Discontinuous Galerkin method for hyperbolic problems

Rubén Sevilla

Zienkiewicz Centre for Computational Engineering,

College of Engineering, Swansea University, Bay Campus, SA1 8EN, Wales, UK

DG Summer School - Barcelona - July 2017

Contents

1 Introduction

These notes are aimed to complement the computer lab on discontinuous Galerkin methods for hyperbolic problems. They are accompanied by two Matlab academic codes for the

solution of the scalar advection equation and the Maxwell's equations in two dimensions using triangular elements.

2 DG for the scalar advection equation

2.1 Problem statement

Let us consider the scalar advection equation of a quantity $u = u(\boldsymbol{x}, t)$ in a open bounded domain $\Omega \subset \mathbb{R}^{\mathtt{n_{sd}}}$ with boundary $\partial\Omega$, where $\mathtt{n_{sd}}$ denotes the number of spatial dimensions,

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{F}(u) = 0, \tag{1}$$

where $\boldsymbol{F}(u) = \boldsymbol{a}u$ is the hyperbolic flux, with $\boldsymbol{a} = (a_1, \ldots, a_{\mathtt{n_{sd}}})$ the advection velocity vector and $t \in (0, T)$.

The boundary is assumed partitioned as $\partial\Omega = \overline{\Gamma}_{\text{in}} \cup \overline{\Gamma}_{\text{out}}$ where Γ_{in} and Γ_{out} are the inflow and outflow parts of the boundary respectively and they are defined as

$$\Gamma_{\text{in}} = \{\boldsymbol{x} \in \partial\Omega \mid a_n < 0\} \qquad \Gamma_{\text{out}} = \{\boldsymbol{x} \in \partial\Omega \mid a_n > 0\}, \tag{2}$$

where $a_n = \boldsymbol{a} \cdot \boldsymbol{n}$ is the normal component of the velocity vector and $\boldsymbol{n}$ is the outward unit normal vector to $\partial\Omega$.

As usual for hyperbolic problems, boundary conditions can only be imposed on the inflow part of the boundary [1]. Here a Dirichlet boundary condition is considered, namely

$$u(\boldsymbol{x}, t) = u_D(\boldsymbol{x}, t) \qquad \text{on } \Gamma_{\text{in}} \times (0, T). \tag{3}$$

Finally, the strong form of the problem must be completed with an initial condition

$$u(\boldsymbol{x}, t_0) = u_0(\boldsymbol{x}) \qquad \text{in } \Omega. \tag{4}$$

2.2 DG weak formulation with flux splitting

The weak formulation for a generic element Ω_e is obtained after multiplication of Equation (1) by a test function v and integration in Ω_e

$$\int_{\Omega_e} v \frac{\partial u}{\partial t} d\Omega + \int_{\Omega_e} v \boldsymbol{\nabla} \cdot \boldsymbol{F}(u) d\Omega = 0. \tag{5}$$

Integrating by parts the second term in Equation (5), the following weak formulation is obtained

$$\int_{\Omega_e} v \frac{\partial u}{\partial t} d\Omega - \int_{\Omega_e} \boldsymbol{\nabla} v \cdot \boldsymbol{F}(u) d\Omega + \int_{\partial\Omega_e} v F_n(u) d\Gamma = 0, \tag{6}$$

where $F_n(u) = u a_n$ is the normal flux on the boundary and $\boldsymbol{n}$ denotes the outward unit normal vector to $\partial\Omega_e$.

In DG methods, the discontinuous nature of the approximation is accounted for by replacing the physical normal flux at the boundary by a consistent normal numerical flux [2, 3]. A natural choice, for the linear hyperbolic system of interest here, is to employ a flux splitting technique [4, 5].

To define the numerical flux, the normal flux across the boundary is first split as

$$F_n(u) = F_n^-(u) + F_n^+(u), \tag{7}$$

where

$$F_n^-(u) = a_n^- u \qquad F_n^+(u) = a_n^+ u, \tag{8}$$

are the inlfow and outflow normal fluxes respectively and

$$a_n^+ := \frac{1}{2}\Big(a_n + |a_n|\Big) \qquad a_n^- := \frac{1}{2}\Big(a_n - |a_n|\Big). \tag{9}$$

The numerical normal flux, evaluated in terms of the trace of the solution on element Ω_e and the trace of the solution u^{out} on the neighbouring element, is then defined as

$$\hat{F}_n(u, u^{\text{out}}) = F_n^+(u) + F_n^-(u^{\text{out}}) = a_n^+ u + a_n^- u^{\text{out}}. \tag{10}$$

Remark 1. *The upwind character of the numerical flux of Equation (10) is clearly observed by noting that*

$$a_n^+ = \begin{cases} a_n & \text{if } a_n > 0 \\ 0 & \text{if } a_n < 0 \end{cases} \qquad a_n^- = \begin{cases} 0 & \text{if } a_n > 0 \\ a_n & \text{if } a_n < 0 \end{cases}, \tag{11}$$

so, for an inflow boundary the numerical normal flux is given by

$$\hat{F}_n(u, u^{out}) = a_n u^{out}. \tag{12}$$

whereas for an outflow boundary the numerical normal flux is given by

$$\hat{F}_n(u, u^{out}) = a_n u. \tag{13}$$

Introducing the definition of the numerical normal flux given in Equation (10) in the weak form of Equation (6), leads to

$$\int_{\Omega_e} v \frac{\partial u}{\partial t} d\Omega - \int_{\Omega_e} \boldsymbol{\nabla} v \cdot \boldsymbol{F}(u) d\Omega + \int_{\partial\Omega_e} v \hat{F}_n(u, u^{\text{out}}) d\Gamma = 0. \tag{14}$$

Alternatively, integrating by parts again the second term in Equation (14), the following DG weak form is obtained

$$\int_{\Omega_e} v \frac{\partial u}{\partial t} d\Omega + \int_{\Omega_e} v \boldsymbol{\nabla} \cdot \boldsymbol{F}(u) d\Omega + \int_{\partial\Omega_e} v a_n^- [\![u]\!] d\Gamma = 0, \tag{15}$$

where it has been used that

$$\hat{F}_n(u, u^{\text{out}}) - F_n(u) = F_n^-(u^{\text{out}}) - F_n^-(u) = a_n^- [\![u]\!]$$

with $[\![u]\!] = u^{\text{out}} - u$ being the jump of the trace of the solution on the boundary.

