A positivity-preserving flux-corrected transport scheme for solving scalar conservation law problems

Joshua E. Hansel[1] Jean C. Ragusa[1] Jean-Luc Guermond[2]

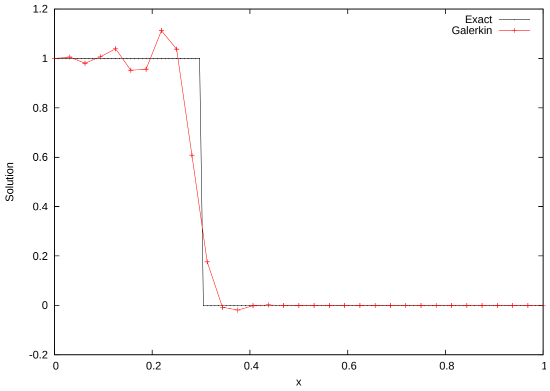
[1]Department of Nuclear Engineering
Texas A&M University

[2]Department of Mathematics
Texas A&M University

`deal.II` Workshop, Summer 2015

Motivation

- Weak solutions to conservation law problems in general are not unique; thus solution via CFEM prone to unphysical oscillations:



Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Objectives

- The objectives of the research are the following:
  - **Accurately solve conservation law problems** using the continuous finite element method (CFEM).
    - Scheme to be presented is 2nd order-accurate in space (for smooth problems).
  - **Prevent spurious oscillations**.
    - Scheme to be presented is not proven to be completely immune to any spurious oscillations but shows good results in practice.
  - **Prevent negativities** for physically non-negative quantities.
    - Scheme to be presented is guaranteed to be positivity-preserving.

Outline

- Presentation of scheme for simple case
    - Problem formulation
    - Monotone low-order scheme
    - High-order entropy viscosity scheme
    - FCT scheme
- Results
- Conclusions

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Conservation Law Models

- Guermond has addressed these objectives for general nonlinear scalar conservation laws using explicit temporal discretizations:

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0$$

- Common examples:

$\mathbf{f}(u) = u\mathbf{v}$	Linear advection equation
$\mathbf{f}(u) = \frac{1}{2}u^2\mathbf{v}$	Burgers equation

- We extend these techniques to include a reaction term and source term and to use implicit and steady-state temporal discretizations:

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) + \sigma u = q$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Problem Formulation

- Scalar linear conservation law model:

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v} u(\mathbf{x},t)) + \sigma(\mathbf{x}) u(\mathbf{x},t) = q(\mathbf{x},t) \qquad (1)$$

$$\sigma(\mathbf{x}) \geq 0, \qquad q(\mathbf{x},t) \geq 0$$

- Define problem by providing initial conditions and some boundary condition, such as Dirichlet:

$$u(\mathbf{x},0) = u^0(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{D} \qquad (2)$$

$$u(\mathbf{x},t) = u^{inc}(\mathbf{x}) \quad \forall \mathbf{x} \in \partial\mathcal{D}^{inc} \qquad (3)$$

- CFEM solution:

$$u_h(\mathbf{x},t) = \sum_{j=1}^{N} U_j(t)\varphi_j(\mathbf{x}), \quad \varphi_j(\mathbf{x}) \in P_h^1 \qquad (4)$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Time Discretization

- Simplest time discretization is forward Euler (FE), which gives the discrete system

$$\mathbf{M}^C\frac{\mathbf{U}^{n+1}-\mathbf{U}^n}{\Delta t}+\mathbf{A}\mathbf{U}^n=\mathbf{b}^n \tag{5}$$

$$M^C_{i,j}\equiv\int\limits_{S_{i,j}}\varphi_i(\mathbf{x})\varphi_j(\mathbf{x})d\mathbf{x} \tag{6}$$

$$A_{i,j}\equiv\int\limits_{S_{i,j}}\left(\mathbf{v}\cdot\nabla\varphi_j(\mathbf{x})+\sigma(\mathbf{x})\varphi_j(\mathbf{x})\right)\varphi_i(\mathbf{x})d\mathbf{x} \tag{7}$$

$$b^n_i\equiv\int\limits_{S_i}q(\mathbf{x},t^n)\varphi_i(\mathbf{x})d\mathbf{x} \tag{8}$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Flux Corrected Transport (FCT) Scheme
Introduction

- Initially developed in 1973 for finite difference discretizations of transport/conservation law problems and recently applied to finite element method.
- Works by adding conservative fluxes to satisfy physical bounds on the solution.
- Employs a high-order scheme and a low-order, monotone scheme.
- Defines a *correction*, or *antidiffusion*, flux, which when added to the low-order scheme, produces the high-order scheme solution.
- Limits this correction flux to enforce the physical bounds imposed.

