Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Home Affairs FDHA
Federal Office of Meteorology and Climatology MeteoSwiss

dusk & dawn - Vector Laplacian

DSL Training Workshop

Contents

- For this exercise, you will **combine the operators** you **developed in the last exercise** to compute yet another quantity from vector analysis
- This quantity is the **Vector Laplacian**

- You will notice that the approximation is of **low order** in this case. We will fix this in the next exercise

The Laplacian

$$\nabla^2(f(\mathbf{p})) = \nabla(\nabla f(\mathbf{p})) = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y} + \frac{\partial^2 f}{\partial^2 z}$$

- Takes a scalar function (scalar field) and returns a scalar
- Informally, the Laplacian of a function f at a point **p** measures by how much the average value of f over small spheres or balls centered at **p** deviates from f(**p**).
- Very wide spread in lots of physical equations
  - e.g. diffusion of chemical components, spread of heat in homogenous materials

The Vector Laplacian

$$\nabla^2 \mathbf{v} = \nabla(\nabla \mathbf{v}) = \nabla(\nabla \cdot \mathbf{v}) - \nabla \times (\nabla \times \mathbf{v})$$

- Takes a vector function (vector field) and returns a vector
- Also very relevant in physics, e.g. in the Navier Stokes Equation

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right) = \rho \mathbf{f} - \nabla p + \mu (\nabla^2 \mathbf{v})$$

f: body forces
p: pressure
mu: viscosity

- $\mu(\nabla^2 \mathbf{v})$ are the viscous stresses in the the fluid

MeteoSwiss

The Vector Laplacian

$$\nabla^2 \mathbf{v} = \nabla (\nabla \mathbf{v}) = \nabla (\nabla \cdot \mathbf{v}) - \nabla \times (\nabla \times \mathbf{v})$$

- This form of the vector Laplacian is quite general
- There are more straightforward for Cartesian coordinates
- However, the **normal component** of the form above lends itself very well to implementation on a FVM mesh:

$$\nabla^2 (\mathbf{v} \cdot \mathbf{n}) = (\nabla \cdot \mathbf{n}) \underbrace{[\nabla \cdot \mathbf{v}]}_{\text{divergence}} - (\nabla \cdot \tau) \underbrace{[\nabla \times \mathbf{v}]}_{\text{curl}}$$

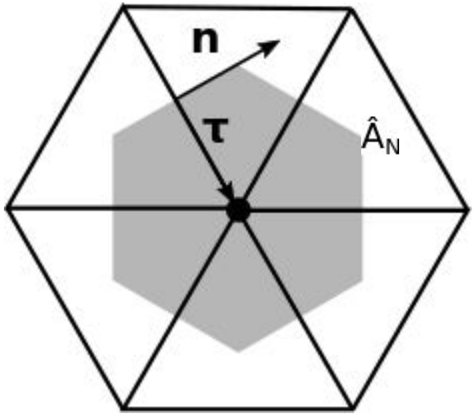
MeteoSwiss

c.f. Hui Wan - Developing and testing a hydrostatic atmospheric dynamical core on triangular grids

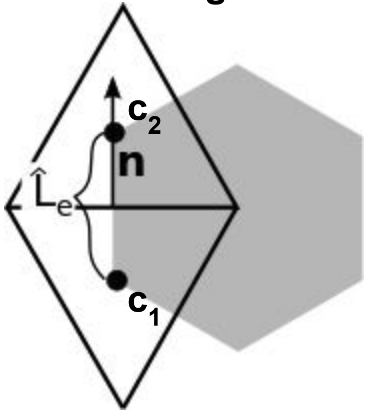
The Vector Laplacian

$$\nabla^2(\mathbf{v}\cdot\mathbf{n}) = \mathrm{grad}_n \underbrace{[\nabla\cdot\mathbf{v}]}_{\text{divergence}} - \mathrm{grad}_\tau \underbrace{[\nabla\times\mathbf{v}]}_{\text{curl}}$$

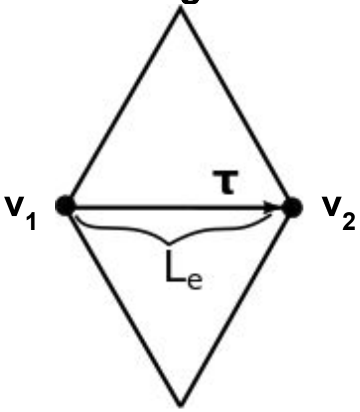
sketch triangular mesh



directional gradient n



directional gradient τ



$$\langle\nabla_n f\rangle_e = \frac{f(\mathbf{c_2}) - f(\mathbf{c_1})}{\hat{L}_e} \qquad \langle\nabla_\tau f\rangle_e = \frac{f(\mathbf{v_2}) - f(\mathbf{v_1})}{L_e}$$

