Discontinous Galerkin Method for numerical simulations

TUM School of Engineering and Design
Technical University of Munich
SuSe 2024

1D acoustic wave equation

Niki Balestrieri

Contents

1 Abstract

In this project we are going to analyze the acoustic wave formulation as a system of first-order equations:

$$\begin{aligned} \rho\frac{\partial v}{\partial t} + \nabla p = 0, \\ \frac{1}{c^2}\frac{\partial p}{\partial t} + \rho\nabla \cdot v = 0. \end{aligned} \tag{1}$$

The first representing the linear momentum and the second the mass conservation. The aim of this discussion is to consider and solve this continous system of equations through the **Discontinous Galerkin Method** while presenting the computational solution [1] and observing if the outcome is in accordance with the true solution.

2 Maximum time step formulation

Before proceeding in the analysis of the maximum time step formulation, we have to apply a mathematical machinery in order to obtain the normal form.

2.1 One-dimensional system form

We recall the structure, from the lecture notes, as

$$\frac{\partial w}{\partial t} + D(f(w)) = 0 \text{ for } f(w) = Aw \tag{2}$$

And fitting equations (1) in (2) we recognize and define:

$$\mathbf{w} = \begin{pmatrix} v \\ p \end{pmatrix}, \mathbf{D} := \begin{pmatrix} \nabla\cdot & 0 \\ 0 & \nabla \end{pmatrix}, \mathbf{A} := \begin{pmatrix} 0 & \frac{1}{\rho} \\ \rho c^2 I & 0 \end{pmatrix} \tag{3}$$

And for this particular one-dimensional case we obtain:

$$\mathbf{w} = \begin{pmatrix} v \\ p \end{pmatrix}, \mathbf{D} := \begin{pmatrix} \frac{\partial}{\partial x}\cdot & 0 \\ 0 & \frac{\partial}{\partial x} \end{pmatrix}, \mathbf{A} := \begin{pmatrix} 0 & \frac{1}{\rho} \\ \rho c^2 & 0 \end{pmatrix} \tag{4}$$

Or, in other notation:

$$\frac{\partial}{\partial t}\begin{bmatrix} v \\ p \end{bmatrix} + \begin{bmatrix} 0 & \frac{1}{\rho} \\ \rho c^2 & 0 \end{bmatrix}\frac{\partial}{\partial x}\begin{bmatrix} v \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5}$$

That can draw an analogy to the linear advection equation:

$$\frac{\partial u}{\partial t} + A\frac{\partial u}{\partial x} = 0 \tag{6}$$

2.2 Maximal transport speed

From the Local Lax–Friedrichs flux, useful formulation to derive our maximum eigenvalues, is defined as:

$$\mathbf{f}^*(\mathbf{w}^-, \mathbf{w}^+) = \frac{1}{2}\left(\mathbf{f}(\mathbf{w}^-) + \mathbf{f}(\mathbf{w}^+)\right) + \frac{C}{2}\hat{\mathbf{n}} \cdot (\mathbf{w}^- - \mathbf{w}^+) \tag{7}$$

[1] The used language is Matlab

where w^- denotes the w value of the current element and w^+ denotes the w value of the neighbouring element. Deriving equation (7) for $\mathbf{f}$ in respect to $\mathbf{w}$ and by multiplying by the normal vector $\hat{n} = (\hat{n}_x)^T$, since we are in one dimension, we obtain the following matrix:

$$\begin{bmatrix} 0 & \hat{n}_x \frac{1}{\rho} \\ \hat{n}_x \rho c^2 & 0 \end{bmatrix} \tag{8}$$

Having as characteristic polynomial $\lambda^2 = c^2$, we can compute the eigenvalues of matrix (8) obtaining the corresponding wave speeds, finding λ to be $\pm$ $\mathbf{c}$.
So we can say that the overall solution is a superposition, or linear combination, of these two wave components propagating in opposite directions ($+c$ on the positive x-axis and $-c$ on the negative) at the speed of sound $\mathbf{c}$.

2.3 Maximum time step

The formulation for this system would be given by [2]

$$\Delta t_{max} \leq \frac{Cr}{max(\lambda)} \frac{h}{k^{1.5}} = \frac{Cr}{c} \frac{h}{k^{1.5}} \tag{9}$$

since we are working with $k \leq 10$, we prefer using $k^1.5$ instead of k^2.
The choice of the speed, and consequentially the time stepping scheme, is of great importance: in fact if, for example, we would have picked a random time step not bounded by the system's eigenvalues, so the wave would have been travelling faster than the element size, the result would have not been the right solution.

3 Matrix formulation

3.1 Weak form

First of all we need the formulation of the weak form for this problem [3], which are respectively

$$\left(\mathbf{l}_i^{(e)}, \frac{\partial \mathbf{v}}{\partial t}\right)_{\Omega_e} - \left(\nabla \cdot \mathbf{l}_i^{(e)}, \frac{1}{\rho} p\right)_{\Omega_e} + \left\langle \mathbf{l}_i^{(e)} \cdot \hat{n}, \frac{1}{\rho} p^* \right\rangle_{\partial\Omega_e} = 0 \tag{10}$$

$$\left(l_i^{(e)}, \frac{\partial p}{\partial t}\right)_{\Omega_e} - \left(\nabla \cdot l_i^{(e)}, \rho c^2 \mathbf{v}\right)_{\Omega_e} + \left\langle l_i^{(e)} \cdot \hat{n}, \rho c^2 v^* \right\rangle_{\partial\Omega_e} = 0 \tag{11}$$

Step by step, from each addendum, we will derive the global matrix formulation, composed of a $\mathcal{M}$ mass matrix, $\mathcal{S}$ stiffness matrix and $\mathcal{F}$ flux matrix. This is an important passage to then define a good functioning of the Matlab script.

