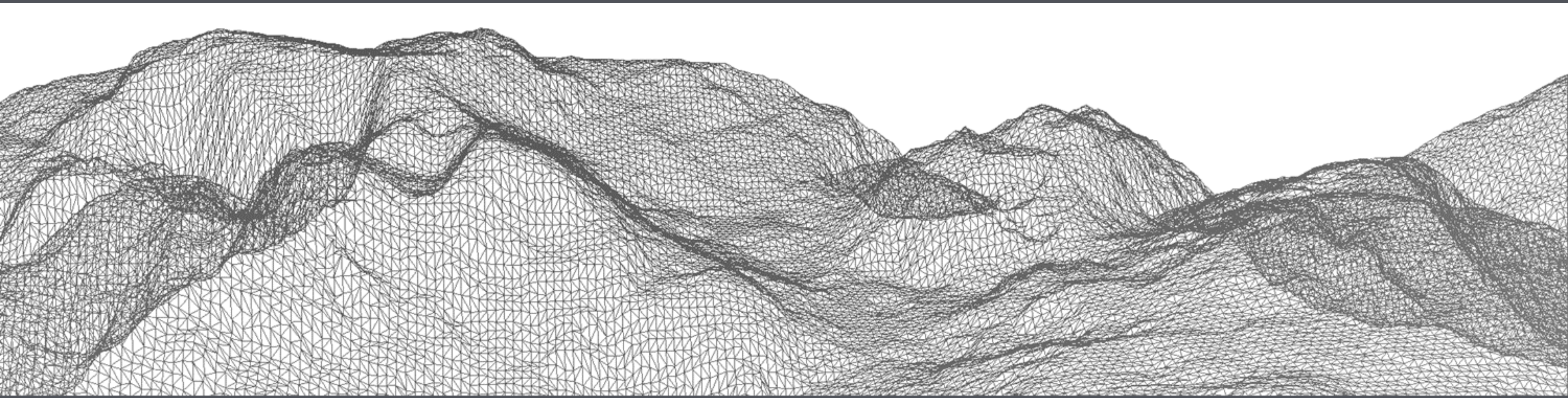
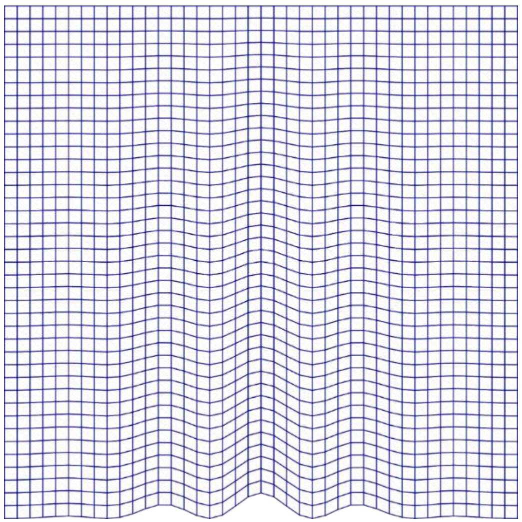
ADVECTION OVER STEEP SLOPES

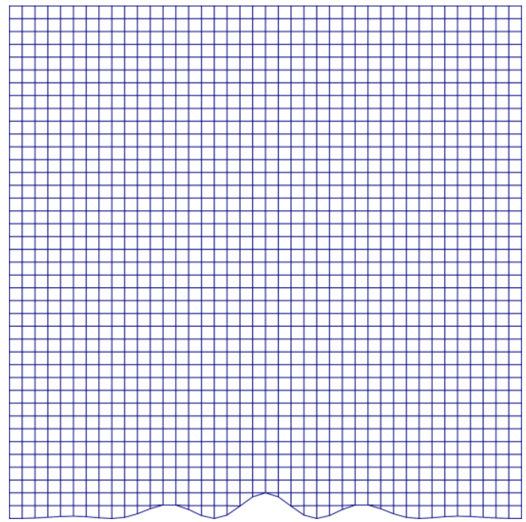


James Shaw @hertzsprrrung
Hilary Weller @hilaryweller0
John Methven
Terry Davies

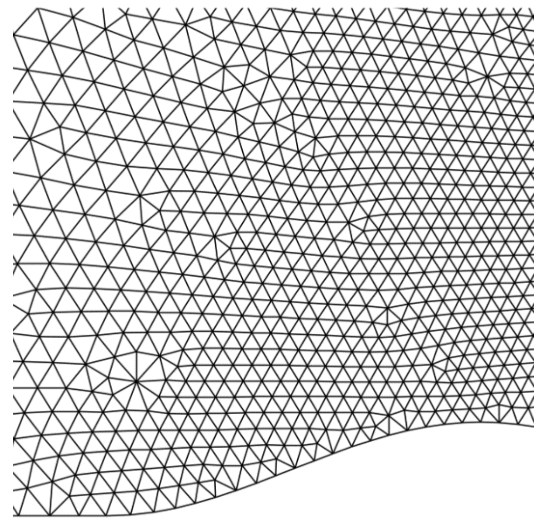
WAYS TO REPRESENT TERRAIN



Terrain following layers



Cut cells

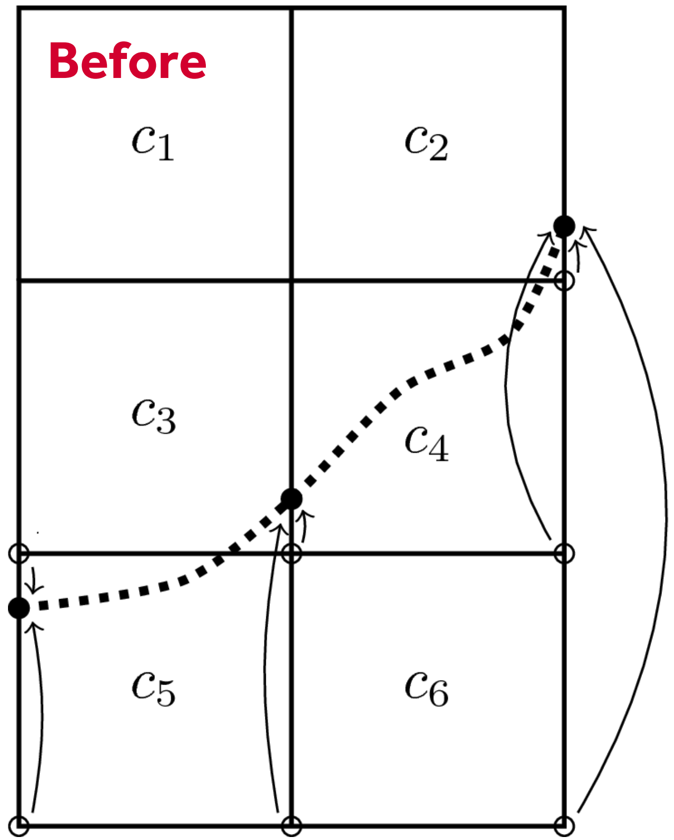


Unstructured

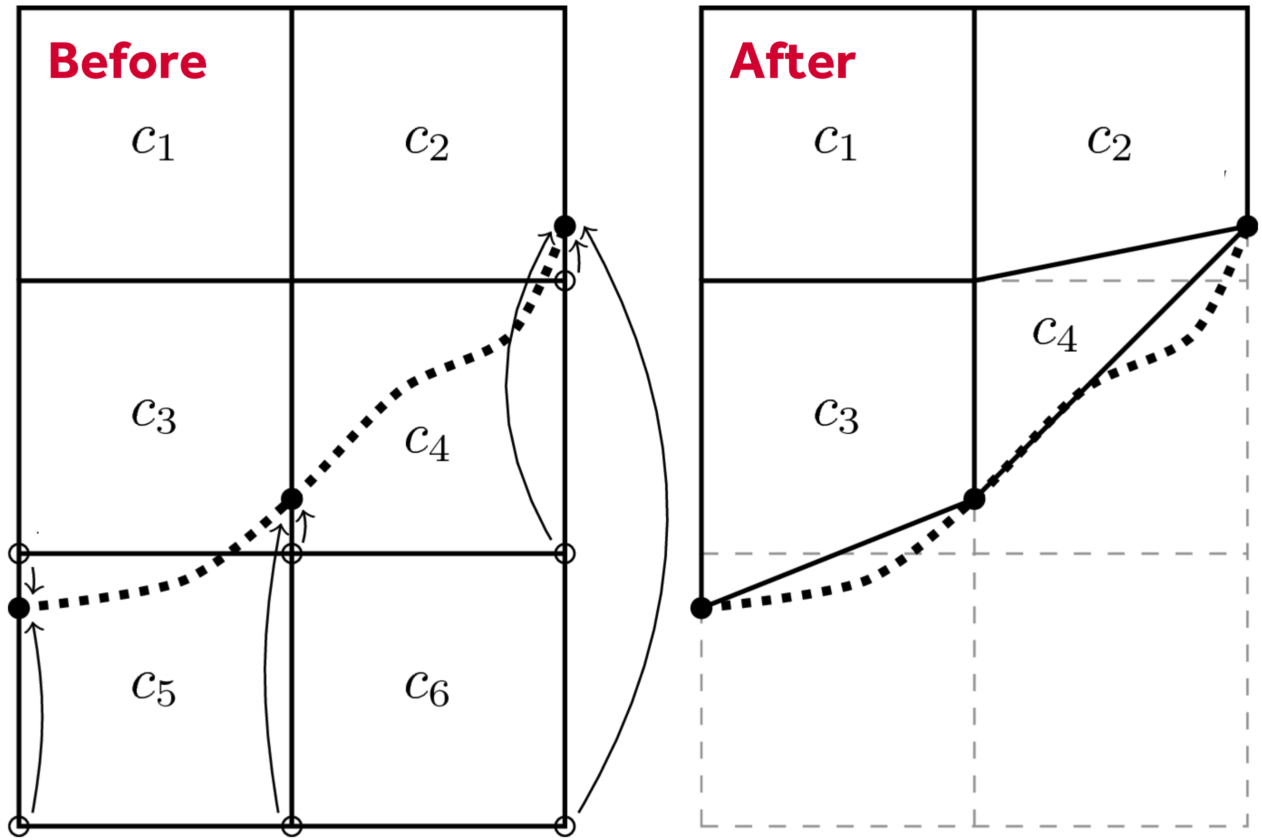
Source: Smolarkiewicz & Szmelter

http://ral.ucar.edu/hap/events/orographic-precip/images/2wed/am/day2-Wed_am_3-Orogunmesh2.ppt

