Rotating Gaussian function for the tracer in dgswem: convergence key

Please take a look at this convergence test. The setup is included in fort.15 and fort.dg files. The solution is to the Gaussian blob rotating about the origin. The boundary conditions are transmissive, and the fluxes are upwinded. In primitive form the scalar transport is:

$$\partial_t \iota + \boldsymbol{u} \cdot \nabla_x \iota = 0. \tag{1}$$

The domain is $\Omega = \left[-\frac{1}{2}, \frac{1}{2}\right]^2$ with symmetric grid directionality (!!beware of uniform grids using triangles in convergence tests, rather, balance the directional bias), with velocity field $\boldsymbol{u} = (y, -x)$. The initial scalar field is $\iota_0 = \alpha e^{-((x+a)^2+(y+b)^2)/\sigma}$, for $a = b = 0.05$, $\alpha = 0.5$, and $\sigma = 0.001$, and set in `prep_DG.F`.

The exact solution may be determined by noticing that since for any $F(x, y)$, where $x = x(t)$ and $y = y(t)$, that

$$\frac{dF}{dt} = \partial_t F + \begin{pmatrix} x' \\ y' \end{pmatrix} \nabla F = 0,$$

which implies that for

$$\boldsymbol{u} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} y \\ -x \end{pmatrix}, \quad \text{we have the system} \quad x' = y \quad \text{and} \quad y' = -x,$$

that may be solved by recombining such that the solution to the second order ODE, $y'' + y = 0$ can be viewed as a generator of the rotation matrix R about the origin. That is, we obtain the clockwise transformation

$$R = \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix}, \tag{2}$$

such that $R\boldsymbol{x}$ yields the exact solution, as set in `DG_hydro_timestep.F`.

We have convergence in the tracer, which is enough to confirm that the tracer is properly implemented, and also that low-level functionality of the code is spot-on (e.g. basis functions and integrations rules).

All tests run on stampede with either 256 or 16 procs to 7958 timesteps (one full revolution about the origin). All raw data is included in the tgz file. Please recall that convergence rates $\mathscr{C}$ are computed using

$$\mathscr{C} = \frac{1}{\ln(2)} \ln\left(\frac{\|e_h\|_{L^2(\Omega_h)}}{\|e_{h/2}\|_{L^2(\Omega_h)}}\right).$$

The raw data is included, as tecplot output files for every case, plus the output from `write_results.F` that computes the *global* L^2 and L^∞ errors, in `dgswem.o*`, in their respective folders.

Notice at $p = 6$ we do not see full optimal convergence. This is first potentially due to machine precision and the span of the test cases. Assume we have machine epsilon $\sim 10^{-14}$, then $(5 \times 10^{-14}) \times 7958 \approx 4 \times 10^{-10}$, so machine precision error accumulates at the same order as model error for this top case, thus it is not indicative of the convergence rate of the scheme. Also, the $p = 6$ is potentially getting swamped with temporal error, as we are using RKSSP(6,4) here and stepping over many timesteps (we need to implement a higher order temporal integrator to test this properly, time integrator should be of order $p + 1$). Note: we see almost uniform superconvergence once beyond preasymptotic behavior, away from machine precision, and above time integration requirements.

The preasymptotic behavior is driven by the extremely sharp variance σ, though one can easily rid themselves of these errors by making a gentler peak, though must be careful in this instance not to introduce boundary error that swamps the approximation error (e.g. you can use a larger grid). All of this can (and has) been teased out, though not in a single graph. Because of these issues, generally it is best to hone into a basic area of the convergence chart, for any one particular study. All of this behavior (discussed above) can be shown with this test example, by only changing the values of $\sigma, a, b, \alpha, dt, T, h, p$, and the RK scheme. If you have any trouble, please let us know.

Additional thought: please make sure your compilers are all using double precision (even your grid generators!!).

p	L^2-error	Convergence rate	$h_0 = 1/h$
1	2.3292×10^{-4}	3.2311	128
2	2.4648×10^{-6}	3.5558	128
3	7.2786×10^{-8}	5.0816	128
4	3.6380×10^{-9}	$4.9360^{\dagger}$	128
5	–	–	128
6	–	–	128
1	2.1870×10^{-3}	1.7614^{*}	64
2	2.8986×10^{-5}	3.4354	64
3	1.6394×10^{-6}	4.1164	64
4	1.1136×10^{-7}	$5.6175^{\dagger}$	64
5	8.0089×10^{-9}	$5.9989^{\dagger}$	64
6	5.2075×10^{-10}	$6.3902^{\dagger}$	64
1	7.4147×10^{-3}	2.0624^{*}	32
2	3.1357×10^{-4}	4.3425^{*}	32
3	2.8435×10^{-5}	4.9768	32
4	5.4672×10^{-6}	$5.4561^{\dagger}$	32
5	5.1219×10^{-7}	$6.0450^{\dagger}$	32
6	4.3684×10^{-8}	$6.8345^{\dagger}$	32
1	3.0970×10^{-2}	–	16
2	6.3615×10^{-3}	–	16
3	8.9539×10^{-4}	–	16
4	2.4000×10^{-4}	–	16
5	3.3819×10^{-5}	–	16
6	4.9851×10^{-6}	–	16

Table 1: The * indicates preasymptotic behavior. The reason for this is the radius of convergence of the Gaussian blob due to the sharp variance (can be computed directly using a Mercer-Roberts plot). The † indicates either machine precision concerns or temporal integration order concerns.