A Fully Lagrangian Advection Scheme

M. Ali Yassaei, John C. Bowman, and Anup Basu
University of Alberta

Dec 6, 2008

www.math.ualberta.ca/∼bowman/talks

Outline

- 2D Advection–Diffusion
- Passive Advection
- Casimir Invariants
- Lagrangian Rearrangement
  - Weighted Bresenham Algorithm
- Average Complexity
- Operator Splitting
  - Diffusion
  - Self-advection
- Energy Decay Rate
- Conclusions

Introduction

- 2D advection–diffusion:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{U} = \boldsymbol{D} \nabla^2 \boldsymbol{U}.$$

Introduction

- 2D advection–diffusion:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{U} = \boldsymbol{D} \nabla^2 \boldsymbol{U}.$$

- $\boldsymbol{U} = (\omega, C)$ represents
  - Scalar vorticity $\omega = \hat{\boldsymbol{z}} \cdot \boldsymbol{\nabla} \times \boldsymbol{v}$,
  - Concentration field C.

Introduction

- 2D advection–diffusion:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{U} = \boldsymbol{D}\nabla^2\boldsymbol{U}.$$

- $\boldsymbol{U} = (\omega, C)$ represents
  - Scalar vorticity $\omega = \hat{\boldsymbol{z}}\cdot\boldsymbol{\nabla}\times\boldsymbol{v}$,
  - Concentration field C.
- The velocity $\boldsymbol{v}$ is incompressible: $\boldsymbol{\nabla}\cdot\boldsymbol{v} = 0$.

Introduction

- 2D advection–diffusion:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\boldsymbol{U} = \boldsymbol{D}\nabla^2\boldsymbol{U}.$$

- $\boldsymbol{U} = (\omega, C)$ represents
  - Scalar vorticity $\omega = \hat{\boldsymbol{z}}\boldsymbol{\cdot}\boldsymbol{\nabla}\times\boldsymbol{v}$,
  - Concentration field C.
- The velocity $\boldsymbol{v}$ is incompressible: $\boldsymbol{\nabla}\boldsymbol{\cdot}\boldsymbol{v} = 0$.
- Diffusion matrix $\boldsymbol{D} = \mathrm{diag}(\nu, D)$:

  ν = fluid viscosity,

  D = diffusion constant for concentration field.

Eulerian vs. Lagrangian

- Passive advection without diffusion:

$$\frac{\partial C}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} C = 0.$$

Eulerian vs. Lagrangian

- Passive advection without diffusion:

$$\frac{\partial C}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} C = 0.$$

- Finite difference:

$$\frac{C_i^{n+1} - C_i^n}{\tau} = -v \frac{C_{i+1}^n - C_{i-1}^n}{2h}.$$

Eulerian vs. Lagrangian

- Passive advection without diffusion:

$$\frac{\partial C}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} C = 0.$$

- Finite difference:

$$\frac{C_i^{n+1} - C_i^n}{\tau} = -v \frac{C_{i+1}^n - C_{i-1}^n}{2h}.$$

- Problems with Eulerian methods:
  - Instability;
  - Upwinding and Lax schemes: numerical diffusion.

Method of Characteristics

- Solution to passive advection problem without diffusion:

$$C(\boldsymbol{x}, t) = C\left(\boldsymbol{\xi}_0(\boldsymbol{x}, t), 0\right).$$

Method of Characteristics

- Solution to passive advection problem without diffusion:

$$C(\boldsymbol{x}, t) = C\left(\boldsymbol{\xi}_0(\boldsymbol{x}, t), 0\right).$$

- Introduce Lagrangian position

$$\boldsymbol{\xi}(t) = \boldsymbol{\xi}_0 + \int_0^t \boldsymbol{v}(\boldsymbol{\xi}(\tau), \tau)\, d\tau,$$

where $\boldsymbol{\xi}(t) = \boldsymbol{x}$ and $\boldsymbol{\xi}_0 = \boldsymbol{\xi}_0(\boldsymbol{x}, t)$ is the initial parcel position.

Method of Characteristics

- Solution to passive advection problem without diffusion:

$$C(\boldsymbol{x}, t) = C\left(\boldsymbol{\xi}_0(\boldsymbol{x}, t), 0\right).$$

- Introduce Lagrangian position

$$\boldsymbol{\xi}(t) = \boldsymbol{\xi}_0 + \int_0^t \boldsymbol{v}(\boldsymbol{\xi}(\tau), \tau)\, d\tau,$$

where $\boldsymbol{\xi}(t) = \boldsymbol{x}$ and $\boldsymbol{\xi}_0 = \boldsymbol{\xi}_0(\boldsymbol{x}, t)$ is the initial parcel position.

- Problem of viewing solution on grid: new Lagrangian positions may not lie on grid points.

Method of Characteristics

- Solution to passive advection problem without diffusion:

$$C(\boldsymbol{x}, t) = C\left(\boldsymbol{\xi}_0(\boldsymbol{x}, t), 0\right).$$

- Introduce Lagrangian position

$$\boldsymbol{\xi}(t) = \boldsymbol{\xi}_0 + \int_0^t \boldsymbol{v}(\boldsymbol{\xi}(\tau), \tau)\, d\tau,$$

where $\boldsymbol{\xi}(t) = \boldsymbol{x}$ and $\boldsymbol{\xi}_0 = \boldsymbol{\xi}_0(\boldsymbol{x}, t)$ is the initial parcel position.

- Problem of viewing solution on grid: new Lagrangian positions may not lie on grid points.

- Solutions:

  – interpolate (semi-Lagrangian): numerical diffusion;

  – Lagrangian rearrangement: project advected parcel centroids onto rearrangment manifold.

Casimir Invariants

- Conservation equation:

$$\frac{dC(\boldsymbol{x}(t),t)}{dt} = \frac{d\boldsymbol{x}}{dt}\boldsymbol{\cdot}\boldsymbol{\nabla} C + \frac{\partial C}{\partial t} = \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla} C + \frac{\partial C}{\partial t} = 0,$$

where $\boldsymbol{v} = d\boldsymbol{x}/dt$.

Casimir Invariants

- Conservation equation:

$$\frac{dC(\boldsymbol{x}(t),t)}{dt} = \frac{d\boldsymbol{x}}{dt}\boldsymbol{\cdot}\boldsymbol{\nabla} C + \frac{\partial C}{\partial t} = \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla} C + \frac{\partial C}{\partial t} = 0,$$

where $\boldsymbol{v} = d\boldsymbol{x}/dt$.

- For any C^1 function f of concentration (or vorticity) field:

$$\frac{d}{dt}\int f(C)\, d\boldsymbol{x} = \int f'(C)\frac{\partial C}{\partial t}\, d\boldsymbol{x} = -\int f'(C)\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla} C\, d\boldsymbol{x}$$
$$= -\int \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla} f(C)\, d\boldsymbol{x} = \int f(C)\boldsymbol{\nabla}\boldsymbol{\cdot}\boldsymbol{v}\, d\boldsymbol{x} = 0.$$

Casimir Invariants

- Conservation equation:

$$\frac{dC(\boldsymbol{x}(t),t)}{dt} = \frac{d\boldsymbol{x}}{dt}\boldsymbol{\cdot}\boldsymbol{\nabla}C + \frac{\partial C}{\partial t} = \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}C + \frac{\partial C}{\partial t} = 0,$$

where $\boldsymbol{v} = d\boldsymbol{x}/dt$.

- For any C^1 function f of concentration (or vorticity) field:

$$\begin{aligned}\frac{d}{dt}\int f(C)\,d\boldsymbol{x} &= \int f'(C)\frac{\partial C}{\partial t}\,d\boldsymbol{x} = -\int f'(C)\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}C\,d\boldsymbol{x} \\ &= -\int \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}f(C)\,d\boldsymbol{x} = \int f(C)\boldsymbol{\nabla}\boldsymbol{\cdot}\boldsymbol{v}\,d\boldsymbol{x} = 0.\end{aligned}$$

- Enforce a discrete analog of this exact infinitesimal property:

$$\frac{d}{dt}\sum_{i,j} f(C_{i,j}) = 0.$$

Parcel Centroid

- Advection map is continuous and area-preserving $\Rightarrow$ rearrangement into distinct nonoverlapping parcels.

Parcel Centroid

- Advection map is continuous and area-preserving $\Rightarrow$ rearrangement into distinct nonoverlapping parcels.
- Represent solution as finite union of piecewise constant functions.

Parcel Centroid

- Advection map is continuous and area-preserving $\Rightarrow$ rearrangement into distinct nonoverlapping parcels.
- Represent solution as finite union of piecewise constant functions.
- The resulting discrete constraint

$$\frac{d}{dt}\sum_{i,j} f(C_{i,j}) = 0$$

is equivalent to imposing parcel rearrangement.

Parcel Centroid

- Advection map is continuous and area-preserving $\Rightarrow$ rearrangement into distinct nonoverlapping parcels.
- Represent solution as finite union of piecewise constant functions.
- The resulting discrete constraint

$$\frac{d}{dt}\sum_{i,j} f(C_{i,j}) = 0$$

  is equivalent to imposing parcel rearrangement.
- Use RK4 to advect the parcel centroids.

Parcel Centroid

- Advection map is continuous and area-preserving $\Rightarrow$ rearrangement into distinct nonoverlapping parcels.
- Represent solution as finite union of piecewise constant functions.
- The resulting discrete constraint

$$\frac{d}{dt}\sum_{i,j} f(C_{i,j}) = 0$$

  is equivalent to imposing parcel rearrangement.
- Use RK4 to advect the parcel centroids.
- Under this linear map, parcel centroid maps to advected parcel centroid.

Parcel Centroid

- Advection map is continuous and area-preserving $\Rightarrow$ rearrangement into distinct nonoverlapping parcels.
- Represent solution as finite union of piecewise constant functions.
- The resulting discrete constraint

$$\frac{d}{dt}\sum_{i,j} f(C_{i,j}) = 0$$

is equivalent to imposing parcel rearrangement.

- Use RK4 to advect the parcel centroids.
- Under this linear map, parcel centroid maps to advected parcel centroid.
- For passive advection without diffusion: only evolve parcel centroids (no need to actually evolve the quadrilateral vertices).

Lagrangian→Eulerian Projection

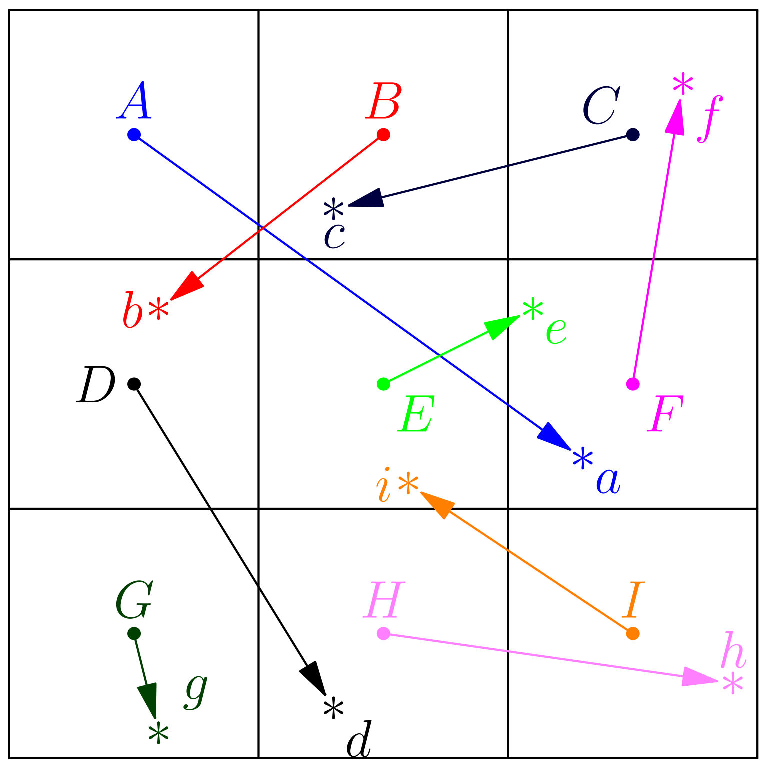
- Advection in Lagrangian frame ⇒ piles and holes.

