Scalable Riemann Solvers with the Discontinuous Galerkin Method for Hyperbolic Network Simulation

Aidan Hamilton [1] Jing-Mei Qiu [1] Hong Zhang [2]

[1]University of Delaware

[2]Illinois Institute of Technology

June 7, 2023

Table of Contents

Hyperbolic Conservation Laws on a Network

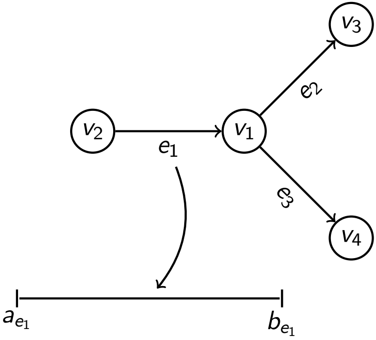
A **network** is a topological graph, i.e a couple (V, E) where

- E is a collection of intervals $[a_e, b_e]$
- V a collection of vertices connecting the intervals

A system of conservation laws naturally extends to a network edgewise

$$\partial_t u_e + \partial_x f(u_e) = 0,$$

- $u_e \in \mathbb{R}^m$
- $f : \mathbb{R}^m \to \mathbb{R}^m$
- Only considering $m = 1, 2$
- Challenge is coupling the 1D problems at vertices



Conservation Laws: Examples

Shallow Water (SWE):

$$\partial_t h + \partial_x(h\nu) = 0,$$
$$\partial_t h\nu + \partial_x(h\nu^2 + \frac{g}{2}h^2) = 0,$$

- $h(x,t)$: water height
- $\nu(x,t)$: water velocity

The Jacobian of the flux function has eigenvalues:

$$\lambda_1(u) = \nu - \sqrt{gh}, \qquad \lambda_2(u) = \nu + \sqrt{gh}$$

The system is *fluvial* if $|\nu| < \sqrt{gh}$, $\lambda_1 < 0$, $\lambda_2 > 0$.

LWR Traffic Flow

$$\partial_t \rho + f(\rho)_x = 0$$

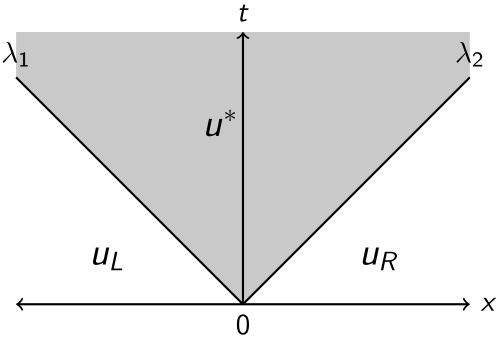
- $\rho(x,t)$: density of cars
- $\rho \in [0,1]$ by normalization
- f is C^2 and strictly concave
- $f(0) = f(1) = 0$
- for my examples $f(\rho) = 4\rho(1-\rho)$

1D Riemann Problem

A Riemann problem for a 1D hyperbolic conservation law is the PDE with a jump initial condition

$$u_0(x) = \begin{cases} u_L, & x < 0, \\ u_R, & x > 0, \end{cases}$$

For 2×2 systems solutions take the form

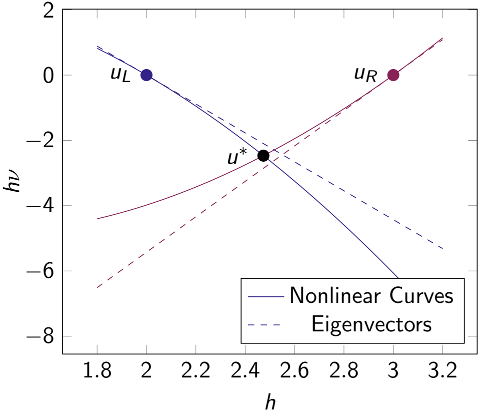


1D Riemann Problem II

The star state u^* is found by:

- finding the intersection of two nonlinear curves in phase space
- the curves are points that can be connected to $u_L,\ u_R$ by shock/rarefaction waves.