3.2 Mass matrix

From the weak form (10) we analyze the first addendum [4]

$$\left(\mathbf{l}_i^{(e)}, \frac{\partial \mathbf{v}}{\partial t}\right)_{\Omega_e} = \int_{\Omega_e} l_i^{(e)}(x) \cdot \frac{\partial}{\partial t} v(x,t) dx = \frac{\partial}{\partial t} \int_{\Omega^e} l_i^{(e)}(x) v(x,t) dx \tag{12}$$

[2] Following the CFL restriction for higher order elements
[3] In bold are identified the vectors, otherwise scalar values
[4] The same process will be applied also to Eq.(11) in the same way.

Now we define

$$v(x,t)\Big|_{\Omega^e} = \sum_{j=0}^{n_p} l_j^{(e)}(x) \cdot v_j^{(e)}(t) \tag{13}$$

And we substitute in (12)

$$\frac{\partial}{\partial t} \int_{\Omega^e} l_i^{(e)}(x) \sum_{j=0}^{n_p} l_j^{(e)}(x) \cdot v_j^{(e)}(t) dx \tag{14}$$

Since we are dealing with a finite summation, where the sum is a row-by-column product, we can operate

$$\frac{\partial}{\partial t} \sum_{j=0}^{n_p} v_j^{(e)}(t) \int_{\Omega^e} l_i^{(e)}(x) \cdot l_j^{(e)}(x) dx \tag{15}$$

Defining for simplicity

$$m_{i,j}^{(e)} = \int_{\Omega^e} l_i^{(e)}(x) \cdot l_j^{(e)}(x) dx \text{ for } i,j = 0,\dots,n_p \tag{16}$$

We can rewrite (15) as

$$\frac{\partial}{\partial t} \sum_{j=0}^{n_p} v_j^{(e)}(t) \cdot m_{i,j}^{(e)} \tag{17}$$

Defining in this way a new block matrix that we are going to call $M_v^{(e)}$ that we can sketch as

$$\frac{\partial}{\partial t} \Big[M^{(e)} \Big]_{row} \Big[v^{(e)}(t) \Big]_{column} \quad \text{for } v^{(e)}(t) = (v_0^{(e)}, v_1^{(e)}, \dots, v_{n_p}^{(e)}) \tag{18}$$

With the same procedure, starting from weak form (11), we obtain a block matrix defined as $M_p^{(e)}$, and putting all the equations together we obtain the following mass matrix:

$$\mathcal{M} = \left[\begin{array}{ccc|ccc} M_v^{(1)} & & & & & \\ & M_v^{(2)} & & & \mathbf{0} & \\ & & \dots & & & \\ & & M_v^{(n)} & & & \\ \hline & & & M_p^{(1)} & & \\ & \mathbf{0} & & & M_p^{(2)} & \\ & & & & & \dots \\ & & & & & M_p^{(n)} \end{array}\right] \tag{19}$$

being a square matrix defined in 4 blocks, respectively of dimensions n and $n(n_p+1)$, resulting in global dimension $M = (2n(n_p+1), 2n(n_p+1))$.
For the notation used, we specify n_p are the degree of Lagrange polynomials, n subintervals and n_p+1 number of nodes for every subinterval.
In the code, the mass matrix has been implemented as directly its inverse:

```
%% MASS MATRIX
[values,derivatives] = evaluate_lagrange_basis(xunit, pg);
Me = values * diag(wg) * values';
M1inv = sparse(kp1*n,kp1*n);
for e=1:n
    M1inv((kp1*e-k):kp1*e,(kp1*e-k):kp1*e) = inv(0.5*h(e)*Me)
end
Minv = blkdiag(M1inv, M1inv);
```

3.3 Stiffness matrix

Proceeding, we analyze from Eq.(10) the second addendum

$$\left(\nabla \cdot \mathbf{l}_i^{(e)}, \frac{1}{\rho} p\right)_{\Omega_e} = \left(\frac{d}{dx} l_i^{(e)}, \frac{p}{\rho}\right) = \int_{\Omega^e} \frac{d}{dx} l_i^{(e)}(x) \cdot \frac{p(x,t)}{\rho} dx \tag{20}$$

We define now

$$p(x,t)\Big|_{\Omega^e} = \sum_{j=0}^{n_p} l_j^{(e)}(x) \cdot p_j^{(e)}(t) \tag{21}$$

Inserting in (20) the formulation, we obtain

$$\int_{\Omega^e} \frac{d}{dx} l_i^{(e)}(x) \cdot \frac{1}{\rho} \sum_{j=0}^{n_p} l_j^{(e)}(x) \cdot p_j^{(e)}(t) dx = \frac{1}{\rho} \sum_{j=0}^{n_p} p_j^{(e)}(t) \int_{\Omega^e} \frac{d}{dx} l_i^{(e)}(x) \cdot l_j^{(e)}(x) dx \tag{22}$$

