Introduction to Finite Volumn Method

Yiming

February 18, 2025

1 黎曼问题求解

计算黎曼问题的代码中的重建 (Reconstruction) 部分补用了 PLM，对应的将时间积分 (Integration) 部分改用了 RK2，以便实现要求，如图 (1,2)。解析解的对比考虑库 sodshock，计算空间范围选择 $(-1, 1)$，分辨率选择 256，边界条件选择外流边界条件。

关于课程中写的黎曼问题求解器（后称之为 diy），和标准库 sodshock 的结果对比不是很好，特别是在流体有一个整体速度时有很大差异，也许是因为我使用 sodshock 不当，但根据一些粗浅的分析我认为我们的 diy 程序是没有问题的。

1.1 流体整体静止

首先我们考虑流体整体静止的问题，即将问题中 $v_{\mathrm{L}} = v_{\mathrm{R}} = 0.2$ 改为 $v_{\mathrm{L}} = v_{\mathrm{R}} = 0$，我们先来看一下 diy 和 sodshock 结果对比。$t = 0$，$t = 0.2$，$t = 1$ 的结果如图 (3–5)，在此轻情况下二者基本符合。

```
    def integrate( self ):
        # 时间积分: RK2

        u0 = copy(self.u);
        self.cons2prim  (  );
        self.reconstruct(  );
        self.flux_all   (  );
        f0 = 1 / self.dx * ( self.f[ : -1 ] - self.f[ 1 : ] );

        u1t = u0 + self.dt * f0;
        self.u = u1t;
        self.cons2prim  (  );
        self.reconstruct(  );
        self.flux_all   (  );
        f1 = 1 / self.dx * ( self.f[ : -1 ] - self.f[ 1 : ] );

        self.u = u0 + self.dt * (f0 + f1) / 2;
        self.t += self.dt;
        return;
    #
#
```

Figure 1: Integration of RK2

```
def reconstruct( self ):

    # PLM限制器的reconstruct方法，参照公式(11)
    dwl = self.w[ self.n_gh - 1 : - self.n_gh ] - self.w[ self.n_gh -2 : - self.n_gh - 1 ];
    dwc = self.w[ self.n_gh : - self.n_gh + 1] - self.w[ self.n_gh - 1 : - self.n_gh ];
    if self.n_gh == 2:
        dwr = self.w[ self.n_gh + 1:] - self.w[ self.n_gh  : - self.n_gh + 1 ];
    else:
        dwr = self.w[ self.n_gh + 1: - self.n_gh + 2] - self.w[ self.n_gh  : - self.n_gh + 1 ];

    dwl = self.minmod( dwl, dwc );
    dwr = self.minmod( dwr, dwc );

    self.wl = self.w[ self.n_gh - 1 : - self.n_gh ] + dwl / 2;
    self.wr = self.w[ self.n_gh : - self.n_gh + 1 ] - dwr / 2;

    return;
```

