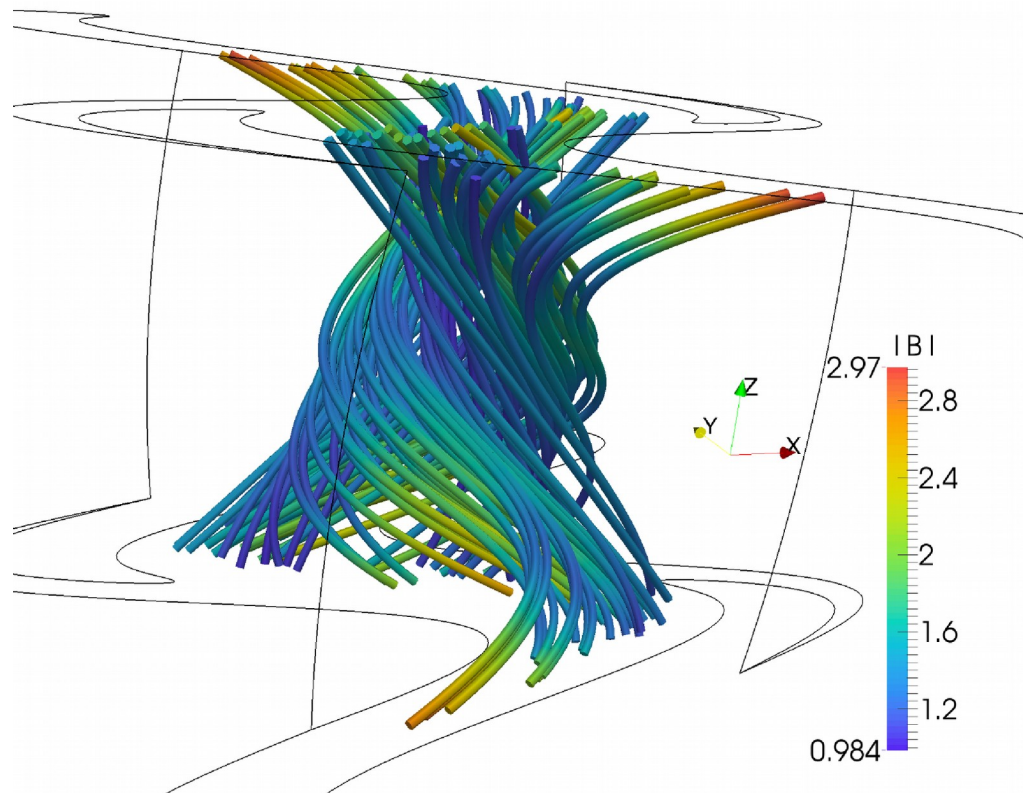# Topology conserving magnetic field relaxation
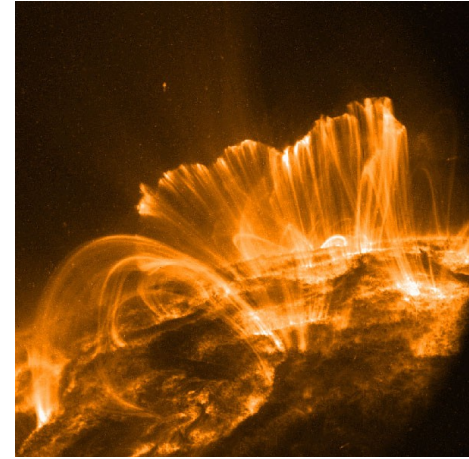
Simon Candelaresi

Force-Free Magnetic Fields

Solar corona: low plasma beta and magnetic resistivity
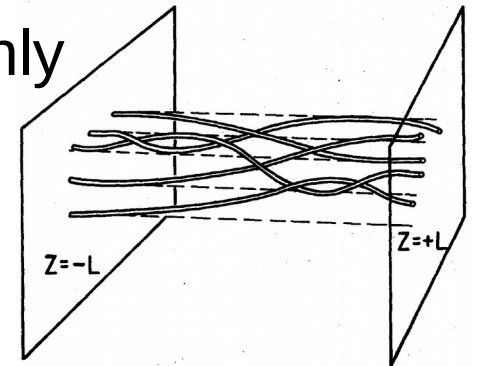
➡ Force-free magnetic fields

➡ Minimum energy state

$$(\nabla \times \mathbf{B}) \times \mathbf{B} = 0 \iff \nabla \times \mathbf{B} = \alpha \mathbf{B}$$

Parker: Equilibrium with the same topology exists only if the twist varies uniformly along the field lines. Strongly braided fields → topological dissipation.
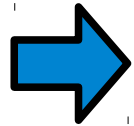
(Parker 1972)

Braided fields from foot point motion complex enough. *(Parker 1983)*

Solutions possible with filamentary current structures (sheets).

(Mikic 1989, Low 2010)

2

Ideal Field Relaxation

Ideal induction eq.: $\dfrac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B})$

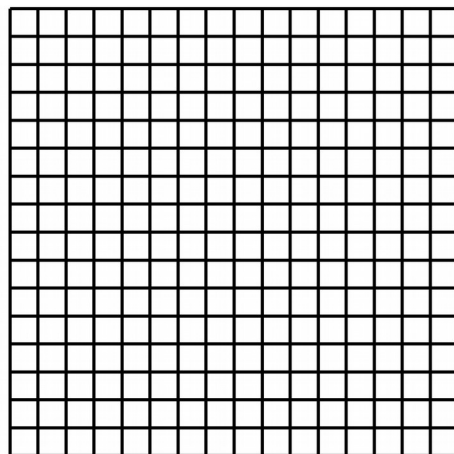➡️ Frozen in magnetic field.

(Batchelor, 1950)

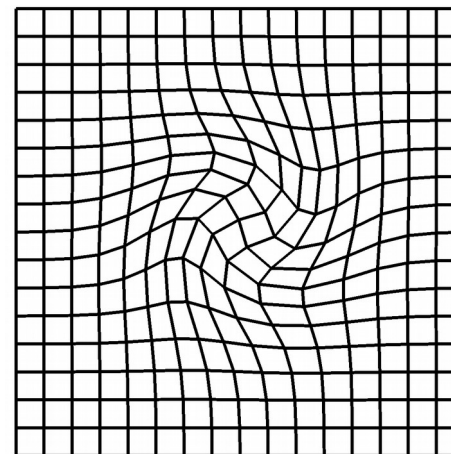☁️ **But**: Numerical diffusion in finite difference Eulerian codes.

(Rembiasz, 2017)

☀️ **Solution**: Lagrangian description of moving fluid particles:

$$\mathbf{x}(\mathbf{X}, 0) = \mathbf{X} \qquad\qquad \mathbf{x}(\mathbf{X}, t)$$

3

Ideal Field Relaxation

Field evolution: $B_i(\mathbf{X}, t) = \dfrac{1}{\Delta} \displaystyle\sum_{j=1}^{3} \dfrac{\partial x_i}{\partial X_j} B_j(\mathbf{X}, 0)$

$$\Delta = \det \left(\frac{\partial x_i}{\partial X_j} \right)$$

Preserves topology and divergence-freeness.