Now we define again, for simplicity

$$s_{i,j}^{(e)} = \int_{\Omega^e} \frac{d}{dx} l_i^{(e)}(x) \cdot l_j^{(e)}(x) dx \tag{23}$$

Defining in this way a new block matrix that we are going to call $S_p^{(e)}$ that we can sketch as

$$\frac{1}{\rho}\left[S^{(e)}\right]_{row}\left[p^{(e)}(t)\right]_{column} \quad \text{for } p^{(e)}(t) = (p_0^{(e)}, p_1^{(e)}, \ldots, p_{n_p}^{(e)}) \tag{24}$$

The matrix $S_v^{(e)}$ is derived in the same way, but instead of coefficient $\frac{1}{\rho}$ has ρc^2.
Putting all together we obtain the matrix of stiffness $\mathcal{S}$ sketched in the following way:

$$\mathcal{S} = \left[\begin{array}{ccc|ccc} & & & S_p^{(1)} & & \\ & \mathbf{0} & & & S_p^{(2)} & \\ & & & & & \ddots \\ & & & & & S_p^{(n)} \\ \hline S_v^{(1)} & & & & & \\ & S_v^{(2)} & & & \mathbf{0} & \\ & & \ddots & & & \\ & & S_v^{(n)} & & & \end{array}\right] \tag{25}$$

For having a similar scheme to the mass matrix, we can simply invert the order of the blocks, having firstly on the upper left the dependence from velocity and lower right from pressure.

In the code, it has been implemented in the right hand side of the equation[5]:

```
% Compute operator at quadrature points and multiply by gradient of test
    function (S matrix)
rhs_v((e-1)*kp1+1:e*kp1)=derivatives*(1/rho*weights.*p_quad);
rhs_p((e-1)*kp1+1:e*kp1)=derivatives*(c^2*rho*weights.*v_quad);
```

[5]The reason will be clearer in section 3.8.

3.4 Flux matrix

We proceed with the third addendum of Eq.(10) and (11), formulation defined on the border $\partial\Omega_e$:

$$\left\langle \mathbf{l}_i^{(e)} \cdot \hat{n}, \frac{1}{\rho} p^* \right\rangle_{\partial\Omega_e} \tag{26}$$

$$\left\langle l_i^{(e)} \cdot \hat{n}, \rho c^2 v^* \right\rangle_{\partial\Omega_e} \tag{27}$$

In this case the analysis is more complex due to the presence of two different fluxes: **Lax–Friedrichs (LF) and the Hybrid Discontinous Galerkin (HDG)**.

3.5 Derivation of Lax-Friedrichs

Starting from Eq.(26):

$$\left\langle \mathbf{l}_i^{(e)} \cdot \hat{n}, \frac{1}{\rho} p^* \right\rangle_{\partial\Omega_e} = \left\langle \mathbf{l}_i^{(e)}, \frac{1}{\rho} p^* \cdot \hat{n} \right\rangle_{\partial\Omega_e} = \left\langle \mathbf{l}_i^{(e)}, \frac{1}{2\rho}(p^- + p^+)\hat{n}^- + \frac{c}{2}(v^- - v^+) \right\rangle_{\partial\Omega_e} \tag{28}$$

And since we are dealing with a 1D equation:

$$\left(\frac{1}{2\rho}(p^- + p^+)\langle \mathbf{l}_i^{(e)} \cdot \hat{n}^- \rangle \right)_{\partial\Omega_e} + \left(\frac{c}{2}\langle \mathbf{l}_i^{(e)}, v^- - v^+ \rangle \right)_{\partial\Omega_e} \tag{29}$$

Adopting the same procedure also for Eq.(27) and compressing both fluxes for velocity and pressure in one formulation, we obtain the expressions for the *flux from the left* f_L^* and the *flux from the right* f_R^* :

$$f_L^* = -\frac{1}{2}\begin{bmatrix} \frac{p^- + p^+}{\rho} \\ \rho c^2 (v^- + v^+) \end{bmatrix} + \frac{c}{2}\left[\begin{pmatrix} v^- \\ p^- \end{pmatrix} - \begin{pmatrix} v^+ \\ p^+ \end{pmatrix} \right] \tag{30}$$

$$f_R^* = \frac{1}{2}\begin{bmatrix} \frac{p^- + p^+}{\rho} \\ \rho c^2 (v^- + v^+) \end{bmatrix} + \frac{c}{2}\left[\begin{pmatrix} v^- \\ p^- \end{pmatrix} - \begin{pmatrix} v^+ \\ p^+ \end{pmatrix} \right] \tag{31}$$

In more explicit words, here we are accounting for all the internal faces without considering the borders: from the left $e \geq 2$ and $i = 0$ and from the right $e \leq n_p - 1$ and $i = n_p$. Different is the situation in the case of $e = 1$ for $i = 0$ and $e = n$ for $i = n_p$. [6]

3.6 Derivation of HDG

We adopt the same mathematical procedure for the HDG:

$$\left\langle \mathbf{l}_i^{(e)} \cdot \hat{n}, \frac{1}{\rho} p^* \right\rangle_{\partial\Omega_e} = \left\langle \mathbf{l}_i^{(e)} \cdot \hat{n}, \frac{c}{2}(v^- - v^+) \cdot \hat{n} + \frac{1}{2\rho}(p^+ + p^+) \right\rangle_{\partial\Omega_e} \tag{32}$$

and simplifying for $\hat{n}$, we notice that the formulation accounting for the inner faces is totally equivalent to Lax-Friedrichs. [7]

[6] We will be back shortly on the boundary conditions.

