Monotone interpolation in Semi-Lagrangian (SL) Advection Scheme.

*MAGIC 2009, Hornsjo, Norway.*

Bawfeh Kingsley Kometa, Math. Dept. NTNU, Trondheim Norway
PART 1

*The Remapped particle-mesh method (RPM) SL Advection Scheme*

Method is applied to the density (continuity) equation

\[(\text{Density-eq}) \quad \rho_t + \nabla \cdot (\rho \mathbf{u}) = 0.\]
Introduction

- Method is applied to the density (continuity) equation

\[ \rho_t + \nabla \cdot (\rho \mathbf{u}) = 0. \]  

(Density-eq)

Method is applied to the density (continuity) equation

\[
\rho_t + \nabla \cdot (\rho u) = 0. \tag{Density-eq}
\]


Method has been tested on:
Introduction

- Method is applied to the density (continuity) equation

\[(\text{Density-eq}) \quad \rho_t + \nabla \cdot (\rho \mathbf{u}) = 0.\]


- Method has been tested on:
  - sine wave advection (1D)
Method is applied to the density (continuity) equation

\[(\text{Density-eq}) \quad \rho_t + \nabla \cdot (\rho \mathbf{u}) = 0.\]


Method has been tested on:
- sine wave advection (1D)
- slotted-cylinder problem (solid-body rotation, 2D)
Method is applied to the density (continuity) equation

\[(\text{Density-eq}) \quad \rho_t + \nabla \cdot (\rho \mathbf{u}) = 0.\]


Method has been tested on:
- sine wave advection (1D)
- slotted-cylinder problem (solid-body rotation, 2D)
- idealized cyclogenesis problem (2D)
Method is applied to the density (continuity) equation

\[
\rho_t + \nabla \cdot (\rho \mathbf{u}) = 0.
\]


Method has been tested on:
- sine wave advection (1D)
- slotted-cylinder problem (solid-body rotation, 2D)
- idealized cyclogenesis problem (2D)
- shallow-water equations in planar geometry (2D)
Method is applied to the density (continuity) equation

\[(\text{Density-eq}) \quad \rho_t + \nabla \cdot (\rho \mathbf{u}) = 0.\]


Method has been tested on:

- sine wave advection (1D)
- slotted-cylinder problem (solid-body rotation, 2D)
- idealized cyclogenesis problem (2D)
- shallow-water equations in planar geometry (2D)
- idealized vortex problem (3D)
Given an initial condition $\rho(x, 0) := \rho^0(x)$, (Density-eq) can be expressed in the Lagrangian formulation as

\[
\begin{aligned}
\frac{DX}{Dt} &= \mathbf{u}, \\
\rho(x, t) &= \int \rho^0(x_0) \delta(x - X(x_0, t)) dA(x_0).
\end{aligned}
\]
Discretization

- Given an initial condition $\rho(x, 0) := \rho^0(x)$, (Density-eq) can be expressed in the Lagrangian formulation as

\[
\begin{align*}
\frac{DX}{Dt} &= u, \\
\rho(x, t) &= \int \rho^0(x_0) \delta(x - X(x_0, t)) dA(x_0).
\end{align*}
\]

- The spatial discretization of (Lagr-Form) involves the use of two meshes (grids):
Discretization

- Given an initial condition $\rho(x, 0) := \rho^0(x)$, (Density-eq) can be expressed in the Lagrangian formulation as

$$\begin{cases}
\frac{DX}{Dt} = u, \\
\rho(x, t) = \int \rho^0(x_0) \delta(x - X(x_0, t)) dA(x_0).
\end{cases}$$

- The spatial discretization of (Lagr-Form) involves the use of two meshes (grids):
  
  (a) a fixed Eulerian grid, $x = (x_k, y_l), \ k, l = 0, \ldots, M,$
Discretization

- Given an initial condition $\rho(x, 0) := \rho^0(x)$, (Density-eq) can be expressed in the Lagrangian formulation as

\[
\begin{cases}
\frac{DX}{Dt} = u, \\
\rho(x, t) = \int \rho^0(x_0) \delta(x - X(x_0, t)) dA(x_0).
\end{cases}
\]