Lagrangian$\longrightarrow$Eulerian Projection

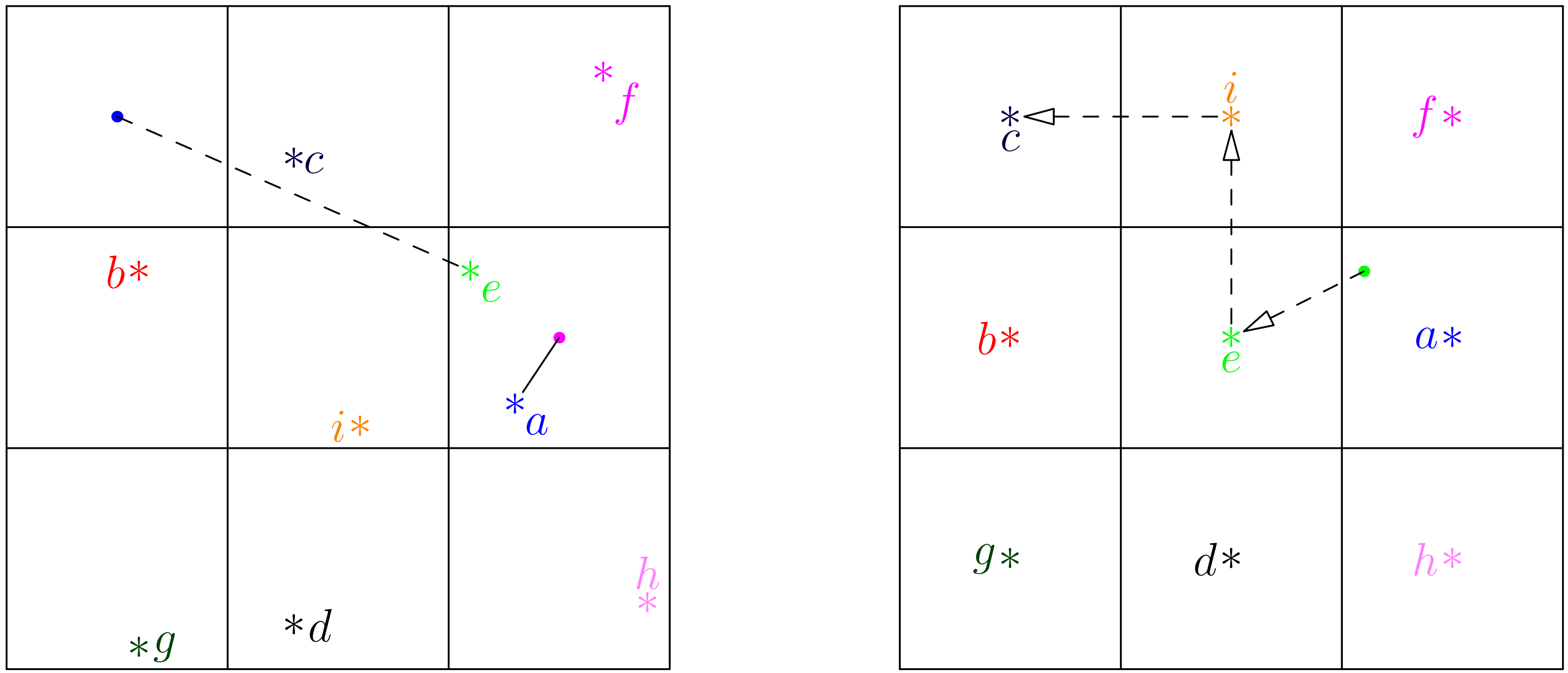
- Advection in Lagrangian frame $\Rightarrow$ piles and holes.
- New state must be a rearrangement of initial state to conserve Casimir invariants.

Lagrangian→Eulerian Projection

- Advection in Lagrangian frame ⇒ piles and holes.
- New state must be a rearrangement of initial state to conserve Casimir invariants.
- How to map excess parcels (*) to holes?

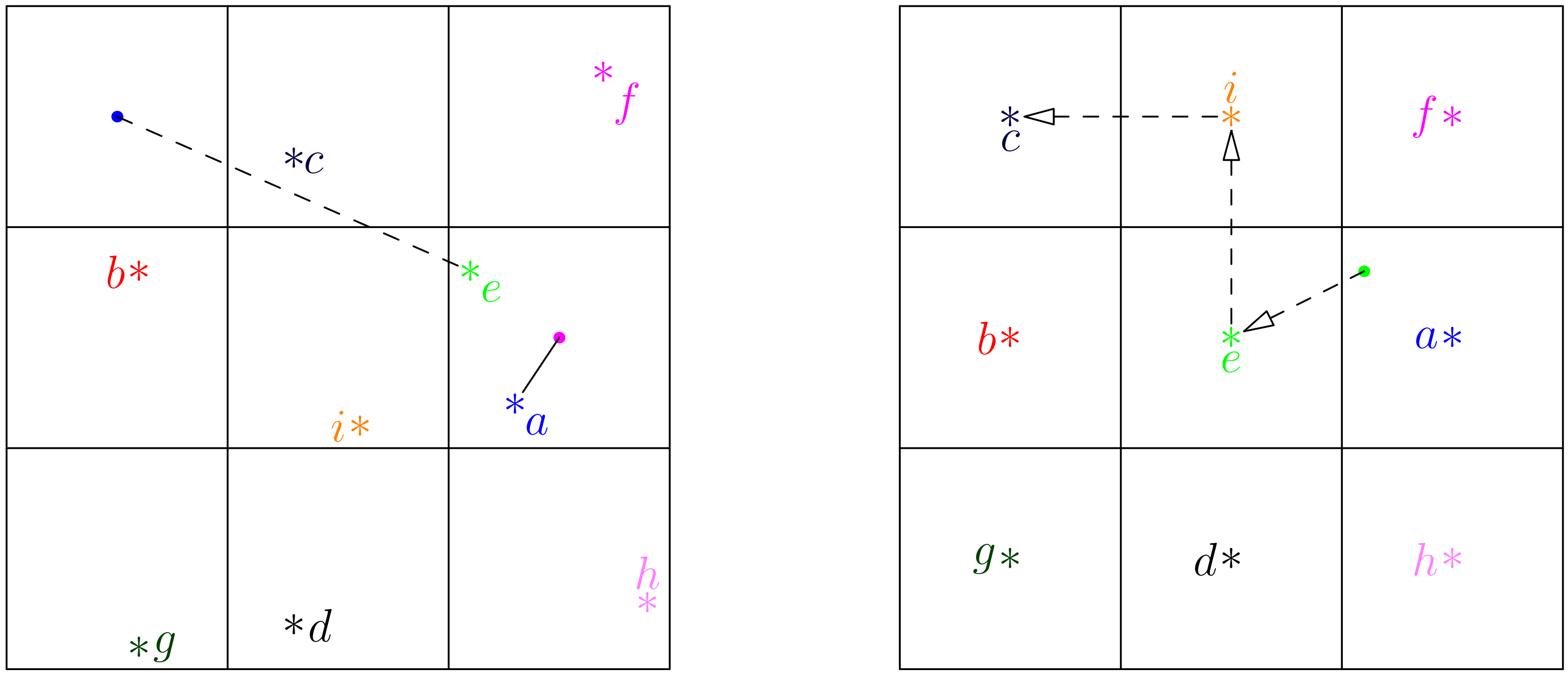


Lagrangian Rearrangement



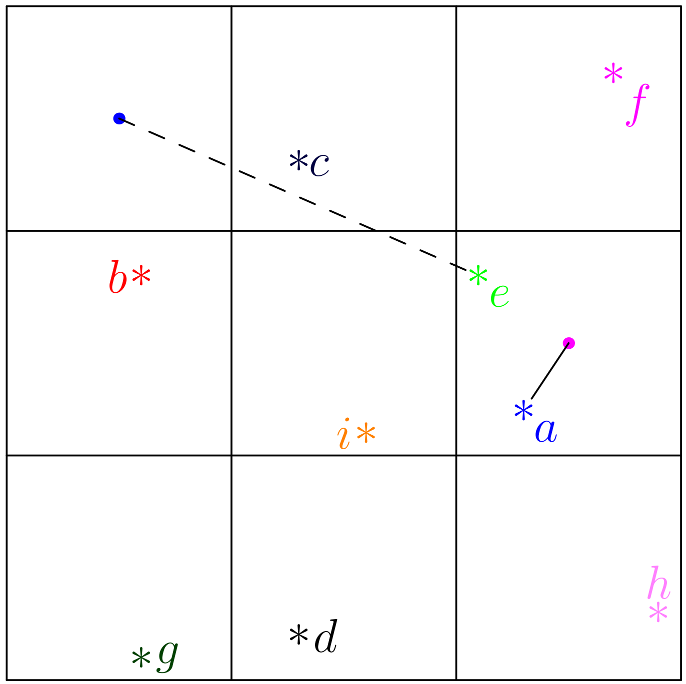
- Start with cells with most parcels.

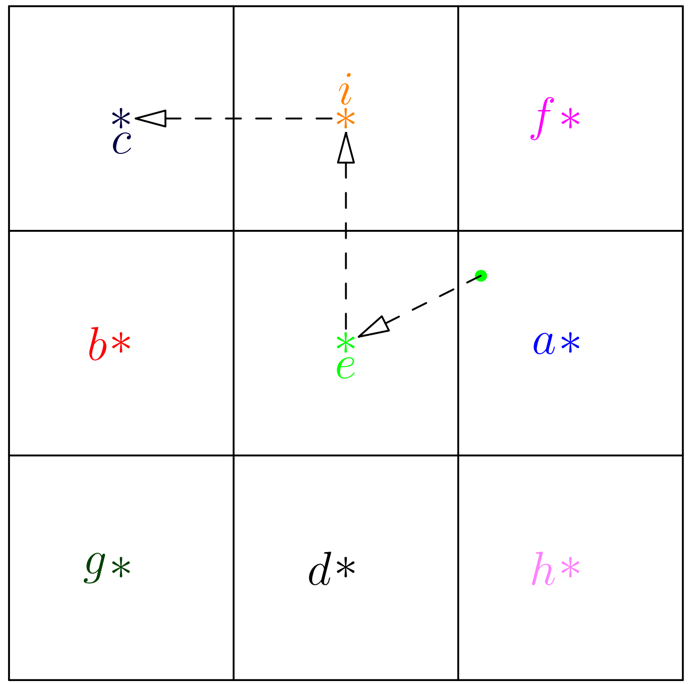
Lagrangian Rearrangement



- Start with cells with most parcels.
- Find nearest hole (search in rectangular shells about pile).

