac{d}{dt}\left\|u_h^k\right\|_{D^k}^2$$

- in a similar fashion, the second term gives:

$$c\sum_{j=1}^{N_p} u_h^k(x_j,t) \sum_{i=1}^{N_p} u_h^k(x_i,t) \int_{D^k} \frac{dl_i^k(x)}{dx} l_j^k(x) dx =$$
$$= c\int_{D^k} \underbrace{\sum_{j=1}^{N_p} u_h^k(x_j,t) l_j^k(x)}_{=u_h^k(x,t)} \underbrace{\sum_{i=1}^{N_p} u_h^k(x_i,t) \frac{dl_i^k(x)}{dx}}_{=\frac{\partial u_h^k(x,t)}{\partial x}} dx = \frac{c}{2}(u_h^k)^2|_{x_1^k}^{x_{N_p}^k}.$$

Developing the right hand side term we take advantage of choosing Lagrange polynomials basis by utilizing the fact that $l_i^k(x_j) = \delta_{ij}$ (δ_{ij} is the Kronecker symbol):

$$\left(\mathbf{u_h^k}\right)^T \left[\left(cu_h^k - f^\star\right)\mathbf{l^k}\right]|_{x_1^k}^{x_{N_p}^k} = \left(\mathbf{u_h^k}\right)^T \left[\left(cu_h^k - f^\star\right)\left(l_1^k(x),\, 0,\, \ldots,\, 0,\, l_{N_p}^k(x)\right)^T\right]|_{x_1^k}^{x_{N_p}^k} =$$
$$= \left[\left(cu_h^k - f^\star\right)u_h^k\right]|_{x_1^k}^{x_{N_p}^k}.$$

Finally, we put everything together and obtain:

$$\frac{d}{dt}\left\|u_h^k\right\|_{D^k}^2 = 2\left[\left(cu_h^k - f^\star\right)u_h^k\right]\Big|_{x_1^k}^{x_{N_p}^k} - c(u_h^k)^2\Big|_{x_1^k}^{x_{N_p}^k} = \left[c(u_h^k)^2 - 2f^\star u_h^k\right]\Big|_{x_1^k}^{x_{N_p}^k}. \tag{46}$$

For stability one wants to have:

$$\frac{d}{dt}\left\|u_h\right\|_\Omega^2 = \sum_{k=1}^{K}\frac{d}{dt}\left\|u_h^k\right\|_{D^k}^2 \leq 0, \tag{47}$$

providing the appropriate solution is not growing in time, that is to say:

$$\frac{d}{dt}\left\|u\right\|_\Omega^2 = -c\left(u^2(L,t) - u^2(0,t)\right) \leq 0, \tag{48}$$

which follows from the integration by parts of the original equation (36) multiplied by $u(x,t)$.

Summing up (46) over all elements, we end up with the same difference of values at the boundaries of the domain Ω as in (48) simply by choosing the appropriate value of the numerical flux at the exterior plus contribution of jumps at every interface between elements related with solution discontinuities there. Since we want that the total contribution of those jumps do not make expression (47) positive, it is enough to impose condition of non-positive contribution of a jump at each interface:

$$c\left((u_h(x^-,t))^2 - (u_h(x^+,t))^2\right) - 2f^\star\left(u_h(x^-,t) - u_h(x^+,t)\right) \leq 0,$$

that is:

$$\left(u^- - u^+\right)\left(c\left(u^- + u^+\right) - 2f^\star\right) \leq 0, \tag{49}$$

where for the sake of brevity we use notation $x^- = x_{N_p}^k$, $x^+ = x_1^{k+1}$, $u^+ = u_h(x^+,t)$, $u^- = u_h(x^-,t)$, implying validity of this condition at all interfaces, i.e. for $k = 1, \ldots, N_p - 1$.

As we agreed we consider linear numerical flux as the simplest form, therefore we look for the appropriate numerical flux as a general linear combination of $u_h(x^+)$ and $u_h(x^-)$ that is the most convenient to write in the form:

$$f^\star = \frac{c}{2}\left(\beta_1(u^- - u^+) + \beta_2(u^- + u^+)\right).$$

Inserting this into the left-hand side of (49), we come to:

$$c(u^- - u^+)\left[(u^- + u^+) - \beta_1(u^- - u^+) - \beta_2(u^- + u^+)\right] \leq 0.$$

To ensure negativeness of this expression regardless particular values of u^+ and u^-, we want to have $[\ldots] = -\beta\frac{|c|}{c}(u^- - u^+)$ providing β is an arbitrary non-negative constant. This restriction leads us straightly to:

$$\beta_1 = \beta\frac{|c|}{c},$$

$$\beta_2 = 1,$$

and therefore the general linear numerical flux is:

$$f^\star = \frac{c}{2}(u^+ + u^-) + \beta\frac{|c|}{2}(u^- - u^+), \quad \beta \geq 0.$$

Notice that in case $\beta = 0$ we have the central numerical flux (and this corresponds to zero contribution from all internal boundaries):

$$f^\star = \frac{c}{2}(u^+ + u^-), \tag{50}$$

whereas setting $\beta = 1$ leads to the purely upwind numerical flux:

$$f^\star = \begin{cases} cu^-, & c > 0, \\ cu^+, & c < 0. \end{cases} \tag{51}$$

Therefore we can expect consistency of the method for all "intermediate" choice of the numerical flux, that is for:

$$f^\star = \frac{c}{2}(u^+ + u^-) + \beta\frac{|c|}{2}(u^- - u^+), \quad 0 \leq \beta \leq 1. \tag{52}$$

However, in our study we will focus just on the central (50) and the purely upwind (51) numerical fluxes.

Having chosen the basis functions for approximation space and the numerical flux, one can get back to (41) or (42) and eventually form the space discretization matrix.

Before we proceed with considering particular problem tackled by the DG method, error estimate needs to be briefly mentioned.

Obviously, increasing number of elements K (that is, refining grid, reducing the size of an element $h = L/K$) and number of points N_p (which results in increasing interpolation order being $N_p - 1$) should cause accuracy gain of the method. Namely, according to [4], in general one has:

$$\|u - u_h\|_\Omega \leq Ch^{N_p - 1/2}. \tag{53}$$

However, this estimates just spatial error and "constant" C, in fact, is time-dependent (that results in at least linear error growth in time).

Demonstration of the Discontinuous Galerkin method

Let us show how the DG method works on practice applying it to a simple "toy problem" (providing $c > 0$):

$$\begin{cases} \dfrac{\partial u}{\partial t} + c\dfrac{\partial u}{\partial x} = 0, \quad 0 < x < L,\, t > 0, \\ u(x,0) = \sin x, \\ u(0,t) = -\sin(ct) \equiv a(t), \end{cases}$$

which has the following exact solution:

$$u(x,t) = \sin(x - ct).$$

As it was discussed, we are looking for the approximate solution on each element $D^k = [x_1^k,\, x_{N_p}^k]$ $(k = 1, \ldots, K)$ in the form:

$$u_h^k(x,t) = \sum_{n=1}^{N_p} \hat{u}_n^k(t)\psi_n^k(x).$$

Plugging this into the weak formulation (41) gives:

$$\sum_{j=1}^{N_p} \frac{d\hat{u}_j^k(t)}{dt} \underbrace{\int_{D^k} \psi_i^k(x)\psi_j^k(x)dx}_{\equiv M_{ij}^k} - \sum_{j=1}^{N_p} c\hat{u}_j^k(t) \underbrace{\int_{D^k} \frac{d\psi_i^k(x)}{dx}\psi_j^k(x)dx}_{\equiv \left(S_{ij}^k\right)^T} = [\underbrace{-c\,(u)^\star}_{=f^\star}\psi_i^k(x)]\Big|_{x_1^k}^{x_{N_p}^k}. \quad (54)$$

Here, in contrast with nodal approach that we illustrated during study of stability, we will use modal polynomial approach. And natural way to do it seems to be choosing the set of functions $\{x^n\}_{n=0}^{N_p-1}$ as basis. However, since to go on with solving problem after applying the DG method, time derivatives need to be explicitly expressed from (54) that requires inverting mass matrices. At this point one can notice the fact that $\int x^i x^j dx \sim \dfrac{1}{i+j-1}$ which may

result in ill-conditioning (since for high-order interpolation the multiplier $\frac{1}{i+j-1}$ is close to zero) of mass matrices and therefore further loss of accuracy.

One way to proceed is to make inverting mass matrices M^k as simple as possible, and in order to do that, we choose the orthonormal Legendre polynomials $\{P_{n-1}\}_{n=1}^{N_p}$ on $[-1,\,1]$ as basis functions and do one-to-one mapping:

$$\psi_n^k(x) = P_{n-1}(\underbrace{r^k}_{\equiv r}), \quad n = 1,\,\ldots,\,N_p, \tag{55}$$

$$[x_1^k,\,x_{N_p}^k] \to [-1,\,1]: \quad r^k(x) = \frac{1}{h^k}(2x - x_1^k - x_{N_p}^k), \tag{56}$$

where the orthonormal Legendre polynomials can be explicitly computed using Rodrigues' formula:

$$P_n(r) = \frac{1}{n!2^n}\sqrt{\frac{2n+1}{2}}\frac{d^n}{dr^n}(r^2-1)^n,$$

or recurrent formula that they obey

$$rP_n(r) = a_n P_{n-1}(r) + a_{n+1}P_{n+1}(r),$$

starting with $P_0(r) = \frac{1}{\sqrt{2}}$, $P_1(r) = \sqrt{\frac{3}{2}}r$ and the notation $a_n = \frac{n}{\sqrt{(2n+1)(2n-1)}}$.

Then, performing the change of variables $x = \frac{1}{2}(x_1^k + x_{N_p}^k) + \frac{r}{2}h^k$ in integrals of (54), we have:

$$M_{ij}^k = \int_{x_1^k}^{x_{N_p}^k} \psi_i^k(x)\psi_j^k(x)dx = \frac{h^k}{2}\int_{-1}^{1} P_{i-1}(r)P_{j-1}(r)dr = \frac{h^k}{2}\delta_{ij},$$

$$S_{ij}^k = \int_{x_1^k}^{x_{N_p}^k} \psi_i^k(x)\frac{d\psi_j^k(x)}{dx}dx = \int_{-1}^{1} P_{i-1}(r)\frac{dP_{j-1}(r)}{dr}dr = S_{ij}.$$

Note that for uniform partitioning (i.e. $h^k = h$) mass matrices are simply assembled into the identity matrix multiplied by h and stiffness matrices are independent of an element number k.

Taking this into account, (54) transforms as follows:

$$\frac{h^k}{2}\frac{d\hat{u}_i^k(t)}{dt} - \sum_{j=1}^{N_p} c\hat{u}_j^k(t)\,(S_{ij})^T = [-f^\star\psi_i^k(x)]|_{x_1^k}^{x_{N_p}^k}. \tag{57}$$

The numerical flux on the left boundary of the domain is chosen to be equal $f^\star|_{x_1^1} = ca(t)$ (purely upwind, due to the boundary condition) and on the right boundary flux is purely outflow $f^\star|_{x_{N_p}^K} = cu_h^K(x_{N_p}^K)$ (no boundary conditions can be imposed).

Between the elements we, first, choose purely **central numerical fluxes**.