- The spatial discretization of (Lagr-Form) involves the use of two meshes (grids):
  - (a) a fixed Eulerian grid, $x = (x_k, y_l)$, $k, l = 0, \ldots, M$,
  - (b) a Lagrangian grid (particle-mesh), $X_\beta(t) = (X_\beta(t), Y_\beta(t))$, $\beta = 0, \ldots, N$ which evolves with time according to the relation

\[
\frac{d}{dt} X_\beta = u_\beta
\]
The particle-mesh is initially taken as a uniform mesh (i.e., $\mathbf{X}_\beta(0) := \mathbf{x}_\beta$)
Discretization

The particle-mesh is initially taken as a uniform mesh (i.e., $X_\beta(0) := x_\beta$).

So the semi-discrete densities are given by

\[ (\text{semi-discrete}) \quad \rho_{k,l}(t) := \sum_\beta m^0_\beta \psi_{k,l}(X_\beta(t)), \quad m^0_\beta := \rho^0(x^0_\beta) A_\beta, \]
The particle-mesh is initially taken as a uniform mesh (i.e., \( \mathbf{X}_\beta(0) := x_\beta \))
So the semi-discrete densities are given by

\[
\rho_{k,l}(t) := \sum_{\beta} m_\beta^0 \psi_{k,l}(\mathbf{X}_\beta(t)), \quad m_\beta^0 := \rho^0(\mathbf{x}_\beta^0) A_\beta,
\]

where \( \psi_{k,l}(\mathbf{x}) \geq 0 \) are basis functions satisfying \( \int \psi_{k,l}(\mathbf{x})dA(\mathbf{x}) = 1 \), and \( A_\beta \) denotes the area of a particle-grid cell.
Discretization

The particle-mesh is initially taken as a uniform mesh (i.e., $X_\beta(0) := x_\beta$). So the semi-discrete densities are given by

$$(\text{semi-discrete}) \quad \rho_{k,l}(t) := \sum_\beta m_\beta^0 \psi_{k,l}(X_\beta(t)), \quad m_\beta^0 := \rho^0(x_\beta^0) A_\beta,$$

where $\psi_{k,l}(x) \geq 0$ are basis functions satisfying $\int \psi_{k,l}(x) dA(x) = 1$, and $A_\beta$ denotes the area of a particle-grid cell. The basis functions are chosen so that they satisfy the partition of unity (PoU) property.
Discretization

The particle-mesh is initially taken as a uniform mesh (i.e., $X_{\beta}(0) := x_{\beta}$).

So the semi-discrete densities are given by

\[
\rho_{k,l}(t) := \sum_{\beta} m_{\beta}^{0} \psi_{k,l}(X_{\beta}(t)), \quad m_{\beta}^{0} := \rho^{0}(x_{\beta}^{0}) A_{\beta},
\]

where $\psi_{k,l}(x) \geq 0$ are basis functions satisfying $\int \psi_{k,l}(x) dA(x) = 1$, and $A_{\beta}$ denotes the area of a particle-grid cell. The basis functions are chosen so that they satisfy the partition of unity (PoU) property

\[
\sum_{k,l} \psi_{k,l}(x) A_{k,l} = 1, \quad A_{k,l} := \Delta x \Delta y.
\]
Discretization

The particle-mesh is initially taken as a uniform mesh (i.e., \( \mathbf{X}_\beta(0) := \mathbf{x}_\beta \))

So the semi-discrete densities are given by

\[
\rho_{k,l}(t) := \sum_\beta m^0_\beta \psi_{k,l}(\mathbf{X}_\beta(t)), \quad m^0_\beta := \rho^0(\mathbf{x}^0_\beta) A_\beta,
\]

where \( \psi_{k,l}(\mathbf{x}) \geq 0 \) are basis functions satisfying \( \int \psi_{k,l}(\mathbf{x}) dA(\mathbf{x}) = 1 \), and \( A_\beta \) denotes the area of a particle-grid cell. The basis functions are chosen so that they satisfy the partition of unity (PoU) property

\[
\sum_{k,l} \psi_{k,l}(\mathbf{x}) A_{k,l} = 1, \quad A_{k,l} := \Delta x \Delta y.
\]

