Vectorial approach to momentum and mass conservation equations

Marius Berge Eide

AST5110: ITA, UiO

m.b.eide@astro.uio.no

October 27, 2014

Overview

1. Restating the problem
   - Momentum equation from N2L
   - Equations to be solved
2. Numerical solution
   - Solving in time
   - Solving in space
   - Equations restated
3. Results
   - Stability check
   - Stable without pressure
   - Stable for small amplitudes
4. To do
   - To do
   - Stagger code

Momentum equation from N2L

- Have a volume element ω with surface $\partial\omega$
- Change in momentum due to internal stresses and external forces:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\omega} \mathbf{p}\,\mathrm{d}v = \int_{\partial\omega} \mathbf{t}\cdot\mathbf{n}\,\mathrm{d}a + \int_{\omega} \mathbf{F}_{\mathrm{ext}}\,\mathrm{d}v \tag{1}$$

$$= \int_{\omega} \nabla\cdot\mathbf{t}\,\mathrm{d}v + \int_{\omega} \mathbf{F}_{\mathrm{ext}}\,\mathrm{d}v \tag{2}$$

onto local form:

$$\frac{\partial\mathbf{p}}{\partial t} + \mathbf{v}\cdot(\nabla\mathbf{p}) + \mathbf{p}\,(\nabla\cdot\mathbf{v}) = \nabla\cdot\mathbf{t} + \nabla p_{\mathrm{gas}} + \rho\mathbf{g} \tag{3}$$

where

$$\mathbf{v}\cdot(\nabla\mathbf{p}) + \mathbf{p}\,(\nabla\cdot\mathbf{v}) = \nabla\cdot(\mathbf{p}\mathbf{v}) \tag{4}$$

To solve

Momentum equation

$$\frac{\partial \mathbf{p}}{\partial t} = -\nabla \cdot (\mathbf{p}\mathbf{v}) + \nabla \cdot \mathbf{t} + \nabla p_{\mathrm{gas}} + \rho \mathbf{g} \tag{5}$$

Mass continuity equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{p} \tag{6}$$

Solving in time

- Euler:

$$\mathbf{f}^t(\mathbf{x}) = \frac{\partial \mathbf{F}(t, \mathbf{x})}{\partial t} = \frac{\mathbf{F}(\mathbf{x}, t + \Delta t) - \mathbf{F}(\mathbf{x}, t)}{\Delta t} + \mathcal{O}(\Delta t^2) \tag{7}$$

giving

$$\mathbf{F}^{t+1}(\mathbf{x}) \approx \mathbf{F}^t(\mathbf{x}) + \mathbf{f}^t(\mathbf{x})\Delta t \tag{8}$$

- Runge-Kutta 4th order

$$\mathbf{F}^{t+1} = \mathbf{F}^t + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{9}$$

where

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{F}^t(\mathbf{x}) \\
\mathbf{k}_2 &= \mathbf{F}^{t+1/2}\left(\mathbf{x} + \frac{\Delta t}{2}\mathbf{k}_1\right) \\
\mathbf{k}_3 &= \mathbf{F}^{t+1/2}\left(\mathbf{x} + \frac{\Delta t}{2}\mathbf{k}_2\right) \\
\mathbf{k}_4 &= \mathbf{F}^{t+1}(\mathbf{x} + \Delta t\mathbf{k}_3)
\end{aligned}$$

Solving in space

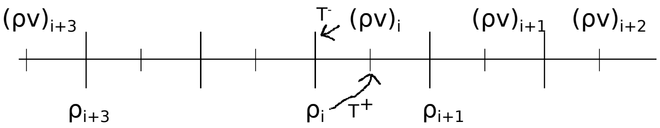

Figure: Staggered grid with cyclic boundary conditions

Interpolate forward half a step ($\Delta x/2$):

$$
\begin{aligned}
T_x^+(\mathbf{g}(x,y,z,\cdots)) &= \mathbf{g}(x + 1/2\Delta x, y, z, \cdots) = \mathbf{g}_{x+1/2} & (10)\\
&= a(\mathbf{g}_x + \mathbf{g}_{x+1}) + b\,(\mathbf{g}_{x-1}\mathbf{g}_{x+2}) + c\,(\mathbf{g}_{x-2} + \mathbf{g}_{x+3}) & (11)\\
&= \begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} \mathbf{g}_x + \mathbf{g}_{x+1} \\ \mathbf{g}_{x-1} + \mathbf{g}_{x+2} \\ \mathbf{g}_{x-2} + \mathbf{g}_{x+3} \end{bmatrix} & (12)
\end{aligned}
$$

$$a = \frac{1}{2} - b - c; \qquad b = -\frac{1}{24} - 5c; \qquad c = \frac{3}{640}$$