[7] Always taking in account that this is valid just for a 1D analysis.

3.7 Boundary conditions

To complete the analysis, now we study the conditions on the border. They will be of two different types: **Dirichlet** and **Absorbing**.
From here, we can actually start appreciating the differences between the two fluxes.

For the **Lax-Friedrichs** flux:
Dirichlet

- $p^+ = -p^- + 2p_D$ with mirroring condition
- $v^+ = v^-$

Absorbing

- Left: $p^+ = -p^- - 2c\rho v^-$; Right:$p^+ = -p^- + 2c\rho v^-$
- $v^+ = v^-$

For the **HDG** flux, we evaluate directly the flux:
Dirichlet

- Left flux pressure: $-\rho c^2 v^- + \frac{1}{c}(p^- - p_D)$
- Left flux velocity: $-\frac{1}{\rho}p_D$
- Right flux pressure: $\rho c^2 v^- + \frac{1}{c}(p^- - p_D)$
- Right flux velocity: $\frac{1}{\rho}p_D$

Absorbing

- Left flux pressure: $-\frac{\rho c^2 v^-}{2} + \frac{c}{2}p^-$
- Left flux velocity: $-\frac{p^-}{2\rho} + \frac{cv^-}{2}$
- Right flux pressure: $\frac{\rho c^2 v^-}{2} + \frac{c}{2}p^-$
- Right flux velocity: $\frac{p^-}{2\rho} + \frac{cv^-}{2}$

In code terms, the fluxes are implemented in a similar way, so as leading example:

```
%% LAX-FRIEDRICHS:
       % compute advective numerical flux on the left
       pminus = p_e(1);
       vminus = v_e(1);

       if (e==1)
           if (boundary)
               % Dirichlet
               pplus = 2*bc(1)-pminus;
               vplus = vminus;
           else
               % Absorbing condition
               pplus = -pminus - 2*c*rho*vminus;
               vplus= vminus;
```

```
    end
else
    pplus = p((e-1)*kp1);
    vplus = v((e-1)*kp1);
end

numflux_lv = -0.5*(pminus+pplus)/rho + (c/2*(vminus-vplus));
numflux_lp = -0.5*rho*c^2*(vminus + vplus) + c/2 * (pminus - pplus);

rhs_v((e-1)*kp1+1) = rhs_v((e-1)*kp1+1) - numflux_lv;
rhs_p((e-1)*kp1+1) = rhs_p((e-1)*kp1+1) - numflux_lp;
```

3.8 Total formulation

Flowing all the terms together, we can summarize this matrix formulation:

$$\dot{w} = (\mathcal{M})^{-1}(\mathcal{S} - \mathcal{F})w \tag{33}$$

As stylistic choice, we will not be computing a separate matrix $\mathcal{F}_{bound}$, since the $\mathcal{F}$ matrix has been constructed in a way that already contains all the informations for the boundary conditions.

4 TASK 2

4.1 Analytic solution

First of all, we analyze the pressure's graphical solution for $n = 20$, $k = 1$, $T_f = 0.2$ and *Dirichlet* boundary conditions, for both Lax-Friedrichs and HDG.
The parameters $c = \rho = 1$ are kept constant.

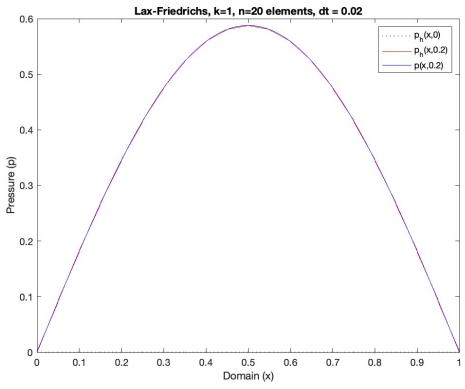

Figure 1: LF n=20, k=1

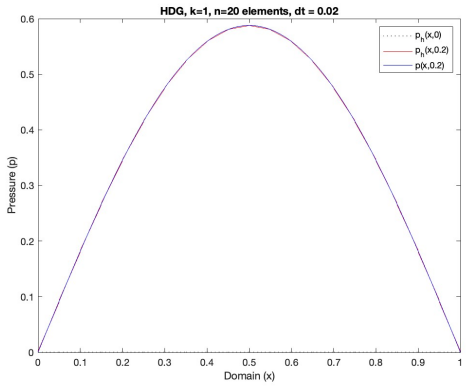

Figure 2: HDG n=20, k=1

We notice how the solutions are perfectly matchable and with the same error approximation, proof of the mathematical derivation established earlier.
So we can conlcude that **the two fluxes are equivalent in 1D**.
Now, we increase the number of elements and polynomial degree to $n = 40$, $k = 4$, $T_f = 0.2$.