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Low-Order Scheme
Definition

- To get the low-order scheme, one does the following:
  - Lumps the mass matrix: $\mathbf{M}^C \rightarrow \mathbf{M}^L$.
  - Adds a low-order diffusion operator: $\mathbf{A} \rightarrow \mathbf{A} + \mathbf{D}^L$.
- This gives the following, where $\mathbf{U}^{L,n+1}$ is the low-order solution:

$$\mathbf{M}^L \frac{\mathbf{U}^{L,n+1} - \mathbf{U}^n}{\Delta t} + (\mathbf{A} + \mathbf{D}^L)\mathbf{U}^n = \mathbf{b}^n \qquad (9)$$

- The diffusion matrix $\mathbf{D}^L$ is assembled elementwise, where K denotes an element, using a local bilinear form b_K and a local low-order viscosity ν^L_K:

$$D^L_{i,j} = \sum_{K \subset S_{i,j}} \nu^L_K b_K(\varphi_j, \varphi_i) \qquad (10)$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Low-Order Scheme
Local Bilinear Form

- The local bilinear form is defined as follows, where $|K|$ denotes the volume of element K, $\mathcal{I}(K)$ is the set of indices corresponding to degrees of freedom with nonempty support on K, and n_K is the cardinality of this set.

$$b_K(\varphi_j,\varphi_i) \equiv \begin{cases} -\frac{1}{n_K-1}|K| & i \neq j, \quad i,j \in \mathcal{I}(K) \\ |K| & i = j, \quad i,j \in \mathcal{I}(K) \\ 0 & i \notin \mathcal{I}(K) \,|\, j \notin \mathcal{I}(K) \end{cases} \tag{11}$$

- Some properties that result from this definition:

$$\sum_j b_K(\varphi_j,\varphi_i) = 0 \tag{12}$$

$$b_K(\varphi_i,\varphi_i) > 0 \tag{13}$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Low-Order Scheme
Low-Order Viscosity

- The low-order viscosity is defined as

$$\nu^L_K \equiv \max_{i\neq j\in\mathcal{I}(K)} \frac{\max(0, A_{i,j})}{-\sum\limits_{T\subset S_{i,j}} b_T(\varphi_j, \varphi_i)} \qquad (14)$$

- This definition is designed to be the smallest number such that the following is guaranteed:

$$D^L_{i,j} \leq -A_{i,j}, \quad j \neq i \qquad (15)$$

- This is used to guarantee that the low-order steady-state matrix $\mathbf{A}^L = \mathbf{A} + \mathbf{D}^L$ is an M-matrix, i.e., a *monotone* matrix: $\mathbf{A}^L\mathbf{U} \geq 0 \Rightarrow \mathbf{U} \geq 0$.

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Low-Order Scheme
Discrete Maximum Principle

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

- In addition to guaranteeing monotonicity and positivity, the low-order viscous terms guarantee the following discrete maximum principle (DMP), where $U^n_{\substack{\max\\\min},i} = \substack{\max\\\min}_{j\in\mathcal{I}(S_i)} U^n_j$:

$$W_i^- \leq U_i^{L,n+1} \leq W_i^+ \qquad \forall i \tag{16}$$

$$W_i^\pm \equiv U^n_{\substack{\max\\\min},i}\left(1 - \frac{\Delta t}{M^L_{i,i}}\sum_j A^L_{i,j}\right) + \frac{\Delta t}{M^L_{i,i}} b_i^n \tag{17}$$