This ensures the conservation of mass.
Examples of basis functions that satisfy the PoU property are given by

\[ \psi_{k,l}(x) := \frac{1}{\Delta x \Delta y} \varphi \left( \frac{x - x_k}{\Delta x} \right) \varphi \left( \frac{y - y_k}{\Delta y} \right), \]
Examples of basis functions that satisfy the PoU property are given by

$$\psi_{k,l}(x) := \frac{1}{\Delta x \Delta y} \varphi \left( \frac{x - x_k}{\Delta x} \right) \varphi \left( \frac{y - y_k}{\Delta y} \right),$$

where $\varphi$ the linear spline or cubic spline functions given respectively by
Examples of basis functions that satisfy the PoU property are given by

$$\psi_{k,l}(x) := \frac{1}{\Delta x \Delta y} \varphi \left( \frac{x - x_k}{\Delta x} \right) \varphi \left( \frac{y - y_k}{\Delta y} \right),$$

where $\varphi$ the linear spline or cubic spline functions given respectively by

\begin{align*}
\text{(Linear-spl) } \psi_{ls} &:= \begin{cases} 
1 - |r|, & |r| \leq 1, \\
0, & |r| > 1,
\end{cases} \\
\text{(Cubic-spl) } \psi_{cs} &:= \begin{cases} 
\frac{2}{3} - |r|^{2} + \frac{1}{2}|r|^{3}, & |r| \leq 1, \\
\frac{1}{6}(2 - |r|)^{3}, & 1 < |r| \leq 2, \\
0, & |r| > 2.
\end{cases}
\end{align*}
RPM Method

(Adapted from the HPM of Frank et al.2002)

The Algorithm is in the following steps [0 – 3]
RPM Method

(Adapted from the HPM of Frank et al. 2002)

The Algorithm is in the following steps $[0 - 3]$

**Step 0:** Reset the particles $\mathbf{X}_\beta^n$ to an Eulerian grid, i.e.,

$$
\mathbf{X}_\beta^n := \mathbf{x}_\beta^n = \mathbf{x}_{i,j}, \quad \beta = 1 + i + j(M + 1), \quad i, j = 0, \ldots, M
$$
RPM Method

(Adapted from the HPM of Frank et al. 2002)

The Algorithm is in the following steps \([0 - 3]\)

**Step 0:** Reset the particles \(X^\beta_n\) to an Eulerian grid, i.e.,

\[
X^\beta_n := x^\beta_n = x_{i,j}, \quad \beta = 1 + i + j(M + 1), \quad i, j = 0, \ldots, M
\]

**Step 1:** Compute the particles “masses” at time level \(t_n\) from the semi-discrete system (semi-discrete), i.e.,

\[
\rho_{k,l}^n = \sum_{k,l} m_{i,j}^n \psi_{k,l}(x_{i,j}),
\]

using local numbering for the \(\beta := \{i, j\}\) for the particle-mesh points.
RPM Method

(Adapted from the HPM of Frank et al.2002)

The Algorithm is in the following steps [0 – 3]

**Step 0:** Reset the particles $\mathbf{X}_\beta^n$ to an Eulerian grid, i.e.,

$$
\mathbf{X}_\beta^n := \mathbf{x}_\beta^n = x_{i,j}, \quad \beta = 1 + i + j(M + 1), \quad i, j = 0, \ldots, M
$$

**Step 1:** Compute the particles “masses” at time level $t_n$ from the semi-discrete system (semi-discrete), i.e.,

$$
\rho_{k,l}^n = \sum_{k,l} m_{i,j}^n \psi_{k,l}(x_{i,j}),
$$

using local numbering for the $\beta := \{i, j\}$ for the particle-mesh points.