For the shallow water equations these solutions look like



Riemann Problem at a Vertex

Riemann Problem at a Vertex v[a]

$$\partial_t u_e + \partial_x f(u_e) = 0, \qquad t \in \mathbb{R}^+, e \in E(v)$$

$$u_e(x, t=0) = \overline{u}_e, \qquad x \in \begin{cases} \mathbb{R}^+ & e \text{ outgoing} \\ \mathbb{R}^- & e \text{ incoming} \end{cases}$$

where $\overline{u}_e$ are constant states.

- well-posedness requires an additional $deg(v)$ coupling conditions
- these are additional modeling choices
- For SWE a choice[b]

$$\begin{aligned} h_e^* &= h_{e'}^* && \text{height continuity} \\ \sum_{\substack{e \in E(v), \\ incoming}} \nu_e^*(v) &= \sum_{\substack{e \in E(v), \\ outgoing}} \nu_e^*(v) && \text{conservation of mass} \end{aligned}$$

- **Fundamental principle:** Connect each $\overline{u}_e$ to a u_e^* by a single wave-curve, that is outgoing from the vertex.

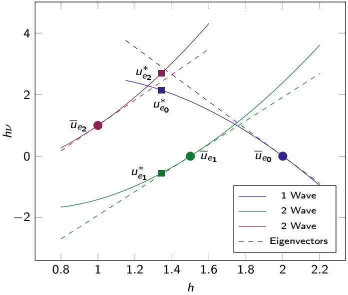


Figure: Example SWE Riemann problem for a 3-branch vertex.

---

[a] Colombo, Herty, and Sachers, "On 2×2 Conservation Laws at a Junction".

[b] Briani, Benedetto Piccoli, and Qiu, "Notes on RKDG Methods for Shallow-Water Equations in

LWR Riemman Problem

Model is an optimization problem[1]

- Consider a traffic distribution matrix $A \in \mathbb{R}^{\text{outdeg}(v) \times \text{indeg}(v)}$, where $A\mathbb{1} = \mathbb{1}$
  - Dictates what proportion of cars moving into a vertex from incoming road j, go to outgoing road i
- Let σ be the point that f attains its max, define function γ^{max}

for incoming roads j

$$\gamma_j^{max}(\rho) = \begin{cases} f(\rho), & \rho \leq \sigma \\ f(\sigma), & \rho > \sigma. \end{cases}$$

for outgoing roads i

$$\gamma_i^{max}(\rho) = \begin{cases} f(\sigma), & \rho \leq \sigma \\ f(\rho), & \rho > \sigma. \end{cases}$$

- Full problem, maximize flux through the vertex

$$\begin{aligned} &\max f(\rho_j^*) \\ &f(\rho_j^*) \in [0, \gamma^{max}(\rho_j)] \quad \text{for } j \text{ incoming} \\ &A(f(\rho_j^*)) \in [0, \gamma^{max}(\rho_0)] \times \ldots \times [0, \gamma^{max}(\rho_{\text{outdeg}(v)})] \end{aligned}$$

- Solve in PETSc using TAO ALLM

[1] Garavello and B. Piccoli, *Traffic Flow on Networks: Conservation Laws Models.*

Discretization

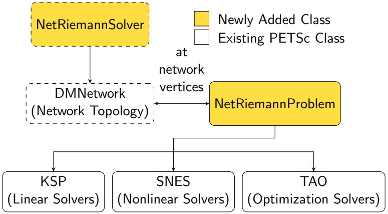
1. Discretize along edges using the standard RKDG framework.
   - Per element I_j evolve

$$\int_{I_j} v\partial_t u_h dx = \int_{I_j} f(u_h)\partial_x v dx - \left(\hat{f}_{j+\frac{1}{2}} v^-|_{x_{j+\frac{1}{2}}} - \hat{f}_{j-\frac{1}{2}} v^+|_{x_{j-\frac{1}{2}}} \right),$$

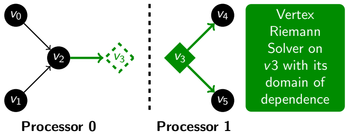
     where $v, u_h \in P^k(I_j)$, and $\hat{f}$ is the numerical flux.
   - Evolve with explicit SSP RK methods (TSSSP)
   - Use a characteristic-wise TVB slope limiter for stability (TSPostStage)
2. Solve network Riemann problem at vertices to get boundary fluxes.