- For example, when there is no reaction term or source term, this reduces to the following DMP, which implies the scheme is local extremum diminishing (LED):

$$U^n_{\min,i} \leq U_i^{L,n+1} \leq U^n_{\max,i} \qquad \forall i \tag{18}$$

Low-Order Scheme

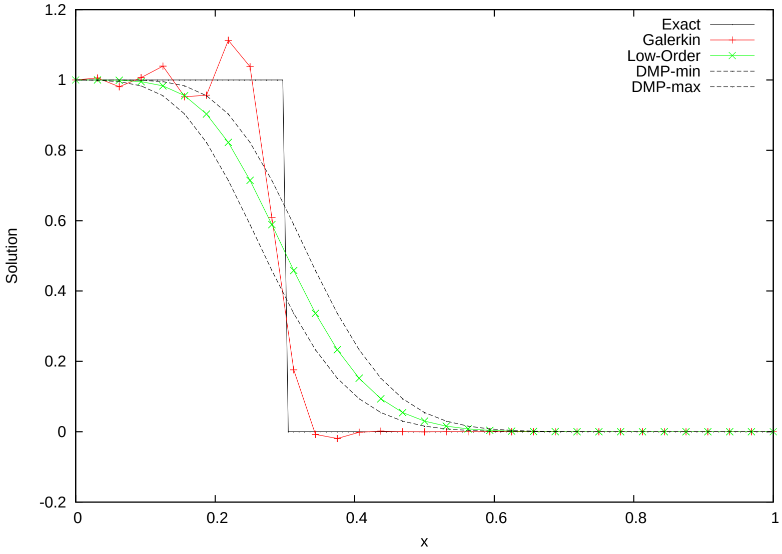
Getting Nonzero Row Entries $\{A_{i,j} : A_{i,j} \neq 0, \quad j = 1 \ldots N\}$

```
void get_matrix_row(
  const SparseMatrix<double> &matrix,
  const unsigned int         &i,
  std::vector<double>        &row_values,
  std::vector<unsigned int>  &row_indices,
  unsigned int               &n_col)
{
  // get first and one-past-end iterator for row
  SparseMatrix<double>::const_iterator it     = matrix.begin(i);
  SparseMatrix<double>::const_iterator it_end = matrix.end(i);

  // determine number of entries in row and resize vectors accordingly
  n_col = it_end - it;
  row_values.resize(n_col);
  row_indices.resize(n_col);

  // loop over columns in row
  for (unsigned int k = 0; it != it_end; ++it, ++k)
  {
    row_values[k]  = it->value();  // get A(i,j)
    row_indices[k] = it->column(); // get j
  }
}
```

Low-Order Scheme
Results Example



Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Entropy Viscosity Scheme
Introduction

- The standard Galerkin CFEM weak solution is not unique. Even with FCT, it would not necessarily converge to the correct, physical weak solution, i.e., the *entropy* solution.
- To converge to the entropy solution, one must ensure that an entropy inequality is satisfied:

$$R(u) \equiv \frac{\partial \eta(u)}{\partial t} + \nabla \cdot \mathbf{f}^{\eta}(u) \leq 0 \qquad (19)$$

  for any convex entropy $\eta(u)$ and corresponding entropy flux $\mathbf{f}^{\eta}(u)$.
- This *entropy residual* $R(u)$ measures entropy production; where it is positive, the inequality is violated, so the residual should be decreased somehow.
- To enforce the inequality, the entropy viscosity method adds viscosity in proportion to local entropy production, thus decreasing local entropy.

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Entropy Viscosity Scheme
User-defined Entropy Function in deal.II

- Below is an example of using ParameterHandler and FunctionParser to let the user choose the entropy function to be $\eta(u) = \frac{1}{2}u^2$:

The input file, "my_input":

```
set Entropy function = 0.5*u*u
```

The code:

```
// create parameter handler and declare entry for entropy function
ParameterHandler parameter_handler;
parameter_handler.declare_entry("Entropy function", "u*u*u/3.0", // default
  Patterns::Anything(), "String for entropy function");

// read the input file and get the entropy function parameter
parameter_handler.read_input("my_input");
std::string entropy_string = parameter_handler.get("Entropy function");

// map of user-defined function parser constants to their values
std::map<std::string, double> constants; // here, this is kept empty

// initialize the function parser
FunctionParser<dim> entropy_function;
entropy_function.initialize("u", entropy_string, constants);
```