This gives (where we still will use ψ notation for a while instead of P):

$$\begin{aligned}\frac{d\hat{u}_i^1(t)}{dt} &= -\frac{c}{h^1}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji} + \psi_i^1(x_{N_p}^1)\psi_j^1(x_{N_p}^1)\Big)\hat{u}_j^1(t) \quad + \\ &+ \quad \psi_i^1(x_{N_p}^1)\psi_j^2(x_{N_p}^1)\hat{u}_j^2(t)\Big] + \frac{2c}{h^1}a(t)\psi_i^1(x_1^1),\end{aligned}$$

$$\begin{aligned}\frac{d\hat{u}_i^k(t)}{dt} &= -\frac{c}{h^k}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji} + \psi_i^k(x_{N_p}^k)\psi_j^k(x_{N_p}^k) - \psi_i^k(x_1^k)\psi_j^k(x_1^k)\Big)\hat{u}_j^k(t) \quad + \\ &+ \quad \psi_i^k(x_{N_p}^k)\psi_j^{k+1}(x_{N_p}^k)\hat{u}_j^{k+1}(t) - \psi_i^k(x_1^k)\psi_j^{k-1}(x_1^k)\hat{u}_j^{k-1}(t)\Big], \quad 2 \le k \le K-1,\end{aligned}$$

$$\begin{aligned}\frac{d\hat{u}_i^K(t)}{dt} &= -\frac{c}{h^K}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji} + 2\psi_i^K(x_{N_p}^K)\psi_j^K(x_{N_p}^K) - \psi_i^K(x_1^K)\psi_j^K(x_1^K)\Big)\hat{u}_j^K(t) \quad - \\ &- \quad \psi_i^K(x_1^K)\psi_j^{K-1}(x_1^K)\hat{u}_j^{K-1}(t)\Big].\end{aligned}$$

Taking into account that due to (55)-(56) we have $\psi_i^k(x_1^k) = P_{i-1}(-1)$ and $\psi_i^k(x_{N_p}^k) = P_{i-1}(1)$, we end up with:

$$\begin{aligned}\frac{d\hat{u}_i^1(t)}{dt} &= -\frac{c}{h^1}\sum_{j=1}^{N_p}\left[\left(-2S_{ji} + P_{i-1}(1)P_{j-1}(1)\right)\hat{u}_j^1(t) \quad + \right. \\ &+ \quad \left. P_{i-1}(1)P_{j-1}(-1)\hat{u}_j^2(t)\right] + \frac{2c}{h^1}a(t)P_{i-1}(-1),\end{aligned}$$

$$\begin{aligned}\frac{d\hat{u}_i^k(t)}{dt} &= -\frac{c}{h^k}\sum_{j=1}^{N_p}\left[\left(-2S_{ji} + P_{i-1}(1)P_{j-1}(1) - P_{i-1}(-1)P_{j-1}(-1)\right)\hat{u}_j^k(t) \quad + \right. \\ &+ \quad \left. P_{i-1}(1)P_{j-1}(-1)\hat{u}_j^{k+1}(t) - P_{i-1}(-1)P_{j-1}(1)\hat{u}_j^{k-1}(t)\right], \quad 2 \le k \le K-1,\end{aligned}$$

$$\begin{aligned}\frac{d\hat{u}_i^K(t)}{dt} &= -\frac{c}{h^K}\sum_{j=1}^{N_p}\left[\left(-2S_{ji} + 2P_{i-1}(1)P_{j-1}(1) - P_{i-1}(-1)P_{j-1}(-1)\right)\hat{u}_j^K(t) \quad - \right. \\ &- \quad \left. P_{i-1}(-1)P_{j-1}(1)\hat{u}_j^{K-1}(t)\right].\end{aligned}$$

Now, after mapping to linear index structure $(k,i) \to (k-1)N_p + i$, space discretization matrix A can be assembled and the formulas above can be written in the vector form:

$$\frac{d\hat{\mathbf{u}}(t)}{dt} = cA\hat{\mathbf{u}}(t) + \mathbf{f}(t),$$

where dimension of the vectors $\hat{\mathbf{u}}(t)$ and $\mathbf{f}(t)$ is KN_p (though $\mathbf{f}(t)$ have only first N_p components not equal to zero that is result of the boundary condition) and the matrix A has the sparsity pattern given on Fig. 5 (for $K = 20,\ N_p = 2$).

Now, we consider purely **upwind numerical fluxes** at all internal boundaries between elements.

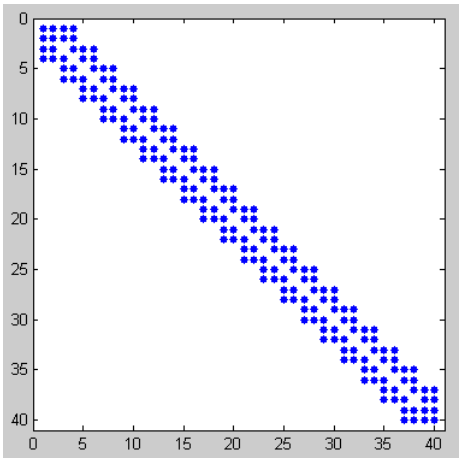

Fig. 5: Discretization matrix sparsity pattern for the DG method using central numerical fluxes

This yields:

$$\frac{d\hat{u}_i^1(t)}{dt} = -\frac{2c}{h^1}\sum_{j=1}^{N_p}\left(-S_{ji}+\psi_i^1(x_{N_p}^1)\psi_j^1(x_{N_p}^1)\right)\hat{u}_j^1(t)+\frac{2c}{h^1}a(t)\psi_i^1(x_1^1),$$

$$\frac{d\hat{u}_i^k(t)}{dt} = -\frac{2c}{h^k}\sum_{j=1}^{N_p}\left[\left(-S_{ji}+\psi_i^k(x_{N_p}^k)\psi_j^k(x_{N_p}^k)\right)\hat{u}_j^k(t)-\psi_i^k(x_1^k)\psi_j^{k-1}(x_1^k)\hat{u}_j^{k-1}(t)\right],$$

$$2\leq k\leq K-1,$$

$$\frac{d\hat{u}_i^K(t)}{dt} = -\frac{2c}{h^K}\sum_{j=1}^{N_p}\left[\left(-S_{ji}+\psi_i^K(x_{N_p}^K)\psi_j^K(x_{N_p}^K)\right)\hat{u}_j^K(t)-\psi_i^K(x_1^K)\psi_j^{K-1}(x_1^K)\hat{u}_j^{K-1}(t)\right].$$

The same can be written in terms of the normalized Legendre polynomials:

$$\frac{d\hat{u}_i^1(t)}{dt} = -\frac{2c}{h^1}\sum_{j=1}^{N_p}\left(-S_{ji} + P_{i-1}(1)P_{j-1}(1)\right)\hat{u}_j^1(t) + \frac{2c}{h^1}a(t)P_{i-1}(-1),$$

$$\frac{d\hat{u}_i^k(t)}{dt} = -\frac{2c}{h^k}\sum_{j=1}^{N_p}\left[\left(-S_{ji} + P_{i-1}(1)P_{j-1}(1)\right)\hat{u}_j^k(t) - P_{i-1}(-1)P_{j-1}(1)\hat{u}_j^{k-1}(t)\right],$$

$$2 \le k \le K-1,$$

$$\frac{d\hat{u}_i^K(t)}{dt} = -\frac{2c}{h^K}\sum_{j=1}^{N_p}\left[\left(-S_{ji} + P_{i-1}(1)P_{j-1}(1)\right)\hat{u}_j^K(t) - P_{i-1}(-1)P_{j-1}(1)\hat{u}_j^{K-1}(t)\right].$$

In the vector form:

$$\frac{d\hat{\mathbf{u}}(t)}{dt} = cA\hat{\mathbf{u}}(t) + \mathbf{f}(t),$$

where the space discretization matrix has the sparsity pattern given on Fig. 6 (for $K = 20,\ N_p = 2$).

Now it rests to do integration in time. Since the point of this demonstration is to illustrate application of the DG method, but not gaining high accuracy in time, to avoid possible stability issues we simply use the Backward Euler scheme for time integration. In both cases, for purely central and purely upwind fluxes, this gives:

$$\frac{\hat{\mathbf{u}}^{n+1} - \hat{\mathbf{u}}^n}{k} = cA\hat{\mathbf{u}}^{n+1} + \mathbf{f}^{n+1} \qquad \Rightarrow \qquad \hat{\mathbf{u}}^{n+1} = \underbrace{(I - kcA)^{-1}}_{\equiv B}\hat{\mathbf{u}}^n + kB\mathbf{f}^{n+1},$$

where $\mathbf{f}^{n+1} = \dfrac{2c}{h^1}a(t_{n+1})\left(P_0(-1),\ \ldots,\ P_{N_p-1}(-1),\ 0,\ 0,\ \ldots,\ 0\right)^T$ and the starting state $\hat{\mathbf{u}}^0$ is determining from the expansion of the initial condition inside each element. And vice

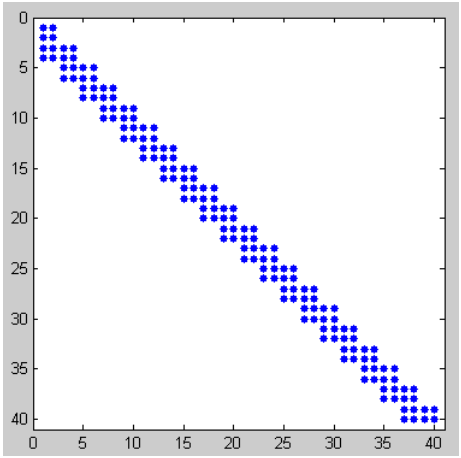

Fig. 6: Discretization matrix sparsity pattern for the DG method using upwind numerical fluxes

versa, once expansion coefficient vector found at desired time, the solution immediately follows by expansion with these coefficients inside each element.

On Fig. 7 and 8 the results are given for the both types of numerical fluxes considered and different interpolation order and number of elements. The following constant parameters were used: length of physical (spatial) domain $L = 10$, velocity $c = 0.05$.

As one can notice, for higher interpolation order ($N_p = 4$) there is no visible difference between use of purely upwind or purely central numerical fluxes (however, as we will see in the next section, a choice of the numerical flux may strongly affect stability condition). For linear interpolation ($N_p = 2$) this difference becomes more and more obvious with growth of time as error increases, nevertheless, this is good from instructive point of view to reveal and illustrate the discontinuous nature of the method.

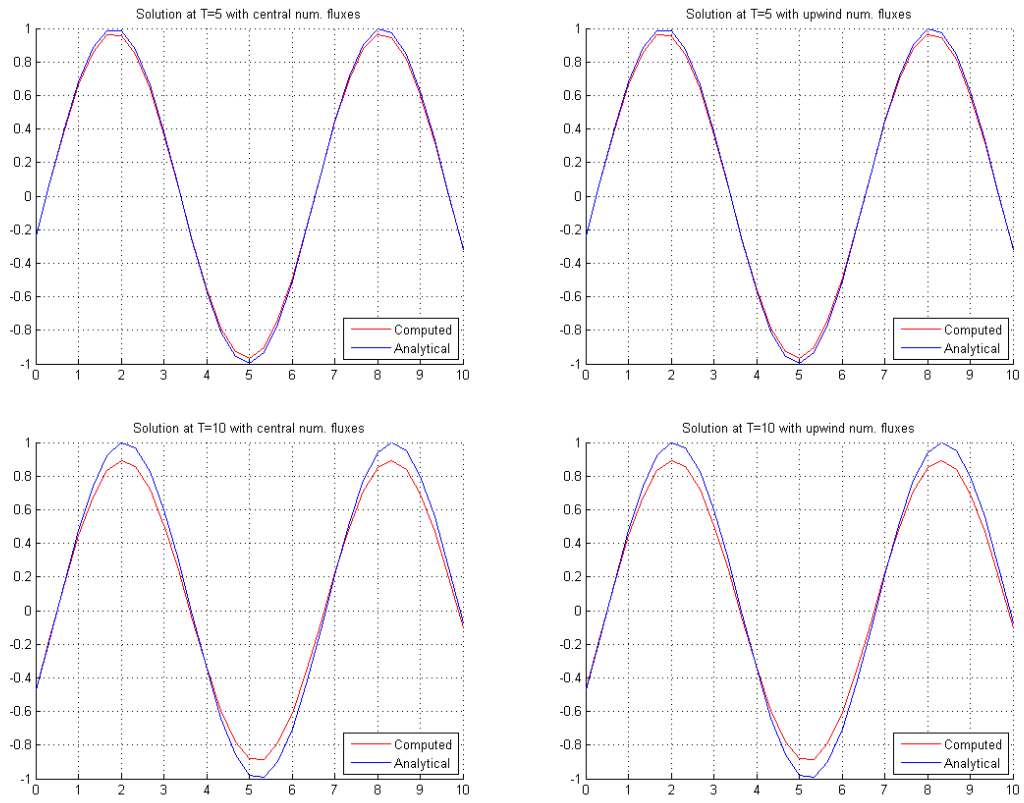

Fig. 7: Solution of advection equation using the DG method with central numerical fluxes. Number of elements: $K = 10$, points inside an element: $N_p = 4$