2.3 Spatial discretisation

To obtain a semi-discrete formulation, the solution is approximated as

$$u(\boldsymbol{\xi}) \simeq u^h(\boldsymbol{\xi}) = \sum_{j=1}^{\mathtt{n_{en}}} \mathrm{u}_j(t) N_j(\boldsymbol{\xi}) \tag{16}$$

on a reference element with local coordinates $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_{\mathtt{n_{sd}}})$. Here $\mathrm{u}_j(t)$ denotes the value of the solution at node j, N_j is the shape function associated with node j and $\mathtt{n_{en}}$ is the total number of nodes on the element. The nodal shape functions span the space of polynomials up to degree k and the number of nodes per element is equal to the dimension of the corresponding approximation space. For triangular elements $\mathtt{n_{en}} = (k+1)(k+2)/2$.

An isoparametric mapping is typically employed to link the reference element $\hat{\Omega}$ and the physical element Ω_e

$$\begin{aligned} \boldsymbol{\varphi} \; : \hat{\Omega} \subset \mathbb{R}^{\mathtt{n_{sd}}} &\longrightarrow \Omega_e \subset \mathbb{R}^{\mathtt{n_{sd}}} \\ \boldsymbol{\xi} &\longmapsto \boldsymbol{\varphi}(\boldsymbol{\xi}) := \sum_{j=1}^{\mathtt{n_{en}}} \boldsymbol{x}_j N_j(\boldsymbol{\xi}), \end{aligned} \tag{17}$$

where $\boldsymbol{x}_j$ are the nodal coordinates of the physical element Ω_e [6].

Introducing the approximation of u given in Equation (16), using the isoparametric mapping of Equation (17) and employing a Galerkin formulation (i.e. selecting the space of weighting functions equal to the space spanned by the interpolation functions), the following semi-discrete system of ordinary differential equations is obtained

$$\mathbf{M}\frac{\partial \mathbf{U}}{dt} - \sum_{k=1}^{\mathtt{n_{sd}}} a_k \mathbf{C}^k \mathbf{U} + \sum_{f=1}^{\mathtt{n_{fa}}} \mathbf{m}^f [\![\mathbf{U}]\!] = \mathbf{0}, \tag{18}$$

with the elemental matrices defined as

$$\mathbf{M}_{ij} = \int_{\hat{\Omega}} N_i N_j |\mathbf{J}| d\Omega \qquad \mathbf{C}^k_{ij} = \int_{\hat{\Omega}} N_i \sum_{l=1}^{\mathtt{n_{sd}}} J^{-1}_{kl} \frac{\partial N_j}{\partial \xi_l} |\mathbf{J}| d\Omega \qquad \mathbf{m}^f_{ij} = \int_{\hat{\Gamma}} N_i N_j \|\mathbf{J}^f\| d\Gamma, \tag{19}$$

where $\mathbf{J} = \partial\boldsymbol{\varphi}/\partial\boldsymbol{\xi}$ denotes the Jacobian of the isoparametric mapping and $\mathbf{J}^f$ is the Jacobian of the restriction of the isoparametric mapping to an element face. This restriction can be written as

$$\begin{aligned} \boldsymbol{\psi} \; : \hat{\Gamma} \subset \mathbb{R}^{\mathtt{n_{sd}}-1} &\longrightarrow \Gamma^f_e \subset \mathbb{R}^{\mathtt{n_{sd}}} \\ \boldsymbol{\zeta} &\longmapsto \boldsymbol{\psi}(\boldsymbol{\zeta}) := \sum_{j=1}^{\mathtt{n_{fn}}} \boldsymbol{x}_j \hat{N}_j(\boldsymbol{\zeta}), \end{aligned} \tag{20}$$

where $\boldsymbol{x}_j$ denote the face nodal coordinates.

2.4 Temporal discretisation

When the system of ordinary differential equations given by Equation (18) is advanced in time using an explicit time marching algorithm, the scheme requires the solution of a system of equations element by element at each time step, avoiding the assembly and solution of a large sparse linear system in each time step. Here a classical fourth order explicit Runge-Kutta scheme is considered, namely

$$\mathbf{U}^{n+1} = \mathbf{U}^n - \frac{\Delta t}{6}\left(\mathbf{R}^{(1)} + \mathbf{R}^{(2)} + \mathbf{R}^{(3)} + \mathbf{R}^{(4)}\right). \tag{21}$$

The four stages are given by

$$\begin{aligned}
\mathbf{R}^{(1)} &= \mathbf{R}(\mathbf{U}^n, t^n) \\
\mathbf{R}^{(2)} &= \mathbf{R}(\mathbf{U}^n + \mathbf{R}^{(1)}\Delta t/2, t^n + \Delta t/2) \\
\mathbf{R}^{(3)} &= \mathbf{R}(\mathbf{U}^n + \mathbf{R}^{(2)}\Delta t/2, t^n + \Delta t/2) \\
\mathbf{R}^{(4)} &= \mathbf{R}(\mathbf{U}^n + \mathbf{R}^{(3)}\Delta t, t^n + \Delta t).
\end{aligned} \tag{22}$$

where $\mathbf{R}$ denotes the residual of the system of ordinary differential equations

$$\mathbf{R} := -\mathbf{M}^{-1}\left(\sum_{k=1}^{\mathtt{n_{sd}}} a_k \mathbf{C}^k \mathbf{U} + \sum_{f=1}^{\mathtt{n_{fa}}} \mathbf{m}^f [\![\mathbf{U}]\!]\right). \tag{23}$$

This time marching algorithm is known to be conditionally stable, with a stability condition given by

$$\Delta t \leq C \frac{h}{\|\boldsymbol{a}\| k^2} \tag{24}$$

where h is the minimum element size, k is the degree of approximation and C is a constant that depends upon the degree of approximation k, see [7] for more details.