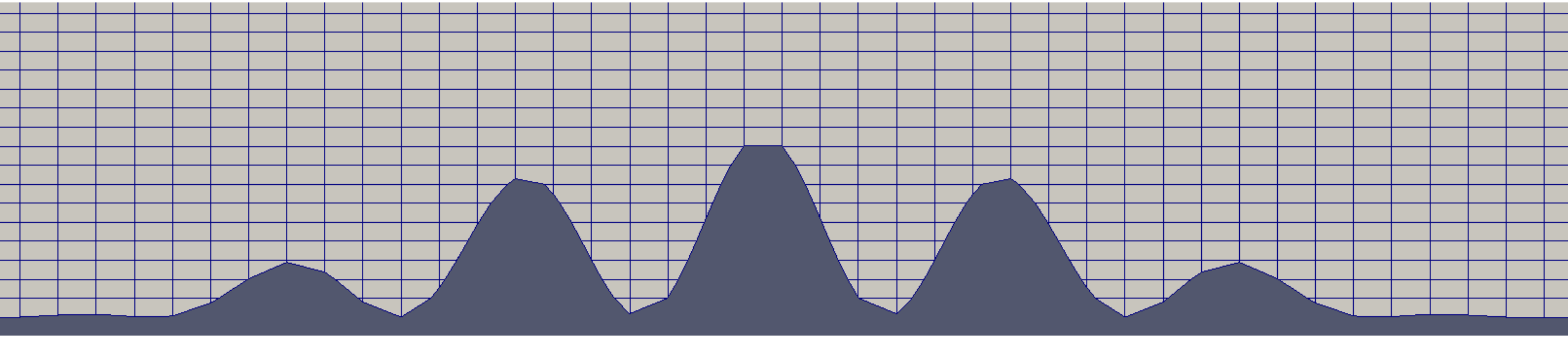
SLANTED CELLS



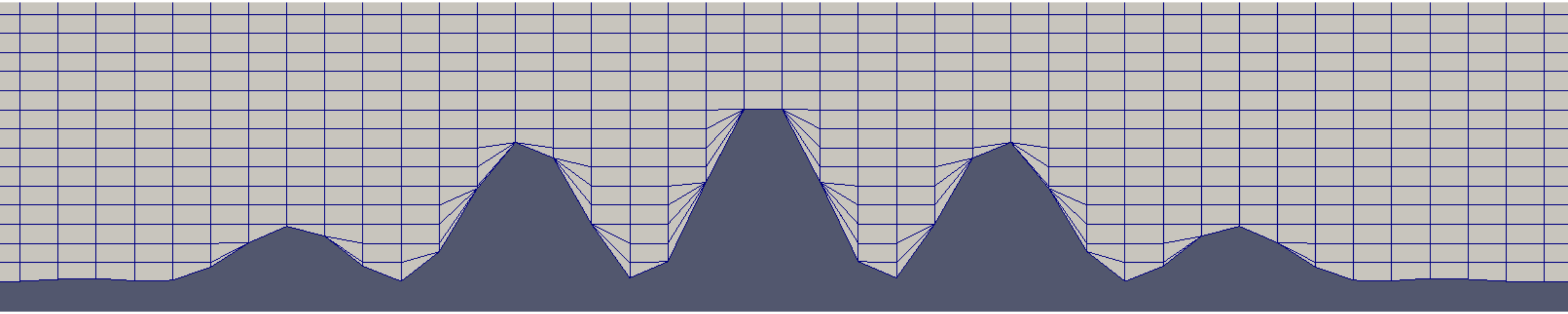
SLANTED CELLS



CUT CELLS



SLANTED CELLS



Source: Shaw & Weller 2016, MWR, dx.doi.org/10.1175/MWR-D-15-0226.1

SLANTED CELLS

- Easy to construct
- Avoid arbitrarily small cells
- Generalise to 3D with arbitrary horizontal meshes

CUBICFIT: AN ADVECTION SCHEME FOR STEEP SLOPES

CUBICFIT: AN ADVECTION SCHEME FOR STEEP SLOPES

- Finite volume
- Eulerian
- Multidimensional cubic approximation
- Method-of-lines with Runge-Kutta timestepping

- No flux correction
- Not monotonic

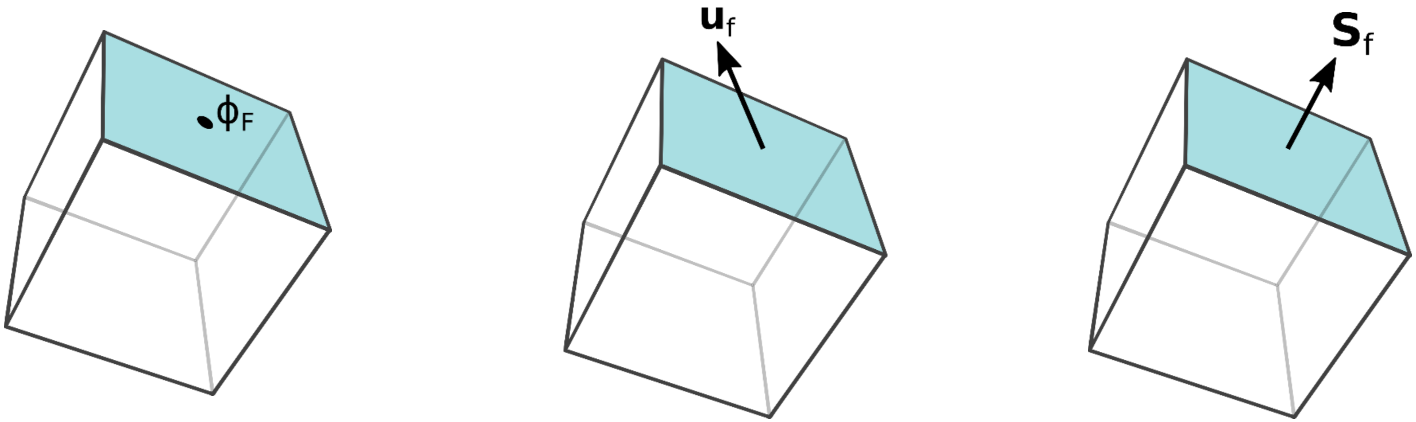
FINITE VOLUME DISCRETISATION

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\boldsymbol{u}\phi) \qquad = 0$$

FINITE VOLUME DISCRETISATION

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\boldsymbol{u}\phi) = 0$$

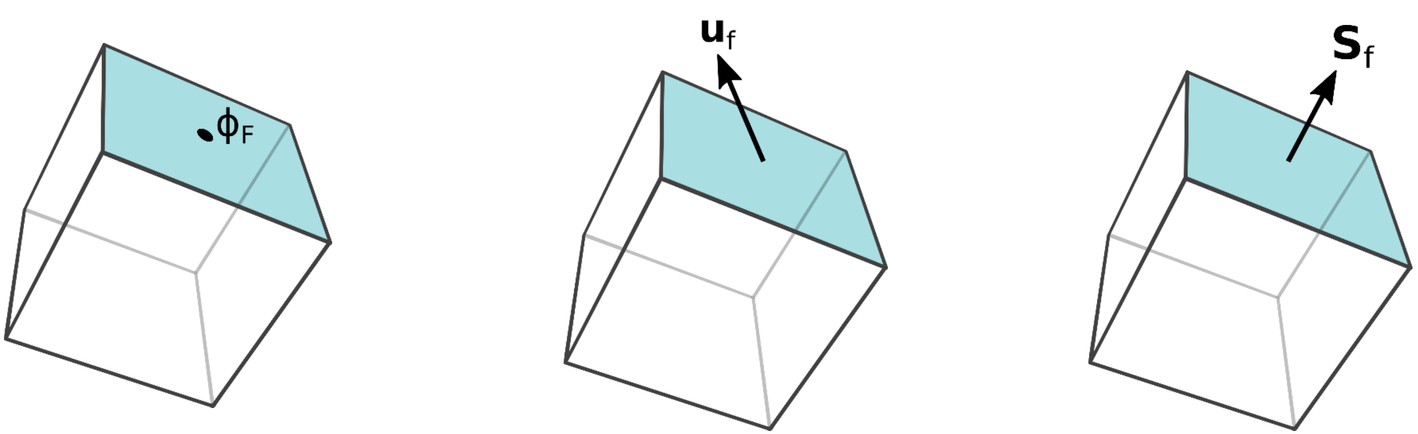
$$\frac{\partial \phi}{\partial t} + \frac{1}{V}\sum_{f} \phi_F \boldsymbol{u}_f \cdot \boldsymbol{S}_f = 0$$



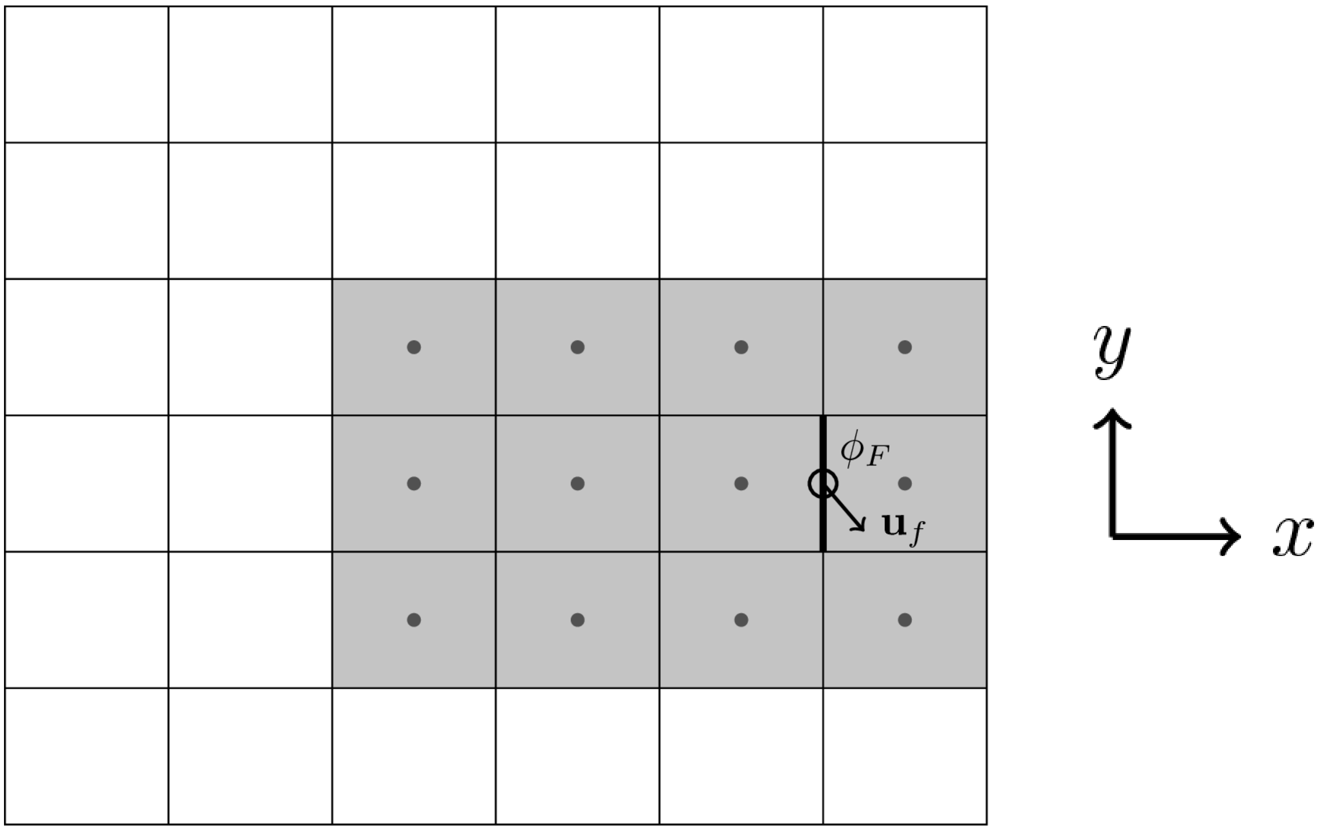
HOW TO ESTIMATE Φ_F?

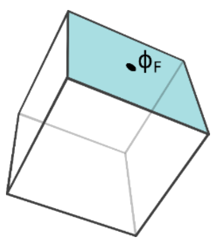
$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\boldsymbol{u}\phi) = 0$$

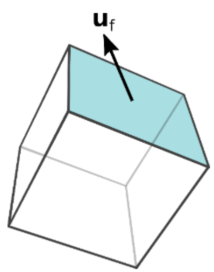
$$\frac{\partial \phi}{\partial t} + \frac{1}{V}\sum_f \phi_F \boldsymbol{u}_f \cdot \boldsymbol{S}_f = 0$$