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Entropy Viscosity Scheme
Entropy Viscosity Definition

- One chooses a convex entropy function $\eta(u)$ such as $\eta(u) = \frac{1}{2}u^2$ and manipulates the conservation law equation to get an entropy residual:

$$R(u) = \frac{\partial \eta}{\partial t} + \frac{d\eta}{du}\left(\nabla \cdot (\mathbf{v}u) + \sigma u - q\right) \tag{20}$$

- Viscosity is set to be proportional to a linear combination of the local entropy residual $R_K(u) = \|R(u)\|_{L^\infty(K)}$ and entropy jumps $J_F(u)$ across the faces:

$$\nu^\eta_K \propto c_R R_K(u_h) + c_J \max_{F\in\partial K} J_F(u_h) \tag{21}$$

- In practice, the entropy viscosity becomes the following, where the denominator is just a normalization constant:

$$\nu^\eta_K = \frac{c_R R_K(u_h) + c_J \max\limits_{F\in\partial K} J_F(u_h)}{\|\eta(u_h) - \bar{\eta}(u_h)\|_{L^\infty(\mathcal{D})}} \tag{22}$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Entropy Viscosity Scheme
High-Order Scheme

- The high-order viscosity does not need to be any greater than the low-order viscosity:

$$\nu_K^{H,n} = \min(\nu_K^L, \nu_K^{\eta,n}) \tag{23}$$

- For the high-order scheme, the mass matrix is not modified; the only change is the addition of the high-order diffusion operator $\mathbf{D}^{H,n}$: $\mathbf{A} \rightarrow \mathbf{A} + \mathbf{D}^{H,n}$:

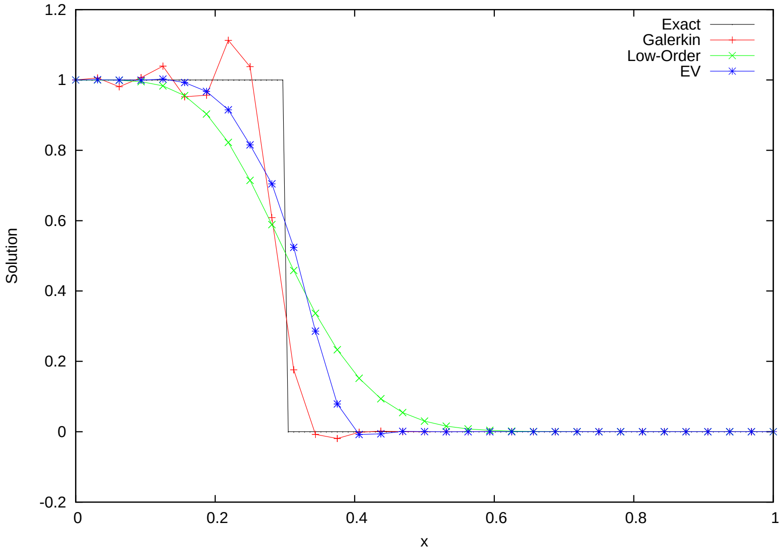
$$\mathbf{M}^C \frac{\mathbf{U}^{H,n+1} - \mathbf{U}^n}{\Delta t} + (\mathbf{A} + \mathbf{D}^{H,n})\mathbf{U}^n = \mathbf{b}^n \tag{24}$$

- The high-order diffusion matrix is computed just as the low-order counterpart, except that $\nu_K^{H,n}$ is used instead of ν_K^L:

$$D_{i,j}^{H,n} = \sum_{K \subset S_{i,j}} \nu_K^{H,n} b_K(\varphi_j, \varphi_i) \tag{25}$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

High-Order Scheme
Results Example



Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Flux Corrected Transport (FCT) Scheme
Correction Flux Definition

- Recall that FCT defines antidiffusive correction fluxes from a low-order, monotone scheme to a high-order scheme. Calling these fluxes **f**, this gives

$$\mathbf{M}^L \frac{\mathbf{U}^{H,n+1} - \mathbf{U}^n}{\Delta t} + (\mathbf{A} + \mathbf{D}^L)\mathbf{U}^n = \mathbf{b}^n + \mathbf{f} \tag{26}$$