2.5 Computational aspects

2.5.1 Computation of elemental matrices

The computation of the elemental mass and convection matrices $\mathbf{M}$ and $\mathbf{C}^k$ in Equation (19) is performed using a numerical quadrature defined on the reference element with $\mathtt{n_{ip}}$ points $\{\boldsymbol{\xi}_g\}_{g=1}^{\mathtt{n_{ip}}}$ and weights $\{\omega\}_{g=1}^{\mathtt{n_{ip}}}$

$$\begin{aligned}
\mathbf{M}_{ij} &\simeq \sum_{g=1}^{\mathtt{n_{ip}}} N_i(\boldsymbol{\xi}_g) N_j(\boldsymbol{\xi}_g) |\mathbf{J}(\boldsymbol{\xi}_\mathbf{g})| \omega_g \\
\mathbf{C}^k_{ij} &\simeq \sum_{g=1}^{\mathtt{n_{ip}}} N_i(\boldsymbol{\xi}_g) \sum_{l=1}^{\mathtt{n_{sd}}} J(\boldsymbol{\xi}_g)^{-1}_{kl} \frac{\partial N_j(\boldsymbol{\xi}_g)}{\partial \xi_l} |\mathbf{J}(\boldsymbol{\xi}_\mathbf{g})| \omega_g
\end{aligned} \tag{25}$$

Similarly the face mass matrix $\mathbf{m}$ in Equation (19) is computed by using a numerical quadrature defined on the reference face with $\mathtt{m_{ip}}$ points $\{\boldsymbol{\zeta}_g\}_{g=1}^{\mathtt{m_{ip}}}$ and weights $\{\varpi\}_{g=1}^{\mathtt{m_{ip}}}$

$$\mathbf{m}_{ij}^f \simeq \sum_{g=1}^{\mathtt{m_{ip}}} N_i(\boldsymbol{\zeta}_g) N_j(\boldsymbol{\zeta}_g) \|\mathbf{J}^f(\boldsymbol{\zeta}_g)\| \varpi_g. \tag{26}$$

In general it is possible to select the number of integration points to integrate exactly the terms of the elemental mass matrix. In contrast, the terms appearing in the convection matrices $\mathbf{C}^k$ and the face mass matrix $\mathbf{m}$ cannot be computed exactly using a numerical quadrature due to the non-polynomial nature of the inverse of the Jacobian $\mathbf{J}^{-1}$ and the norm of $\mathbf{J}^f$ [8].

2.5.2 Quadrature-free implementation

When the isoparametric mapping of Equation (17) is affine, its Jacobian becomes constant for each element. This can be used to accelerate the code substantially by adopting a quadrature-free implementation of the DG method [9, 10]. In this situation, the elemental matrices are computed as

$$\mathbf{M} = |\mathbf{J}| \hat{\mathbf{M}} \qquad \mathbf{C}^k = |\mathbf{J}| \sum_{l=1}^{\mathtt{n_{sd}}} J_{kl}^{-1} \hat{\mathbf{C}}^l \qquad \mathbf{m}^f = \|\mathbf{J}^f\| \hat{\mathbf{m}}, \tag{27}$$

where the matrices

$$\hat{\mathbf{M}}_{ij} = \int_{\hat{\Omega}} N_i N_j d\Omega \qquad \hat{\mathbf{C}}_{ij}^l = \int_{\hat{\Omega}} N_i \frac{\partial N_j}{\partial \xi_l} d\Omega \qquad \hat{\mathbf{m}}_{ij}^f = \int_{\hat{\Gamma}} N_i N_j d\Gamma, \tag{28}$$

are computed only once in the reference element and stored. This leads to an efficient implementation of the DG method where in each time step the evaluation of the residual of the system of ordinary differential equations given by Equation (23) reduces to matrix-vector product operations. It is also worth noting that in this situation, it is possible to compute the integrals of the matrices in Equation (28) exactly by using a numerical quadrature of order $2k$.

Finally, it is important to note that, when meshes of triangular or tetrahedral elements are considered, the isoparametic element between the reference element $\hat{\Omega}$ and the physical element Ω_e is affine if the edges/faces of the physical element are straight/planar. For other type of elements the condition is more restrictive. Recently a mesh generation technique to exploit this property and accelerate a DG solver has been developed [10].

2.5.3 Implementation details

The codes provided use long names for many of the key variables in order to facilitate its understanding. This section provides a brief description of the most important variables.