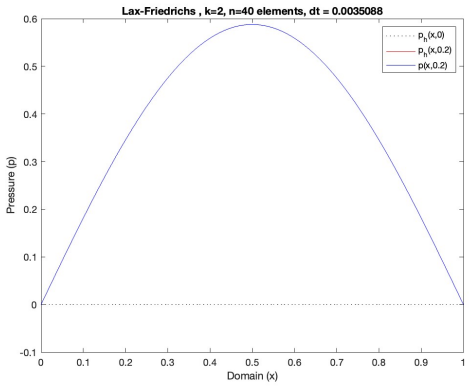

Figure 3: LF n=40, k=2

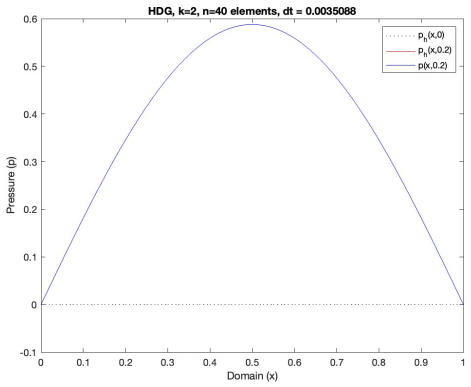

Figure 4: HDG n=40, k=2

Increasing k and n, the infinity error and the L2 norm error decreased.
Following, a more complete analysis of the error's dependences.

4.2 Error Convergence

We proceed analyzing how the error in norm L_2 is related to the number of elements and polynomial degree. For *Dirichlet* boundary conditions the two fluxes are identical, but for coherence we will still represent both.

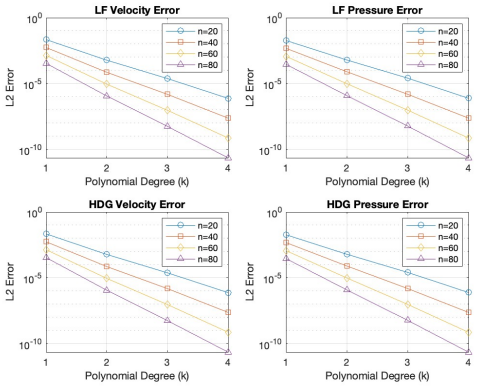

Figure 5: Error convergence over k

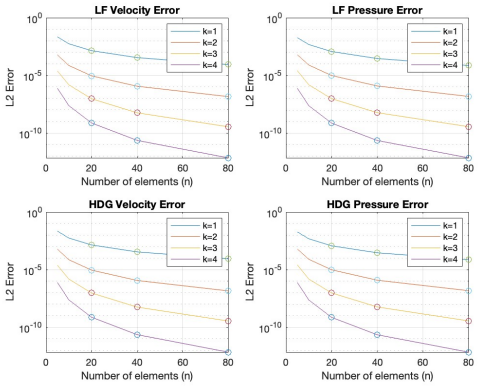

Figure 6: Error convergence over n

And the numerical values outcomes for the pressure:

Lax-Friedrichs Pressure Errors:

	k=1	k=2	k=3	k=4
n=5	0.018777	0.00062065	2.602e-05	7.7101e-07
n=10	0.0047924	7.9928e-05	1.5449e-06	2.3863e-08
n=20	0.0011755	9.8359e-06	9.6479e-08	7.3813e-10
n=40	0.00029167	1.2207e-06	5.9224e-09	2.2892e-11
n=80	7.2618e-05	1.5205e-07	3.6431e-10	7.1896e-13

Figure 7: L2 error norm LF

HDG Pressure Errors:

	k=1	k=2	k=3	k=4
n=5	0.018777	0.00062065	2.602e-05	7.7101e-07
n=10	0.0047924	7.9928e-05	1.5449e-06	2.3863e-08
n=20	0.0011755	9.8359e-06	9.6479e-08	7.3813e-10
n=40	0.00029167	1.2207e-06	5.9224e-09	2.2892e-11
n=80	7.2618e-05	1.5205e-07	3.6431e-10	7.1896e-13

Figure 8: L2 error norm HDG

5 TASK 3

Now we change the parameters in $\rho = 1.2$ and $c = 340$ with initial conditions $v(x, 0) = 0$ and $p(x, 0) = e^{-(x-0.5)^2/0.02^2}$.
In this case we analyze the **HDG flux** with its *Absorbing* boundary condition and equal time steps of $dt = 0.0006s$ until a $T_f = 0.003s$.
For completeness we represent both velocity and pressure.

