Numerical Simulations of the EPDiff and two-component EPDiff equations

Lennon Ó Náraigh*

School of Mathematics and Statistics,
University College Dublin, Belfield, Dublin 4, Ireland

Anyone else

(Dated: October 31, 2025)

Abstract here.

I. EPDIFF

We study the **EPDiff equation** (Euler–Poincaré equation on the diffeomorphism group) in n dimensions:

$$\frac{\partial \boldsymbol{m}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{m} + (\nabla\boldsymbol{u})^T\boldsymbol{m} + \boldsymbol{m}(\nabla\cdot\boldsymbol{u}) = 0, \tag{1}$$

where

$$\boldsymbol{m} = (1-\alpha^2\Delta)\boldsymbol{u}, \tag{2}$$

and where:

- $\boldsymbol{u}(\mathbf{x},t) \in \mathbb{R}^n$ is the velocity field,
- $\boldsymbol{m}(\mathbf{x},t)$ is the corresponding momentum density,
- $\alpha > 0$ is a length-scale parameter,
- and Δ is the Laplacian operator.

Hence, $\boldsymbol{u}$ can be obtained from $\boldsymbol{m}$ via convolution:

$$\boldsymbol{u} = (1-\alpha^2\Delta)^{-1}\boldsymbol{m} \iff \boldsymbol{u} = K * \boldsymbol{m}. \tag{3}$$

The nonlinear terms in Equation (1) represent convection, stretching, and expansion, respectively.

We immediately specialize to two dimensions, where we have $\boldsymbol{u} = (u,v)$, $\boldsymbol{m} = (m_x, m_y)$, and:

$$\frac{\partial m_x}{\partial t} + (\boldsymbol{u}\cdot\nabla)m_x + m_x\frac{\partial u}{\partial x} + m_y\frac{\partial v}{\partial x} + (\nabla\cdot\boldsymbol{u})m_x = 0, \tag{4a}$$

$$\frac{\partial m_y}{\partial t} + (\boldsymbol{u}\cdot\nabla)m_y + m_x\frac{\partial u}{\partial y} + m_y\frac{\partial v}{\partial y} + (\nabla\cdot\boldsymbol{u})m_y = 0. \tag{4b}$$

We solve Equations (4) on a doubly periodic domain $\Omega = [0, L_x] \times [0, L_y]$, with given initial conditions on $\boldsymbol{m}(\mathbf{x}, t=0) = \boldsymbol{m}_0(\mathbf{x})$, with initial conditions on $\boldsymbol{u}(\mathbf{x}, t=0) = \boldsymbol{u}_0(\mathbf{x})$ obtained

* Corresponding author. Email: onaraigh@maths.ucd.ie

via convolution. To solve Equations (4) numerically, it will be more useful to recast them as:

$$\frac{\partial u}{\partial t} = -K * \left[(\boldsymbol{u}\cdot\nabla)m_x + m_x\frac{\partial u}{\partial x} + m_y\frac{\partial v}{\partial x} + (\nabla\cdot\boldsymbol{u})m_x \right], \tag{5a}$$

$$\frac{\partial v}{\partial t} = -K * \left[(\boldsymbol{u}\cdot\nabla)m_y + m_x\frac{\partial u}{\partial y} + m_y\frac{\partial v}{\partial y} + (\nabla\cdot\boldsymbol{u})m_y \right]. \tag{5b}$$

II. NUMERICAL SIMULATIONS: METHODOLOGY

We solve Equation (5) using a fully explicit pseudospectral scheme. The domain Ω is discretized with $N_x \times N_y$ grid points and uniform spacings $\Delta x = L_x/N_x$ and $\Delta y = L_y/N_y$. Temporal integration employs a classical fourth-order Runge–Kutta (RK4) method with fixed timestep Δt. At each timestep, the products in Equation (5) are evaluated in real space while derivatives are calculated in Fourier space. The transformations between real space and Fourier spare are performed using FFTs. The inverse Helmholtz operator $(1-\alpha^2\Delta)^{-1}$ is implemented as a diagonal Fourier multiplier $(1+\alpha^2|\mathbf{k}|^2)^{-1}$. To prevent aliasing, a two-thirds wavenumber cutoff is applied as a spectral mask.