**Step 2:** Step forward the particles by solving (Xtic-eq) forward in time for the arrival points (via, say, some Runge-Kutta method). E.g.

$$
\mathbf{X}_{i,j}^{n+1} = \mathbf{X}_{i,j}^n + \Delta t u_{i,j}^{n+1/2}, \quad u_{i,j}^{n+1/2} := u(\mathbf{X}_{i,j}^n, t^{n+1/2})
$$
RPM Method

**Step 3:** Update the densities via (semi-discrete), i.e.,

\[ \rho_{k,l}^{n+1} = \sum_{i,j} m_{i,j}^{n} \psi_{k,l}(X_{i,j}^{n+1}), \]
Some numerical results

We consider the continuity equation

\[ \rho_t + (\rho u)_x = 0, \]

(Example) with periodic boundary conditions on \([0, 1]\) and initial condition

\[ \rho_0(x) = \sin(2\pi x) \]

where the velocity field is \(u(x, t) = U = 1\).
Some numerical results

Figure: Linear convection of the sine wave function $\rho_0(x) = \sin(2\pi x)$ with a uniform advection speed $U = 1$. Domain = $[0, 1)$, periodic BCs; $\Delta x = 1/32, \Delta t = 0.12\Delta x/U$ using 20 time steps. (a) Using linear spline interpolation $\text{Error}(l_2) = 0.0399$; (b) Using cubic spline interpolation. $\text{Error}(l_2) = 0.0597$. 

Monotone interpolation in Semi-Lagrangian (SL) Advection Scheme. – p.10/29
Some numerical results

Figure: Linear convection of the Gaussian function $\rho_0(x) = \exp[-(x - x_0)^2/(2\lambda^2)]$, $x_0 = 0$, $\lambda = 1/32$ with a uniform advection speed $U = 1$. Domain = $[-1, 1)$, periodic BCs; $\Delta x = 1/32$, $\Delta t = 0.12\Delta x/U$ using 20 time steps. (a) Using linear spline interpolation $\text{Error}(l_2) = 0.0993$; (b) Using cubic spline interpolation. $\text{Error}(l_2) = 0.0187$. 

(Linear splines) 

(Exact) 

(num) 

(Linear convection of the Gaussian function) 

(density at $T = 0.075$) 

(Linear splines) 

(Exact) 

(num) 

(Linear convection of the Gaussian function) 

(density at $T = 0.075$) 

(Cubic splines) 

(Exact) 

(num) 

(Linear convection of the Gaussian function) 

(density at $T = 0.075$)
PART 2

A class of monotone interpolation schemes

P. K. Smolarkiewicz & G. A. Grell (1992)
Introduction

- An interpolation technique based on
Introduction

- An interpolation technique based on
  - the “advection-interpolation equivalence” idea of Smolarkiewicz and Grell.
An interpolation technique based on
- the “advection-interpolation equivalence” idea of Smolarkiewicz and Grell.
- Flux-Corrected Transport (FCT) advection algorithm.
Introduction

- An interpolation technique based on
  - the “advection-interpolation equivalence” idea of Smolarkiewicz and Grell.
  - Flux-Corrected Transport (FCT) advection algorithm.
- Replace interpolation problem by an equivalent advection problem.
An interpolation technique based on

- the “advection-interpolation equivalence” idea of Smolarkiewicz and Grell.
- Flux-Corrected Transport (FCT) advection algorithm.

Replace interpolation problem by an equivalent advection problem.

Solve the advection problem via a FCT scheme.
An interpolation technique based on

- the “advection-interpolation equivalence” idea of Smolarkiewicz and Grell.

- Flux-Corrected Transport (FCT) advection algorithm.

- Replace interpolation problem by an equivalent advection problem.

- Solve the advection problem via a FCT scheme.