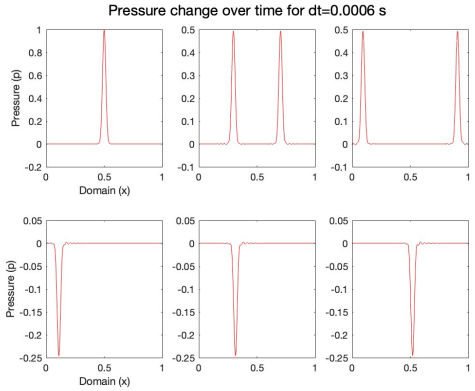

Figure 9: Absorbing pressure change

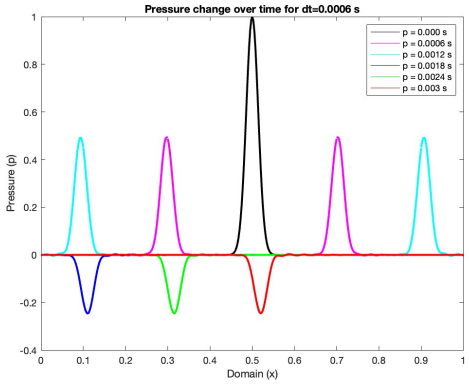

Figure 10: Absorbing pressure change

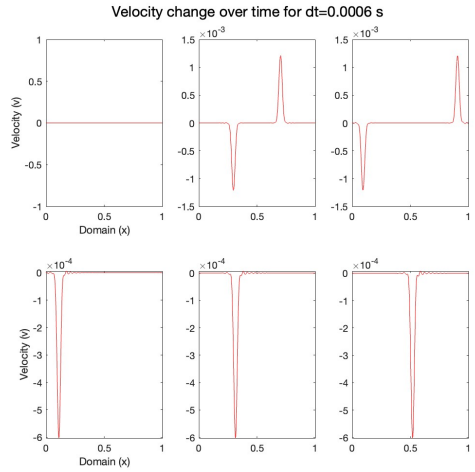

Figure 11: Absorbing velocity change

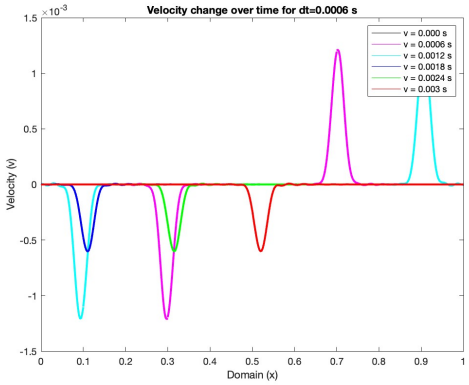

Figure 12: Absorbing velocity change

Now we implement the HDG with *Dirichlet* boundary conditions:

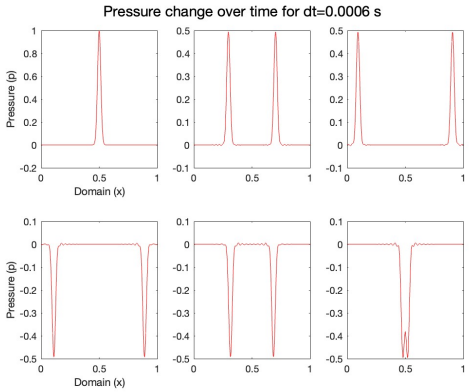

Figure 13: Dirichlet pressure change

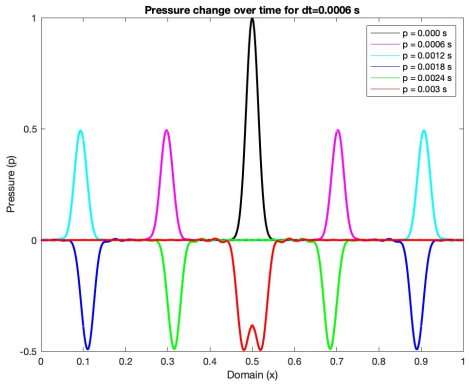

Figure 14: Dirichlet pressure change

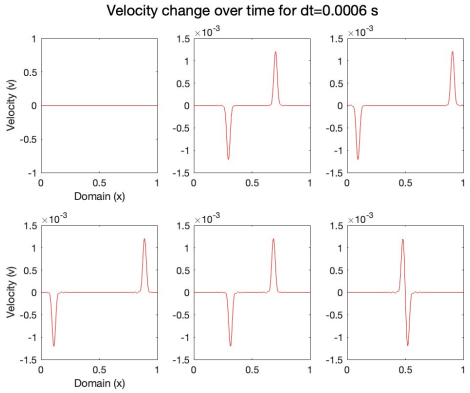

Figure 15: Dirichlet velocity change

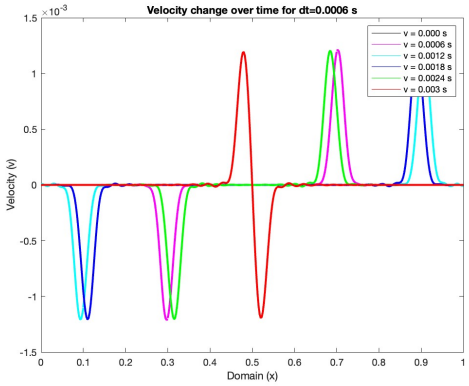

Figure 16: Dirichlet velocity change

The difference between the two boundaries, in terms of the **pressure** in this case, is intrinsic in the mathematical derivation. The HDG flux with *Dirichlet* boundary condition oscillates more, since its values are fixed in the beginning and end of the domain. So when the initial condition is different than zero and in particular an exponential form, the function starts oscillating.
While with *Absorbing* boundaries, this phenomenon is smoothened out due to the freedom it has on the boundary of the domain.

6 TASK 4

6.1 Maximum Courant number

Using the same conditions as Task 2, but with a different formulation for the Courant number (k^2 instead of $k^{1.5}$) we search for the maximum Courant number in the case of $k = 4$, $n = 80$ and $T_f = 0.2$. Graphically:

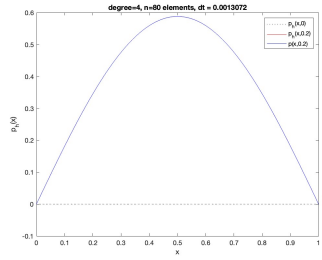

Figure 17: Cr=1.67

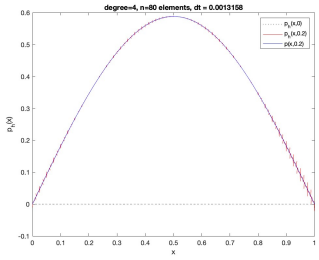

Figure 18: Cr=1.68

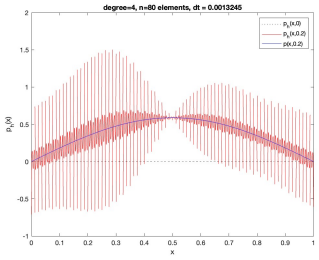

Figure 19: Cr=1.69

Figure 20: Courant number variation

With these settings, the maximum Courant number is 1.68.