Solving in space

$$T_x^{+}(\mathbf{g}(x,y,z,\cdots)) = \mathbf{g}_{x+1/2} = \begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} \mathbf{g}_x + \mathbf{g}_{x+1} \\ \mathbf{g}_{x-1} + \mathbf{g}_{x+2} \\ \mathbf{g}_{x-2} + \mathbf{g}_{x+3} \end{bmatrix} \tag{13}$$

$$\partial_{,x}^{+}(\mathbf{g}(x,y,z,\cdots)) = \mathbf{g}'_{x+1/2} = \begin{bmatrix} a' & b' & c' \end{bmatrix} \cdot \begin{bmatrix} \mathbf{g}_x - \mathbf{g}_{x+1} \\ \mathbf{g}_{x-1} - \mathbf{g}_{x+2} \\ \mathbf{g}_{x-2} - \mathbf{g}_{x+3} \end{bmatrix} \tag{14}$$

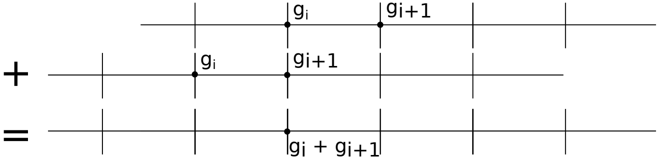

Figure: Numpy's `ROLL(array, steps)` causing cyclic boundaries, using this to interpolate and differentiate

Equations restated

Solve only for momentum ρv and density ρ, meaning the velocity can be found (at ρv's location):

$$v = \frac{\rho v}{T_z^+ [\rho]} \tag{15}$$

Momentum equation

$$\frac{\partial \rho v}{\partial t} = -\partial_{,z}^+ \left[\frac{1}{\rho} \left(T_z^- \left((\rho v)^2 \right) \right) \right] + \partial_{,z}^+ [p_{\mathrm{gas}} + Q] + T_z^+ [\rho] \, g_z \tag{16}$$

Mass continuity equation

$$\frac{\partial \rho}{\partial t} = -\partial_{,z}^- [\rho v] \tag{17}$$

Stability check

[http://77.237.250.152/wordpress/studier/stability_web.mp4]

Stable without pressure

[http://77.237.250.152/wordpress/studier/alive.mp4]

Stable for small amplitudes

[http://77.237.250.152/wordpress/studier/working.mp4]

To do:

1. Remember. There is no such thing as negative density
2. Try with sine peak as initial condition for ρ
3. Try to get the artificial diffusion implemented:

$$\nabla \cdot \mathbf{t} = \nabla \cdot \mathbf{Q} \overset{\text{1D}}{=} \mu \frac{\mathrm{d}v}{dx} \tag{18}$$

   with $\mu = \rho c_s \lambda$ and $\lambda = \nu \Delta x$.
4. Don't forget that there's no such thing as negative density

stagger.py

```
#!/usr/bin/env python

# AST5110
# Mon 29/09/2014
# Marius Berge Eide

import matplotlib.pylab as plt
import matplotlib.animation as animation
from numpy import *


# Exercise 2

# Operators

# Derivative

def deriv(farray, dx, dim=0, direction=1):
    """Derivative of array FARRAY along dimension DIM in
    direction DIRECTION, where:
     1: Shift 1/2 step forward
    -1: shift 1/2 step backward
    and DX is step length"""

    c =  3./640
    b = -1./24           - 5.*c
    a =  1.      - 3.*b  + 5.*c

    coeffs = array([a,b,c])  / dx

    p = +direction

    start= array([farray, \
            roll(farray, -p,     axis=dim), \
            roll(farray, -2*p,   axis=dim)  ])

    stop = array([roll(farray, p,     axis=dim), \
            roll(farray, 2*p,   axis=dim), \
            roll(farray, 3*p,   axis=dim)  ])

    return dot(coeffs, (start - stop))

def interp(farray, dim=0, direction=1):
    """Interpolates the array FARRAY along dimension DIM in
    direction DIRECTION returning the values at shifted 1/2 * DIRECTION
    location"""

    c =  3./256
    b = -1./16           - 3.*c
    a =  1./2   - b      - c

    coeffs = array([a,b,c])

    p = +direction

    start= array([farray, \
            roll(farray, -p,     axis=dim), \
            roll(farray, -2*p,   axis=dim)  ])

    stop = array([roll(farray, p,     axis=dim), \
            roll(farray, 2*p,   axis=dim), \
            roll(farray, 3*p,   axis=dim)  ])

    return dot(coeffs, (start + stop))
```

What happens when you have negative density

[http://77.237.250.152/wordpress/studier/blowup_web.mp4]