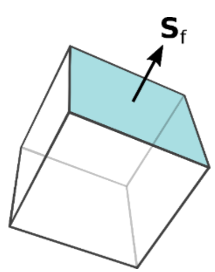


UPWIND-BIASED STENCIL

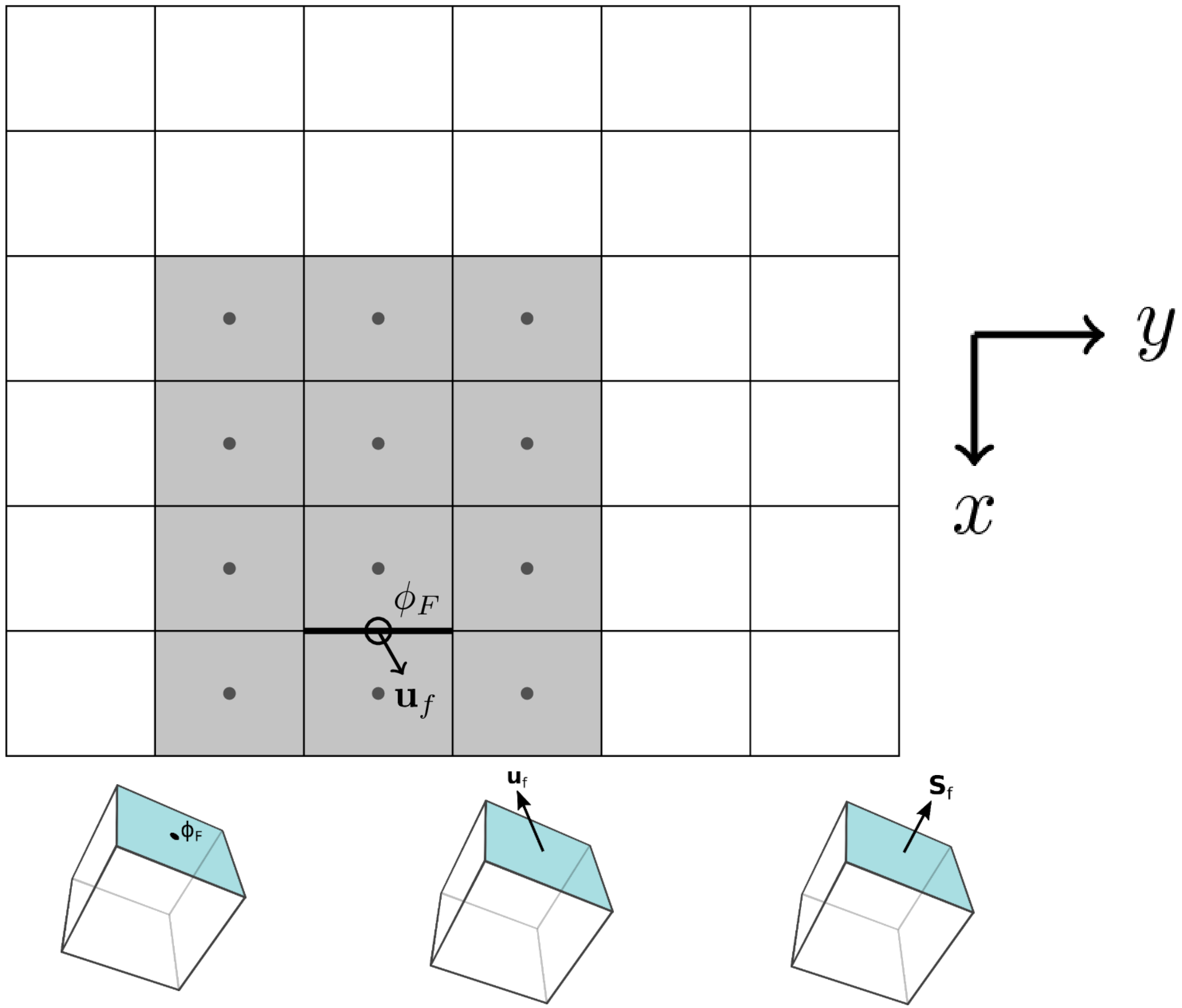




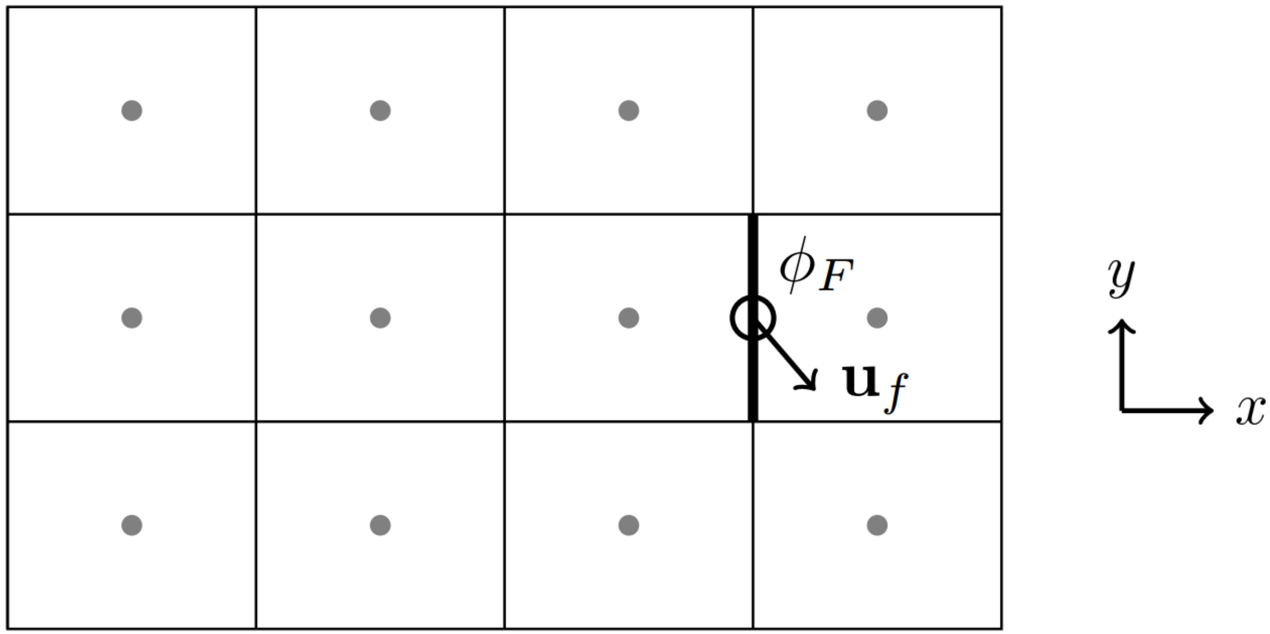




STENCIL-LOCAL COORDINATES



LEAST SQUARES FIT



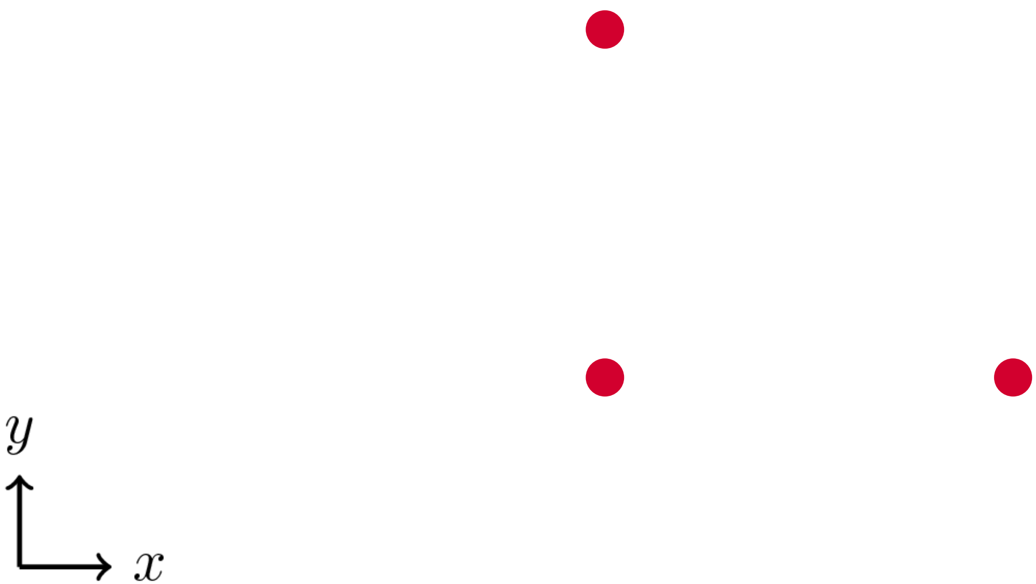
$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 xy^2$$

Φ_F IS CHEAP TO COMPUTE

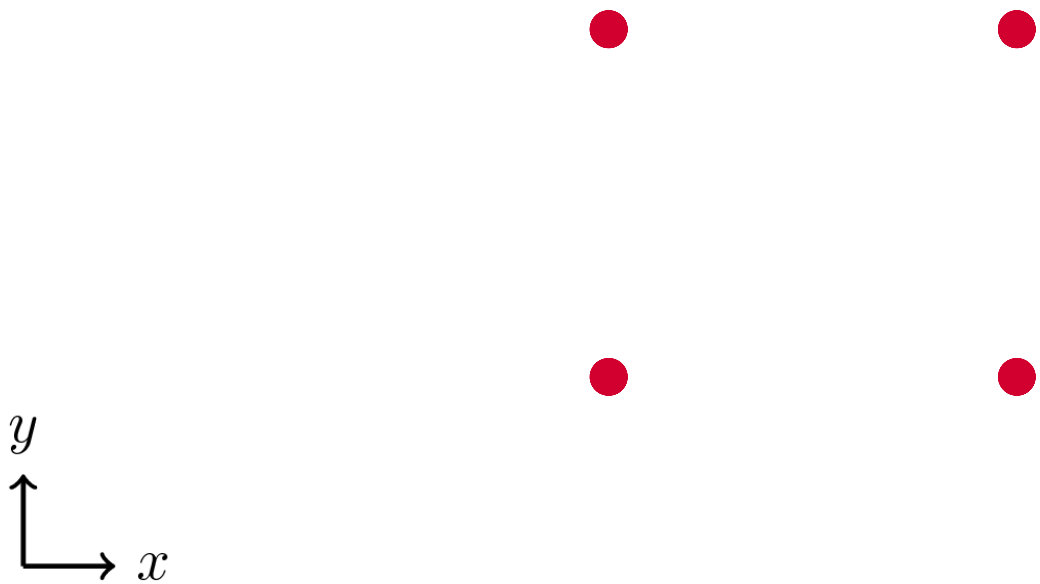
$$\phi_F = a_1 = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{12} \end{bmatrix} \cdot \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{12} \end{bmatrix}$$

