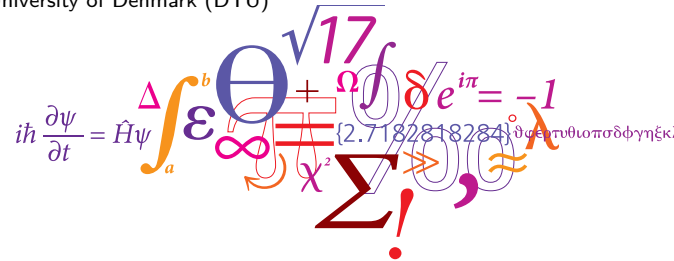
Introduction to discontinuous Galerkin methods

Matthias Wiesenberger

Plasma Physics and Fusion Energy, Technical University of Denmark (DTU)



DTU Physics
Department of Physics

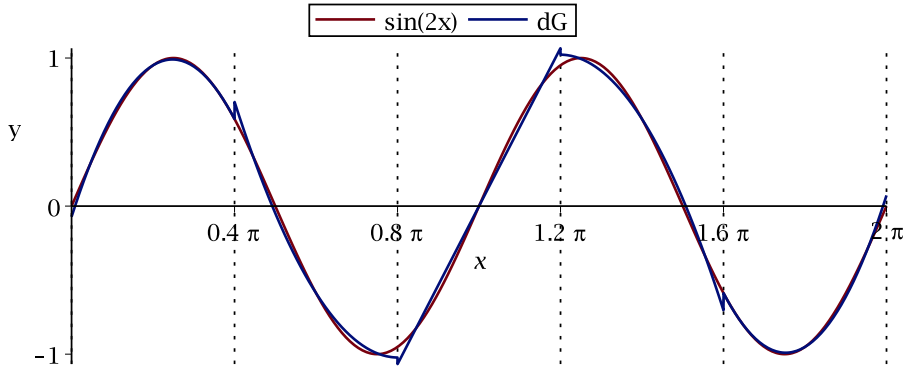
Outline

- Introduction
- Numerical methods
  - Integration
  - Derivatives
  - Local discontinuous Galerkin
- Numerical Experiments
- Conclusions

Discontinuous Galerkin (dG) methods

- are a whole class of methods e.g. Runge Kutta dG (RKDG), local dG (LDG), interior penalty methods, or hybridized dG ($\rightarrow$ G. Giorgiani's talk)
- combine advantages of finite element methods (ease of handling difficult geometry), finite volume (flux conservation) and finite differences (absence of global mass matrix, easily parallelisable)
- 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
  Review article, covers RKDG schemes for advection type problems including e.g shock capturing schemes (slope limiters), as well as LDG for parabolic and elliptic equations
- Notation unfortunately horrible (overwhelming generality evades practicality)
- This presentation: `https://feltor-dev.github.io/doc/dg/html/dg_introduction.pdf`

A polynomial in each cell



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

A polynomial in each cell

$$p_{nk}(x) := \begin{cases} p_k\left(\frac{2}{h}(x - x_n)\right), & \text{for } x - x_n \in \left[-\frac{h}{2}, \frac{h}{2}\right] \\ 0, & \text{else.} \end{cases} \tag{1}$$

- p_k are Legendre polynomials (in this work at least)
- orthogonal and complete
- P is the number of base polynomials per cell (typically 3 or 4)
- In each cell we have X-space and L-space (x_j^a and w_j are abscissas and weights of Gauss-Legendre quadrature)

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

$$f_j = \sum_{k=0}^{P-1} B_{jk} \bar{f}^k \quad B_{jk} := p_k(x_j^a) \tag{2b}$$

Interpolation and Integration

We directly have an order P interpolation formula (just evaluate the polynomials at x)

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

Integrating polynomials yields

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

equals Gauss-Legendre integration (of order $2P-1$). Note that its implementation is a reduction.

