1 Introduction to discontinuous Galerkin methods

This writeup is based on Reference [7]. A useful book to consult is Reference [6].

In recent years, discontinuous Galerkin (dG) methods have been investigated as an excellent alternative to finite difference and finite volume schemes in numerical simulations involving both parabolic as well as hyperbolic problems (for the advection dominated case see, for example, the review article [4]). Such methods combine many advantages of finite element methods (such as the ease of handling complicated geometry) with properties more commonly associated with finite difference approximations. Examples of the latter includes the absence of a global mass matrix. The main idea of a dG method is to approximate the solution of a differential equation by a polynomial in each grid-cell. Higher/lower order methods can be constructed by simply increasing/decreasing the degree of the polynomials used in each cell. In classical finite element methods continuity is required across cell boundaries. In contrast, dG methods allow discontinuities across cell boundaries, which adds to the flexibility of the method.

For the discretization of second derivatives we discuss the so-called local discontinuous Galerkin (LDG) method [3]. The LDG method and its advantages can also be used for the discretization of elliptic equations (including Poisson's equation). Reference [2] highlights the relation of the method to interior penalty and other alternative methods. A superconvergence result for the LDG approach was proven on Cartesian grids [5], where the order of convergence is $1/2$ better than on arbitrary meshes. Reference [8] later showed a similar result for general, nonlinear elliptic equations.

The downside of the dG methods is their rather complex and unintuitive notation in the existing mathematical literature. Often, algorithms are described in terms of arbitrary sets of polynomials and spatial grids. We propose an adapted, simplified, and in our view more practical notation for orthogonal grids and Legendre polynomials, which can be implemented straightforwardly. We reformulate the LDG method in terms of adjoint matrices and naturally develop the symmetry of the resulting discretization. Doing so, we also propose a new discretization for the general elliptic equation. Unfortunately, our discretization has a wider stencil than the existing ones, but our numerical experiments indicate superconvergent properties and the resulting matrix equation is better conditioned than the old one. To the knowledge of the author these findings are unpublished to date.

1.1 The Legendre polynomials

First, let us consider the one-dimensional case. For simplicity and ease of implementation we choose an equidistant grid with N cells $C_n = [x_{n-1/2}, x_{n+1/2}]$ of size h; with this choice we are able to construct basis functions of $P(C_n)$, the space of polynomials of degree at most $P-1$ on C_n, by using orthogonal Legendre polynomials[1]. The Legendre polynomials can be recursively defined on $[-1,1]$ by setting $p_0(x) = 1$, $p_1(x) = x$ and (see e.g. [1])

$$(k+1)p_{k+1}(x) = (2k+1)xp_k(x) - kp_{k-1}(x). \tag{1}$$

The so constructed Legendre polynomials are orthogonal on $[-1,1]$. We write x_j^a and w_j, $j = 0,\dots,P-1$ denoting the abscissas and weights of the Gauss–Legendre quadrature on the

[1] Often, in the literature the order of the polynomials is denoted by k. Pay attention, that we use the number of polynomial coefficients P instead. We have $P = k+1$

interval $[-1,1]$. Then we note that for $k,l = 0,\dots,P-1$

$$\int_{-1}^{1} p_k(x)p_l(x)\,\mathrm{d}x = \sum_{j=0}^{P-1} w_j p_k(x_j^a)p_l(x_j^a) = \frac{2}{2k+1}\delta_{kl}, \tag{2}$$

since Gauss–Legendre quadrature is exact for polynomials of degree at most $2P-1$.

The discrete completeness relation can then be written as

$$\sum_{k=0}^{P-1} \frac{2k+1}{2} w_j p_k(x_i^a)p_k(x_j^a) = \delta_{ij}. \tag{3}$$

Given a real function $f:[-1,1]\to\mathbb{R}$ we define $f_j := f(x_j^a)$ and

$$\bar{f}^k := \frac{2k+1}{2}\sum_{j=0}^{P-1} w_j p_k(x_j^a) f_j \tag{4}$$

Now let us define the forward transformation matrix by $F^{kj} := \frac{2k+1}{2} w_j p_k(x_j^a)$ and the backward transformation matrix by $B_{kj} := p_j(x_k^a)$.

> **Note**
>
> Reference [6] identifies B as the Vandermonde matrix and further denotes f_i as the **nodal** representation and $\bar{f}^k$ as the **modal** representation of $f(x)$.

Then, using Eq. (4), we get

$$\bar{f}^k = \sum_{j=0}^{P-1} F^{kj} f_j \tag{5a}$$

$$f_j = \sum_{k=0}^{P-1} B_{jk}\bar{f}^k, \tag{5b}$$

We call $\bar{f}^k$ the values of f in L-space and f_j the values of f in X-space.

