R3D2 Documentation

Release 0.1

Alice Harpole and Ian Hawke

Nov 29, 2018

Contents

R3D2 (Relativistic Reactive Riemann problem solver for Deflagrations and Detonations) solves the Riemann problem for the *relativistic* Euler equation. It also includes the option to include reaction terms, for “infinitely” fast reactions, leading to deflagrations and detonations.

Models of ideal hydrodynamics, where there is no viscosity or dissipation, can have solutions with *discontinuities* such as shocks. A simple case is the Riemann Problem, where two constant states are separated by a barrier. After the barrier is removed the solution develops, with *waves* (such as shocks and rarefactions) separating constant states.

The Riemann Problem has three main uses. Efficient, often approximate, solvers are an integral part of many modern hydrodynamic evolution codes. Second, the exact solution is a standard test for such codes. Finally, the solver can illustrate features of discontinuous solutions in more complex scenarios.

This code is intended for exploring possible solutions and relativistic effects, or for comparing against a compressible code with reactive sources. It is optimized for use with Jupyter notebooks. It is **not** intended for use within a HRSC evolution code: the performance is far too poor, and the assumptions made to extreme.

CHAPTER 1

Installation

A standard:

```
python setup.py install
```

or:

```
pip install r3d2
```

should work.

CHAPTER 2

Usage

Import the equations of state, State class, and Riemann Problem class:

```
>>> from r3d2 import eos_defns, State, RiemannProblem
```

Set up an equation of state:

```
>>> eos = eos_defns.eos_gamma_law(5.0/3.0)
```

Set up the left and right states:

```
>>> U_L = State(rho=1.0, v=0.0, vt=0.0, eps=1.5, eos=eos)
>>> U_R = State(rho=0.125, v=0.0, vt=0.0, eps=1.2, eos=eos)
```

Solve the Riemann Problem:

```
>>> rp = RiemannProblem(U_L, U_R)
```

The output can be examined for details of the solution and its wave structure. For example, the three waves are each built of wave *sections*, which can be examined to check their type, via e.g.

```
>>> rp.waves[0].wave_sections
```

and its speed (or the range of speeds) can be examined via

```
>>> rp.waves[0].wavespeed
```

The states that the waves separate can be found via, e.g.,

```
>>> rp.waves[0].q_r
```

and the detailed values via

```
>>> rp.waves[0].q_r.state()
```

However, the classes are optimized for display in a Jupyter notebook. See the documentation for more detail.

CHAPTER 3

Contents

3.1 Riemann Problems

The code solves Riemann Problems for the relativistic Euler equations

$$\partial_t \begin{pmatrix} D \\ S_x \\ S_t \\ \tau \end{pmatrix} + \partial_x \begin{pmatrix} S_x \\ S_x v_x + p \\ S_t v_x \\ (\tau + p) v_x \end{pmatrix} = 0.$$

For further details on this system of equations, see the Living Review of Martí and Müller, particularly section 3.1 for the equations and section 8.5 for the solution of the Riemann Problem.

The initial data is piecewise constant: two states $w_{L,R}$ are specified, each in terms of $w = (\rho_0, v_x, v_t, \epsilon)$, (the specific rest mass density, normal (x) and tangential (t) velocity components, and the specific internal energy). At $t = 0$ the data is set by w_L for $x < 0$ and w_R for $x > 0$. Each state has associated with it an equation of state (EOS) to close the set of equations: the EOS does not need to be the same for each state.

3.1.1 Code

To set up a Riemann problem, first set up a left and right state. Each state has its own equation of state. Here we use the first test from the Test Bench section of the Living Review:

```
In [1]: from r3d2 import eos_defns, State, RiemannProblem
```

```
In [2]: eos = eos_defns.eos_gamma_law(5.0/3.0)
        test_1_U_left = State(10.0, 0.0, 0.0, 2.0, eos, label="L")
        test_1_U_right = State(1.0, 0.0, 0.0, 1.5e-6, eos, label="R")
        test_1_rp = RiemannProblem(test_1_U_left, test_1_U_right)
```

The Riemann Problem will produce output directly in the notebook:

```
In [3]: test_1_rp
```

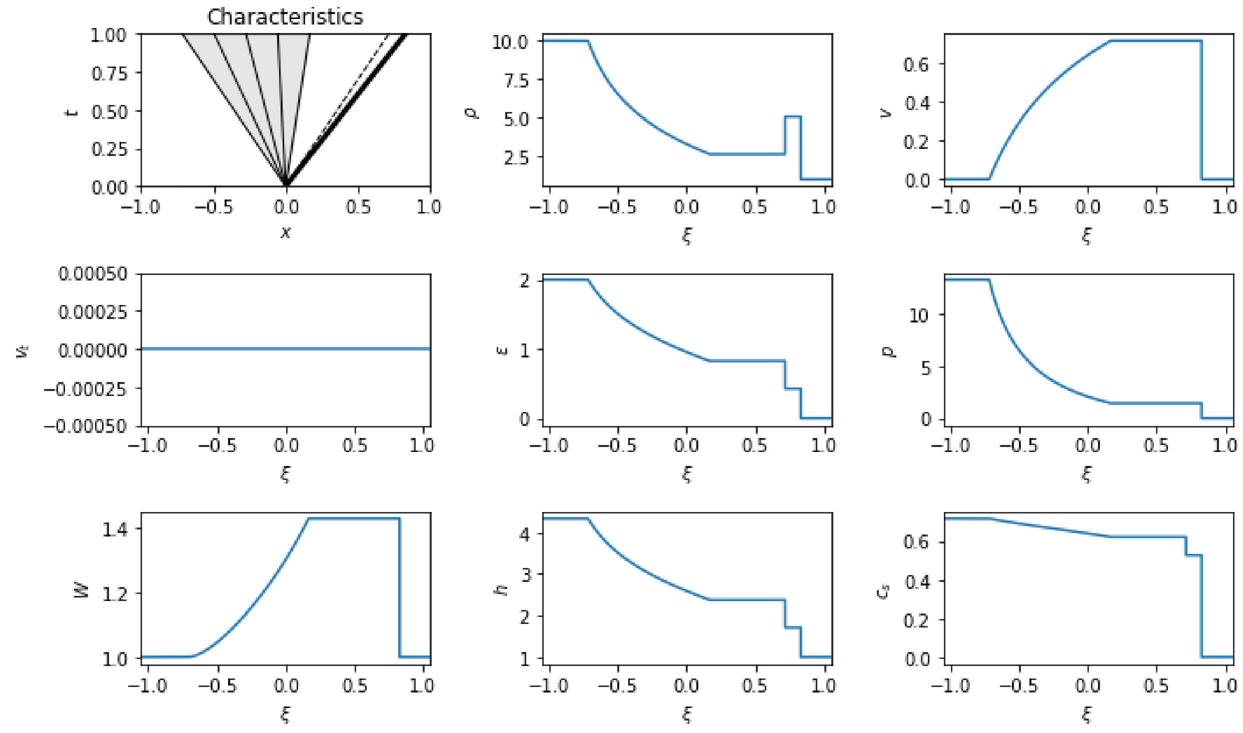
$$\begin{cases}\begin{pmatrix}\rho\\ v_x\\ v_t\\ \epsilon\end{pmatrix}_L = \begin{pmatrix}10.0000\\ 0.0000\\ 0.0000\\ 2.0000\end{pmatrix}, \\ \begin{pmatrix}\rho\\ v_x\\ v_t\\ \epsilon\end{pmatrix}_R = \begin{pmatrix}1.0000\\ 0.0000\\ 0.0000\\ 0.0000\end{pmatrix},\end{cases} \implies \mathcal{R}_{\leftarrow}\mathcal{C}\mathcal{S}_{\rightarrow}, \quad p_* = 1.4479, \quad \begin{cases}\mathcal{R}_{\leftarrow} : \lambda^{(0)} \in [-0.7161, 0.1672],\\ \mathcal{C} : \lambda^{(1)} = 0.7140,\\ \mathcal{S}_{\rightarrow} : \lambda^{(2)} = 0.8284,\end{cases} \begin{cases}\begin{pmatrix}\rho\\ v_x\\ v_t\\ \epsilon\end{pmatrix}_{\star_L} = \begin{pmatrix}2.639\\ 0.714\\ 0.000\\ 0.822\end{pmatrix}\\ \begin{pmatrix}\rho\\ v_x\\ v_t\\ \epsilon\end{pmatrix}_{\star_R} = \begin{pmatrix}5.070\\ 0.714\\ 0.000\\ 0.428\end{pmatrix}\end{cases} \tag{3.1}$$

This output should be interpreted as follows. First it displays the initial states $\mathbf{U}_{L,R}$. It then gives the resulting wave pattern: in this case a left going rarefaction, a contact, and a right going shock. It then gives the pressure p_* in the central states (the pressure and normal velocity do not jump across the contact). Next it gives the speeds of each wave. Finally, it gives the constant states between the left wave and the contact $\mathbf{U}_{*_L}$ and between the contact and the right state $\mathbf{U}_{*_R}$.

We can also display the output graphically in the notebook:

```
In [4]: from IPython.display import display_png
```

```
In [5]: display_png(test_1_rp)
```



The first (top left) plot is the wave pattern in the (x, t) plane. Rarefactions are shaded with thin solid lines. Contacts are dashed lines. Discontinuities such as shocks are thick solid lines.

All the other plots show a single quantity as a function of space at a fixed time, or equivalently as a function of the characteristic variable $\xi = x/t$.