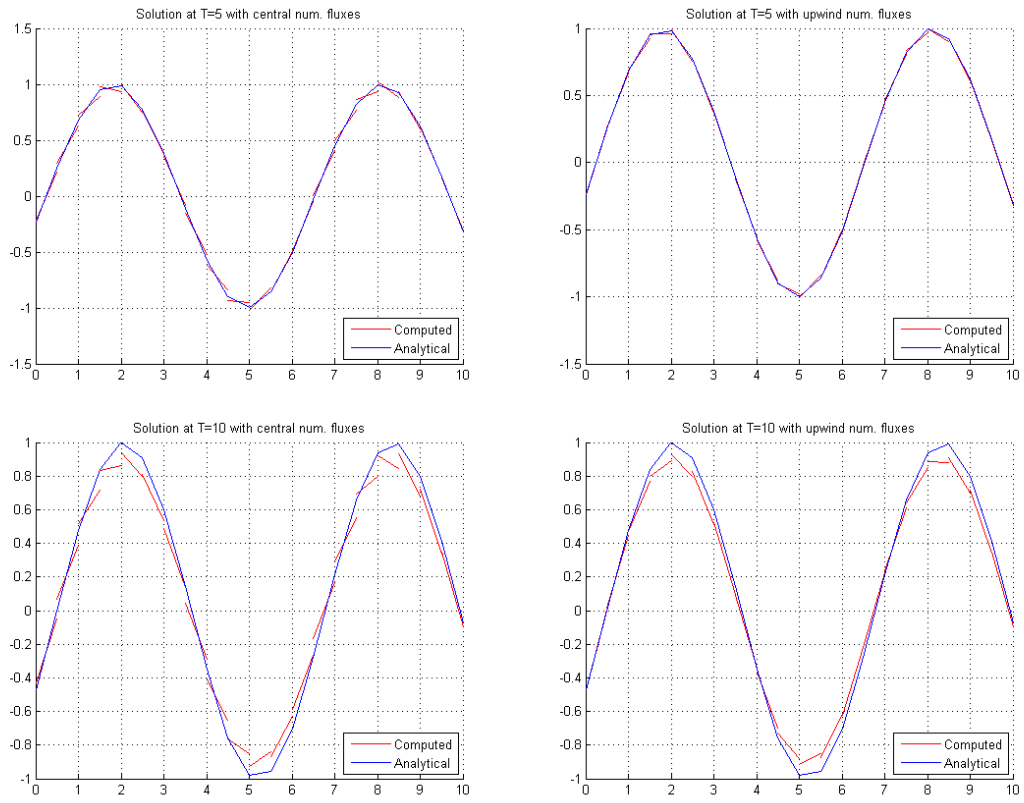

Fig. 8: Solution of advection equation using the DG method with central numerical fluxes. Number of elements: $K = 20$, points inside an element: $N_p = 2$

4 Application to the equations of electromagnetics

Since the aim of this work is the numerical solution of electromagnetic wave propagation problem, we start by introducing Maxwell's equations.

The famous Maxwell's set of equations (written in Electrostatic CGS units system in order to have symmetry between electrical and magnetic fields which in this system have the same dimension) reads:

$$\begin{cases} \nabla \cdot \mathbf{E} = 4\pi\rho, \\ \nabla \cdot \mathbf{B} = 0, \\ \nabla \times \mathbf{E} = -\dfrac{1}{c}\dfrac{\partial \mathbf{B}}{\partial t}, \\ \nabla \times \mathbf{B} = \dfrac{4\pi}{c}\mathbf{J} + \dfrac{1}{c}\dfrac{\partial \mathbf{E}}{\partial t}, \end{cases} \tag{58}$$

where $\mathbf{E}$, $\mathbf{B}$ are electrical and magnetic fields correspondingly, $\mathbf{J}$ is vector of current density, c is the speed of light, ρ is the charge density.

This incorporates Gauss' laws for electrical (more precisely, Gauss-Coulomb law) and magnetic fields (the first two equations), Faraday's law of induction and Ampere's circuital law with Maxwell's correction (namely, displacement current, the last term in the right hand side of the fourth equation).

Here we also assume that dielectric permeability and magnetic susceptibility are both equal to unity $\epsilon = \mu = 1$ (that is, no polarization and magnetization effects occur), and moreover we will focus on the electromagnetic wave propagation problems in free space, therefore we have:

$$\begin{aligned} \rho &= 0, \\ \mathbf{J} &= \mathbf{0}. \end{aligned}$$

For the sake of simplicity we consider just the one dimensional case.

Let us set:

$$\begin{aligned} \mathbf{E} &= (0,\, 0,\, E_z(x,t)) \equiv (0,\, 0,\, E(x,t)), \\ \mathbf{B} &= (0,\, B_y(x,t),\, 0) \equiv (0,\, B(x,t),\, 0). \end{aligned}$$

Then:

$$\nabla \times \mathbf{E} = \det \begin{pmatrix} \mathbf{e_x} & \mathbf{e_y} & \mathbf{e_z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ 0 & 0 & E \end{pmatrix} = -\frac{\partial E}{\partial x}\mathbf{e_y},$$

$$\nabla \times \mathbf{B} = \det \begin{pmatrix} \mathbf{e_x} & \mathbf{e_y} & \mathbf{e_z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ 0 & B & 0 \end{pmatrix} = \frac{\partial B}{\partial x}\mathbf{e_z}.$$

So the last two equations in (58) governing dynamics (the first two are satisfied automatically) take the scalar form:

$$\begin{cases} \dfrac{\partial E}{\partial t} = -c\dfrac{\partial B}{\partial x}, \\ \dfrac{\partial B}{\partial t} = -c\dfrac{\partial E}{\partial x}, \end{cases} \tag{59}$$

that (after substitution $u = B$, $v = -E$) corresponds exactly to (15), the example considered before. However, here we focus on use of high-order methods.

First, performing space discretization and denoting in general the corresponding discretization operators as A_E, A_B (that particularly might come from DG method and which may not be the same for E and B due to the different boundary conditions), we arrive at:

$$\begin{cases} \dfrac{d\mathbf{E}}{dt} = cA_B\mathbf{B}, \\ \dfrac{d\mathbf{B}}{dt} = cA_E\mathbf{E}, \end{cases} \tag{60}$$

where we write $\mathbf{E}$, $\mathbf{B}$ as vectors implying the vectors with the spatial values as the components.

As we showed in the example in the first part (in particular case of the full discretization in space and time), for the wave equation the Leap-Frog method can be effectively applied on a staggered grid. Let us consider this in details.

According to the previous publications ([2, 7]), general staggered Leap-Frog methods for time integration of the second and the fourth order of accuracy are further introduced.

Consider a system of ODEs:

$$\begin{cases} u' = f(t, v), \\ v' = g(t, u). \end{cases}$$

StaggeredLF2 reads:

$$\begin{cases} u^{n+1} = u^n + kf(t_{n+1/2}, v^{n+1/2}), \\ v^{n+3/2} = v^{n+1/2} + kg(t_{n+1}, u^{n+1}). \end{cases}$$

StaggeredLF4 is given by:

$$\begin{cases} u^{n+1} = u^n + \dfrac{22}{24}\alpha_1 + \dfrac{1}{24}\alpha_3 + \dfrac{1}{24}\alpha_5, \\ v^{n+3/2} = v^{n+1/2} + \dfrac{22}{24}\beta_1 + \dfrac{1}{24}\beta_3 + \dfrac{1}{24}\beta_5, \end{cases}$$

where:

$$\begin{cases} \alpha_1 = kf(t_{n+1/2}, v^{n+1/2}), \\ \alpha_2 = kg(t_n, u^n), \\ \alpha_3 = kf(t_{n-1/2}, v^{n+1/2} - \alpha_2), \\ \alpha_4 = kg(t_{n+1}, u^n + \alpha_1), \\ \alpha_5 = kf(t_{n+3/2}, v^{n+1/2} + \alpha_4), \end{cases}$$

and:

$$\begin{cases} \beta_1 = kg(t_{n+1}, u^{n+1}), \\ \beta_2 = kf(t_{n+1/2}, v^{n+1/2}), \\ \beta_3 = kg(t_n, u^{n+1} - \beta_2), \\ \beta_4 = kf(t_{n+3/2}, v^{n+1/2} + \beta_1), \\ \beta_5 = kg(t_{n+2}, u^{n+1} + \beta_4). \end{cases}$$

The StaggeredLF2 rule for our generally discretized in space problem (60) is as following:

$$\begin{cases} \dfrac{\mathbf{E}^{n+1} - \mathbf{E}^n}{k} = cA_B\mathbf{B}^{n+1/2}, \\ \dfrac{\mathbf{B}^{n+3/2} - \mathbf{B}^{n+1/2}}{k} = cA_E\mathbf{E}^{n+1}. \end{cases} \tag{61}$$

However, we will focus on the higher-order accuracy scheme StaggeredLF4.

Application of the StaggeredLF4 to our particular case yields:

$$\begin{cases} \dfrac{\mathbf{E}^{n+1} - \mathbf{E}^n}{k} = \left(A_B + \dfrac{1}{24}c^2k^2A_BA_EA_B\right)c\mathbf{B}^{n+1/2}, \\ \dfrac{\mathbf{B}^{n+3/2} - \mathbf{B}^{n+1/2}}{k} = \left(A_E + \dfrac{1}{24}c^2k^2A_EA_BA_E\right)c\mathbf{E}^{n+1}. \end{cases} \tag{62}$$

Expressing $\mathbf{E}^{n+1}$ from the first expression in (62) and plugging into the second one, we have:

$$\mathbf{E}^{n+1} = \mathbf{E}^n + kc\left(A_B + \frac{1}{24}k^2c^2A_BA_EA_B\right)\mathbf{B}^{n+1/2},$$

$$\begin{aligned}\mathbf{B}^{n+3/2} = \mathbf{B}^{n+1/2} + kc\left(A_E + \frac{1}{24}k^2c^2A_EA_BA_E\right)\mathbf{E}^{n+1} = \\ = kc\left(A_E + \frac{1}{24}k^2c^2A_EA_BA_E\right)\mathbf{E}^n + \\ + \left(I + k^2c^2\left(A_E + \frac{1}{24}k^2c^2A_EA_BA_E\right)\left(A_B + \frac{1}{24}k^2c^2A_BA_EA_B\right)\right)\mathbf{B}^{n+1/2}.\end{aligned}$$

These expressions can be written in the matrix form:

$$\begin{pmatrix}\mathbf{E}^{n+1}\\ \mathbf{B}^{n+3/2}\end{pmatrix} = \underbrace{\begin{pmatrix}I & S_1\\ S_2 & I + S_2S_1\end{pmatrix}}_{\equiv C}\begin{pmatrix}\mathbf{E}^n\\ \mathbf{B}^{n+1/2}\end{pmatrix}, \tag{63}$$

where we denote:

$$S_1 = kc\left(A_B + \frac{1}{24}k^2c^2A_BA_EA_B\right), \tag{64}$$

$$S_2 = kc\left(A_E + \frac{1}{24}k^2c^2A_EA_BA_E\right). \tag{65}$$

Now, given space discretization matrices A_E, A_B, the amplification matrix C can be computed with help of (64), (65) and (63) allows to perform explicit time integration, providing stability condition holds.

