EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Continuum Traffic Simulation

J. Sewall and D. Wilkie and P. Merrell and M. C. Lin

University of North Carolina at Chapel Hill

Abstract

We present a novel method for the synthesis and animation of realistic traffic flows on large-scale road networks. Our technique is based on a continuum model of traffic flow we extend to correctly handle lane changes and merges, as well as traffic behaviors due to changes in speed limit. We demonstrate how our method can be applied to the animation of many vehicles in a large-scale traffic network at interactive rates and show that our method can simulate believable traffic flows on publicly-available, real-world road data. We furthermore demonstrate the scalability of this technique on many-core systems.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based modeling Simulation and Modeling [I.6.8]: Types of Simulation—Animation

1. Introduction

As the world's population grows, traffic management is becoming a mounting challenge for many cities and towns across the globe. Increasing attention has been devoted to the modeling, simulation, and visualization of traffic flows to investigate causes of traffic congestion and accidents, to study the effectiveness of roadside hardware, signs and other barriers, to improve policies and guidelines related to traffic regulation, and to assist urban development and the design of highway and road systems. In addition, with increasing volumes of traffic data and software tools capable of visualizing urban scenes (such as Google Maps and Virtual Earth), there is a growing need to add realistic street traffic in virtual worlds for virtual tourism, feature animation, special effects, traffic monitoring, and many other applications.

There is a vast amount of literature on modeling and simulation of traffic flows, with existing traffic simulation techniques generally focusing on either agent-based *microscopic* or continuum-based *macroscopic* models. However, little attention has been paid to the possibility of extending macroscopic models to produce detailed 3D animations and visualization of traffic flows. In this paper, we present a fast method for efficient simulations of large-scale, real-world networks of traffic using continuum dynamics that maintains discrete vehicle information to display each vehicle. Our continuum approach describes realistic behavior and is efficient — we describe the movement of many vehicles with a single computational cell — while individual vehicle information facilitates visual representation and allows per-vehicle information to influence the large-scale simulation. Our technique produces detailed, real-time animations of enormous traffic flows on multi-lane highways and winding rural roads at *more than* 100*x faster than real time*.

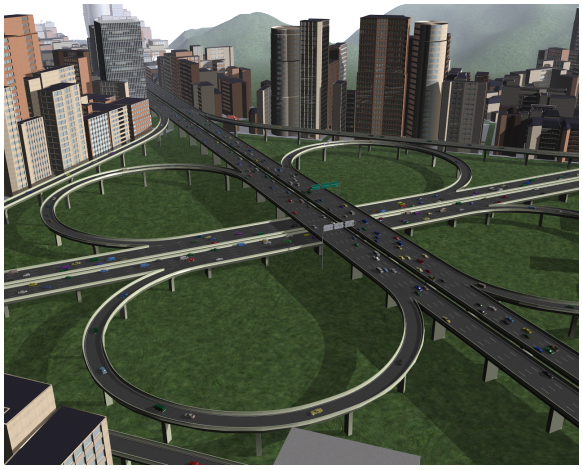

Figure 1: *A bird's-eye view of animated traffic*

Our approach adapts a single-lane continuum flow model to handle multi-lane traffic by introducing a novel model of lane changes and using a discrete visual representation for

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

each vehicle. We compare our technique's efficiency with that of agent-based methods, and demonstrate that our technique can effectively utilize the processing power of many-core shared memory architectures for scalable simulation. Figure 1 shows a snapshot of the highway traffic in an urban scene simulated by our continuum model.

2. Related work

The ubiquity of vehicle traffic in everyday life has generated considerable interest in models of traffic behavior, and in the last 60 years, a large body of research in the area has been developed. The problem of traffic simulation — given a road network, a behavior model, and initial car states, how the traffic in the system evolves – has been extensively studied. Most of the existing methods are designed to explore specific phenomena, such as traffic jams and unsteady, "stop-and-go patterns" of traffic, or to evaluate network configurations to aid in real-world traffic engineering.

The most popular category of traffic simulation methods is broadly termed *microscopic* simulation. This classification includes discrete agent-based methods, wherein each car is treated as a discrete autonomous agent with arbitrarily complex rules governing their behavior. Most agent-based methods use some form of the "car-following" set of rules described in the dissertation of Gerlough [Ger55] and the work of Newell [New61]. The most prominent traffic simulation systems, such as NETSIM [BdLCW82], INTEGRATION [ABB*97], and MITSIM [YK96], have been agent-based.

Nagel and Schreckenberg [NS92] applied cellular automata to the problem traffic simulation. The efficiency and simplicity of these models has led to a great deal of interest and extensions to the Nagel-Schreckenberg model. The survey by Chowdhury et al. [CSS00] and the work of Treiber and Helbing [TH01] describe these models in detail.

Continuous, or *macroscopic*, models of traffic flow have been studied since the seminal papers of Lighthill and Whitham [LW55] and Richards [Ric56]. These papers propose a nonlinear scalar conservation law (known as the LWR model) for density of traffic and the resulting flow; the existence of shock and rarefaction waves in the solutions is demonstrated as well. Newell [New61] extended this model to certain non-equilibrium cases, and Payne [Pay71] and Whitham [Whi74] developed a second-order system of equations derived from the equations of gas dynamics capable of describing types of non-equilibrium flow. Limitations of the Payne-Whitham (PW) model became apparent; for certain configurations, the model predicts negative velocities of traffic! Daganzo [Dag95] traced the deficiencies in the model to the isotropic nature of the gas dynamics from which the model was derived. Aw and Rascle [AR00] and Zhang [Zha02] described a modification of the PW model that eliminated the nonphysical behavior; we use their 'ARZ' model to handle the continuum flow of traffic along lanes.

A third class of simulation methods, called *mesoscopic* methods, uses a continuum representation of traffic but uses Boltzmann-type mesoscale equations to traffic dynamics. This approach was pioneered by Prigogine and Andrews [PA60] and improved upon by Nelson et al. [NBS97], Shvetsov and Helbing [SH99], and others.

There exists some long-standing interest in the simulation of aggregated, agent-like entities. Lamarche and Donikan [LD04] present a technique for the simulation of crowds using high-level roadmaps for navigation and a local reactive scheme for collision avoidance and Paris et al. [PPD07] describe an improved reactive collision avoidance method using a sophisticated prediction model. We refer the readers to recent surveys, such as [PAB08, PKL08], for more detailed reviews on virtual crowds and multi-agent simulation.

There have been comparatively few papers on the animation of vehicles in graphics; Go et al. [GVK05] describe a method for animating vehicles from a control and motion-planning perspective, and the technique of van den Berg et al. [vdBSLM09] reconstructs trajectories given starting and ending pairs of vehicle data using a prioritized planning technique.

Hyperbolic conservation laws have received comparatively little attention in the graphics community; Sewall et al. [SGTL09] describe a technique for animating shockwaves in a gas. The ARZ model our technique built up shares certain similarities with those same gas laws — including the presence of shocks and rarefactions.

3. Method

In this section, we describe the data structures and our method for the simulation of traffic flow in detail.

3.1. Overview

We simulate traffic on a network of roads; each road has one or more lanes and is connected to other roads via interchanges. See Sec. 3.2 for details on how we represent these networks, and how they may be synthesized or adapted from real-world data.