$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 xy^2$$

ESTIMATING Φ_F NEAR BOUNDARIES

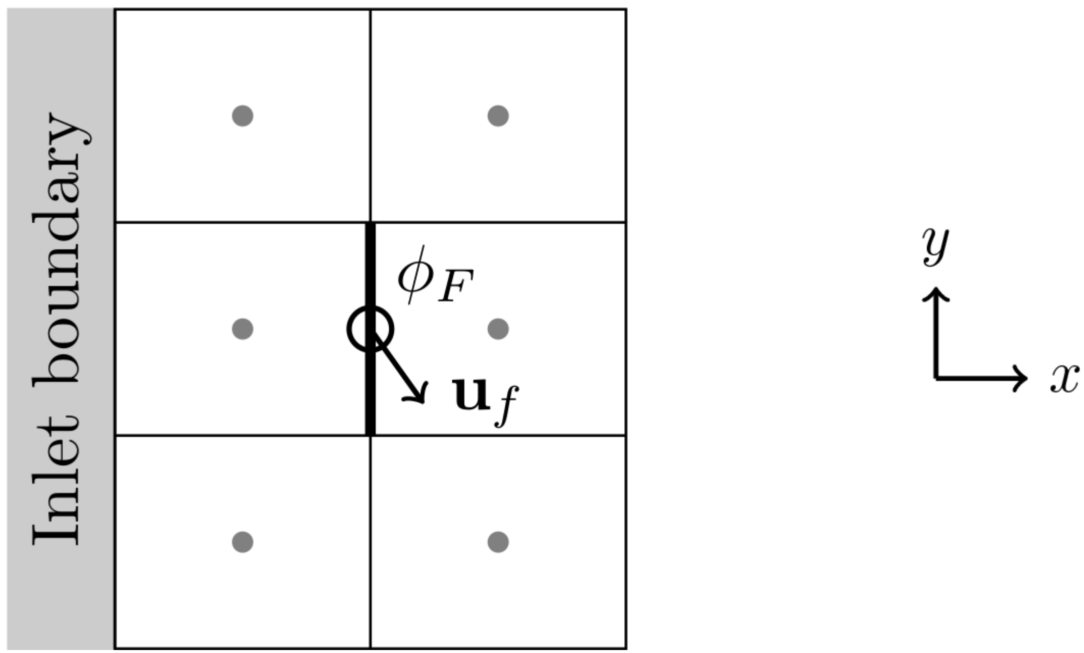


$$\phi = a_1 + a_2 x + a_3 y$$



$$\phi = a_1 + a_2 x + a_3 y + a_4 x^2$$

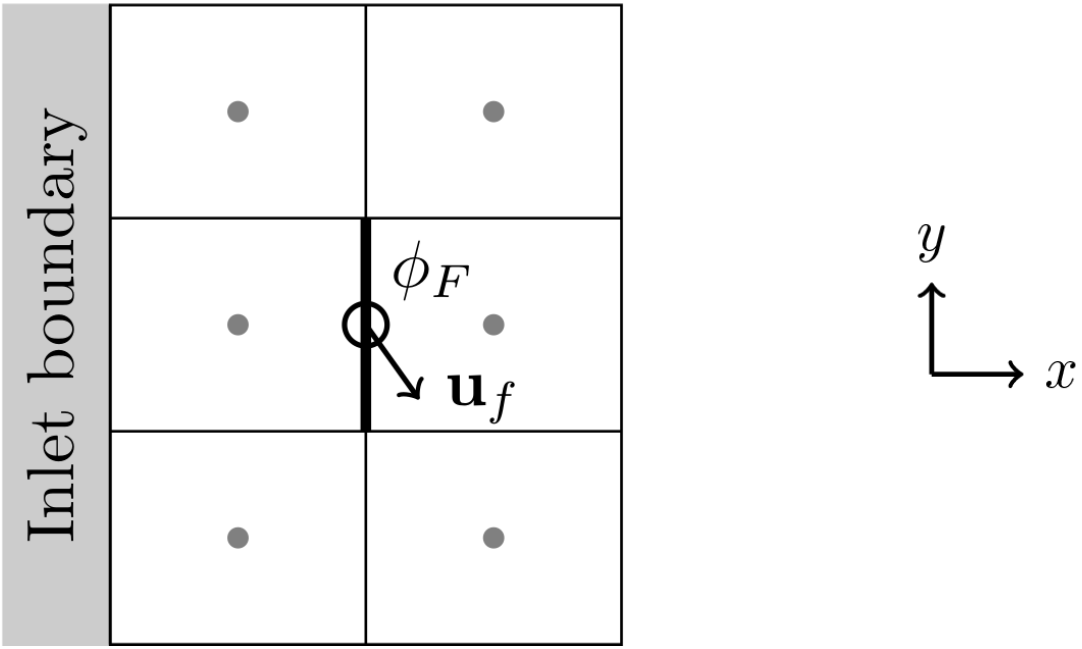
ESTIMATING Φ_F NEAR BOUNDARIES



$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 xy^2$$

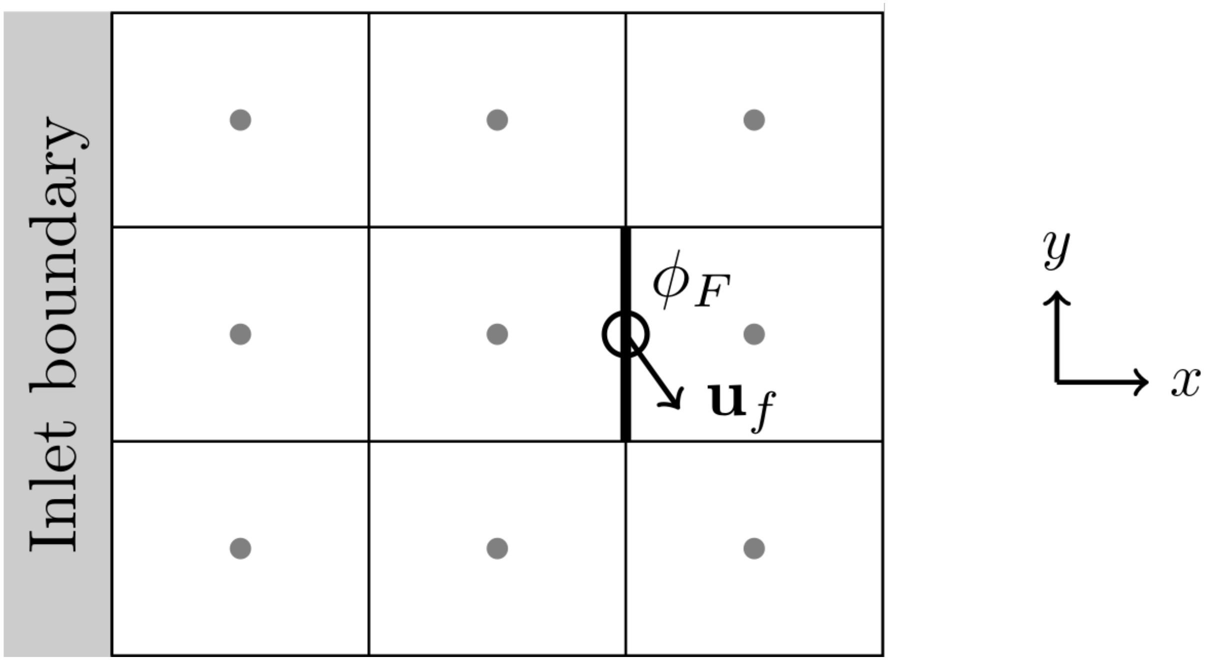
?

ESTIMATING Φ_F NEAR BOUNDARIES



$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 y^2 + a_6 xy^2$$

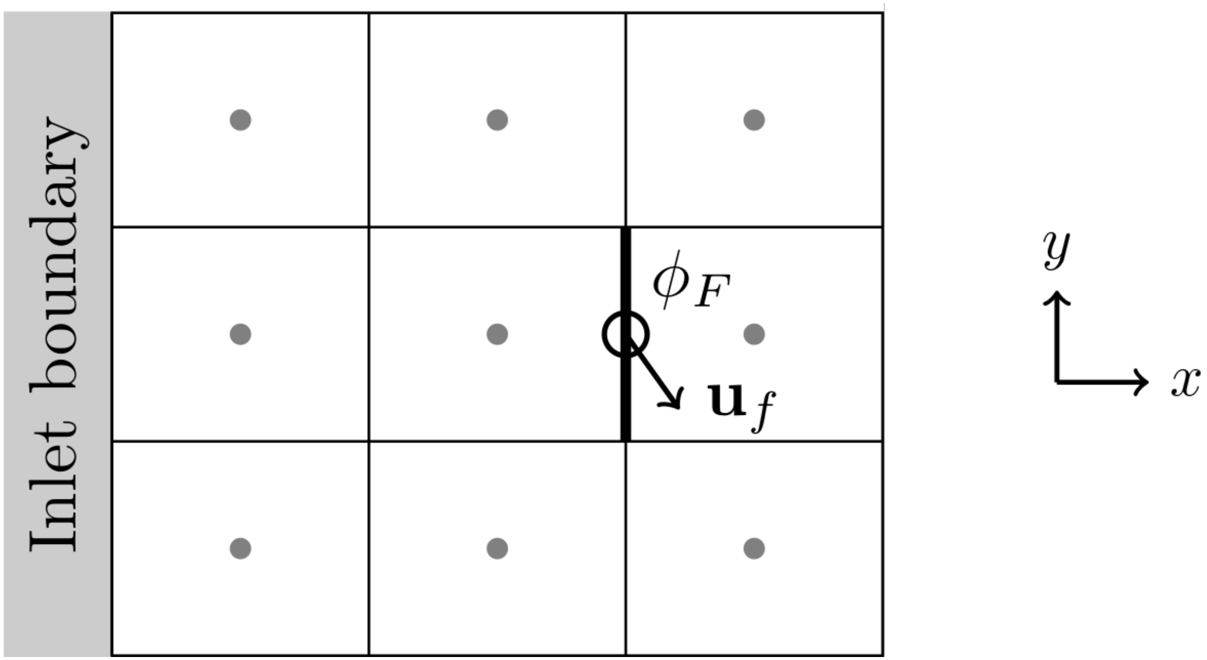
ESTIMATING Φ_F NEAR BOUNDARIES



$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 xy^2$$

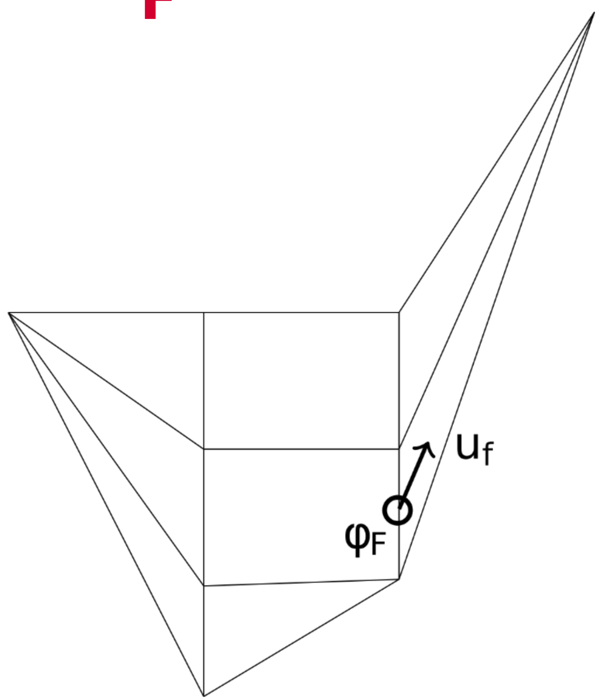
?

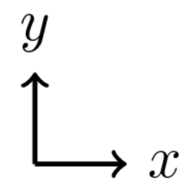
ESTIMATING Φ_F NEAR BOUNDARIES



$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 x y^2$$

ESTIMATING Φ_F NEAR BOUNDARIES





$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 xy^2$$

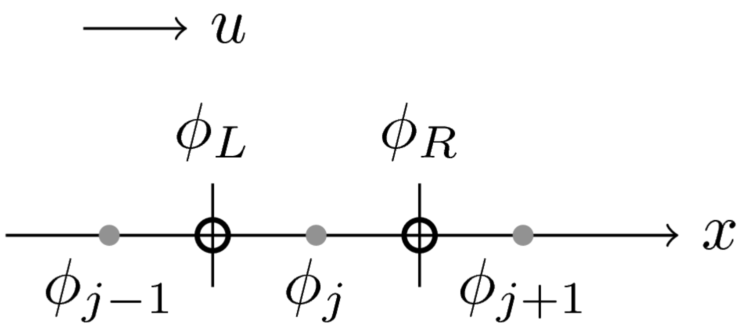
?

University of Reading

What is the most suitable polynomial for a given distribution of points?

What is the highest degree polynomial that ensures numerically stable advection?

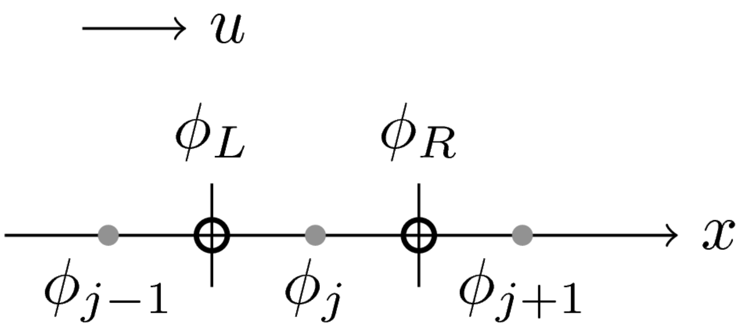
VON NEUMANN STABILITY



$$\frac{\partial \phi_j^{(n)}}{\partial t} = -u\frac{\phi_R - \phi_L}{\Delta x}$$

VON NEUMANN STABILITY

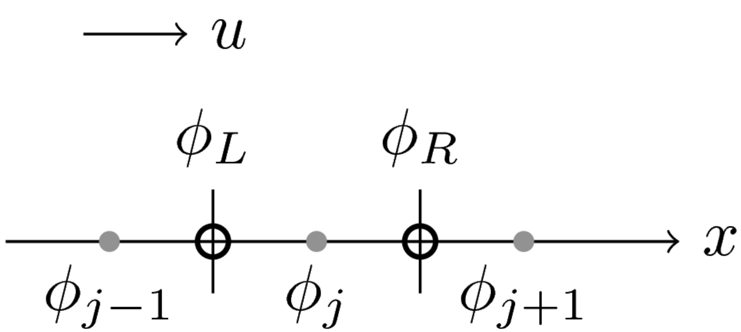
University of Reading



$$\frac{\partial \phi_j^{(n)}}{\partial t} = -u \frac{\phi_R - \phi_L}{\Delta x}$$

$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$
$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

