Tensor-Product Preconditioner For Very High Order Discontinuous Galerkin Discretizations

Tom Feldhausen

December 4, 2025

Abstract

Solving a multi-dimensional PDE by means of a Discontinuous Galerkin method in a higher order solution space requires the solution of large linear systems during time integration. When storing the semi-discrete system, communication dominates the computational costs such that matrix-free implementations turn out to be more efficient in practice. In order to efficiently integrate implicit time integration into this framework, we need matrix-free preconditioners. This project aims to compare the tensor-product preconditioners of Diosady [DM17] and Pazner [PP18] for the example of linear advection and provide an optimized C++ deal.II [ABB$^+$24] implemenation. As a first step, we provide a basic matrix-based DG implementation of two-dimensional linear advection with upwind flux.

The Discontinuous Galerkin For Two-Dimensional Linear Advection

As a first step, we provide a matrix-based DG discretization for the two-dimensional linear advection system

$$\begin{aligned}
&\partial_t u(x,t) + a(x,t)\cdot\nabla u(x,t) = 0 \quad \forall x \in [0,1]^2, \quad 0 \le t \le T\\
&u(x,0) = u_0(x) \quad \forall x \in [0,1]^2\\
&u(x,t) = u_{\text{in}}(x) \quad \forall x \in \partial[0,1]^2
\end{aligned}$$

for a solution function u. Following Kronbichler and Persson [**?**], we arrive at the weak formulation that for all $v \in H^1(\Omega)$ it needs to hold that

$$\int_\Omega \partial_t uv - \int_\Omega ua\cdot\nabla v + \int_{\partial\Omega}(a\cdot n)uv = 0, \tag{1}$$

after integrating in space, multiplying by a test function v and integrating by parts.
Now instert basis functions of order p, i.e.

$$u = \sum_{i=0}^{p} u_i\Phi_i, v = \sum_{j=0}^{p} v_j\Phi_j.$$

Then condition (1) is equivalent to the element-wise formulation for each basis function, i.e.

$$\int_{\Omega_e} \partial_t u_i(t)\Phi_i\Phi_j - \int_{\Omega_e} u_i\Phi_i a\cdot\nabla\Phi_j + \int_{\partial\Omega_e}(a\cdot n)u_i\Phi_i\Phi_j = 0 \tag{2}$$

being true for all $0 \le i,j \le p$ and $\Omega_e \subset \Omega$.
Analogously we can write this in matrix form, which yields the system

$$\frac{dU(t)}{dt} = M^{-1}[(B+G)U + G_{\text{bound}}].$$

We consider a tensor-product mesh with $N = N_x \times N_y$ rectangular elements. Then $x : \Omega_{\text{ref}} \to \Omega_e$ maps from the reference element $\Omega_{\text{ref}} = [-1,1]^2$ to the actual element $\Omega_e = [x_0, y_0] \times [x_1, y_1]$ and can be explicitly written as

$$\mathbf{x}(\boldsymbol{\zeta}) = A\,\boldsymbol{\zeta} + \mathbf{b}, \qquad A = \frac{1}{2}\begin{pmatrix} x_1 - x_0 & 0 \\ 0 & y_1 - y_0 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} \frac{x_1+x_0}{2} \\ \frac{y_1+y_0}{2} \end{pmatrix}.$$

With this at hand, we can explicitly write out the mass matrix as

$$\begin{aligned}
M_{ji} &= \int_{\Omega_e} \Phi_j(x)\Phi_i(x)dx\\
&= \int_{x_0}^{y_0}\int_{x_1}^{y_1} \Phi_j(x_0,x_1)\Phi_i(x_0,x_1)dx_0dx_1\\
&= \int_{-1}^{1}\int_{-1}^{1} \Phi_j(\zeta_1,\zeta_2)\Phi_i(\zeta_1,\zeta_2)\|\det(J_e^{-1})\|d\zeta_1d\zeta_2.
\end{aligned}$$

Using Gauss-Legendre quadrature with one-dimensional weights w_q and points x_q, we get the approximation

$$\begin{aligned} M_{ji} = & \int_{\Omega_e} \Phi_j(x)\Phi_i(x)dx \\ \approx & \sum_{q_0}\sum_{q_1} w_{q_0}w_{q_1}\phi_{i_1}(x_{q_0})\phi_{i_2}(x_{q_1})\phi_{j_1}(x_{q_0})\phi_{j_2}(x_{q_1})\|(y_0-x_0)(y_1-x_1)\| \end{aligned}$$

when inserting the tensor-product form $\Phi_i = \phi_{i_1}\phi_{i_2}$ and exploiting $\|\det(J_e^{-1})\| = \|(y_0 - x_0)(y_1 - x_1)\|$.
Similarly, we can derive an expression for the volume matrix and calculate