The structure `referenceElement` stores all the information related to the reference element. It contains:

- `IPcoordinates`: Coordinates of the element integration points. Dimension $\mathtt{n_{ip}} \times \mathtt{n_{sd}}$.
- `IPweights`: Weights of the element integration points. Dimension $\mathtt{n_{ip}} \times 1$.
- `N`: Element shape functions evaluated at the integration points. Dimension $\mathtt{n_{ip}} \times \mathtt{n_{en}}$.
- `Nxi,Neta`: Derivatives of the element shape functions evaluated at the integration points. Dimension $\mathtt{n_{ip}} \times \mathtt{n_{en}}$.
- `IPcoordinates1d`: Coordinates of the face integration points. Dimension $\mathtt{m_{ip}} \times (\mathtt{n_{sd}} - 1)$.
- `IPweights1d`: Weights of the face integration points. Dimension $1 \times \mathtt{m_{ip}}$.
- `N1d`: Face shape functions evaluated at the integration points. Dimension $\mathtt{m_{ip}} \times \mathtt{n_{fn}}$.
- `faceNodes`: Array of dimension $\mathtt{n_{fa}} \times \mathtt{n_{fn}}$. Row f contains the local numbering of the element nodes in the f-th face.
- `innerNodes`: List of interior nodes.
- `faceNodes1d`: Local node numbering for the face. Dimension $1 \times \mathtt{n_{fn}}$.
- `NodesCoord`: Coordinates of the element nodes used for interpolation. Dimension $\mathtt{n_{en}} \times \mathtt{n_{sd}}$.
- `NodesCoord1d`: Coordinates of the face nodes used for interpolation. Dimension $\mathtt{n_{fn}} \times (\mathtt{n_{sd}} - 1)$.
- `degree`: Degree of approximation (k in these notes).

The arrays `intFaces` and `extFaces` contain geometric information used to compute the integrals on interior and exterior faces respectively. `intFaces` is an array of dimension $\mathtt{n^i_{fa}} \times 5$, where $\mathtt{n^i_{fa}}$ is the total number of interior faces. In two dimensions only the first four columns are of interest and they contain, for a given internal face (i.e. for a given row of `intFaces`)

- The first element sharing this face.
- The local face number for the first element.
- The second element sharing this face.
- The local face number for the second element.

Similarly, `extFaces` is an array of dimension $\mathtt{n^e_{fa}} \times 2$, where $\mathtt{n^e_{fa}}$ is the total number of exterior faces. For a given external face (i.e. for a given row of `extFaces`)

- The element sharing this face.

- The local face number .

In the code provided, `intFaces` and `extFaces` are used to perform two different loops. One for internal faces, where the jump of the solution is known given the two elements sharing this face and another one for external faces where specific expressions of the boundary term must be implemented to account for the boundary conditions.

Figure 1 shows the flowchart of the DG code for solving hyperbolic problems.

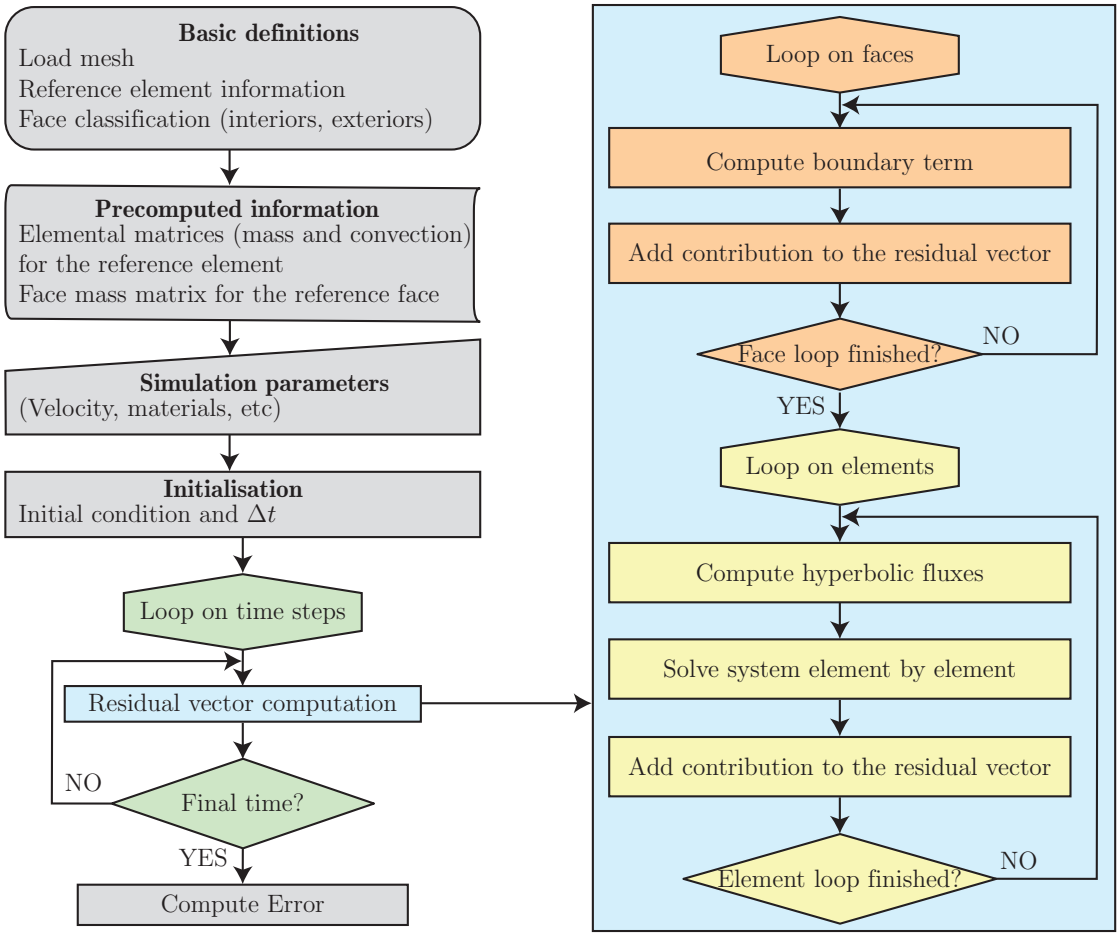

Figure 1: Flowchart of the DG code for solving hyperbolic problems.

2.6 Numerical example

A numerical example is considered to illustrate the concepts presented in this section. The scalar advection equation is solved in the domain $\Omega = [0, 3] \times [0, 1/2]$ using triangular elements with straight edges. Two meshes of the computational domain Ω are represented in Figure 2.

Figure 2: Two computational meshes of the domain $[0, 3] \times [0, 1/2]$.