Grid evolution: $\dfrac{\partial \mathbf{x}(\mathbf{X}, t)}{\partial t} = \mathbf{u}\left(\mathbf{x}(\mathbf{X}, t), t\right)$

Magneto-frictional term: $\mathbf{u} = \gamma \mathbf{J} \times \mathbf{B}$ $\qquad \mathbf{J} = \nabla \times \mathbf{B}$

$$\Rightarrow \quad \frac{\mathrm{d}E_{\mathrm{M}}}{\mathrm{d}t} < 0 \qquad \textit{(Craig and Sneyd 1986)}$$

Numerical Curl Operator

Compute $\mathbf{J} = \nabla \times \mathbf{B}$ on a distorted grid:

$$\frac{\partial B_i}{\partial x_j} = X_{\alpha,j}(x_{i,\alpha\beta}B^0_\beta \Delta^{-1} + x_{i,\beta}B^0_{\beta,\alpha}\Delta^{-1} - x_{i,\beta}B^0_\beta \Delta^{-2}\Delta_{,\alpha})$$

$$B^0_i = B_i(0)$$

(Craig and Sneyd 1986)

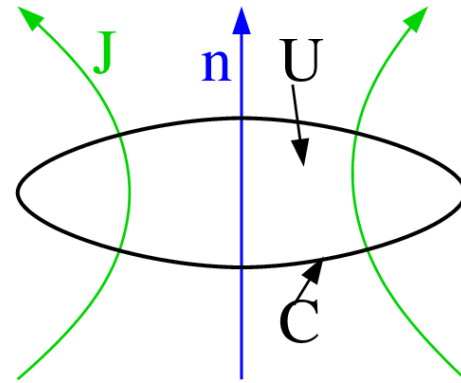Multiplication of several terms leads to high numerical errors.

Current not divergence free: $\nabla \cdot \mathbf{J} \neq 0$

Only reaching a certain force-freeness. *(Pontin et al. 2009)*

Mimetic Numerical Operators

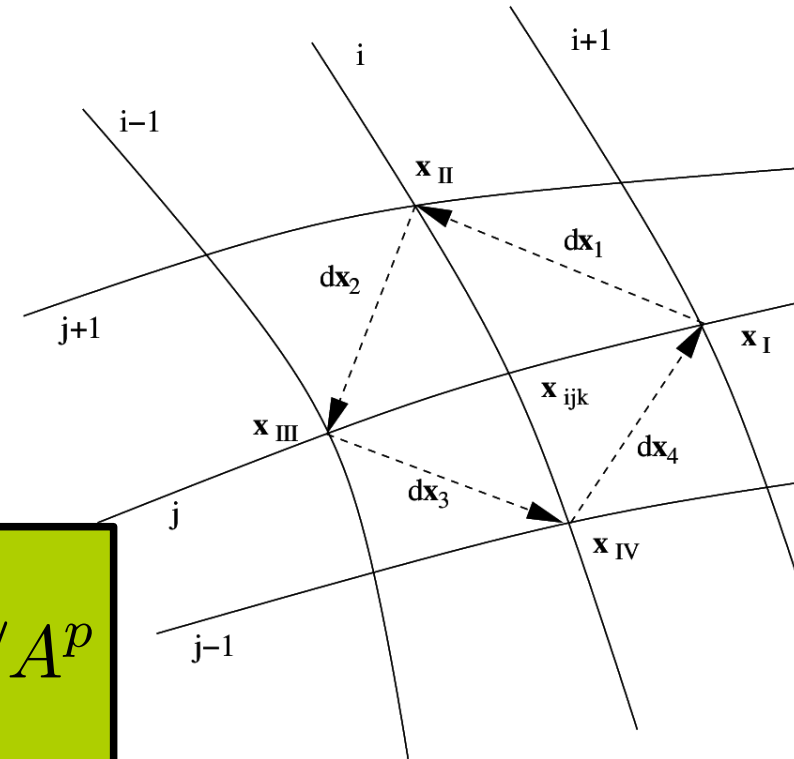$$I = \int_U \mathbf{J} \cdot \mathbf{n} \, \mathrm{d}S = \oint_C \mathbf{B} \cdot \mathrm{d}\mathbf{r}$$

Discretized:

$$I \approx \mathbf{J}(\mathbf{X}_{ijk}) \cdot \mathbf{n}A = \sum_{r=1}^{4} \mathbf{B}_r \cdot \mathrm{d}\mathbf{x}_r$$

$$\mathbf{J}(\mathbf{X}_U) \approx \mathbf{J}(\mathbf{X_{ijk}}), \quad \mathbf{X}_U \in U$$

3 planes will give 3 l.i. normal vectors:

$$I^p = \mathbf{J}(\mathbf{X}_{ijk}) \cdot \mathbf{n}^p = \sum_{r=1}^{4} \mathbf{B}_r^p \cdot \mathrm{d}\mathbf{x}_r / A^p$$

Inversion yields $\mathbf{J}$ with $\nabla \cdot \mathbf{J} = 0$. *(Hyman, Shashkov 1997)*

6

Methods

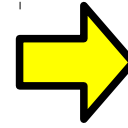Ideal (non-resistive) evolution

Frozen in magnetic field ➡ use Lagrangian method

(Batchelor, 1950)

Preserves topology and divergence-freeness.

Magneto-frictional term: $\mathbf{u} = \gamma \mathbf{J} \times \mathbf{B}$ $\mathbf{J} = \nabla \times \mathbf{B}$

➡ $\dfrac{\mathrm{d}E_{\mathrm{M}}}{\mathrm{d}t} < 0$ *(Craig and Sneyd 1986)*

Fluid with pressure: $\mathbf{u} = \mathbf{J} \times \mathbf{B} - \beta \nabla \rho$

Fluid with inertia: $\mathrm{d}\mathbf{u}/\mathrm{d}t = (\mathbf{J} \times \mathbf{B} - \nu \mathbf{u} - \beta \nabla \rho)/\rho$

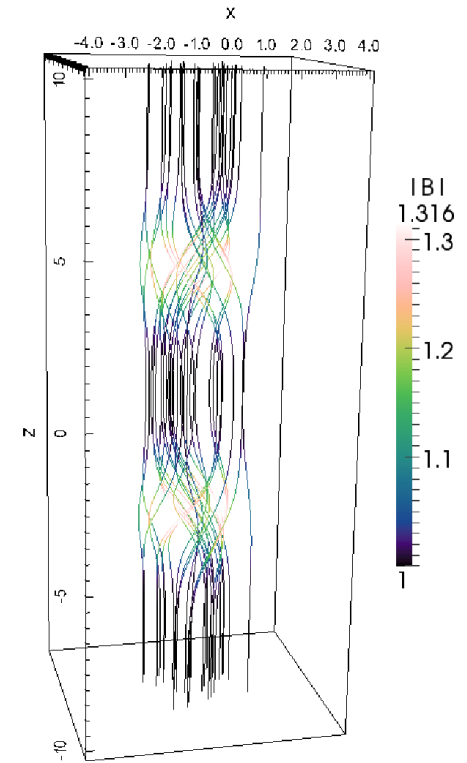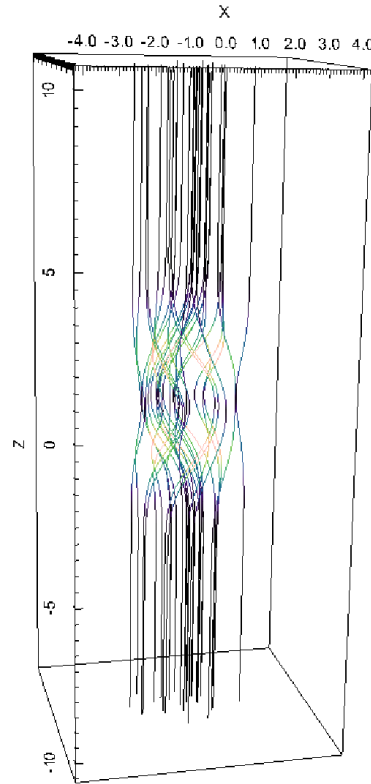For $\mathbf{J} = \nabla \times \mathbf{B}$ use mimetic numerical operators.