$$\begin{aligned} B_{ji} = & \int_{\Omega_e} \Phi_i(a\cdot\nabla\Phi_j) \\ = & \int_{x_0}^{y_0}\int_{x_1}^{y_1} \Phi_i(z_0,z_1)(a(z_0,z_1,t)\cdot\nabla\Phi_i(z_0,z_1))dz_0dz_1 \\ = & \int_{x_0}^{y_0}\int_{x_1}^{y_1} \Phi_i(z_0,z_1)(a_0(z_0,z_1,t)\partial_{z_0}\Phi_i(z_0,z_1) + a_1(z_0,z_1,t)\partial_{z_1}\Phi_i(z_0,z_1))dz_0dz_1 \\ = & \int_{1}^{-1}\int_{1}^{-1} \Phi_i(\zeta_0,\zeta_1)(a_0(\zeta_0,\zeta_1,t)\partial_{z_0}\Phi_i(\zeta_0,\zeta_1) + a_1(\zeta_0,\zeta_1,t)\partial_{z_1}\Phi_i(\zeta_0,\zeta_1)) \\ & \|(y_0-x_0)(y_1-x_1)\|d\zeta_0d\zeta_1 \\ = & \sum_{q_0}\sum_{q_1} w_{q_0}w_{q_1}\phi_{i_1}(x_{q_0})\phi_{i_2}(x_{q_1})(a_0(x_{q_0})\phi'_{j_1}(x_{q_0}) + a_1(x_{q_0})\phi_{j_1}(x_{q_0})\phi'_{j_2}(x_{q_1}))\phi_{j_2}(x_{q_1}) \\ & \|(y_0-x_0)(y_1-x_1)\|. \end{aligned}$$

The face terms are slightly more difficult to derive as in two dimensions we have four faces that are either interior and face other elements or that are part of the domain boundary. This is why for a specific derivation, we need to choose an explicit flux and face. In this example, we choose an upwind flux for information flow between elements, i.e.

$$\widehat{au} = \begin{cases} (a\cdot n)u^-, & a\cdot n > 0, \\ (a\cdot n)u^+, & a \le 0 \end{cases}$$

and focus on the left face. As this is one-dimensional, we will only need the $x_1(\zeta)$ part of the transformation, that concerns the x_1-axis. The corresponding determinant of the inverse Jacobian is simply $(y_1 - x_1)$ for our tensor-product grid. Generally it holds that

$$\begin{aligned} & \int_{\partial\Omega_e} \widehat{au}(\Phi_i^R(z),\Phi_i^L(z))\Phi_j(z)dz \\ = & \int_{-1}^{1}(y_1-x_1)\phi_{j_0}(-1)\phi_{j_1}(\zeta)\widehat{au}(\Phi_i^R(zeta),\Phi_j^L(\zeta))d\zeta \\ \approx & \sum_q w_q(y_1-x_1)\phi_{j_0}(-1)\phi_{j_1}(x_q)\widehat{au}(u_i^R\phi_{j_0}^R(-1)\phi_{j_1}^R(x_1(x_q)), u_i^L\phi_{j_0}^L(-1)\phi_{j_1}^L(x_1(x_q))). \end{aligned}$$

The value of $\widehat{au}(u_i^R\phi_{j_0}^R(-1)\phi_{j_1}^R(x_1(x_q)), u_i^L\phi_{j_0}^L(-1)\phi_{j_1}^L(x_1(x_q)))$ depends on the following cases.

Case 1A: Inflow Boundary.

$$G_{\mathrm{bound}j} \approx \quad \sum_q w_q(y_1 - x_1)\phi_{j_0}(-1)\phi_{j_1}(x_q)a(x_0(-1), x_1(x_q), t)u(x_0(-1), x_1(x_q), t)$$

Case 1B: Outflow Boundary.

$$G_{ji^R} \approx \quad \sum_q w_q(y_1 - x_1)\phi_{j_0}(-1)\phi_{j_1}(x_q)a(x_0(-1), x_1(x_q), t)\phi^R_{i_0}(-1)\phi^R_{j_1}(x_q)u^R_i$$

Case 2A: Inflow Interior Face.

$$G_{ji^L} \approx \quad \sum_q w_q(y_1 - x_1)\phi_{j_0}(-1)\phi_{j_1}(x_q)a(x_0(-1), x_1(x_q), t)\phi^L_{i_0}(-1)\phi^L_{j_1}(x_q)u^L_i$$

Case 2B: Outflow Interior.

$$G_{ji^R} \approx \quad \sum_q w_q(y_1 - x_1)\phi_{j_0}(-1)\phi_{j_1}(x_q)a(x_0(-1), x_1(x_q), t)\phi^R_{i_0}(-1)\phi^R_{j_1}(x_q)u^R_i$$

Now that we set up the semi-discretization in space, we can now focus on time integration. We can evolve (**??**) in time by appling a Runge–Kutta method. In our experiments, we set $N_x = 20, N_y = 20$ and $p = 2$. Figure 1 shows the convergence behaviour for different explicit integrators of order 1 to 4.
For implicit DIRK schemes of order 1 to 4, we get the result illustrated in figure 2.
Combining this with the FDM integrator of Diosady yields figure 3.
Using Pazners KSVD-based integrator gives the results in figure 4.

Bibliography

[ABB+24] Daniel Arndt, Wolfgang Bangerth, Bruno Blais, Marc Fehling, Rene Gassmöller, Timo Heister, Luca Heltai, Sebastian Kinnewig, Martin Kronbichler, Matthias Maier, et al. The deal. ii library, version 9.6. *Journal of Numerical Mathematics*, 32(4):369–380, 2024.

[DM17] Laslo T Diosady and Scott M Murman. Tensor-product preconditioners for higher-order space–time discontinuous galerkin methods. *Journal of Computational Physics*, 330:296–318, 2017.

[PP18] Will Pazner and Per-Olof Persson. Approximate tensor-product preconditioners for very high order discontinuous galerkin methods. *Journal of computational physics*, 354:344–369, 2018.