Figure 2: Reconstruction of PLM

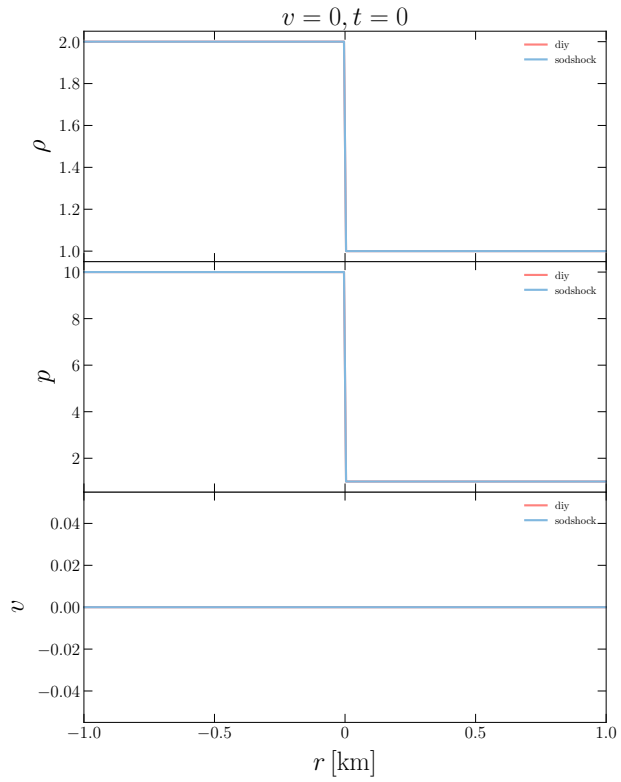

Figure 3: $v = 0, t = 0$ 两种黎曼问题求解器的对比

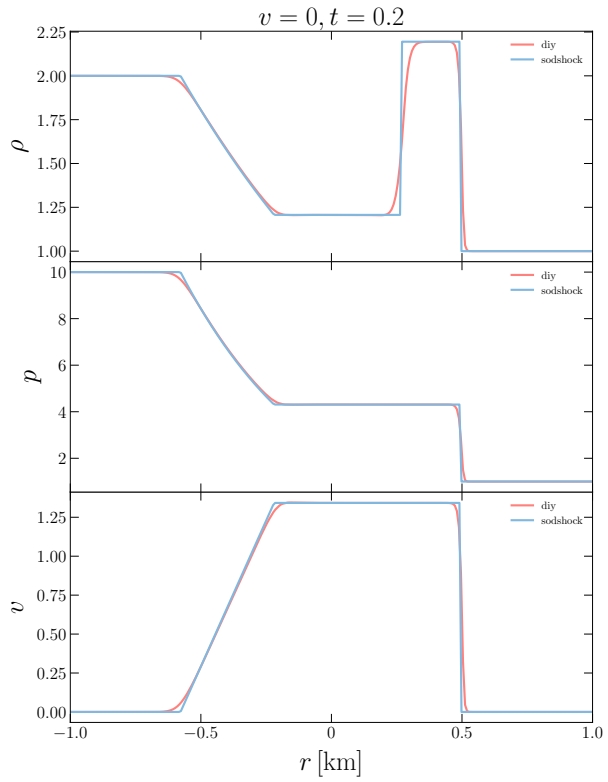

Figure 4: $v = 0, t = 0.2$ 两种黎曼问题求解器的对比

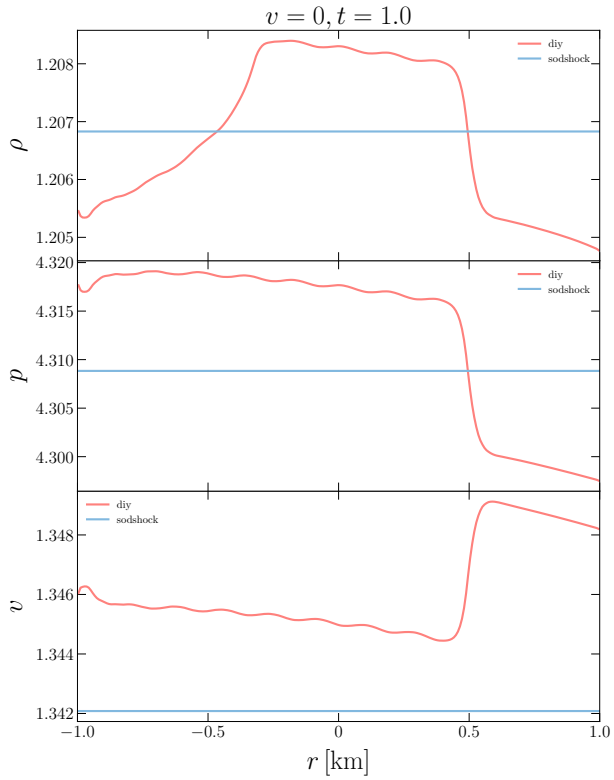

Figure 5: $v = 0, t = 1$ 两种黎曼问题求解器的对比

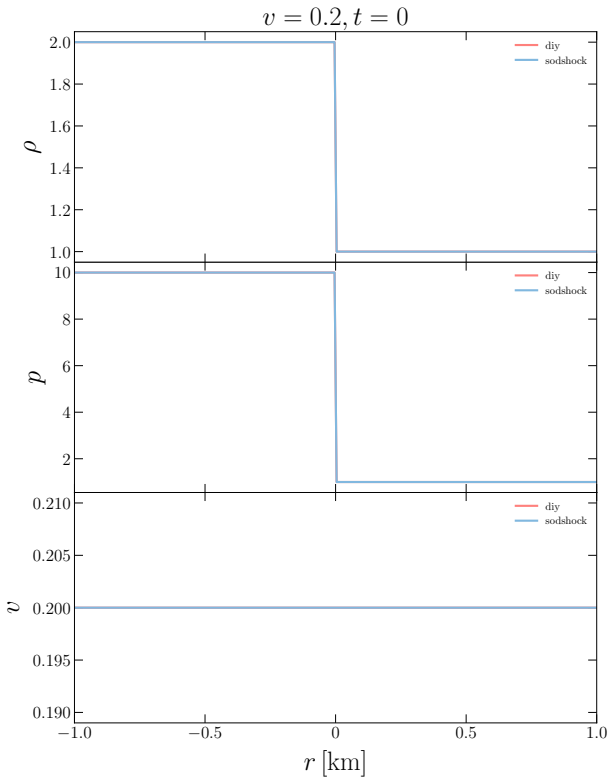

Figure 6: $v = 0.2, t = 0$ 两种黎曼问题求解器的对比

1.2 流体整体移动

考虑加上一个流体整体的速度，即 $v_{\mathrm{L}} = v_{\mathrm{R}} = 0.2$，再比较两个求解器的结果。