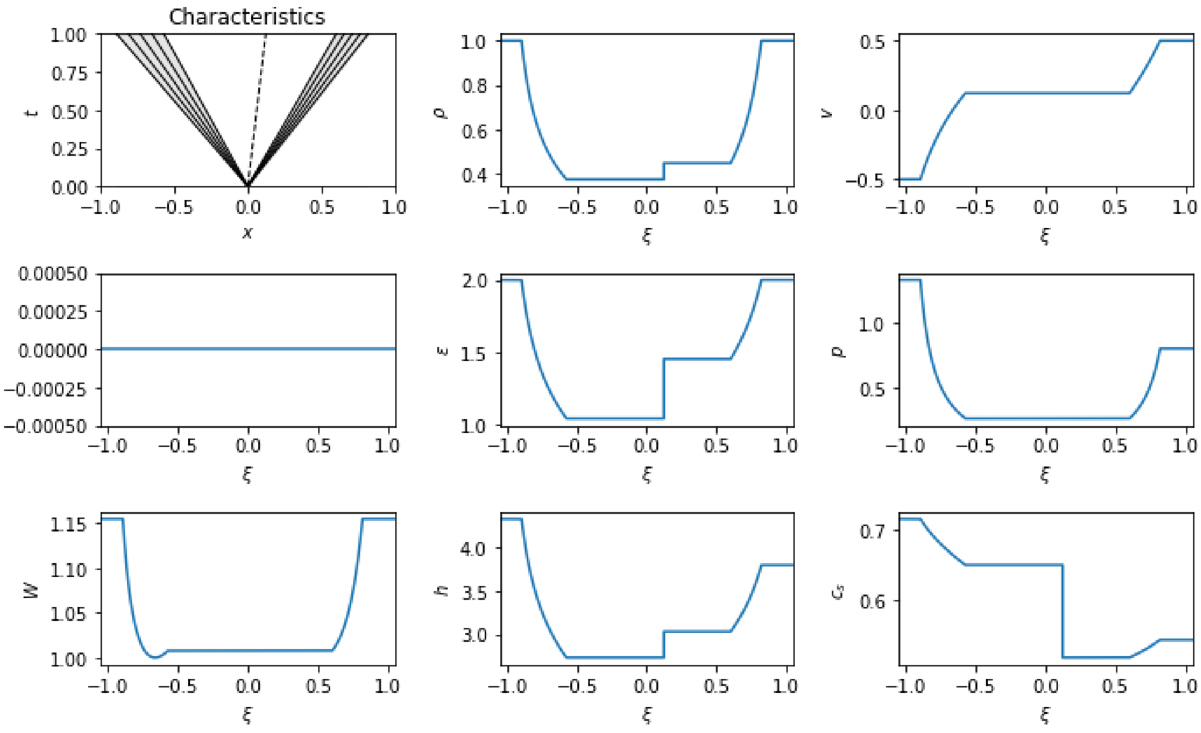
Changing equation of state

There is no need for the two states to have the same equation of state. For example:

```
In [6]: eos_air = eos_defns.eos_gamma_law(1.4)
        U_vary_eos_L = State(1.0, -0.5, 0.0, 2.0, eos, label="L")
        U_vary_eos_R = State(1.0, +0.5, 0.0, 2.0, eos_air, label="R")
        test_vary_eos_rp = RiemannProblem(U_vary_eos_L, U_vary_eos_R)
        test_vary_eos_rp
```

$$\begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_L = \begin{pmatrix} 1.0000 \\ -0.5000 \\ 0.0000 \\ 2.0000 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_R = \begin{pmatrix} 1.0000 \\ 0.5000 \\ 0.0000 \\ 2.0000 \end{pmatrix}, \end{cases} \implies \mathcal{R}_{\leftarrow}\mathcal{C}\mathcal{R}_{\rightarrow}, \quad p_* = 0.2598, \quad \begin{cases} \mathcal{R}_{\leftarrow} : \lambda^{(0)} \in [-0.8955, -0.5734], \\ \mathcal{C} : \lambda^{(1)} = 0.1224, \\ \mathcal{R}_{\rightarrow} : \lambda^{(2)} \in [0.6019, 0.8202], \end{cases} \begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star L} = \begin{pmatrix} 0.3 \\ 0.1 \\ 0.0 \\ 1.0 \end{pmatrix} \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star R} = \begin{pmatrix} 0.4 \\ 0.1 \\ 0.0 \\ 1.4 \end{pmatrix} \end{cases} \tag{3.2}$$

```
In [7]: display_png(test_vary_eos_rp)
```



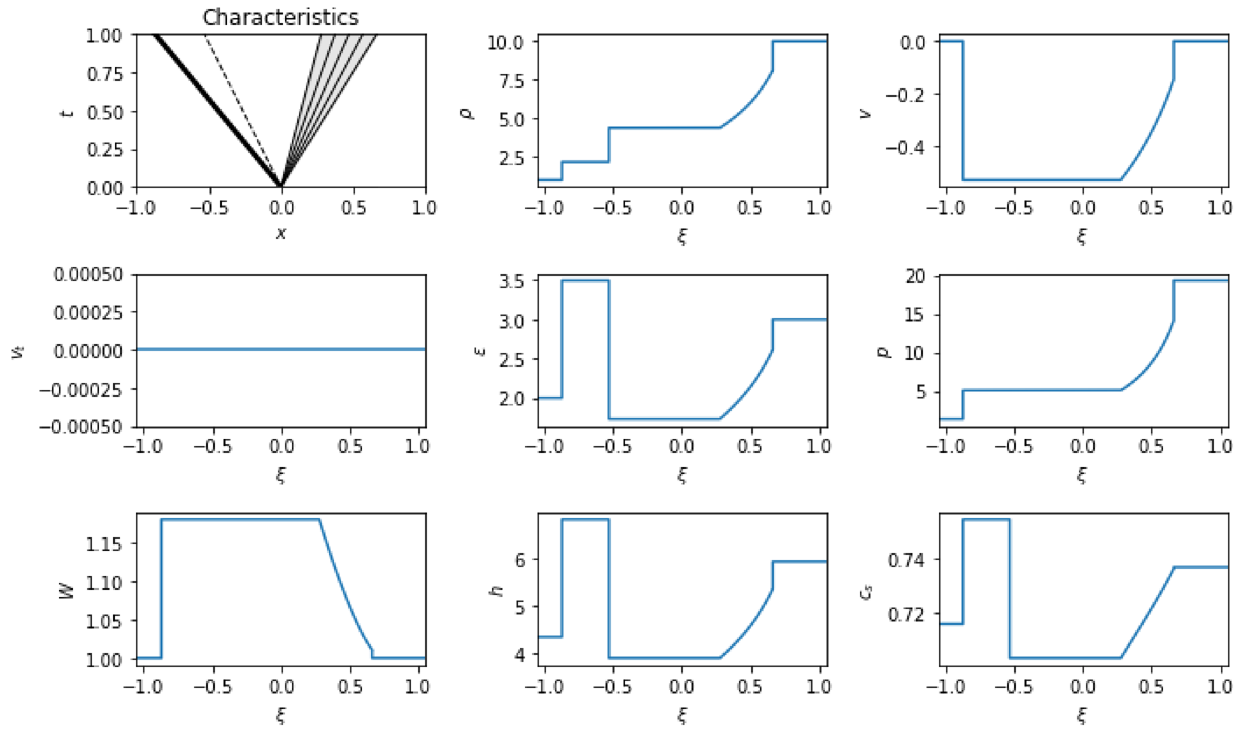
Reactive problems

To include a reaction, make the equation of state of at least one of the states be a reactive EOS:

```
In [8]: q_unburnt = 0.1
        gamma = 5/3
        Cv = 1.0
        t_ignition = 2
        eos_burnt = eos_defns.eos_gamma_law(gamma)
        eos_unburnt = eos_defns.eos_gamma_law_react(gamma, q_unburnt, Cv, t_ignition, eos_burnt)
        U_reactive_left = State(1, 0, 0, 2, eos_burnt)
        U_reactive_right = State(10, 0, 0, 3, eos_unburnt)
        test_reactive_rp = RiemannProblem(U_reactive_left, U_reactive_right)
        test_reactive_rp
```

$$\left\{\begin{array}{l} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 1.0000 \\ 0.0000 \\ 0.0000 \\ 2.0000 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \\ q \end{pmatrix} = \begin{pmatrix} 10.0000 \\ 0.0000 \\ 0.0000 \\ 3.0000 \\ 0.1000 \end{pmatrix}, \end{array}\right. \implies \mathcal{S}_{\leftarrow}\mathcal{C}\left(\mathcal{R}_{\rightarrow}\mathcal{CJDF}_{\rightarrow}\right), \quad p_* = 5.0558, \quad \begin{cases} \mathcal{S}_{\leftarrow} : \lambda^{(0)} = -0.8710, \\ \mathcal{C} : \lambda^{(1)} = -0.5305, \\ \left(\mathcal{R}_{\rightarrow}\mathcal{CJDF}_{\rightarrow}\right) : \lambda^{(2)} \in [0.2763, 0.6612], \end{cases} \tag{3.3}$$

```
In [9]: display_png(test_reactive_rp)
```



3.2 eos_defns

This is a set of pre-defined equations of state (EOS) for use with the Riemann Solver.

In general an equation of state means specifying the internal energy in terms of two thermodynamic variables, such as the volume and the temperature. All other quantities are derived from the Maxwell relations.

For our purposes, an EOS is a Python *dictionary* containing the essential relations needed.

3.2.1 Functions needed

1. `p_from_rho_eps(rho, eps)`. $p(\rho_0, \epsilon)$. Pressure given rest mass density and specific internal energy.

2. `h_from_rho_eps(rho, eps)`. $h(\rho_0, \epsilon)$. Specific enthalpy given rest mass density and specific internal energy.
3. `cs_from_rho_eps(rho, eps)`. $c_s(\rho_0, \epsilon)$. Speed of sound given rest mass density and specific internal energy.
4. `h_from_rho_p(rho, p)`. $h(\rho_0, p)$. Specific enthalpy given rest mass density and pressure.

3.2.2 Information needed for reactions

1. `q_available`. q. Amount of energy available for a reaction.
2. `t_ignition`. t_i. Temperature at which the reaction takes place.
3. `t_from_rho_eps(rho, eps)`. $T(\rho_0, \epsilon)$. Temperature given rest mass density and specific internal energy.
4. `eos_inert`. The equation of state to use after the reaction has taken place.

3.2.3 Provided EOS

1. `eos_gamma_law(gamma)`. Standard γ-law EOS where $e(V, T) = C_V T$ and so $p(\rho_0, \epsilon) = (\gamma - 1)\rho_0\epsilon$.
2. `eos_gamma_law_react(gamma, q, Cv, t_i, eos_inert)`. Reactive EOS where $e(V, T) = C_V T + q$ and so $p(\rho_0, \epsilon) = (\gamma - 1)\rho_0(\epsilon - q)$, where q is the chemical binding energy.

3.3 States

A Riemann Problem is specified by the state of the material to the left and right of the interface. In this hydrodynamic problem, the state is fully determined by an equation of state and the variables

$$\mathbf{U} = \begin{pmatrix} \rho_0 \\ v_x \\ v_t \\ \epsilon \end{pmatrix},$$

where ρ_0 is the rest-mass density, v_x the velocity normal to the interface, v_t the velocity tangential to the interface, and ϵ the specific internal energy.