The advection velocity vector is given by $\boldsymbol{a} = (1, 0)$, the inflow boundary condition is $u_D(\boldsymbol{x}, t) = \sin(\pi(x_1 - t))$ and the initial condition is simply $u_0(\boldsymbol{x}) = 0$.

The numerical solution obtained on the coarse mesh of Figure 2 with cubic approximation at different time instants is shown in Figure 3 to illustrate the evolution of the quantity u in time. It can be observed how the boundary condition at the inflow generates a wave that propagates from left to right due to the positive sign of the horizontal component of the velocity field.

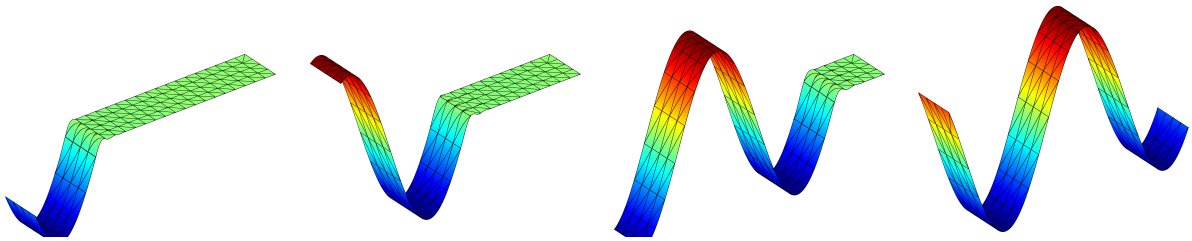

Figure 3: Solution of the scalar advection equations at four time instants.

Next, Figure 4 shows three solutions using different meshes and different degrees of approximation to illustrate the increased accuracy offered by high-order approximations. First, Figure 4(a) shows the solution computed on the coarsest mesh of Figure 2 with linear elements. The low accuracy of this computation can be observed by the high discontinuities between elements. By using the second mesh in Figure 2 with linear elements, the jump of the solution between elements is reduced but still its discontinuous character is clearly observed in Figure 4(b). Finally, Figure 4(c) shows the solution computed on the coarsest mesh of Figure 2 with quadratic elements. The higher accuracy of this computation is clearly observed by the substantial reduction of the jump of the solution in the internal faces.

2.7 Exercise

Modify the code provided to solve the scalar advection equation by introducing a *central numerical flux*

$$\hat{F}_n(u, u^{\text{out}}) = \frac{1}{2}\Big(F_n(u) + F_n(u^{\text{out}})\Big) \tag{29}$$

and compare the solutions obtained against the solution with an upwind numerical flux.

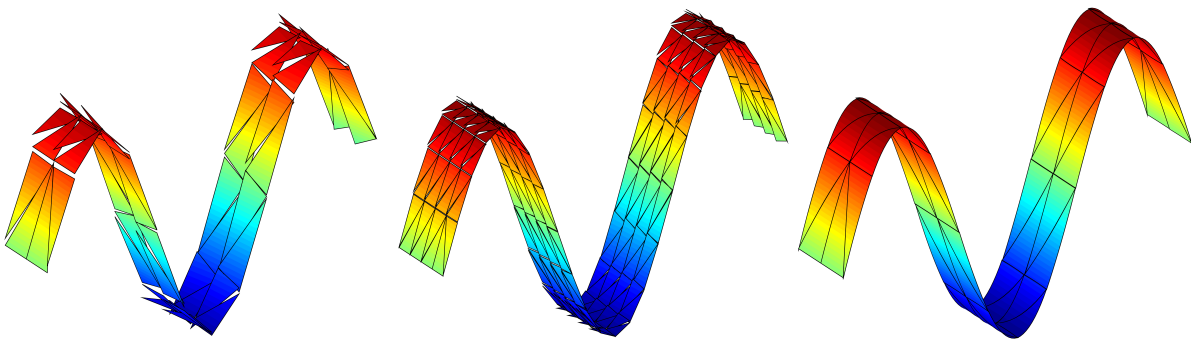

Figure 4: Solution of the scalar advection equation at four time instants.

3 DG for the Maxwell's equations

3.1 Problem statement

Maxwell's curl equations for a linear isotropic non-conducting medium can be written as a hyperbolic system of conservation laws

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{0}, \tag{30}$$

where $\boldsymbol{U}$ is the vector of conserved variables and $\boldsymbol{F} = \left[\boldsymbol{F}_1, \ldots, \boldsymbol{F}_{\mathtt{n_{sd}}}\right]$ the hyperbolic fluxes. In three dimensions, they are given by

$$\boldsymbol{U} = \begin{pmatrix} \varepsilon E_1 \\ \varepsilon E_2 \\ \varepsilon E_3 \\ \mu H_1 \\ \mu H_2 \\ \mu H_3 \end{pmatrix} \qquad \boldsymbol{F}_1 = \begin{pmatrix} 0 \\ H_3 \\ -H_2 \\ 0 \\ -E_3 \\ E_2 \end{pmatrix} \qquad \boldsymbol{F}_2 = \begin{pmatrix} -H_3 \\ 0 \\ H_1 \\ E_3 \\ 0 \\ -E_1 \end{pmatrix} \qquad \boldsymbol{F}_3 = \begin{pmatrix} H_2 \\ -H_1 \\ 0 \\ -E_2 \\ E_1 \\ 0 \end{pmatrix}$$

with $\boldsymbol{E} = (E_1, E_2, E_3)$ and $\boldsymbol{H} = (H_1, H_2, H_3)$ being the electric and magnetic field intensity vectors respectivey, ε the electric permittivity and μ the magnetic permeability [11].