Implementation Overview

- **DMNetwork:** manage distributed network
- **NetRiemannProblem:** specify and solve *local* vertex Riemann problems
- **NetRiemannSolver:** scalably solve a collection of NetRiemannProblems on vertices of a DMNetwork.



Distributed Vertex Riemann Solver



NetRiemannProblem: Solver Reuse

- **The problem:** Every vertex Riemann problem requires some sort of PETSc solver, SWE (SNES), Traffic (TAO).
- Thankfully, Riemann problem structures (Jacobians etc..) only depend on the vertex degree.
- **Solution** Solver objects can be cached and reused. Have maps:
  - (SWE) $deg(v) \rightarrow SNES$
  - (Traffic) $(indeg(v), outdeg(v)) \rightarrow TAO$
- Can reuse symbolic solves in direct LU factorization

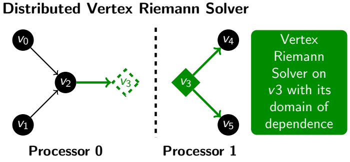
NetRiemannSolver: How it's used

Setup

1. Given a DMNetwork, it creates a cloned copy
2. NetRiemannProblem objects can be assigned to the vertices of DMNetwork
3. NetRiemannSolver generates a PetscSection and corresponding Vec for the Riemann problems at each assigned vertex.

Solve

1. The *local* Riemann data for are set into the Vec (NetRiemannSolver handles indexing, not PetscSection)
2. NetRiemannSolver then solves all Riemann problems
   - basic interlacing, solve local Riemann problems while waiting for communication of nonlocal Riemann data
   - optional SolveBegin; ... SolveEnd; interface for further interlacing

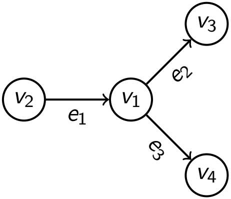


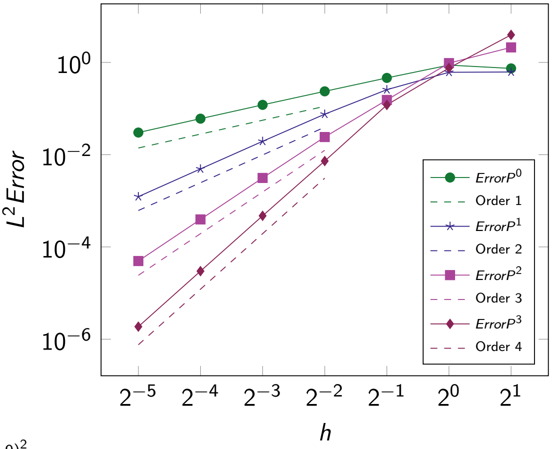
RHS Evaluation Algorithm

Algorithm 1 DG RHS Evaluation for a Hyperbolic Network

for vertex v in DMNetwork **do**
    for $e \in E(v)$ **do**
        Evaluate $u_e(v)$
        Place $u_e(v)$ in NetRiemannSolver
    end for
end for
NetRiemannSolverSolveBegin() ▷ Communication here
for Edge e in DMNetwork **do**
    for Cell c in edge e **do**
        DG update (not vertex fluxes)
    end for
end for
NetRiemannSolverSolveEnd() ▷ Communication here
▷ All vertex fluxes $\hat{f}_v$ are now available
for Edge e in DMNetwork **do**
    Update boundary fluxes using $\hat{f}_v$
end for

SWE Convergence Test





Initial Data

$$
\begin{aligned}
h_1(x,0) &= 1 + e^{-5(x-9)^2}, \\
h_1\nu_1(x,0) &= h_1(x,0)/2, \\
h_2(x,0) &= h_3(x,0) = 1 + e^{-5(10-9)^2}, \\
h_2\nu_2(x,0) &= h_3\nu_3(x,0) = h_2(x,0)/4.
\end{aligned}
$$