3.3.1 Defining a state

In `r3d2` we define a state from an equation of state and the values of the key variables:

```
In [1]: from r3d2 import eos_defns, State
```

```
In [2]: eos = eos_defns.eos_gamma_law(5.0/3.0)
        U = State(1.0, 0.1, 0.0, 2.0, eos)
```

Inside the notebook, the state will automatically display the values of the key variables:

```
In [3]: U
```

$$\begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 1.0000 \\ 0.1000 \\ 0.0000 \\ 2.0000 \end{pmatrix} \tag{3.4}$$

Adding a label to the state for output purposes requires an extra keyword:

```
In [4]: U2 = State(10.0, -0.3, 0.1, 5.0, eos, label="L")
        U2
```

$$\begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_L = \begin{pmatrix} 10.0000 \\ -0.3000 \\ 0.1000 \\ 5.0000 \end{pmatrix} \tag{3.5}$$

3.3.2 Reactive states

If the state has energy available for reactions, that information is built into the equation of state. The definition of the equation of state changes: the definition of the state itself does not:

```
In [5]: q_available = 0.1
        t_ignition = 10.0
        Cv = 1.0
        eos_reactive = eos_defns.eos_gamma_law_react(5.0/3.0, q_available, Cv, t_ignition, eos)
        U_reactive = State(5.0, 0.1, 0.1, 2.0, eos_reactive, label="Reactive")
        U_reactive
```

$$\begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \\ q \end{pmatrix}_{Reactive} = \begin{pmatrix} 5.0000 \\ 0.1000 \\ 0.1000 \\ 2.0000 \\ 0.1000 \end{pmatrix} \tag{3.6}$$

3.3.3 Additional functions

A state knows its own wavespeeds. Given a wavenumber (the left acoustic wave is `0`, the middle contact or advective wave is `1`, and the right acoustic wave is `2`), we have:

```
In [6]: print("Left wavespeed of first state is {}".format(U.wavespeed(0)))
        print("Middle wavespeed of second state is {}".format(U2.wavespeed(1)))
        print("Right wavespeed of reactive state is {}".format(U.wavespeed(2)))
```

```
Left wavespeed of first state is -0.6636390537799616
Middle wavespeed of second state is -0.3
Right wavespeed of reactive state is 0.7615771981098584
```

A state will return the key *primitive* variables $(\rho, v_x, v_t, \epsilon)$:

```
In [7]: print("Primitive variables of first state are {}".format(U.prim()))
```

```
Primitive variables of first state are [1.   0.1 0.   2. ]
```

A state will return all the variables it computes, which is $\rho, v_x, v_t, \epsilon, p, W, h, c_s$: the primitive variables as above, the pressure p, Lorentz factor W, specific enthalpy h, and speed of sound c_s:

```
In [8]: print("All variables of second state are {}".format(U.state()))
```

```
All variables of second state are [1.         0.1        0.         2.         1.33333333 1.00503782
 4.33333333 0.71611487]
```

3.4 Waves

The solution to the Riemann problem is built out of *waves*. These are regions where the variables change, separating constant states. There are two basic types of wave: those across which the variables change continuously (which, for fluid problems, are known as *rarefactions*), and those across which the variables change discontinuously (which break down into more subclasses).

We will assume that the state is known on one side of the wave, which will be denoted $\mathbf{U}_k$. The state on the other side will be unknown. Given the value of one variable, typically the pressure p, on the unknown side, the solution across the wave can be found.

The equations here are adapted from the Living Review of Martí and Müller. The approach to reactive waves follows the paper of Zhang and Zheng, extended to the Relativistic case.

3.4.1 Wave sections

In general any wave can be built from multiple sections, pieced together. A wave may "split" into two discontinuous pieces moving at different speeds. A wave may contain a discontinuous section attached to a continuous section. In the most complex cases, the wave may have an arbitrarily large number of sections.

In the inert case, and when the equation of state is convex, each wave contains a single section. If the pressure decreases across the wave, $p_k > p$, then the wave is a continuous rarefaction. If the pressure increases across the wave, $p_k < p$, then the wave is a discontinuous shock. If the pressure does not change, the wave is either trivial (the state does not change) or a contact discontinuity (where quantities except p and v_x may jump).

In the reactive case, all the reactions that we can model will take place "instantly". This means that reactions must take place across discontinuous wave sections. The reactive waves may now contain multiple wave sections. If the equation of state is convex, and the reaction exothermic, then across a reactive discontinuous wave the pressure increases. Therefore if the pressure is to decrease across the wave as a whole, there must be an inert *precursor* shock before the reactive discontinuity.

Finally, we note that not all reactive discontinuous waves are stable. This can be checked by looking at the wave speed on the unknown side of the wave. If the wave speed leads to characteristics coming *out* of the discontinuity, then the wave is unstable. In this case we need to add another wave section. We find the fastest stable reactive discontinuity (a *Chapman-Jouget* discontinuity), where the wave speed matches the speed of the discontinuity, and attach to it an inert rarefaction.

3.4.2 Rarefactions

Across a rarefaction the normal velocity satisfies

$$\frac{\mathrm{d}v_x}{\mathrm{d}p} = \pm \frac{1}{\rho_0 h W^2 c_s} \frac{1}{\sqrt{1 + g\left(\xi_\pm, v_x, v_t\right)}}.$$

The sign corresponds to the wavenumber: plus for the right acoustic wave and minus for the left. In the code this corresponds to `lr_sign = wavenumber - 1`.

The function g quantifies the effect of the tangential velocity, and is

$$g = \frac{\left(v_t\right)^2 \left(\xi_\pm - 1\right)}{\left(1 - \xi_\pm v_x\right)^2}.$$

The wavespeed itself is $\xi_\pm$.

We also solve for the rest mass density and specific internal energy across the wave using

$$\frac{\mathrm{d}}{\mathrm{d}p} \begin{pmatrix} \rho_0 \\ \epsilon \end{pmatrix} = \begin{pmatrix} \frac{1}{h c_s^2} \\ \frac{p}{\rho_0^2 h c_s^2} \end{pmatrix}.$$

Having solved across the wave, the tangential velocity follows as

$$v_t = h_k W_k \left(v_t\right)_k \frac{1 - v_x^2}{h^2 + \left(h_k W_k \left(v_t\right)_k\right)^2}.$$

The relation for the tangential velocity holds across both continuous and discontinuous waves.

3.4.3 Shocks

Across an inert shock we first find the post shock density and enthalpy by solving the nonlinear equation

$$h^2 - h_k^2 - \left(\frac{h}{\rho_0} + \frac{h_k}{\left(\rho_0\right)_k}\right)\left(p - p_k\right) = 0.$$

This assumes the post shock pressure p_k is known, and that the equation of state gives the enthalpy $h_k = h\left(\left(\rho_0\right)_k, p_k\right)$.

From this we can compute the mass flux j across the shock, as

$$j = \sqrt{\frac{p_k - p}{\frac{h^2 - h_k^2}{p - p_k} - \frac{2h_k}{p_k}}}.$$

This gives the shock velocity as

$$v_S = \pm \frac{\left(\rho_0\right)_k^2 W_k^2 \left(v_x\right)_k \pm j^2 \sqrt{1 + \frac{\left(\rho_0\right)_k^2 W_k^2 \left(1 - \left(v_x\right)^2\right)}{j^2}}}{\left(\rho_0\right)_k^2 W_k^2 + j^2}.$$

Again, the sign corresponds to the wave number.

Given the shock velocity we can compute the shock "Lorentz factor" $W_S = (1 - v_S^2)^{-1/2}$, from which the post shock normal velocity is

$$v_x = \left(h_k W_k v_S \pm \frac{p - p_k}{j\sqrt{1 - v_S^2}}\right)\left[h_k W_k + \left(p - p_k\right)\left(\frac{1}{\left(\rho_0\right)_k W_k} \pm \frac{\left(v_x\right)_k W_S}{j}\right)\right]^{-1}.$$

The tangential velocity follows as in the rarefaction case.

3.4.4 Detonation

A detonation is a discontinuous reactive wave section across which the pressure increases. The equations are identical to those for the shock case, but the interpretation changes. All "known" (pre shock) variables have the reactive equation of state. All "unknown" (post shock) variables use the inert equation of state - the reaction has taken place across the discontinuity.

As noted above, it is possible for the resulting detonation wave section to be unstable. In general, detonations fall into two classes: *weak* detonations (which are unstable) and *strong* detonations (which are stable). For the stable strong detonations the characteristic waves impinge on the discontinuity from both sides. For the unstable weak detonations the characteristics only enter the discontinuity on one side.

If an unstable weak detonation is found, the section should be replaced by the fastest detonation that is stable. This Chapman-Jouget, or CJ detonation, is where the characteristic wave is parallel to the discontinuity. In this case, the post detonation pressure will not match the required post wave pressure, and an additional rarefaction wave section is needed to complete the wave.

3.4.5 Deflagration

A deflagration is a discontinuous wave section across which the pressure decreases. As a reaction cannot take place across a rarefaction wave, this requires a discontinuity. However, this discontinuity will reduce the temperature along with the pressure. This means that, unless the material was already at the right temperature to react, any reaction across this wave would be unphysical.

The solution is to start with an inert *precursor* shock which raises the temperature of the material to the ignition temperature. This first section follows the exact equations in the shock section above. Next, there is a deflagration wave section, across which the reaction takes place and the pressure drops. Again, this follows the shock equations, but with the same interpretation as in the detonation case.

Again, the deflagration wave section need not be stable. As with detonations, deflagrations fall into two classes: *weak* deflagrations (which are stable), and *strong* deflagrations (which are unstable). For the stable weak deflagrations the characteristic waves from one side impinge on the discontinuity, but not the other. For the unstable strong deflagrations, neither set of characteristic waves impinges on the discontinuity.

Again, if an unstable strong deflagration is found it is replaced with a CJ deflagration, and the full wave is completed with a rarefaction wave section.