A. Validation

We validate the numerical method with respect to a 'momentum strip' initial condition,

$$m_{0,x}(x,y) = M\delta_\epsilon(x-x_c)H_{[-w/2,w/2]}(y), \qquad m0, y = 0. \tag{6a}$$

where δ_ϵ is nascent delta function:

$$\delta_\epsilon(x) = \frac{1}{\sqrt{2\pi\epsilon^2}}\mathrm{e}^{-x^2/2\epsilon^2}, \tag{6b}$$

with $\epsilon = 2\Delta x$, and $H_{[-w/2,w/2]}(y)$ is the Heaviside step function:

$$H_{[-w/2,w/2]}(y) = \begin{cases} 1, & y \in [-w/2, w/2], \\ 0, & \text{otherwise} \end{cases} \tag{6c}$$

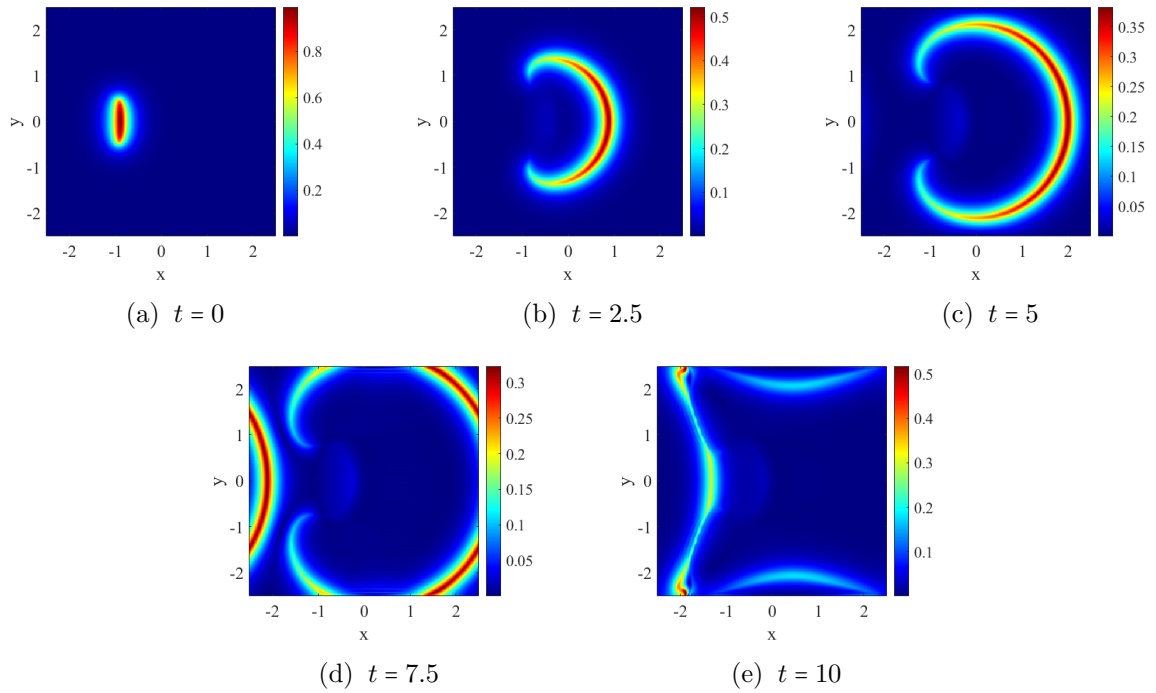
The initial condition $\boldsymbol{u}_0(x,y)$ is obtained via convolution, with M chosen such that $\max_{(x,y)\in\Omega} u_0(x,y) = 1$. Parameter values are given in Table I. The numerical parameters are $N_x = N_y = 512$ and $\Delta t = 10^{-3}$. Snapshots of the results are shown in Figure 1 for $\sqrt{u^2+v^2}$. These agree qualitatively with the 'momentum strip' initial condition given in References [1, 2]. A precise quantitative comparison with these earlier works is not possible because these earlier works do no provide numerical parameter values.