In two dimensions, Maxwell's equations can be decoupled into the so-called transverse electric (TE) and transverse magnetic (TM) modes. These two sets of equations can also be written in the conservation form of (30), with the following definitions for the conserved variables and the fluxes

$$\boldsymbol{U} = \begin{pmatrix} \varepsilon E_1 \\ \varepsilon E_2 \\ \mu H_3 \end{pmatrix} \qquad \boldsymbol{F}_1 = \begin{pmatrix} 0 \\ H_3 \\ E_2 \end{pmatrix} \qquad \boldsymbol{F}_2 = \begin{pmatrix} -H_3 \\ 0 \\ -E_1 \end{pmatrix} \tag{31}$$

for the TE mode, and,

$$\boldsymbol{U} = \begin{pmatrix} \mu H_1 \\ \mu H_2 \\ \varepsilon E_3 \end{pmatrix} \qquad \boldsymbol{F}_1 = \begin{pmatrix} 0 \\ -E_3 \\ -H_2 \end{pmatrix} \qquad \boldsymbol{F}_2 = \begin{pmatrix} E_3 \\ 0 \\ H_1 \end{pmatrix} \tag{32}$$

for the TM mode.

It is also convenient to write the hyperbolic fluxes as

$$\boldsymbol{F}_k(\boldsymbol{U}) = \boldsymbol{A}_k \boldsymbol{U} \tag{33}$$

for $k = 1, \ldots, \mathtt{n_{sd}}$. For instance, for the TE mode in two dimensions, the Jacobian matrices $\boldsymbol{A}_k$ are given by

$$\boldsymbol{A}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1/\mu \\ 0 & 1/\varepsilon & 0 \end{pmatrix} \qquad \boldsymbol{A}_2 = \begin{pmatrix} 0 & 0 & -1/\mu \\ 0 & 0 & 0 \\ -1/\varepsilon & 0 & 0 \end{pmatrix}. \tag{34}$$

Two boundary conditions are usually encountered in electromagnetic problems, namely boundary conditions on unbounded domains and boundary conditions at material interfaces.

When simulating electromagnetic problems in unbounded domains, it is necessary to impose a condition that prescribes the asymptotic behaviour of the solution at infinity [12]. The so-called Silver-Müller radiation boundary condition ensures the existence and uniqueness of the solution of an electromagnetic problem in an unbounded domain and is given by

$$\lim_{r\to\infty}\left(\boldsymbol{x}\times(\nabla\times\boldsymbol{E})+r\frac{\partial\boldsymbol{E}}{\partial t}\right)=\boldsymbol{0},\qquad \lim_{r\to\infty}\left(\boldsymbol{x}\times(\nabla\times\boldsymbol{H})+r\frac{\partial\boldsymbol{H}}{\partial t}\right)=\boldsymbol{0}, \tag{35}$$

where $r=\|\boldsymbol{x}\|_2$.

A detailed derivation and explanation of the boundary conditions for material interfaces can be found in [11]. In several applications, the boundary is made of a highly electrical conducting material that avoids the penetration of electromagnetic fields inside. Such materials are know as perfect electric conductors (PEC), characterised by a vanishing tangential electric field at the surface. On the surface of a PEC, the boundary conditions for the Maxwell's equations are

$$\boldsymbol{n}\times\boldsymbol{E}=\boldsymbol{0}\qquad \boldsymbol{n}\cdot\boldsymbol{H}=0. \tag{36}$$

For the TE mode in two dimensions, the second condition is automatically satisfied and the boundary condition reads

$$n_2E_1-n_1E_2=0. \tag{37}$$

Additionally, Rankine-Hugoniot jump conditions must hold when a boundary discontinuity is propagating at a certain velocity [13]. For a system of conservation laws such as (30), the Rankine-Hugoniot conditions state that the jump of the normal flux is proportional to the jump of the conserved variable, with the proportionality constant being the eigenvalues of the normal Jacobian matrix, namely

$$[\![\boldsymbol{F}_n]\!]=\lambda_j[\![\boldsymbol{U}]\!], \tag{38}$$

where λ_j are the eigenvalues of the matrix $\boldsymbol{A}_n$. For the TE mode in two dimensions the normal flux is

$$\boldsymbol{F}_n=\boldsymbol{F}_1n_1+\boldsymbol{F}_2n_2=\begin{pmatrix}-n_2H_3\\ n_1H_3\\ n_1E_2-n_2E_1\end{pmatrix} \tag{39}$$

and the normal Jacobian matrix is given by

$$\boldsymbol{A}_n=\boldsymbol{A}_1n_1+\boldsymbol{A}_2n_2=\begin{pmatrix}0 & 0 & -n_2/\mu\\ 0 & 0 & n_1/\mu\\ -n_2/\varepsilon & n_1/\varepsilon & 0\end{pmatrix}. \tag{40}$$

The eigenvalues of $\boldsymbol{A}_n$ are

$$\lambda_1 = -c \qquad \lambda_2 = 0 \qquad \lambda_3 = c, \tag{41}$$

where $c = 1/\sqrt{\varepsilon\mu}$ is the velocity of the electromagnetic wave in the corresponding medium.

Finally, the strong form of the problem must be completed with an initial condition

$$\boldsymbol{U}(\boldsymbol{x}, t_0) = \boldsymbol{U}_0(\boldsymbol{x}) \qquad \text{in } \Omega. \tag{42}$$

3.2 DG weak formulation with flux splitting

The derivation of the weak formulation for the Maxwell's equations follows the same rationale presented for the scalar advection equation in Section 2.2. The weak formulation for a generic element Ω_e is obtained after multiplication of (30) by a vector test functions, $\boldsymbol{V}$, and integration in Ω_e

$$\int_{\Omega_e} \boldsymbol{V} \cdot \frac{\partial \boldsymbol{U}}{\partial t} d\Omega - \int_{\Omega_e} \boldsymbol{\nabla V} : \boldsymbol{F}(\boldsymbol{U}) d\Omega + \int_{\partial\Omega_e} \boldsymbol{V} \cdot \boldsymbol{F}_n(\boldsymbol{U}) d\Gamma = 0, \tag{43}$$

where the integration by parts has been already performed on the divergence term.