Particular problem: electromagnetic waves between two metallic plates

Now we are ready to apply all the techniques described above for finding high-order solution approximation, the DG method for spatial discretization and the StaggeredLF4 scheme for

integration in time.

Particular case that we focus on is one-dimensional wave problem of finding electromagnetic field between 2 plates of perfect conducting metall that implies homogeneous Dirichlet boundary conditions for electrical field:

$$E(0,t) = E(L,t) = 0. \tag{66}$$

Imposing this condition automatically determines behavior at the boundaries of magnetic field due to validity of Maxwell's equations close to boundary. Indeed, taking time derivatives of (66) and utilizing (59) we arrive at

$$B_x(0,t) = B_x(L,t) = 0. \tag{67}$$

As it was mentioned before when the wave equation problem was reduced to (16) and also discussed during StaggeredLeapFrog2 scheme demonstration (both in the first part of the current work), the conditions (67) are redundant from mathematical point of view, since they follow from the equations and thereby are satisfied automatically, but in our approach, when discretizations in space and time are sequential, it is important to write them down separately because they are necessary parts of space discretization operators A_E and A_B that we want to construct.

Therefore, the problem to solve is as follows:

$$\begin{cases} \dfrac{\partial E}{\partial t} = -c\dfrac{\partial B}{\partial x}, \quad 0 < x < L,\, t > 0, \\ \dfrac{\partial B}{\partial t} = -c\dfrac{\partial E}{\partial x}, \quad 0 < x < L,\, t > 0, \\ E(0,t) = E(L,t) = 0, \\ B_x(0,t) = B_x(L,t) = 0, \\ E(x,0) = \sin\left(\dfrac{\pi x}{L}\right), \\ B(x,0) = 0, \end{cases} \tag{68}$$

where initial conditions were chosen in order to simply the exact solution guess:

$$\begin{cases} E(x,t) = & \sin\left(\dfrac{\pi x}{L}\right)\cos\left(\dfrac{\pi ct}{L}\right), \\ B(x,t) = & -\cos\left(\dfrac{\pi x}{L}\right)\sin\left(\dfrac{\pi ct}{L}\right), \end{cases} \tag{69}$$

allowing to compare results with.

We start from discretization in space and we will follow exactly the same line as in the previous part, when the advection equation "toy problem" was considered.

According to the DG method, the approximated solutions are sought inside each element $D^k = [x_1^k,\, x_{N_p}^k]$ $(k = 1,\, \ldots,\, K)$:

$$E_h^k(x,t) = \sum_{n=1}^{N_p} \hat{E}_n^k(t)\psi_n^k(x),$$

$$B_h^k(x,t) = \sum_{n=1}^{N_p} \hat{B}_n^k(t)\psi_n^k(x).$$

The weak formulation follows:

$$\begin{cases} \sum_{j=1}^{N_p} \frac{d\hat{E}_j^k(t)}{dt} \underbrace{\int_{D^k} \psi_i^k(x)\psi_j^k(x)dx}_{\equiv M_{ij}^k} - \sum_{j=1}^{N_p} c\hat{B}_j^k(t) \underbrace{\int_{D^k} \frac{d\psi_i^k(x)}{dx}\psi_j^k(x)dx}_{\equiv \left(S_{ij}^k\right)^T} = [-\underbrace{c\,(B)^\star}_{\equiv f_B^\star}\psi_i^k(x)]\big|_{x_1^k}^{x_{N_p}^k}, \\ \sum_{j=1}^{N_p} \frac{d\hat{B}_j^k(t)}{dt} M_{ij}^k - \sum_{j=1}^{N_p} c\hat{E}_j^k(t) \left(S_{ij}^k\right)^T = [-\underbrace{c\,(E)^\star}_{\equiv f_E^\star}\psi_i^k(x)]\big|_{x_1^k}^{x_{N_p}^k}. \end{cases}$$

As in the demonstration of the DG method applied to the advection equation that was given before, we choose Legendre polynomials as basis functions and rewrite the weak formulation (we omit some details and explanations avoiding repetition of what has already been said in the previous part of the work):

$$\begin{cases} \frac{h^k}{2}\frac{d\hat{E}_i^k(t)}{dt} - \sum_{j=1}^{N_p} c\hat{B}_j^k(t) \left(S_{ij}\right)^T = [-f_B^\star\psi_i^k(x)]\big|_{x_1^k}^{x_{N_p}^k}, \\ \frac{h^k}{2}\frac{d\hat{B}_i^k(t)}{dt} - \sum_{j=1}^{N_p} c\hat{E}_j^k(t) \left(S_{ij}\right)^T = [-f_E^\star\psi_i^k(x)]\big|_{x_1^k}^{x_{N_p}^k}. \end{cases}$$

Now we come to the point where numerical fluxes on the boundaries of the domain have to be chosen in order to satisfy the boundary conditions.

For electrical field we have homogeneous Dirichlet boundary conditions whose approximation is straightforward: $f_E^\star|_{x_1^1} = f_E^\star|_{x_{N_p}^K} = 0$.

The condition $f_E^\star|_{x_1^1} = 0$ can be looked at as zeroing the central numerical flux between the left boundary of the leftmost element ($k = 1$) giving contribution $cE_h^1(x_1^1)$ and the right boundary of some "ghost" element to the left of it bringing value $-cE_h^1(x_1^1)$. Absolutely the same thing can be done with attaching artificial element to the rightmost element ($k = K$) and look at the condition $f_E^\star|_{x_{N_p}^K} = 0$ as the result of central numerical flux approximation between them. These boundary conditions are illustated on Fig. 9.

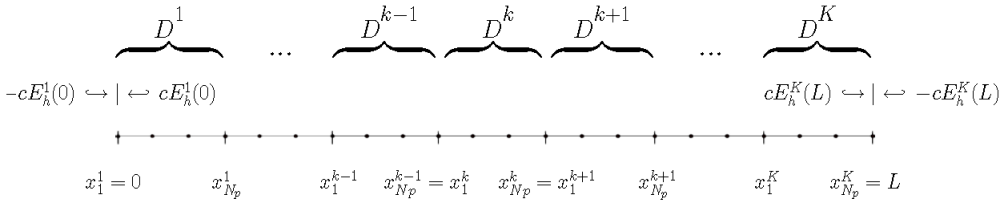

Fig. 9: Imposing Dirichlet boundary conditions in the DG method

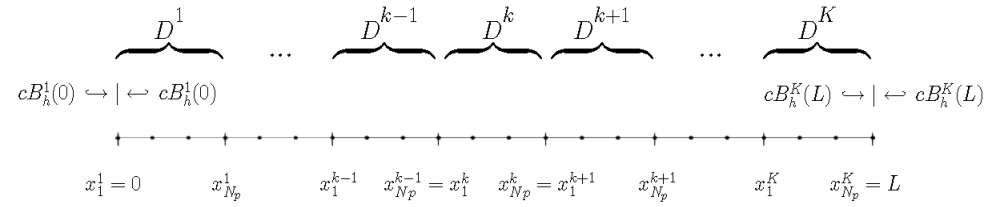

Fig. 10: Imposing Neumann boundary conditions in the DG method

In a similar fashion, homogeneous Neumann boundary conditions for the magnetic field can be treated. Again, employing "ghost element" principle we can consider condition $\frac{\partial B}{\partial x} = 0$, say, on the left boundary of the domain, as equality of $cB_h^1(x_1^1)$ to exactly the same value coming from the "ghost" element placed to the left of the considered one ($k = 1$), then using central numerical flux yields $f_B^\star|_{x_1^1} = \frac{1}{2}\left(cB_h^1(x_1^1) + cB_h^1(x_1^1)\right) = cB_h^1(\underbrace{x_1^1}_{=0})$. The same considerations can be applied to the right boundary of the domain, this leads to the analogous boundary condition $f_B^\star|_{x_{N_p}^K} = cB_h^K(\underbrace{x_{N_p}^K}_{=L})$.

Next, we consider internal boundaries between all the elements.

We firstly start with choosing the purely **central numerical fluxes** for this purpose.

The weak formulation for all the internal and the boundary elements gives:

$$
\begin{aligned}
\frac{d\hat{E}_i^1(t)}{dt} &= -\frac{c}{h^1}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji}+\psi_i^1(x_{N_p}^1)\psi_j^1(x_{N_p}^1)-2\psi_i^1(x_1^1)\psi_j^1(x_1^1)\Big)\hat{B}_j^1(t)+ \\
&\qquad +\psi_i^1(x_{N_p}^1)\psi_j^2(x_{N_p}^1)\hat{B}_j^2(t)\Big], \\
\frac{d\hat{B}_i^1(t)}{dt} &= -\frac{c}{h^1}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji}+\psi_i^1(x_{N_p}^1)\psi_j^1(x_{N_p}^1)\Big)\hat{E}_j^1(t)+\psi_i^1(x_{N_p}^1)\psi_j^2(x_{N_p}^1)\hat{E}_j^2(t)\Big], \\
\frac{d\hat{E}_i^k(t)}{dt} &= -\frac{c}{h^k}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji}+\psi_i^k(x_{N_p}^k)\psi_j^k(x_{N_p}^k)-\psi_i^k(x_1^k)\psi_j^k(x_1^k)\Big)\hat{B}_j^k(t) \quad + \\
&\quad + \quad \psi_i^k(x_{N_p}^k)\psi_j^{k+1}(x_{N_p}^k)\hat{B}_j^{k+1}(t)-\psi_i^k(x_1^k)\psi_j^{k-1}(x_1^k)\hat{B}_j^{k-1}(t)\Big], \\
&\qquad 2\le k\le K-1, \\
\frac{d\hat{B}_i^k(t)}{dt} &= -\frac{c}{h^k}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji}+\psi_i^k(x_{N_p}^k)\psi_j^k(x_{N_p}^k)-\psi_i^k(x_1^k)\psi_j^k(x_1^k)\Big)\hat{E}_j^k(t) \quad + \\
&\quad + \quad \psi_i^k(x_{N_p}^k)\psi_j^{k+1}(x_{N_p}^k)\hat{E}_j^{k+1}(t)-\psi_i^k(x_1^k)\psi_j^{k-1}(x_1^k)\hat{E}_j^{k-1}(t)\Big], \\
&\qquad 2\le k\le K-1, \\
\frac{d\hat{E}_i^K(t)}{dt} &= -\frac{c}{h^K}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji}+2\psi_i^K(x_{N_p}^K)\psi_j^K(x_{N_p}^K)-\psi_i^K(x_1^K)\psi_j^K(x_1^K)\Big)\hat{B}_j^K(t)- \\
&\qquad -\psi_i^K(x_1^K)\psi_j^{K-1}(x_1^K)\hat{B}_j^{K-1}(t)\Big], \\
\frac{d\hat{B}_i^K(t)}{dt} &= -\frac{c}{h^K}\sum_{j=1}^{N_p}\Big[\Big(-2S_{ji}-\psi_i^K(x_1^K)\psi_j^K(x_1^K)\Big)\hat{E}_j^K(t)-\psi_i^K(x_1^K)\psi_j^{K-1}(x_1^K)\hat{E}_j^{K-1}(t)\Big].
\end{aligned}
$$