- Subtracting the high-order scheme equation from this gives the definition of **f**:

$$\mathbf{f} \equiv -(\mathbf{M}^C - \mathbf{M}^L)\frac{\mathbf{U}^{H,n+1} - \mathbf{U}^n}{\Delta t} + (\mathbf{D}^L - \mathbf{D}^{H,n})\mathbf{U}^n \tag{27}$$

- Decomposing **f** into internodal fluxes $F_{i,j}$ such that $f_i = \sum_j F_{i,j}$, where $\Delta_{j,i}[\mathbf{y}]$ denotes $y_j - y_i$:

$$F_{i,j} = -M^C_{i,j}\Delta_{j,i}\left[\frac{\mathbf{U}^{H,n+1} - \mathbf{U}^n}{\Delta t}\right] + (D^L_{i,j} - D^{H,n}_{i,j})\Delta_{j,i}[\mathbf{U}^n] \tag{28}$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Flux Corrected Transport (FCT) Scheme

Implementation of Correction Fluxes

$$F_{i,j} = -M^C_{i,j}\Delta_{j,i}\left[\frac{\mathbf{U}^{H,n+1}-\mathbf{U}^n}{\Delta t}\right] + (D^L_{i,j} - D^{H,n}_{i,j})\Delta_{j,i}[\mathbf{U}^n]$$

```
for (; cell != endc; ++cell)
  // loop over lines of cell
  for (int line = 0; line < GeometryInfo<dim>::lines_per_cell; ++line)
  {
    if (!cell->line(line)->user_flag_set())
    {
      // mark line so that the same flux isn't unnecessarily recomputed
      cell->line(line)->set_user_flag();

      // get dof indices on line
      cell->line(line)->get_dof_indices(line_dof_indices);
      unsigned int i = line_dof_indices[0];
      unsigned int j = line_dof_indices[1];

      // compute correction flux F(i,j)
      double Fij = -MC(i,j) * (dUdt(j) - dUdt(i))
        + (DL(i,j) - DH(i,j)) * (U_old(j) - U_old(i));

      // store flux in global sparse matrix
      F.set(i, j,  Fij);
      F.set(j, i, -Fij); // F(j,i) = -F(i,j)
    }
  }
```

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Flux Corrected Transport (FCT) Scheme

FCT Overview

Introduction
- Motivation
- Objectives
- Outline

Methodology
- Formulation
- Time Discretization
- FCT Scheme Overview
- Low-Order Scheme
- High-Order Scheme
- FCT Scheme

Results

Conclusions

- Recall that the objective of FCT is to limit these antidiffusive fluxes to enforce some physical bounds.
- The chosen bounds take the form of the DMP satisfied by the low-order scheme:

$$W_i^- \leq U_i^{n+1} \leq W_i^+ \qquad \forall i \tag{29}$$

- This is achieved by applying a limiting coefficient $L_{i,j}$ to each internodal flux $F_{i,j}$:

$$\mathbf{M}^L \frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} + \mathbf{A}^L \mathbf{U}^n = \mathbf{b} + \mathbf{L} \cdot \mathbf{F} \tag{30}$$

- Each limiting coefficient is between zero and unity: $0 \leq L_{i,j} \leq 1$.
    - If all $L_{i,j}$ are zero, then the low-order scheme is produced.
    - If all $L_{i,j}$ are one, then the high-order scheme is produced.

Flux Corrected Transport (FCT) Scheme
Limiting Coefficients Definition

- The enforced bounds can be rearranged to bound the limited flux sums with bounds which we call $Q_i^\pm$:

