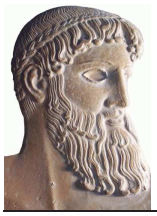
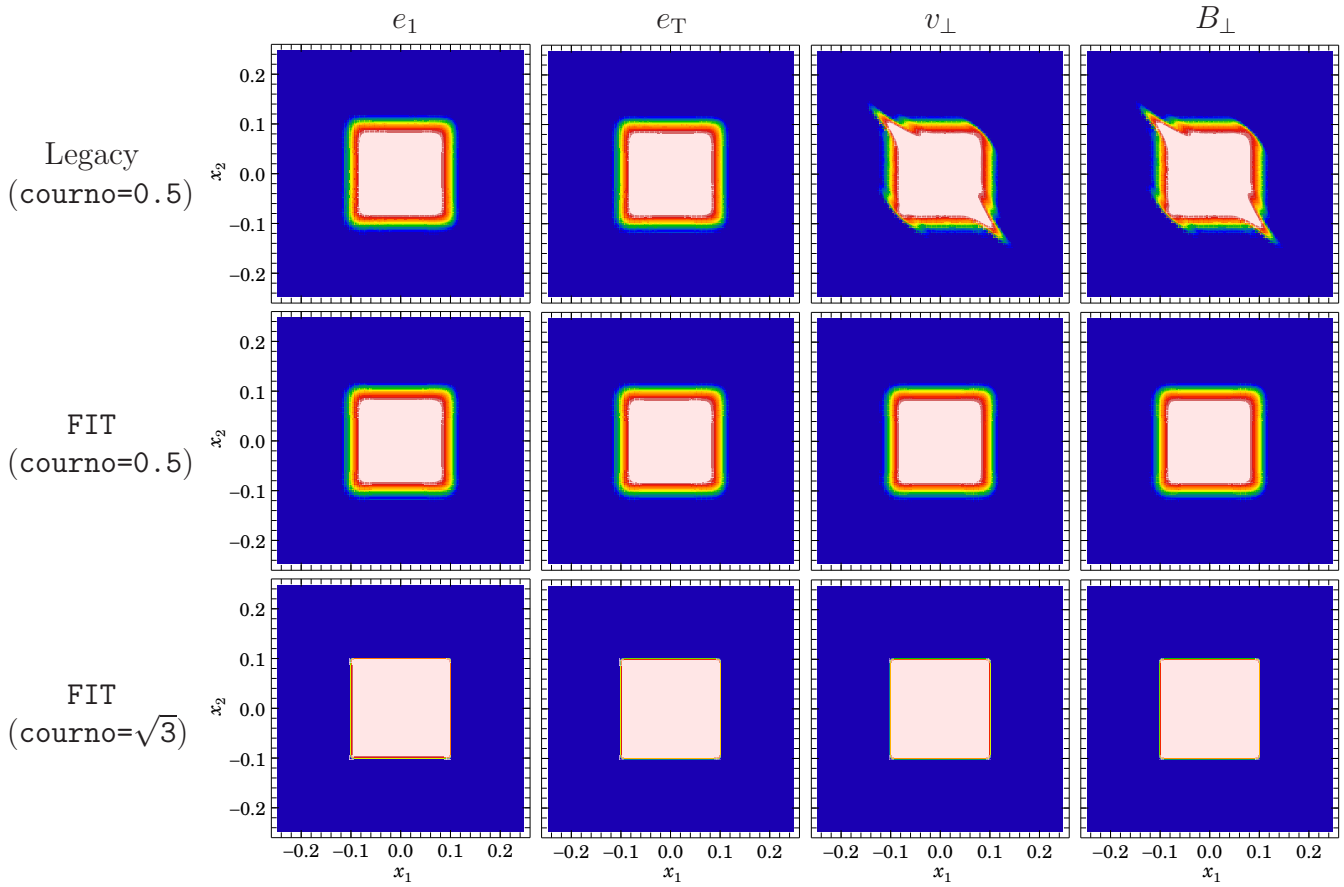


ZEUS-3D 2-D Gallery #1: 2-D Advection



New to Version 3.6 is the *Finely Interleaved Transport* algorithm—`FIT`—which is engaged by setting the `dzeus36` parameter `trnvrsn=1`. `FIT` cures a number of long-standing numerical issues with *ZEUS*, but does so without introducing any new numerical techniques, *per se*, such as `CT` or `CMoC` did. Rather `FIT` is a rearrangement of the overall structure of the *ZEUS* transport algorithm used in all known versions of the code up to and including my own Version 3.5 (which I now refer to as *Legacy Transport*; `trnvrsn=0`) into more finely resolved modules. I'll return to the algorithmic details of `FIT` at the end of this document.