Our simulation describes the flow of traffic through a system of nonlinear *hyperbolic conservation laws* that represent traffic as a continuum along lanes. Many systems of equations have been developed to describe the flow of traffic with varying degrees of completeness. Our system augments the model recently proposed by Aw and Rascle [AR00] and Zhang [Zha02], which we refer to as the Aw-Rascle-Zhang (ARZ) model, following Lebacque [LMHS07]. This equation is described in Sec. 3.3.3.

To obtain a numerical solution to the ARZ equations, we use a Finite Volume Method (FVM) spatial discretization combined with a Riemann solver to determine the fluxes between adjacent computational cells. See Sec. 3.3.1 for details

on the FVM and how it applies to conservation laws like the ARZ equations, and see Sec. 3.3.4 for a description of the Riemann solver we use.

The ARZ equations describe the motion of traffic along individual lanes; to handle merges and lane changes, our solution combines continuum-level dynamics with discrete car information. We describe this representation in detail in Sec. 3.5.

The continuum approach to traffic simulation has numerous advantages in efficiency and robustness; however, to display the motion of traffic for visualization and animation purposes, we use "car particles", or *carticles*, that represent the individual cars in the flow of traffic. These carticles are moved along by the underlying continuum flow, and can play decision-making roles in parts of the simulation — particularly in regards to lane changes and merges. Sec. 3.4 describes these in detail.

Finally, while our method is efficient enough to simulate heavy volumes of traffic on large networks on a single processor, it is simple to parallelize and scales well on multi-core machines so as to make colossal-scale simulations possible. We describe how our method can be implemented to further benefit from parallel systems in Appendix B.

3.1.1. Basic Solution Procedure

We simulate traffic by taking successive iterations (i.e. time steps) of our solver. For a given timestep, our solver operates as follows:

1. Solve the ARZ equations along each lane, taking into account boundary information.
2. Initiate and advance lane changes.
3. Advance carticles according to the solution in each lane.
4. Apply relaxation of relative velocity.
5. Update network state.

These steps will be explained in greater detail below.

3.2. Representation of Road Networks

Our simulation operates in the domain of road networks, and the fidelity and realism of the results depends on the quality and detail of the road network. In this section, we present a robust data structure for representing a road network suitable for our simulation.

3.2.1. Features of Road Networks

Road networks can be arbitrarily complicated; in addition to information about lanes, we could also have descriptions of road quality and conditions, information about driveways and parking spaces along the road, and a host of other types of data.

Our simulator currently handles multi-lane road segments with varying speed limits, and our data structure is designed to efficiently store and answer queries about such information. However, additional of information can be easily included in our data structure.

Roads A road network could be described as a collection of roads and information about how they are connected — roads have some spatial description of the path they take, a number of lanes, and they stop and end at other roads. However, such a naïve description fails to properly capture many features we desire in a road network; how should we describe a highway interchange, where several lanes split off from roads, take a curving path, and join another road? For this reason, our data structure treats roads simply as descriptions of spatial information; in our implementation, each road provides a sequence of connected lines that describes its path in space.

Lanes We have chosen the lane to be the atomic data type in our data structure. This is motivated by their relationship to roads and other lanes — in that a single lane may belong to many roads and be adjacent to many different lanes along its length — and by our simulation methodology. Since we solve the ARZ equations for the flow of traffic along each lane, it is more efficient to have long and unbroken lanes to minimize the need to handle special-case boundary conditions.

Each lane is parametrized by its length in space to the unit interval, and properties of the lane — such as speedlimit, the road to which it belongs and obtains its spatial description from, and the other lanes it may be adjacent to — are mapped to this parametric interval.

This parametrization of lane properties is motivated by the need for various queries during simulation. While advancing the solution along each lane, for example, we wish to know what the speed limit is in the current cell, and if it differs from that in the next. Similarly, when merging traffic, we wish to know which lanes (if any) are to the left and to the right of the current position along a lane.

Each adjacency between lanes in our system carries with it additional information about how 'desirable' the corresponding lane change is; this is captured with a probability distribution and allows our system to model capture the behavior of traffic around frequently-used offramps.

3.3. Numerical Traffic Simulation with Gas-like Laws

We simulate the flow of traffic along lanes with a numerical discretization of a *hyperbolic conservation law*. This is a class of partial differential equations commonly associated with physical laws and with gas dynamics in particular. The basis for our traffic flow, the so-called Aw-Rascle-Zhang (ARZ) model ([AR00], [Zha02]), is one such law that is closely related to the hyperbolic systems of equations that describe gas dynamics.

3.3.1. Conservation Laws & the Finite Volume Method

The ARZ model is a conservation law; it fits the general model

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \tag{1}$$

where subscripts denote differentiation, $\mathbf{q}$ is a vector-valued quantity of unknowns, and $\mathbf{f}(\mathbf{q})$ is a vector-valued function of the unknowns. The choice of $\mathbf{f}$ (known as the *flux function*) uniquely characterizes the dynamics of the system. Solutions to Eq. (1) may be readily discretized with the Finite Volume Method (FVM) of numerical discretization to obtain the following:

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n - \frac{\Delta t}{\Delta x}\left[\mathbf{f}(\mathbf{q}(b)) - \mathbf{f}(\mathbf{q}(a))\right] \tag{2}$$

Here $\mathbf{Q}_i^n = \mathbf{Q}_i(t^n)$ and $\Delta t = t^{n+1} - t^n$. More information on hyperbolic conservation laws and the FVM can be found in Leveque [Lev02].

3.3.1.1. Flux Calculations Eq. (2) is a straightforward update scheme; what remains to be computed are the quantities $\mathbf{f}(\mathbf{q}(b))$ and $\mathbf{f}(\mathbf{q}(a))$ — that is, the flux that occurs at the boundaries between cells. This can be difficult for nonlinear $\mathbf{f}$ (such as that found in the ARZ system of equations) and accounts for the bulk of the computation in our numerical scheme. The problem of determining these fluxes is termed the *Riemann problem*, and we discuss its solution for the ARZ model in Sec. 3.3.4.

3.3.1.2. Computation of Δx for Each Lane Each lane j in our simulation is divided into a number of discrete cells of equal length Δx_j. The cell length Δx_j varies only slightly from lane to lane; when preparing the simulation, we suggest a "target" Δx that all lanes should have, and for each lane j of length L_j, we determine the number of cells in the lane N_j and the related cell length Δx_j as follows:

$$N_j = \left\lfloor \frac{L_j}{\Delta x} \right\rfloor, \quad \Delta x_j = \frac{L_j}{N_j} \tag{3}$$

This ensures that all cells in a given lane have the same length. So long as all lanes are at least Δx in length, we are ensured that $\Delta x_j > \frac{\Delta x}{2} \quad \forall j$. In general, we choose the target Δx to be greater than the length of the longest vehicle in the simulation (see Sec. 3.4).

3.3.1.3. Timestep Restrictions The time step Δt in Eq. (2) must be chosen to satisfy the Courant-Friedrichs-Lewy (CFL) condition for the integration to be stable. According to this condition, we must have:

$$\Delta t < \min_j \left(\frac{\Delta x_j}{\lambda_{\max j}} \right) \tag{4}$$

With the minimum taken over all lanes j. $\lambda_{\max j}$ is the *maximum speed* in lane j at the current timelevel; these speeds are determined while solving the Riemann problem at each interface (given by Eqs. (9) and (10)). This has ramifications on how we choose to apply Eq. (2); since we must use the same Δt at each cell, we must compute *all* speeds λ before integrating any cells.