VON NEUMANN STABILITY



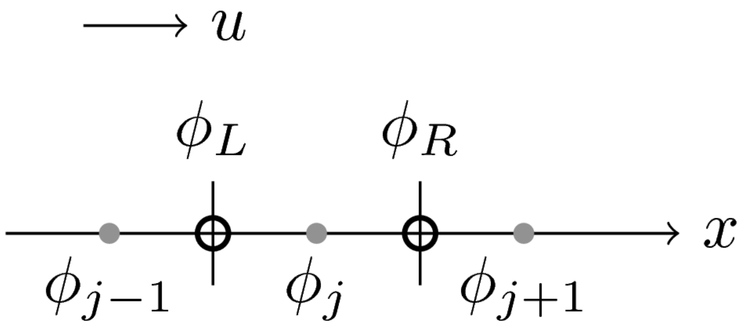
$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

- Assume perfect timestepping

VON NEUMANN STABILITY

University of Reading

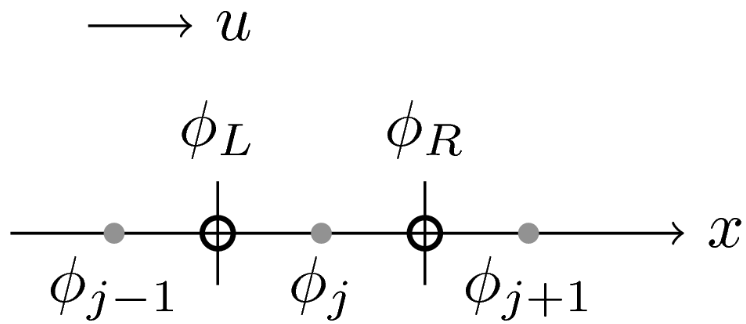


$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

- Assume perfect timestepping
- Assume wave-like solution $\phi_j^{(n)} = A^n e^{ijk\Delta x}$

VON NEUMANN STABILITY

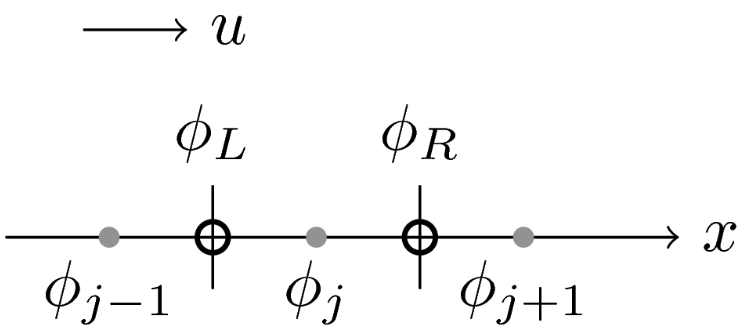


$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

- Assume perfect timestepping
- Assume wave-like solution $\phi_j^{(n)} = A^n e^{ijk\Delta x}$
- Introduce constraints:
  - $|A| <= 1$

VON NEUMANN STABILITY

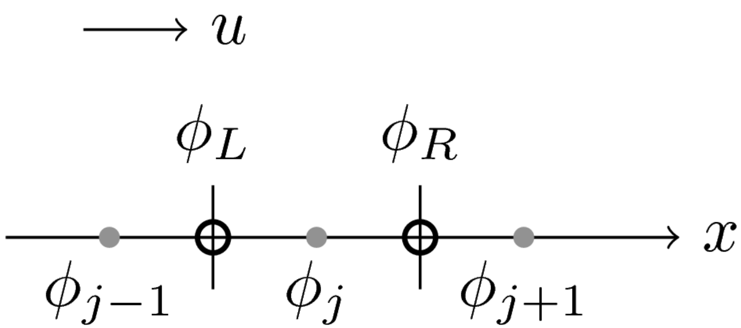


$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

- Assume perfect timestepping
- Assume wave-like solution $\phi_j^{(n)} = A^n e^{ijk\Delta x}$
- Introduce constraints:
  - $|A| <= 1$
  - arg(A) < 0 for Co > 0

VON NEUMANN STABILITY



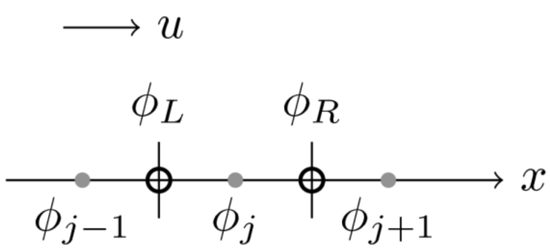
$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

- Assume perfect timestepping
- Assume wave-like solution $\phi_j^{(n)} = A^n e^{ijk\Delta x}$
- Introduce constraints:
  - $|A| <= 1$
  - arg(A) < 0 for Co > 0
  - No more damping than first-order upwind (w_u=1, w_d=0)

VON NEUMANN STABILITY

2-point approximation



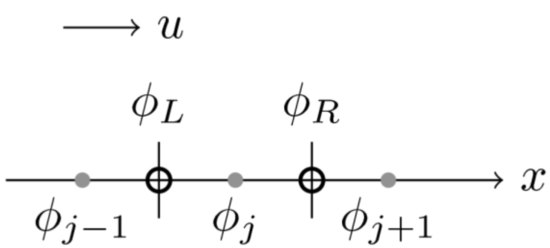
$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

VON NEUMANN STABILITY

University of Reading

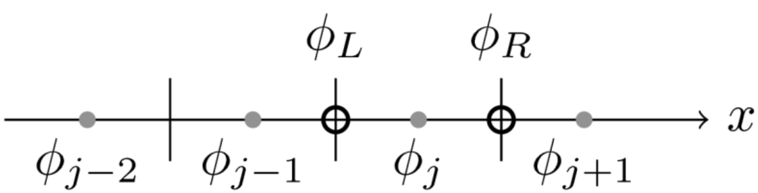
2-point approximation



$$\phi_L = w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_u \phi_j + w_d \phi_{j+1}$$

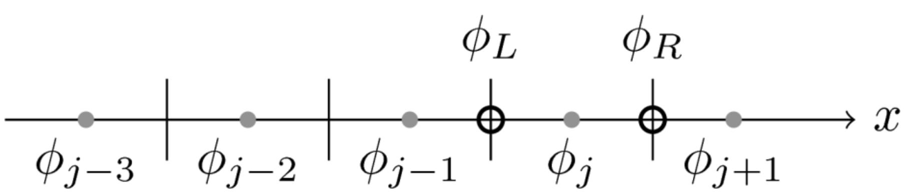
3-point approximation



$$\phi_L = w_{uu} \phi_{j-2} + w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_{uu} \phi_{j-1} + w_u \phi_j + w_d \phi_{j+1}$$

4-point approximation



$$\phi_L = w_{uuu} \phi_{j-3} + w_{uu} \phi_{j-2} + w_u \phi_{j-1} + w_d \phi_j$$

$$\phi_R = w_{uuu} \phi_{j-2} + w_{uu} \phi_{j-1} + w_u \phi_j + w_d \phi_{j+1}$$

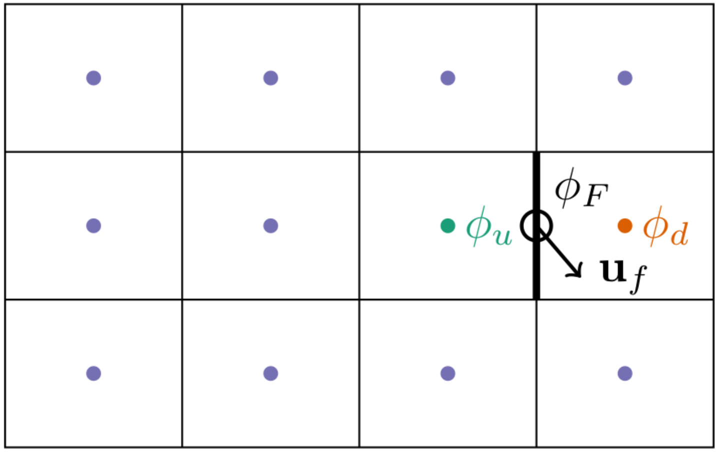
VON NEUMANN STABILITY

$$0.5 \leq w_u \leq 1$$

$$0 \leq w_d \leq 0.5$$

$$w_u - w_d \geq \max_{p \in P}(|w_p|)$$

$$\phi_F = \begin{bmatrix} w_u \\ w_d \\ w_3 \\ \vdots \\ w_{12} \end{bmatrix} \cdot \begin{bmatrix} \phi_u \\ \phi_d \\ \phi_3 \\ \vdots \\ \phi_{12} \end{bmatrix}$$



POLYNOMIAL FIT ALGORITHM

University of Reading

1. Generate candidate polynomials
2. Test each candidate against von Neumann stability criteria
3. Choose the best candidate that satisfies the criteria

$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 x y^2$$

$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y$$

$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x y^2$$

$$\phi = a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^2 y + a_8 x y^2$$

$$\vdots$$

$$\phi = a_1 + a_2 x + a_3 y$$

$$\phi = a_1 + a_2 x + a_3 x^2$$

$$\phi = a_1 + a_2 y + a_3 y^2$$

$$\phi = a_1 + a_2 x$$

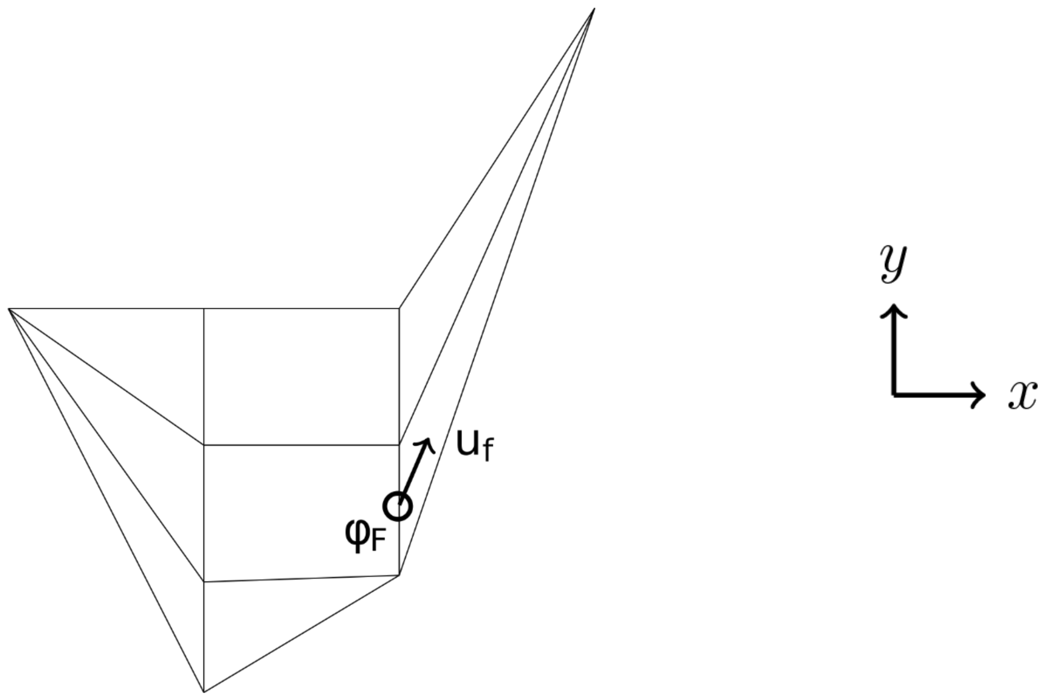
$$\phi = a_1 + a_2 y$$

$$0.5 \le w_u \le 1$$

$$0 \le w_d \le 0.5$$

$$w_u - w_d \ge \max_{p \in P}(|w_p|)$$

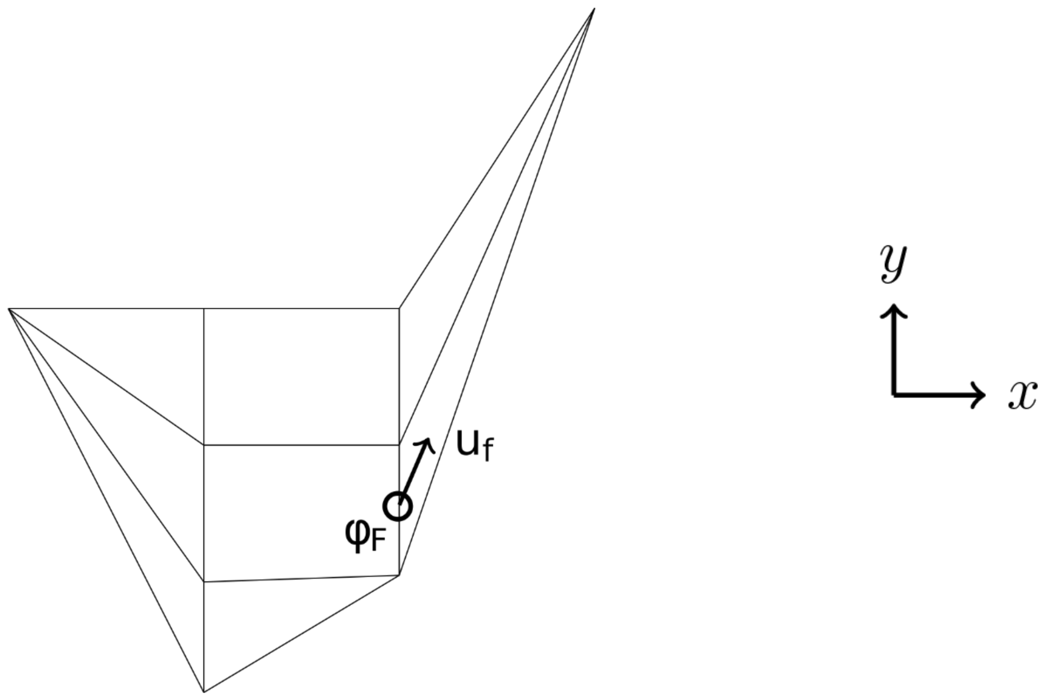
ESTIMATING Φ_F NEAR BOUNDARIES



$$\phi = a_1 + a_2 x + a_3 y + a_4 x y + a_5 x^2 + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 x y^2$$

?