This particular test problem—advection in 2-D—is an excellent exemplar of the utility of `FIT`. A uniform 100×100 Cartesian grid on a domain $(x_1, x_2) = (-0.25 : 0.25, -0.25 : 0.25)$ with periodic boundary conditions is initialised with $\rho = v_1 = v_2 = c_s = 1$ and $v_\perp = \vec{B} = 0$. A 40×40 zone square pulse with amplitude 0.001 is initialised in each of e_1, $v_\perp$, $B_\perp$ (and thus e_T) at the grid centre and, since $v_1 = v_2 = 1$, the pulses are advected 45° relative to the grid axes (bottom left to top right). All tests are run to $t = 0.5$ and, with periodic boundaries, the pulses cycle through the grid once and return to the centre where they began. Click on any image in the table above for an animation.

Being a test in pure advection, all source terms (*e.g.*, ∇p, $\vec{J} \times \vec{B}$, *etc.*) are set to zero, including artificial viscosity (`qcon=0`, `qlin=0`). In principle, all non-advected quantities such as ρ, v_1, *etc.*, could be reset to their initial values after each MHD step, though with the small amplitudes of the pulses, this is not necessary. The length of the time step is controlled by the Courant number (`courno`) given in the left column of the table above. The top row of images is the result using legacy transport (`trnvrsn=0`), the second row with `FIT` (`trnvrsn=1`), and the third row with `FIT` and a CFL-limited timestep.

Unbeknownst to anyone, I believe, until I stumbled upon it in 2010, no version of *ZEUS* from Stone & Norman's *ZEUS-2D* up to and including my own Version 3.5 has been able to properly advect a 2-D square wave in perpendicular vector components! The left two panels in the top row shows that Legacy transport is capable of returning pulses of scalars intact to the grid centre (with perfectly acceptable diffusion for a second-order algorithm) but not $v_\perp$ and $B_\perp$ (top right two panels). After once around the grid, both show a pronounced "striping instability" perpendicular to the direction of advection which get worse with time leading to the ultimate destruction of the square wave. The only way to delay (but not prevent) this problem using Legacy transport is to reduce `courno`.

On the other hand `FIT` (second row) cures the problem completely and, as seen in the third row, advection using `FIT` is CFL-limited[1] which, among other things, means the square pulse is advected through the grid with virtually no diffusion. Legacy transport is most certainly not CFL-limited.

Curiously, if advection is along one of the coordinate axes, Legacy transport does not produce any striping. It is only when advection is at some angle relative to the grid axes does the problem show up, and indeed this clue was instrumental in designing `FIT`. I would point out that in a typical simulation where fluid flow is not in one prominent direction, these first-order errors tend to cancel out, and the striping instability is unlikely to show up. In fact, I have never seen anything in any of my production runs over the past 25 years that I would, even in hindsight, identify as being related to this numerical artefact.

So what is `FIT`?

From the beginning, *ZEUS*' algorithmic design was based on principles of operator- and directional-splitting, where the *source terms* (∇p, $p\nabla\cdot\vec{v}$, viscosity) are treated in a separate module from the *transport terms* ($\nabla\cdot\rho\vec{v}$, *etc.*). In its earliest form (before the introduction of $\vec{B}$ and $e_{\rm T}$), the code would:

1. apply source terms to velocity components and internal energy density;
2. compute momentum components from the face-centred density and partially-updated velocity;
3. transport density and internal energy density;
4. transport momentum components;
5. update velocities from fully-updated momentum and density;
6. go to 1.

In this algorithm, the velocity components are not updated until ρ and *all* momentum components are fully transported. Thus, and for example, If ρ and s_1 were transported in the x_1-direction, v_1 used to transport s_2 in the x_1-direction still has the value it had *before* the x_1-transport of ρ and s_1.