It should be noted that obeying the CFL condition does not mandate that simulation timesteps be smaller than what would be desired for display. For example: in a network with a maximum speed of 100 km/s, $\lambda_{\max}$ will not exceed $27.\bar{7}$ m/s; this gives $\Delta t < \Delta x \cdot 0.0036 s/m$. In a simulation, we are free to choose any Δx — generally, this is a multiple of car lengths. Even for the smallest cell size we have used in our experiments — $\Delta x = 9m\ (= 2 \times 4.5m)$, this gives $\Delta t < 0.324s$, which is on par with the frame rate desired for most conventional visualization techniques.

3.3.2. Numerical Update Procedure

Given cell values $\mathbf{Q}^n$ at time t_n for all lanes j, we compute $\mathbf{Q}^{n+1}$ as follows:

1. At each interface between cells, compute the speeds λ_i and fluxes F_i^n by solving the Riemann problem at that interface (described in Sec. 3.3.4)
2. Find the speed with largest magnitude and compute timestep length Δt as described in Sec. 3.3.1.3.
3. For each cell i, advance to next time $\mathbf{Q}^{n+1}$ using Eq. (2) and the fluxes from step 1.

3.3.3. Aw-Rascle-Zhang Model

The ARZ model ([AR00], [Zha02]) can be written as a conservation law of the form

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0, \quad \mathbf{q} = \begin{bmatrix} \rho \\ y \end{bmatrix}, \ \mathbf{f}(\mathbf{q}) = \begin{bmatrix} \rho u \\ y u \end{bmatrix} \tag{5}$$

Here ρ is the density of traffic, i.e. "cars per car length", u is the velocity of traffic, and y the "relative flow" of traffic.

y, ρ, and u are related by the equation:

$$y(\rho, u) = \rho\,(u - u_{\text{eq}}(\rho)) \tag{6}$$

where $u_{\text{eq}}(\rho)$ is the "equilibrium velocity" for ρ. There is a single criterion on u_{eq} that must be satisfied (see Appendix A) on this function; the following is satisfactory for our purposes:

$$u_{\text{eq}}(\rho) = u_{\max}\left(1 - \rho^{\gamma}\right) \tag{7}$$

In the above equation, $u_{\max}$ is the speed limit of the road, and γ some parameter > 0. Using Eqs. (6) and (7), we can write u in terms of y and ρ:

$$u(\rho, y) = \frac{y}{\rho} + u_{\text{eq}}(\rho) \tag{8}$$

In what follows, we shall interchangeably use $u_{\text{eq}}(\rho)$ and u_{eq} as well as $u(\rho, y)$ and u.

It is possible to rewrite the ARZ equations in terms of ρ and u (the *primitive* variables), but the system will not longer fit Eq. (1). ρ and y are the *conservative* variables for the ARZ system; these quantities have special significance that allows

us to solve for them using all of the tools associated with conservation laws (i.e. Eq. (2), and much of what follows).

3.3.4. Basic Riemann Problem for the ARZ model

To compute fluxes such as $\mathbf{f}(\mathbf{q}(b))$ and $\mathbf{f}(\mathbf{q}(a))$ in Eq. (2), we must be able to determine the value of $\mathbf{q}$ between the piecewise-constant states in adjacent cells.

Given initial constant states $\mathbf{q}_l$ for $x < 0$ and $\mathbf{q}_r$ for $x > 0$ (with components which we term ρ_l, y_l, and u_l and ρ_r, y_r, and u_r, respectively), what happens for $t > 0$? In the case of the ARZ model, there are several distinct possibilities. Depending on the relative values of $\mathbf{q}_l$ and $\mathbf{q}_r$, we expect the solution to consist of two or more distinct "regions" of self-similar solutions traveling with varying speeds. See Fig. 2 for a graphical depiction of the Riemann problem. The following is an abbreviated analysis of the ARZ equations; for more detail, please refer to [Sew10].

3.3.4.1. Waves and speeds The eigenstructure of the Jacobian of the flux function defined in Eq. (5) is the key to determining the speeds in the system (for Eq. (4)) and the structure for the Riemann problem; the eigenvalues are:

$$\lambda_0 = u + \rho u'_{\text{eq}} \tag{9}$$

$$\lambda_1 = u \tag{10}$$

with corresponding eigenvectors

$$\mathbf{r}_0 = \begin{bmatrix} 1 \\ \frac{y}{\rho} \end{bmatrix}, \quad \text{and} \quad \mathbf{r}_1 = \begin{bmatrix} 1 \\ \frac{y}{\rho} - \rho u'_{\text{eq}} \end{bmatrix} \tag{11}$$

Field Classification We can regard the pair of eigenvalues and eigenvectors as distinct 'families' of solution fields — we refer to solutions associated with λ_0 and $\mathbf{r}_0$ as the '0-family' of solutions, and those associated with λ_1 and $\mathbf{r}_1$ as the '1-family' of solutions.

While the ARZ equations are clearly nonlinear (5), different families may exhibit different characteristics — some linear, some nonlinear. By classifying these fields as either *genuinely nonlinear* or *linearly degenerate*, we obtain information about what types of solutions to expect. For more detail on how this classification is performed, see Appendix A. For this system, the first family of solutions (those associated with λ_0 and $\mathbf{r}_0$) is genuinely nonlinear — the related waves deform with propagation. The second family of solutions (λ_0 and $\mathbf{r}_0$) is linearly degenerate and behaves as a linear system.

3.3.4.2. Intermediate State To compute the left- and right-going fluctuations in our update scheme, we need to compute the value of $\mathbf{q}_0$ — that is to say, the value of $\mathbf{q}$ at $x = 0$ for $t > 0$, given $\mathbf{q}_l$ and $\mathbf{q}_r$. In general, $\mathbf{q}_0$ can be $\mathbf{q}_l$, $\mathbf{q}_r$, or the intermediate value $\mathbf{q}_m$. The left state $\mathbf{q}_l$ and the intermediate state $\mathbf{q}_m$ are separated by the line $x = \lambda_0 t$, and $\mathbf{q}_m$ and $\mathbf{q}_r$ are separated by the line $x = \lambda_1 t$. For the ARZ model we are considering, we know that $\lambda_1 = u_r > 0$ and therefore

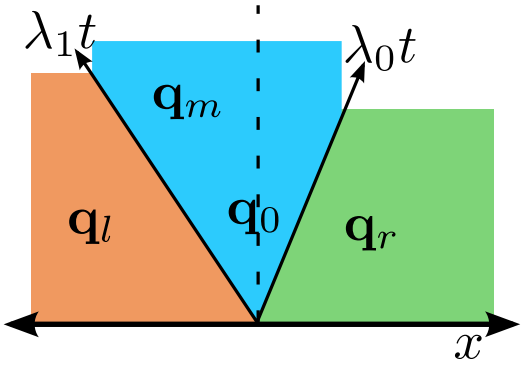

Figure 2: *A schematic of a Riemann problem; the up-axis represents both time and* $\mathbf{Q}$*. Here, we see an intermediate state* $\mathbf{Q}_m$ *arising between* $\mathbf{Q}_l$ *and* $\mathbf{Q}_r$*. We seek the value of* $\mathbf{Q}_0$ *— in this example, where* $\lambda_0 < 0$ *and* $\lambda_1 > 0$*,* $\mathbf{Q}_0 = \mathbf{Q}_m$*.*