Let us now consider an interval $[a,b]$ and an equidistant discretization by N cells with cell center x_n and grid size $h = \frac{b-a}{N}$; in addition, we set $x_{nj}^a := x_n + \frac{h}{2}x_j^a$. Given a function $f:[a,b]\to\mathbb{R}$ we then define $f_{nj} := f(x_{nj}^a)$ and note that

$$\bar{\boldsymbol{f}} = (\mathbf{1}\otimes F)\boldsymbol{f} \tag{6a}$$

$$\boldsymbol{f} = (\mathbf{1}\otimes B)\bar{\boldsymbol{f}}, \tag{6b}$$

where f_{nj} are the elements of $\boldsymbol{f}$, $\mathbf{1}\in\mathbb{R}^{N\times N}$ is the identity matrix and $F,B\in\mathbb{R}^{P\times P}$. Furthermore, we use $\otimes$ to denote the Kronecker product which is bilinear and associative. The discontinuous Galerkin expansion f_h of a function f in the interval $[a,b]$ can then readily be given as

$$f_h(x) = \sum_{n=1}^{N}\sum_{k=0}^{P-1} \bar{f}^{nk} p_{nk}(x), \tag{7}$$

where

$$p_{nk}(x) := \begin{cases} p_k\left(\frac{2}{h}(x - x_n)\right), & \text{for } x - x_n \in \left[-\frac{h}{2}, \frac{h}{2}\right] \\ 0, & \text{else.} \end{cases} \tag{8}$$

As an example, we plot Eq. (7) for $f(x) = \sin(2x)$ in Fig. 1. Already with a very low resolution of

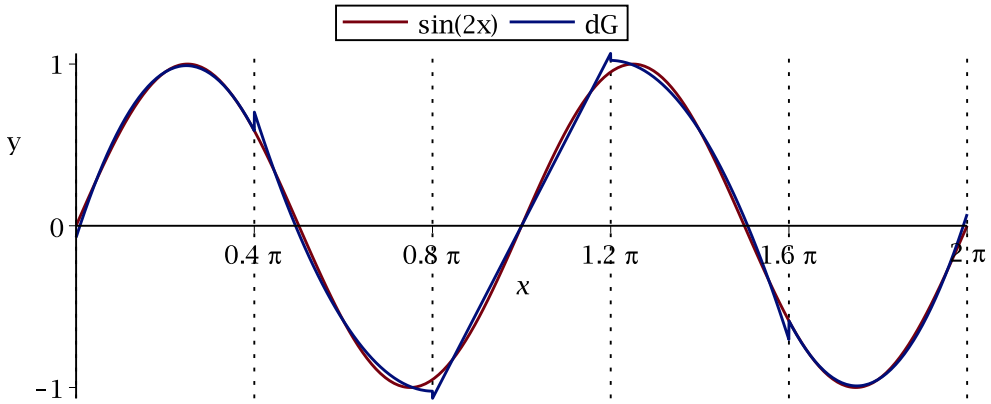

Figure 1: Discretization of a sine function with second order polynomials, $P = 3$, on $N = 5$ grid cells. Dotted lines depict the cell boundaries.

$P = 3$ and $N = 5$ we get an acceptable function approximation. We clearly see the discontinuities at the cell boundaries.

The use of Legendre polynomials yields a natural approximation of the integrals of f via Gauss–Legendre quadrature

$$\langle f_h, g_h\rangle := \int_a^b f_h g_h \,\mathrm{d}x = \sum_{n=1}^{N}\sum_{j=0}^{P-1} \frac{hw_j}{2} f_{nj} g_{nj} = \sum_{n=1}^{N}\sum_{k=0}^{P-1} \frac{h}{2k+1} \bar{f}^{nk}\bar{g}^{nk} \tag{9a}$$

$$\|f_h\|_{L_2}^2 := \int_a^b |f_h|^2 \,\mathrm{d}x = \sum_{n=1}^{N}\sum_{j=0}^{P-1} \frac{hw_j}{2} f_{nj}^2 = \sum_{n=1}^{N}\sum_{k=0}^{P-1} \frac{h}{2k+1} \left(\bar{f}^{nk}\right)^2. \tag{9b}$$

With these formulas we have a simple, accurate, and fast method to evaluate integrals on the entire domain. This is applied, for example, to compute errors in the L_2-norm.

We now define some useful quantities that simplify our notation (note that $i, j = 0, \dots, P-1$)

$$S_{ij} := \int_{-h/2}^{h/2} p_i\left(\frac{2}{h}x\right) p_j\left(\frac{2}{h}x\right) \,\mathrm{d}x = \frac{h}{2i+1}\delta_{ij} \tag{10a}$$

$$T^{ij} := S_{ij}^{-1} = \frac{2i+1}{h}\delta_{ij} \tag{10b}$$

$$W^{ij} := \frac{hw_j}{2}\delta_{ij} \tag{10c}$$

$$V_{ij} := W_{ij}^{-1} = \frac{2}{hw_j}\delta_{ij}. \tag{10d}$$

Employing these relations we can write

$$\langle f_h, g_h\rangle = \boldsymbol{f}^{\mathrm{T}}(\mathbf{1}\otimes W)\boldsymbol{g} = \bar{\boldsymbol{f}}^{\mathrm{T}}(\mathbf{1}\otimes S)\bar{\boldsymbol{g}} \tag{11}$$

and

$$F = TB^{\mathrm{T}}W. \tag{12}$$

Furthermore, we note that

$$M_{ij} := \int_{-h/2}^{h/2} p_i\left(\frac{2}{h}x\right)\partial_x p_j\left(\frac{2}{h}x\right)\,\mathrm{d}x \tag{13a}$$
$$R_{ij} := p_i(1)p_j(1) = 1 = R^{\mathrm{T}}_{ij} \tag{13b}$$
$$L_{ij} := p_i(-1)p_j(-1) = (-1)^{i+j} = L^{\mathrm{T}}_{ij} \tag{13c}$$
$$RL_{ij} := p_i(1)p_j(-1) = (-1)^{j} \tag{13d}$$
$$LR_{ij} := p_i(-1)p_j(1) = (-1)^{i} = RL^{\mathrm{T}}_{ij}. \tag{13e}$$

In order to compute the elements of M_{ij} we first note that $M_{ij} = 0$ for $i > j-1$ as $\partial_x p_j(x)$ is a polynomial of degree $j-1$. Then we use integration by parts to show that

$$(M+L) = (R-M)^{\mathrm{T}}. \tag{14}$$

Therefore, we conclude that $M_{ij} = 1-(-1)^{i+j}$ for $i \leq (j-1)$.