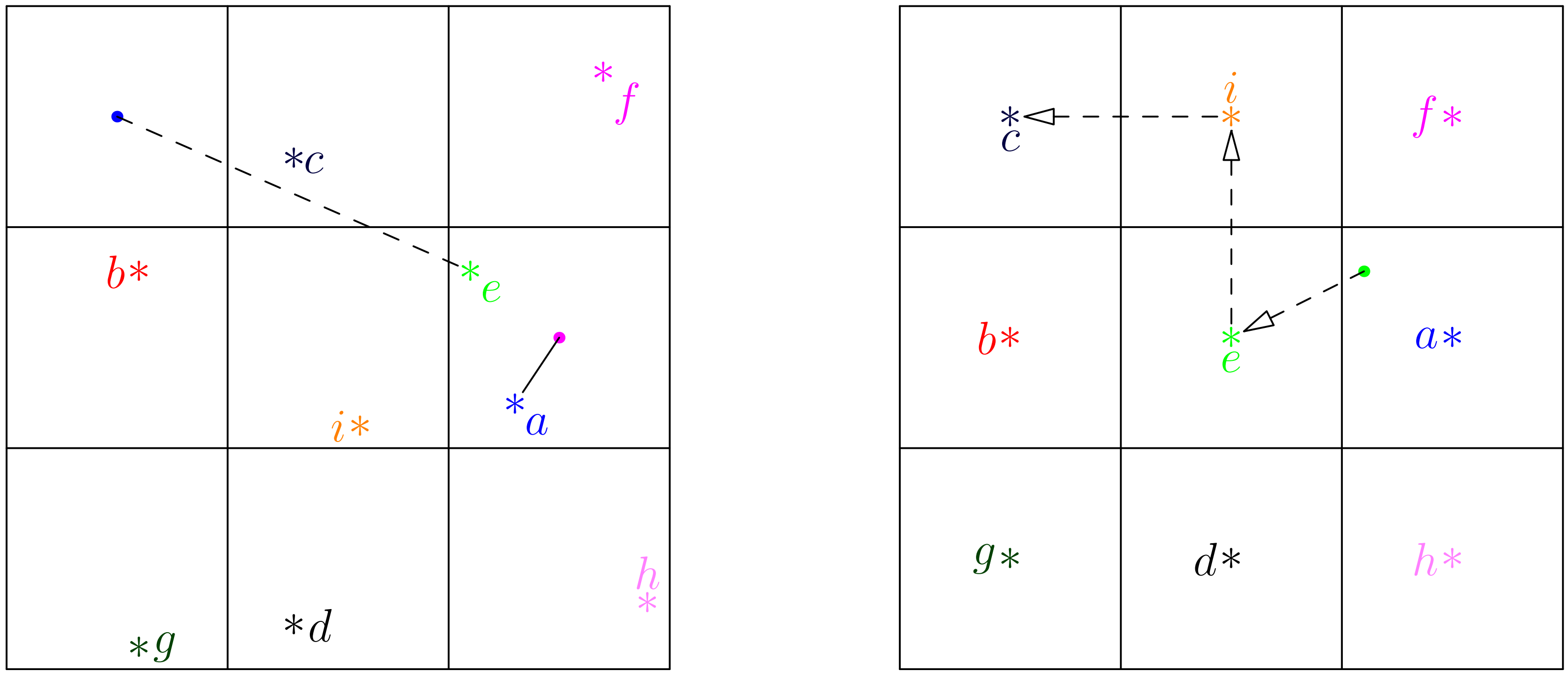
Lagrangian Rearrangement





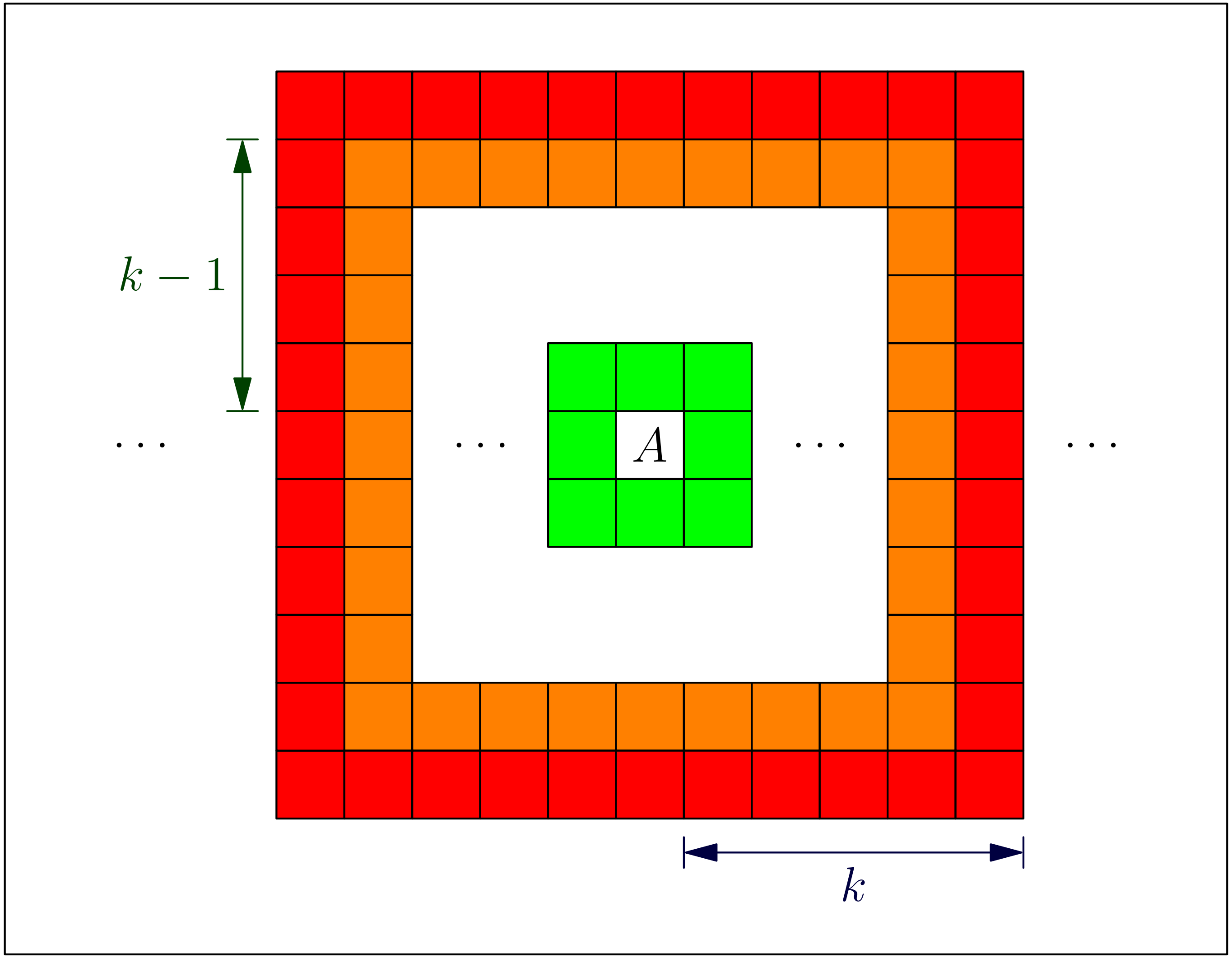
- Start with cells with most parcels.
- Find nearest hole (search in rectangular shells about pile).
- Discretize path from pile to hole.

Lagrangian Rearrangement



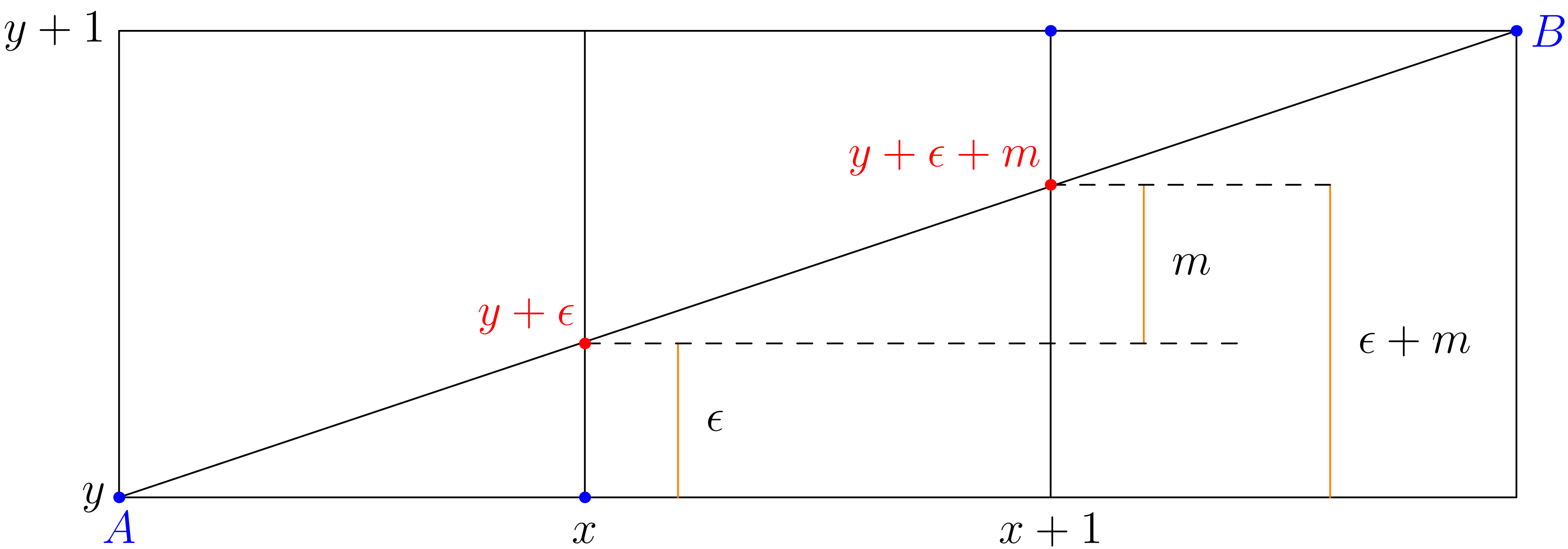
- Start with cells with most parcels.
- Find nearest hole (search in rectangular shells about pile).
- Discretize path from pile to hole.
- Push chain of parcels toward hole.

Searching kth Rectangular Shell



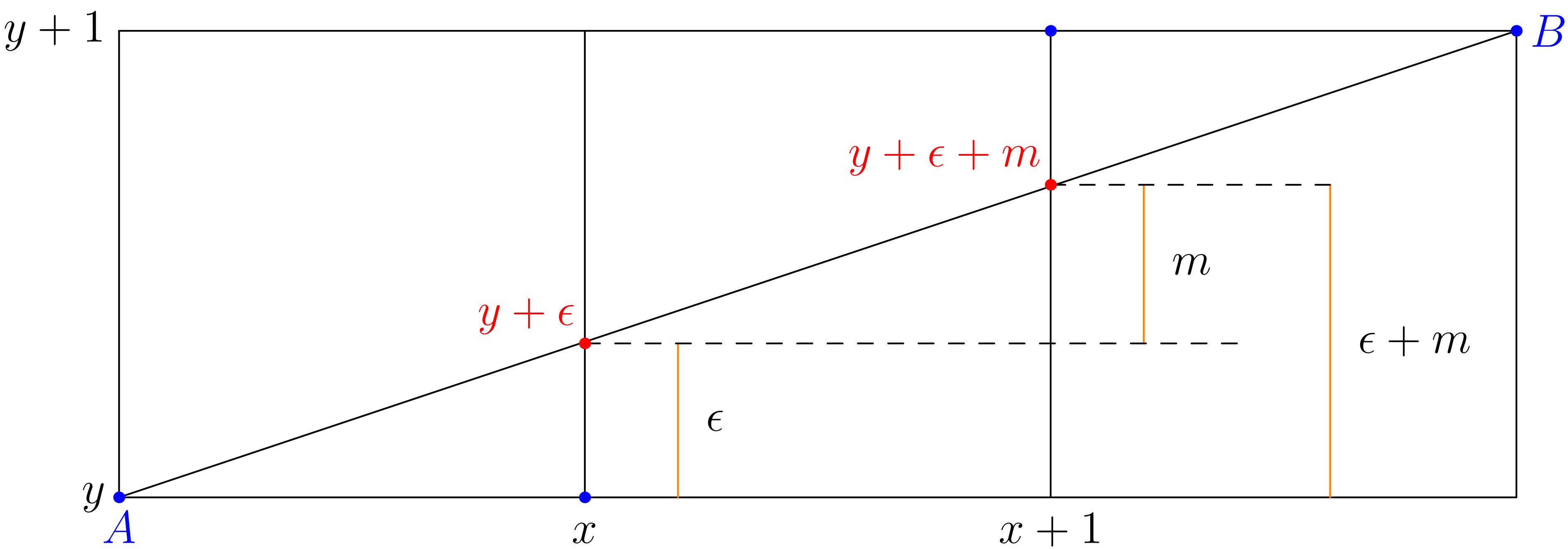
Bresenham Algorithm

- Discretize path from pile to hole:
  - Reduce to case $0 \leq m \leq 1$.
  - Choose $(x+1, y)$ or $(x+1, y+1)$.



