Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

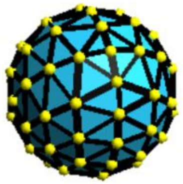
Swiss Confederation

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

dusk & dawn - Diffusion Exercise

DSL Training Workshop

Exercises Powered By



atlas
https://github.com/ecmwf/atlas

Contents

- For this exercise, you will implement a simple diffusion solver using dusk & dawn
- You want to use **Finite Differences**, but your boss gave you a **Finite Volume** mesh
- You can work around this using the **Neighbor Chain** concept of dusk & dawn!

- Please note that the method we are going to implement is going to be of low order and dissipative (simple averaging for interpolation)! It is not at all intended to highlight the state of the art in solving diffusion equations etc.

Diffusion / Heat Equation

The evolution of a Temperature Field T in a homogeneous material with thermal diffusivity κ is defined by the Heat equation:

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 (T)$$

- T: Temperature [°K]
- t: Time [s]
- κ: Thermal Diffusivity [m^2/s]

MeteoSwiss

Diffusion / Heat Equation

The evolution of a Temperature Field T in a homogeneous material with thermal diffusivity κ is defined by the Heat equation:

spatial derivative → FD/FVM stencil

$$\underbrace{\frac{\partial T}{\partial t}}_{\text{temporal derivative} \rightarrow \text{time stepper}} = \kappa \overbrace{\nabla^2}^{\text{spatial derivative} \rightarrow \text{FD/FVM stencil}}(T)$$

- T: Temperature [°K]
- t: Time [s]
- κ: Thermal Diffusivity [m^2/s]

temporal derivative → time stepper

Discretization of the Laplacian

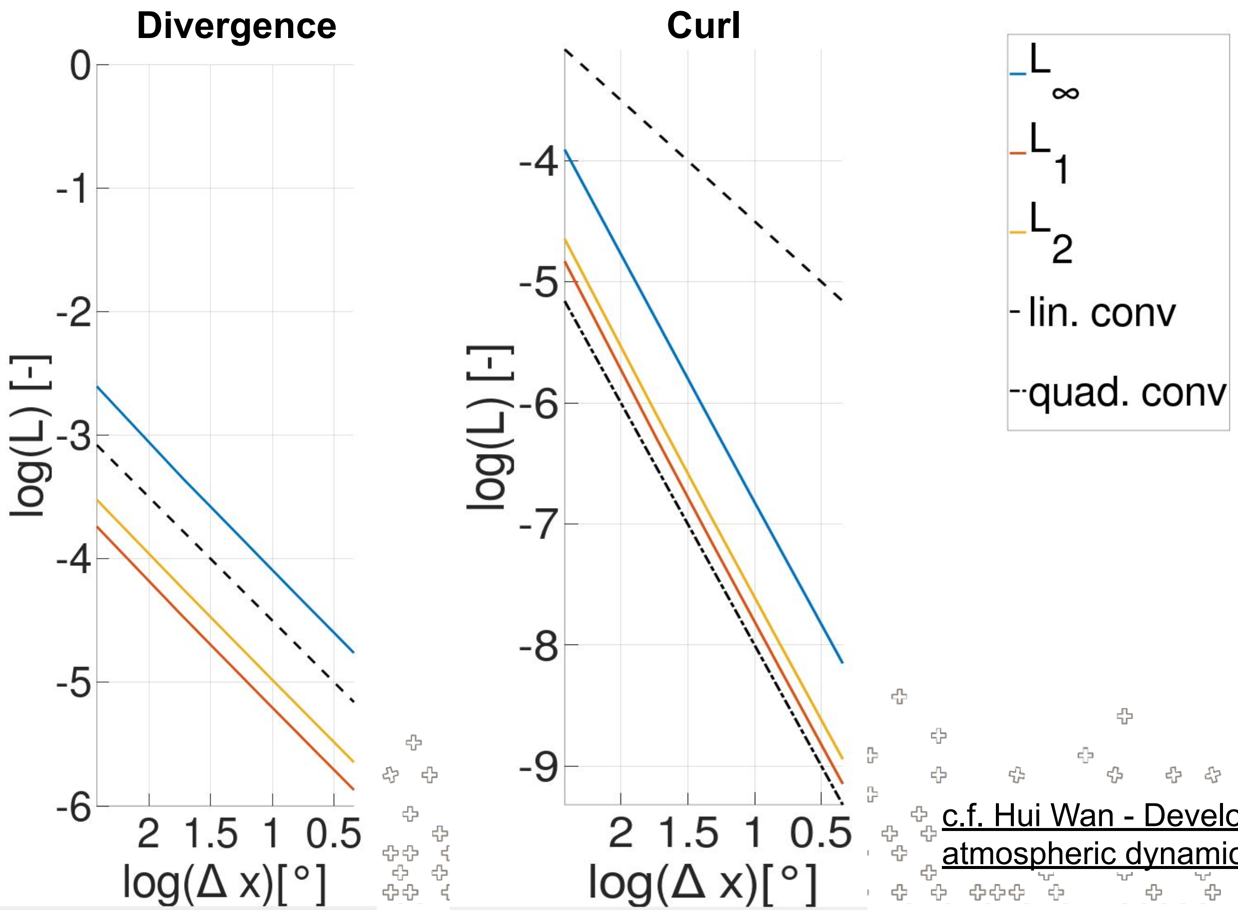
- We have seen how to compute the **vector** Laplacian on an FVM mesh in the last exercise
- For Cartesian coordinates, there is a simple connection between Laplacian and vector Laplacian