$\mathbf{q}_0$ cannot be $\mathbf{q}_r$. We are left to determine if $\mathbf{q}_0$ is $\mathbf{q}_l$ or $\mathbf{q}_m$, and if $\mathbf{q}_m$, what the value of $\mathbf{q}_m$ along $x = 0$ is. Using the so-called *Riemann invariants* of the system of equations (see Appendix A), we determine:

$$\rho_m = \left(\rho_l^\gamma + \frac{u_l - u_r}{u_{\max}} \right)^{\frac{1}{\gamma}} \tag{12}$$

$$u_m = u_r \tag{13}$$

3.3.5. Structure of the Riemann problem

For $\mathbf{q}_m$ to exist, the speeds of the system (Eqs. (9) and (10)) must be distinct. This occurs when $\rho_l > 0$; in this case, there are three distinct regions of the solution: $\mathbf{q}_l$ for $x \leq \lambda_0 t$, $\mathbf{q}_m$ for $\lambda_0 < \frac{x}{t} < \lambda_1$, and $\mathbf{q}_r$ for $x \geq \lambda_1 t$. In the case where $\rho_l = 0$, $\lambda_0 = \lambda_1$ and $\mathbf{q}_m$ vanishes. We shall deal with cases where $\rho_l = 0$ or $\rho_r = 0$ separately below.

We know from Sec. 3.3.4.1 that the 0-family solutions are always shock or rarefaction waves. To determine which, we must consider the 0-family speeds λ_{0l} and λ_{0m} on either side of the nonlinear wave:

$$\lambda_{0l} = u_l - u_{\max} \gamma \rho_l^\gamma \tag{14}$$

$$\lambda_{0m} = u_r - u_{\max} \gamma \rho_l^\gamma + \gamma (u_r - u_l) \tag{15}$$

When $\lambda_{0l} < \lambda_{0m}$, the solution is a rarefaction, and when $\lambda_{0l} > \lambda_{0m}$, the solution is a shock. From Eqs. (14) and (15), we can determine that

$$\lambda_{0l} > \lambda_{0m} \Leftrightarrow u_l > u_m \tag{16}$$

3.3.5.1. Classification of Solutions One can identify 6 distinct conditions on the states $\mathbf{q}_l$ and $\mathbf{q}_r$ that determine the structure of the solution of any Riemann problem in the system. The derivation of these cases is discussed at length in Appendix A.

3.3.6. Inhomogeneous Riemann Problem

The above discussion on the solution to the Riemann problem for the ARZ system of equations has assumed that $u_{\max}$ remains constant in space — i.e. that the speedlimit on either side of the interface is the same.

Clearly speedlimits vary from road to road and change even along a single lane, and the effects of these variations in speedlimits have discernible effects on traffic flow. At a decrease in speedlimit, we expect traffic to slow and increase in density, while an increase in speedlimit might cause traffic to accelerate and rarefy.

Whereas the solution to the Riemann problem developed above is a *homogenous* Riemann problem, when speedlimits on either side of a cell interface differ, we wish to solve the *inhomogeneous* Riemann problem.

Formally, given initial constant states $\mathbf{q}_l$ (subject to speedlimit $u_{\max l}$) for $x < 0$ and $\mathbf{q}_r$ for $x > 0$ (subject to speedlimit $u_{\max r}$), what happens for $t > 0$? We expect solutions to follow the same basic structure of the homogeneous Riemann problem described previously, but rather than have a single intermediate state $\mathbf{q}_m$ emerge between $\mathbf{q}_l$ and $\mathbf{q}_r$, we expect there to be as many as two intermediate states divided at $x = 0$ by the jump in $u_{\max}$. We term these states $\mathbf{q}_{ml}$ and $\mathbf{q}_{mr}$.

Lebacque et al. introduced the concepts of *supply* and *demand* to solve for $\mathbf{q}_{ml}$ and $\mathbf{q}_{mr}$; see [LHSM05] for the details of their approach.

3.3.7. Relaxation of 'Relative Flow'

We have hitherto discussed the ARZ equations as homogeneous conservation laws — they fit the form of the conservation law shown in Eq. (1) with the right-hand side of the equation as 0. This ensures that each primitive variable ρ and y is conserved in the system. Such a property is useful when describing natural phenomena, since most such equations are derived from conservation principles themselves.

In our case, while ρ should certainly be conserved in regular traffic (we don't want cars to appear or disappear spontaneously), we may wish to relax the conservation of relative flow y. As discussed in Aw and Rascle [AR00], conserving this quantity leads to an unnatural dependence on initial conditions; vehicles with no traffic ahead of them (such as situations found in Case 5 in Appendix A) will not accelerate beyond a quantity related to their initial value of y.

To correct this, Aw and Rascle suggest adding a small relaxation term to the right-hand side of Eq. (5). Rather than have the two quantities equal to the zero vector $[0,0]^{\mathrm{T}}$, we include a scaled quantity $-\frac{y}{\rho\tau} = \frac{u_{\mathrm{eq}}-u}{\tau}$ to the right-hand side of the second equation. Here τ is a time constant (typically greater than 1) representing the propensity for acceleration in the system. The modified ARZ system then becomes:

$$\begin{aligned} \rho + \rho u &= 0 \\ y + yu &= \frac{u_{\mathrm{eq}} - u}{\tau} \end{aligned} \tag{17}$$

The first equation is unchanged, while the second encourages the velocity of traffic to slowly increase towards the speed limit.

Although we have modified our underlying equations, the previously presented method for the solution of the Riemann problem remains unchanged. To account for the new system shown in Eq. (17), we take a relaxation step after performing the integration step shown in Eq. (2). The quantity obtained from that update ($\mathbf{Q}^{n+1}$) we instead label $\mathbf{Q}^{n+1^*}$ and the following is applied:

$$y^{n+1} = y^{n+1^*} - \Delta t \frac{u_{eq}^{n+1^*} - u^{n+1^*}}{\tau} \tag{18}$$

Note that the equation for ρ remains unchanged.

3.3.8. Boundary Conditions

Each lane necessarily has a start and end, and to properly integrate the solution at the first and last cells in each lane as per Eq. (2), we must have some way of determining the flux at these boundaries. At such boundaries, lacking one of $\mathbf{q}_l$ or $\mathbf{q}_r$, we must solve a 'one-sided' Riemann problem.

Inflow We prescribe upstream traffic for lanes that start at a boundary, possibly in a time-dependent manner. We perform a full Riemann solve as in Sec. 3.3.4 with the right state $\mathbf{q}_r$ the first cell of the relevant lane, and a given $\mathbf{q}_l$ the incoming traffic.

'Starvation' Inflow As a special case, when *no* traffic should flowing into lane, there is no 1-wave, and the 2-wave simply propagates to the left — the numerical flux is obtained through a Riemann solver similar to that used for case 4 in Appendix A. Certainly $\lambda_1 = u_r \geq 0$, and as there is no wave from the 1-family, we know that $\mathbf{q}_0 = \mathbf{q}_l = [0,0]^{\mathrm{T}}$

Outflow For lanes that end at a boundary, we may impose a specific state, just as with the inflow case — we could, for example, impose a high-density, low-velocity condition that captures the behavior of an out-of-network traffic jam, or we could have a zero-density, high-velocity condition to represent an empty road.