When the magnetic terms were introduced—first by myself for my Ph.D. thesis, and then a much better algorithm by Stone & Norman (1992)—this same "coarse" operator splitting was adopted. Thus, and for example, in all versions of *ZEUS* before and including my own Version 3.5, no magnetic field component is updated until *all* induced electric field components (*a.k.a.*, *emfs*) are computed.

The striping instability is a direct result of this coarse modularity.

[1]The CFL condition requires that the numerical time step, $\delta t_{\rm Z}$, be no larger than the smallest time scale of any physical process in any zone. In this problem, with all speeds $v_1 = v_2 = c_{\rm s} = 1$ and $\delta x = 0.005$, the CFL-limited time-step is $\delta t_{\rm CFL} = \delta x/v_{\rm max} = 0.005$. However, the way *ZEUS* calculates the time step is to add all physical time steps in reciprocal square, then take the square root and inverse. Thus, $(\delta t_{\rm Z})^{-1} = \sqrt{(\delta t_{v_1})^{-2} + (\delta t_{v_2})^{-2} + (\delta t_{c_{\rm s}})^{-2}} \Rightarrow \delta t_{\rm Z} = 0.005/\sqrt{3}$ for this problem. Therefore, the maximum "Courant number" (the factor by which one can multiply $\delta t_{\rm Z}$ and still be CFL compliant) is $\sqrt{3}$. Note that taking `courno` even one part in 10^6 beyond $\sqrt{3}$ breaks up the advected pulse into unrecognisable zone-to-zone oscillations even by $t = 0.5$; the transition from CFL stable to unstable is stunningly sudden.

FIT seeks to apply the "finest" degree of operator splitting possible by adopting the following principle:

A primitive variable must be updated as soon as quantities affecting it have been updated.

Thus, instead of waiting until all momentum components are updated before updating the velocity components, FIT updates v_1 as soon as ρ and s_1 have been transported in the x_1-direction. That way, the value of v_1 used to transport s_2 in the x_1-direction has benefit of a partial transport step. It is this modification that cured the algorithm of the striping instability in this 2-D advection problem. Similarly, as soon as ε_1 is evaluated, B_2 and B_3 are partially updated with $\partial_3\varepsilon_1$ and $-\partial_2\varepsilon_1$ respectively. That way, when ε_2 $(= v_3B_1 - v_1B_3)$ is evaluated, it has benefit of a partially updated B_3. It is this step that cured the algorithm of striping in the 2-D Alfvén wave transport problem.

In hindsight, by not updating the primitive variables as often as possible renders incomplete the time-centring of the algorithm. Indeed, FIT also fixes the problem of rarefaction shocks, which Sam Falle (2002, ApJ, 577, L123) suggested were caused by an imbalance in the order of accuracy of the code (second order space; only first order time). If this is true, FIT is an important step—maybe the last step—in rendering *ZEUS* a truly second order code.

As it turns out, there are numerous opportunities to interleave Legacy transport more finely and, until every one is exploited, certain flavours of the striping instability persist. In this regard, there are at least two aspects of FIT that remain unsettled in this release.

1. While the current implementation of FIT cures the rarefaction shock problem when the internal energy equation is used, rarefaction shocks are still an issue for the total energy equation, though to a much lesser extent than with Legacy transport.

2. The one test problem I know of where Version 3.6 is still vulnerable to the striping instability is flux loop advection (Gardiner & Stone, 2005, J. Comput. Phys., 205, 509). While I know what has to be done to fix the problem, I haven't figured out yet how to implement it efficiently in *ZEUS*.

On the former, there evidently remain some unexploited opportunities for fine interleaving in the total energy equation algorithm, or perhaps some of the steps already there are out of proper sequence. This is still an open question.

On the latter, it turns out that the computation of a single component of the *emf* is not the finest possible modularity of the algorithm. In fact, $\varepsilon_1 = v_2B_3 - v_3B_2 \equiv \varepsilon_{23} - \varepsilon_{32}$ has *two* distinct terms and, in principle, one could update B_2 with ε_{23} even before computing ε_{32} that depends upon B_2. Doing this exploits the finest "interleavability" of the algorithm that I can identify, and is the last step needed to cure the problem of flux-loop advection striping. I know this because I've written a small, bare-bones induction code that demonstrates this; I just haven't found a way yet to install it into *ZEUS* without disrupting the even-more-critical CMoC algorithm.

D. Clarke, May 2016.