The Vector Laplacian

$$\nabla^2 (\mathbf{v} \cdot \mathbf{n}) = \mathrm{grad}_n \underbrace{[\nabla \cdot \mathbf{v}]}_{\text{divergence}} - \mathrm{grad}_\tau \underbrace{[\nabla \times \mathbf{v}]}_{\text{curl}}$$

- Divergence is located on cells
- Curl is located on vertices

→ Perfect fit to compute vector Laplacian of normal component of velocity on edges

Geometrical Factors

- Just like for the computation of the curl and gradient, we again need geometrical factors to compute the directional gradients
- We need to make sure that the meshing library always returns the cell neighbor $\mathbf{c}_1$ first ("in the direction of the normal"), and $\mathbf{c}_2$ second ("in the opposite direction of the normal")
- The same argument goes for the tangential gradient

→ for the gradient along the **normal** the mesh has the correct property, **no factor needed**
→ for the gradient along the **tangent** a factor called `tangent_orientation` is given

Exercise

- For the exercise, you are about to re-use the divergence and curl operators you already implemented in the previous exercise
- The vector field is again made up of the spherical harmonics:

Field:

$$\begin{cases} u(x,y) := & \frac{1}{4}\sqrt{\frac{105}{2\pi}} \cos 2x \, \cos^2 y \, \sin y \quad , \\ v(x,y) := & \frac{1}{2}\sqrt{\frac{15}{2\pi}} \cos x \, \cos y \, \sin y \quad . \end{cases}$$

Vector Laplacian:

$$\nabla^2 \mathbf{v} = \begin{bmatrix} -\left(\sqrt{\frac{105}{2\pi}}\right) \cos(2 \cdot x) \cos(y) \cos(y) \sin(y) \\ \left(-2 \cdot \sqrt{\frac{15}{2\pi}}\right) \cos(x) \sin(y) \cos(y) \end{bmatrix}$$

Variable Reference

An overview over all variables is given below. The ones in bold are the ones you're supposed to be writing to. The others can be treated as read only.

u: Field[Edge], v: Field[Edge]	vector field / spherical harmonics on edges
nx: Field[Edge], ny: Field[Edge]	cell normals on edges (x and y component)
L: Field[Edge]	edge lengths
A: Field[Cell]	cell areas
uv_div: Field[Cell]	divergence of vector field /spherical harmonics on cells
uv_curl: Field[Vertex]	curl of vector field / spherical harmonics on vertices
grad_of_curl: Field[Edge]	tangential gradient of curl
grad_of_div: Field[Edge]	normal gradient of divergence
uv_nabla_2: Field[Edge]	normal component of vector gradient of $uv \cdot n$
L: Field[Edge]	edge length
dualL: Field[Edge]	dual edge length $\hat{L}_e$ (c.f. slide 6)
A: Field[Cell]	cell area
dualA: Field[Vertex]	dual cell area $\hat{A}_N$ (c.f. slide 6)

...

Variable Reference

An overview over all variables is given below. The ones in bold are the ones you're supposed to be writing to. The others can be treated as read only.

`tangent_orientation: Field[Edge]`	field indicating which tangential gradients need to be flipped
`edge_orientation_vertex: Field[Vertex>Edge]`	sparse dimension that indicates which normals need to be flipped for curl computation
`edge_orientation_cell: Field[Cell>Edge]`	sparse dimension that indicates which normals need to be flipped for gradient / div computation (+1/-1)

Hints

- A few text hints are given on the next slide
- A skeleton of the solution is given on the last slide
- Please use it only if you're seriously stuck

Hints

- The gradients are weighted reductions
- You only need to add three additional lines to the ops you have written before

Hints

```
with levels_upward as k:
    # compute curl (on vertices)
    uv_curl =
    # compute divergence (on cells)
    uv_div =
    # first term of of nabla2 (gradient of curl)
    grad_of_curl =
    # second term of of nabla2 (gradient of divergence)
    grad_of_div =
    # finalize nabla2 (difference between the two gradients)
    uv_nabla2 =
```