7 TASK 5

7.1 Gauss quadrature and Gauss-Lobatto quadrature

Replacing Gauss quadrature with Gauss-Lobatto quadrature in the script used for Task 2, keeping $k = 4$, $n = 80$ and $T_f = 0.2$, we notice three major differences:

- Slight increasing of the error
- Stability with larger time steps
- Diagonal elemental mass matrix M_e

Here for comparison, the elemental matrix M_e for Gauss-Lobatto and Gauss quadrature, and the error values.

```
Me =

    0.1000         0         0         0         0
         0    0.5444         0         0         0
         0         0    0.7111         0         0
         0         0         0    0.5444         0
         0         0         0         0    0.1000
```

Figure 21: Me Gauss-Lobatto

```
Me =

    0.0889    0.0259   -0.0296    0.0259   -0.0111
    0.0259    0.4840    0.0691   -0.0605    0.0259
   -0.0296    0.0691    0.6321    0.0691   -0.0296
    0.0259   -0.0605    0.0691    0.4840    0.0259
   -0.0111    0.0259   -0.0296    0.0259    0.0889
```

Figure 22: Me Gauss quadrature

```
Velocity error in maximum norm 3.9072e-12 in L2 norm 1.3796e-12
Pressure error in maximum norm 3.929e-12 in L2 norm 1.5053e-12
```

Figure 23: Error Gauss-Lobatto

```
Velocity error in maximum norm 1.3164e-12 in L2 norm 6.9451e-13
Pressure error in maximum norm 1.4736e-12 in L2 norm 7.1896e-13
```

Figure 24: Error Gauss quadrature

Keeping Gauss-Lobatto quadrature and setting up the same parameters as Task 4 for the analysis of the maximum Courant number, we can observe that in this case the maximum stable value is exactly doubled respect using Gauss quadrature.

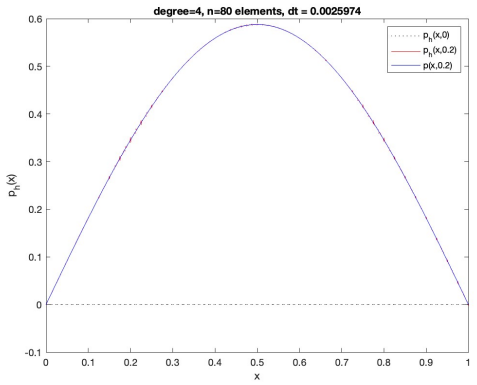

Figure 25: Cr=3.34

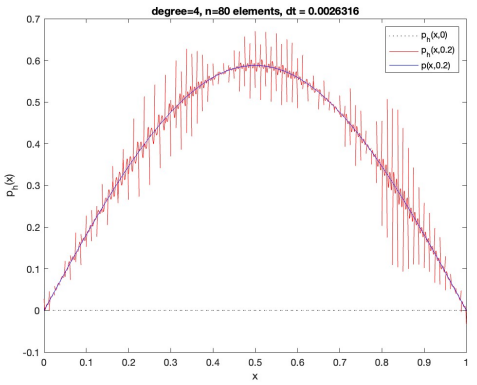

Figure 26: Cr=3.35

This happens because Gauss-Lobatto quadrature, with points including the boundaries, provides better stability properties for the same spatial discretization.

8 TASK 6

8.1 Consistent mass matrix and lumped mass matrix

For the discussion of the consistent mass matrix and lumped mass matrix, the code had to be modified. Since we have to plot the eigenvalue spectrum of the operator $\mathcal{M}^{-1}(\mathcal{S}^T - \mathcal{F})$ over the stability region of Runge-Kutta, we need a comprehensive matrix to gather all the previously defined matrixes.

```
%% S^T - F MATRIX FOR GAUSS QUADRATURE
% Calculate (S^T - F) matrix by calling evaluate_wave_rhs on the vectors of
% the canonical base of R^(2n*nk1)

ST_minus_F_gauss = sparse(2*kp1*n,2*kp1*n);
for j = 1 : 2*kp1*n
    base_vector = sparse(2*kp1*n, 1);
    base_vector(j) = 1;
    matrix_column = evaluate_wave_rhs(base_vector, c, rho, [0,0], values_gauss,
        derivatives_gauss, wg_gauss, flux, boundary);
    ST_minus_F_gauss(:, j) = matrix_column;

end
eigenvalues_gauss = eig(full(Minv_gauss*ST_minus_F_gauss))*dt;
```

And looking at on the polynomial degree $k = 2$ and $k = 5$, we can appreciate the graphical the result.