We introduce the notation (10) and (13) mainly for ease of implementation. If a block-matrix class is written and the operations $+$, $-$ and $*$ are defined on it, the assembly of the derivative matrices is simplified to a large extent.

1.2 Indefinite Integrals

The approximation of an indefinite integral $\int_a^x f_h(x')\,\mathrm{d}x'$ is a bit more involved since the integration boundary x does not need to coincide with a cell boundary. First, integrating the Legendre differential equation $\frac{\mathrm{d}}{\mathrm{d}x}\left[(1-x^2)\frac{\mathrm{d}}{\mathrm{d}x}p_j(x)\right] + j(j+1)p_j(x) = 0$ and with the help of Eq. (1) and a second recursion for Legendre-Polynomials $(x^2-1)\mathrm{d}p_j(x)/\,\mathrm{d}x = jxp_j(x) - jp_{j-1}(x)$ we compute

$$\int_{-1}^{x} p_0(x')\,\mathrm{d}x' = p_1(x) + p_0(x) \qquad \int_{-1}^{x} p_j(x')\,\mathrm{d}x' = \frac{p_{j+1}(x) - p_{j-1}(x)}{2j+1} \quad \forall\, j > 0 \tag{15}$$

Notice that the integral of a polynomial of order P yields a polynomial of order $P+1$. The projection integral onto a base polynomial $p_i(x)$ yields

$$N_{00} := \int_{-1}^{1} p_0(x)\int_{-1}^{x} p_0(x')\,\mathrm{d}x'\,\mathrm{d}x = 2$$
$$N_{ij} := \int_{-1}^{1} p_i(x)\int_{-1}^{x} p_j(x')\,\mathrm{d}x'\,\mathrm{d}x = \frac{2}{(2i+1)(2j+1)}\left[\delta_{i(j+1)} - \delta_{i(j-1)}\right] \quad i,j \neq 0,0 \tag{16}$$

Our idea is to split the integral $F_h(x) = \int_a^x f_h(x')\,\mathrm{d}x'$ into two parts:

$$F_h(x) := \int_a^x f_h(x')\,\mathrm{d}x' = \int_a^{x_{n-1/2}} f_h(x')\,\mathrm{d}x' + \int_{x_{n-1/2}}^x f_h(x')\,\mathrm{d}x' \tag{17}$$

where n is the cell number such that $x_{n-1/2} \leq x < x_{n+1/2}$. Inserting $f_h(x) = \bar{f}_{nj}p_{nj}(x)$ and projecting onto a base polynomial $p_{ni}(x)$ yields

$$\bar{F}_{nk} := T_{ki}\int_{x_{n-1/2}}^{x_{n+1/2}} p_{ni}(x)\int_a^x f_h(x')\,\mathrm{d}x'\,\mathrm{d}x = \delta_{k0}\sum_{m=1}^{n-1} h\bar{f}_{m0} + \frac{h^2}{4}T_{ki}N_{ij}\bar{f}_{nj} \tag{18}$$

where we choose the proper normalization $h^2/4$ due to the double integrals in N_{ij}. Notice that if $f_h(x)$ is discretized using P polynomial coefficients $F_h(x)$ would in principle need $P+1$ coefficients to be the exact integral. However, we usually truncate the coefficients at P to match the order of the discretization. Numerical tests show that the corresponding truncation error is extremely small.

1.3 Discretization of first derivatives

From here on we write $f_h(x) = \bar{f}^{ni}p_{ni}(x)$ and imply the summation over cell index n and polynomial index i. The first naive idea to get an approximation to the first derivative of $f_h(x)$ is to simply set $f_x(x) = \partial_x f_h(x) = \bar{f}^{ni}\partial_x p_{ni}(x)$ in the interior of each cell n. Unfortunately, in this approach we loose one polynomial order and the discretization for $P = 1$ is plain wrong. We would like our discretization to become finite differences in the limit $P = 1$.