Stopped Outflow When no traffic should flow out of a lane, a 1-wave of increasing density and decreasing velocity travels backwards through the lane. The one-sided Riemann solver for this case is identical to case 1 in Appendix A with $u_r = 0$. Substituting this into Eqs. (12) and (13), we have:

$$\rho_m = \left(\rho_l^{\gamma} + \frac{u_l}{u_{\max}}\right)^{\frac{1}{\gamma}} \tag{19}$$

$$u_m = 0 \tag{20}$$

We know that $\lambda_s \leq 0$ and therefore $\mathbf{q}_0 = \mathbf{q}_m$ (see Appendix A). In the degenerate case where $\lambda_s = 0$, $\mathbf{q}_l = \mathbf{q}_m = [0,0]^{\mathrm{T}}$.

It will also sometimes make sense to specify that the *flux* is zero – we proceed with a full Riemann solution with $\mathbf{q}_l = \mathbf{q}_r$.

3.4. Visual Representation of Vehicles

We use a discrete, particle like representation of vehicles for graphical rendering in our system, called "carticles". These primarily serve to provide a visual representation of traffic, but also play a role in deciding when to begin lane changes (see Sec. 3.5). Each lane i has an associated set of carticles $C_i = \{c_0, c_1, c_2 \ldots\}$; each carticle in turn has a minimal amount of state associated with it:

Position — the parametric position $s \in [0,1]$ of the rear axle of the carticle along the lane.

Lane change state — an enumerant that signifies that the carticle is changing lanes and if so, which direction (left or right) the change is in.

Lane change progress — a scalar $s_m \in [0,1]$ representing the progress of the carticle's current lane change — 0 signifies that the carticle has not begun to turn while 1 represents a carticle that has completed its lane change.

Vehicle type — an enumerant representing the type of vehicle of the carticle. Currently, our simulation technique only uses information about the length of each vehicle type and position of the rear axle relative to the overall length.

3.4.1. Specifying Initial and Boundary Conditions

Carticles may also be used to give intuitive initial and boundary conditions for a simulation. While we are welcome to explicitly specify the ρ and y along each lane for $t = 0$, it is often more useful and natural to place discrete vehicles (with velocities) as initial conditions.

Similarly, while we are free to prescribe numerical fluxes at boundaries (see Sec. 3.3.8), it is sometimes more convenient to specify discrete arrival times and velocities at inflow boundaries. This is accomplished by simply converting discrete carticles into an equivalent continuum representation; see Section 3.4.3.

3.4.2. Carticle Motion

The parametric position s_j of each carticle j in the system is advanced at each simulation step via the simple ODE:

$$s_j'(t) = \frac{u_{\text{lane}}(s_j(t), t)}{L_{\text{lane}}} \tag{21}$$

Here $u_{\text{lane}}(s_j(t), t)$ represents the velocity field of the lane to which carticle j belongs (defined in Eq. (8)) and L_{lane} the length of said lane. Since the evaluation of the right-hand side of Eq. (21) consists of inexpensive interpolation of discrete data, we use explicit 4^{th} order Runge-Kutta to integrate each carticle's position. When s_j is greater than 1 — and thus has traveled beyond the end of the lane — we either remove the carticle from the simulation altogether (in the case of an external boundary condition) or "pass" it to the next lane and adjust s_j appropriately.

3.4.3. Carticles at the Continuum Level

Carticles represent the positions of vehicles in our method, but in order to have the underlying continuum simulation reflect the position of these vehicles, we must "seed" the discrete cells along each lane with the appropriate density and velocities of each carticle. This happens at the beginning of a simulation, where we must account for any vehicles that are initially given in the network, and also at network inflow boundaries when incoming vehicles are specified discretely.

When all Δx_j (see Sec. 3.3.1.2) are greater than the length of any vehicle type, we can be certain that the interval associated with each carticle overlaps no more than 2 grid cells. We interpret the quantity ρ stored at each grid cell as "cars per car length" (as described in Sec. 3.3.3); thus, for each cell i a carticle j with velocity u_j overlaps, we compute the updated density (ρ_i') and velocity (u_i') at i from their original values $[\rho_i, u_i]^{\mathrm{T}}$ and the contribution of j:

$$\Delta\rho_i = \frac{o_{i,j}}{\Delta x_{\text{lane}}}$$

$$\rho_i' = \rho_i + \Delta\rho_i \tag{22}$$

$$u_i' = \frac{\rho_i u_i + \Delta\rho_i u_j}{\rho_i'} \tag{23}$$

Here $o_{i,j}$ is the length (in real, not parametric space) that the carticle j overlaps cell i.

3.5. Lane Changes and Merges

We handle the movement of vehicles from one lane to another (interchangeably called a lane change or a merge) using a combination of information from carticles and the density/flow data from the continuum model.

Our method arises from the following observations:

- A lane change generally takes place on a longer timescale than a single simulation step.
- Once a vehicle begins a lane change, it continues to move into the other lane until the lane change is finished.

Based on these observations, we initiate lane changes on a per-carticle basis. When certain conditions we describe below are met, a carticle will be marked as being in a either a left or right lane change and the simulation will account for the lateral movement of that carticle and the underlying flux between lanes.

Starting a lane change We initiate lane changes based on some simple rules that are used to compute a signed *merge factor* that ultimately determines if a vehicle will change lanes:

- At a vehicle's position along a lane, there are as many as two adjacent lanes to move to, but we require that the adjacency on each side continue for a long enough distance forward to make the lane change possible, given the vehicle's current velocity.
- We also require that there be no vehicles in the potential path of the lane change; we look for carticles in the cells in the immediate vicinity of the cell that neighbors the lane-change candidate. If any vehicles are present with trajectories would overlap the potential lane change path in the next timestep, we remove the path from consideration.
- Finally, if there is at least one suitable adjacent lane, we determine the desirability of changing lanes — the aforementioned merge factor.

Real drivers change lanes to ensure a certain path is taken (i.e. to be able to make certain turns or take exit ramps), to move into faster traffic, and for a variety of other reasons. Our system accounts for two such motivations:

Routing distributions Each adjacency between two lanes has a distribution that represents the propensity of traffic to make a lane change there (see Sec. 3.2). If we determine that a lane change is possible according to the aforementioned criteria, then we query this distribution to see if the vehicle should initiate this lane change. This feature allows our simulator to model common offramps that experience heavy volume of traffic; for most adjacencies, such as that between two lanes on a multi-lane highway, the corresponding distribution would be zero.

Overtaking slow traffic The second variety of lane change we model is based on perceived increase in attainable velocity; if the traffic ahead of a vehicle is moving much more slowly than the traffic in a neighboring lane, change to that faster lane is attractive.

The following formula is applied to each adjacent lane $k \in \{l, r\}$ to determine the merge factor m_k:

$$m_k = \frac{u_{\text{adj}_k}}{u_{\text{ahd}}} \tag{24}$$

Here u_{adj_k} is the continuum velocity in the cell in the adjacent lane k that neighbors the candidate vehicle and u_{ahd} the velocity in the cell ahead of the candidate vehicle.

If there is more than one candidate lane being considered, we pick the lane with the largest merge factor. Now if this ultimate merge factor exceeds a threshold — we found the value 1.1 worked well in our experiments — we initiate a lane change.

3.5.1. Lane Changes at the Continuum Level

As we have discussed, lane changes are initiated at the carticle level and must be carried to completion. Furthermore, a lane change will generally require several simulation steps to perform. We must account for the transition of the vehicle at the (continuum) dynamics level to properly reflect effects of the lane change, but a straightforward transfer of the density and velocity corresponding to the vehicle over the course of the lane change is not sufficient.