$$Q_i^- \leq \sum_j L_{i,j} F_{i,j} \leq Q_i^+ \tag{31}$$

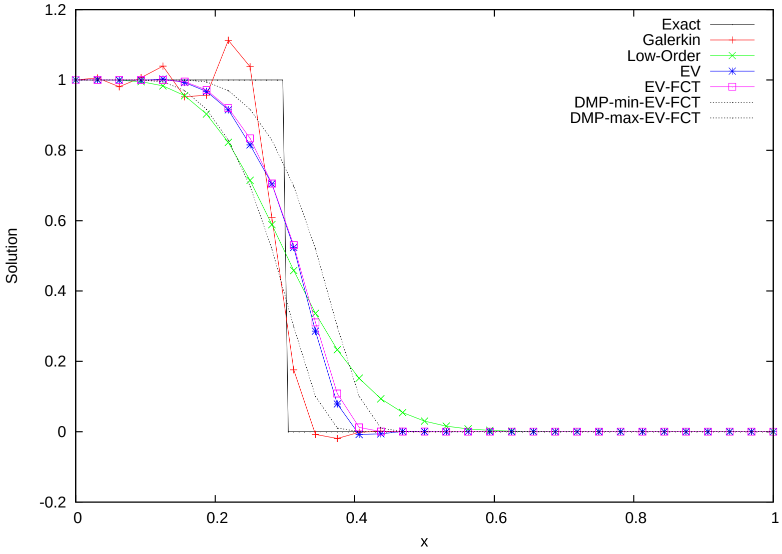
$$F_i^- \equiv \sum_{j:F_{i,j}<0} F_{i,j} \qquad F_i^+ \equiv \sum_{j:F_{i,j}>0} F_{i,j} \tag{32}$$

$$L_i^\pm \equiv \begin{cases} 1 & F_i^\pm = 0 \\ \min\left(1, \frac{Q_i^\pm}{F_i^\pm}\right) & F_i^\pm \neq 0 \end{cases} \tag{33}$$

$$L_{i,j} \equiv \begin{cases} \min(L_i^+, L_j^-) & F_{i,j} \geq 0 \\ \min(L_i^-, L_j^+) & F_{i,j} < 0 \end{cases} \tag{34}$$

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

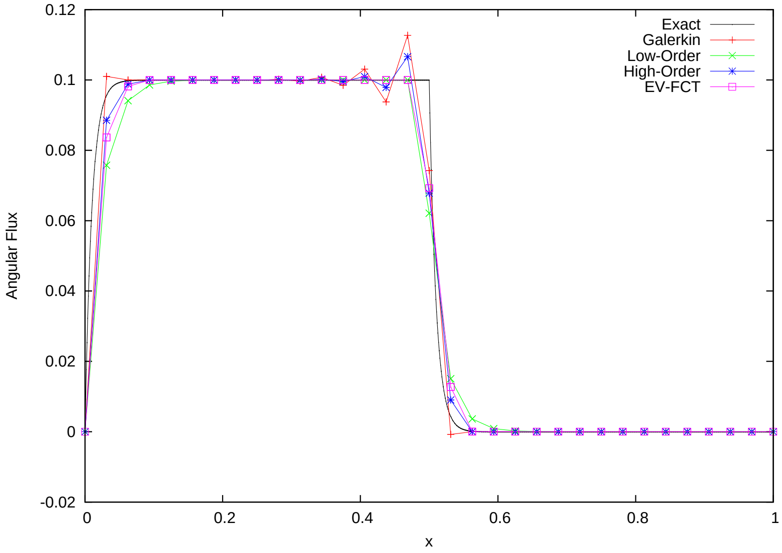
Flux Corrected Transport (FCT) Scheme
Results Example



Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

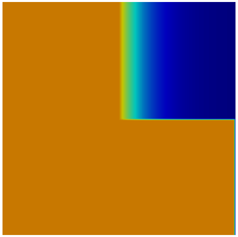
Results

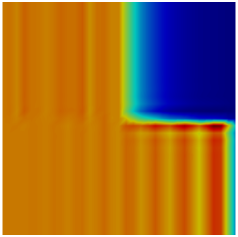
1-D Source Problem Results

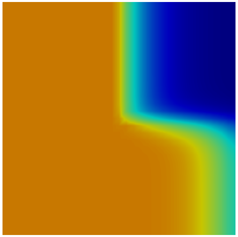


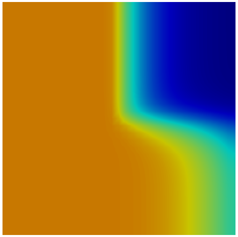
Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