B. Grid convergence

We have run the simulation in the previous subsection multiple times with ever-increasing grid resolution, as summarized in Table II. For the simulation at mesh refinement N, record

Parameter	Value
L_x	5
L_y	5
α	0.2
x_c	-1
w	1

TABLE I. List of parameter values corresponding to Figure 1



(a) $t = 0$ (b) $t = 2.5$ (c) $t = 5$

(d) $t = 7.5$ (e) $t = 10$

FIG. 1. First set of results: shown is the speed $\sqrt{u^2 + v^2}$ at various times. Physical parameter values as in Table I. Initial conditions and numerical parameters as in the text.

the L^1 error at fixed time $t = t_0 := 2.5$

$$\varepsilon(N) = \int_\Omega \left(|u_N - u_{2N}| + |v_{2N} - v_{2N}|\right) \mathrm{d}x\mathrm{d}y, \qquad \text{at } t = t_0 \tag{7}$$

and tabulate the results. We use the L^1 norm here because of its association with solutions of bounded variation – precisely the kind of singular solutions studied in the present work. Here, simulations at resolution $N_x = N_y = N$ are compared with simulations at resolution $N_x = N_y = 2N$, with the simulations on the coarse grid interpolated on to the corresponding finer grid (linear interpolation). The results are also plotted in Figure 2. The results show a trend $\varepsilon(N) \sim N^{-1}$, which is rather poor convergence. For smooth data, a pseudospectral discretization ought to yield exponential convergence. However, the simulated initial conditions are not smooth – they correspond to a singular solution. As such, slow convergence such with $\varepsilon(N) \sim N^{-1}$ is expected. A visual inspection of results at different resolutions confirms the convergence (albeit slow) – see Figure 3. From this figure, it can be seen that there is very little qualitative or quantitative difference in the solution behaviour beyond

N	$\varepsilon(N)$
128^2	0.4408
256^2	0.2067
512^2	0.1363
1024^2	0.0611
2048^2	–

TABLE II. Mesh refinement study. Physical parameters as in Table I. Timestep: $\Delta t = 10^{-3}$.

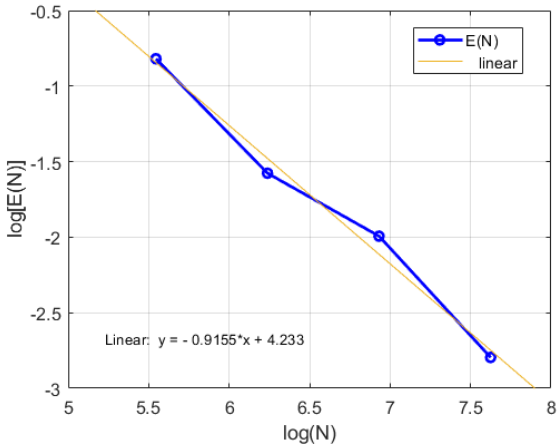


FIG. 2. Convergence study. Parameters as in Table I. Timestep: $\Delta t = 10^{-3}$. The error between successive simulations is computed as in Equation (7).

$N = 512$. Hence, this resolution is used in the rest of the present work. Finally, to test the extent to which the simulation results depend on the value of Δt, we have run two simulations with the same conditions as before at temporal resolutions $\Delta t = 10^{-4}$ and $\Delta t = 10^{-5}$ and evaluated

$$\varepsilon(\Delta t) = \int_\Omega \left(|u_{\Delta t} - u_{10\Delta t}| + |v_{\Delta t} - v_{10\Delta t}|\right) \mathrm{d}x\mathrm{d}y, \qquad \text{at } t = t_0; \tag{8}$$

we report $\varepsilon(\Delta t = 10^{-4}) = 4.6035 \times 10^{-7}$, such that the simulations are very well resolved in the temporal domain, at $\Delta t = 10^{-3}$.