The same can be written in more convenient form (so that discretization matrix elements are independent of an element index), in terms of the normalized Legendre polynomials:

$$
\begin{aligned}
\frac{d\hat{E}_i^1(t)}{dt} &= -\frac{c}{h^1}\sum_{j=1}^{N_p}\Big[\big(-2S_{ji}+P_{i-1}(1)P_{j-1}(1)-2P_{i-1}(-1)P_{j-1}(-1)\big)\hat{B}_j^1(t)+ \\
&\qquad +P_{i-1}(1)P_{j-1}(-1)\hat{B}_j^2(t)\Big], \\
\frac{d\hat{B}_i^1(t)}{dt} &= -\frac{c}{h^1}\sum_{j=1}^{N_p}\Big[\big(-2S_{ji}+P_{i-1}(1)P_{j-1}(1)\big)\hat{E}_j^1(t)+P_{i-1}(1)P_{j-1}(-1)\hat{E}_j^2(t)\Big],
\end{aligned}
$$

$$
\begin{aligned}
\frac{d\hat{E}_i^k(t)}{dt} &= -\frac{c}{h^k}\sum_{j=1}^{N_p}\Big[\left(-2S_{ji}+P_{i-1}(1)P_{j-1}(1)-P_{i-1}(-1)P_{j-1}(-1)\right)\hat{B}_j^k(t) \quad + \\
&\quad + \quad P_{i-1}(1)P_{j-1}(-1)\hat{B}_j^{k+1}(t)-P_{i-1}(-1)P_{j-1}(1)\hat{B}_j^{k-1}(t)\Big], \\
&\qquad\qquad 2\le k\le K-1, \\
\frac{d\hat{B}_i^k(t)}{dt} &= -\frac{c}{h^k}\sum_{j=1}^{N_p}\Big[\left(-2S_{ji}+P_{i-1}(1)P_{j-1}(1)-P_{i-1}(-1)P_{j-1}(-1)\right)\hat{E}_j^k(t) \quad + \\
&\quad + \quad P_{i-1}(1)P_{j-1}(-1)\hat{E}_j^{k+1}(t)-P_{i-1}(-1)P_{j-1}(1)\hat{E}_j^{k-1}(t)\Big], \\
&\qquad\qquad 2\le k\le K-1, \\
\frac{d\hat{E}_i^K(t)}{dt} &= -\frac{c}{h^K}\sum_{j=1}^{N_p}\Big[\left(-2S_{ji}+2P_{i-1}(1)P_{j-1}(1)-P_{i-1}(-1)P_{j-1}(-1)\right)\hat{B}_j^K(t)- \\
&\qquad\qquad - P_{i-1}(-1)P_{j-1}(1)\hat{B}_j^{K-1}(t)\Big], \\
\frac{d\hat{B}_i^K(t)}{dt} &= -\frac{c}{h^K}\sum_{j=1}^{N_p}\Big[\left(-2S_{ji}-P_{i-1}(-1)P_{j-1}(-1)\right)\hat{E}_j^K(t)\;-\;P_{i-1}(-1)P_{j-1}(1)\hat{E}_j^{K-1}(t)\Big].
\end{aligned}
$$

The same procedure can be done in case when the purely **upwind numerical fluxes** are used at all the internal boundaries.

The weak formulation yields:

$$
\begin{aligned}
\frac{d\hat{E}_i^1(t)}{dt} &= -\frac{2c}{h^1}\sum_{j=1}^{N_p}\left(-S_{ji}+\psi_i^1(x_{N_p}^1)\psi_j^1(x_{N_p}^1)-\psi_i^1(x_1^1)\psi_j^1(x_1^1)\right)\hat{B}_j^1(t)\ , \\
\frac{d\hat{B}_i^1(t)}{dt} &= -\frac{2c}{h^1}\sum_{j=1}^{N_p}\left(-S_{ji}+\psi_i^1(x_{N_p}^1)\psi_j^1(x_{N_p}^1)\right)\hat{E}_j^1(t)\ , \\
\frac{d\hat{E}_i^k(t)}{dt} &= -\frac{2c}{h^k}\sum_{j=1}^{N_p}\Big[\left(-S_{ji}+\psi_i^k(x_{N_p}^k)\psi_j^k(x_{N_p}^k)\right)\hat{B}_j^k(t)\;-\;\psi_i^k(x_1^k)\psi_j^{k-1}(x_1^k)\hat{B}_j^{k-1}(t)\Big], \\
&\qquad\qquad 2\le k\le K-1,
\end{aligned}
$$

$$\frac{d\hat{B}_i^k(t)}{dt} = -\frac{2c}{h^k}\sum_{j=1}^{N_p}\left[\left(-S_{ji}+\psi_i^k(x_{N_p}^k)\psi_j^k(x_{N_p}^k)\right)\hat{E}_j^k(t) - \psi_i^k(x_1^k)\psi_j^{k-1}(x_1^k)\hat{E}_j^{k-1}(t)\right],$$

$$2 \le k \le K-1,$$

$$\frac{d\hat{E}_i^K(t)}{dt} = -\frac{2c}{h^K}\sum_{j=1}^{N_p}\left[\left(-S_{ji}+\psi_i^K(x_{N_p}^K)\psi_j^K(x_{N_p}^K)\right)\hat{B}_j^K(t) - \psi_i^K(x_1^K)\psi_j^{K-1}(x_1^K)\hat{B}_j^{K-1}(t)\right],$$

$$\frac{d\hat{B}_i^K(t)}{dt} = -\frac{2c}{h^K}\sum_{j=1}^{N_p}\left[-S_{ji}\hat{E}_j^K(t) - \psi_i^K(x_1^K)\psi_j^{K-1}(x_1^K)\hat{E}_j^{K-1}(t)\right].$$

Rewriting the same in terms of Legendre polynomials:

$$\frac{d\hat{E}_i^1(t)}{dt} = -\frac{2c}{h^1}\sum_{j=1}^{N_p}\left(-S_{ji}+P_{i-1}(1)P_{j-1}(1)-P_{i-1}(-1)P_{j-1}(-1)\right)\hat{B}_j^1(t),$$

$$\frac{d\hat{B}_i^1(t)}{dt} = -\frac{2c}{h^1}\sum_{j=1}^{N_p}\left(-S_{ji}+P_{i-1}(1)P_{j-1}(1)\right)\hat{E}_j^1(t),$$

$$\frac{d\hat{E}_i^k(t)}{dt} = -\frac{2c}{h^k}\sum_{j=1}^{N_p}\left[\left(-S_{ji}+P_{i-1}(1)P_{j-1}(1)\right)\hat{B}_j^k(t) - P_{i-1}(-1)P_{j-1}(1)\hat{B}_j^{k-1}(t)\right],$$

$$2 \le k \le K-1,$$

$$\frac{d\hat{B}_i^k(t)}{dt} = -\frac{2c}{h^k}\sum_{j=1}^{N_p}\left[\left(-S_{ji}+P_{i-1}(1)P_{j-1}(1)\right)\hat{E}_j^k(t) - P_{i-1}(-1)P_{j-1}(1)\hat{E}_j^{k-1}(t)\right],$$

$$2 \le k \le K-1,$$

$$\frac{d\hat{E}_i^K(t)}{dt} = -\frac{2c}{h^K}\sum_{j=1}^{N_p}\left[\left(-S_{ji}+P_{i-1}(1)P_{j-1}(1)\right)\hat{B}_j^K(t) - P_{i-1}(-1)P_{j-1}(1)\hat{B}_j^{K-1}(t)\right],$$

$$\frac{d\hat{B}_i^K(t)}{dt} = -\frac{2c}{h^K}\sum_{j=1}^{N_p}\left[-S_{ji}\hat{E}_j^K(t) - P_{i-1}(-1)P_{j-1}(1)\hat{E}_j^{K-1}(t)\right].$$

This allows to form space discretization matrices A_E and A_B (which also incorporate boundary conditions for electrical and magnetic fields) for the both cases of the numerical

flux choice, and once it is done, the problem reduces to the set of ODE problems:

$$\begin{cases} \dfrac{d\hat{\mathbf{E}}}{dt} = cA_B\hat{\mathbf{B}}, \\ \dfrac{d\hat{\mathbf{B}}}{dt} = cA_E\hat{\mathbf{E}}, \end{cases}$$

with initial values $\hat{\mathbf{E}}(0)$, $\hat{\mathbf{B}}(0)$ computed by means of Legendre polynomial basis expansions of initial conditions of the original problem (68).

Next, we proceed to perform integration in time applying StaggeredLF4 scheme according to (63)-(65):

$$\begin{pmatrix} \hat{\mathbf{E}}^{n+1} \\ \hat{\mathbf{B}}^{n+3/2} \end{pmatrix} = \begin{pmatrix} I & S_1 \\ S_2 & I + S_2S_1 \end{pmatrix} \begin{pmatrix} \hat{\mathbf{E}}^{n} \\ \hat{\mathbf{B}}^{n+1/2} \end{pmatrix},$$

where we denote $S_1 = kc\left(A_B + \dfrac{1}{24}k^2c^2A_BA_EA_B\right)$, $\quad S_2 = kc\left(A_E + \frac{1}{24}k^2c^2A_EA_BA_E\right)$.

Finally, we utilize Legendre polynomial basis expansions again to pass from the coefficients $\hat{\mathbf{E}}$, $\hat{\mathbf{B}}$ to the real values of the fields E, B in space at the final time of integration.

Fig. 11-14 show results for both choices of numerical flux and different values of the time step (that is seen on the plots by varying total time of integration T keeping the same number of time steps): regular and critical (i.e. the time step size corresponding to the stability region border - when instability just starts to occur) . The following parameters were used: length of physical (spatial) domain $L = 10$, the light propagation speed $c = 0.9$, number of elements $K = 10$, number of points inside an element $N_p = 4$, number of time steps $M = 20$.

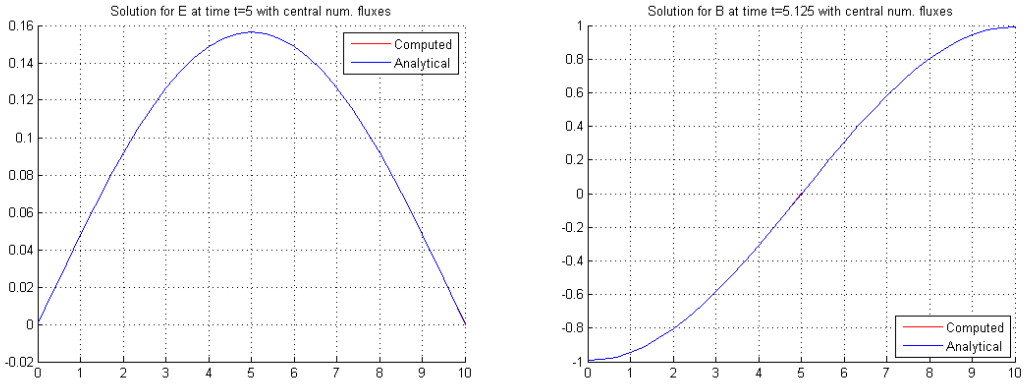

Fig. 11: Solution of Maxwell's equations using the DG method with central numerical fluxes and the time step such that stability condition is surely fulfilled (numerical solution totally fits analytical)

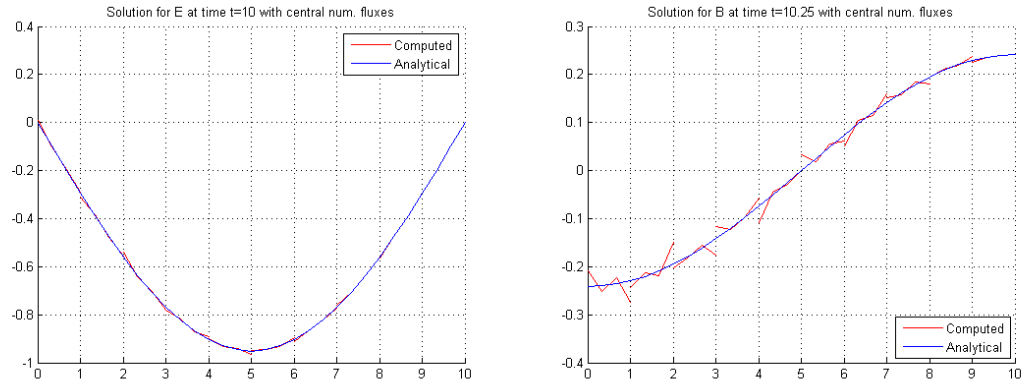

Fig. 12: Solution of Maxwell's equations using the DG method with central numerical fluxes and the time step such that stability condition starts being violated