A first derivative

A first try (imply summation over repeated indices)

$$\partial_x f_h(x) = \bar{f}^{ni} \partial_x p_{ni}(x) \quad ??$$

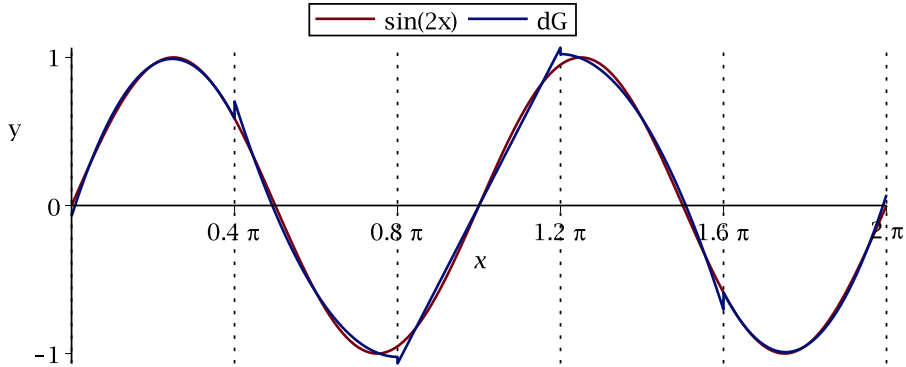
Loose one order and completely wrong for $P = 1$

Weak formulation (project with base functions, integrate by part)

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

Remember $f_h(x)$ is double valued on cell-boundaries !?

Double valued on edges



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

Numerical fluxes

Freely choose the flux $f_h \to \hat{f}$ on cell boundaries.

The flux $\hat{f}$ is what defines a dG method!

$$\hat{f_C}(x) = \frac{1}{2}\left(\lim_{\varepsilon \to 0, \varepsilon > 0} f_h(x+\varepsilon) + \lim_{\varepsilon \to 0, \varepsilon > 0} f_h(x-\varepsilon)\right), \text{ centered} \tag{3a}$$

$$\hat{f_F}(x) = \lim_{\varepsilon \to 0, \varepsilon > 0} f_h(x+\varepsilon), \text{ forward} \tag{3b}$$

$$\hat{f_B}(x) = \lim_{\varepsilon \to 0, \varepsilon > 0} f_h(x-\varepsilon), \text{ backward} \tag{3c}$$

There are many more (e.g. slope limiters, Godunov flux, Lax-Friedrich ...)

Centered derivative

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

- M and RL are $P \times P$ block matrices.
- Leads to familiar finite difference formulas for $P = 1$!

Local discontinuous Galerkin

$$-\frac{\partial}{\partial x}\left(\chi(x)\frac{\partial \phi(x)}{\partial x}\right) = \rho(x) \tag{4}$$

Split into first order equations

$$j' = \partial_x \phi, \tag{5a}$$
$$j = \chi j', \tag{5b}$$
$$\rho = -\partial_x j. \tag{5c}$$

- Apply same procedure on each equation
- Choose fluxes such that $\hat{\phi}$ is independent of j_h, but $\hat{j}$ might depend on ϕ_h to penalize the jumps between cells (local dG) $\rightarrow \rho = D_x^{\mathrm{T}} \chi D_x + J$
- Self-adjoint (symmetric) discretization easily invertible with a (preconditioned) conjugate gradient method

Local discontinuous Galerkin

On structured grids the discretization is immediately extensible to higher dimensions:

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

Solve on $[0,\pi]\times[0,\pi]$ with Dirichlet boundary condition

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

		Forward		Backward			Centered		
# of cells	ε_{res}	iterations	L^2 error	iterations	L^2 error	Order	iterations	L^2 error	Order
				$P = 3$					
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

Experiences with dG

- In practice it is easier to work in configuration space than with the polynomial coefficients
- Well parallelizable (we have implementations for OpenMP, Cuda, MPI, MPI+OpenMP as well as MPI+Cuda)
- A higher order method does not automatically lead to a more accurate solution (in practice $P = 3$ or $P = 4$ is absolutely sufficient).
- Beware of supraconvergence (inversion of $\partial_x^2 \phi = \rho$ converges, but $\partial_x^2 \phi$ does not necessarily converge on its own)
- LDG quite robust; works well for elliptic equations of the form $\sum_{ij} \partial_i \left(\chi^{ij} \partial_j \phi \right) = \rho$ with χ positive (semi-)definite and also $(1 + \xi \Delta)\phi = \rho$