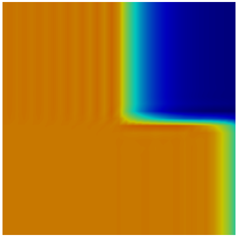
Results
2-D Void-to-Absorber Problem Results

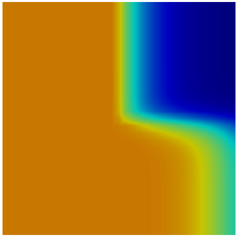

(a) Exact


(b) Galerkin


(c) Galerkin-FCT


(d) Low-order

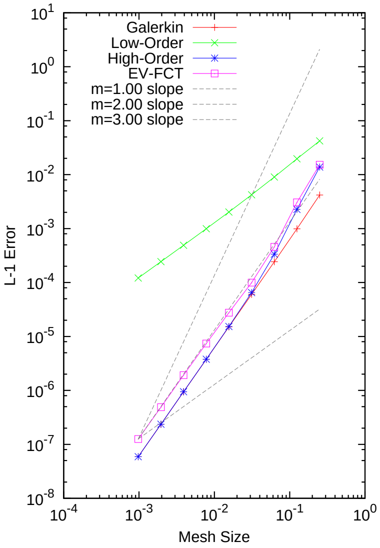

(e) EV

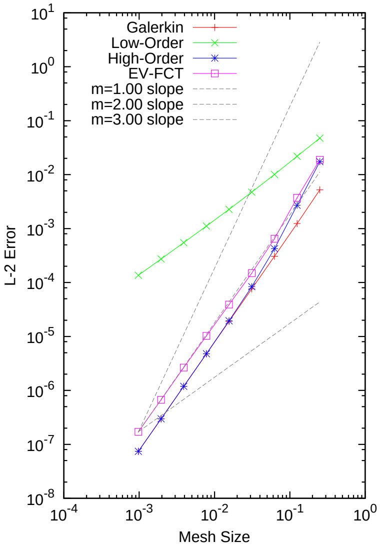

(f) EV-FCT

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Results

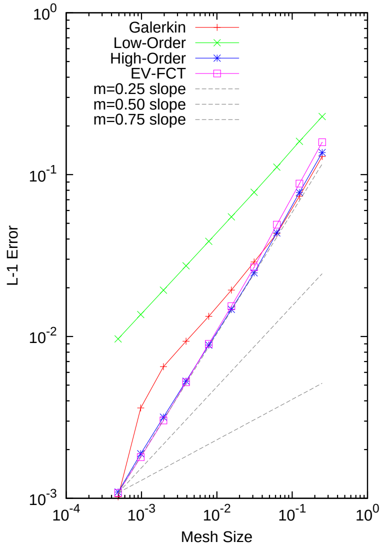
1-D Smooth Problem Convergence Results (Using FE)

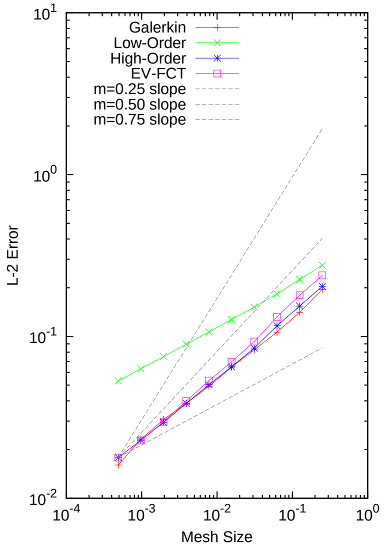




Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

1-D Non-smooth Problem Convergence Results (Using SSPRK33)





Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions

Conclusions

- The CFEM scheme presented for solving conservation law problems is
    - 2nd-order-accurate
    - Positivity-preserving
    - Not *guaranteed* monotone, but rarely not
    - Discrete-maximum-principle preserving
    - Valid in an arbitrary number of dimensions
    - Valid for general meshes
- Results were shown for the explicit, scalar, linear case. More results are in progress.
- `deal.II` provides the elements and flexibility necessary for an algorithm based on FCT.

Introduction
Motivation
Objectives
Outline
Methodology
Formulation
Time Discretization
FCT Scheme Overview
Low-Order Scheme
High-Order Scheme
FCT Scheme
Results
Conclusions