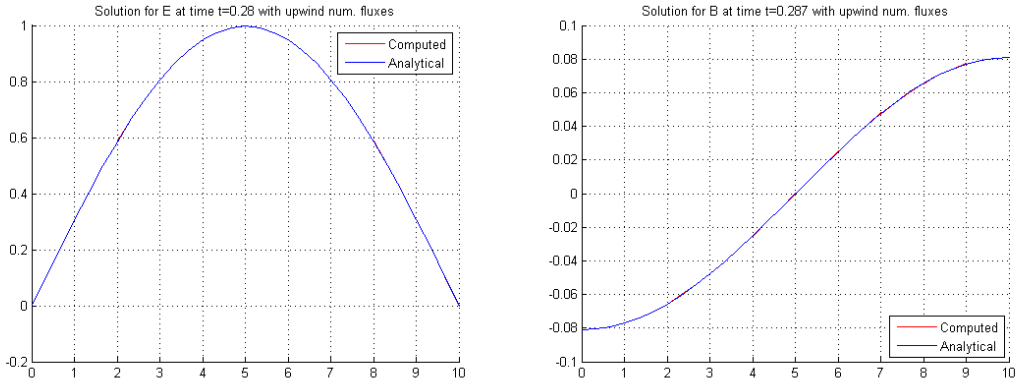

Fig. 13: Solution of Maxwell's equations using the DG method with upwind numerical fluxes and the time step such that stability condition is surely fulfilled (numerical solution totally fits analytical)

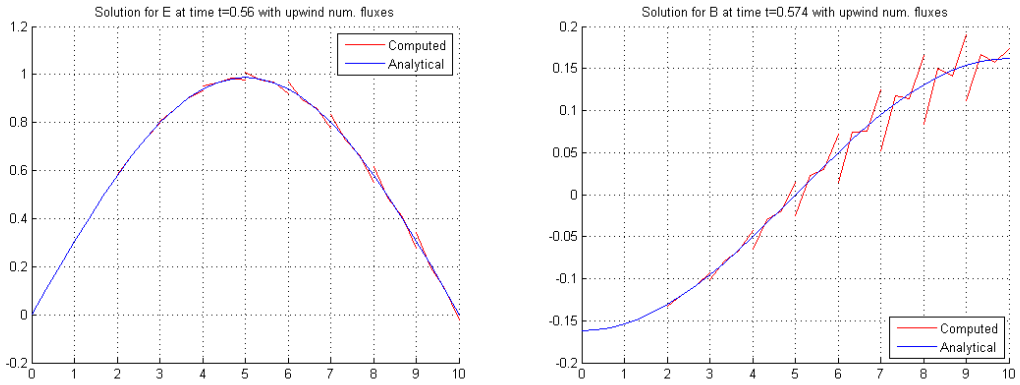

Fig. 14: Solution of Maxwell's equations using the DG method with upwind numerical fluxes and the time step such that stability condition starts being violated

5 Conclusions and final remarks

In the present work, overview of different numerical schemes, general notions and essential properties of numerical methods were pedagogically considered and accent was made on application of the Discontinuous Galerkin method to do spatial discretization first and then employ the StaggeredLeapFrog4 finite difference scheme to perform integration in time of a linear hyperbolic problem, namely electromagnetic wave propagation problem was covered.

Discontinuous Galerkin methods due to (53) allow to achieve any desired high order accuracy in space by refining mesh or increasing interpolation order inside an element, whereas the StaggeredLF4 scheme gives the fourth order of accuracy in time. The latter, being an explicit scheme has limitations dictated by the stability restrictions.

It happened to be not feasible to explicitly express stability condition by means of finding spectral radius of amplification matrix of the whole DG-StaggeredLF4 method. Complication is related with the fact that the amplification matrix C in (63) turns out to be extremely close to the identity matrix (that is no wonder due to the fact that for quite fine spatial grid we obviously have a diagonally dominated matrix with values on the diagonal close to unities).

The particular problem of electromagnetics was considered to apply the discussed method and study stability properties depending on choice of the numerical flux on internal boundaries between all the elements. Choice of the numerical flux, that is an essential ingredient of the DG method, as it was demonstrated at the end of the second part, does not have strong impact on the solution when high-order interpolation is used, however, according to the example giving in the third part, a numerical flux might affect stability properties

of the method. In the present work two typical choices of the linear numerical flux were considered - purely upwind and purely central. The results obtained and plotted at the end of the previous part allow us to draw conclusion that for modal approach utilizing Legendre polynomial basis functions to approximate solution by the DG method use of the purely central numerical fluxes is much more preferable in comparison with purely upwind due to less strict limitation on time step size dictated by the stability issue. This can be seen on those plots where instability starts to occur, these results are plotted for different choices of total time of integration (that is different maximal values of time steps providing the total number of time steps is fixed) for the purely upwind and purely central numerical fluxes cases. This allows us to conclude that the purely central numerical flux gives an opportunity to use approximately more than 15 times greater time step being compared to the purely upwind numerical flux case.

It still needs to be verified if the same result holds for the more commonly chosen nodal approach, that is using Lagrangian polynomial basis for solution approximation inside an element.

6 Acknowledgements

The present work was made under assistance and collaboration with Stéphane Lanteri, Victorita Dolean and Stéphane Descombes. The topic of the current pedagogical research work was also proposed by them and, therefore, the work would not only be completed, but would not even appear at all without their direct participation.

7 Appendix - MATLAB Codes

MATLAB codes for the programs that were used within the text of the current work for demonstations are given below in the following order:

- StaggeredLF2 scheme demonstration for the wave equation
- Demonstration of the DG method for the advection equation - purely central internal numerical fluxes
- Demonstration of the DG method for the advection equation - purely upwind internal numerical fluxes
- Maxwell's equations problem in between 2 metallic plates - purely central internal numerical fluxes
- Maxwell's equations problem in between 2 metallic plates - purely upwind internal numerical fluxes
- Auxiliary function for symbolic computation of normalized Legendre polynomials used in the DG method