- The resulting interpolant is known to be monotone and less dissipative than other monotone or shape-preserving interpolants [Rasch, Williamson, 1990].
Advection-Interpolation equivalence

Given a scalar variable

\[(\text{scalar-var}) \quad \psi : \Omega \rightarrow \mathbb{R},\]

known at grid points \(x_i \in \Omega \subset \mathbb{R},\)
Advection-Interpolation equivalence

Given a scalar variable

\[(\text{scalar-var}) \quad \psi : \Omega \rightarrow \mathbb{R},\]

known at grid points \(x_i \in \Omega \subset \mathbb{R},\)

we want to approximate \(\psi(x_d)\) where \(x_d \in \Omega\) is an arbitrary point.
Given a scalar variable

\[ \psi : \Omega \rightarrow \mathbb{R}, \]

known at grid points \( x_i \in \Omega \subset \mathbb{R} \),
we want to approximate \( \psi(x_d) \) where \( x_d \in \Omega \) is an arbitrary point.
Stoke’s theorem yields

\[ \psi(x_d) - \psi(x_i) = \int_{x_i}^{x_d} \psi'(y) dy. \]
Given a scalar variable

\[
(\text{scalar-var}) \quad \psi : \Omega \to \mathbb{R},
\]

known at grid points \( x_i \in \Omega \subset \mathbb{R} \),

we want to approximate \( \psi(x_d) \) where \( x_d \in \Omega \) is an arbitrary point.

Stoke’s theorem yields

\[
\psi(x_d) - \psi(x_i) = \int_{x_i}^{x_d} \psi'(y) \, dy.
\]

Integrating along the line segment joining \( x_i \) and \( x_d \), we obtain

\[
\phi(x_d, 1) - \phi(x_i, 0) = \int_{0}^{1} \phi_y(z(t), t) \, dy,
\]
Given a scalar variable

\[ \psi : \Omega \rightarrow \mathbb{R}, \]

known at grid points \( x_i \in \Omega \subset \mathbb{R}, \)

we want to approximate \( \psi(x_d) \) where \( x_d \in \Omega \) is an arbitrary point.

Stoke’s theorem yields

\[ \psi(x_d) - \psi(x_i) = \int_{x_i}^{x_d} \psi'(y)dy. \]

Integrating along the line segment joining \( x_i \) and \( x_d \), we obtain

\[ \phi(x_d, 1) - \phi(x_i, 0) = \int_0^1 \phi_y(z(t), t)dy, \]

where \( \phi \) is defined on the line segment,

\[ C_i := \{ y = z(t) := x_i - t(x_i - x_d) \mid 0 \leq t \leq 1 \} \]

by \( \phi(y, t) := \psi(z(t)) \).
Putting $dy = z'(t)dt = -(x_i - x_d)dt$ and writing $U = x_i - x_d$ we have

$$\phi(x_d, 1) - \phi(x_i, 0) = -\int_0^1 (\phi(y, t)U)_y dt.$$
Putting $dy = z'(t)dt = -(x_i - x_d)dt$ and writing $U = x_i - x_d$ we have

$$\phi(x_d, 1) - \phi(x_i, 0) = -\int_0^1 (\phi(y, t)U)_y dt.$$  

This is equivalent to solving the pure advection problem
Putting $dy = z'(t)dt = -(x_i - x_d)dt$ and writing $U = x_i - x_d$ we have

\[
(\text{change-of-var}) \quad \phi(x_d, 1) - \phi(x_i, 0) = - \int_0^1 (\phi(y, t)U)_y dt.
\]

This is equivalent to solving the pure advection problem

\[
(\text{advection-eq}) \quad \begin{cases} 
\phi_t + (\phi U)_y = 0, & 0 < t \leq 1, \\
\phi(x_i, 0) = \psi(x_i), 
\end{cases}
\]
Solving the advection problem

- An upwind scheme applied gives

\[
\phi_i^1 = \phi_i^0 + \alpha (\phi_{i-1}^0 - \phi_i^0),
\]

where \( \alpha = \frac{U \Delta t}{\Delta x} \), is the Courant number, with \( U = x_i - x_d \), defined such that \( x_i \) is the closest grid point to \( x_d \).
An upwind scheme applied gives

$$\phi_i^1 = \phi_i^0 + \alpha (\phi_{i-1}^0 - \phi_i^0),$$

where $\alpha = \frac{U \Delta t}{\Delta x}$, is the Courant number, with $U = x_i - x_d$, defined such that $x_i$ is the closest grid point to $x_d$.