C. Energy conservation

We monitor the conserved energy

$$E(t) = \tfrac{1}{2} \int_\Omega \boldsymbol{u} \cdot \boldsymbol{m} \, \mathrm{d}x \, \mathrm{d}y \tag{9}$$

over the course of a simulation out until the final simulation time T, where $T = 10$. Although the chosen temporal discretization does not manifestly conserve energy, with the small timetep chosen, the change in the energy $[E(T) - E(0)]/E(0)$ is 0.0045% (Figure 4), confirming the validity of the simulations.

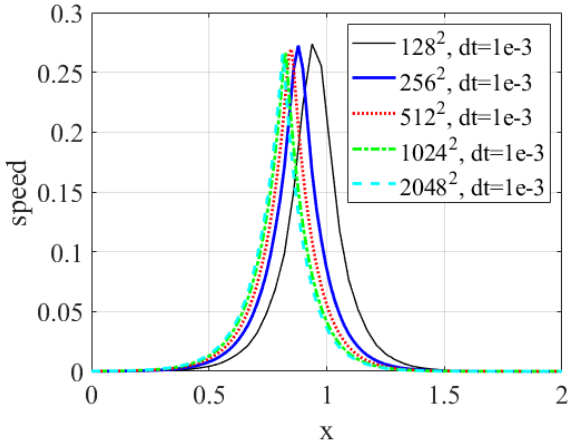


FIG. 3. Snapshots of $u(x, L_y/2, t_0)$ at various levels of mesh refinement, where $t_0 = 2.5$. Other parameters as in Table I. Timestep: $\Delta t = 10^{-3}$.

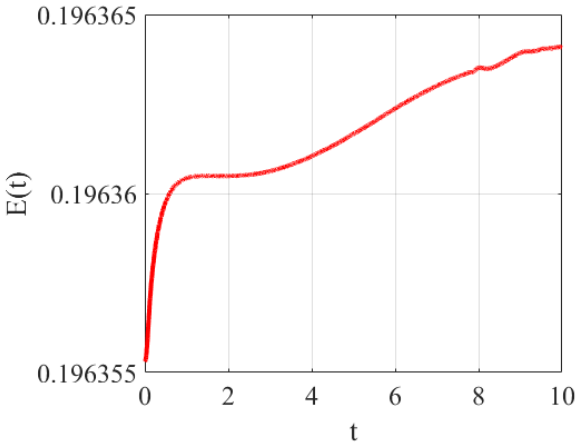


FIG. 4. A plot of the energy $E(t)$ over time. Physical parameters as in Table I. Numerical parameters: $\Delta t = 10^{-3}$, $N_x = N_y = 512$.