A vehicle performing a lane change effectively occupies a space in *both lanes simultaneously* — its motion dictates the behavior of traffic behind it in both the lane it is leaving and the one it is entering. For this reason, once a vehicle begins to switch lanes, we *duplicate* its density and velocity information in the adjacent cell and proceed with the simulation until the lane change has completed, at which point the density and velocity representing the vehicle in the lane it left is removed. This violates conservation for a brief period of time, but the ultimate result is a superior description of the effects of a lane change and density after the lane change is the same as it was beforehand. We describe how we convert carticles to their corresponding continuum-level information in Sec. 3.4.

4. Results

4.1. Examples

We have tested our technique on a number of synthetic road networks, and on a large 'clover-leaf' interchange; see Figs. 1 and 3 for visual depiction. Fig. 1 shows an overview of a 'cloverleaf' freeway interchange depicted in our companion video. Figs. 3(a) and 3(b) show scenes from traffic traveling along a freeway, and Fig. 3(c) shows traffic flowing near an offramp on a highway system. These scenes are small windows into a road network filled with two to three thousand vehicles at any given time.

4.2. Comparison with agent-based simulation

To better understand the performance of our technique as compared to a microscopic, agent-based simulation method, we have timed simulations using our technique and a popular, state-of-the-art traffic simulator for a variety of scenarios. The Simulation of Urban MObility (SUMO) project [sum09] is an open-source traffic simulation package originating from the Centre for Applied Informatics at the University of Cologne and the Institute of Transport Research at the German Aerospace Centre. SUMO is based on a microscopic car-following model of traffic flow, and we would therefore expect its performance to be linear in the number of vehicles in the simulation.

The road network description format for each simulator varies greatly, so chose a simple simulation network as the basis for comparison to ensure that our simulator and SUMO would be operating on precisely the same input. The network is a 6-lane straight stretch of freeway 10km long. We provided input data to each simulation as series of vehicles entering the network; each scenario ran for a constant period of time but varied in the number of vehicles emitted over the interval.

The result of this simulation is shown in Fig. 4. Note that

(a) A freeway in a city (b) Traffic at an interchange (c) Vehicles exiting via offramp

Figure 3: *Images from our simulator*

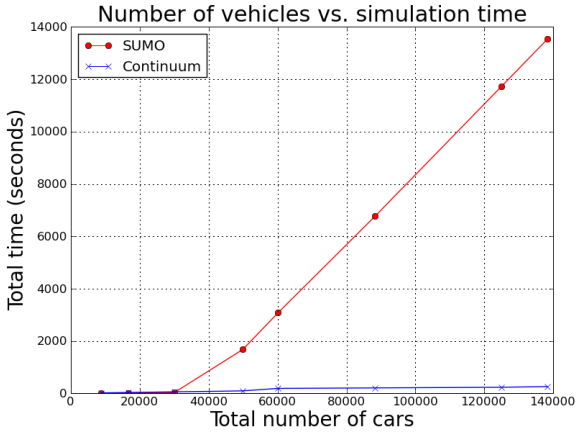

Figure 4: *Comparison on performance scaling of agent-based SUMO in red curve (top) vs. our simulator in blue line (bottom) as the number of cars increases.*

as there is no parallel implementation of SUMO available, we performed our timings for both simulators on a single processor.

We observe nearly linear performance in the number of cars for SUMO in scenarios with a small number of cars, but a dramatic drop in performance as the number of cars increases. In contrast, our simulator maintains a nearly linear performance over all ranges of inputs, but has a larger constant overhead than SUMO for a small number of cars. The qualitative results here are roughly what we would expect: SUMO should have some cost per vehicle being simulated, resulting in performance linear in the number of vehicles in the simulator, while our technique has a constant cost associated with the total network size, regardless of the number of vehicles, as well as a much smaller added cost per vehicle in the network.

4.3. Scaling of parallel implementation

Like many numeric techniques based on hyperbolic equations, our technique is very amenable to parallel computation. The majority of the computation required for our simulation involves solving the Riemann problem at each cell, and this can be done independently for each interface between cells. In practice, it makes sense to divide the work into coarser tasks involving multiple lanes. Detailed discussion of how our technique has been parallelized can be found in Appendix B; a naïve parallelization has produced sub-linear but encouraging parallelism — approximately 5x on an 8 core machine based on Intel i7 architecture.

5. Conclusion

We have presented a method for the generation of realistic traffic animations. Our method is augments a continuum model with a facility to extract discrete results. We have reported preliminary results on parallelization and demonstrated our method's ability to generate traffic on large, real-world road networks.

5.1. Limitations and Future work

Our approach is mainly limited in the scope of traffic-related phenomena it can handle. Vehicle collisions, road conditions, and weather, and the distinctions of the vast array of vehicle types and driver types are not currently considered in our prototype system.

The technique we have described is also not suitable for simulating traffic where specific routes are to be followed by each car; to take advantages of the computational efficiency offered by continuum methods, we cannot route each vehicle individually. We hope to extend our method to allow for a subset of vehicles that can follow specific paths in a network while allowing the other traffic to react and flow naturally.

Our technique is limited to networks of highway-class roads at this point. It is desirable to simulate urban, street-level traffic with detailed intersections. We are exploring how our method may be augmented to describe traffic on a wider range of road types.

Many of the aforementioned limitations are derived from the aggregate nature of the continuum approach, we are investigating the idea of a coupled continuum-discrete traffic

simulation to take advantage of the strengths of each: continuum models are fast and can handle large areas inexpensively, while discrete models are capable of describing more individualistic behavior.

Acknowledgments

The authors would like to thank Avneesh Sud at Microsoft and the members of the GAMMA group at UNC. This research is supported in part by the Army Research Office, Intel Corporation, National Science Foundation, and RDECOM.

References

[ABB*97] ALGERS S., BERNAUER E., BOERO M., BREHERET L., TARANTO C. D., DOUGHERTY M., FOX K., GABARD J. F.: Smartest project: Review of micro-simulation models. *EU project No: RO-97-SC 1059* (1997).

[AR00] AW A., RASCLE M.: Resurrection of "second order" models of traffic flow. *SIAM Journal of Applied Mathmatics*, 60 (2000), 916–938.

[BdLCW82] BYRNE A., DE LASKI A., COURAGE K., WALLACE C.: *Handbook of computer models for traffic operations analysis*. Tech. Rep. FHWA-TS-82-213, Washington, D.C., 1982.

[CSS00] CHOWDHURY D., SANTEN L., SCHADSCHNEIDER A.: Statistical Physics of Vehicular Traffic and Some Related Systems. *Physics Reports 329* (2000), 199.

[Dag95] DAGANZO C. F.: Requiem for second-order fluid approximations of traffic flow. *Transportation Research B*, 29 (1995), 277–286.

[Ger55] GERLOUGH D. L.: *Simulation of freeway traffic on a general-purpose discrete variable computer*. PhD thesis, UCLA, 1955.

[GVK05] GO J., VU T., KUFFNER J.: Autonomous behaviors for interactive vehicle animations. In *International Journal of Graphical Models* (2005).

[Lax72] LAX P.: *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shockwaves*. No. 11 in SIAM Regional Conference Series in Applied Mathematics. SIAM, 1972.

[LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum 23*, 3 (2004), 509–518.

[Lev02] LEVEQUE R. J.: *Finite Volume Methods for Hyperbolic Problems*. Cambgridge University Press, New York, New York, 2002.