首先我注意到比较困惑的一件事是，即使按照 sodshock 的 tutorial 设置了类似 $\mathrm{left_s tate} = (\mathrm{pressure} = 10., \mathrm{density} = 2., \mathrm{velocity} = 0.2)$，我发现流体的结构仍然不动，所以我粗暴的对横坐标做了一个坐标变换 $x' = x + 0.2t$，之后对比两个求解器结果，如图 (6−8)。可以看到密度和压强对的上，但是速度出现了差异，而且，特别是对于图 (8)，可以看到速度有一个整体的偏差，而且这个偏差在 0.2 左右，让我怀疑是否是这个整体速度没有进入到 sodshock 中。

我又补充了两组模拟，是 sodshock 自身的 $v = 0$ 和 $v = 0.2$ 结果对比，如图 (8) 以及 diy 自身的 $v = 0$ 和 $v = 0.2$ 结果对比，如图 (9)。我们可以看到的是，对于 sodshock，在未扰动区 $v = 0$ 和 $v = 0.2$ 两种情况下速度不同，但是扰动区速度竟然一致，这明显不正确，除非速度定义不同。diy 自身的结果至少看到有这样的 0.2 速度偏差，所以我觉得 diy 程序应该是可以通过整体加一个速度的测试？所以目前看来 diy 程序还是可信的。