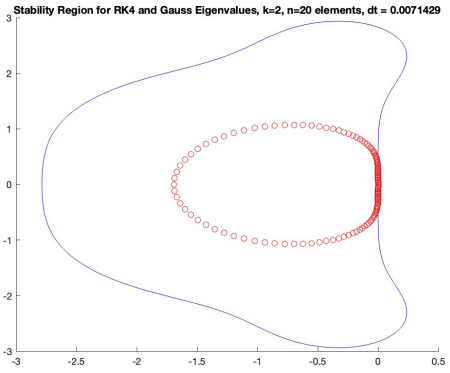

Figure 27: Gauss quadrature k=2

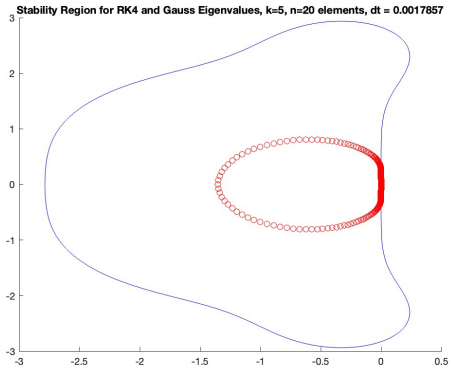

Figure 28: Gauss quadrature k=5

So is clear how, both of the cases are stable on the Runge-Kutta region, but increasing k, the eigenvalues region shrinks, leaving space for bigger time steps keeping its stability.
For this particular case is also interesting to see the variation of the eigenvalues' sprectrum using **Gauss quadrature or Gauss-Lobatto quadrature**, taking inspiration from Task 5. Both methods are computationally implemented in an analogous way.
As we already established, Gauss-Lobatto quadrature allows bigger time steps and this reflects also in the choice of the maximum Courant number, which is doubled compared to Gauss quadrature. Keeping the same parameters:

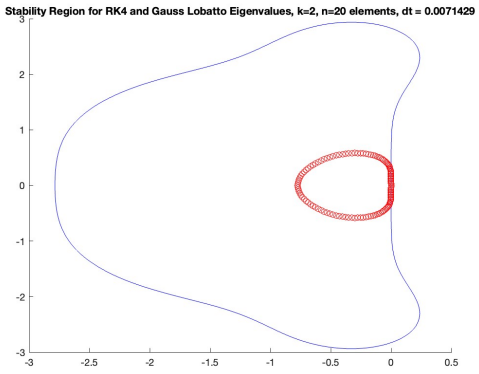

Figure 29: Gauss-Lobatto quadrature k=2

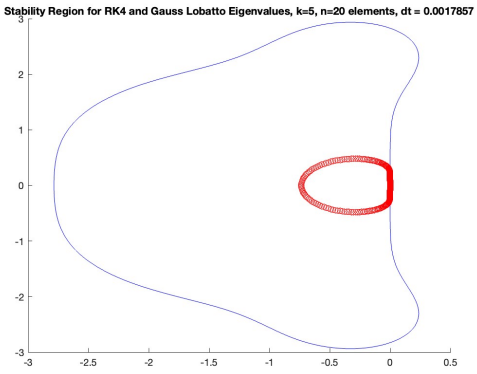

Figure 30: Gauss-Lobatto quadrature k=5

Is evident how Gauss-Lobatto reconferms as more stable, even if slightly less precise.

8.2 Courant number

Now, working on the time steps through the Courant number, we can appreciate the difference between Gauss quadrature and Gauss-Lobatto quadrature in terms of stability in the RK4 region.

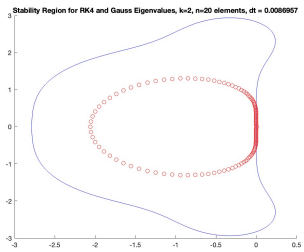

Figure 31: Gauss Cr=0.5

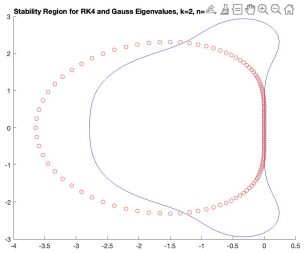

Figure 32: Gauss Cr=0.9

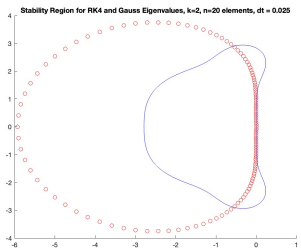

Figure 33: Gauss Cr=1.5

Figure 34: Courant number variation

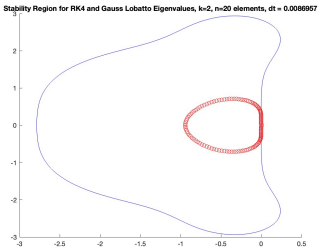

Figure 35: Gauss-Lobatto Cr=0.5

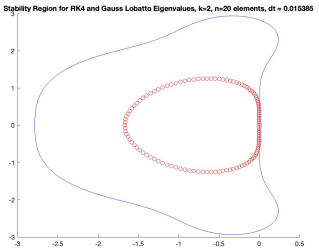

Figure 36: Gauss-Lobatto Cr=0.9

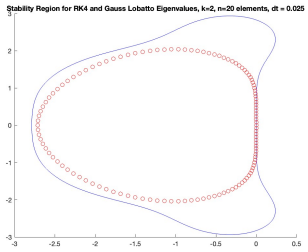

Figure 37: Gauss-Lobatto Cr=1.5

Figure 38: Courant number variation

Once again, is evident how stable is Gauss-Lobatto quadrature compared to Gauss quadrature, allowing bigger time steps but keeping its stability intact.