A Lax-Friedrichs scheme gives

$$\phi_i^1 = \frac{1}{2} (\phi_{i+1}^0 + \phi_{i-1}^0) - \frac{\alpha}{2} (\phi_{i+1}^0 - \phi_{i-1}^0).$$
Solving the advection problem

- An upwind scheme applied gives

\[
\phi_i^1 = \phi_i^0 + \alpha (\phi_{i-1}^0 - \phi_i^0),
\]

where \( \alpha = \frac{U \Delta t}{\Delta x} \), is the Courant number, with \( U = x_i - x_d \), defined such that \( x_i \) is the closest grid point to \( x_d \).

- A Lax-Friedrichs scheme gives

\[
\phi_i^1 = \frac{1}{2} (\phi_{i+1}^0 + \phi_{i-1}^0) - \frac{\alpha}{2} (\phi_{i+1}^0 - \phi_{i-1}^0).
\]

Choosing \( \Delta t = 1 \) we get as a “reasonable” approximation \( \psi(x_d) \approx \phi_i^1 \).
Solving the advection problem

Thus $\psi(x_i)$ is convected through a distance $U \Delta t (\Delta t = 1)$ to obtain an approximation for $\psi(x_d)$. The following sketches illustrate this idea more clearly.

**Figure**: Linear convection of the $\psi$ by a constant velocity field $U$. 
Low order advection schemes have the tendency to cause ‘smearing’ (i.e., diffusion) of the wave profile around discontinuities or steep gradients.
FCT algorithm: Motivation

- Low order advection schemes have the tendency to cause ‘smearing’ (i.e., diffusion) of the wave profile around discontinuities or steep gradients.

- High order fluxes ‘build up’ existing shock or contact discontinuities or introduce spurious ‘wiggles’ in the neighbourhood of such discontinuities.
FCT algorithm: Motivation

- Low order advection schemes have the tendency to cause ‘smearing’ (i.e., diffusion) of the wave profile around discontinuities or steep gradients.

- High order fluxes ‘build up’ existing shock or contact discontinuities or introduce spurious ‘wiggles’ in the neighbourhood of such discontinuities.

- The idea behind the FCT algorithm is to reduce the amount of diffusion caused by the low order fluxes and at the same time eliminate the spuriosity of high order fluxes.
Low order advection schemes have the tendency to cause ‘smearing’ (i.e., diffusion) of the wave profile around discontinuities or steep gradients.

High order fluxes ‘build up’ existing shock or contact discontinuities or introduce spurious ‘wiggles’ in the neighbourhood of such discontinuities.

The idea behind the FCT algorithm is to reduce the amount of diffusion caused by the low order fluxes and at the same time eliminate the spuriosity of high order fluxes.

This way the suitable properties of the low order fluxes (such as monotonicity) are combine with the high accuracy of the high order fluxes.
FCT algorithm: Motivation

- Low order advection schemes have the tendency to cause ‘smearing’ (i.e., diffusion) of the wave profile around discontinuities or steep gradients.

- High order fluxes ‘build up’ existing shock or contact discontinuities or introduce spurious ‘wiggles’ in the neighbourhood of such discontinuities.

- The idea behind the FCT algorithm is to reduce the amount of diffusion caused by the low order fluxes and at the same time eliminate the spuriousity of high order fluxes.

- This way the suitable properties of the low order fluxes (such as monotonicity) are combine with the high accuracy of the high order fluxes.

Numerical examples: Interpolation

Given the Gaussian function

\[ G(x; x_0, \lambda) := \exp[-(x - x_0)^2 / (2\lambda^2)], \quad x \in [-1, 1], \]

where \( x_0, \lambda \) are given parameters, we shall interpolate \( G \) at midpoints of the grid, \( x = -1 : \Delta x : 1 \), namely, at \( x_d = x - \frac{\Delta x}{2} \).