(Hyman, Shashkov 1997)

Own GPU code GLEMuR: (https://github.com/SimonCan/glemur).

(Candelaresi et al. 2014)

7

Simulations

- GPU code GLEMuR (**G**pu-based **L**agrangian mim**E**tic **M**agnetic **R**elaxation)

- line tied boundaries

- mimetic vs. classic
  (Candelaresi et al. 2014)

Nvidia Tesla K40

we know:

$$\lim_{t \to \infty} \mathbf{B}(t)$$

$$\lim_{t \to \infty} \mathbf{x}(t)$$

we know:

$$\lim_{t \to \infty} \mathbf{B}(t)$$

8

Quality Parameters

Deviation from the expected relaxed state:

$$\sigma_{\mathbf{x}} = \sqrt{\frac{1}{N} \sum_{ijk} (\mathbf{x}(\mathbf{X}_{ijk}) - \mathbf{x}_{\mathrm{relax}}(\mathbf{X}_{ijk}))^2}$$

$$\sigma_{\mathbf{B}} = \sqrt{\frac{1}{N} \sum_{ijk} (\mathbf{B}(\mathbf{X}_{ijk}) - \mathbf{B}_{\mathrm{relax}}(\mathbf{X}_{ijk}))^2}$$

Free magnetic energy:

$$E_{\mathrm{M}}^{\mathrm{free}} = E_{\mathrm{M}} - E_{\mathrm{M}}^{\mathrm{bkg}}$$

$$E_{\mathrm{M}} = \int_V \mathbf{B}^2/2 \ \mathrm{d}V \qquad \mathbf{B}^{\mathrm{bkg}} = B_0 \hat{e}_z$$

Quality Parameters

For a force-free field: $\nabla \times \mathbf{B} = \alpha \mathbf{B}$

$$\mathbf{B} \cdot \nabla \alpha = 0$$

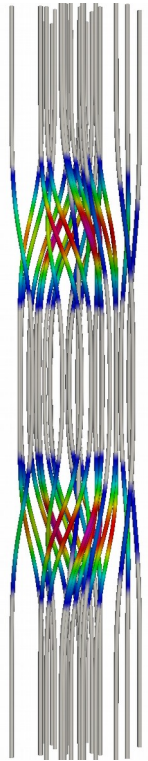⮞ Force-free parameter does not change along field lines.

⮞ Measure the change of $\alpha^* = \dfrac{\mathbf{J} \cdot \mathbf{B}}{\mathbf{B}^2}$ along field lines:

$$\epsilon^* = \max_{i,j} \left(a_r \frac{\alpha^*(\mathbf{X}_i) - \alpha^*(\mathbf{X}_j)}{|\mathbf{X}_i - \mathbf{X}_j|} \right); \quad \mathbf{X}_i, \mathbf{X}_j \in s_\alpha$$

Particular field line: $s_\alpha = \{(0, 0, Z) : Z \in [-L_z/2, L_z/2]\}$
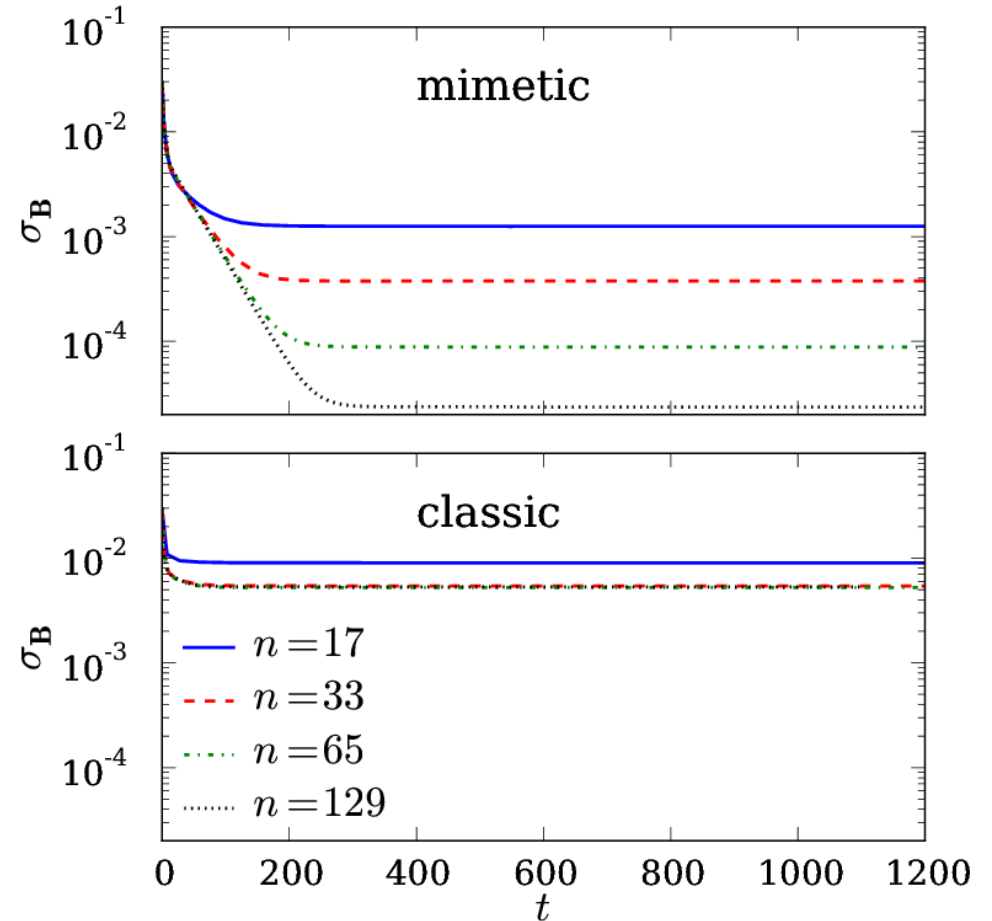
Field Relaxation