3.4.6 Code

The `Wave` class in `r3d2` in intended for internal use, but can be used to construct single waves directly. We need a known state with an equation of state first:

```
In [1]: from r3d2 import eos_defns, State, wave
```

```
In [2]: eos = eos_defns.eos_gamma_law(5.0/3.0)
        U = State(1.0, 0.1, 0.05, 0.2, eos, label="known")
        U
```

$$\begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{known} = \begin{pmatrix} 1.0000 \\ 0.1000 \\ 0.0500 \\ 0.2000 \end{pmatrix} \tag{3.7}$$

We then solve for a left going (wavenumber `0`) inert acoustic wave with post-wave pressure of 10 or 0.1:

```
In [3]: wave_1 = wave.Wave(U, unknown_value=10, wavenumber=0)
        wave_2 = wave.Wave(U, unknown_value=0.1, wavenumber=0)
```

The first wave is a shock. When output in the notebook, it gives the type, wave number, and wave speed:

```
In [4]: wave_1
```

$$\mathcal{S}_{\leftarrow} : \lambda^{(0)} = -0.9614 \tag{3.8}$$

The second wave is a rarefaction. In this case, as it's a continuous wave, its wave speed is a range:

```
In [5]: wave_2
```

$$\mathcal{R}_{\leftarrow} : \lambda^{(0)} \in [-0.3209, -0.2383] \tag{3.9}$$

The right going (wavenumber `2`) acoustic waves follow the same structure:

```
In [6]: wave_right = wave.Wave(U, unknown_value=100, wavenumber=2)
        wave_right
```

$$\mathcal{S}_{\rightarrow} : \lambda^{(2)} = 0.9978 \tag{3.10}$$

The advective waves are different. The pressure must be constant across the wave, so if anything jumps it must be related to the state: either the thermodynamic variables or the equation of state:

```
In [7]: eos2 = eos_defns.eos_gamma_law(4.0/3.0)
        U2 = State(2.0, U.v, -0.3, U.p/2.0*3.0, eos2)
        wave_contact = wave.Wave(U, unknown_value=U2, wavenumber=1)
        wave_contact
```

$$\mathcal{C} : \lambda^{(1)} = 0.1000 \tag{3.11}$$

For a reactive wave, we first need a reactive equation of state:

```
In [8]: eos_reactive = eos_defns.eos_gamma_law_react(5.0/3.0, 0.1, 1.0, 1.0, eos)
        U_reactive = State(5.0, 0.0, 0.0, 2.0, eos_reactive)
        U_reactive
```

$$\begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \\ q \end{pmatrix} = \begin{pmatrix} 5.0000 \\ 0.0000 \\ 0.0000 \\ 2.0000 \\ 0.1000 \end{pmatrix} \tag{3.12}$$

We can now connect this to states with large ($p = 10$) and small ($p = 0.1$) pressures again:

```
In [9]: wave_1_reactive = wave.Wave(U_reactive, unknown_value=10, wavenumber=0)
        wave_2_reactive = wave.Wave(U_reactive, unknown_value=0.1, wavenumber=0)
```

The analogue of the inert shock is the strong detonation:

```
In [10]: wave_1_reactive
```

$$\mathcal{SDT}_{\leftarrow} : \lambda^{(0)} = -0.7979 \tag{3.13}$$

The analogue of the rarefaction is the weak deflagration:

```
In [11]: wave_2_reactive
```

$$(\mathcal{CJDF}_{\leftarrow}\mathcal{R}_{\leftarrow}) : \lambda^{(0)} \in [-0.6097, 0.7607] \tag{3.14}$$

In this case we have a CJ deflagration, which is attached to a rarefaction to complete the wave.

3.5 The effect of tangential velocities

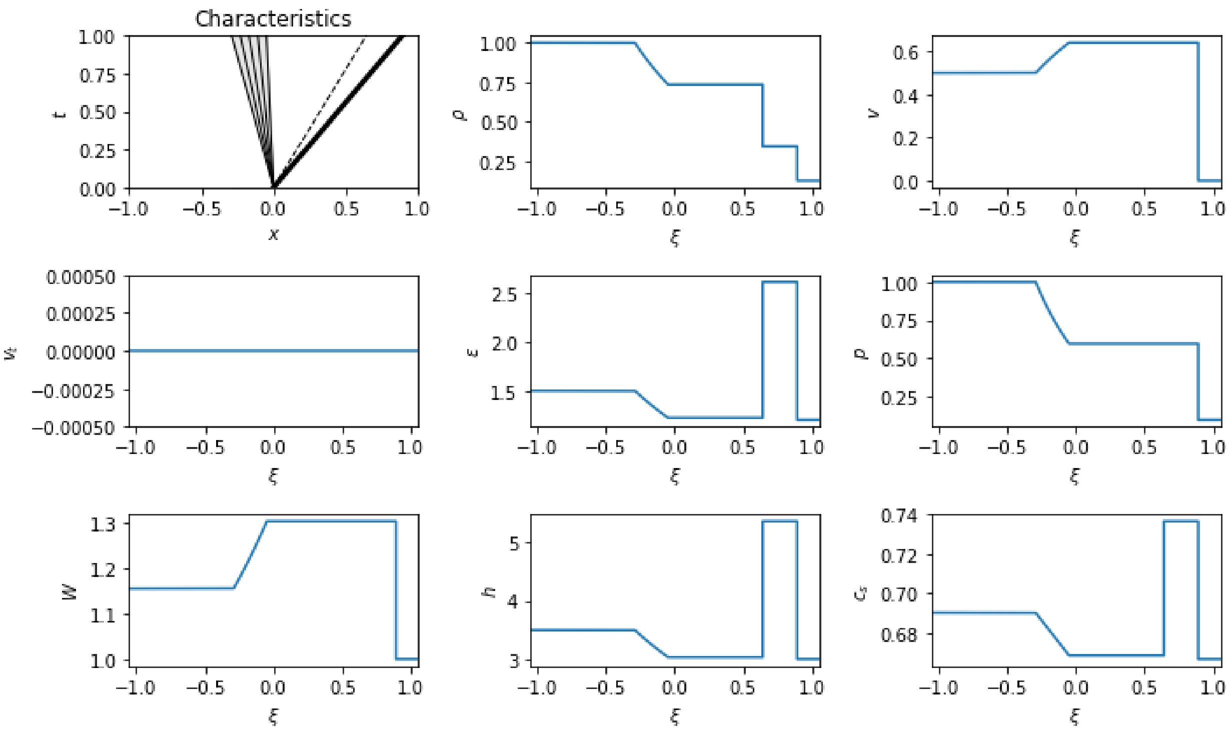
It was first noted by Rezzolla, Zanotti and Pons that there is one effect that is specific to *relativistic* Riemann problems. Thanks to the Lorentz factor coupling the tangential velocities more tightly to the other terms, it is possible to change the wave pattern in the problem solely by changing the tangential velocity. This cannot happen in the Newtonian case.

The examples used in their paper use a *modified* relativistic Sod problem as a base. We set that problem up first:

```
In [1]: from r3d2 import eos_defns, State, RiemannProblem
        from IPython.display import display, display_png
```

```
In [2]: eos = eos_defns.eos_gamma_law(5.0/3.0)
        U_left_sod_modified = State(1.0, 0.5, 0.0, 1.5, eos)
        U_right_sod_modified = State(0.125, 0.0, 0.0, 1.2, eos)
        rp_sod_modified = RiemannProblem(U_left_sod_modified, U_right_sod_modified)
        display(rp_sod_modified)
        display_png(rp_sod_modified)
```

$$\begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 1.0000 \\ 0.5000 \\ 0.0000 \\ 1.5000 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 0.1250 \\ 0.0000 \\ 0.0000 \\ 1.2000 \end{pmatrix}, \end{cases} \implies \mathcal{R}_{\leftarrow}\mathcal{C}\mathcal{S}_{\rightarrow}, \quad p_* = 0.5974, \quad \begin{cases} \mathcal{R}_{\leftarrow} : \lambda^{(0)} \in [-0.2902, -0.0488], \\ \mathcal{C} : \lambda^{(1)} = 0.6407, \\ \mathcal{S}_{\rightarrow} : \lambda^{(2)} = 0.8900, \end{cases} \begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star_L} = \begin{pmatrix} 0.7341 \\ 0.6407 \\ 0.0000 \\ 1.2207 \end{pmatrix} \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star_R} = \begin{pmatrix} 0.3427 \\ 0.6407 \\ 0.0000 \\ 2.6154 \end{pmatrix} \end{cases} \tag{3.15}$$

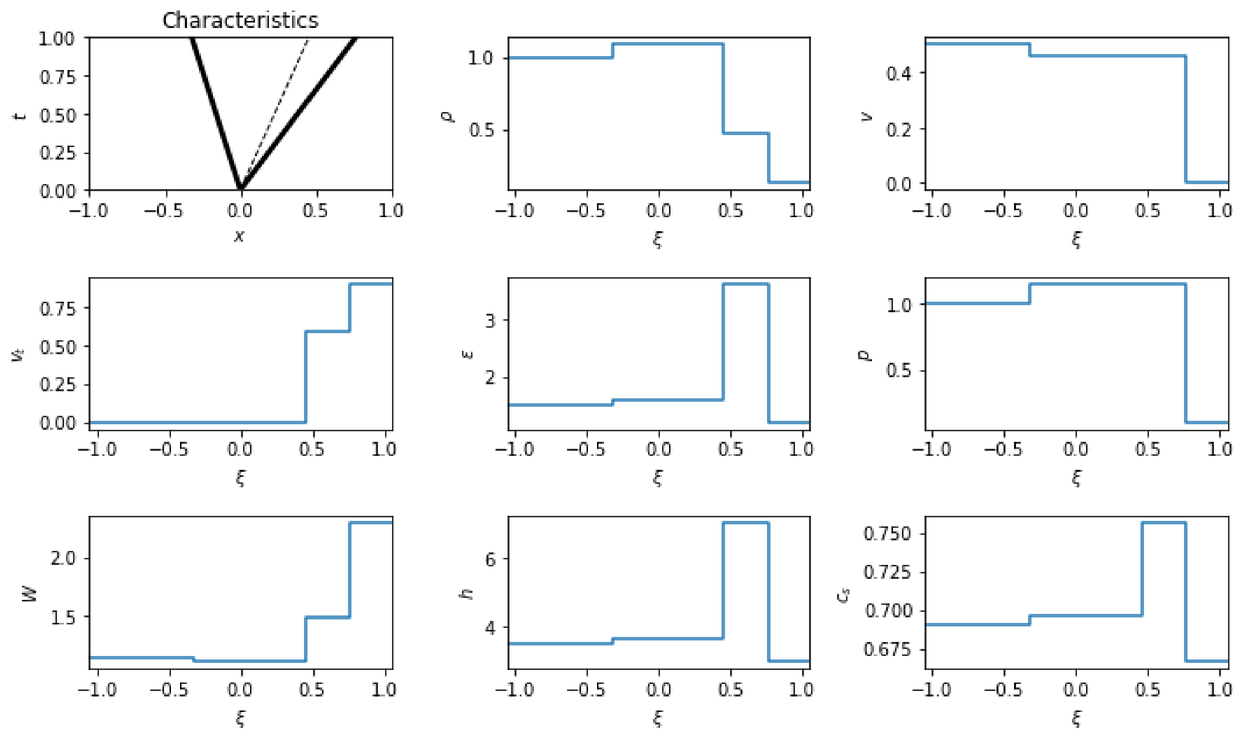


We see the wave pattern has a left going rarefaction and a right going shock.