ESTIMATING Φ_F NEAR BOUNDARIES



$$\phi = a_1 + a_2 x + a_3 y + a_4 x y + a_5 x^2$$

NUMERICAL EXPERIMENTS

1. Schär horizontal advection over orography
2. “Slug” advection over orography

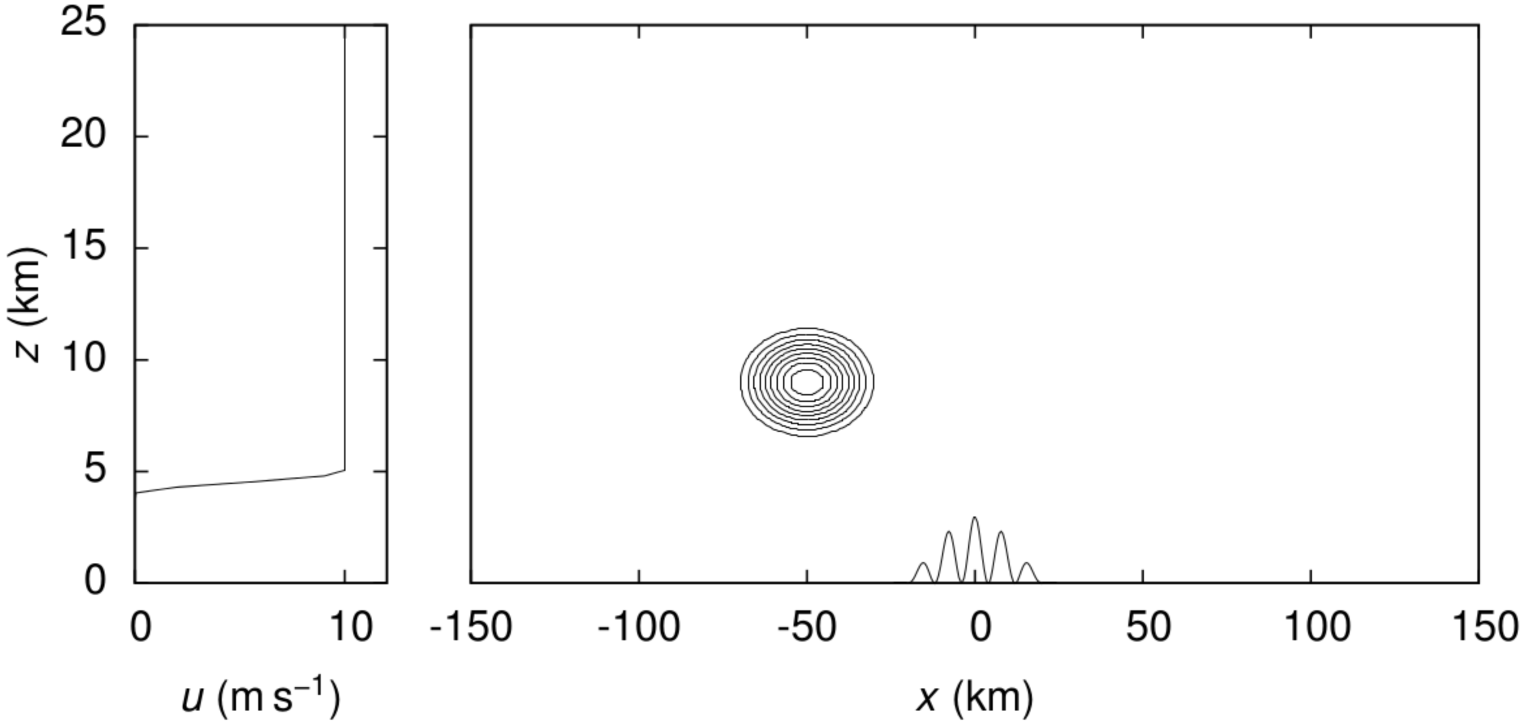
NUMERICAL EXPERIMENTS

1. Schär horizontal advection over orography
2. “Slug” advection over orography

Compare

- cubicFit
- linearUpwind

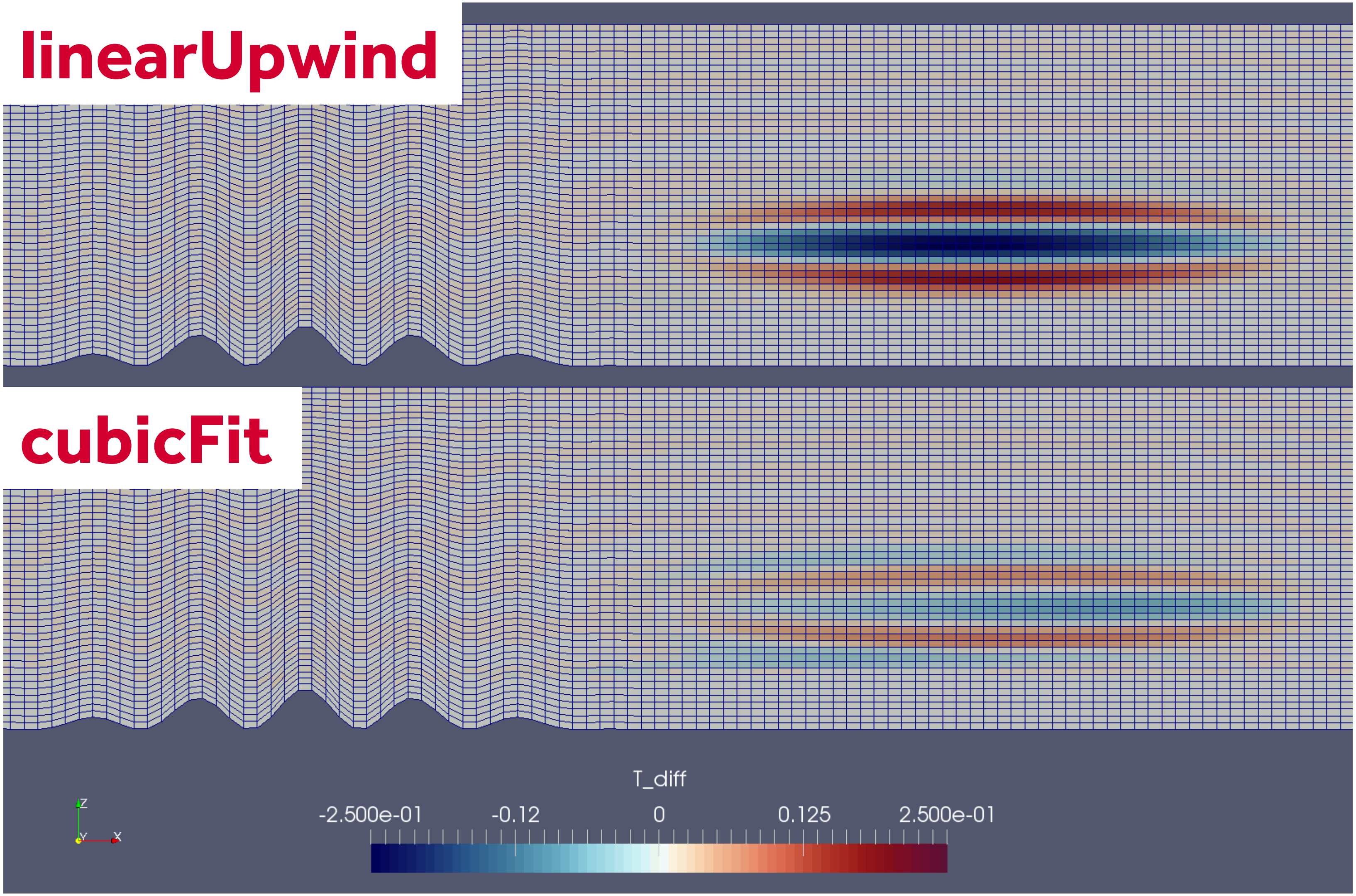
SCHÄR HORIZONTAL ADVECTION



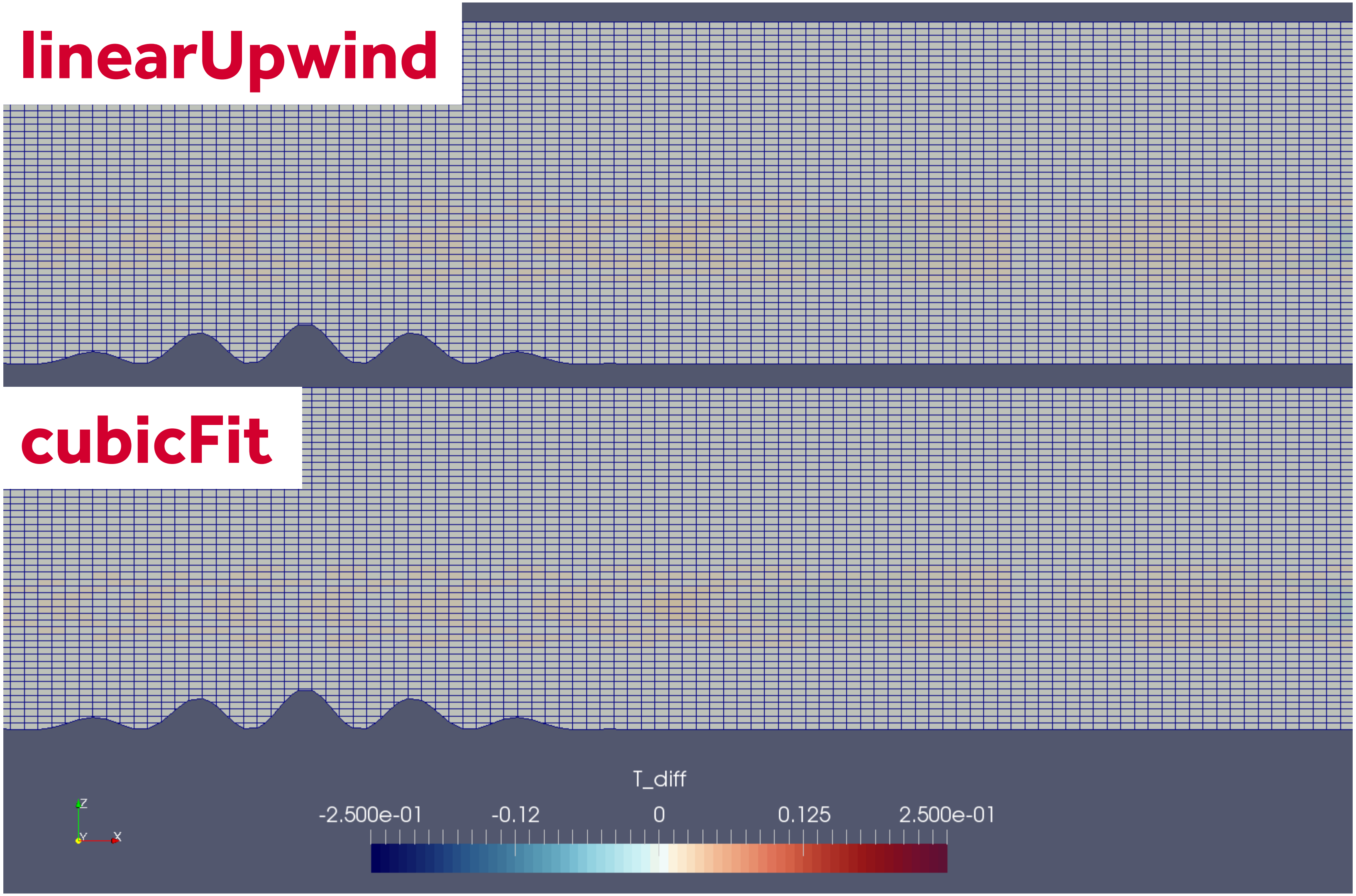
Horizontal wind profile, surface terrain profile and initial tracer