Following the same ideas presented in Section 2.2, the DG weak formulation for the Maxwell's equations can be written as

$$\int_{\Omega_e} \boldsymbol{V} \cdot \frac{\partial \boldsymbol{U}}{\partial t} d\Omega + \int_{\Omega_e} \boldsymbol{V} \cdot (\boldsymbol{\nabla} \cdot \boldsymbol{F}(\boldsymbol{U}))\, d\Omega + \int_{\partial\Omega_e} \boldsymbol{V} \cdot \boldsymbol{A}_n^- [\![\boldsymbol{U}]\!] d\Gamma = 0, \tag{44}$$

where the numerical normal flux obtained by a flux splitting technique is defined as

$$\hat{\boldsymbol{F}}(\boldsymbol{U}, \boldsymbol{U}^{\text{out}}) = \boldsymbol{F}_n^+(\boldsymbol{U}) + \boldsymbol{F}_n^-(\boldsymbol{U}^{\text{out}}) = \boldsymbol{A}_n^+ \boldsymbol{U} + \boldsymbol{A}_n^- \boldsymbol{U}^{\text{out}} \tag{45}$$

and a second integration by parts is performed on the weak form after introducing the numerical normal flux, leading to the jump term on the boundary of an element.

The matrices $\boldsymbol{A}_n^+$ and $\boldsymbol{A}_n^-$ are defined, after diagonalising the matrix $\boldsymbol{A}_n = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{-1}$, as

$$\boldsymbol{A}_n^+ = \frac{1}{2}(\boldsymbol{A}_n + \boldsymbol{A}_n^{|\!|}), \qquad \boldsymbol{A}_n^- = \frac{1}{2}(\boldsymbol{A}_n - \boldsymbol{A}_n^{|\!|}), \tag{46}$$

where the *absolute value of* $\boldsymbol{A}_n$ is defined as $\boldsymbol{A}_n^{|\!|} := \boldsymbol{P}\boldsymbol{\Lambda}^\star\boldsymbol{P}^{-1}$ and $\boldsymbol{\Lambda}^\star$ is a diagonal matrix containing the absolute value of the eigenvalues of $\boldsymbol{A}_n$ [14, 15].

Using the expressions above, the boundary term appearing in the weak form of Equation (44) can be written as

$$\boldsymbol{A}_n^- [\![\boldsymbol{U}]\!] = \frac{1}{2}\left[[\![H_3]\!] - \sqrt{\varepsilon/\mu} [\![n_1 E_2 - n_2 E_1]\!] \right] \begin{pmatrix} -n_2 \\ n_1 \\ -\sqrt{\mu/\varepsilon} \end{pmatrix}. \tag{47}$$

A common approach for simulating problems in unbounded domains is to consider a first order approximation of the radiation boundary condition given by Equation (35). This is simply imposed by selecting the incoming flux to be zero, that is

$$\boldsymbol{U}^{\text{out}} = \boldsymbol{0}. \tag{48}$$

For a PEC boundary it is important to note that the jump of the quantity $n_1 E_2 - n_2 E_1$ is determined using the boundary condition given by Equation (37), but the jump of H_3 is not determined by Equation (37) and the Rankine-Hugoniout jump conditions must be used. Solving the system of equations given by the Rankine-Hugoniout jump conditions, the following expression is obtained

$$H_3^{\text{out}} = H_3 - \sqrt{\varepsilon/\mu}(n_1 E_2 - n_2 E_1). \tag{49}$$

3.3 Spatial and temporal discretisations

The semi–discrete formulation is obtained by approximating the solution $\boldsymbol{U}$ as

$$\boldsymbol{U}(\boldsymbol{\xi}) \simeq \boldsymbol{U}^h(\boldsymbol{\xi}) = \sum_{j=1}^{\mathtt{n_{en}}} \mathbf{U}_j(t) N_j(\boldsymbol{\xi}) \tag{50}$$

on a reference element with local coordinates $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_{\mathtt{n_{sd}}})$. Here $\mathbf{U}_j(t)$ denotes the value of the solution at node j, N_j is the shape function associated with node j and $\mathtt{n_{en}}$ is the total number of nodes on the element.

Introducing the approximation of the solution in the weak formulation of Equation (44) and selecting the space of the weighting functions to be the same as the space of the interpolation functions, leads to the following system of ordinary differential equations

$$\sum_{j=1}^{\mathtt{n_{en}}} M_{ij} \frac{d\mathbf{U}_j}{dt} + \sum_{j=1}^{\mathtt{n_{en}}} \left(\sum_{k=1}^{\mathtt{n_{sd}}} C_{ij}^k \boldsymbol{A}_k \right) \mathbf{U}_j - \sum_{j=1}^{\mathtt{n_{fn}}} \left(\sum_{f=1}^{\mathtt{n_{fa}}} m_{ij}^f \boldsymbol{A}_n^- \right) [\![\mathbf{U}_j]\!] = \boldsymbol{0}, \tag{51}$$

for each node of the element Ω_e, that is $i = 1, \ldots, \mathtt{n_{en}}$.

The system of ordinary differential equations given by Equation (51) is advanced in time using the classical fourth order explicit Runge-Kutta scheme described in Section 2.4 where the velocity is now c.

3.4 Numerical example

A numerical example is considered to illustrate the concepts presented in this section. The TE mode of the Maxwell's equations is solved in the domain $\Omega = [0, 1]^2$ using triangular elements with straight edges.