On cell boundaries the derivative of $p_{ni}(x)$ is actually not well defined, which is why we now retain to a weak formulation of derivatives. Consider

$$\int_{C_n} \partial_x f_h(x)p_{ni}(x)\,\mathrm{d}x = f_h p_{ni}|_{x_{n-1/2}}^{x_{n+1/2}} - \int_{C_n} f_h(x)\partial_x p_{ni}(x)\,\mathrm{d}x. \tag{19}$$

The approximation $f_h(x)$ is double valued on the cell boundaries, which is why we replace the boundary terms by

$$\int_{C_n} f_x p_{ni}(x)\,\mathrm{d}x = \hat{f}p_{ni}|_{x_{n-1/2}}^{x_{n+1/2}} - \int_{C_n} \bar{f}^{nk}p_{nk}\partial_x p_{ni}\,\mathrm{d}x, \tag{20}$$

where $\hat{f}(x)$ is the numerical flux across cell boundaries and we call $f_x(x)$ the numerical approximation to the first derivative. We will use three different fluxes in this work.

$$\hat{f_C}(x) = \tfrac{1}{2}(\lim_{\varepsilon\to 0,\varepsilon>0} f_h(x+\varepsilon) + \lim_{\varepsilon\to 0,\varepsilon>0} f_h(x-\varepsilon)), \tag{21a}$$

$$\hat{f_F}(x) = \lim_{\varepsilon\to 0,\varepsilon>0} f_h(x+\varepsilon), \tag{21b}$$

$$\hat{f_B}(x) = \lim_{\varepsilon\to 0,\varepsilon>0} f_h(x-\varepsilon), \tag{21c}$$

which we call the centered, the forward and the backward flux respectively. For $f\colon [a,b] \to \mathbb{R}$ and periodic boundary conditions, we assume that

$$\lim_{\varepsilon\to 0,\varepsilon>0} f(b+\varepsilon) = \lim_{\varepsilon\to 0,\varepsilon>0} f(a+\varepsilon), \qquad \lim_{\varepsilon\to 0,\varepsilon>0} f(a-\varepsilon) = \lim_{\varepsilon\to 0,\varepsilon>0} f(b-\varepsilon), \tag{22}$$

for homogeneous Dirichlet boundary conditions we assume that

$$\hat{f}(a) = \hat{f}(b) = 0, \tag{23}$$

and for homogeneous Neumann boundaries we assume that

$$\hat{f}(a) = \lim_{\varepsilon\to 0,\varepsilon>0} f_h(a+\varepsilon), \qquad \hat{f}(b) = \lim_{\varepsilon\to 0,\varepsilon>0} f_h(b-\varepsilon). \tag{24}$$

As we see the choice of $\hat{f}$ is not unique. It actually is the crucial ingredient in every dG method. Depending on what flux we choose, we arrive at various approximations to the derivative, e.g. for $P = 1$ (i.e. a piecewise constant approximation in each cell) our scheme reduces to the classic centered, forward and backward finite difference schemes respectively. In a way all the ingenuity of a dG method lies in the choice of the numerical flux.

For the following discussion we choose the centered flux $\hat{f}_C$ and note that the derivation is analogous for $\hat{f}_F$ and $\hat{f}_B$. We arrive at

$$\begin{aligned} \bar{f}^{ni}_x = T^{ij} \Big[& \frac{1}{2}\left(\bar{f}^{(n+1)k} p_k(-1) + \bar{f}^{nk} p_k(1)\right) p_j(1) \\ & - \frac{1}{2}\left(\bar{f}^{nk} p_k(-1) + \bar{f}^{(n-1)k} p_k(1)\right) p_j(-1) - \bar{f}^{nk} M_{kj} \Big] \end{aligned} \tag{25}$$

where we used that $p_{nk}(x_{n+1/2}) \equiv p_k(1)$ and $p_{nk}(x_{n-1/2}) \equiv p_k(-1)$ holds true for all n. Together with the previously defined quantities in (13) we can write

$$\begin{aligned} \bar{\mathbf{f}}_x &= (\mathbf{1}\otimes T)\circ\left[\frac{1}{2}(\mathbf{1}^+\otimes RL + \mathbf{1}\otimes(M-M^{\mathrm{T}}) - \mathbf{1}^-\otimes LR)\right]\bar{\mathbf{f}} \\ &=: (\mathbf{1}\otimes T)\circ \bar{D}^0_{x,per}\bar{\mathbf{f}}, \end{aligned} \tag{26}$$

using $M + M^{\mathrm{T}} = R - L$ from Eq. (14) and

$$\mathbf{1}^- f^n := f^{n-1} \tag{27}$$

$$\mathbf{1}^+ f^n := f^{n+1}. \tag{28}$$

We define

$$D^0_{x,per} := (\mathbf{1}\otimes F^T)\circ \bar{D}^0_{x,per}\circ(\mathbf{1}\otimes F).$$

If our coefficients are given in X-space, we note with the help of Eq. (12)

$$\mathbf{f}_x = (\mathbf{1}\otimes V)\circ D^0_{x,per}\mathbf{f}, \tag{29}$$

where $\bar{D}^{per}_x$ and D^{per}_x are skew-symmetric matrices.