Magnetic streamlines:
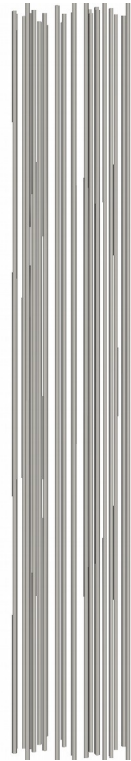
Grid distortion at mid-plane:
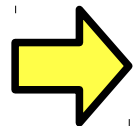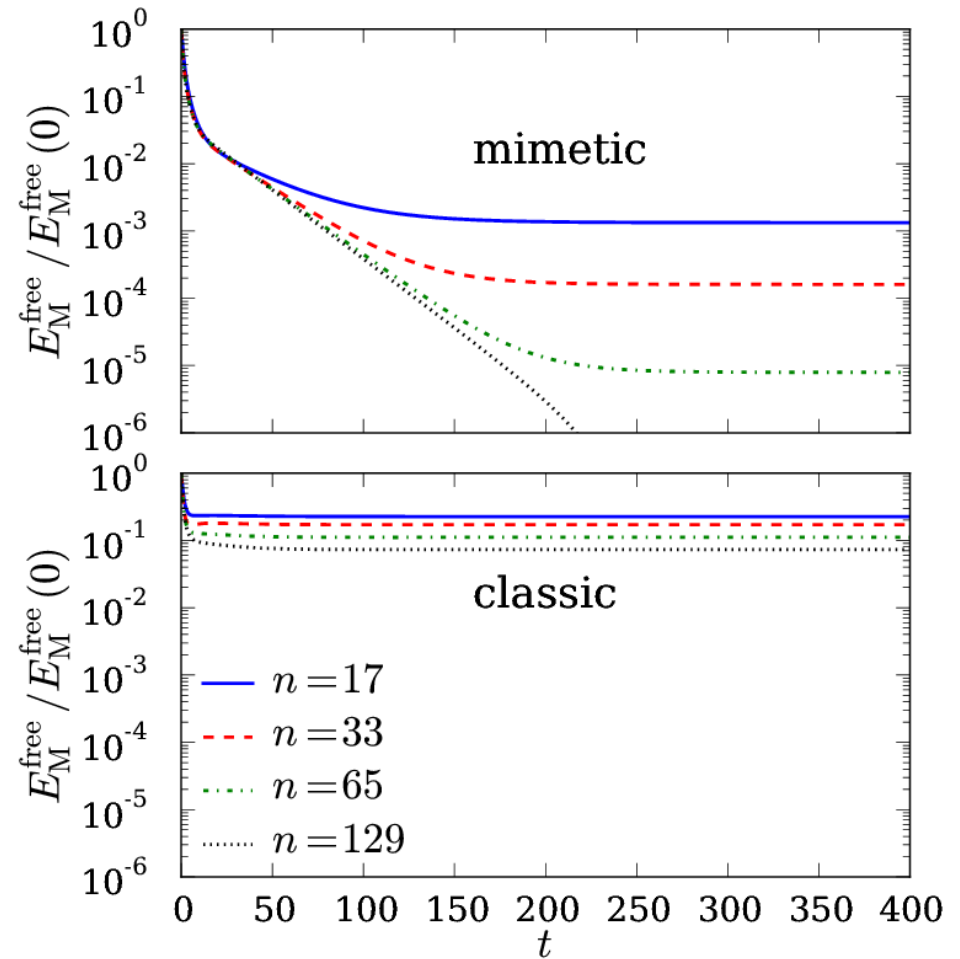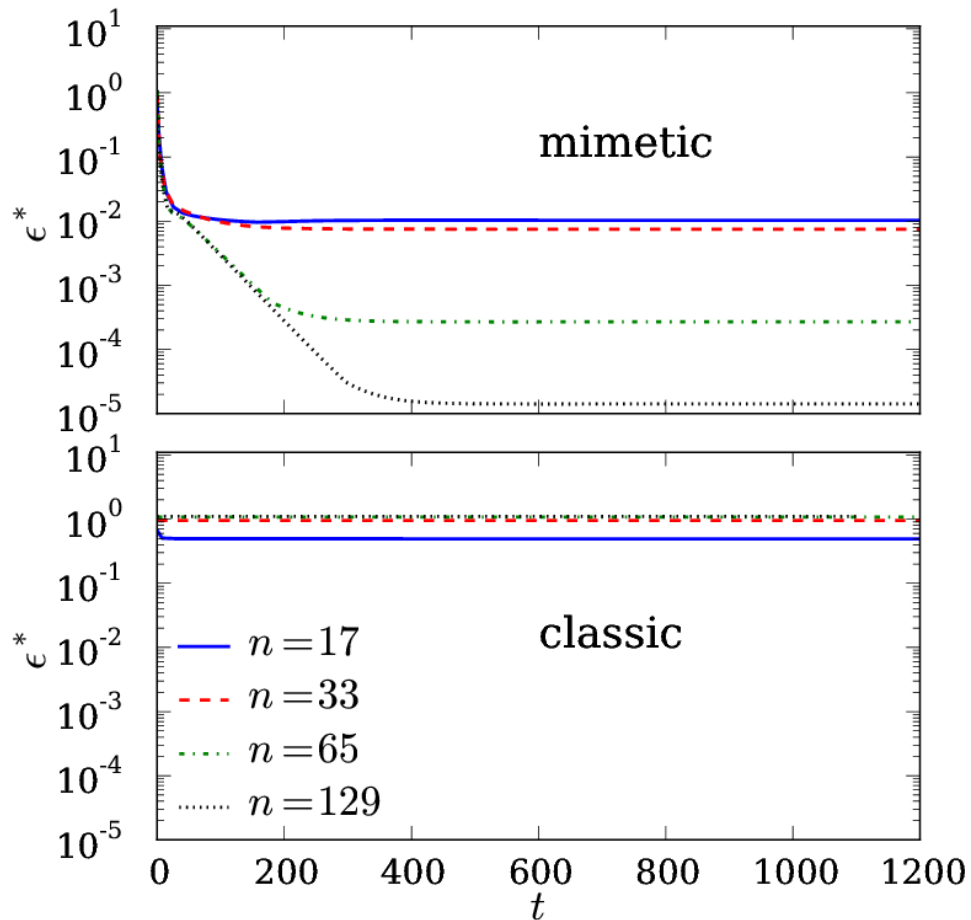