Bresenham Algorithm

- Discretize path from pile to hole:
  - Reduce to case $0 \leq m \leq 1$.
  - Choose $(x+1, y)$ or $(x+1, y+1)$.



- Problem: multiple pushing of parcels $\Rightarrow$ visible streaks.

Weighted Bresenham Algorithm

- Randomize path:

0	1	8	4	2	3	7	6	9	4	4	6	3	0	4	1	8	3	5	2	4
5	8	3	3	8	9	3	3	7	9	5	4	4	8	4	1	1	4	3	6	3
2	0	0	9	8	7	2	4	5	1	4	1	7	2	6	7	3	6	5	0	6
4	1	6	6	7	0	3	9	2	5	9	2	7	7	4	6	6	5	1	6	2
7	3	5	0	4	1	3	0	4	2	5	9	9	4	5	6	0	2	9	3	6
0	8	9	8	9	8	5	1	1	6	2	7	6	4	4	0	6	2	8	5	5
7	6	5	9	1	2	3	5	3	2	4	2	5	0	7	2	0	8	6	5	5
4	5	1	5	4	7	2	6	3	8	3	3	9	5	1	1	3	2	6	3	3
9	7	1	9	0	9	6	6	9	5	6	5	8	8	9	5	6	6	5	6	0
4	8	9	3	2	0	1	1	8	8	1	8	2	4	6	5	7	8	7	8	0
1	0	6	3	9	2	5	4	7	4	5	9	6	5	5	0	2	8	4	7	9
6	8	3	8	0	9	7	0	2	6	0	9	3	9	1	6	0	0	9	0	4
6	4	0	4	8	8	0	0	6	0	2	6	4	2	1	8	2	0	8	1	1
1	8	6	6	6	3	2	9	2	6	1	4	6	8	0	4	2	6	5	3	2
7	1	5	4	2	8	8	2	2	3	4	8	4	5	1	4	3	0	9	1	3
0	7	6	3	4	0	1	0	5	4	2	3	7	8	9	6	3	8	7	2	0
6	7	7	0	9	7	8	7	3	4	4	3	1	2	1	4	7	4	9	7	9
8	3	3	9	7	5	0	4	9	1	1	3	6	1	5	3	8	6	8	1	2
9	7	8	6	0	6	0	9	5	8	7	1	4	0	1	6	7	4	9	4	8
7	1	9	3	9	7	0	8	6	0	0	9	8	0	9	3	3	6	5	7	2
4	5	9	6	8	7	4	4	4	1	0	1	3	4	4	7	5	7	2	3	0
8	7	1	7	3	5	4	8	5	3	8	3	2	6	0	5	0	7	4	5	3
2	3	7	3	6	8	3	7	8	3	9	9	8	1	8	4	9	3	9	1	4
2	0	4	6	2	9	7	9	0	6	8	1	1	1	0	2	8	1	4	1	7
1	9	4	6	5	1	9	3	7	7	5	5	8	4	3	5	4	6	7	0	8
9	0	4	5	2	2	4	4	0	5	5	9	9	2	6	5	5	3	5	2	2
1	7	3	2	5	6	8	2	0	6	1	3	9	9	8	2	6	2	2	6	6
1	7	4	2	9	5	8	3	4	9	7	9	3	5	9	1	9	1	0	3	7
0	1	3	1	6	9	2	9	6	5	9	3	0	3	3	8	0	8	1	7	9
1	6	5	9	3	1	1	4	7	5	5	1	6	0	8	3	4	6	0	3	9
3	6	7	7	7	5	3	5	0	5	7	2	3	1	4	1	8	2	8	3	3

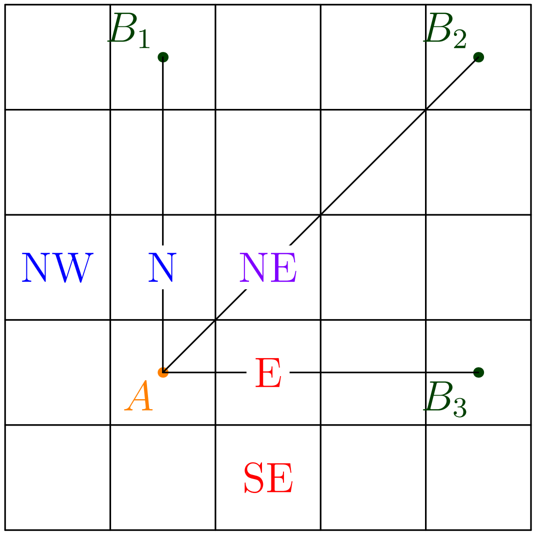
Weighted Bresenham Algorithm

- Randomize path:

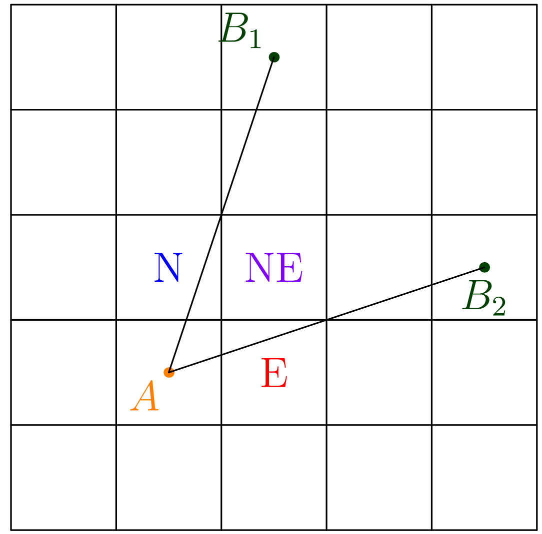
0	1	8	4	2	3	7	6	9	4	4	6	3	0	4	1	8	3	5	2	4
5	8	3	3	8	9	3	3	7	9	5	4	4	8	4	1	1	4	3	6	3
2	0	0	9	8	7	2	4	5	1	4	1	7	2	6	7	3	6	5	0	6
4	1	6	6	7	0	3	9	2	5	9	2	7	7	4	6	6	5	1	6	2
7	3	5	0	4	1	3	0	4	2	5	9	9	4	5	6	0	2	9	3	6
0	8	9	8	9	8	5	1	1	6	2	7	6	4	4	0	6	2	8	5	5
7	6	5	9	1	2	3	5	3	2	4	2	5	0	7	2	0	8	6	5	5
4	5	1	5	4	7	2	6	3	8	3	3	9	5	1	1	3	2	6	3	3
9	7	1	9	0	9	6	6	9	5	6	5	8	8	9	5	6	6	5	6	0
4	8	9	3	2	0	1	1	8	8	1	8	2	4	6	5	7	8	7	8	0
1	0	6	3	9	2	5	4	7	4	5	9	6	5	5	0	2	8	4	7	9
6	8	3	8	0	9	7	0	2	6	0	9	3	9	1	6	0	0	9	0	4
6	4	0	4	8	8	0	0	6	0	2	6	4	2	1	8	2	0	8	1	1
1	8	6	6	6	3	2	9	2	6	1	4	6	8	0	4	2	6	5	3	2
7	1	5	4	2	8	8	2	2	3	4	8	4	5	1	4	3	0	9	1	3
0	7	6	3	4	0	1	0	5	4	2	3	7	8	9	6	3	8	7	2	0
6	7	7	0	9	7	8	7	3	4	4	3	1	2	1	4	7	4	9	7	9
8	3	3	9	7	5	0	4	9	1	1	3	6	1	5	3	8	6	8	1	2
9	7	8	6	0	6	0	9	5	8	7	1	4	0	1	6	7	4	9	4	8
7	1	9	3	9	7	0	8	6	0	0	9	8	0	9	3	3	6	5	7	2
4	5	9	6	8	7	4	4	4	1	0	1	3	4	4	7	5	7	2	3	0
8	7	1	7	3	5	4	8	5	3	8	3	2	6	0	5	0	7	4	5	3
2	3	7	3	6	8	3	7	8	3	9	9	8	1	8	4	9	3	9	1	4
2	0	4	6	2	9	7	9	0	6	8	1	1	1	0	2	8	1	4	1	7
1	9	4	6	5	1	9	3	7	7	5	5	8	4	3	5	4	6	7	0	8
9	0	4	5	2	2	4	4	0	5	5	9	9	2	6	5	5	3	5	2	2
1	7	3	2	5	6	8	2	0	6	1	3	9	9	8	2	6	2	2	6	6
1	7	4	2	9	5	8	3	4	9	7	9	3	5	9	1	9	1	0	3	7
0	1	3	1	6	9	2	9	6	5	9	3	0	3	3	8	0	8	1	7	9
1	6	5	9	3	1	1	4	7	5	5	1	6	0	8	3	4	6	0	3	9
3	6	7	7	7	5	3	5	0	5	7	2	3	1	4	1	8	2	8	3	3

- Find quasi-optimal local path based on Lagrangian position.

- **Theorem 1:** *The weighted Bresenham algorithm produces a finite path between any two points on a regular lattice. For a unit square lattice, at most* $\lceil 1.82x \rceil$ *steps are needed to connect two points a distance* x *apart.*

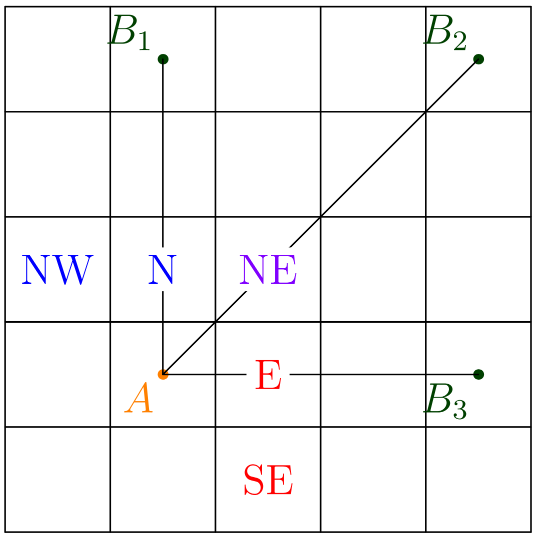


(a)

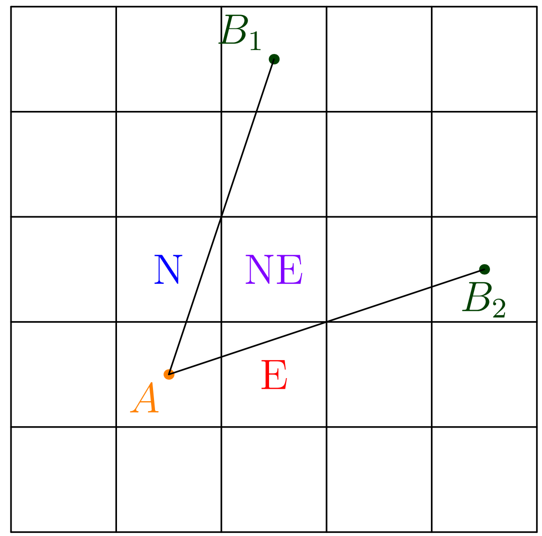


(b)