Now, by modifying the tangential velocity of the right state, the wave pattern changes:

```
In [3]: U_right_vt_0_9 = State(0.125, 0.0, 0.9, 1.2, eos)
        rp_vt_0_9 = RiemannProblem(U_left_sod_modified, U_right_vt_0_9)
        display(rp_vt_0_9)
        display_png(rp_vt_0_9)
```

$$\begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 1.0000 \\ 0.5000 \\ 0.0000 \\ 1.5000 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 0.1250 \\ 0.0000 \\ 0.9000 \\ 1.2000 \end{pmatrix}, \end{cases} \implies \mathcal{S}_{\leftarrow}\mathcal{C}\mathcal{S}_{\rightarrow}, \quad p_* = 1.1509, \quad \begin{cases} \mathcal{S}_{\leftarrow} : \lambda^{(0)} = -0.3224, \\ \mathcal{C} : \lambda^{(1)} = 0.4549, \\ \mathcal{S}_{\rightarrow} : \lambda^{(2)} = 0.7638, \end{cases} \begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star_L} = \begin{pmatrix} 1.0879 \\ 0.4549 \\ 0.0000 \\ 1.5868 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star_R} = \begin{pmatrix} 0.4748 \\ 0.4549 \\ 0.5873 \\ 3.6363 \end{pmatrix}. \end{cases} \tag{3.16}$$

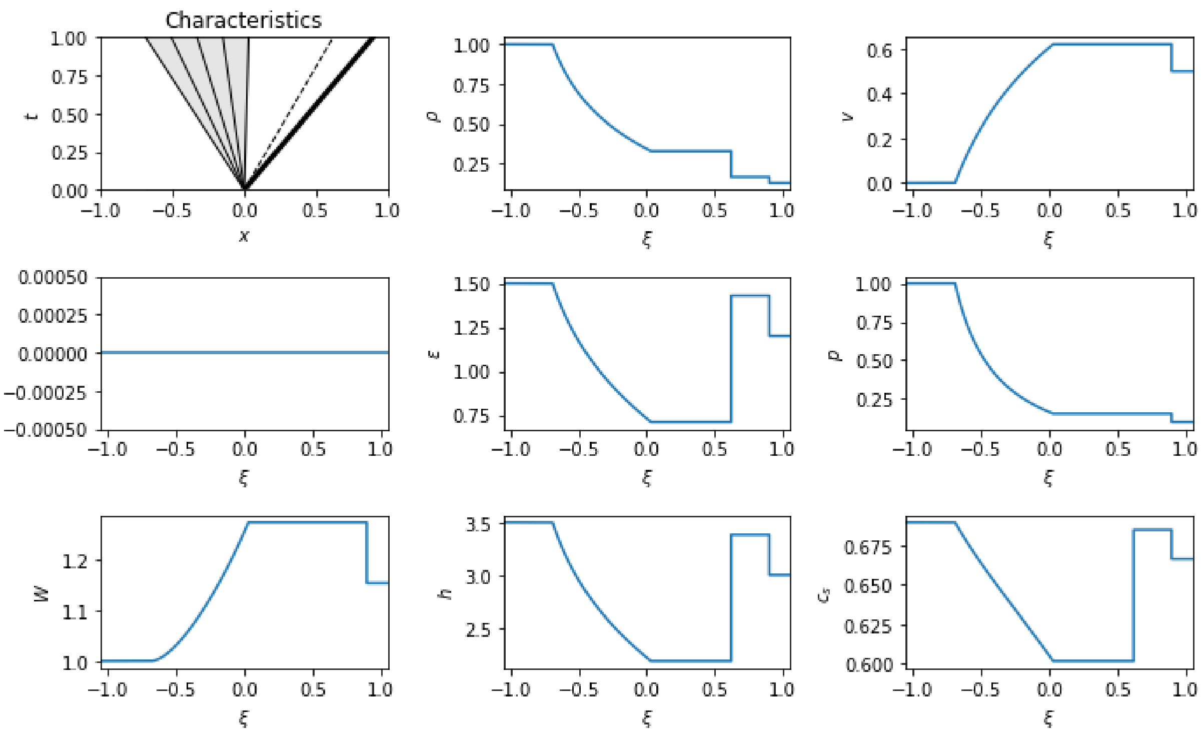


The result is a left going *shock*, rather than a rarefaction.

A shock-rarefaction problem can be modified, via a tangential velocity, to a two rarefaction problem in a similar fashion:

```
In [4]: U_left_sod_modified2 = State(1.0, 0., 0.0, 1.5, eos)
        U_right_sod_modified2 = State(0.125, 0.5, 0.0, 1.2, eos)
        rp_sod_modified2 = RiemannProblem(U_left_sod_modified2, U_right_sod_modified2)
        display(rp_sod_modified2)
        display_png(rp_sod_modified2)
```

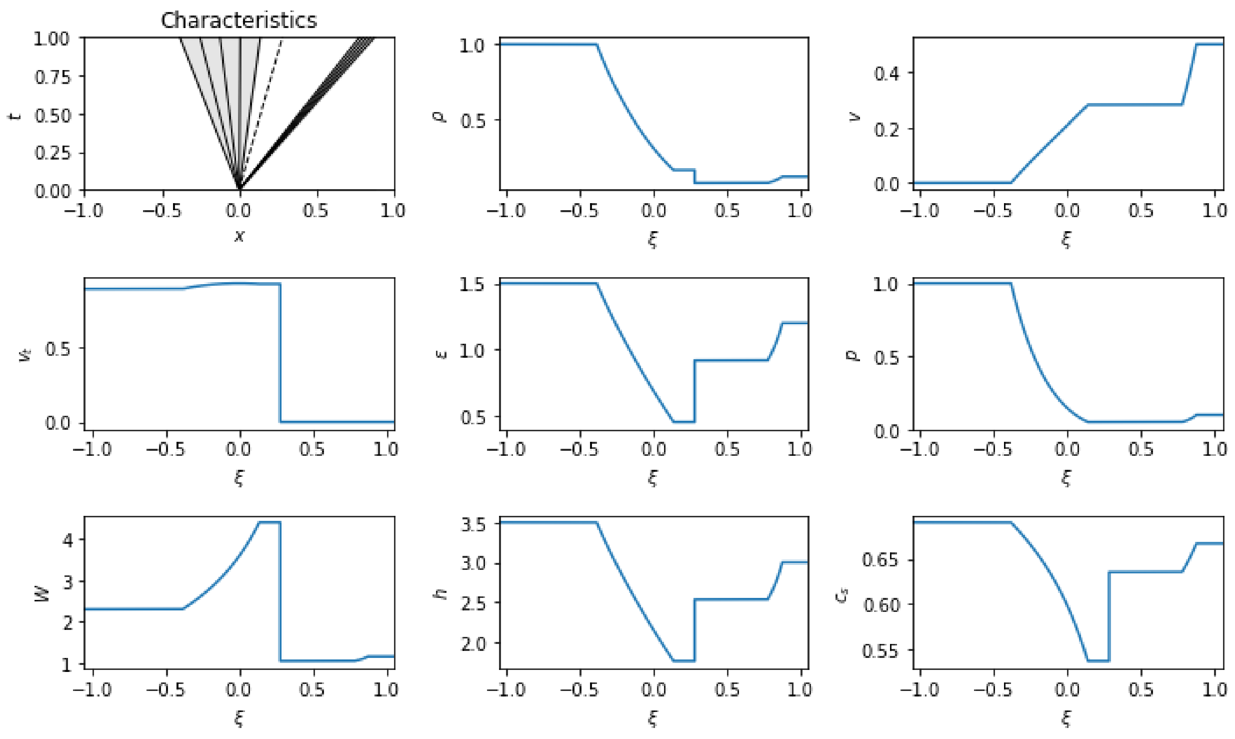
$$\left\{ \begin{array}{c} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 1.0000 \\ 0.0000 \\ 0.0000 \\ 1.5000 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 0.1250 \\ 0.5000 \\ 0.0000 \\ 1.2000 \end{pmatrix}, \end{array} \right. \implies \mathcal{R}_{\leftarrow}\mathcal{C}\mathcal{S}_{\rightarrow}, \quad p_* = 0.1546, \quad \left\{ \begin{array}{l} \mathcal{R}_{\leftarrow} : \lambda^{(0)} \in [-0.6901, 0.0309], \\ \mathcal{C} : \lambda^{(1)} = 0.6206, \\ \mathcal{S}_{\rightarrow} : \lambda^{(2)} = 0.8996, \end{array} \right. \quad \left\{ \begin{array}{c} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star L} = \begin{pmatrix} 0.3262 \\ 0.6206 \\ 0.0000 \\ 0.7108 \end{pmatrix} \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star R} = \begin{pmatrix} 0.1621 \\ 0.6206 \\ 0.0000 \\ 1.4302 \end{pmatrix} \end{array} \right. \tag{3.17}$$