$t = 0 \qquad t = 10 \qquad t = 430$
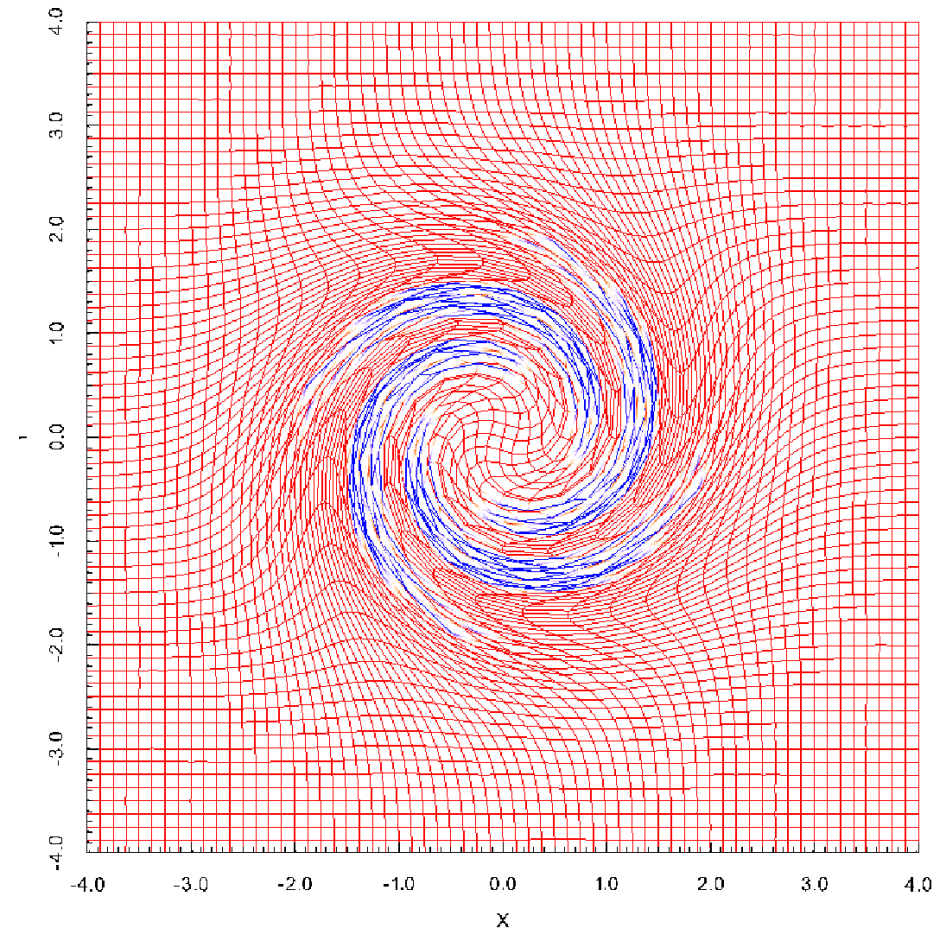
Relaxation Quality
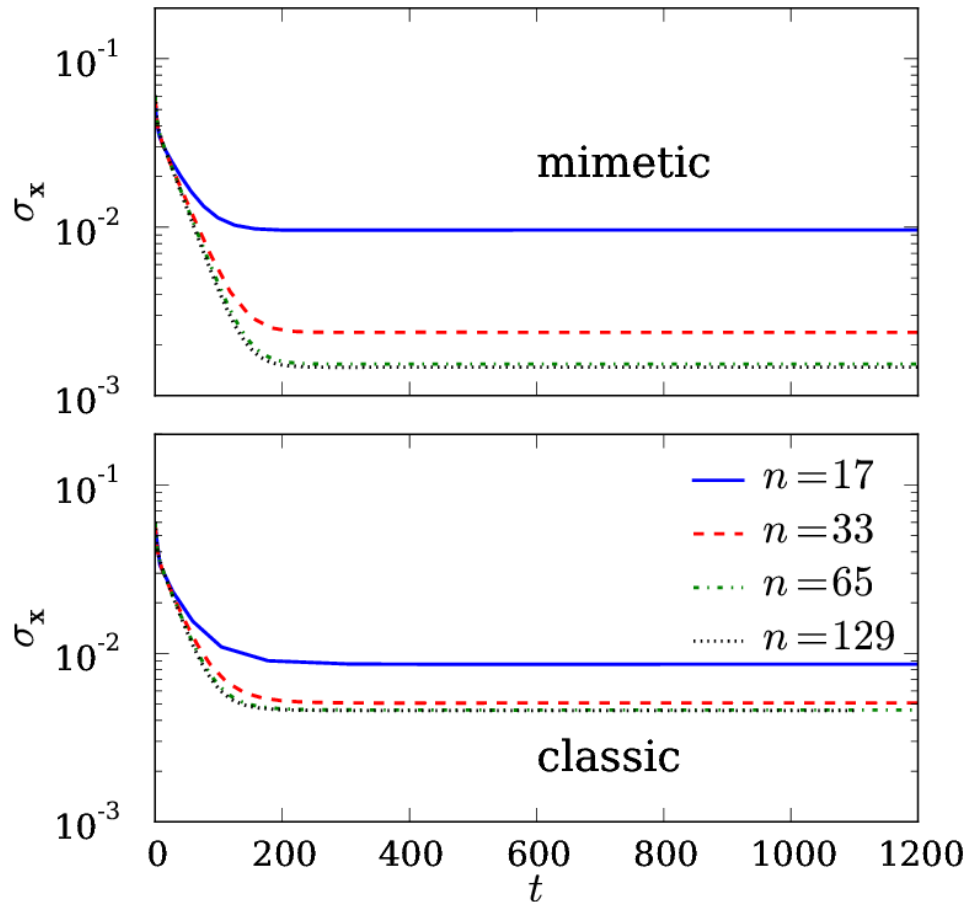
Closer to the analytical solution by 3 orders of magnitude.

Relaxation Quality

Closer to force-free state by 5 orders of magnitude.

Performance Gain

	mimetic vs. classic
floating point operations	1/2
computation time (gross)	1/2
previous code*	x100

*serial code using classical finite differences and an implicit solver

(Craig and Sneyd 1986)

Limitations
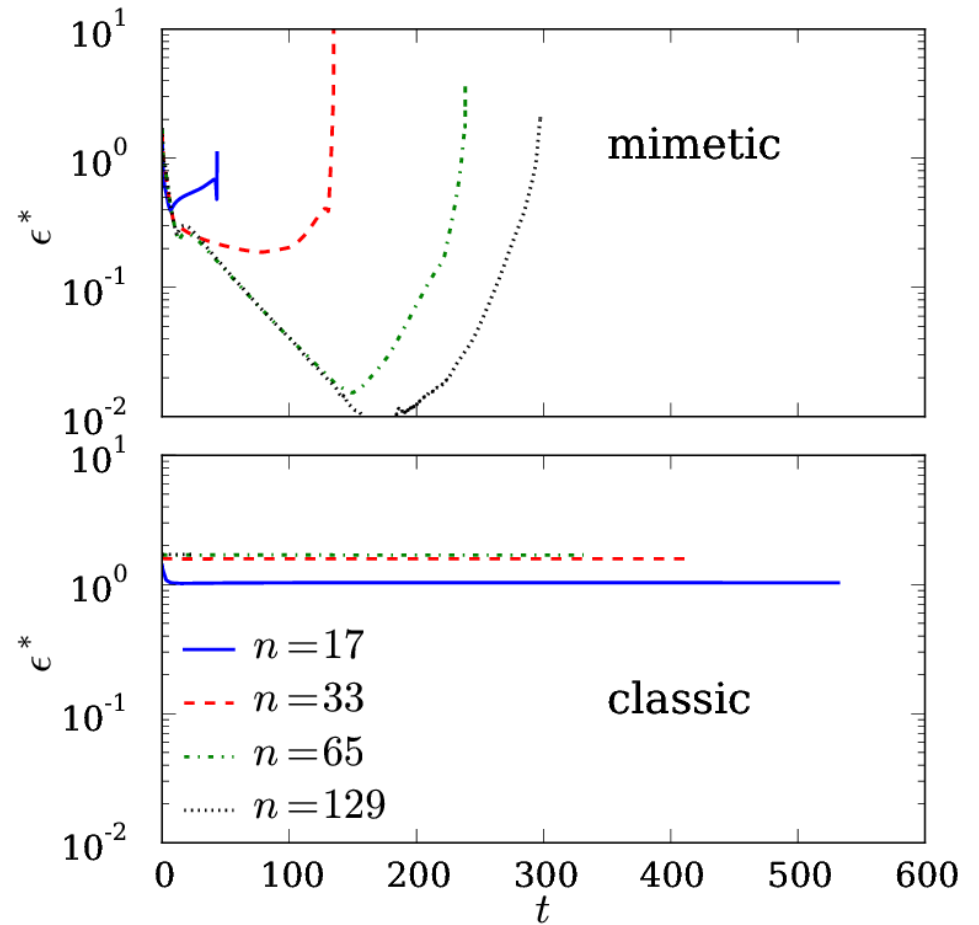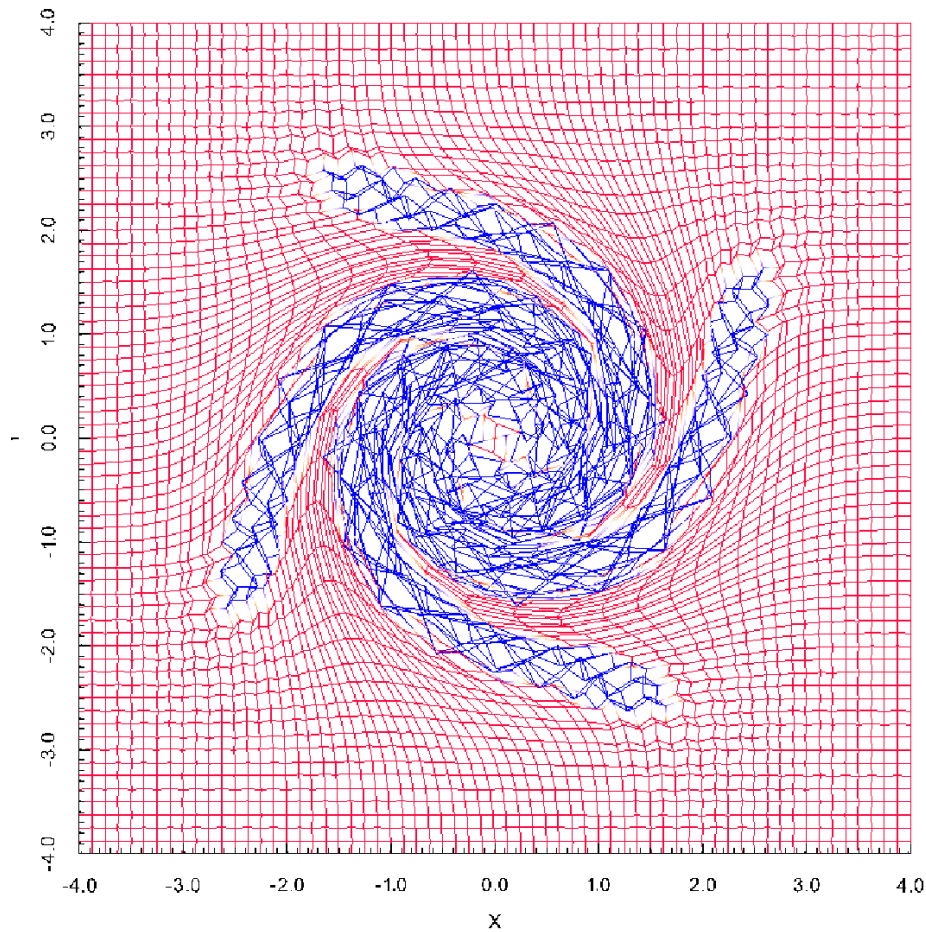
red: convex
blue: concave