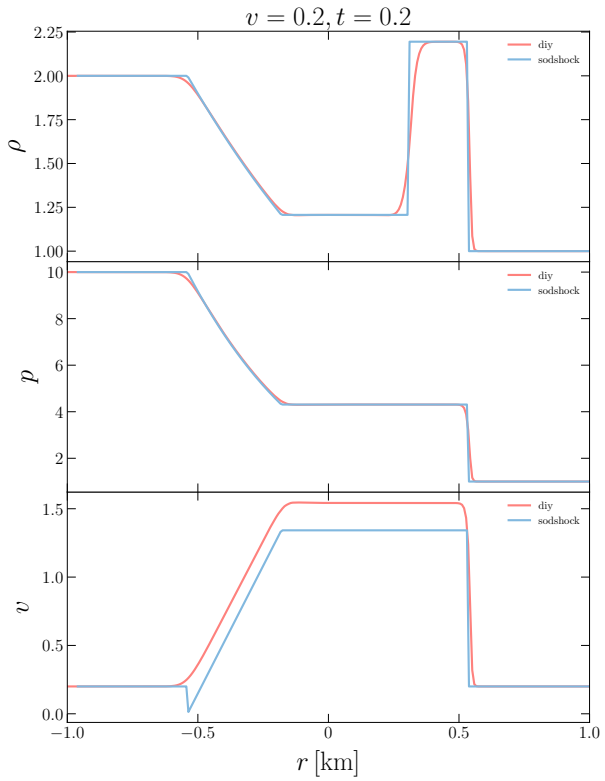

Figure 7: $v = 0.2, t = 0.2$ 两种黎曼问题求解器的对比

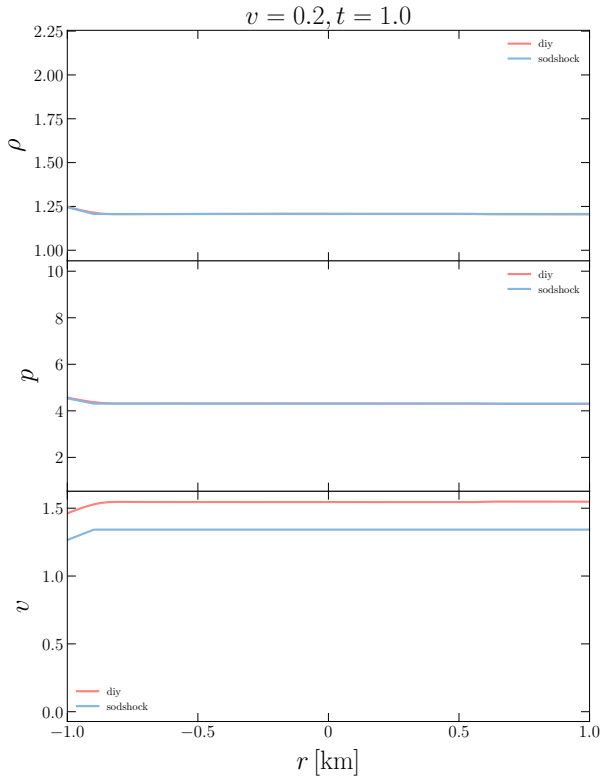

Figure 8: $v = 0.2, t = 1$ 两种黎曼问题求解器的对比

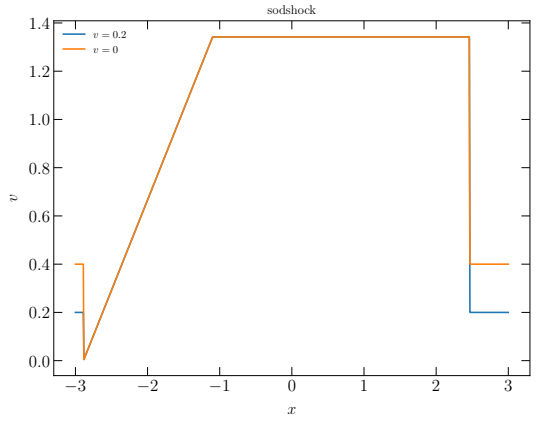

Figure 9: sodshock 的 $v = 0$，$v = 0.2$ 两种情况对比

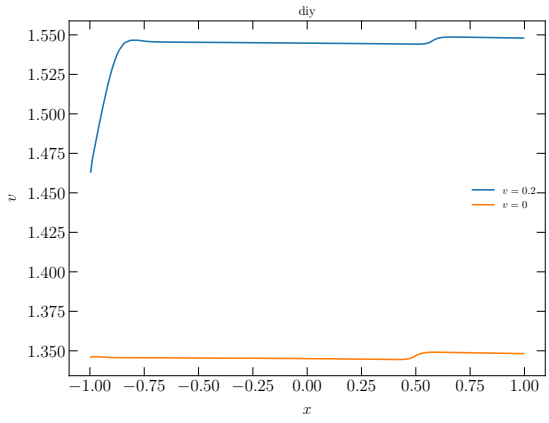

Figure 10: diy 的 $v = 0$，$v = 0.2$ 两种情况对比