The material parameters are $\varepsilon = 1$ and $\mu = 1$, corresponding to free space. The initial condition is given by

$$\boldsymbol{U}_0(\boldsymbol{x}) = \begin{pmatrix} 0 \\ 0 \\ e^{-50[(x_1-0.5)^2+(x_2-0.5)^2]} \end{pmatrix}$$

and a first order absorbing boundary condition is considered on the boundary of the computational domain, simulating the propagation of a pulse in an unbounded domain.

The numerical solution obtained on a coarse mesh with 32 elements and with quintic approximation at different time instants is shown in Figure 5 to illustrate the evolution of the conserved variable $\boldsymbol{U}$ in time. It can be observed how the pulse is absorbed when reaching the outer boundary despite minor reflections are observed due to the first order approximation of the exact radiation boundary condition.

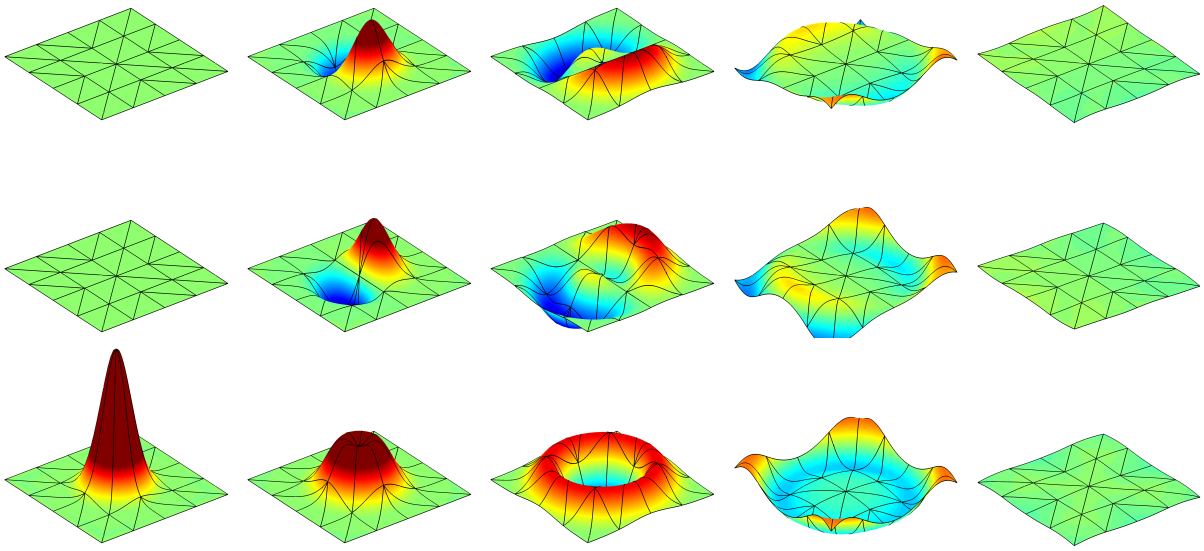

Figure 5: Solution of the Maxwell's equations at five time instants. Each row represents a different components of the vector of conserved variables $\boldsymbol{U}$.

3.5 Exercise

Implement the necessary routines to introduce a PEC boundary condition in the code provided and solve the problem in Section 3.4 with the boundary being a PEC.

References

[1] R. J. LeVeque, *Finite volume methods for hyperbolic problems*. Cambridge university press, 2002.

[2] B. Cockburn, "An introduction to the discontinuous Galerkin method for convection-dominated problems," *Advanced numerical approximation of nonlinear hyperbolic equations*, pp. 151–268, 1998.

[3] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, "The development of discontinuous Galerkin methods," in *Discontinuous Galerkin Methods*, pp. 3–50, Springer, 2000.

[4] J. Donea and A. Huerta, *Finite Element Methods for Flow Problems*. Wiley, 2005.

[5] J. S. Hesthaven and T. Warburton, "Nodal discontinuous Galerkin methods: Algorithms, analysis, and applications," vol. 54 of *Texts in Applied Mathematics*, New York: Springer, 2008.

[6] O. C. Zienkiewicz and R. L.Taylor, *The Finite Element Method*, vol. 1. The basis. Butterwort-Heinemann, fifth ed., 2000.

[7] J. Hesthaven and T. Warburton, "Discontinuous galerkin methods for the time-domain maxwell's equations," *ACES Newsletter*, vol. 19, pp. 10–29, 2004.

[8] R. Sevilla, S. Fernández−Méndez, and A. Huerta, "Comparison of high–order curved finite elements," *International Journal for Numerical Methods in Engineering*, vol. 87, no. 8, pp. 719–734, 2011.

[9] H. L. Atkins and C. W. Shu, "Quadrature-free implementation of discontinuous Galerkin method for hyperbolic equations," *AIAA Journal*, vol. 36, no. 5, pp. 775–782, 1998.

[10] R. Sevilla, O. Hassan, and K. Morgan, "The use of hybrid meshes to improve the efficiency of a discontinuous Galerkin method for the solution of Maxwell's equations," *Computers & Structures*, vol. 137, pp. 2–13, 2014.

[11] C. A. Balanis, *Advanced Engineering Electromagnetics*. New York: John Wiley and Sons, 1989.

[12] D. Givoli, *Numerical Methods for Problems in Infinite Domains*. Amsterdam: Elsevier, 1992.

[13] R. J. LeVeque, *Numerical methods for conservation laws*. Lectures in Mathematics ETH Zürich, Basel: Birkhäuser Verlag, second ed., 1992.

[14] A. V. Kabakian, V. Shankar, and W. F. Hall, "Unstructured grid-based discontinuous galerkin method for broadband electromagnetic simulations," *Journal of Scientific Computing*, vol. 20, no. 3, pp. 405–431, 2004.

[15] J. S. Hesthaven and T. Warburton, "Nodal high-order methods on unstructured grids I. Time-domain solution of Maxwell's equations," *Journal of Computational Physics*, vol. 181, no. 1, pp. 186–221, 2002.