From Eq. (26) we are now able to show the matrix representation of the one-dimensional discrete derivative for periodic boundary conditions that can be used in the implementation

$$\bar{D}^0_{x,per} = \frac{1}{2}\begin{pmatrix} (M-M^{\mathrm{T}}) & RL & & & -LR \\ -LR & (M-M^{\mathrm{T}}) & RL & & \\ & -LR & \dots & & \\ & & & \dots & RL \\ RL & & & -LR & (M-M^{\mathrm{T}}) \end{pmatrix} \tag{30}$$

We also write down the expressions resulting from the forward and backward fluxes $\hat{f}_F$ and $\hat{f}_B$ respectively:

$$\bar{D}^+_{x,per} = \begin{pmatrix} -(M+L)^{\mathrm{T}} & RL & & & 0 \\ 0 & -(M+L)^{\mathrm{T}} & RL & & \\ & 0 & \dots & & \\ & & & \dots & RL \\ RL & & & 0 & -(M+L)^{\mathrm{T}} \end{pmatrix} \tag{31}$$

and

$$\bar{D}^-_{x,per} = \begin{pmatrix} (M+L) & 0 & & & -LR \\ -LR & (M+L) & 0 & & \\ & -LR & \dots & & \\ & & & \dots & 0 \\ 0 & & & -LR & (M+L) \end{pmatrix}. \tag{32}$$

Note that for $P = 1$ we recover the familiar finite difference approximations of the first derivative.

	D^+_x (forward)		D^-_x (backward)		D^0_x (centered)	
	left	right	left	right	left	right
periodic	$-(M+L)^{\mathrm{T}}$	$-(M+L)^{\mathrm{T}}$	$(M+L)$	$(M+L)$	$\frac{1}{2}(M-M^{\mathrm{T}})$	$\frac{1}{2}(M-M^{\mathrm{T}})$
Dirichlet	$-M^{\mathrm{T}}$	$-(M+L)^{\mathrm{T}}$	$(M+L)$	$-M^{\mathrm{T}}$	$\frac{1}{2}(M-M^{\mathrm{T}}+L)$	$\frac{1}{2}(M-M^{\mathrm{T}}-R)$
Neumann	$-(M+L)^{\mathrm{T}}$	M	M	$(M+L)$	$\frac{1}{2}(M-M^{\mathrm{T}}-L)$	$\frac{1}{2}(M-M^{\mathrm{T}}+R)$

Table 1: Upper left and lower right matrix entries for various boundary conditions. For Dirichlet and von Neumann BC the upper right and lower left entries are zero.

Finally we note the boundary terms for homogeneous Dirichlet and Neumann boundaries in Table (1) noticing that only the corner entries of the matrices change.

In our notation the local character of the dG method is apparent. To compute the derivative in one cell we only use values of neighboring cells. Therefore, the method is well suited for parallelization which we will exploit in our implementation.

The generalization to higher dimensions is immediate. All the matrices derived above can readily be extended via the appropriate Kronecker products. The space complexity of the matrices derived is $\mathcal{O}(P^2N)$ in one and $\mathcal{O}(P^3N^2)$ in two dimensions.

Finally, let us remark that we found it practical to always operate on coefficients in X-space, i.e. we use Eq. (29) for our implementations and thus use $\boldsymbol{f}$ rather than $\bar{\boldsymbol{f}}$ to represent the approximation. Function products are easily computed coefficient-wise in X-space, i.e. we use

$$(fg)_{ni} = f_{ni} g_{ni} \tag{33}$$

to represent the corresponding products.

2 Discretization of elliptic equations

We are now ready to discretize the one-dimensional general elliptic equation

$$-\frac{\partial}{\partial x}\left(\chi(x)\frac{\partial\phi}{\partial x}(x)\right) = \rho(x) \tag{34}$$

on the interval $[a,b]$. Here, $\chi(x)$ and $\rho(x)$ are given functions.

We either choose periodic, Dirichlet, or Von-Neumann boundary conditions on the left and right border for ϕ. As a first step we rewrite Eq. (34) into two first order differential equations and a function product:

$$j' = \partial_x\phi, \tag{35a}$$
$$j = \chi j', \tag{35b}$$
$$\rho = -\partial_x j. \tag{35c}$$

Our plan is to simply use one of the discretizations developed in the last section for the first equation (35a) and its negative adjoint for the third equation (35c). Recall that the adjoint of a square matrix A is defined by the scalar product, i.e.

$$\boldsymbol{f}^{\mathrm{T}}(\mathbf{1}\otimes W)\circ A\boldsymbol{g} = \boldsymbol{g}^{\mathrm{T}}(A^{\mathrm{T}}\circ(1\otimes W))\boldsymbol{f} =: \boldsymbol{g}^{\mathrm{T}}(\mathbf{1}\otimes W)A^{\dagger}\boldsymbol{f}.$$

From here we immediately get the relation

$$A^{\dagger} \equiv (\mathbf{1}\otimes V)\circ A^{\mathrm{T}}\circ(\mathbf{1}\otimes W). \tag{36}$$

There is a close connection between symmetric, $A = A^{\mathrm{T}}$, and self-adjoint, $A = A^{\dagger}$, matrices. If and only if the matrix A is symmetric, then $(\mathbf{1}\otimes V)\circ A$ is self-adjoint. Of course, we have $(AB)^{\dagger} = B^{\dagger}A^{\dagger}$ and $(A^{\dagger})^{\dagger} = A$.