Dam Break on Shallow Water Network

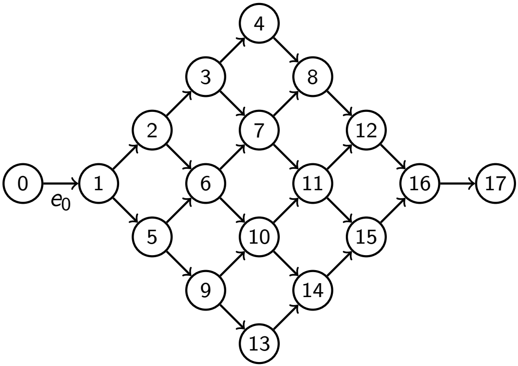


Figure: Grid graph variant used in the dam break test

Traffic Network Example

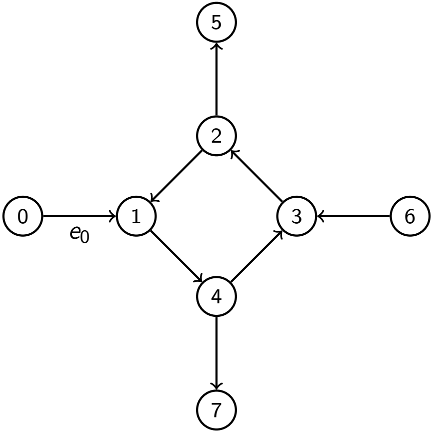
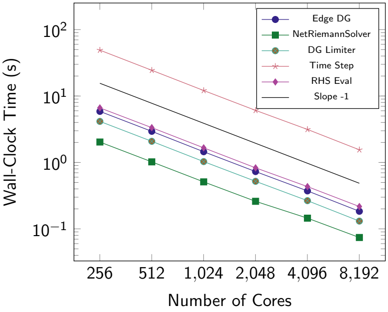


Figure: Traffic Circle Network

Scalable Simulation of Mississippi River Network

- Simulate SWE on the Mississippi river system, with data obtained from the NHDPlus dataset[a]
- contains 892,740 edges and 872,300 vertices
- Discretize with elements of length 10m (the resolution of the dataset), P^2 basis and a 5-stage SSPRK2 method[b], with $\approx 10^9$ degrees of freedom.
- Simulation conducted on the Theta supercomputer at ALCF. Theta has 4,392 nodes, each with 64 1.3GHz Intel Xeon Phi 7230 cores with 16 GiB of MCDRAM per node.



[a] McKay et al., "NHDPlus Version 2: user guide".

[b] Ketcheson, "Highly Efficient Strong Stability-Preserving Runge–Kutta Methods with Low-Storage Implementations".

Future Work

1. Implement Blood Flow Networks
2. Large scale simulation for traffic flow
3. General code improvements. Move out of a PETSc branch into its own separate thing.
4. Move various pieces of my work into PETSc, particular DMNetwork improvements.

References

Briani, Maya, Benedetto Piccoli, and Jing-Mei Qiu. "Notes on RKDG Methods for Shallow-Water Equations in Canal Networks". In: *J. Sci. Comput.* 68.3 (Sept. 2016), pp. 1101–1123. ISSN: 0885-7474. DOI: 10.1007/s10915-016-0172-2. URL: https://doi.org/10.1007/s10915-016-0172-2.

Colombo, R. M., M. Herty, and V. Sachers. "On 2×2 Conservation Laws at a Junction". In: *SIAM Journal on Mathematical Analysis* 40.2 (2008), pp. 605–622. DOI: 10.1137/070690298. eprint: https://doi.org/10.1137/070690298. URL: https://doi.org/10.1137/070690298.

Garavello, M. and B. Piccoli. *Traffic Flow on Networks: Conservation Laws Models*. AIMS series on applied mathematics. American Institute of Mathematical Sciences, 2006. ISBN: 9781601330000. URL: https://books.google.com/books?id=LVwYAwAACAAJ.

Ketcheson, David I. "Highly Efficient Strong Stability-Preserving Runge–Kutta Methods with Low-Storage Implementations". In: *SIAM Journal on Scientific Computing* 30.4 (2008), pp. 2113–2136. DOI: 10.1137/07070485X. eprint: https://doi.org/10.1137/07070485X. URL: https://doi.org/10.1137/07070485X.

McKay, L et al. "NHDPlus Version 2: user guide". In: *National Operational Hydrologic Remote Sensing Center, Washington, DC* (2012).