[LHSM05] LEBACQUE J.-P., HAJ-SALEM H., MAMMAR S.: Second order traffic flow modeling: supply-demand analysis of the inhomogeneous riemann problem and of boundary conditions. In *10th EURO Working Group Transportation Meeting* (Poznan, Poland, September 2005).

[LMHS07] LEBACQUE J.-P., MAMMAR S., HAJ-SALEM H.: The aw-rascle and zhang's model: Vacuum problems, existence and regularity of the solutions of the riemann problem. *Transportation Research Part B*, 41 (2007), 710–721.

[LW55] LIGHTHILL M. J., WHITHAM G. B.: On kinematic waves. ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London A229*, 1178 (May 1955), 317–345.

[NBS97] NELSON P., BUI D., SOPASAKIS A.: A novel traffic stream model deriving from a bimodal kinetic equilibrium. In *Proceedings of the 1997 IFAC meeting, Chania, Greece* (1997), pp. 799–804.

[New61] NEWELL G.: Nonlinear effects in the dynamics of car following. *Operations Research 9*, 2 (1961), 209–229.

[NS92] NAGEL K., SCHRECKENBERG M.: A cellular automaton model for freeway traffic. *Journal de Physique I 2*, 12 (December 1992), 2221–2229.

[PA60] PRIGOGINE I., ANDREWS F. C.: A Boltzmann like approach for traffic flow. *Operations Research 8*, 789 (1960).

[PAB08] PELECHANO N., ALLBECK J. M., BADLER N. I.: *Virtual Crowds: Methods, Simulation and Control*. Morgan and Claypool Publishers, 2008.

[Pay71] PAYNE H. J.: Models of freeway traffic and control. *Mathematical Models of Public Systems 1* (1971), 51–60. Part of the Simulation Councils Proceeding Series.

[PKL08] PETTRÉ J., KALLMANN M., LIN M. C.: Motion planning and autonomy for virtual humans. In *ACM SIGGRAPH 2008 classes* (2008), pp. 1–31.

[PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum 26*, 3 (2007), 665–674.

[Ric56] RICHARDS P. I.: Shock waves on the highway. *Operations Research 4*, 1 (1956), 42–51.

[Sew10] SEWALL J.: *Efficient, Scalable Traffic and Compressible Fluid Simulations from Hyperbolic Models*. PhD thesis, University of North Carolina at Chapel Hill, April 2010.

[SGTL09] SEWALL J., GALOPPO N., TSANKOV G., LIN M. C.: Visual simulation of shockwaves. *Graphical Models 71*, 4 (July 2009), 126–138.

[SH99] SHVETSOV V., HELBING D.: Macroscopic dynamics of multilane traffic. *Physical Review E 59*, 6 (1999), 6328–6339.

[sum09] Sumo - simulation of urban mobility, October 2009. http://sumo.sourceforge.net/index.shtml.

[TH01] TREIBER M., HELBING D.: Microsimulations of freeway traffic including control measures. *Automatisierungstechnik*, 49 (2001), 478–484.

[vdBSLM09] VAN DEN BERG J., SEWALL J., LIN M., MANOCHA D.: Virtualized traffic: Reconstructing traffic flows from discrete spatio-temporal data. In *IEEE Virtual Reality* (2009).

[Whi74] WHITHAM G. B.: *Linear and nonlinear waves*. John Wiley and Sons, New York, New York, 1974.

[YK96] YANG Q., KOUTSOPOULOS H.: A Microscopic Traffic Simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C 4*, 3 (1996), 113–129.

[Zha02] ZHANG H. M.: A non-equilibrium traffic model deviod of gas-like behavior. *Transportation Research B*, 36 (2002), 275–290.

Appendix A: The Aw-Rascle-Zhang System of Equations

Field Classification

Following Lax [Lax72], if the quantity

$$\nabla\lambda_i \cdot \mathbf{r}_i \tag{25}$$

is always nonzero, the i-family is said to be *genuinely nonlinear*; that is to say, we expect nonlinear phenomena such as shocks and rarefaction waves to appear. Conversely, if Eq. (25) is zero for the i-family, that field is termed *linearly degenerate* and we expect solutions of this family to propagate as a constant jump in one or more fields of $\mathbf{q}$. For the 0-family of the ARZ system:

$$\nabla\lambda_0 \cdot \mathbf{r}_0 = 2u'_{\text{eq}} + \rho u''_{\text{eq}} \tag{26}$$

So long as $2u'_{\text{eq}} + \rho u''_{\text{eq}} \neq 0$, the 0-family of solutions is genuinely nonlinear and the associated solutions will contain shocks and rarefaction waves. For the 1-family:

$$\nabla\lambda_1 \cdot \mathbf{r}_1 = 0 \tag{27}$$

Thus the 1-family of solutions is linearly degenerate and solutions associated with this family will propagate as constant discontinuities.

Riemann Invariants In the same spirit of the field classifications discussed above, we can determine what quantities are preserved across each solution family; we know that $\mathbf{q}_l$ and $\mathbf{q}_r$ are connected through an unknown intermediate state $\mathbf{q}_m$ and these invariants help us determine this intermediate state. A quantity ω_i is a *Riemann invariant* for the i-family of solutions if it satisfies the following equation:

$$\nabla\omega_i \cdot \mathbf{r}_i = 0 \tag{28}$$

The invariant for the first wave (corresponding to λ_0 and $\mathbf{r}_0$) can be computed by substituting Eq. (11) into Eq. (28):

$$\omega_0 = \frac{y}{\rho} = u - u_{\text{eq}} \tag{29}$$

The 1-family is linearly degenerate, so Eq. (29) is clearly satisfied by Eq. (27):

$$\omega_1 = \lambda_1 = u \tag{30}$$

The 0-family invariant looks similar to (6); intuitively, we can say that this 'equilibrium velocity' is conserved across waves from the first family. The 1-family invariant simply states that the velocity u does not change across waves of the second family.

Classification of solutions

For $\mathbf{q}_m$ to exist, the speeds of the system (Eqs. (9) and (10)) must be distinct. This occurs when $\rho_l > 0$; in this case, there are three distinct regions of the solution: q_l for $x \leq \lambda_0 t$, q_m for $\lambda_0 < \frac{x}{t} < \lambda_1$, and q_r for $x \geq \lambda_1 t$. In the case where $\rho_l = 0$, $\lambda_0 = \lambda_1$ and q_m vanishes. We shall deal with cases where $\rho_l = 0$ or $\rho_r = 0$ separately below.

We know from Sec. A that the 0-family solutions are always shock or rarefaction waves. To determine which, we must consider the 0-family speeds λ_{0l} and λ_{0m} on either side of the nonlinear wave:

$$\lambda_{0l} = u_l - u_{\max}\gamma\rho_l^{\gamma} \tag{31}$$

$$\lambda_{0m} = u_r - u_{\max}\gamma\rho_l^{\gamma} + \gamma(u_r - u_l) \tag{32}$$

When $\lambda_{0l} < \lambda_{0m}$, the solution is a rarefaction, and when $\lambda_{0l} > \lambda_{0m}$, the solution is a shock. From Eqs. (31) and (32), we can determine that

$$\lambda_{0l} > \lambda_{0m} \Leftrightarrow u_l > u_m \tag{33}$$

Case 0 $u_r = u_l$

This is a degenerate case, but bears consideration since the assumptions in the remaining cases are based on strict inequalities. In this case, Eq. (12) predicts $\rho_m = \rho_l$, so $q_m = q_l$. Clearly, there is no wave in the 0-family; we are left with the usual contact discontinuity in the 1-family.

q_o **for case 0** Since $q_m = q_l$,

$$q_o = q_l \tag{34}$$

regardless of the sign of λ_{0l}.