$$\nabla^2 \mathbf{v} = \nabla^2 \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} \\ \frac{\partial^2 v}{\partial^2 x} + \frac{\partial^2 v}{\partial^2 y} \end{bmatrix}$$

- It turns out, the vector laplacian is just the laplacian of each of the vectors components
- However, the procedure outlined in the last exercise is not appropriate!

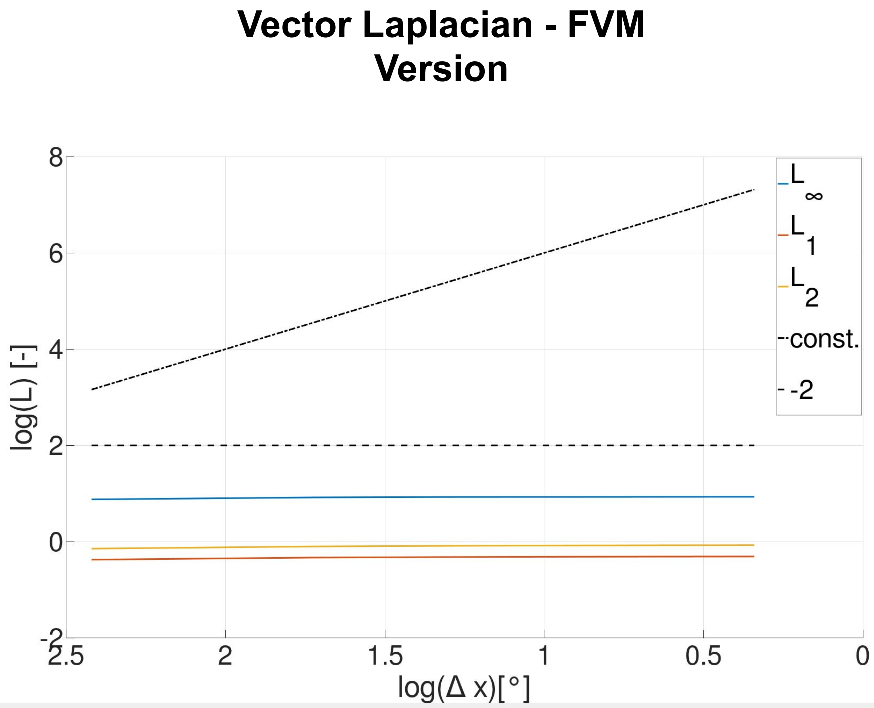
MeteoSwiss

Discretization of the Laplacian - Convergence



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

Discretization of the Laplacian - Convergence



MeteoSwiss

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

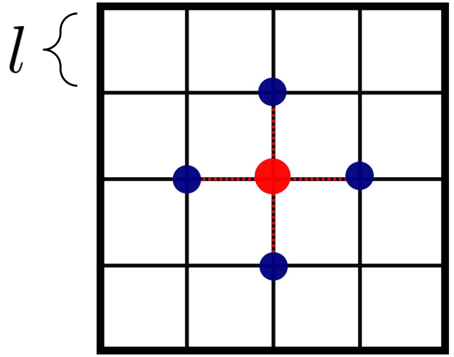
Discretization of the Laplacian - Convergence

- Error of the **FVM vector Laplacian does not converge**
- We have to use another formulation
- We know that the central **Finite Difference** approximation to the Laplacian **has second order**