22.07.09 23:44 D:\MATLAB\LF2_stable.m 1 of 3

```
%% StaggeredLF2 scheme demonstration by Dmitry Ponomarev (22/07/2009).


% Define space and time intervals and velocity
L=10;
T=10;
c=1.5;



% Define uniform space and time grid

N=49; % 50 space intervals
M=99; % 100 time intervals

k=T/(M+1);
h=L/(N+1);

x=zeros(N+2);
t=zeros(M+2);

x=0:h:L;
t=0:k:T;


% Here, k=0.1, h=0.2, c=1.5
% CFL condition k <= h/c is satisfied (1 < 4/3), thus we have stability


% Define desired time values to plot the solution at
times=[0, 0.01*T, 0.03*T, 0.05*T, 0.07*T, 0.1*T, 0.3*T, 0.5*T, 0.7*T, T];


% Initialization of variables
y=zeros(1,N+2);
y_ex=zeros(1,N+2);
u=zeros(M+2,N+2);
v=zeros(M+2,N+2);
%u_ex=zeros(1,N+2);
%v_ex=zeros(1,N+2);


% Initial conditions
u(1,:)=pi*c/L*cos(pi*(x+h/2)/L);
v(1,:)=zeros(1,N+2);

% Boundary conditions on v
v(:,1)=zeros(1,M+2);
v(:,(N+2))=zeros(1,M+2);

% Computation in time

% Time loop
for n=1:(M+1)

    % Computation in space
```

```
    % Separated calculation utilizing boundary condition
    u(n+1,1)=u(n,1)+k*c/h*v(n,2);

    % Space loop
    for j=2:(N+1)
        u(n+1,j)=u(n,j)+k*c/h*(v(n,j+1)-v(n,j));
        v(n+1,j)=v(n,j)+k*c/h*(u(n,j)-u(n,j-1)+k*c/h*(v(n,j+1)-2*v(n,j)+v(n,j-1)));
    end

    % Separated calculation utilizing boundary condition
    u(n+1,N+2)=u(n+1,N+1);

end

% Solutions for the auxilary variables u and v may be verified
%  syms t_;
%
%  % Exact solutions for u and v
%  u_ex=pi*c/(2*L)*(cos(pi/L*(x+h/2-c*t_))+cos(pi/L*(x+h/2+c*t_)));
%  v_ex=pi/(2*L)*(-cos(pi/L*(x-c*(t_+k/2)))+cos(pi/L*(x+c*(t_+k/2))));
%
%  for i=1:length(times)
%      t_=times(i);
%
%      figure
%      plot(x,eval(u_ex),'-b', x,u(1+round((M+1)*t_/T),:),'-.r');
%      legend('Exact solution', 'StaggeredLF2');
%      title(['Plot for u at t=',num2str(t_)]);
%      grid on;
%
%      figure
%      plot(x,eval(v_ex),'-b', x,v(1+round((M+1)*t_/T),:),'-.r');
%      legend('Exact solution', 'StaggeredLF2');
%      title(['Plot for v at t=',num2str(t_)]);
%      grid on;
%  end


% Verification of the solution for y

syms t_;
% Exact solition for y
y_ex=0.5*(sin(pi/L*(x-c*t_))+sin(pi/L*(x+c*t_)));

 for i=1:length(times)
    t_=times(i);

% Boundary condition on y
    y(1)=0;

% Integrating u/c over space to get y
    for j=2:(N+2)
       y(j)=y(j-1)+h/c*u(1+round((M+1)*t_/T),j-1);
    end
```

```
% % Alternatively, to recover y, we can integrate v in time
%     for j=2:(N+2)
%        y(j)=sin(pi*x(j)/L)+k*sum(v(1:round(1+(M+1)*t_/T),j));
%     end

    figure
    plot(x,eval(y_ex),'-b', x,y,'-.r');
    title(['Solution of the wave equation at t=',num2str(t_)])
    legend('Exact solution', 'StaggeredLF2');
    grid on
end
```

```
%% Advection equation DG-BackwardEuler solver by Dmitry Ponomarev (22/07/2009).

% We use Legendre basis functions and as internal numerical fluxes we take
% purely central fluxes.


K=20; % number of elements = space intervals
Np=2; % number of points inside an element = interpolation order + 1

L=10; % spatial interval
c=0.05; % velocity
h=L/K; % size of an element


% Initialization

M=2; % number of time points
T=10; % total time of integration

dt=T/(M-1); % time step size
t=0:dt:T; % time discretization

X=0:h:L; % spatial interval partitioning into elements
x=zeros(K,Np); % grid matrix; first index stands for an element number, second for a↙
point inside it


u=zeros(M,K*Np); % solution vector
u_=zeros(M,K*Np); % vector of expansion coefficients

S_=zeros(Np,Np); % stiffness matrix

A=zeros(K*Np,K*Np); % discretization matrix
B=zeros(K*Np,K*Np); % amplification matrix
lmbds_A=zeros(1,K*Np); % spectrum of A
lmbds_B=zeros(1,K*Np); % spectrum of B

I=eye(K*Np,K*Np); % auxiliary identity matrix

% Initial condition
syms x_;
b=sin(x_);

% Boundary condition (on the left boundary)
a=-sin(c*t);


% Generating grid

for k=1:K
    for l=1:Np
        x(k,l)=X(k)+h*(l-1)/(Np-1);
    end
end

% Transforming initial condition into expansion coefficient vector
```

```
syms x_;
for k=1:K
    for l=1:Np
        j=(k-1)*Np+l;
        u_(1,j)=2/h*int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,l-1)*b,x_,x(k,1),x↙
(k,Np));
    end
end


% Computing the stiffness matrix

k=1; % take arbitrary element, since it is the same for all elements
for i=1:Np
    for j=1:Np
       tmp=diff(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,i-1),x_);
       S_(i,j)=int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,j-1)*tmp,x_,x(k,1),x(k,↙
Np));
    end
end


% Applying the DG method to obtain spatial discretization matrix A

for i=1:K*Np
    for j=1:K*Np
        if (mod(i,Np)~=0) i_=mod(i,Np); else i_=Np; end
        if (mod(j,Np)~=0) j_=mod(j,Np); else j_=Np; end
        if ((i<=Np) && (j<=Np))
% k=1; % the first element (left boundary)
% Since boundary condition is not homogeneous, it doesn't contribute to the matrix
            A(i,j)=-2*S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A(i,j+Np)=legendre_norm_symb(1,i_-1)*legendre_norm_symb(-1,j_-1);
        elseif ((i>(K-1)*Np) && (j>(K-1)*Np))
% k=K; % the last element (right boundary)
% Since c is positive, no boundary conditions condition can be imposed
% here: the value here is completely defined by the equation
            A(i,j)=-2*S_(i_,j_)+2*legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-↙
1);
            A(i,j)=A(i,j)-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,j_-1);
            A(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
        elseif ((j>Np) && (floor((j-1)/Np)==floor((i-1)/Np)) && (j<=(K-1)*Np))
% k=1+floor((j-1)/Np); % all internal elements
            A(i,j)=-2*S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A(i,j)=A(i,j)-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,j_-1);
            A(i,j+Np)=legendre_norm_symb(1,i_-1)*legendre_norm_symb(-1,j_-1);
            A(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
        end
    end
end

A=-1/h*A;


% Computing amplification matrix
```

```
B=inv(I-c*dt*A);


% Computing contribution from (left) boundary condition and integrating
% altogether using BackwardEuler scheme

f=zeros(1,K*Np);

for m=1:(M-1)

    for i=1:Np
        f(i)=2/h*c*a(m+1)*legendre_norm_symb(-1,i-1);
    end

    u_(m+1,:)=B*u_(m,:)'+dt*B*f';

    m % displays current time step to track status and estimate computational time

end

% Recover solution from its expansion coefficient vector

for m=1:M
    u(m,:)=zeros(1,K*Np);
    for k=1:K
        for l=1:Np
            j=(k-1)*Np+l;
            for i=1:Np
                u(m,j)=u(m,j)+u_(m,(k-1)*Np+i)*legendre_norm_symb((2*x(k,l)-x(k,1)-x(k,↙
Np))/h,i-1);
            end
        end
    end

end


% Plotting solution

figure;
hold on;
grid on;
for k=1:K
    plot(x(k,:),u(M,(1+(k-1)*Np):k*Np), '-r');
    plot(x(k,:),sin(x(k,:)-c*t(M)), '-b');
    legend('Computed', 'Analytical', 'Location', 'SouthEast');
%    legend('Computed', 'Analytical');

end
title(['Solution at T=', num2str(T), ' with central num. fluxes']);


% Stability checks

lmbds_A=eig(A);
```

```
lmbds_B=eig(B);

max(real(lmbds_A)) % maximal real eigenvalue of discretization matrix
max(abs(lmbds_B)) % spectral radius of amplification matrix
```

```
%% Advection equation DG-BackwardEuler solver by Dmitry Ponomarev (22/07/2009).

% We use Legendre basis functions and as internal numerical fluxes we take
% purely upwind fluxes.


K=10; % number of elements = space intervals
Np=4; % number of points inside an element = interpolation order + 1

L=10; % spatial interval
c=0.05; % velocity
h=L/K; % size of an element


% Initialization

M=2; % number of time points
T=10; % total time of integration

dt=T/(M-1); % time step size
t=0:dt:T; % time discretization

X=0:h:L; % spatial interval partitioning into elements
x=zeros(K,Np); % grid matrix; first index stands for an element number, second for a↙
point inside it


u=zeros(M,K*Np); % solution vector
u_=zeros(M,K*Np); % vector of expansion coefficients

S_=zeros(Np,Np); % stiffness matrix

A=zeros(K*Np,K*Np); % discretization matrix
B=zeros(K*Np,K*Np); % amplification matrix
lmbds_A=zeros(1,K*Np); % spectrum of A
lmbds_B=zeros(1,K*Np); % spectrum of B

I=eye(K*Np,K*Np); % auxiliary identity matrix

% Initial condition
syms x_;
b=sin(x_);

% Boundary condition (on the left boundary)
a=-sin(c*t);


% Generating grid

for k=1:K
    for l=1:Np
        x(k,l)=X(k)+h*(l-1)/(Np-1);
    end
end

% Transforming initial condition into expansion coefficient vector
```

```
syms x_;
for k=1:K
    for l=1:Np
        j=(k-1)*Np+l;
        u_(1,j)=2/h*int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,l-1)*b,x_,x(k,1),x↙
(k,Np));
    end
end


% Computing the stiffness matrix

k=1; % take arbitrary element, since it is the same for all elements
for i=1:Np
    for j=1:Np
       tmp=diff(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,i-1),x_);
       S_(i,j)=int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,j-1)*tmp,x_,x(k,1),x(k,↙
Np));
    end
end


% Applying the DG method to obtain spatial discretization matrix A

for i=1:K*Np
    for j=1:K*Np
        if (mod(i,Np)~=0) i_=mod(i,Np); else i_=Np; end
        if (mod(j,Np)~=0) j_=mod(j,Np); else j_=Np; end
        if ((i<=Np) && (j<=Np))
% k=1; % the first element (left boundary)
% Since boundary condition is not homogeneous, it doesn't contribute to the matrix
            A(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
        elseif ((i>(K-1)*Np) && (j>(K-1)*Np))
% k=K; % the last element (right boundary)
% Since c is positive, no boundary conditions condition can be imposed
% here: the value here is completely defined by the equation
            A(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
        elseif ((j>Np) && (floor((j-1)/Np)==floor((i-1)/Np)) && (j<=(K-1)*Np))
% k=1+floor((j-1)/Np); % all internal elements
            A(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
        end
    end
end

A=-2/h*A;


% Computing amplification matrix

B=inv(I-c*dt*A);


% Computing contribution from (left) boundary condition and integrating
```

```
% altogether using BackwardEuler scheme

f=zeros(1,K*Np);

for m=1:(M-1)

    for i=1:Np
        f(i)=2/h*c*a(m+1)*legendre_norm_symb(-1,i-1);
    end

    u_(m+1,:)=B*u_(m,:)'+dt*B*f';

    m % displays current time step to track status and estimate computational time

end

% Recover solution from its expansion coefficient vector

for m=1:M
    u(m,:)=zeros(1,K*Np);
    for k=1:K
        for l=1:Np
            j=(k-1)*Np+l;
            for i=1:Np
                u(m,j)=u(m,j)+u_(m,(k-1)*Np+i)*legendre_norm_symb((2*x(k,l)-x(k,1)-x(k,↙
Np))/h,i-1);
            end
        end
    end

end


% Plotting solution

figure;
hold on;
grid on;
for k=1:K
    plot(x(k,:),u(M,(1+(k-1)*Np):k*Np), '-r');
    plot(x(k,:),sin(x(k,:)-c*t(M)), '-b');
    legend('Computed', 'Analytical', 'Location', 'SouthEast');
%    legend('Computed', 'Analytical');
end
title(['Solution at T=', num2str(T), ' with upwind num. fluxes']);


% Stability checks

lmbds_A=eig(A);
lmbds_B=eig(B);

max(real(lmbds_A)) % maximal real eigenvalue of discretization matrix
max(abs(lmbds_B)) % spectral radius of amplification matrix
```

```
%% Maxwell's equation DG-StaggeredLF4 solver by Dmitry Ponomarev (22/07/2009).

% We use Legendre basis functions and as internal numerical fluxes we take
% purely central fluxes.


K=10; % number of elements = space intervals
Np=4; % number of points inside an element  = interpolation order + 1

L=10; % spatial interval
c=0.9; % light propagation speed
h=L/K; % size of an element

M=21; % number of time points
T=10; % time of integration


% Initialization

dt=T/(M-1); % time step size
t=0:dt:T; % time discretization

X=0:h:L; % spatial interval partitioning into elements
x=zeros(K,Np); % grid matrix; first index stands for an element number, second for a↙
point inside it

E=zeros(M,K*Np); % electric field vector
E_=zeros(M,K*Np); % vector of coefficients of electric field expansion
B=zeros(M,K*Np); % magnetic field vector
B_=zeros(M,K*Np); % vector of coefficients of magnetic field expansion
w=zeros(M,2*K*Np); % combined electric and magnetic fields vector
w_=zeros(M,2*K*Np); % combined vector of coefficients

S_=zeros(Np,Np); % stiffness matrix

% Discretization matrices for electric and magnetic fields

A_E=zeros(K*Np,K*Np);
A_B=zeros(K*Np,K*Np);

% Some auxiliary matrices

S1=zeros(K*Np,K*Np);
S2=zeros(K*Np,K*Np);
C1=zeros(K*Np,K*Np);
C2=zeros(K*Np,K*Np);

I=eye(K*Np,K*Np);

% Amplification matrix and its eigenvalues

C=zeros(2*K*Np,2*K*Np);
lmbds_C=zeros(1,2*K*Np);

% Initial conditions
```

```
syms x_;
a=sin(pi*x_/L);
b=cos(pi*x_/L)*sin(-pi*c*0.5*dt/L); % this is not zero due to the staggered grid

% Generating grid

for k=1:K
    for l=1:Np
        x(k,l)=X(k)+h*(l-1)/(Np-1);
    end
end


% Transforming initial conditions into expansion coefficient vectors

syms x_;
for k=1:K
    for l=1:Np
        j=(k-1)*Np+l;
        E_(1,j)=2/h*int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,l-1)*a,x_,x(k,1),x(k,Np));
        B_(1,j)=2/h*int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,l-1)*b,x_,x(k,1),x(k,Np));
    end
end


% Computing the stiffness matrix

k=1; % take arbitrary element, since it is the same for all elements
for i=1:Np
    for j=1:Np
       tmp=diff(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,i-1),x_);
       S_(i,j)=int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,j-1)*tmp,x_,x(k,1),x(k,Np));
    end
end


% Applying the DG method to obtain spatial discretization matrices A_E and A_B

for i=1:K*Np
    for j=1:K*Np
        if (mod(i,Np)~=0) i_=mod(i,Np); else i_=Np; end
        if (mod(j,Np)~=0) j_=mod(j,Np); else j_=Np; end
        if ((i<=Np) && (j<=Np))
% k=1; % the first element (left boundary)
% We use homogeneneous conditions: Dirichlet for E, Neumann for B (by utilizing ghost cell principle)
            A_E(i,j)=-S_(i_,j_)+0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_E(i,j+Np)=0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(-1,j_-1);
            A_B(i,j)=-S_(i_,j_)+0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j)=A_B(i,j)-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,j_-1);
            A_B(i,j+Np)=0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(-1,j_-1);
```

```
        elseif ((i>(K-1)*Np) && (j>(K-1)*Np))
% k=K; % the last element (right boundary)
% Again we use homogeneneous conditions: Dirichlet for E, Neumann for B (by utilizing
right ghost cell)
            A_E(i,j)=-S_(i_,j_)-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,
j_-1);
            A_E(i,j-Np)=-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j)=A_B(i,j)-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,j_-
1);
            A_B(i,j-Np)=-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
        elseif ((j>Np) && (floor((j-1)/Np)==floor((i-1)/Np)) && (j<=(K-1)*Np))
% k=1+floor((j-1)/Np); % all internal elements
            A_E(i,j)=-S_(i_,j_)+0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-
1);
            A_E(i,j)=A_E(i,j)-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,j_-
1);
            A_E(i,j-Np)=-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
            A_E(i,j+Np)=0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(-1,j_-1);
            A_B(i,j)=-S_(i_,j_)+0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-
1);
            A_B(i,j)=A_B(i,j)-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,j_-
1);
            A_B(i,j-Np)=-0.5*legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j+Np)=0.5*legendre_norm_symb(1,i_-1)*legendre_norm_symb(-1,j_-1);
        end
    end
end

A_E=-2/h*A_E;
A_B=-2/h*A_B;


% Forming amplification matrix C

S1=dt*c*(A_B+1/24*(dt*c)^2*A_B*A_E*A_B);
S2=dt*c*(A_E+1/24*(dt*c)^2*A_E*A_B*A_E);
C1=cat(1,I,S2);
C2=cat(1,S1,I+S2*S1);
C=cat(2,C1,C2);

% Forming combined electric and magnetic fields expansion coefficients vector
w_(1,:)=cat(2,E_(1,:),B_(1,:));


% Applying the StaggeredLF4 scheme to perform time integration

for m=1:(M-1)

    w_(m+1,:)=C*w_(m,:)';

    m % displays current time step to track status and estimate computational time

end
```

```
% Recover electric and magnetic fields from the expansion coefficients vector

% In order to decrease running time we recover solutions just at the final time of integration,
% but when interested in dynamics on stability boundary, uncommenting this loop will allow
% keeping some transient solutions for plotting

m=M;
%for m=1:M %

    E(m,:)=zeros(1,K*Np);
    B(m,:)=zeros(1,K*Np);

    for k=1:K
        for l=1:Np
            j=(k-1)*Np+l;
            for i=1:Np
                E(m,j)=E(m,j)+w_(m,(k-1)*Np+i)*legendre_norm_symb((2*x(k,l)-x(k,1)-x(k,Np))/h,i-1);
                B(m,j)=B(m,j)+w_(m,(K+k-1)*Np+i)*legendre_norm_symb((2*x(k,l)-x(k,1)-x(k,Np))/h,i-1);
            end
        end
    end
%end


% Plotting solutions for electric and magnetic fields

figure;
hold on;
grid on;
for k=1:K
    plot(x(k,:),E(M,(1+(k-1)*Np):k*Np), '-r');
    plot(x(k,:),sin(pi*x(k,:)/L)*cos(pi*c*t(M)/L), '-b');
    legend('Computed', 'Analytical');
end
title(['Solution for E at time t=', num2str(T), ' with central num. fluxes']);

figure;
hold on;
grid on;
for k=1:K
    plot(x(k,:),B(M,(1+(k-1)*Np):k*Np), '-r');
    plot(x(k,:),cos(pi*x(k,:)/L)*sin(-pi*c*(t(M)+0.5*dt)/L), '-b');
    legend('Computed', 'Analytical', 'Location', 'SouthEast');
%    legend('Computed', 'Analytical');
end
title(['Solution for B at time t=', num2str(T+dt/2), ' with central num. fluxes']);

% Stability checks

lmbds_C=eig(C);
max(abs(lmbds_C)) % spectral radius of the amplification matrix
```

```
%% Maxwell's equation DG-StaggeredLF4 solver by Dmitry Ponomarev (22/07/2009).

% We use Legendre basis functions and as internal numerical fluxes we take
% purely upwind fluxes.


K=10; % number of elements = space intervals
Np=4; % number of points inside an element = interpolation order + 1

L=10; % spatial interval
c=0.9; % light propagation speed
h=L/K; % size of an element

M=21; % number of time points
T=0.56; % 10*0.9/0.05=10/18~=0.56; time of integration


% Initialization

dt=T/(M-1); % time step size
t=0:dt:T; % time discretization

X=0:h:L; % spatial interval partitioning into elements
x=zeros(K,Np); % grid matrix; first index stands for an element number, second for a↙
point inside it

E=zeros(M,K*Np); % electric field vector
E_=zeros(M,K*Np); % vector of coefficients of electric field expansion
B=zeros(M,K*Np); % magnetic field vector
B_=zeros(M,K*Np); % vector of coefficients of magnetic field expansion
w=zeros(M,2*K*Np); % combined electric and magnetic fields vector
w_=zeros(M,2*K*Np); % combined vector of coefficients

S_=zeros(Np,Np); % stiffness matrix

% Discretization matrices for electric and magnetic fields

A_E=zeros(K*Np,K*Np);
A_B=zeros(K*Np,K*Np);

% Some auxiliary matrices

S1=zeros(K*Np,K*Np);
S2=zeros(K*Np,K*Np);
C1=zeros(K*Np,K*Np);
C2=zeros(K*Np,K*Np);

I=eye(K*Np,K*Np);

% Amplification matrix and its eigenvalues

C=zeros(2*K*Np,2*K*Np);
lmbds_C=zeros(1,2*K*Np);

% Initial conditions
```

```
syms x_;
a=sin(pi*x_/L);
b=cos(pi*x_/L)*sin(-pi*c*0.5*dt/L); % this is not zero due to the staggered grid

% Generating grid

for k=1:K
    for l=1:Np
        x(k,l)=X(k)+h*(l-1)/(Np-1);
    end
end


% Transforming initial conditions into expansion coefficient vectors

syms x_;
for k=1:K
    for l=1:Np
        j=(k-1)*Np+l;
        E_(1,j)=2/h*int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,l-1)*a,x_,x(k,1),x↙
(k,Np));
        B_(1,j)=2/h*int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,l-1)*b,x_,x(k,1),x↙
(k,Np));
    end
end


% Computing the stiffness matrix

k=1; % take arbitrary element, since it is the same for all elements
for i=1:Np
    for j=1:Np
       tmp=diff(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,i-1),x_);
       S_(i,j)=int(legendre_norm_symb((2*x_-x(k,1)-x(k,Np))/h,j-1)*tmp,x_,x(k,1),x(k,↙
Np));
    end
end


% Applying the DG method to obtain spatial discretization matrices A_E and A_B

for i=1:K*Np
    for j=1:K*Np
        if (mod(i,Np)~=0) i_=mod(i,Np); else i_=Np; end
        if (mod(j,Np)~=0) j_=mod(j,Np); else j_=Np; end
        if ((i<=Np) && (j<=Np))
% k=1; % the first element (left boundary)
% We use homogeneneous conditions: Dirichlet for E, Neumann for B (by utilizing ghost↙
cell principle)
            A_E(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j)=A_B(i,j)-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(-1,j_-1);
        elseif ((i>(K-1)*Np) && (j>(K-1)*Np))
% k=K; % the last element (right boundary)
% Again we use homogeneneous conditions: Dirichlet for E, Neumann for B (by utilizing↙
right ghost cell)
```

```
            A_E(i,j)=-S_(i_,j_);
            A_E(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
        elseif ((j>Np) && (floor((j-1)/Np)==floor((i-1)/Np)) && (j<=(K-1)*Np))
% k=1+floor((j-1)/Np); % all internal elements
            k_=1+floor((j-1)/Np);
            A_E(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_E(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j)=-S_(i_,j_)+legendre_norm_symb(1,i_-1)*legendre_norm_symb(1,j_-1);
            A_B(i,j-Np)=-legendre_norm_symb(-1,i_-1)*legendre_norm_symb(1,j_-1);
        end
    end
end

A_E=-2/h*A_E;
A_B=-2/h*A_B;


% Forming amplification matrix C

S1=dt*c*(A_B+1/24*(dt*c)^2*A_B*A_E*A_B);
S2=dt*c*(A_E+1/24*(dt*c)^2*A_E*A_B*A_E);
C1=cat(1,I,S2);
C2=cat(1,S1,I+S2*S1);
C=cat(2,C1,C2);


% Forming combined electric and magnetic fields expansion coefficients vector
w_(1,:)=cat(2,E_(1,:),B_(1,:));


% Applying the StaggeredLF4 scheme to perform time integration

for m=1:(M-1)

    w_(m+1,:)=C*w_(m,:)';

    m % displays current time step to track status and estimate computational time

end


% Recover electric and magnetic fields from the expansion coefficients vector

% In order to decrease running time we recover solutions just at the final time of↙
integration,
% but when interested in dynamics on stability boundary, uncommenting this loop will↙
allow
% keeping some transient solutions for plotting

m=M;
%for m=1:M %

    E(m,:)=zeros(1,K*Np);
    B(m,:)=zeros(1,K*Np);
```

```
    for k=1:K
        for l=1:Np
            j=(k-1)*Np+l;
            for i=1:Np
                E(m,j)=E(m,j)+w_(m,(k-1)*Np+i)*legendre_norm_symb((2*x(k,l)-x(k,1)-x(k,↙
Np))/h,i-1);
                B(m,j)=B(m,j)+w_(m,(K+k-1)*Np+i)*legendre_norm_symb((2*x(k,l)-x(k,1)-x↙
(k,Np))/h,i-1);
            end
        end
    end
%end


% Plotting solutions for electric and magnetic fields

figure;
hold on;
grid on;
for k=1:K
    plot(x(k,:),E(M,(1+(k-1)*Np):k*Np), '-r');
    plot(x(k,:),sin(pi*x(k,:)/L)*cos(pi*c*t(M)/L), '-b');
    legend('Computed', 'Analytical');
end
title(['Solution for E at time t=', num2str(T), ' with upwind num. fluxes']);

figure;
hold on;
grid on;
for k=1:K
    plot(x(k,:),B(M,(1+(k-1)*Np):k*Np), '-r');
    plot(x(k,:),cos(pi*x(k,:)/L)*sin(-pi*c*(t(M)+0.5*dt)/L), '-b');
    legend('Computed', 'Analytical', 'Location', 'SouthEast');
%    legend('Computed', 'Analytical');
end
title(['Solution for B at time t=', num2str(T+dt/2), ' with upwind num. fluxes']);

% Stability checks

lmbds_C=eig(C);
max(abs(lmbds_C)) % spectral radius of the amplification matrix
```