The function product in Eq. (35b) is computed pointwisely on the Gaussian abscissas.

$$\begin{aligned}\boldsymbol{j}' &= (\mathbf{1}\otimes V)\circ D_x\boldsymbol{\phi},\\ \boldsymbol{j} &= \boldsymbol{\chi}\boldsymbol{j}',\\ \boldsymbol{\rho} &= (\mathbf{1}\otimes V)\circ D_x^{\mathrm{T}}\boldsymbol{j} = (\mathbf{1}\otimes V)D_x^{\mathrm{T}}\circ\chi\circ(\mathbf{1}\otimes V)\circ D_x\boldsymbol{\phi},\end{aligned} \tag{37}$$

where D_x is either D_x^+, D_x^-, or D_x^0 with the correct boundary terms. Note, that [5] originally only proposed to use the forward or backward discretization for D_x. Equation (37) is indeed a self-adjoint discretization for the second derivative. However, it turns out that Eq. (37) is inconsistent for given $\rho(x)$. The solution does not converge. This problem is solved according to [5] by adding a jump term to the flux of j:

$$\hat{j}(x_{n+1/2}) \to \hat{j}(x_{n+1/2}) + [\phi(x_{n+1/2})], \tag{38}$$

where $[\phi(x_{n+1/2}] := \phi^n(x_{n+1/2}) - \phi^{n+1}(x_{n+1/2})$ is the jump term of ϕ at $x_{n+1/2}$. That means we have to alter our discretization (37) according to

$$\rho = (\mathbf{1} \otimes V)[D_x^{\mathrm{T}} \circ \chi \circ (\mathbf{1} \otimes V) \circ D_x + J]\phi \tag{39}$$

where

$$\bar{J} = \begin{pmatrix} (L+R) & -RL & & & -LR \\ -LR & (L+R) & -RL & & \\ & -LR & \dots & & \\ & & & \dots & -RL \\ -RL & & & -LR & (L+R) \end{pmatrix}. \tag{40}$$

for periodic boundaries. Again we give the correct boundary terms for Dirichlet and Neumann boundary conditions in Table 2.

	left	right
periodic	$L+R$	$L+R$
Dirichlet	$L+R$	$L+R$
Neumann	R	L

Table 2: Top left and bottom right entries for jump matrix

Note that J is symmetric, thus the overall discretization remains self-adjoint. Indeed, with $D_x = D_x^+$ Eq. (39) recovers the discretization proposed by [5]. In addition, we remark that the centered discretization in Eq. (39) is symmetric with respect to an inversion of the coordinate system $x \to -x$ even for double Dirichlet or Neumann boundaries, while the forward and backward discretization is not.

We note again that the generalization to two or more dimensions is straightforward in a rectangular grid using Kronecker products.

2.1 Numerical experiments

As an example we solve Eq. (34) in two dimensions for

$$\chi(x,y) = 1 + \sin(x)\sin(y)$$
$$\rho(x,y) = 2\sin(x)\sin(y)\left[\sin(x)\sin(y) + 1\right] - \sin^2(x)\cos^2(y) - \cos^2(x)\sin^2(y)$$

on the domain $D = [0,\pi] \times [0,\pi]$ for double Dirichlet boundary conditions. The analytical solution is given by $\phi(x,y) = \sin(x)\sin(y)$.

Do not try to insert ϕ into Eq. (39) directly. The result sub-optimally converges to the analytical ρ or may not converge at all, a phenomenon that is called supraconvergence. Only as a discretization for elliptic or parabolic equations the stencil (39) works fine.

We note that Eq. (39), when multiplied by $(\mathbf{1} \otimes W)$, has the form of a symmetric matrix equation $A\boldsymbol{x} = \boldsymbol{b}$. This equation is solved by a conjugate gradient method with $\mathbf{1} \otimes V$ as a diagonal preconditioner. We use a truncation criterion based on the norm of the residuum

$$||\boldsymbol{r}_k|| < \varepsilon_{res}||\boldsymbol{b}||_{L_2} + \varepsilon_{res} \tag{41}$$

where $\boldsymbol{r}_k = A\boldsymbol{x}_k - \boldsymbol{b}$ is the residuum of the $k-th$ iteration. In Table 3 we summarize our results.

		Forward		Backward			Centered		
# of cells	ε_{res}	iterations	L^2 error	iterations	L^2 error	Order	iterations	L^2 error	Order
				$P=1$					
17^2	1.0E-04	33	1.40E-01	33	1.40E-01	-	13	1.10E-01	-
34^2	1.0E-05	78	7.50E-02	78	7.50E-02	0.90	25	6.17E-02	0.83
68^2	1.0E-06	175	3.87E-02	175	3.87E-02	0.95	54	3.29E-02	0.91
136^2	1.0E-07	396	1.97E-02	396	1.97E-02	0.98	124	1.70E-02	0.95
				$P=2$					
17^2	1.0E-05	102	2.46E-03	102	2.46E-03	-	47	4.10E-03	-
34^2	1.0E-06	226	5.93E-04	226	5.93E-04	2.05	114	1.10E-03	1.90
68^2	1.0E-07	485	1.46E-04	485	1.46E-04	2.02	259	2.86E-04	1.94
136^2	1.0E-08	1052	3.64E-05	1052	3.64E-05	2.01	580	7.30E-05	1.97
				$P=3$					
17^2	1.0E-06	181	4.77E-05	181	4.77E-05	-	113	5.37E-06	-
34^2	1.0E-07	403	5.22E-06	403	5.22E-06	3.19	259	3.67E-07	3.87
68^2	1.0E-08	893	5.93E-07	892	5.93E-07	3.14	583	2.64E-08	3.80
136^2	1.0E-09	1946	6.97E-08	1946	6.97E-08	3.09	1277	1.92E-09	3.78
				$P=4$					
17^2	1.0E-08	357	4.62E-07	357	4.62E-07	-	221	7.60E-07	-
34^2	1.0E-09	793	2.47E-08	795	2.47E-08	4.22	498	5.54E-08	3.78
68^2	1.0E-09	1637	1.48E-09	1637	1.48E-09	4.06	1035	3.80E-09	3.87
136^2	1.0E-10	3505	9.13E-11	3505	9.13E-11	4.02	2223	2.49E-10	3.93
				$P=5$					
17^2	1.0E-09	581	1.57E-08	580	1.57E-08	-	354	2.16E-09	-
34^2	1.0E-10	1277	3.62E-10	1277	3.62E-10	5.44	782	3.51E-11	5.95
68^2	1.0E-11	2751	8.39E-12	2752	8.39E-12	5.43	1697	6.68E-13	5.71
136^2	1.0E-12	5816	2.03E-13	5816	2.03E-13	5.37	3597	4.01E-14	4.06