- **Theorem 1:** *The weighted Bresenham algorithm produces a finite path between any two points on a regular lattice. For a unit square lattice, at most $\lceil 1.82x \rceil$ steps are needed to connect two points a distance x apart.*



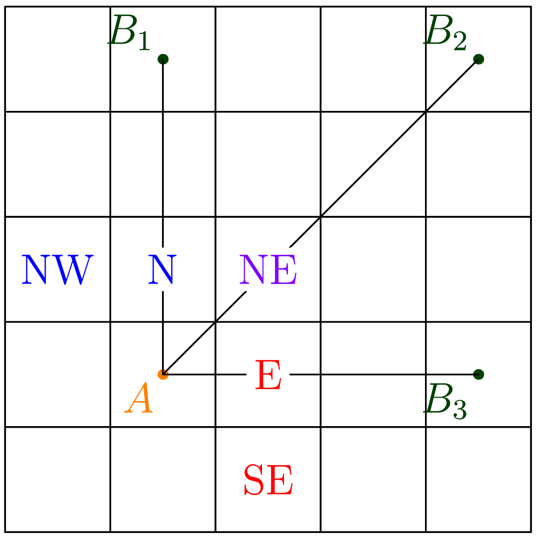
(a)



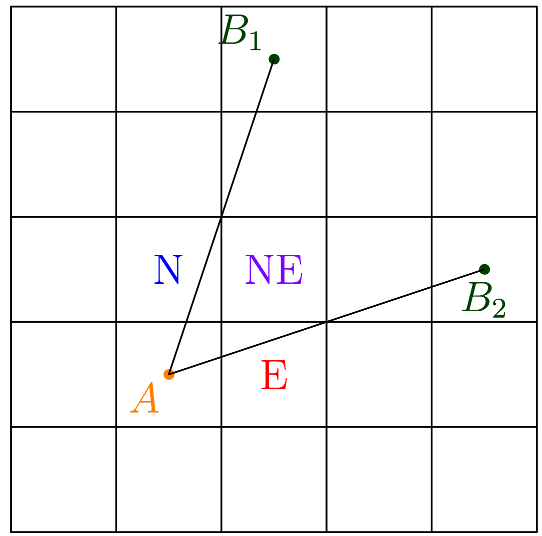
(b)

- Parcel chains: select parcels with minimal weight.

- **Theorem 1:** *The weighted Bresenham algorithm produces a finite path between any two points on a regular lattice. For a unit square lattice, at most* $\lceil 1.82x \rceil$ *steps are needed to connect two points a distance* x *apart.*



(a)



(b)

- Parcel chains: select parcels with minimal weight.
- Multiple holes in same shell: minimize the error.

Approximate Cost/Chain

- Searching for hole:

$$\sum_{k=1}^{\infty} 8k \left(1 - \frac{1}{e}\right)^{4k(k-1)} \approx 8.4.$$

Approximate Cost/Chain

- Searching for hole:

$$\sum_{k=1}^{\infty} 8k \left(1 - \frac{1}{e}\right)^{4k(k-1)} \approx 8.4.$$

- Identifying the path:

$$1.82 \sum_{k=1}^{\infty} k\sqrt{2} \left(1 - \frac{1}{e}\right)^{4k(k-1)} \left(\frac{1}{e}\right) \frac{8k}{1 - \left(1 - \frac{1}{e}\right)^{8k}} \approx 8.6.$$

Approximate Cost/Chain

- Searching for hole:

$$\sum_{k=1}^{\infty} 8k \left(1 - \frac{1}{e}\right)^{4k(k-1)} \approx 8.4.$$

- Identifying the path:

$$1.82 \sum_{k=1}^{\infty} k\sqrt{2} \left(1 - \frac{1}{e}\right)^{4k(k-1)} \left(\frac{1}{e}\right) \frac{8k}{1 - \left(1 - \frac{1}{e}\right)^{8k}} \approx 8.6.$$

- Pushing a chain of parcels:

$$1.82 \sum_{k=1}^{\infty} k\sqrt{2} \left(1 - \frac{1}{e}\right)^{4k(k-1)} \approx 2.7.$$

Diffusion

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{U} = \boldsymbol{D} \nabla^2 \boldsymbol{U}.$$

- Use operator splitting to include diffusion:

$\boldsymbol{U}(t) = \boldsymbol{U}(t_1, t_2)$

$$\frac{\partial \boldsymbol{U}}{\partial t_1} = -\boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{U}, \quad \frac{\partial \boldsymbol{U}}{\partial t_2} = \boldsymbol{D} \nabla^2 \boldsymbol{U}$$

$$\Rightarrow \Delta \boldsymbol{U} = -\boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{U} \Delta t_1 + \boldsymbol{D} \nabla^2 \boldsymbol{U} \Delta t_2.$$

Diffusion

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\boldsymbol{U} = \boldsymbol{D}\nabla^2\boldsymbol{U}.$$

- Use operator splitting to include diffusion:

$\boldsymbol{U}(t) = \boldsymbol{U}(t_1, t_2)$

$$\frac{\partial \boldsymbol{U}}{\partial t_1} = -\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\boldsymbol{U}, \quad \frac{\partial \boldsymbol{U}}{\partial t_2} = \boldsymbol{D}\nabla^2\boldsymbol{U}$$

$$\Rightarrow \Delta\boldsymbol{U} = -\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\boldsymbol{U}\,\Delta t_1 + \boldsymbol{D}\nabla^2\boldsymbol{U}\,\Delta t_2.$$

- Crank–Nicholson scheme solves for diffusive part:

$$\frac{\boldsymbol{U}(t+\tau) - \boldsymbol{U}(t)}{\tau} = \boldsymbol{D}\frac{\nabla^2\boldsymbol{U}(t+\tau) + \nabla^2\boldsymbol{U}(t)}{2}.$$

Diffusion

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\boldsymbol{U} = \boldsymbol{D}\nabla^2\boldsymbol{U}.$$

- Use operator splitting to include diffusion:

$\boldsymbol{U}(t) = \boldsymbol{U}(t_1, t_2)$

$$\frac{\partial \boldsymbol{U}}{\partial t_1} = -\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\boldsymbol{U}, \quad \frac{\partial \boldsymbol{U}}{\partial t_2} = \boldsymbol{D}\nabla^2\boldsymbol{U}$$

$$\Rightarrow \Delta\boldsymbol{U} = -\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\boldsymbol{U}\Delta t_1 + \boldsymbol{D}\nabla^2\boldsymbol{U}\Delta t_2.$$

- Crank–Nicholson scheme solves for diffusive part:

$$\frac{\boldsymbol{U}(t+\tau) - \boldsymbol{U}(t)}{\tau} = \boldsymbol{D}\frac{\nabla^2\boldsymbol{U}(t+\tau) + \nabla^2\boldsymbol{U}(t)}{2}.$$

- In the advection equation $\partial\widetilde{\boldsymbol{U}}/\partial t = -\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\widetilde{\boldsymbol{U}}$:

– Calculate $\widetilde{\boldsymbol{U}}$, interpolate to Eulerian grid.

- Finite difference:

$$\frac{\boldsymbol{U} - \widetilde{\boldsymbol{U}}}{\tau} = \boldsymbol{D}\nabla^2 \left(\frac{\boldsymbol{U} + \widetilde{\boldsymbol{U}}}{2} \right).$$

– Calculate $\widetilde{\boldsymbol{U}}$, interpolate to Eulerian grid.

- Finite difference:

$$\frac{\boldsymbol{U} - \widetilde{\boldsymbol{U}}}{\tau} = \boldsymbol{D}\nabla^2 \left(\frac{\boldsymbol{U} + \widetilde{\boldsymbol{U}}}{2} \right).$$

- Multigrid:

– Let $\boldsymbol{\mathcal{L}} = \mathbf{1} + \dfrac{\tau}{2}\boldsymbol{D}\nabla^2 \quad \Rightarrow \quad \boldsymbol{\mathcal{L}}(-\tau)\boldsymbol{U} = \boldsymbol{\mathcal{L}}(\tau)\widetilde{\boldsymbol{U}}.$

– Calculate $\widetilde{\boldsymbol{U}}$, interpolate to Eulerian grid.

- Finite difference:

$$\frac{\boldsymbol{U}-\widetilde{\boldsymbol{U}}}{\tau}=\boldsymbol{D}\nabla^2\left(\frac{\boldsymbol{U}+\widetilde{\boldsymbol{U}}}{2}\right).$$

- Multigrid:

  – Let $\boldsymbol{\mathcal{L}}=\mathbf{1}+\dfrac{\tau}{2}\boldsymbol{D}\nabla^2 \quad\Rightarrow\quad \boldsymbol{\mathcal{L}}(-\tau)\boldsymbol{U}=\boldsymbol{\mathcal{L}}(\tau)\widetilde{\boldsymbol{U}}$.

- Contribution of diffusion to the Lagrangian solution:

  – Calculate $\boldsymbol{U}-\widetilde{\boldsymbol{U}}$.

  – Project to Lagrangian frame.

  – Add to parcel values.

Self-Advection

- Velocity is now a functional of $\boldsymbol{U}$ determined by 2D vorticity equation:

$$\frac{\partial \omega}{\partial t} + \boldsymbol{v} \boldsymbol{\cdot} \boldsymbol{\nabla} \omega = \boldsymbol{D} \nabla^2 \omega.$$

Self-Advection

- Velocity is now a functional of $\boldsymbol{U}$ determined by 2D vorticity equation:

$$\frac{\partial \omega}{\partial t} + \boldsymbol{v} \boldsymbol{\cdot} \boldsymbol{\nabla} \omega = \boldsymbol{D} \nabla^2 \omega.$$

- Use multigrid solver: compute stream function $\psi = \nabla^{-2} \omega$.

Self-Advection

- Velocity is now a functional of $\boldsymbol{U}$ determined by 2D vorticity equation:

$$\frac{\partial \omega}{\partial t} + \boldsymbol{v} \boldsymbol{\cdot} \boldsymbol{\nabla} \omega = \boldsymbol{D} \nabla^2 \omega.$$

- Use multigrid solver: compute stream function $\psi = \nabla^{-2} \omega$.
- Calculate $\boldsymbol{v} = \hat{\boldsymbol{z}} \times \boldsymbol{\nabla} \psi$ from ψ.

Self-Advection

- Velocity is now a functional of $\boldsymbol{U}$ determined by 2D vorticity equation:

$$\frac{\partial \omega}{\partial t} + \boldsymbol{v} \boldsymbol{\cdot} \boldsymbol{\nabla} \omega = \boldsymbol{D} \nabla^2 \omega.$$

- Use multigrid solver: compute stream function $\psi = \nabla^{-2} \omega$.
- Calculate $\boldsymbol{v} = \hat{\boldsymbol{z}} \times \boldsymbol{\nabla} \psi$ from ψ.
- Problem: calculating $\boldsymbol{v}$ from rearranged $\omega \Rightarrow$ pushing errors accumulate:
  - Propagation of error *via* advection term $\boldsymbol{v} \boldsymbol{\cdot} \boldsymbol{\nabla} \omega$.
  - Introduces large gradients in ω and $C \Rightarrow$ excessive diffusion.

Self-Advection

- Velocity is now a functional of $\boldsymbol{U}$ determined by 2D vorticity equation:

$$\frac{\partial \omega}{\partial t} + \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\omega = \boldsymbol{D}\nabla^2\omega.$$

- Use multigrid solver: compute stream function $\psi = \nabla^{-2}\omega$.

- Calculate $\boldsymbol{v} = \hat{\boldsymbol{z}}\times\boldsymbol{\nabla}\psi$ from ψ.