Finite Differences

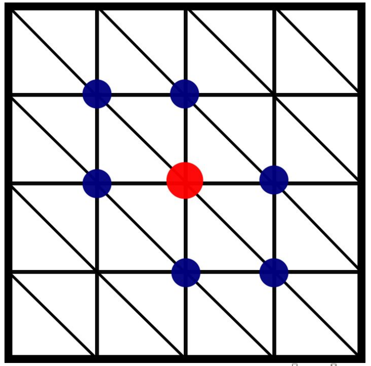
- Classic Finite Differences:

$$\langle \nabla^2(T) \rangle = \frac{T_{n2} + T_{n1} + T_{n3} + T_{n4} - 4T_{n0}}{l^2}$$



- Not immediately clear how to perform on a (triangular) FVM mesh

$$\langle \nabla^2(T) \rangle = ?$$



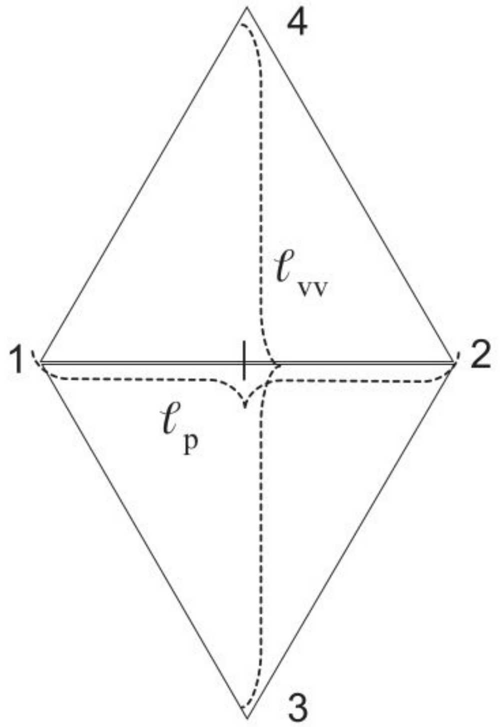
MeteoSwiss

Finite Differences on FVM Meshes

- By using the Neighbor Chain concept you can access the Vertex neighbors indicated with integers 1-4 on the right for each edge **e**

- This allows you to define a standard central Finite Difference stencil on a triangular mesh:

$$\langle \nabla^2(T) \rangle = \frac{T_{n2} + T_{n1} - 2T_e}{\left(\frac{1}{2}l_p\right)^2} + \frac{T_{n4} + T_{n3} - 2T_e}{\left(\frac{1}{2}l_{vv}\right)^2}$$



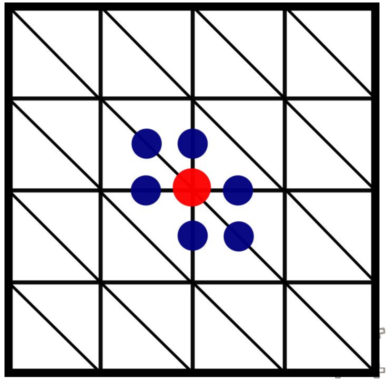
Zängl et. al https://doi.org/10.1002/qj.2378