Table 3: Accuracy for the dG method proposed for various number of cells N and polynomial degrees in each cell P.

We observe that the forward and backward discretizations give the same results, which is due to the symmetry of the sine functions. The order of the relative error in the L_2-norm coincides

with the predicted convergence of order P for all P.

On the other side, the centered discretization needs significantly less iterations to achieve the same error in the residuum. This indicates that the centered discretization is better conditioned than the forward and backward discretizations. Also, we observe a superconvergent error of order $P+1$ for $P=3$ and $P=5$. In order to exclude symmetry reasons for these phenomena, we repeated the computations on the domain $D=[0,\pi/2]\times[0,\pi/2]$ using Dirichlet boundaries on the left and Neumann boundaries on the right side. We find equal results; only the equality in the results for forward and backward discretization is broken. To the knowledge of the author this superconvergence has not yet been observed before.

3 Interpolation and Projection

Eq. (7) provides a natural way to interpolate any dg expanded function $f_h(x)$ on any point in the interval $[a,b]$. The interpolation has the same order P as the expansion and can be formulated as a matrix vector multiplication

$$f_h(x_0) = \sum_{n=1}^{N}\sum_{k=0}^{P-1} p_{nk}(x_0)\bar{f}^{nk} =: \sum_{n=1}^{N}\sum_{k=0}^{P-1} I_{nk}\bar{f}^{nk} \tag{42}$$

where the sparse interpolation matrix I has as many rows as there are points to interpolate. I has P^d entries per line, where d is the dimensionality of the grid.

3.1 Interpolation and Projection

A special case emerges when the list of points to interpolate is made up by the Gaussian abscissas of another grid. Suppose we want to divide each cell C_n of the original grid into M equidistant subcells. We use the letter c to denote the coarse grid and the letter f to denote the fine grid. We denote $q_{ml}(x)$ the polynomials on the fine grid and x^F_{mj} the corresponding Gaussian abscissas. If a vector $\boldsymbol{f}$ is given on the coarse grid, we can simply interpolate it onto the fine grid analogous to Eq. (42) via

$$f^F_{mj} = f_h(x^F_{mj}) = p_{nk}(x^F_{mj})F^{ki}f^C_{ni} =: {Q_{mj}}^{ni} f^C_{ni} \tag{43}$$

where we denote the special interpolation matrix with Q and implicitly assume the sum of repeated indices. No information is lost in this process if the cells C_n are divided by an integer number M and the number of polynomials is the same, i.e. f^F and f^C represent exactly the same expansion $f_h(x)$.

$$f^C(x) = \bar{f}^{nk}_C p_{nk}(x) = f^F(x) = \bar{f}^{ml}_F q_{ml}(x) \tag{44}$$

Vice versa, given an expansion $f^F(x)$ on the fine grid, we can compute the projection integrals from the fine grid to the coarse grid. It can be shown that

$$\bar{f}^{nk}_C := T^{ks}_C \int \mathrm{d}x f^F(x) p_{ns}(x) = T^{ks}_C W^{ij}_m p_{ns}(x_{mj}) f^F_{mi} \tag{45}$$

$$f^C_{nt} = B_{tk}\bar{f}^{nk}_C = V^c_{tk}{(Q^T)^{nk}}_{mj} W^{ji}_m f^F_{mi} =: {P_{nt}}^{mi} f^F_{mi} \tag{46}$$

from where we directly conclude that

$$P = Q^\dagger = V^C Q^T W_F \tag{47}$$

i.e. the projection matrix is the adjoint of the interpolation matrix. We can also proof that

$$P \circ Q = 1_C \tag{48}$$

which is a reformulation of Eq. (44). Note that $Q \circ P \neq 1$. The projection is not loss-free but it conserves the integral value of the function on the fine grid and the error in the L_2-norm between the vectors on fine and coarse grid is minimal (proof?).

3.2 Grid transformations

The above transformation Eq. (47) is valid only if the number of cells in the fine grid is an integer multiple of the number of cells in the coarse grid. If this is not the case, the adjoint of the interpolation matrix does not(!) give a valid projection matrix (tried this out in the code). However, what we can always do is to find the least common multiple grid (c) of two given grids (a) and (b). Then we can interpolate loss-free from the given grid (a) to the common grid (c) and then project back from (c) to (b). In this way we get the forward and backward transformation matrices

$$\mathcal{T} = P_{c2b} Q_{a2c} \tag{49}$$

$$\mathcal{T}^\dagger = P_{c2a} Q_{b2c} \tag{50}$$

If the least common grid reduces to either grid (a) or (b) the transformation matrix reduces to either P_{a2b} or Q_{a2b} as expected.