Adapted from Schär et al. 2002, MWR

BASIC TERRAIN FOLLOWING

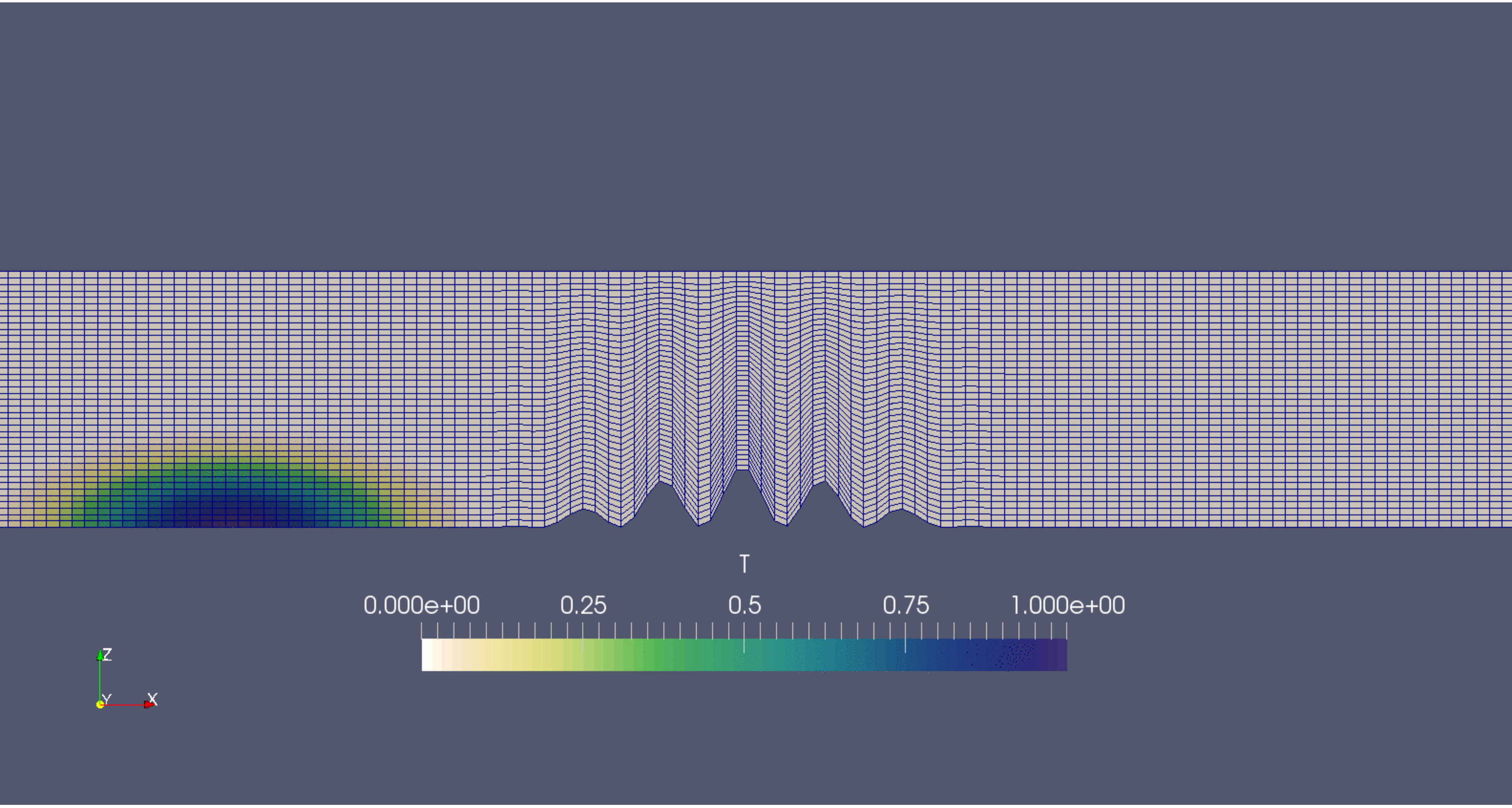


CUT CELLS

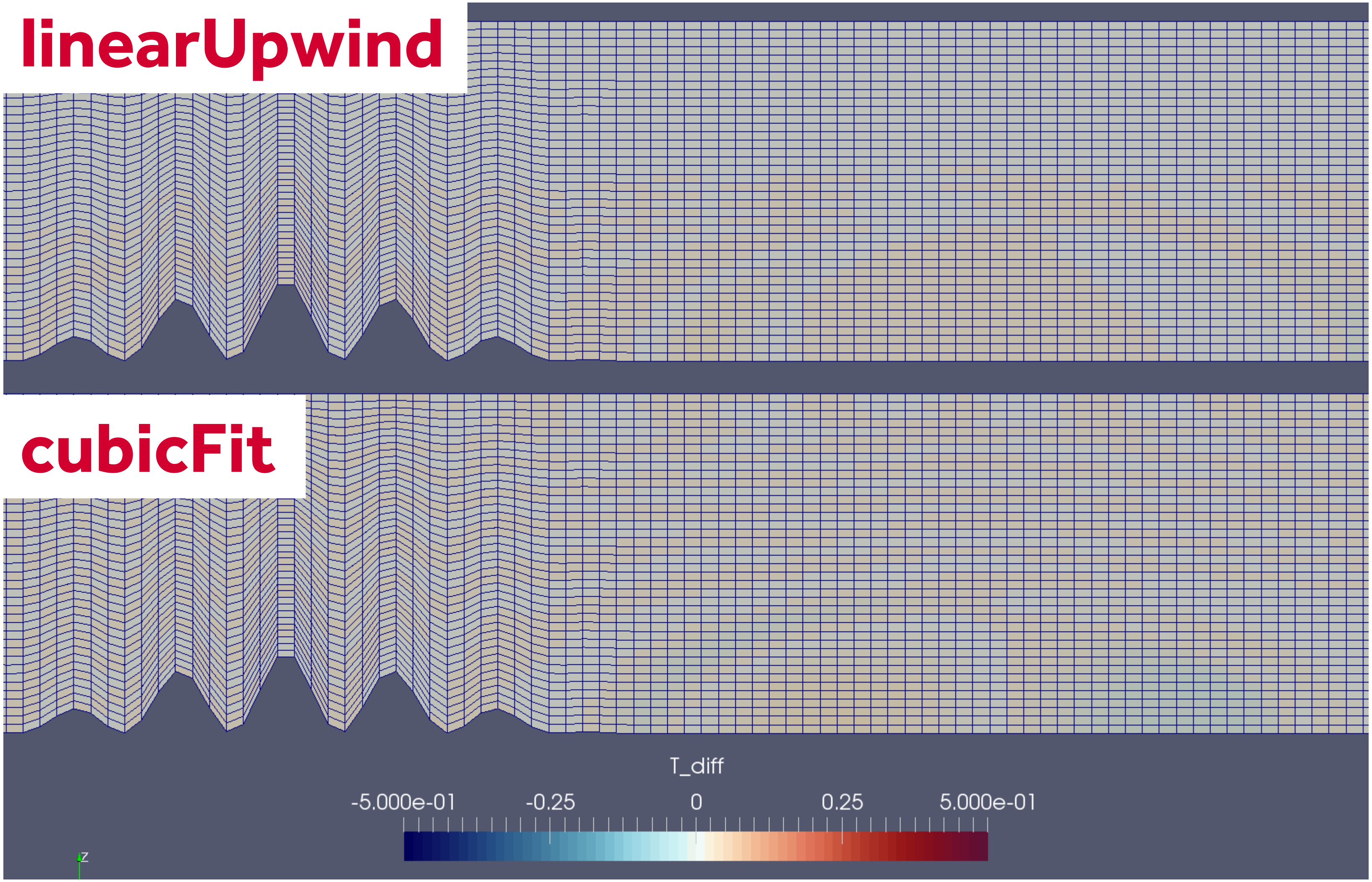

linearUpwind
cubicFit
T_diff
-2.500e-01
-0.12
0
0.125
2.500e-01
Z
Y
X

University of Reading

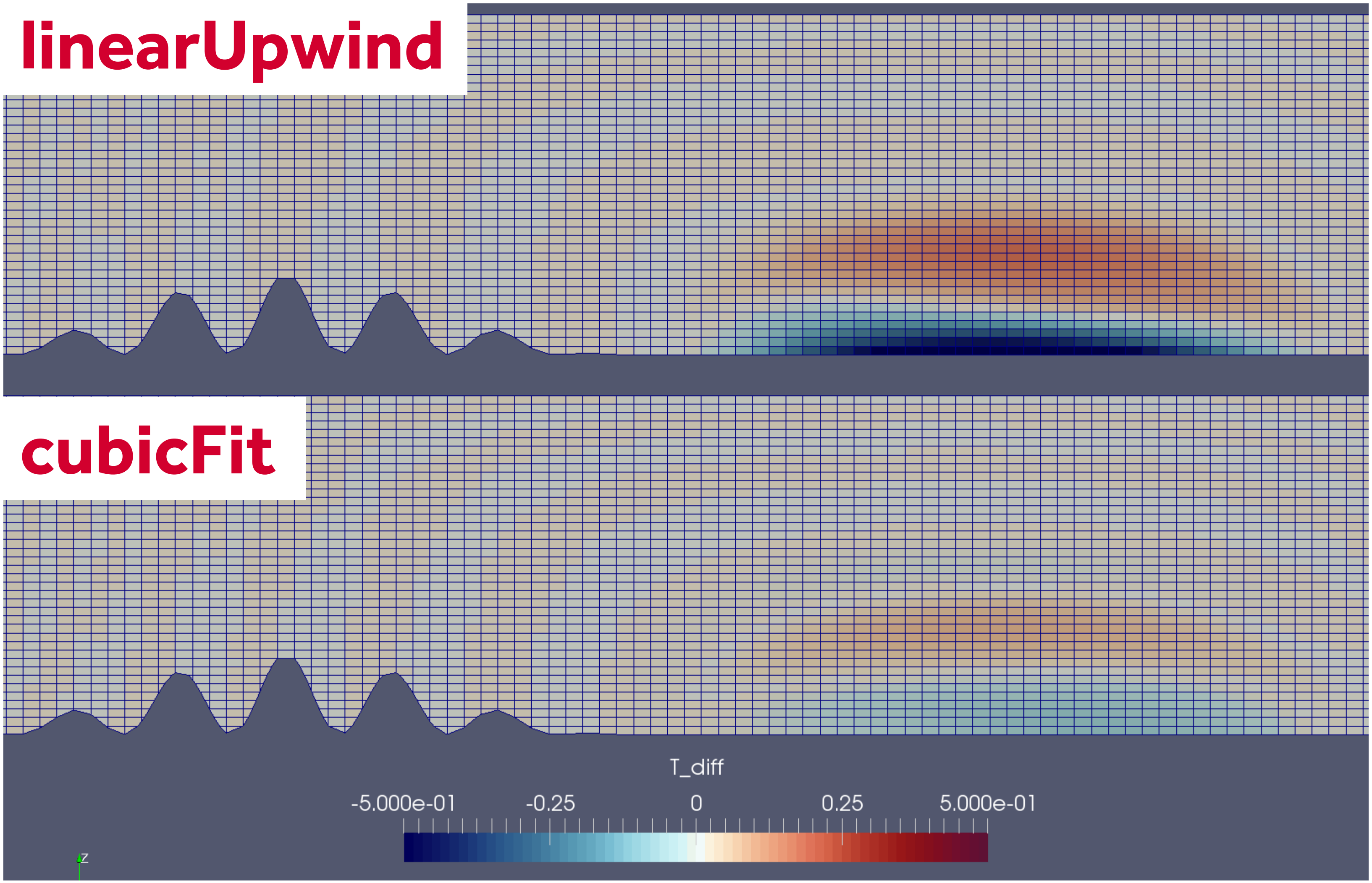
"SLUG" ADVECTION TEST



BASIC TERRAIN FOLLOWING

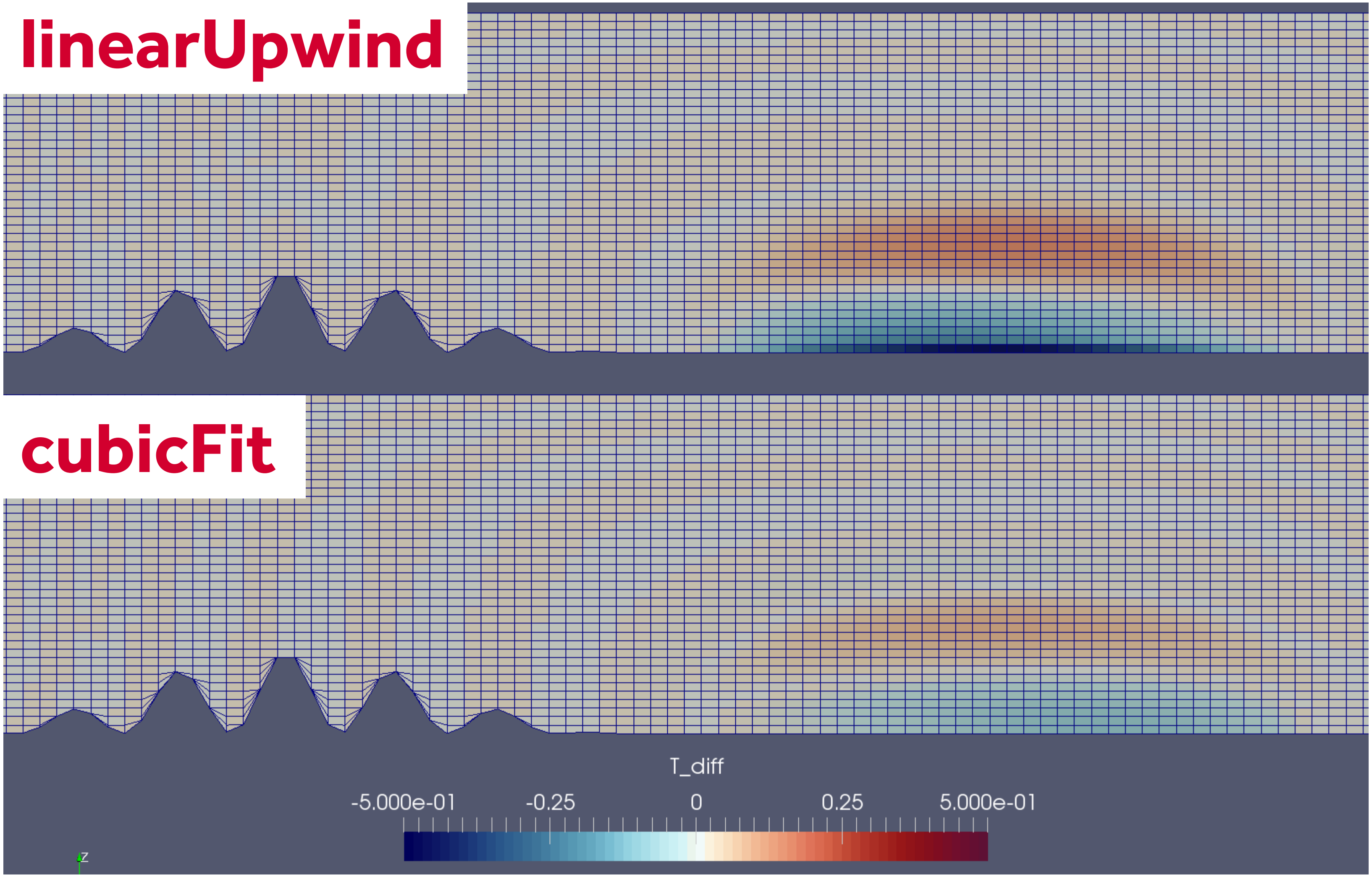


CUT CELLS

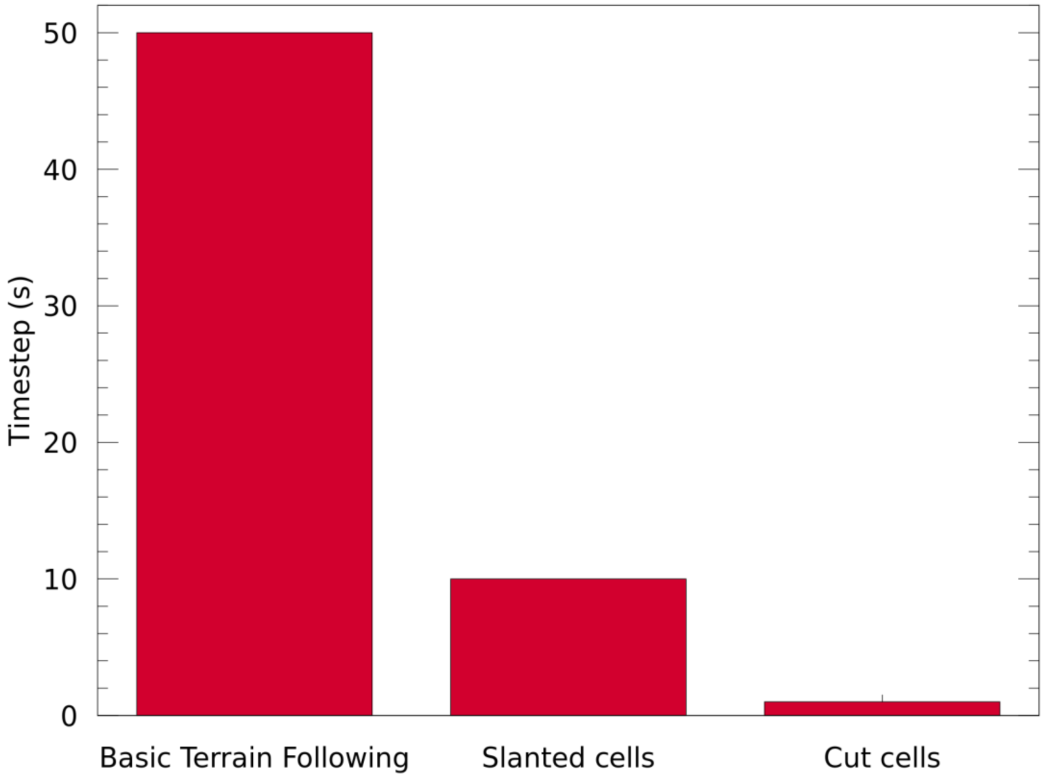


SLANTED CELLS

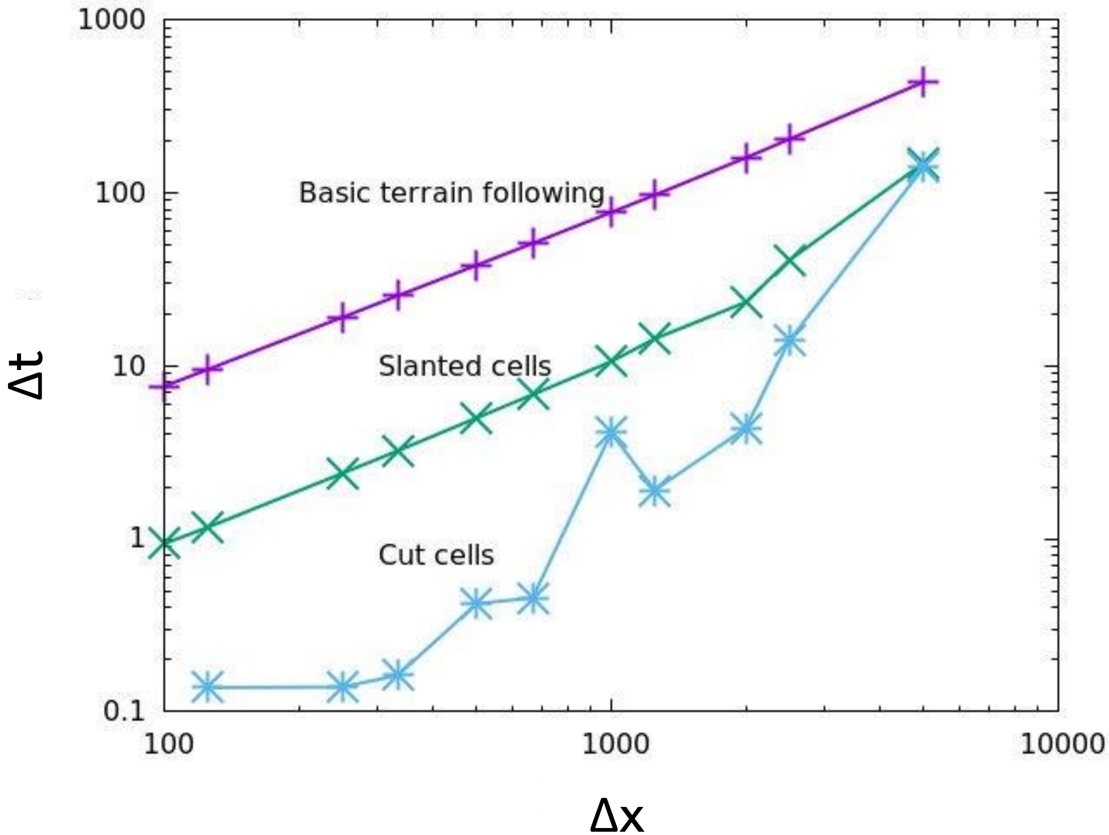
University of Reading