![Monotone interpolation in Semi-Lagrangian (SL) Advection Scheme. – p.19/29](image)
Numerical examples: Interpolation

Figure: Interpolation of the Gaussian cone $G(x; 0, 1/32)$, at the points $x_d = x - \Delta x/2$, where $x = -1 : \Delta x : 1, \Delta x = 2/N$ are grid points. Comparing the upwind scheme and its FCT version, one different grids ($\Delta x = 2/N, 2^2 \leq N \leq 2^8$).
Numerical examples: Interpolation

Figure: Interpolation of the Gaussian cone $G(x; 0, 1/32)$, at the points $x_d = x - \Delta x/2$, where $x = -1 : \Delta x : 1, \Delta x = 2/N$ are grid points. Comparing Lax-Friedriechs schemes with its FCT version, using fluxes of different higher order advection schemes ($\Delta x = 2/N, 2^2 \leq N \leq 2^8$).
We now test the interpolation results in a semi-Lagrangian convection scheme. The test problem is the linear convection of a Gaussian cone.

\[
\begin{aligned}
\rho_t + (\rho u)_x &= 0, \\
\rho(x, 0) &= G(x; x_0, \lambda),
\end{aligned}
\]

where \( u = 2 \), is the convection velocity.
Numerical examples: SL advection

Figure: Semi-Lagrangian linear convection of the Gaussian cone $G(x; 0, 1/16)$. Grid points: $x \in [-1, 1]$, $\Delta x = 2/N$, $N = 2^8$, with periodic BCs. $[u = 2, T = 1, \Delta t = T/40, \text{Courant} = 6.4]$. Method for interpolation: Lax-Friedriechs. Error($L_2$) = 0.0465. Using MATLAB’s expm function (together with centered differences for spatial discretization) introduces some phase error. Here error($L_2$) = 0.0620.
Numerical examples: SL advection

Figure: Semi-Lagrangian linear convection of the Gaussian cone $G(x; 0, 1/16)$. Grid points: $x \in [-1, 1]$, $\Delta x = 2/N$, $N = 2^8$, with periodic BCs. $[u = 2, T = 1, \Delta t = T/40$, Courant $= 6.4]$. Method for interpolation: Lax-Friedriechs (FCT with Lax-Wendroff). Error($L_2$) = 0.0023.
Numerical examples: SL advection

Figure: Semi-Lagrangian linear convection of the Gaussian cone $G(x; 0, 1/16)$. Grid points: $x \in [-1, 1], \Delta x = 2/N, N = 2^8$, with periodic BCs. $[u = 2, T = 1, \Delta t = T/40, \text{Courant} = 6.4]$. Method for interpolation: Lax-Friedriechs (FCT with Crowley4). Error($L_2$) = $4.5 \times 10^{-4}$. 

Monotone interpolation in Semi-Lagrangian (SL) Advection Scheme. – p.25/29
Numerical examples: SL advection

**Figure**: Semi-Lagrangian linear convection of the Gaussian cone \( G(x; 0, 1/16) \). Grid points: \( x \in [-1, 1] \), \( \Delta x = 2/N \), \( N = 2^8 \), with periodic BCs. \([u = 2, T = 0.1, \Delta t = T/40, \text{Courant} = 0.64]\). Method for interpolation: Upwind. Error \((L_2) = 0.0542\).
Numerical examples: SL advection

Figure: Semi-Lagrangian linear convection of the Gaussian cone $G(x; 0, 1/16)$. Grid points: $x \in [-1, 1], \Delta x = 2/N, N = 2^8$, with periodic BCs. $[u = 2, T = 0.1, \Delta t = T/40$, Courant = 0.64]. Method for interpolation: Upwind (FCT with Lax-Wendroff). Error($L_2$) = 0.0024.
Numerical examples: SL advection

Figure:  Semi-Lagrangian linear convection of the Gaussian cone $G(x; 0, 1/16)$. Grid points: $x \in [-1, 1]$, $\Delta x = 2/N$, $N = 2^8$, with periodic BCs. $[u = 2, T = 0.1, \Delta t = T/40$, Courant = 0.64]. Method for interpolation: Upwind (FCT with Crowley4). Error($L_2$) = 0.0023.
Conclusion

- Thanks!
- Questions?