4 Spectral and modal filtering techniques

It is well known from Godunov's theorem that linear high order schemes for hyperbolic (advection) problems are prone to oscillations. Discontinuous Galerkin methods are no different in that regard especially the higher order ones and if the numerical flux $f = nu$ is computed via pointwise multiplication of the nodal representations of n and u via $f_{ni} = n_{ni} u_{ni}$ [6]. This is due to aliasing effects. Recall that pointwise multiplication is different from actually multiplying the two polynomials n_h and u_h and projecting the result back onto the lower polynomial space. The problem is also known in pure spectral methods like Fourier representations.

In finite difference and Fourier codes what is often done is adding so-called hyperdiffusion (or **spectral viscosity**) to stabilize a purely hyperbolic system.

$$\partial_t n + \partial_x (nu) = \nu (-1)^{s+1} \partial_x^{2s} n \tag{51}$$

Since Fourier modes are Eigenfunctions of the Laplace operator this equation reads in Fourier space (disregaring the convection term)

$$\partial_t n_k = -\nu k^{2s} n_k \tag{52}$$

which when discretized in time using the Euler method reads

$$n_k^{n+1} = (1 - \nu\Delta t k^{2s}) n_k^n \tag{53}$$

Here we see that the damping of modes is highest for large wave-numbers and leaves small wave-numbers intact. This can be read in a more general setting as applying a low-pass filter $\sigma(k)$ to the solution that acts on large wavenumbers

$$n_k^{n+1} = \sigma(k) n_k^n \tag{54}$$

where often $\sigma(k) = \exp(-\nu\Delta t k^{2s})$ is proposed. The above Euler discretization is then obtained by Taylor-expanding the exponential (an implicit Euler is obtained with the Pade approximation of the expontential).

It is now possible to also discretise the hyperdiffusion directly in a dG framework, however, this has downsides regarding the performance since the Laplace operator is much more expensive than a simple pointwise multiplication as done in Fourier codes and its solution likely needs to be done implicitly in order to preserve the time-step restrictions.

The better solution in a dG framework is to apply **modal filtering** and is motivated in much the same way as the spectral viscosity above. First we consider that the Legendre polynomials satisfy a Sturm-Liouville type differential equation.

$$\frac{d}{\mathrm{d}x}\left[(1-x^2)\frac{d}{dx}\right] p_n(x) = -n(n+1)p_n(x) \quad x \in [-1,1] \tag{55}$$

This means that the Legendre polynomials are Eigenfunctions of the Sturm-Liouville operator. If we write

$$\partial_t n = \nu(-1)^{s+1}\left[\frac{d}{\mathrm{d}x} m(x) \frac{d}{dx}\right]^s n \tag{56}$$

where $m(x)$ is $(1-x^2)$ properly scaled onto each discrete cell, and use a dG approach to discretize this equation we get for each mode

$$\partial_t n_k = -\nu(k(k+1))^s n_k \tag{57}$$

or in a time-discreized way

$$n_k^{n+1} = (1 - \nu\Delta t(k(k+1))^s) n_k^n \tag{58}$$

which is entirely analogous to the derivation above. In the same way we can read this as applying a filter function to the Legendre modes.

$$n_k^{n+1} = \sigma(k) n_k^n \tag{59}$$

with $\sigma(k) = \exp(-\nu\Delta t(k(k+1))^s)$. Note that the modal filter, as the spectral filter, does not inhibit oscillations from happening (it is not shock-capturing) but it dampens grid-scale oscillations and can stabilize the system.

References

[1] M. Abramowitz and I. Stegun. *Handbook of mathematical functions*. Dover Publishing Inc. New York, 10th edition, 1972. URL http://people.math.sfu.ca/~cbm/aands/.

[2] D. Arnold, F. Brezzi, B. Cockburn, and L. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779, 2002. doi:10.1137/S0036142901384162.

[3] B. Cockburn and C. W. Shu. The local discontinuous galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.*, 35(6):2440–2463, 1998. doi:10.1137/S0036142997316712.

[4] B. Cockburn and C. W. Shu. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comput.*, 16(3):173–261, 2001. doi:10.1023/a:1012873910884.

[5] B. Cockburn, G. Kanschat, I. Perugia, and D. Schötzau. Superconvergence of the Local Discontinuous Galerkin Method for Elliptic Problems on Cartesian Grids. *SIAM J. Numer. Anal.*, 39(1):264–285, 2002. doi:10.1137/S0036142900371544.

[6] J. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Texts in Applied Mathematics. Springer Media, 2008.

[7] M. Wiesenberger. *Gyrofluid computations of filament dynamics in tokamak scrape-off layers*. PhD thesis, University of Innsbruck, 2014. URL http://resolver.obvsg.at/urn:nbn:at:at-ubi:1-1799.

[8] S. Yadav, A. Pani, and E. Park. Superconvergent discontinuous galerkin methods for nonlinear elliptic equations. *Math. Comput.*, 82(283):1297–1335, 2013. doi:10.1090/S0025-5718-2013-02662-2.