- Problem: calculating $\boldsymbol{v}$ from rearranged $\omega \Rightarrow$ pushing errors accumulate:

  – Propagation of error *via* advection term $\boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla}\omega$.

  – Introduces large gradients in ω and $C \Rightarrow$ excessive diffusion.

- Solution: use interpolated rather than rearranged values: $\boldsymbol{v}_I\boldsymbol{\cdot}\boldsymbol{\nabla}\omega$, $\nu\nabla^2\omega_I$, and $D\nabla^2 C_I$.

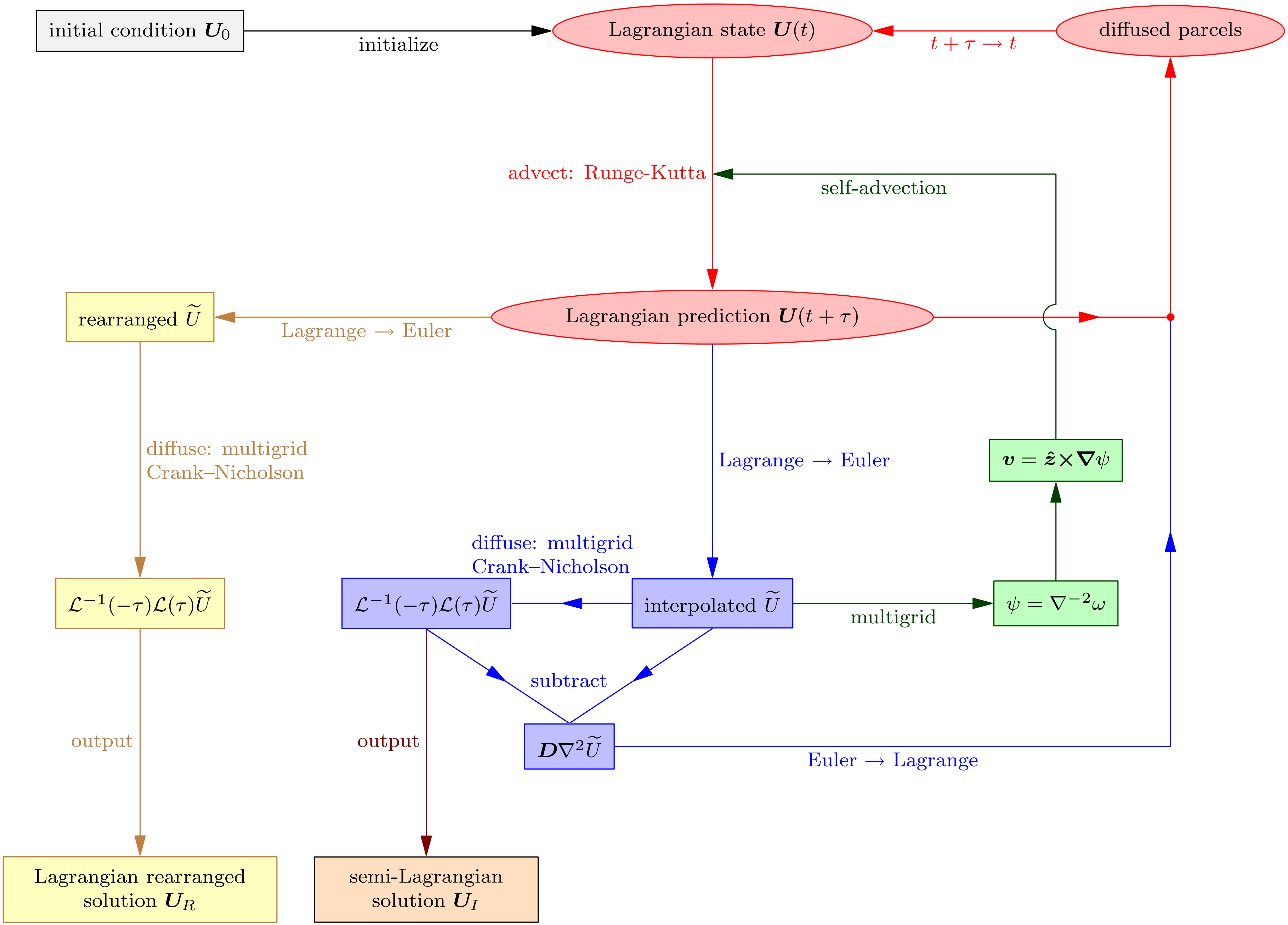
Self-Advection

- Velocity is now a functional of $\boldsymbol{U}$ determined by 2D vorticity equation:

$$\frac{\partial \omega}{\partial t} + \boldsymbol{v} \boldsymbol{\cdot} \boldsymbol{\nabla} \omega = \boldsymbol{D} \nabla^2 \omega.$$

- Use multigrid solver: compute stream function $\psi = \nabla^{-2} \omega$.
- Calculate $\boldsymbol{v} = \hat{\boldsymbol{z}} \times \boldsymbol{\nabla} \psi$ from ψ.
- Problem: calculating $\boldsymbol{v}$ from rearranged $\omega \Rightarrow$ pushing errors accumulate:
  - Propagation of error *via* advection term $\boldsymbol{v} \boldsymbol{\cdot} \boldsymbol{\nabla} \omega$.
  - Introduces large gradients in ω and $C \Rightarrow$ excessive diffusion.
- Solution: use interpolated rather than rearranged values: $\boldsymbol{v}_I \boldsymbol{\cdot} \boldsymbol{\nabla} \omega$, $\nu \nabla^2 \omega_I$, and $D \nabla^2 C_I$.
- This interpolation does not destroy the conservation of Casimirs: velocity need not be a rearrangement.

Summary


initial condition $\boldsymbol{U}_0$
initialize
Lagrangian state $\boldsymbol{U}(t)$
$t+\tau \to t$
diffused parcels
advect: Runge-Kutta
self-advection
rearranged $\widetilde{U}$
Lagrange $\to$ Euler
Lagrangian prediction $\boldsymbol{U}(t+\tau)$
diffuse: multigrid
Crank–Nicholson
Lagrange $\to$ Euler
$\boldsymbol{v} = \hat{\boldsymbol{z}} \times \boldsymbol{\nabla}\psi$
diffuse: multigrid
Crank–Nicholson
$\mathcal{L}^{-1}(-\tau)\mathcal{L}(\tau)\widetilde{U}$
$\mathcal{L}^{-1}(-\tau)\mathcal{L}(\tau)\widetilde{U}$
interpolated $\widetilde{U}$
multigrid
$\psi = \nabla^{-2}\omega$
subtract
output
output
$\boldsymbol{D}\nabla^2\widetilde{U}$
Euler $\to$ Lagrange
Lagrangian rearranged solution $\boldsymbol{U}_R$
semi-Lagrangian solution $\boldsymbol{U}_I$

Simulations: 2 Test Cases

- Semi-Lagrangian solution vs. Lagrangian rearrangement:

Simulations: 2 Test Cases

- Semi-Lagrangian solution vs. Lagrangian rearrangement:
- Grid scale $h = 1.95 \times 10^{-3}$, time step $\tau = 1.95 \times 10^{-2}$.

Simulations: 2 Test Cases

- Semi-Lagrangian solution vs. Lagrangian rearrangement:
- Grid scale $h = 1.95 \times 10^{-3}$, time step $\tau = 1.95 \times 10^{-2}$.
- Initial condition:

$$v_x = \sin(2\pi x)\cos(2\pi y), \quad v_y = -\cos(2\pi x)\sin(2\pi y).$$

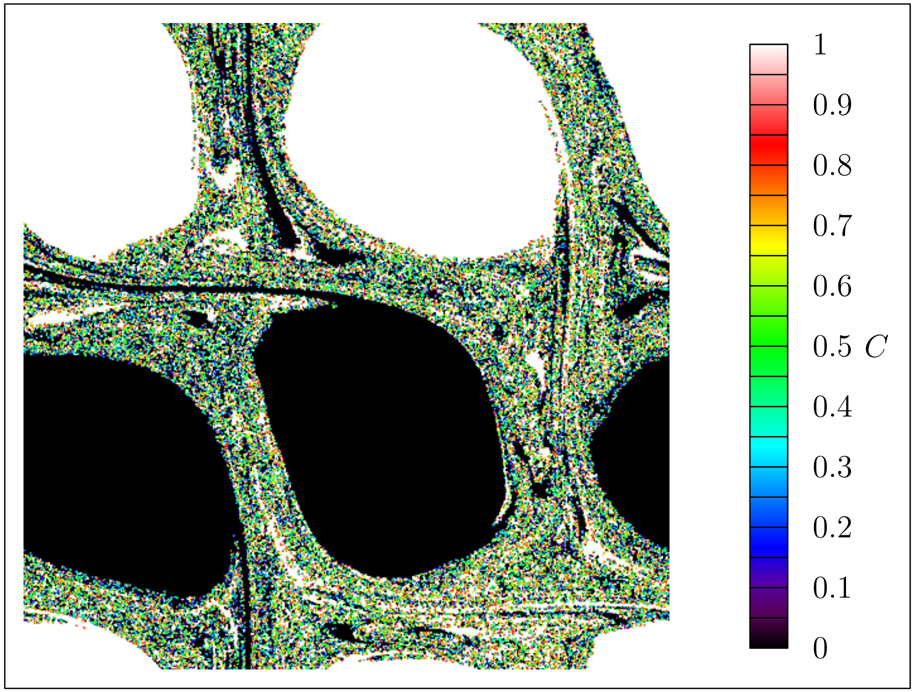
  – Self-advection with no diffusion:

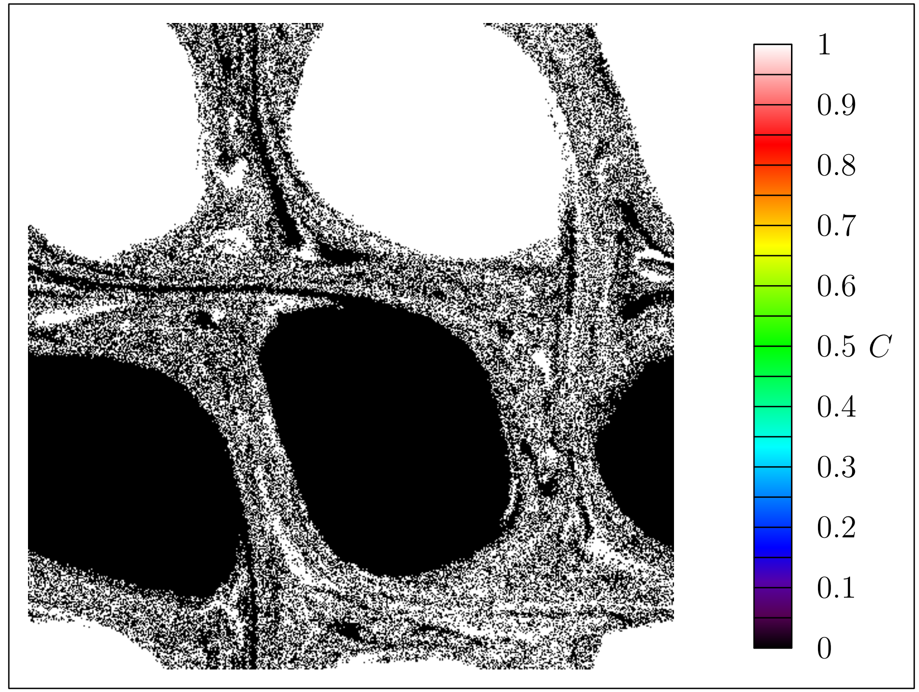
    0 (black) and 1 (white) initial condition for C.