22.07.09 23:56 D:\MATLAB\legendre_norm_symb.m 1 of 1

```
%% Orthonormal Legendre polynomial generator by Dmitry Ponomarev (22/07/2009).

% We utilize Rodrigues' formula in order to have symbolic polynomial
% expression of order n with respect to x_.


function [P_]=legendre_norm_symb(x_,n)
if n==0
    P_=1/sqrt(2);
else
    syms r;
    tmp=eval(1/(2^n*factorial(n))*sqrt((2*n+1)/2)*diff((r^2-1)^n,n));
    P_=subs(tmp,x_);
end
end
```

References

[1] Ascher, U. M.

Numerical methods for evolutionary differential equations.

Computational science and engineering , vol. 5, SIAM (2008)

[2] Ghrist, M., Fornberg, B., Driscoll, T.A.

Staggered time integrations for wave equations.

SIAM J. Nummer. Anal. 38, 718-741 (2000)

[3] Hairer, E., Norsett, S.P., Wanner, G.

Solving ordinary differential equations I - Nonstiff problems.

Springer Series in Computational Mathematics, vol. 8. 2nd rev. ed. 1993. Corr. 3rd printing, Springer (2008)

[4] Hesthaven, J., Warburton, T.

Nodal discontinuous Galerkin methods: algorithms, analysis, and applications.

Texts in Applied Mathematics, vol. 54, Springer (2008)

[5] Press, W.A., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.

Numerical recipes - The art of scientific computing.

3rd ed., Cambridge University Press (2007)

[6] Thide, B.

Electromagnetic Field Theory.

Upsilon Books (2008)

[7] Verwer, J.G.

On Time staggering for wave equations.

SIAM J. Sci. Comput. 33, 139-154 (2007)

Contents

1 Introduction . . . 3

2 Approximation of ODEs and PDEs with finite differences . . . 4

2.1 Runge-Kutta methods . . . 5

2.2 Linear multistep methods . . . 6

2.3 Stability, consistency, convergence . . . 7

2.4 Stability on examples . . . 13

2.4.1 Heat equation . . . 13

2.4.2 Wave equation . . . 18

3 Discontinuous Galerkin method . . . 34

4 Application to the equations of electromagnetics . . . 51

5 Conclusions and final remarks . . . 66

6 Acknowledgements . . . 68

7 Appendix - MATLAB Codes . . . 69

Unité de recherche INRIA Sophia Antipolis

2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes

4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique

615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur

INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)

http://www.inria.fr

ISSN 0249-6399