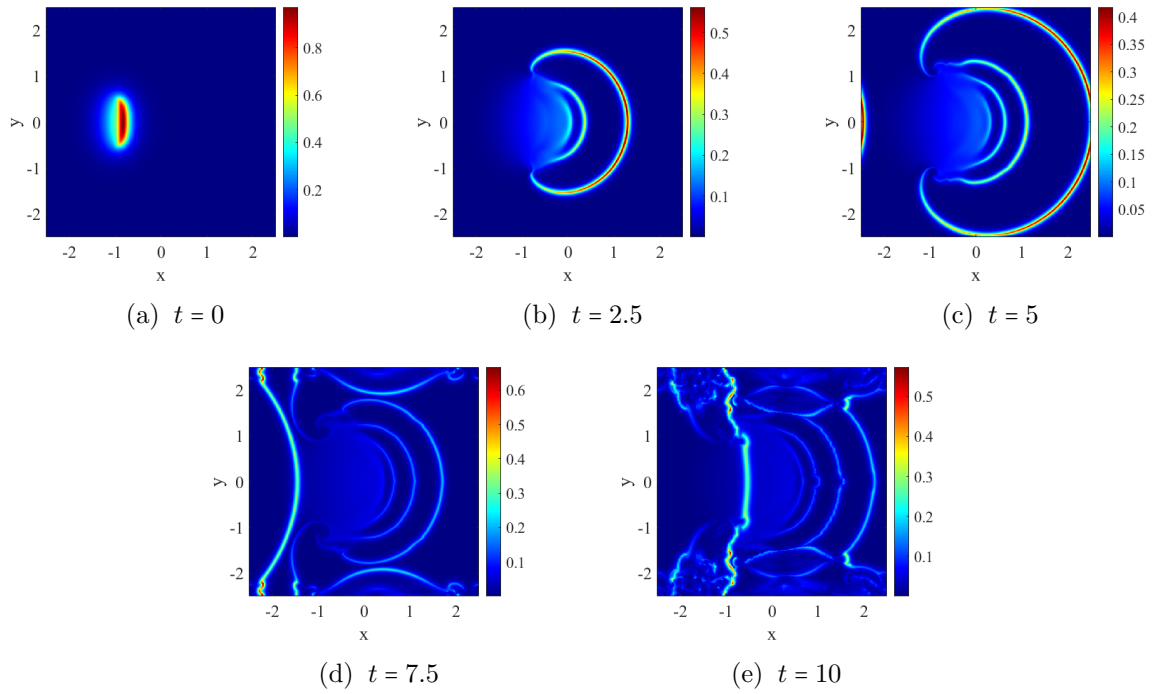
D. Further test cases

We examine further sets of initial conditions to validate the solver. We first of all initialize with the previous 'momentum strip' initial conditions (6) and construct $\boldsymbol{u}_0$ via a convolution

$$\boldsymbol{u}_0(x, y) = \left(1 - \sigma^2 \Delta\right)^1 \boldsymbol{m}_0. \tag{10}$$

We thereafter make use of the convolution operator $K = (1 - \alpha^2 \Delta)^{-1}$ in the EPDiff equation, where α is *less than* σ, $\alpha < \sigma$. In this way, we generate the 'wave trains' seen previously in Reference [1]. Results in Figure 5. We use the physical parameters in Table III with the exception $\sigma = 0.2$ and $\alpha = \sigma/4$. Lastly, we look at initial conditions involving two momentum strips:

$$m_{0,x}(x, y) = 2M\delta_\epsilon(x - x_{c1})H_{[a_1,b_1]}(y) + M\delta_\epsilon(x - x_{c2})H_{[a_2,b_2]}(y), \tag{11}$$



(a) $t = 0$ (b) $t = 2.5$ (c) $t = 5$

(d) $t = 7.5$ (e) $t = 10$

FIG. 5. Snapshot of the speed $\sqrt{u^2 + v^2}$ at various times, for the initial condition 10.

with parameters as given in Table III (any parameters not explicitly referenced here are the same as those in Table I; in particular, $\alpha = 0.2$). The value M is again chosen such that $\max_{(x,y)\in\Omega} u_0(x, y) = 1$.

Parameter	Value
x_{c1}	-0.75
x_{c2}	-0.25
a_1	-0.25
b_1	0.75
a_2	-0.75
b_2	0.25

TABLE III. List of parameter values for initial conditions giving rise to collisions

Furthermore, we have $m_{0,y}(x, y) = 0$, and $\boldsymbol{u}_0(x, y) = (1 - \alpha^2\Delta)^{-1}\boldsymbol{m}_0(x, y)$. Snapshots of the speed at various times are shown in Figure 6 and again resemble their counterparts in Reference [1]. Again, a strictly quantitative comparison is not possible because that earlier work does not report parameter values. However, the resemblance between the two simulation sets is very clear – two interacting singular solutions, each supported on curves, which collide and pass through one another, before the onset of 'wrap-around' due to the periodic boundary conditions.