For concave cells the method becomes unstable.
But: results before crash better than classic method.

Code Details

✸ written in C++

✸ 6th order Runge-Kutta time stepping

✸ running on GPUs

✸ periodic and line-tied boundaries

✸ VTK data format

✸ post processing routines in Python

```cpp
// compute the norm of JxB/B**2
__global__ void JxB_B2(REAL *B, REAL *J, REAL *JxB_B2, int dimX, int dimY, int dimZ) {
        int i = threadIdx.x + blockDim.x * blockIdx.x;
        int j = threadIdx.y + blockDim.y * blockIdx.y;
        int k = threadIdx.z + blockDim.z * blockIdx.z;
        int p = threadIdx.x;
        int q = threadIdx.y;
        int r = threadIdx.z;
        int l;
        REAL B2;

        // shared memory for faster communication, the size is assigned dynamically
        extern __shared__ REAL s[];
        REAL *Bs = s;                           // magnetic field
        REAL *Js = &s[3 * dimX * dimY * dimZ];       // electric current density
        REAL *JxBs = &Js[3 * dimX * dimY * dimZ];    // JxB

        // copy from global memory into shared memory
        if ((i < dev_p.nx) && (j < dev_p.ny) && (k < dev_p.nz)) {
                for (l = 0; l < 3; l++) {
                        Bs[l + p*3 + q*dimX*3 + r*dimX*dimY*3] = B[l + (i+1)*3 + (j+1)*(dev_p.nx+2)*3 + (k+1)*(dev_p.nx+2)*(dev_p.ny+2)*3];
                        Js[l + p*3 + q*dimX*3 + r*dimX*dimY*3] = J[l + i*3 + j*dev_p.nx*3 + k*dev_p.nx*dev_p.ny*3];
                }

                cross(&Js[0 + p*3 + q*dimX*3 + r*dimX*dimY*3],
                        &Bs[0 + p*3 + q*dimX*3 + r*dimX*dimY*3],
                        &JxBs[0 + p*3 + q*dimX*3 + r*dimX*dimY*3]);

                B2 = dot(&Bs[0 + p*3 + q*dimX*3 + r*dimX*dimY*3], &Bs[0 + p*3 + q*dimX*3 + r*dimX*dimY*3]);

                // return result into global memory
                JxB_B2[i + j*dev_p.nx + k*dev_p.nx*dev_p.ny] = norm(&JxBs[0 + p*3 + q*dimX*3 + r*dimX*dimY*3])/B2;
        }
}
```

16

User Interface

Fortran name lists

```
&comp
    nx = 33;  ny = 33;  nz = 33
/

&start
    Lx = 0.6;      Ly = 0.6;      Lz = 1.0
    Ox = -0.3;     Oy = -0.3;     Oz = -0.5
    bInit = "sheared"
    ampl = 1.
    initDist = "initShearX"
    initShear0 = 0.7
    initShearK = 1.
    fRestart = t
/
```

Bash commands

```
gm_ci_run
gm_inspectrun
gm_newrun
```

time_series.dat

#	it	t	dt	maxDelta	JxB_B2Max	epsilonStar	B2	convex
	0	1.29540e-06	1.29540e-06	1.78814e-07	1.50932e+02	2.81160e+02	7.12394e-01	-1.00000e+00
	1	3.08508e-06	1.78967e-06	1.19209e-07	1.13175e+02	2.96738e+02	7.12170e-01	-1.00000e+00
	2	5.76647e-06	2.68139e-06	1.78814e-07	8.83429e+01	3.15884e+02	7.11882e-01	-1.00000e+00
	3	9.47096e-06	3.70449e-06	1.19209e-07	7.67879e+01	3.36120e+02	7.11536e-01	-1.00000e+00
	4	1.50212e-05	5.55028e-06	1.78814e-07	6.44194e+01	3.57402e+02	7.11085e-01	-1.00000e+00
	5	2.13638e-05	6.34253e-06	7.74860e-07	5.44002e+01	3.73753e+02	7.10636e-01	-1.00000e+00

Post-Processing
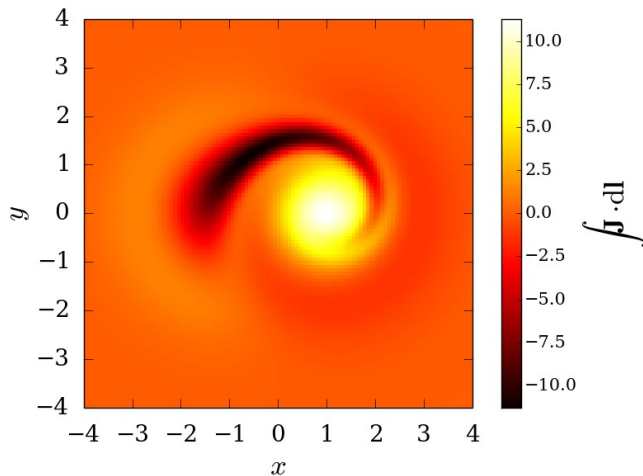
streamlines
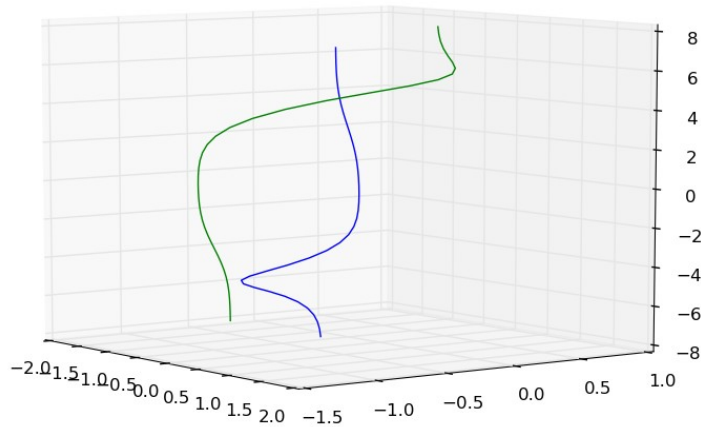
field line mapping
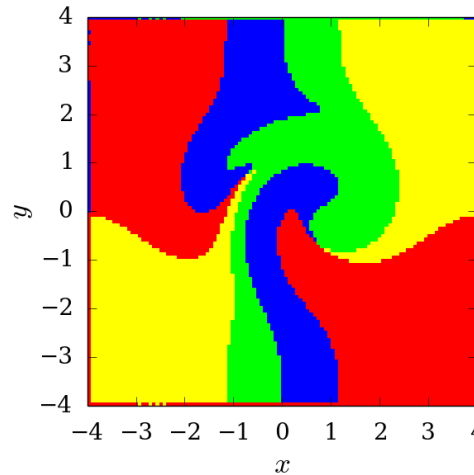
Poincaré maps
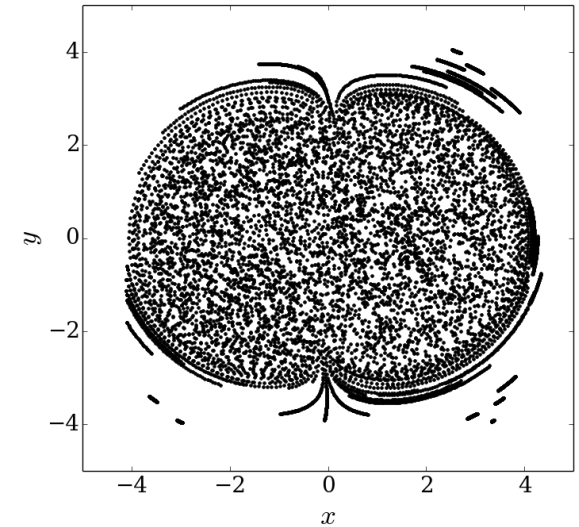
line integration