And now with tangential velocity:

```
In [5]: U_left_vt_0_9 = State(1.0, 0.0, 0.9, 1.5, eos)
        rp2_vt_0_9 = RiemannProblem(U_left_vt_0_9, U_right_sod_modified2)
        display(rp2_vt_0_9)
        display_png(rp2_vt_0_9)
```

$$\begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 1.0000 \\ 0.0000 \\ 0.9000 \\ 1.5000 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 0.1250 \\ 0.5000 \\ 0.0000 \\ 1.2000 \end{pmatrix}, \end{cases} \implies \mathcal{R}_{\leftarrow}\mathcal{C}\mathcal{R}_{\rightarrow}, \quad p_* = 0.0513, \quad \begin{cases} \mathcal{R}_{\leftarrow} : \lambda^{(0)} \in [-0.3838, 0.1375], \\ \mathcal{C} : \lambda^{(1)} = 0.2809, \\ \mathcal{R}_{\rightarrow} : \lambda^{(2)} \in [0.7773, 0.8750], \end{cases} \begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star L} = \begin{pmatrix} 0.1683 \\ 0.2809 \\ 0.9324 \\ 0.4573 \end{pmatrix} \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix}_{\star R} = \begin{pmatrix} 0.0838 \\ 0.2809 \\ 0.0000 \\ 0.9189 \end{pmatrix} \end{cases} \tag{3.18}$$

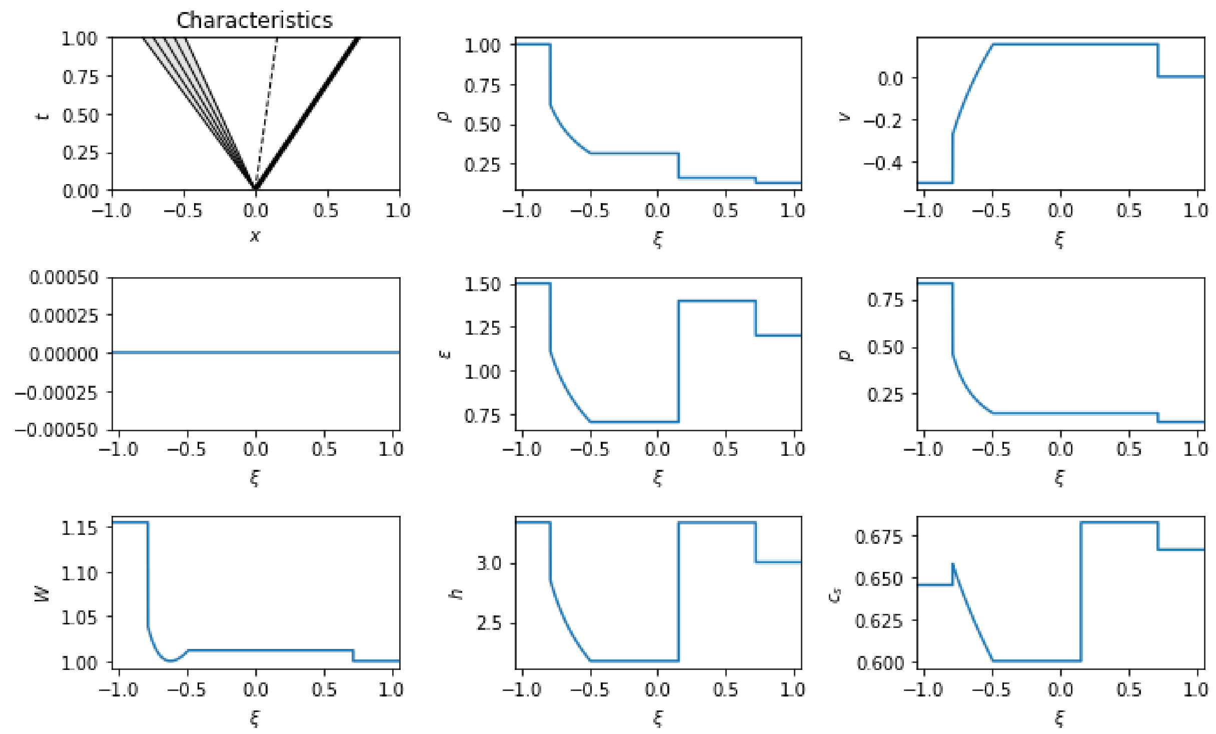


3.6 Reactive case

An interesting question is on the potential impact of this relativistic effect on detonations and deflagrations. Let us set up a reference reactive problem and add tangential velocities:

```
In [6]: eos_reactive = eos_defns.eos_gamma_law_react(5.0/3.0, 0.25, 1.0, 1.0, eos)
        U_reactive = State(1.0, -0.5, 0.0, 1.5, eos_reactive)
        U_burnt = State(0.125, 0.0, 0., 1.2, eos)
        rp_reactive_base = RiemannProblem(U_reactive, U_burnt)
        display(rp_reactive_base)
        display_png(rp_reactive_base)
```

$$\begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \\ q \end{pmatrix} = \begin{pmatrix} 1.0000 \\ -0.5000 \\ 0.0000 \\ 1.5000 \\ 0.2500 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 0.1250 \\ 0.0000 \\ 0.0000 \\ 1.2000 \end{pmatrix}, \end{cases} \implies \quad (\mathcal{CJDF}_{\leftarrow}\mathcal{R}_{\leftarrow})\,\mathcal{CS}_{\rightarrow}, \quad p_* = 0.1464, \quad \begin{cases} (\mathcal{CJDF}_{\leftarrow}\mathcal{R}_{\leftarrow}) : \lambda^{(0)} \in [-0.7887, -0.4919], \\ \mathcal{C} : \lambda^{(1)} = 0.1532, \\ \mathcal{S}_{\rightarrow} : \lambda^{(2)} = 0.7187, \end{cases} \tag{3.19}$$

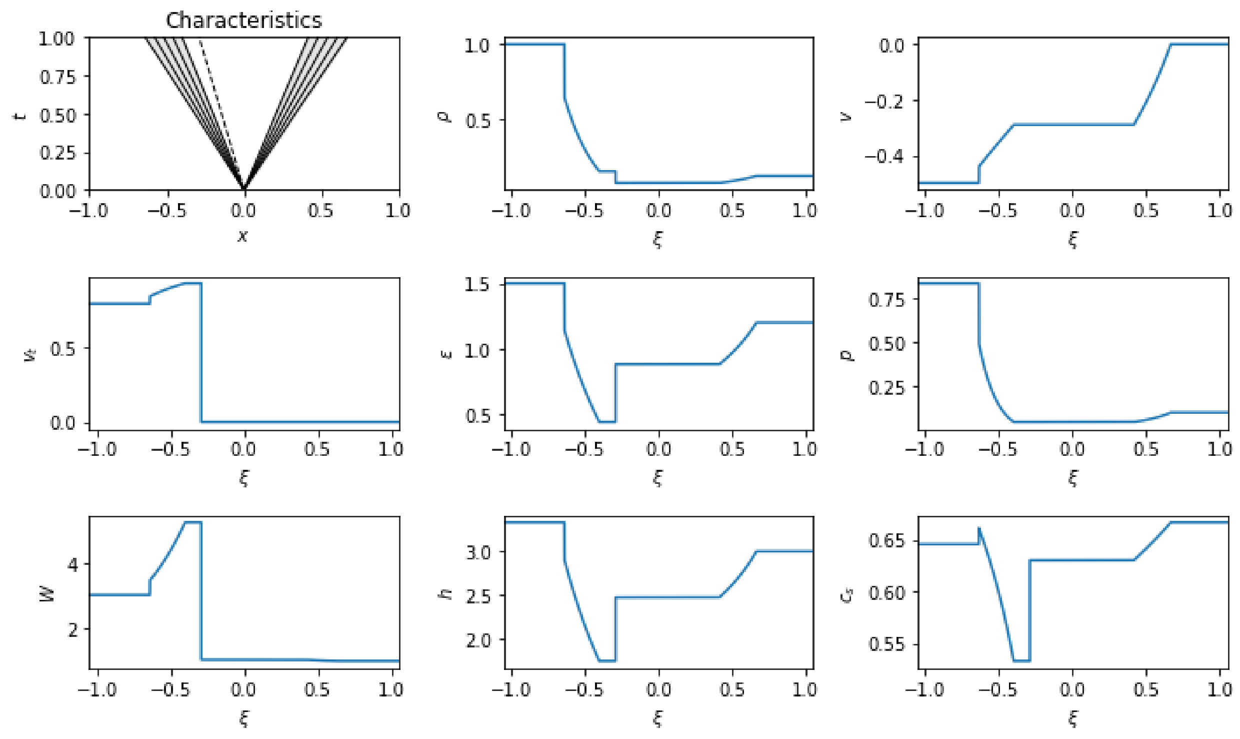


We see that this is a deflagration-shock problem.

Now add tangential velocities to the unburnt material:

```
In [7]: U_reactive_vt_0_9 = State(1.0, -0.5, 0.8, 1.5, eos_reactive)
        rp_reactive_vt_reactive = RiemannProblem(U_reactive_vt_0_9, U_burnt)
        display(rp_reactive_vt_reactive)
        display_png(rp_reactive_vt_reactive)
```

$$\begin{cases} \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \\ q \end{pmatrix} = \begin{pmatrix} 1.0000 \\ -0.5000 \\ 0.8000 \\ 1.5000 \\ 0.2500 \end{pmatrix}, \\ \begin{pmatrix} \rho \\ v_x \\ v_t \\ \epsilon \end{pmatrix} = \begin{pmatrix} 0.1250 \\ 0.0000 \\ 0.0000 \\ 1.2000 \end{pmatrix}, \end{cases} \implies (\mathcal{CJDF}_{\leftarrow}\mathcal{R}_{\leftarrow})\,\mathcal{CR}_{\rightarrow}, \quad p_* = 0.0464, \quad \begin{cases} (\mathcal{CJDF}_{\leftarrow}\mathcal{R}_{\leftarrow}) : \lambda^{(0)} \in [-0.6382, -0.4005], \\ \mathcal{C} : \lambda^{(1)} = -0.2901, \\ \mathcal{R}_{\rightarrow} : \lambda^{(2)} \in [0.4159, 0.6667], \end{cases} \tag{3.20}$$



Again, by adding tangential velocities to the right state the shock has become a rarefaction.

3.7 Parametric plots

When looking at Riemann problems, it is common to think about how a *wave* connects a *state* that is known to other states. For example, the Hugoniot curve gives the set of states that can be connected to a known state by a shock. Solutions to Riemann problems can be illustrated by the intersections of such curves in some parametric space.

Here we will illustrate this by showing how the curves for various problems behave in the pressure-velocity $(P - v)$ space.