Interpolation

- In order for the method on the last slide to work, the temperature field needs to be defined on the vertices as well as the edges
- Due to the nature of the proposed reduction, evolution in time is only possible on edges (since only there ∇^2T can be computed)

→Temperature needs to be interpolated from edges to vertices

$$\langle T^v \rangle = \frac{1}{|\mathcal{N}(v)|} \sum_{j=1}^{\mathcal{N}(v)} T_j^e$$



MeteoSwiss

Time Stepping

For this Exercise a very simple predictor / corrector time stepper is used:

$$T^* = T^{n-1} + 1/2 \cdot \Delta t \cdot T_t^{n-1}$$

$$T_t^n = f(T^*) \longleftarrow \kappa \cdot \langle \nabla^2 (T_e) \rangle$$

$$T^{n+1} = T^{n-1} + \Delta t \cdot T_t^n$$

MeteoSwiss

Boundary Conditions

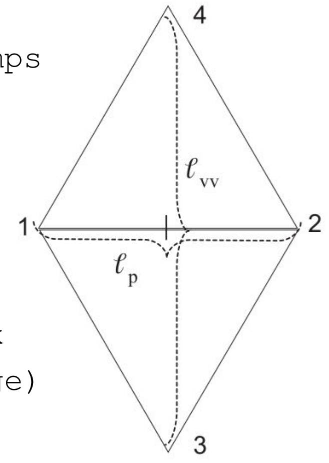
Boundary Conditions are as simple as possible: We just state that no heat may leave the domain (perfect isolation):

$$\langle \nabla^2 (T_e) \rangle = 0 \qquad | e \in \mathcal{B}e$$

Variable Reference

An overview over all variables is given below. The ones in bold are out or "in-out" (both written to and being read from) variables. The others can be treated as read only.

TV: Field[Vertex]	Temperature on Vertices
TE: Field[Edge],	Temperature on Edges
TEinit: Field[Edge],	Field to store initial value of Edge Temps for time stepper
TE_t: Field[Edge],	Temporal derivative of temperature
TEnabla2: Field[Edge],	∇T on Edges
inv_primal_edge_length: Field[Edge],	$1/(0.5 * l_p)$ in figure
inv_vert_vert_length: Field[Edge],	$1/(0.5 * l_{vv})$ in figure
nnbhV: Field[Vertex],	number of edge neighbors for each vertex
boundary_edge: Field[Edge],	boundary edge mask (true if boundary edge)
kappa: Field[Edge],	thermal diffusitivity constant
dt: Field[Edge]	time step



MeteoSwiss

Hints

- Hints are on the next slides
- Please consider them only if you're seriously stuck
- Some general hints are on the next slides, on the slides after that parts of the solution are revealed

MeteoSwiss

Hints

- To compute the Laplacian use sum_overThe geometric quantities are required to compute the correct weights for the above operation
- Computing the laplacian requires two operations: the reduction over the neighbors as well as subtracting the central contribution

Final Algorithm

1. Save old state of temperature (TEinit = …)
2. Execute Predictor step (TE = TEinit + …)
3. Interpolate Temperature from Edges to Nodes (TV = sum_over(....))
4. Compute ∇^2T (TEnabla2 = …)
5. Enforce Boundary Conditions (if (boundary_edge): ….)
6. Build right hand side / f() (TE_t = ….)
7. Execute Corrector Step (TE = TEinit + ….)

MeteoSwiss

Skeleton

```
with levels_upward:
    # initialize
    TEinit = ...
    # predict
    TE = ...
    # interpolate temperature from edges to vertices
    TV = ...
    # compute nabla2 using the finite differences
    # build ODEs
    if (boundary_edge):
        TE_t = 0.
    else:
        TE_t = ...
```

MeteoSwiss