save and read as vtk file
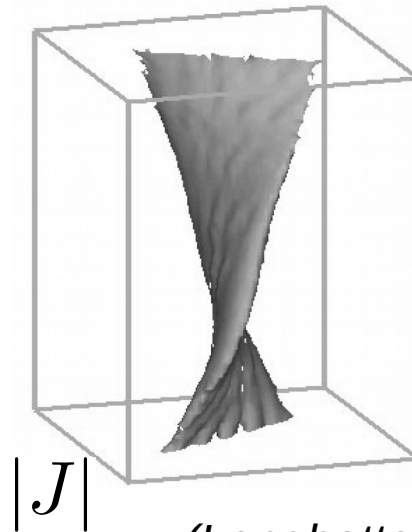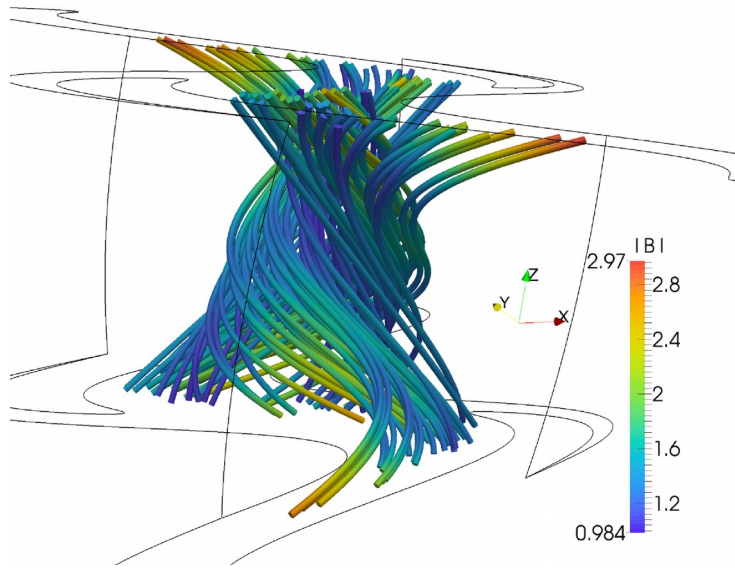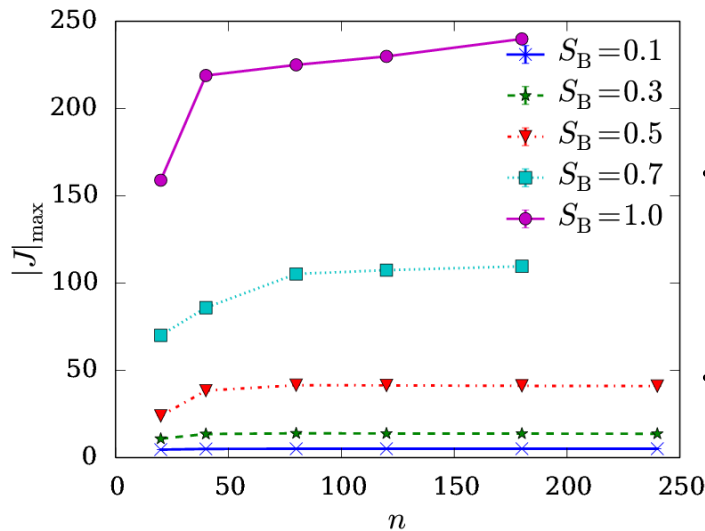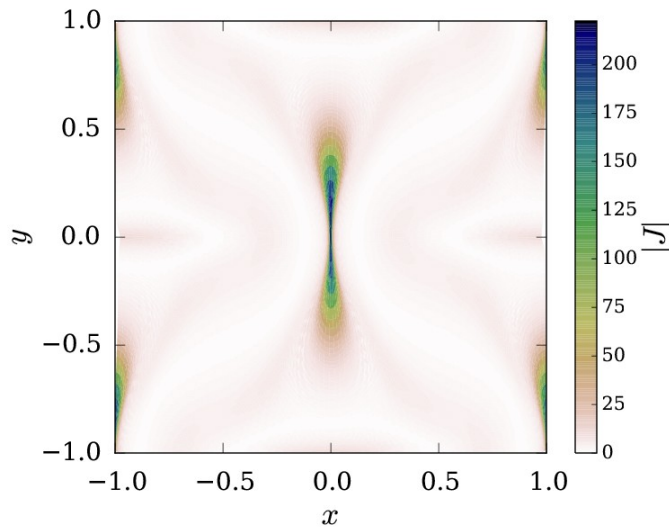
```
s0 = gm.streamInit(tol = 0.01)
```

stream.vtk
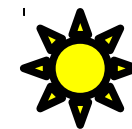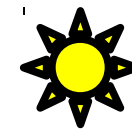
```
sr = gm.readStream()
```

