Fv

Jonathan Ulmer

August 14, 2025

Outline

```python
from typing import Callable
import numpy as np
from scipy.sparse import spdiags
from scipy.sparse.linalg import spsolve
from numpy.typing import NDArray
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

None

Standard Diffusion equation

$$
\begin{aligned}
\nabla \cdot (D(x)\nabla c) &= f(x) && \text{in } \Omega \\
c(x) &= 0 && \text{on } \partial\Omega
\end{aligned}
$$

Derivation of the 1D Finite Volume Method

The Finite Volume method considers the differential equation in Integral form over disjunct $(Q_i \cap Q_j = \emptyset, i \neq j)$ reference cells Q_i, $\bigcup_{i=1}^{N} Q_i = \Omega$ and calculates the integral over them, with an integral over the reference cell boundaries using Stokes integration.

$$
\int_{Q_i} \nabla \cdot (D(x)\nabla c) = \int_{Q_i} f(x)\,\mathrm{d}x \qquad i = 1, \dots, N \tag{1}
$$

$$
\int_{\partial Q_i} D(x)\nabla c \cdot \vec{n}\,\mathrm{d}S = \int_{Q_i} f(x)\,\mathrm{d}x \qquad i = 1, \dots, N \tag{2}
$$

The Finite Volume Method then considers the solution piecewise constant on Q. This creates discontinuities on the cell boundaries, where the values are not uniquely defined. The Finite Volume

Finite Volume 1D

Program Structure

For convenience in Explanation and Execution, we bundle all required information for solving a 1D system into a python class, which is structured as follows

```
class FVSolver:
    N : int
    h : np.float64
    x : NDArray[np.float64]
    D : Callable
    f : NDArray[np.float64]
    c : NDArray[np.float64]

    _T : NDArray[np.float64]

<<Init>>

<<Assemble Matrix>>

<<Boundary>>

<<Solve>>
```

Multiscale

In 1D

```
def set_multiscale_transmissions(self,
↪ resolution)->NDArray[np.float64]:
    micro_basis = np.zeros((self.N -1)*resolution)
    for i in range(self.N -1):
        micro_fv = FVSolver(resolution , self.D ,
        ↪ domain=(self.x[i] , self.x[i+1]))
        micro_fv.set_boundary(bc=(0.,1.))
        micro_fv.assemble_matrix()
        phi = micro_fv.solve()

        micro_basis[resolution * i:resolution*(i+1)] = phi
        hm = micro_fv.h
        self._T[i] = -hm * np.sum(((phi[1:] - phi[:-1])/hm)**2
        ↪ * self.D(micro_fv.x[:-1]))
    return micro_basis
```

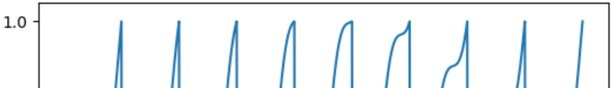
$$T_{\pm} = -\int_Q D(x)(\phi'_{\pm}(x))^2 \,\mathrm{d}x$$

Cleanup

```
None
```

```
from importlib import reload
import src.fvsolver
from src.fvsolver import FVSolver
reload(src.fvsolver)
epsilon = 0.1
D = lambda x: 1 / (2+1.9 * np.cos(2 * np.pi* x / epsilon))
fv = FVSolver(10 ,  D)
fv.assemble_matrix()
fv.set_boundary()
c_course = fv.solve()
plt.plot(c_course)
```

```
mb = fv.set_multiscale_transmissions(100)
plt.plot(mb)
```



2D

```
import scipy as sp
import numpy as np
class FVSolver2D:
   N : int
   M : int
   h_x : np.float64
   h_y : np.float64
   x : NDArray[np.float64]
   y : NDArray[np.float64]
   D : Callable
   f : NDArray[np.float64]
   c : NDArray[np.float64]

   _T_x : NDArray[np.float64]
   _T_y : NDArray[np.float64]

<<Init 2D>>

<<Assemble 2D Matrix>>

   def set_boundary(self , bc=(0.,0. , 0. , 0.)):
      self.f[ 0,1:-1]= bc[0]
```

2D Multiscale

```
def set_multiscale_transmissions(self, resolution):
    microscale_basis_x = np.zeros((self._T_x.shape[0] ,
    ↪  self._T_x.shape[1] , resolution))
    microscale_basis_y = np.zeros((self._T_y.shape[0] ,
    ↪  self._T_y.shape[1] , resolution))
    for i in range(self._T_x.shape[0]):
        for j in range(self._T_x.shape[1]):
            #Do mircroscale x
            D_micro = lambda x: self.D(x, self.y[j])
            fv_micro = FVSolver(resolution , D_micro,
            ↪  domain=(self.x[i] , self.x[i+1]))
            fv_micro.assemble_matrix()
            fv_micro.set_boundary(bc=(0.,1.))
            phi =fv_micro.solve()
            microscale_basis_x[i,j,:] = phi
            self._T_x[i,j] =   -fv_micro.h * self.h_y*
            ↪  np.sum(((phi[1:] - phi[:-1])/fv_micro.h)**2 *
            ↪  D_micro(fv_micro.x[:-1]))

    for i in range(self._T_y.shape[0]):
        for j in range(self._T_y.shape[1]):
            # Do microscale y
```

Reference Solution

Solution of the 2D Laplace equation:

$$-\Delta u(x,y) = f(x,y) \qquad \text{in} \quad \Omega \tag{5}$$

$$u(x,y) = 0 \qquad \text{on} \quad \Gamma_D \tag{6}$$

where $f(x,y) = 2*(x+y-x^2-y^2)$ the analytical solution is

$$u(x,y) = x*(1-x)*y*(1-y)$$