Case 1 $u_r < u_l$

As per Eq. (33), the characteristics in the 0-family are colliding and a shock appears. The shock represents in increase in density (because $\rho_m > \rho_l$ and a decrease in velocity (because $u_r < u_l$). The 1-family has a contact discontinuity in ρ as always. The shock's speed is given by

$$\lambda_s = \frac{\rho_m u_m - \rho_l u_l}{\rho_m - \rho_l} \tag{35}$$

Which is an application of the Rankine-Huginot condition for the first equation of our system; see Leveque [Lev02] for information on this condition.

q_o **for case 1** Following the above discussion, q_0 for case 1 depends on the sign of Eq. (35):

$$q_0 = \begin{cases} q_l & \lambda_s \geq 0, \\ q_m & \lambda_s < 0 \end{cases} \tag{36}$$

Case 2 $u_r - u_{\max}\rho_l^{\gamma} < u_l < u_r$

Since $u_l < u_r$, Eq. (33) predicts a rarefaction wave in the 0-family. The 1-family is a contact discontinuity, as always. The intermediate density ρ_m is less than ρ_l.

Centered rarefaction waves Rarefactions represent regions of smooth variation and cannot be described with a single speed. We can use λ_{ol} and λ_{om} (see Eqs. (14) and (15)) to determine if the rarefaction is to the left or right of q_o (in which case q_0 is q_m or q_l, respectively) or if q_o is inside the rarefaction. If this is the case, we have what is termed a *centered* or *transonic* rarefaction; we

must do some extra work to determine what the structure of the rarefaction is and specifically evaluate it at $x = 0$. For the ARZ system of equation, $\mathbf{q}_0$ in a centered rarefaction is given by the following:

$$\tilde{\rho}(0) = \left(\frac{u_l + u_{\max}\rho_l^\gamma}{(\gamma+1)\,u_{\max}} \right)^{\frac{1}{\gamma}} \tag{37}$$

$$\tilde{u}(0) = \frac{\gamma}{\gamma+1}\left(u_l + u_{\max}\rho_l^\gamma\right) \tag{38}$$

q_0 **for case 2** For case 2, q_0 depends on the signs of both Eq. (14) and Eq. (15):

$$q_0 = \begin{cases} q_l & \lambda_{0l} \geq 0 \\ q_m & \lambda_{0m} \leq 0 \\ \tilde{q}(0, q_l, q_m) & \lambda_{0l} < 0 \text{ and } \lambda_{0m} > 0 \end{cases} \tag{39}$$

Here $\tilde{q}$ corresponds to the centered rarefaction state described above.

Case 3 $u_l \leq u_r - u_{\max}\rho_l^\gamma$
Now Eq. (12) cannot be evaluated; the density of the intermediate state ρ_m is 0. To observe the Riemann invariants ω_0 and ω_1 from Sec. A, u_m is given by the following:

$$u_m = u_l + u_{\max}\rho_l^\gamma$$

and

$$\lambda_{0m} = u_l + u_{\max}\rho_l^\gamma$$

In this case, $\lambda_{0l} = u_l - u_{\max}\gamma\rho_l^\gamma < \lambda_{0m} = u_l + u_{\max}\rho_l^\gamma$; we should connect q_l to q_m through a rarefaction wave. Now there is a jump in both ρ and u to get to q_r; we can imagine this as a fictitious velocity wave where $u_m = u_l + u_{\max}\rho_l^\gamma$ jumps to u_r, followed by the usual 1-wave — a contact discontinuity in ρ from $\rho_m = 0$ to ρ_r.
The rarefaction wave discussed above will be centered if $\lambda_{0l} < 0$ (since $\lambda_{0m} > 0$). Regardless of q_m, the value of q_o depends on the sign of λ_{0l} (just as with case 2).

q_0 **for case 3** q_0 for this case is identical to that of case 2, except that we can eliminate the possibility of $q_0 = q_m$, since $\lambda_{0m} > 0$:

$$q_0 = \begin{cases} q_l & \lambda_{0l} \geq 0 \\ \tilde{q}(0, q_l, q_m) & \lambda_{0l} < 0 \end{cases} \tag{40}$$

We wish to properly handle the absence of traffic (i.e. when ρ_l or ρ_r are 0). In fact, since u is not well-defined for $\rho = 0$ (see Eq. (8)), there are only two cases for problems involving "vacuum" states, and they can be treated as sub-cases of case 3:

Case 4 $\rho_l = 0$, $\rho_r > 0$ Clearly, the absence of traffic on the left should have no effect the traffic on the right (drivers in front will not change their behavior if there are no cars behind them); no q_m reasonably exists. There are no waves of the first family; we are left with the usual contact discontinuity in the second family.

q_o **for case 4**

$$q_0 = q_l = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{41}$$

Case 5 $\rho_l > 0$, $\rho_r = 0$ This case is treated the same way that we handle the first family of waves in case 3; we expect a rarefaction wave to appear. There is no jump in ρ, however, so no wave from the second family appears.

q_0 **for case 5** q_o for this case is identical to case 3 (see Eq. (40)).

Appendix B: Parallel Algorithm

One advantage of our approach is the ease with which it may made to take advantage of parallel hardware. In theory, the computation of each flux through the solution of the Riemann problem (i.e. Sec. 3.3.4) is independent. Likewise, the given these fluxes, the time integration of the unknowns (see Eq. (2)) is independent at each cell, and so too is the advection of each carticle as described in Sec. 3.4.

In practice, this is too fine a granularity to be useful, so we choose the handle the advancement of the solution along each lane as a task. We partition all lanes among the available processors and the various kernels we have mentioned above — the computation of fluxes, integration of the continuum equations, and the advection of the carticles — are all applied in parallel.

The amount of work performed in each lane is directly proportional to the number of cells in that lane, and thus the length (eq. (3) ensures that the number of cells in each lane is proportional to its length). To ensure that the workload assigned to each processor is roughly equal, we perform the partitioning of lanes in such a way as to have the number of cells assigned to each processor be as even as possible. Since the number and lengths of lanes can be safely be assumed to be constant throughout the simulation, we can compute this as a static partition.

The problem of jobs of varying length among multiple processors so as to minimize the total processing time is known as the *makespan*. This is an NP-complete problem that is closely related to the more familiar *bin-packing* problem. While is has been proven that there are no fully polynomial approximation algorithms for these techniques, there are $(1+\varepsilon)$ polynomial approximation algorithms that are suitable for the static problem we wish to solve.

The parallelization scheme outlined above works well for computing flow along lane. The amount of communication between lanes is a constant amount of data — the computation of fluxes at boundaries only requires the last grid cell of the incoming lane and the first grid cell of the outgoing one. Very little memory is shared and the effect of the computer's memory hierarchy is limited, and this is the scheme that we currently use in our implementation.

In the presence of lane changes, where lanes may be adjacent for a period proportional to their length, lanes may have greater communication needs. Our makespan-based partitioning scheme fails to account for adjacencies and could suffer from latency problems due to increased bandwidth and memory hierarchy issues, and the task of producing a partitioning that considers adjacency information in addition to lane length to achieve maximum throughput is a promising problem we hope to explore in future work.