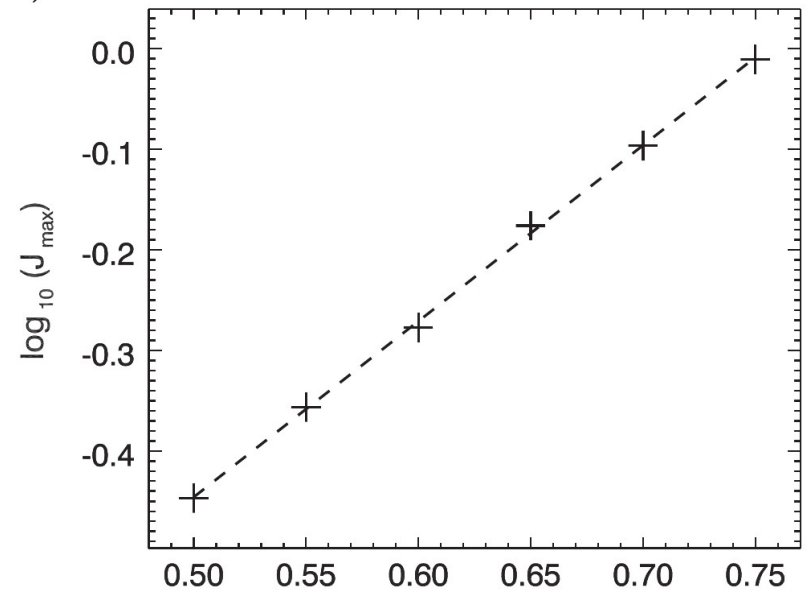# Distorted Magnetic Fields

$|J|$

(Longbottom 1998)
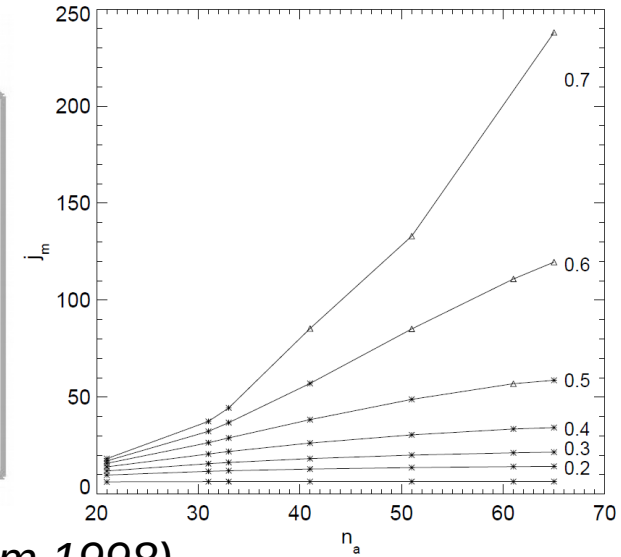
resolved current concentrations

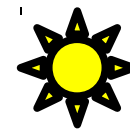shear leads to strong currents

(Candelaresi et al. 2015)

19

Exponential Increase in Current
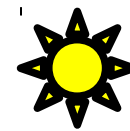

$|\mathbf{J}|$
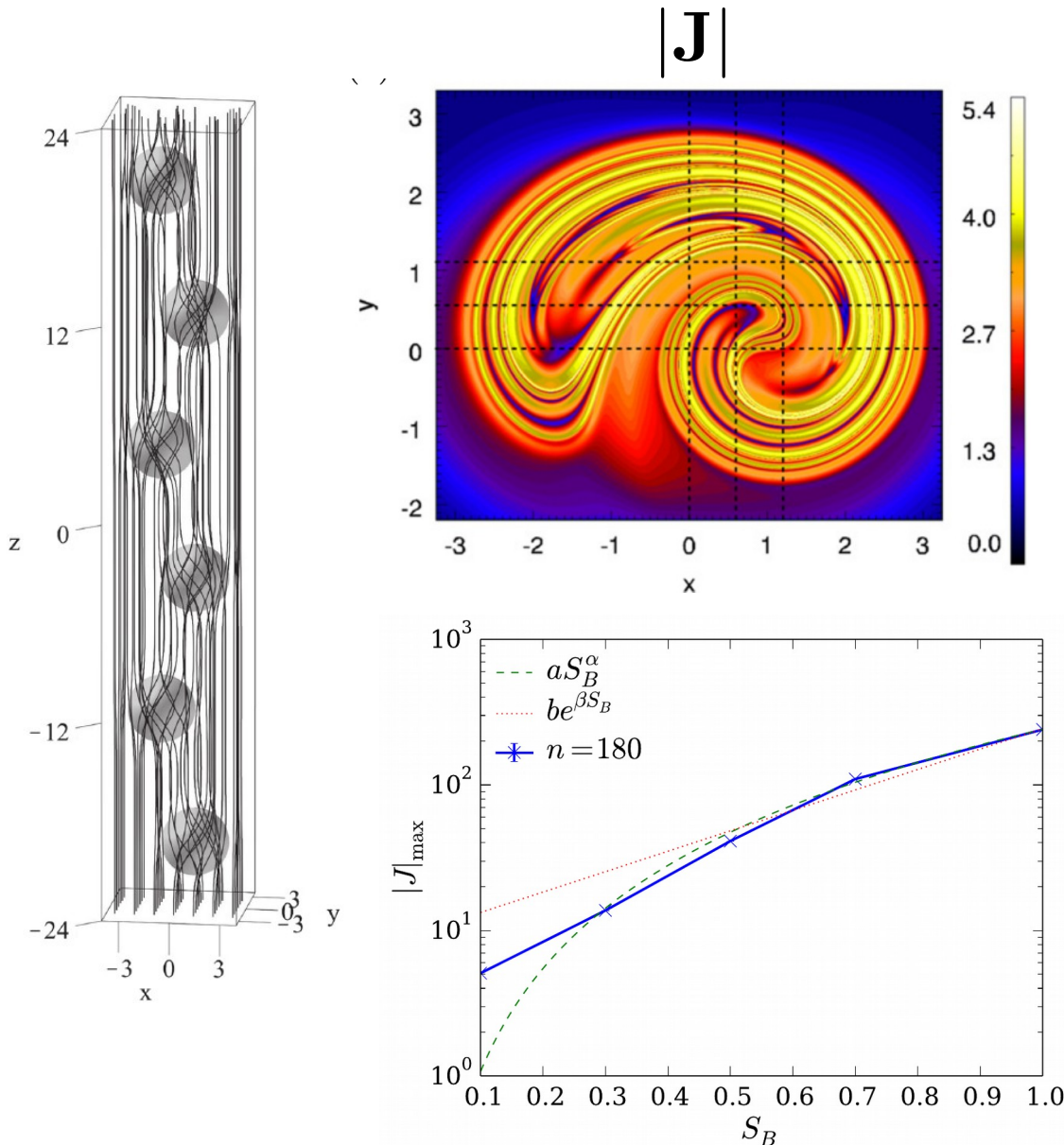
(Pontin 2015)
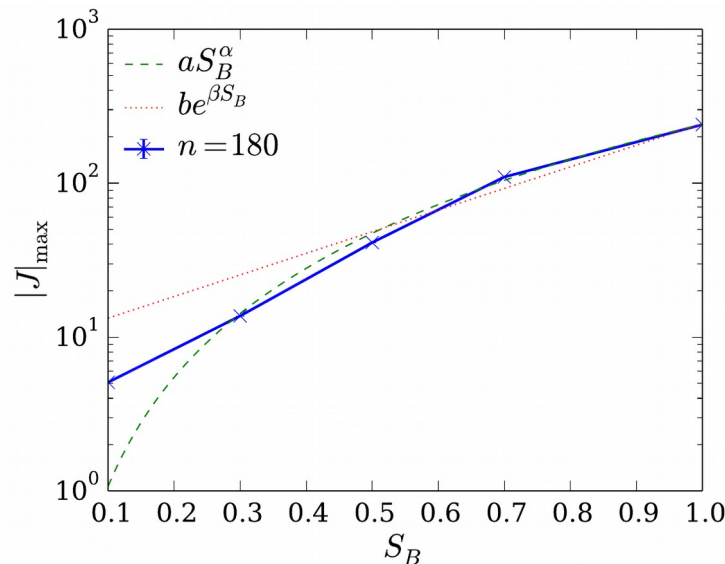
(Candelaresi 2015)

☀ lengths decrease exp. with complexity

☀ current increases exp. with complexity

Conclusions

- Topology preserving relaxation of magnetic fields.

- GLEMuR: (https://github.com/SimonCan/glemur)

- Current concentrations not singular.

- Current increases strongly with field complexity.

simon.candelaresi@gmail.com