3.7.1 Inert case

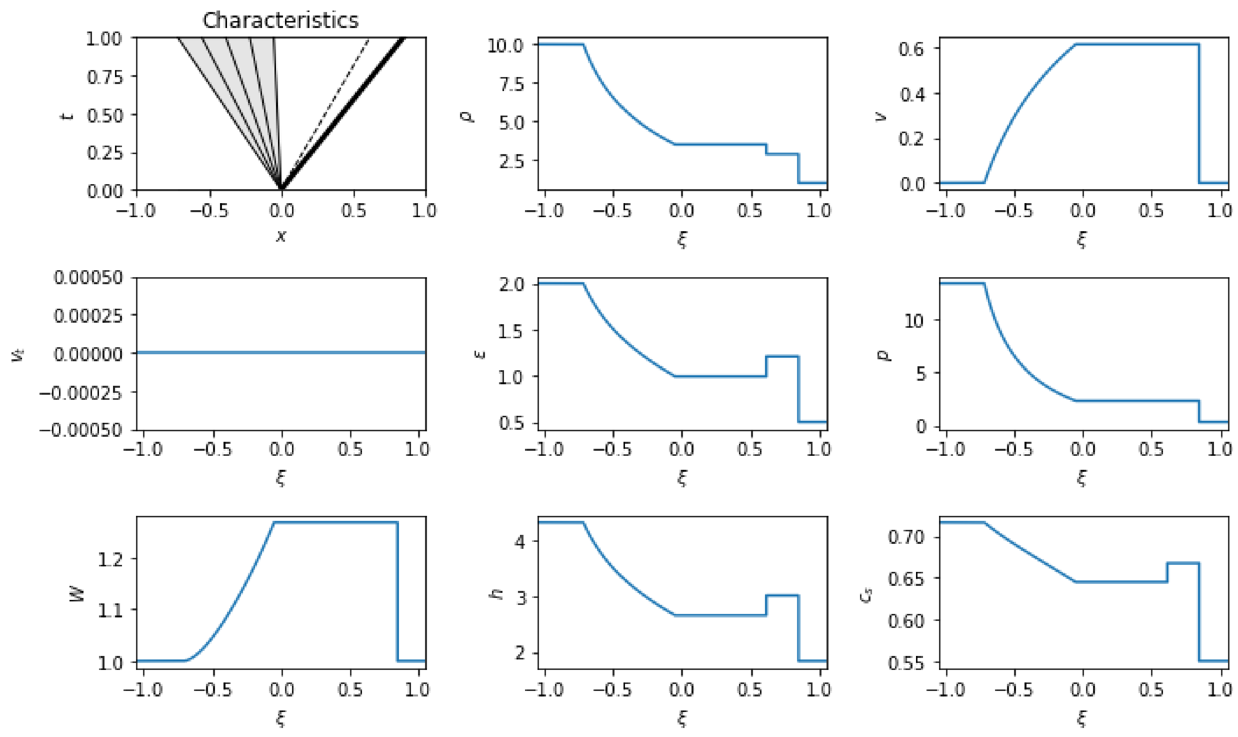
```
In [1]: from r3d2 import eos_defns, State, RiemannProblem, utils
        from matplotlib import pyplot
        %matplotlib inline
```

```
In [2]: gamma = 5.0/3.0
        eos = eos_defns.eos_gamma_law(gamma)
        test_1_U_left = State(10.0, 0.0, 0.0, 2.0, eos, label="L")
        test_1_U_right = State(1.0, 0.0, 0.0, 0.5, eos, label="R")

        test_1_rp = RiemannProblem(test_1_U_left, test_1_U_right)
```

This Riemann problem has a simple rarefaction-shock structure: a left going rarefaction, a contact, and a right going shock.

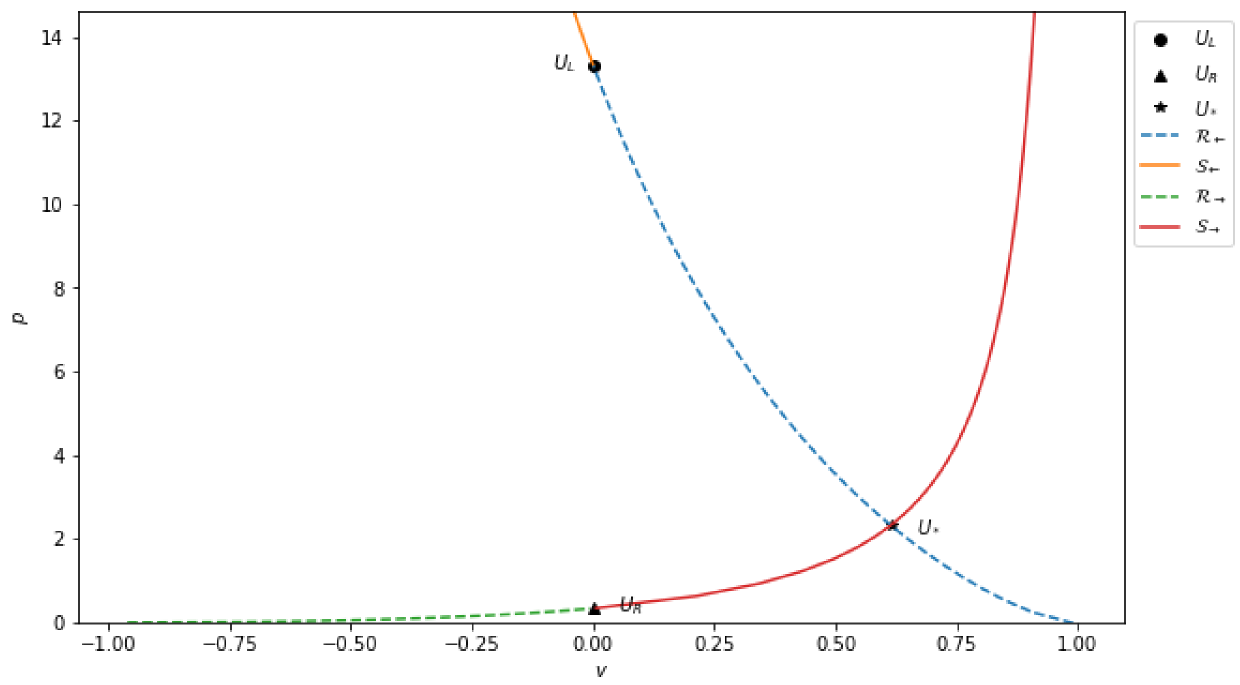
```
In [3]: from IPython.display import display, display_png
        display_png(test_1_rp)
```



The solution in parameter space shows the curves that can connect to the left and right states. Both a shock and a rarefaction can connect to either state. However, the only intersection (for the central, "star" state) is when a left going rarefaction connects to the left state, and a right going shock connects to the right state:

```
In [4]: fig = pyplot.figure(figsize=(10,6))
        ax = fig.add_subplot(111)
        utils.plot_P_v(test_1_rp, ax, fig)
```

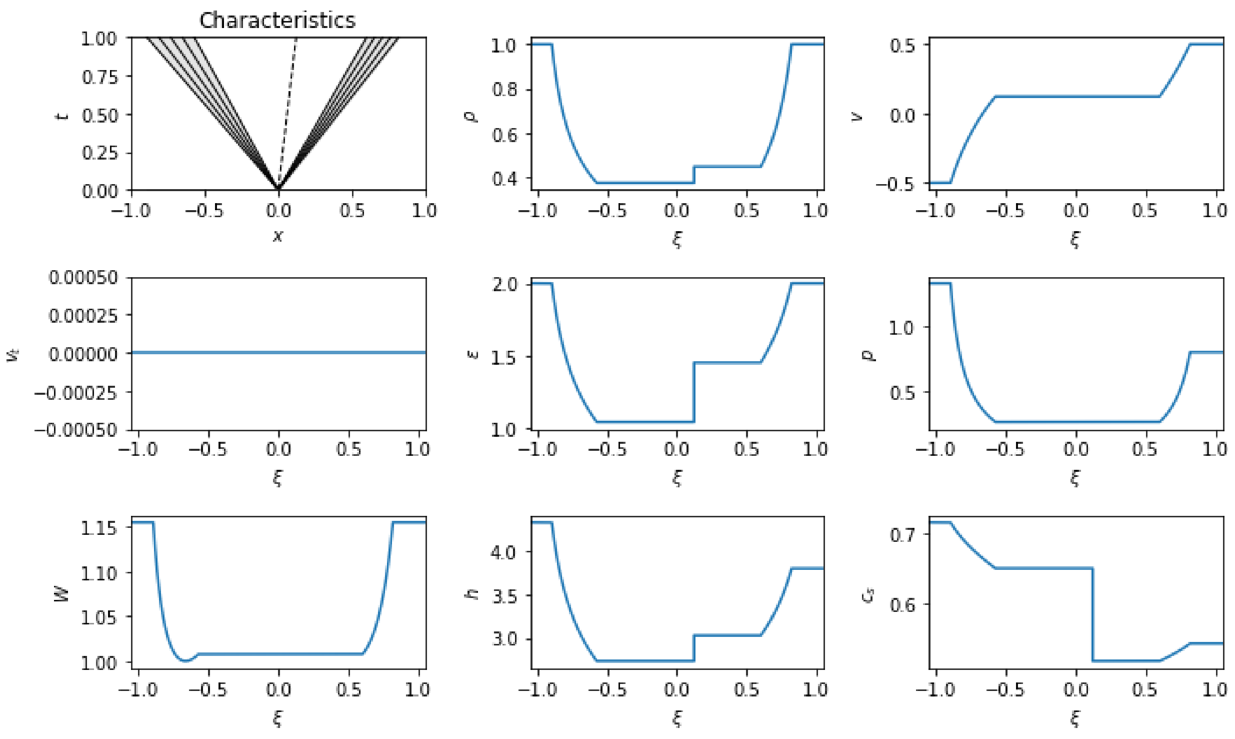
```
/home/alice/anaconda3/lib/python3.6/site-packages/scipy/integrate/odepack.py:218: ODEintWarning: Exce
  warnings.warn(warning_msg, ODEintWarning)
```