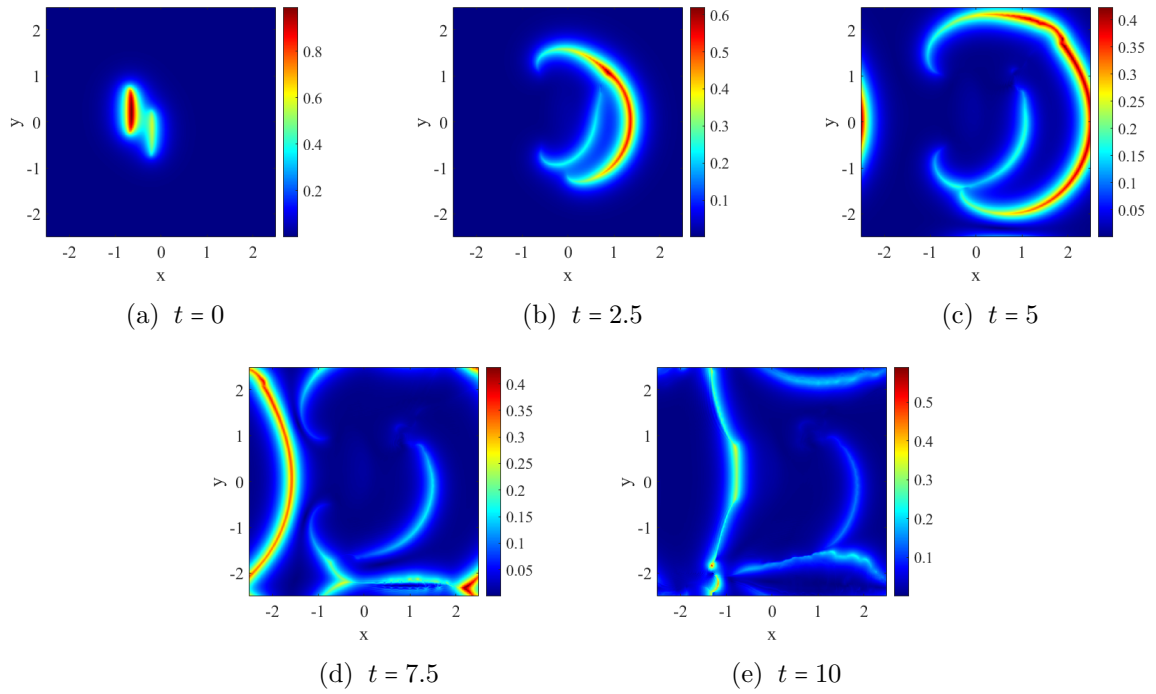


FIG. 6. Snapshot of the speed $\sqrt{u^2 + v^2}$ at various times, for the initial condition 11.

CPU	GPU
14,612 s	394 s

TABLE IV. Timings for an ordinary CPU-based simulation and a GPU-accelerated simulation

E. GPU acceleration

We make use of the GPU acceleration in Matlab, which is available on machines equipped with NVIDIA graphics cards. The GPU acceleration is achieved by declaring all large arrays (`gpuArray`) at the start of the computation, thereby performing FFTs and pointwise operations on the GPU device. We compare the execution time of a simulation with and without acceleration in Table IV. The benchmark is for a simulation running until a final time $t = 10$, with a resolution $N_x = N_y = 512$, with $\Delta t = 10^{-3}$, using the initial condition (6). For a fair comparison, no intermediate outputs are generated – the arrays are only outputted at the end of the simulation. The machine used for the test was equipped with an AMD Ryzen 9 12-core CPU (4.40 GHz, 190 GB RAM) and an NVIDIA GeForce RTX 3050 GPU with 101 GB Memory. The results in the table show a 37-fold speedup when using the GPU as opposed to the CPU.

[1] Darryl D Holm and Martin F Staley. Interaction dynamics of singular wave fronts. *arXiv preprint arXiv:1301.1460*, 2013.

[2] Alina Chertock, Philip Du Toit, and Jerrold Eldon Marsden. Integration of the epdiff equation by particle methods. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(3):515–534, 2012.