  – Self-advection with diffusion:

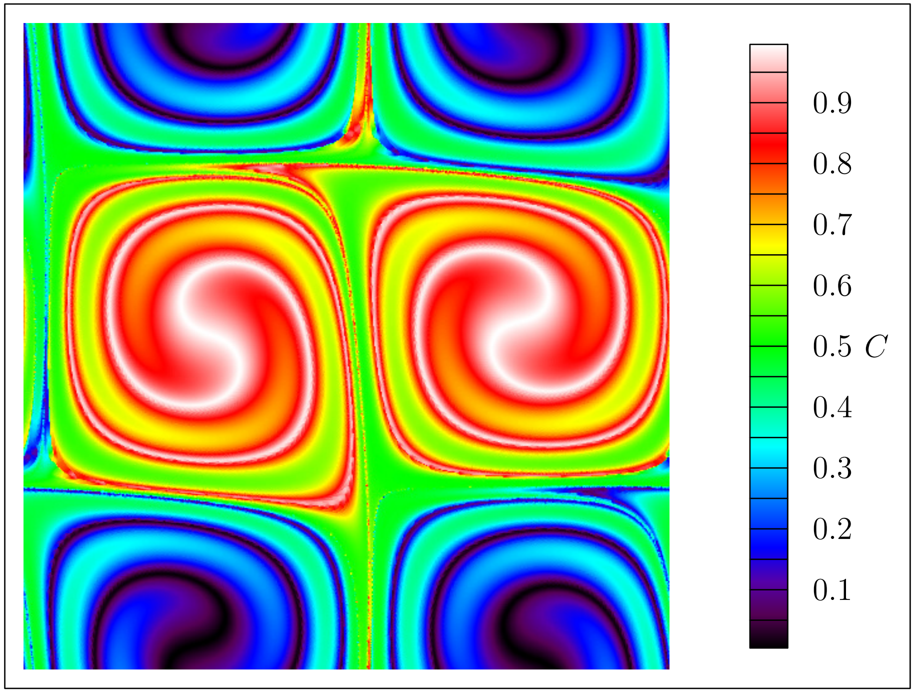
    $D = \nu = 2 \times 10^{-6}$.

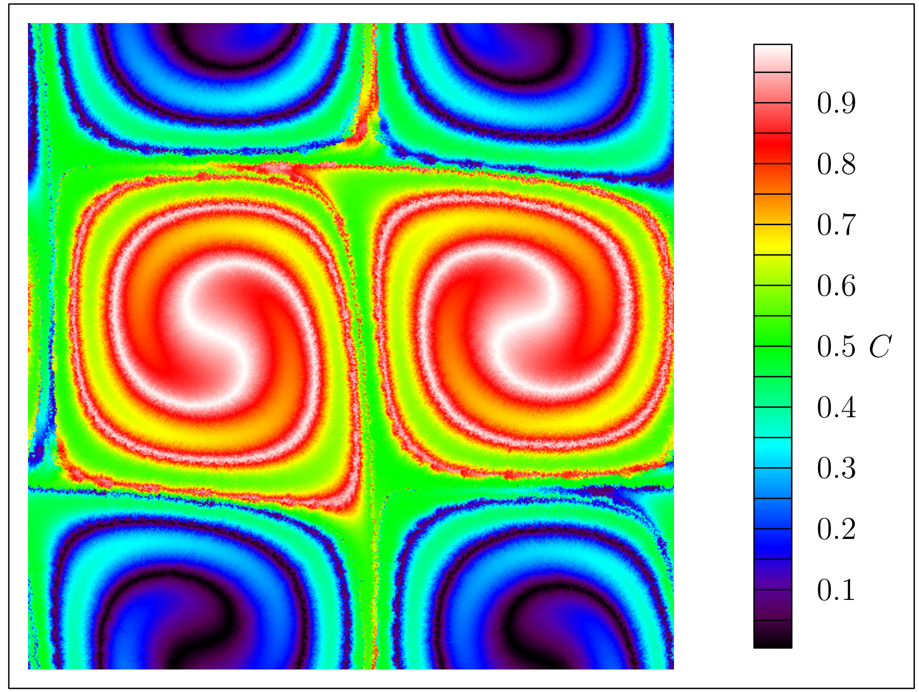
Semi-Lagrangian vs. Lagrangian Rearrangement After 750 Time Steps ($\boldsymbol{D} = \mathbf{0}$)



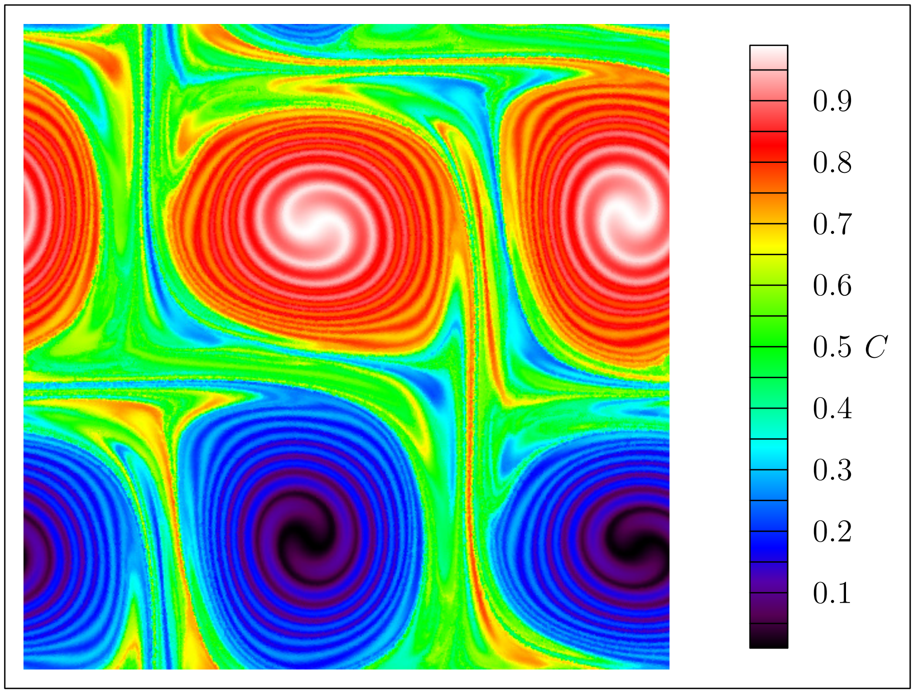


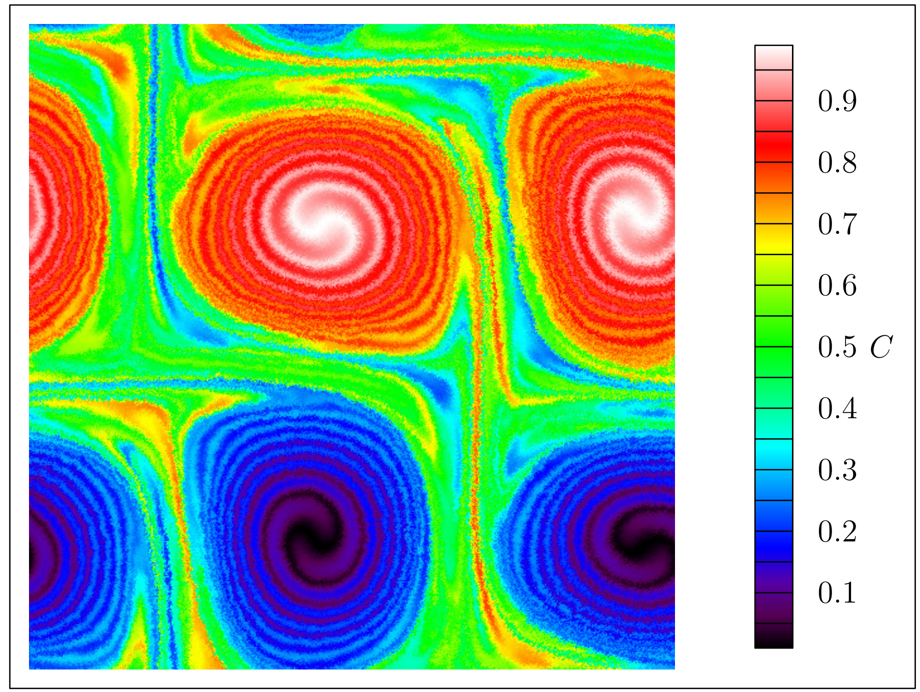
Semi-Lagrangian vs. Lagrangian Rearrangement After 100 Time Steps ($D = \nu = 2 \times 10^{-6}$).



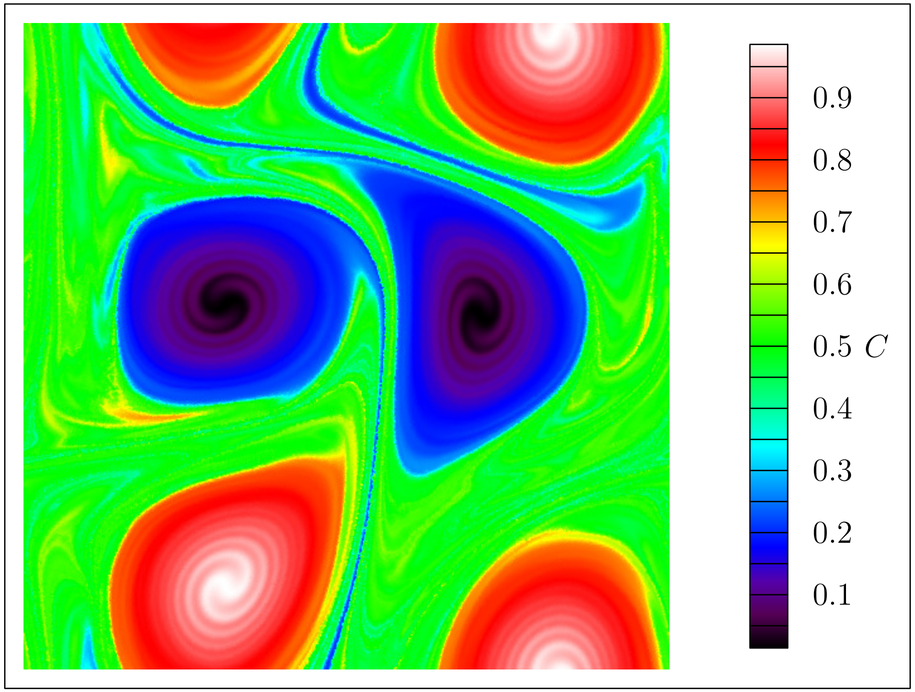


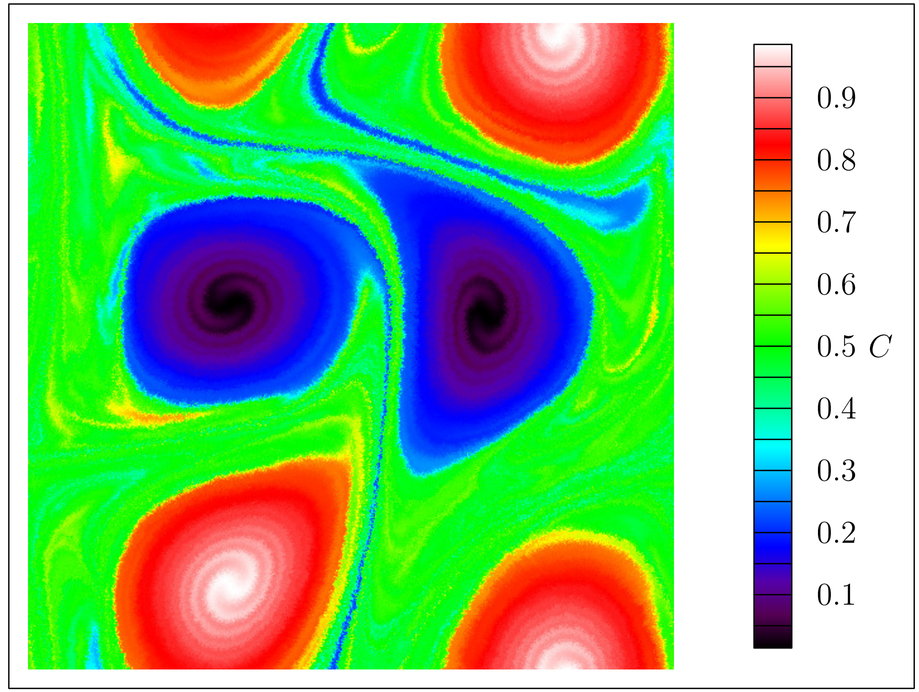
Semi-Lagrangian vs. Lagrangian Rearrangement After 500 Time Steps ($D = \nu = 2 \times 10^{-6}$).