"SLUG" ADVECTION TIMESTEPS



MAXIMUM TIMESTEPS

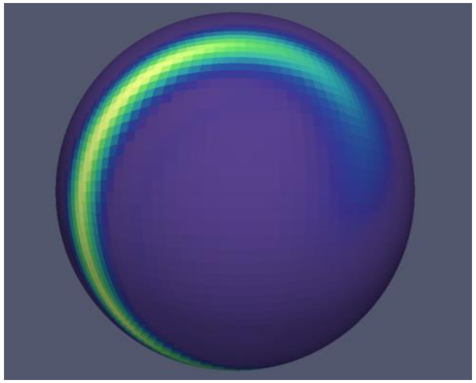
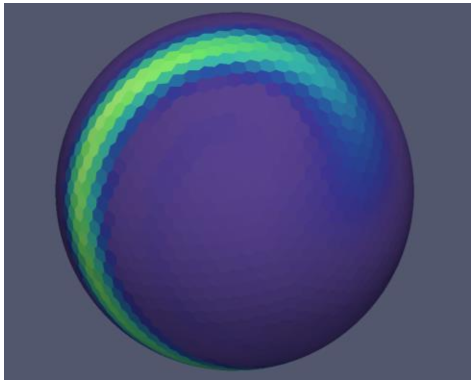
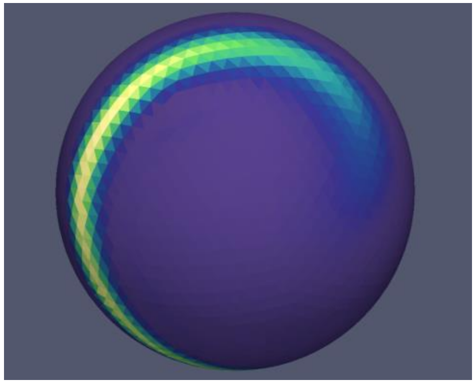


University of Reading

CONCLUSIONS

- cubicFit is cheap to compute (dot product of two vectors)
- cubicFit is suitable for many types of mesh
- Maximum timesteps on slanted cells scale predictably with mesh spacing

FUTURE WORK

Contact me: @hertzsprrrung or js102@zepler.net

Slides and additional resources: goo.gl/jLR7vW