3.7.2 Varying EOS

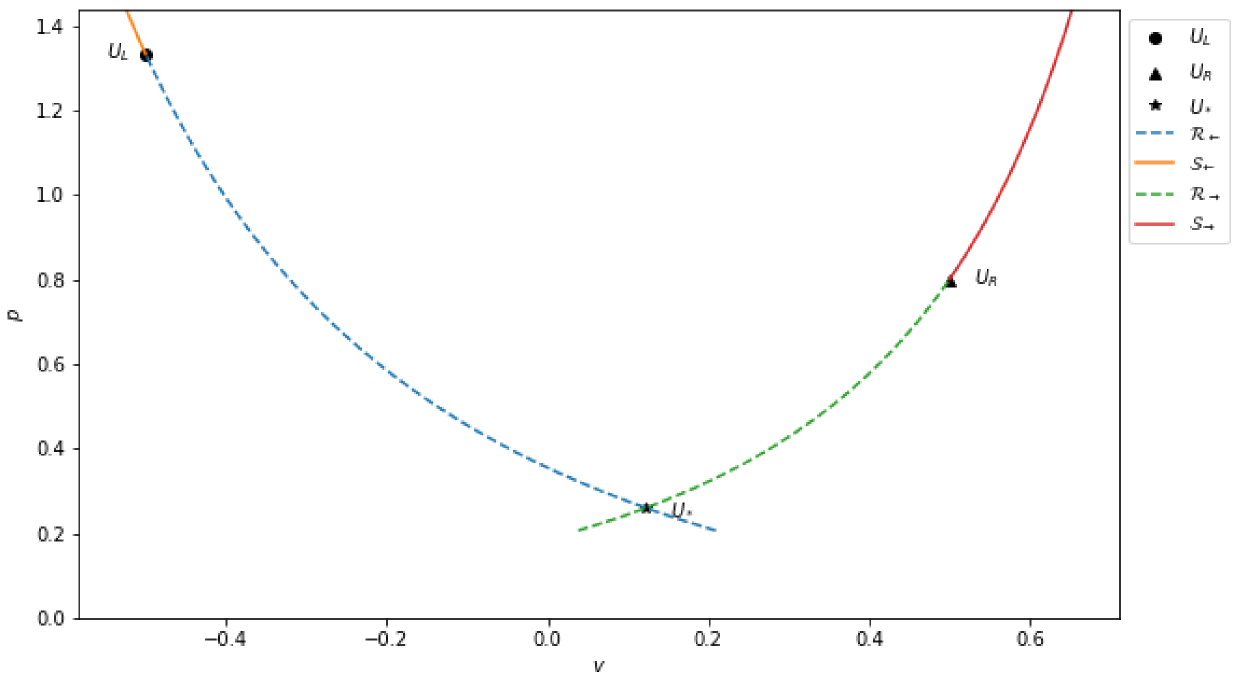
As noted previously, it's not necessary for the equation of state to match in the different states. Here is a problem with two inert EOSs. This problem is solved by two rarefactions.

```
In [5]: gamma_air = 1.4
        eos_air = eos_defns.eos_gamma_law(gamma_air)
        U_vary_eos_L = State(1.0, -0.5, 0.0, 2.0, eos, label="L")
        U_vary_eos_R = State(1.0, +0.5, 0.0, 2.0, eos_air, label="R")
        test_vary_eos_rp = RiemannProblem(U_vary_eos_L, U_vary_eos_R)
        display_png(test_vary_eos_rp)
```



The parametric solution in this case shows the pressure decreasing across both curves along rarefactions to get to the star state:

```
In [6]: fig2 = pyplot.figure(figsize=(10,6))
        ax2 = fig2.add_subplot(111)
        utils.plot_P_v(test_vary_eos_rp, ax2, fig2)
```

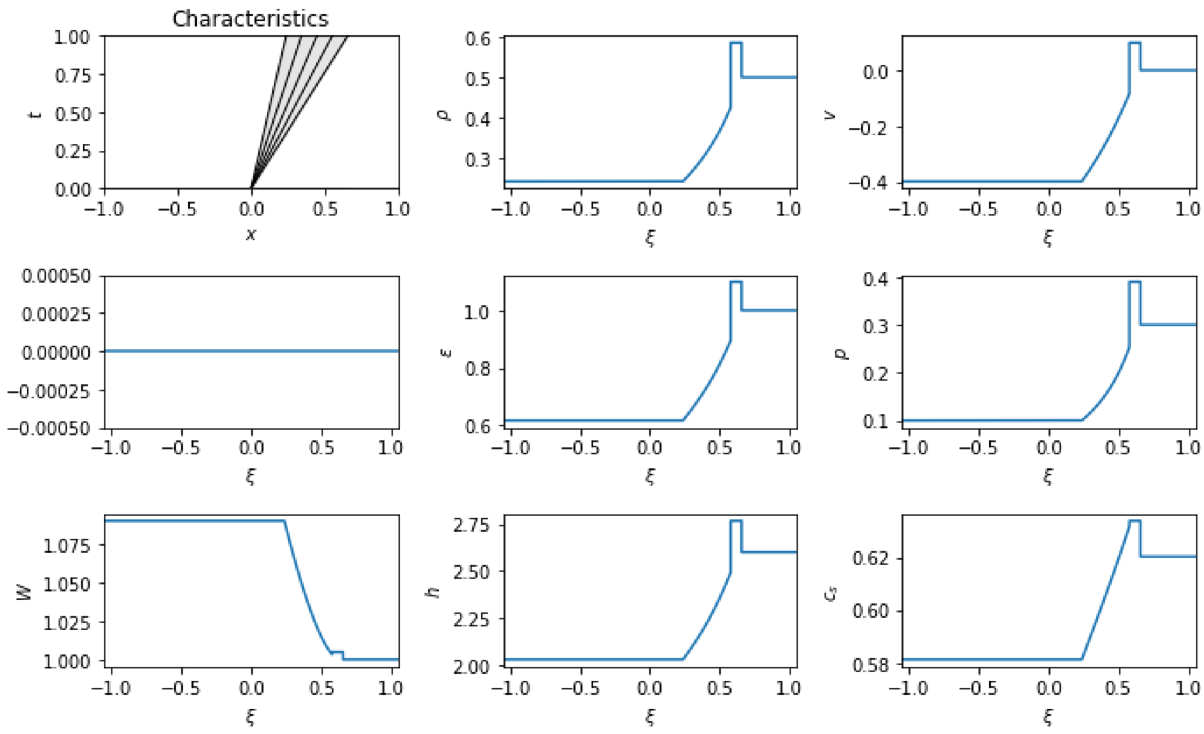


3.7.3 Reactive cases

Once reactions start, the behaviour gets more complex, and the parametric pictures can clarify some aspects. An example would be a single deflagration wave that is preceded by a shock to ignite the reaction:

```
In [7]: eos = eos_defns.eos_gamma_law(5.0/3.0)
        eos_reactive = eos_defns.eos_gamma_law_react(5.0/3.0, 0.1, 1.0, 1.0, eos)
        U_reactive_right = State(0.5, 0.0, 0.0, 1.0, eos_reactive)
        U_reactive_left = State(0.24316548798524526, -0.39922932397353039, 0.0,
                                0.61686385086179807, eos)
        test_precursor_rp = RiemannProblem(U_reactive_left, U_reactive_right)
        display_png(test_precursor_rp)
```

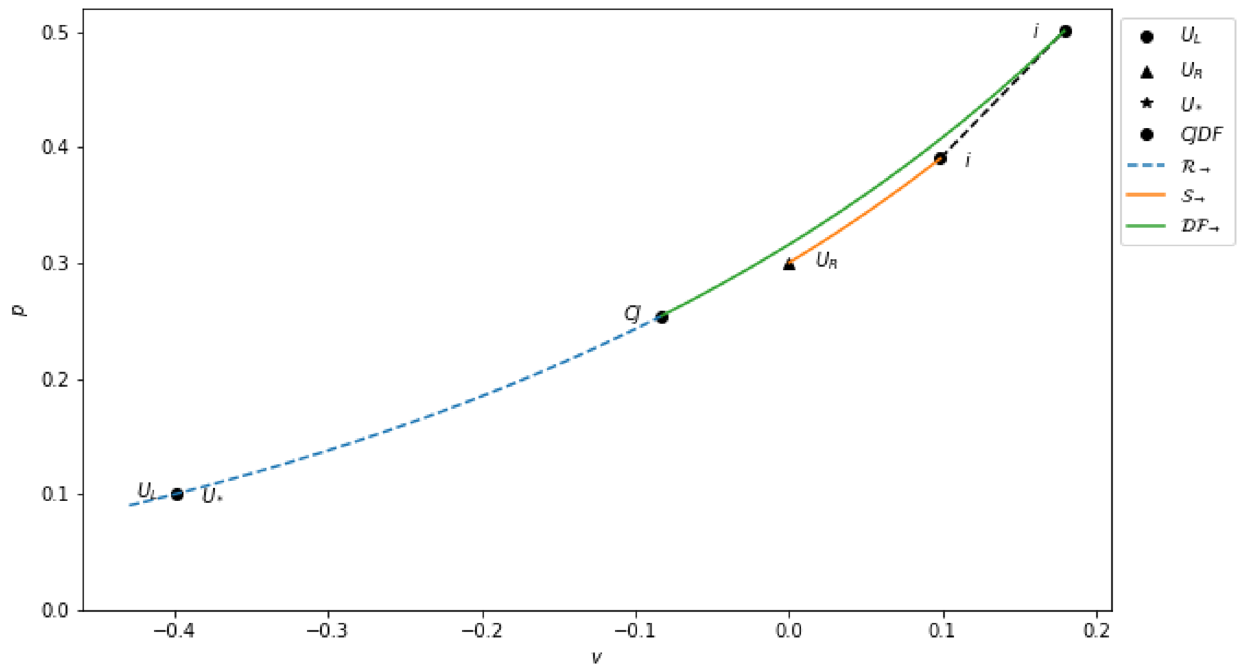
```
/home/alice/anaconda3/lib/python3.6/site-packages/scipy/integrate/odepack.py:218: ODEintWarning: Exce
  warnings.warn(warning_msg, ODEintWarning)
```



The structure here looks like previous Riemann problems, but there is in fact only one wave. The right state is connected directly to the left state across a compound wave formed from a precursor shock, which raises the temperature until the gas ignites, then a Chapman-Jouget deflagration which is attached to a rarefaction.

In parameter space we see this behaviour more directly:

```
In [8]: fig3 = pyplot.figure(figsize=(10,6))
        ax3 = fig3.add_subplot(111)
        utils.plot_P_v(test_precursor_rp, ax3, fig3)
```



The smooth joining of the deflagration and rarefaction curves at the Chapman-Jouget (CJ) point is expected, as the propagation speed of the waves must match there. The ignition point (where the temperature is high enough for the reaction to take place) is illustrated by the "i".