Semi-Lagrangian vs. Lagrangian Rearrangement After 1000 Time Steps ($D = \nu = 2 \times 10^{-6}$).





Energy Decay Rate

$$\frac{\partial C}{\partial t} + \boldsymbol{v}\boldsymbol{\cdot}\boldsymbol{\nabla} C = D\nabla^2 C.$$

- Evolution of concentration energy:

$$\frac{1}{2}\frac{\partial}{\partial t}\int C^2\, d\boldsymbol{x} = -D\int |\boldsymbol{\nabla} C|^2\, d\boldsymbol{x}.$$

Energy Decay Rate

$$\frac{\partial C}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} C = D \nabla^2 C.$$

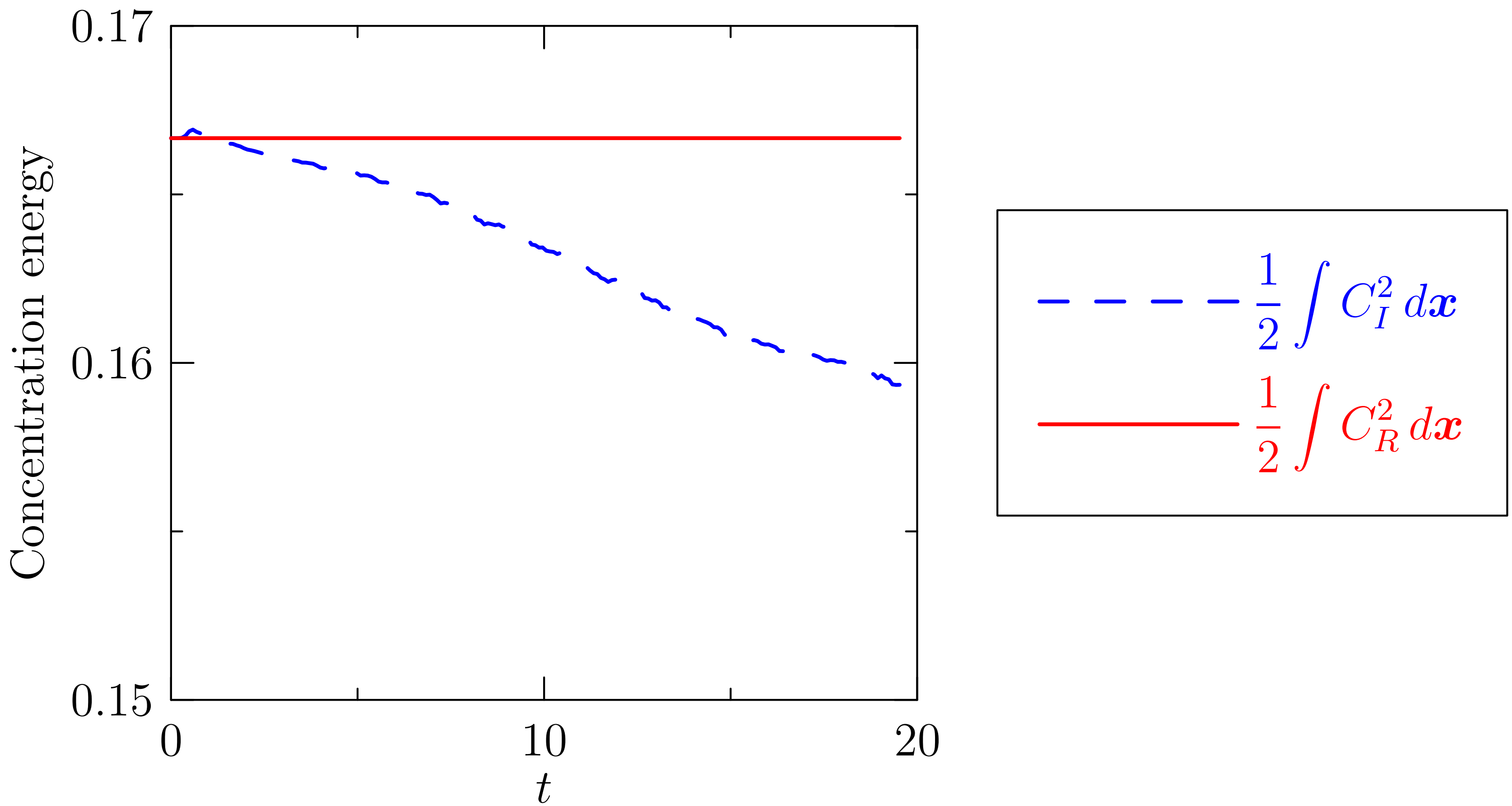
- Evolution of concentration energy:

$$\frac{1}{2}\frac{\partial}{\partial t}\int C^2 \, d\boldsymbol{x} = -D \int |\boldsymbol{\nabla} C|^2 \, d\boldsymbol{x}.$$

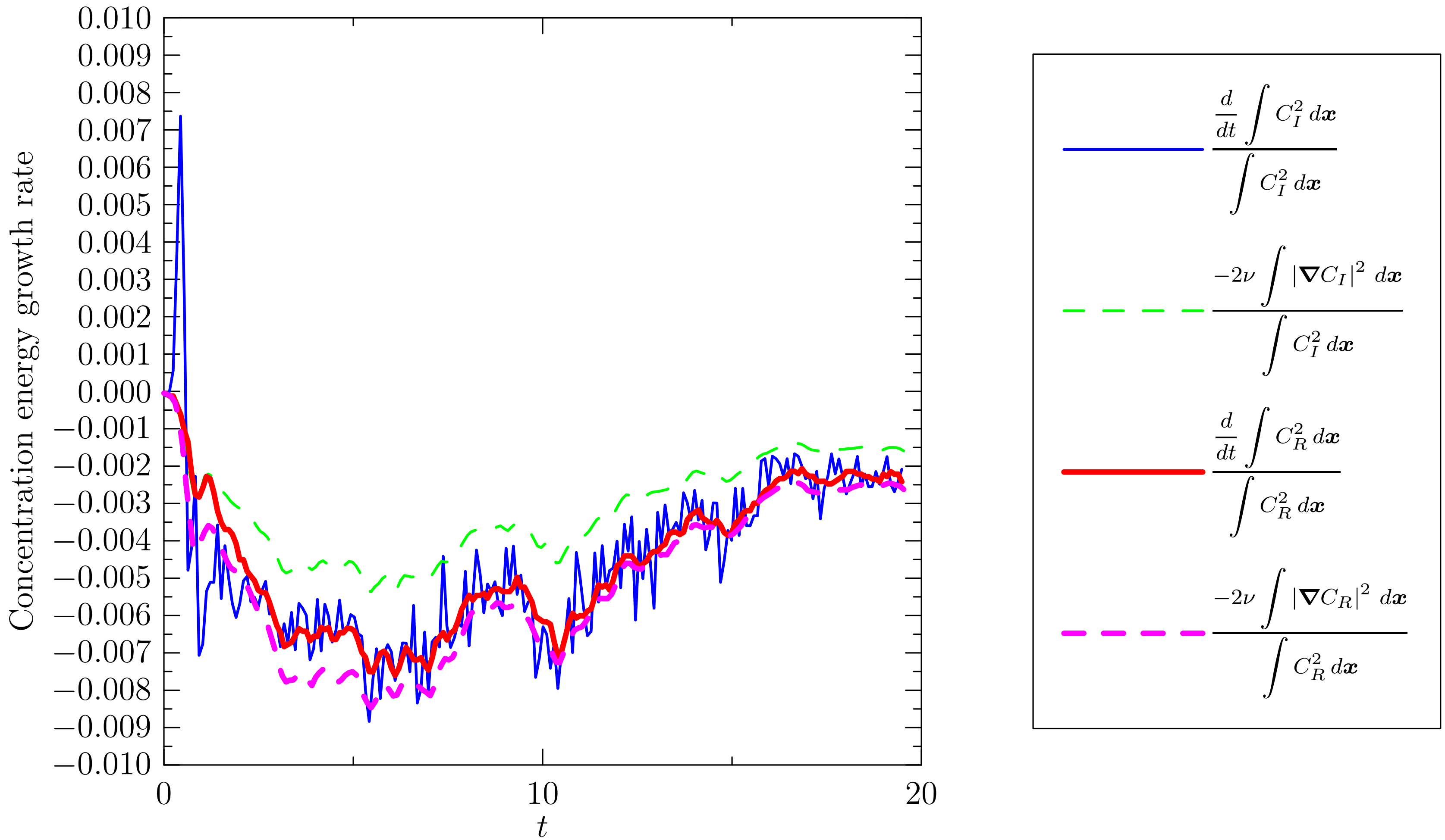
- Compare

$$\frac{\dfrac{\partial}{\partial t}\displaystyle\int C^2 \, d\boldsymbol{x}}{\displaystyle\int C^2 \, d\boldsymbol{x}} \quad \text{and} \quad \frac{-2D \displaystyle\int |\boldsymbol{\nabla} C|^2 \, d\boldsymbol{x}}{\displaystyle\int C^2 \, d\boldsymbol{x}}.$$

Energy Evolution ($\nu = D = 0$)



Energy Decay Rate ($D = \nu = 2 \times 10^{-6}$).



Conclusions

- New numerical method Lagrangian rearrangement respects Casimir invariants.

Conclusions

- New numerical method Lagrangian rearrangement respects Casimir invariants.
- Based on a weighted Bresenham Lagrangian-to-Eulerian projection algorithm.

Conclusions

- New numerical method Lagrangian rearrangement respects Casimir invariants.
- Based on a weighted Bresenham Lagrangian-to-Eulerian projection algorithm.
- Fully Lagrangian:
  - Projected solution is used only for viewing;
  - Error does not propagate to future time steps.

Conclusions

- New numerical method Lagrangian rearrangement respects Casimir invariants.
- Based on a weighted Bresenham Lagrangian-to-Eulerian projection algorithm.
- Fully Lagrangian:
  - Projected solution is used only for viewing;
  - Error does not propagate to future time steps.
- Can combine with:
  - Diffusion

    ($\Rightarrow$ more consistent energy behaviour than interpolation);
  - Self-advected flow.

Conclusions

- New numerical method Lagrangian rearrangement respects Casimir invariants.
- Based on a weighted Bresenham Lagrangian-to-Eulerian projection algorithm.
- Fully Lagrangian:
  - Projected solution is used only for viewing;
  - Error does not propagate to future time steps.
- Can combine with:
  - Diffusion

    ($\Rightarrow$ more consistent energy behaviour than interpolation);
  - Self-advected flow.
- Complexity $\mathcal{O}(n)$.

Asymptote: 2D & 3D Vector Graphics Language

Andy Hammerlindl, John C. Bowman, Tom Prince

`http://asymptote.sf.net`

(freely available under the GNU public license)

Asymptote Lifts TeX to 3D

$$\int_{-\infty}^{+\infty} e^{-\alpha x^2}\, dx = \sqrt{\frac{\pi}{\alpha}}$$

Acknowledgements: Orest Shardt (U. Alberta)

3D Graphs