NATIONAL
INSTITUTE OF
AEROSPACE

Face-Averaged Nodal-Gradient Approach to Cell-Centered Finite-Volume Method on Triangular Grids

This work was supported by the Hypersonic Technology Project, through the Hypersonic Airbreathing Propulsion Branch of the NASA Langley Research Center, under Contract No. 80LARC17C0004

Hiroaki Nishikawa, *National Institute of Aerospace*
Jefferey A. White, *NASA Langley Research Center*

AIAA AVIATION Forum | 17-21 June 2019, Dallas, Texas

Contents

Cell-centered method and motivation

Efficient gradient method by Zhang

Efficient reconstruction scheme

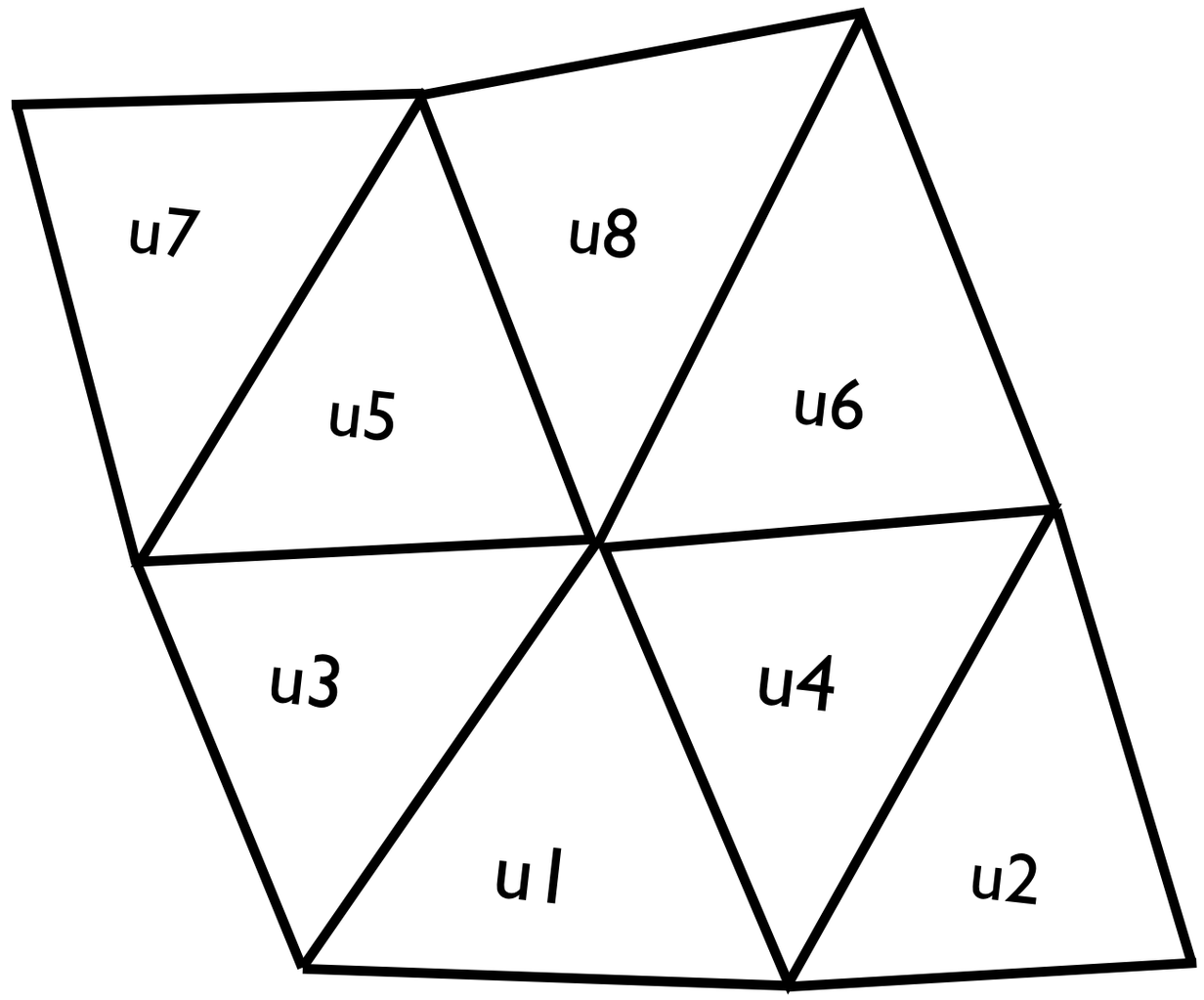
Results

Future work

Navier-Stokes System

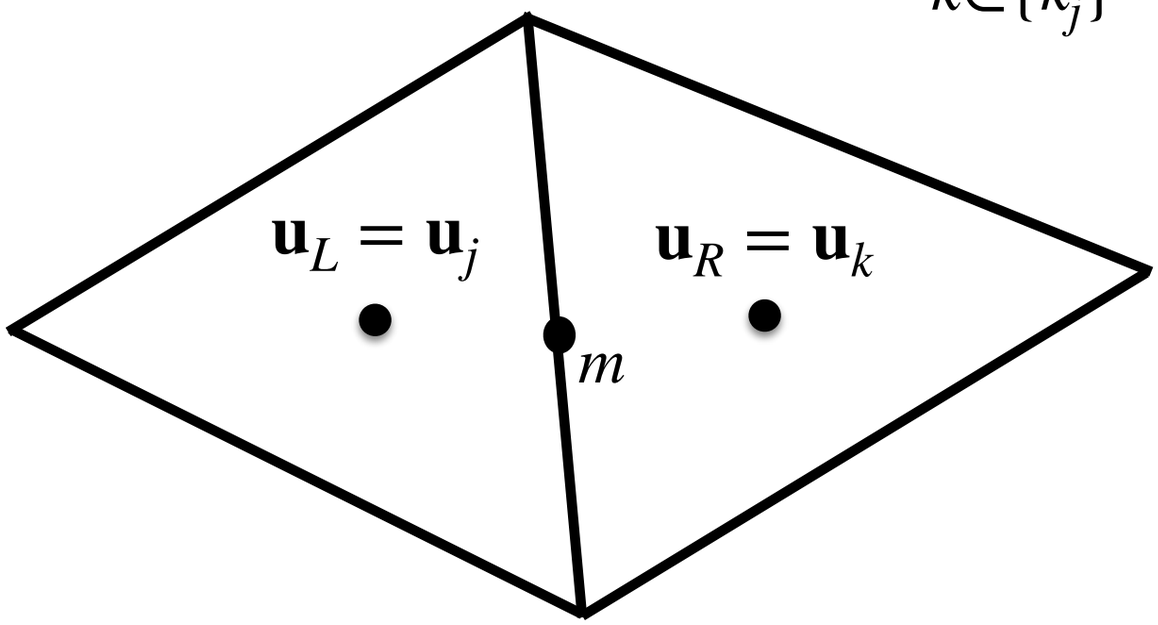
Cell-centered finite-volume discretization on triangles

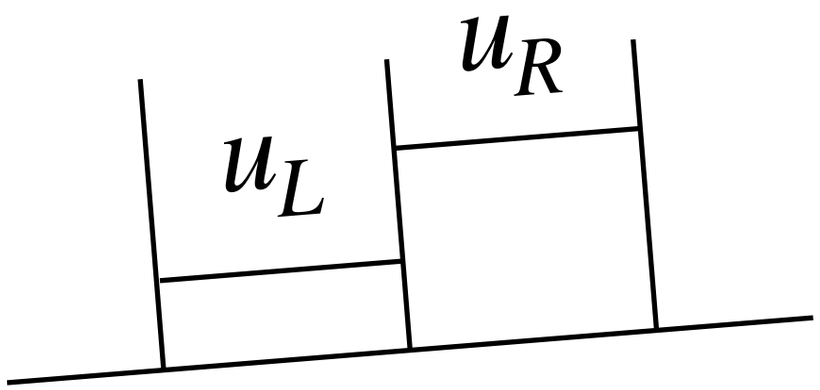
$$\partial_t \mathbf{u} + \partial_x(\mathbf{f}^{inv} + \mathbf{f}^{vis}) + \partial_y(\mathbf{g}^{inv} + \mathbf{g}^{vis}) = 0$$



Inviscid Discretization

$$\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R) dS$$

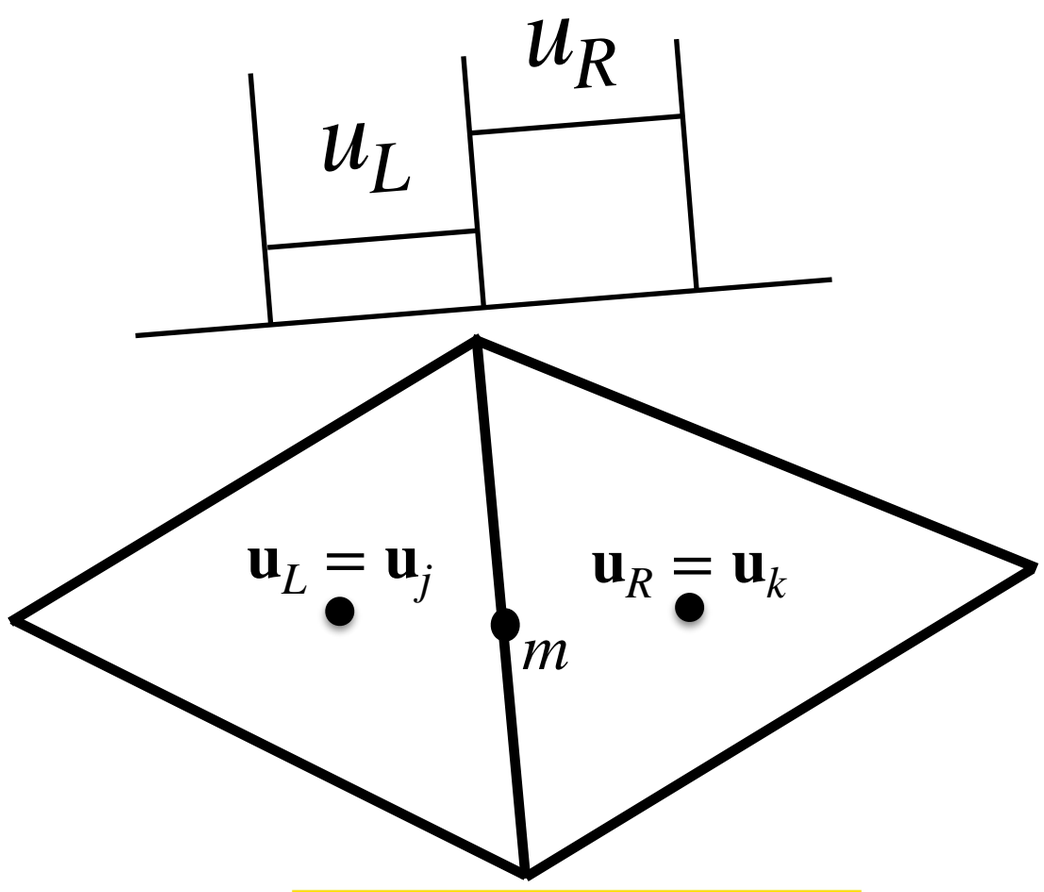


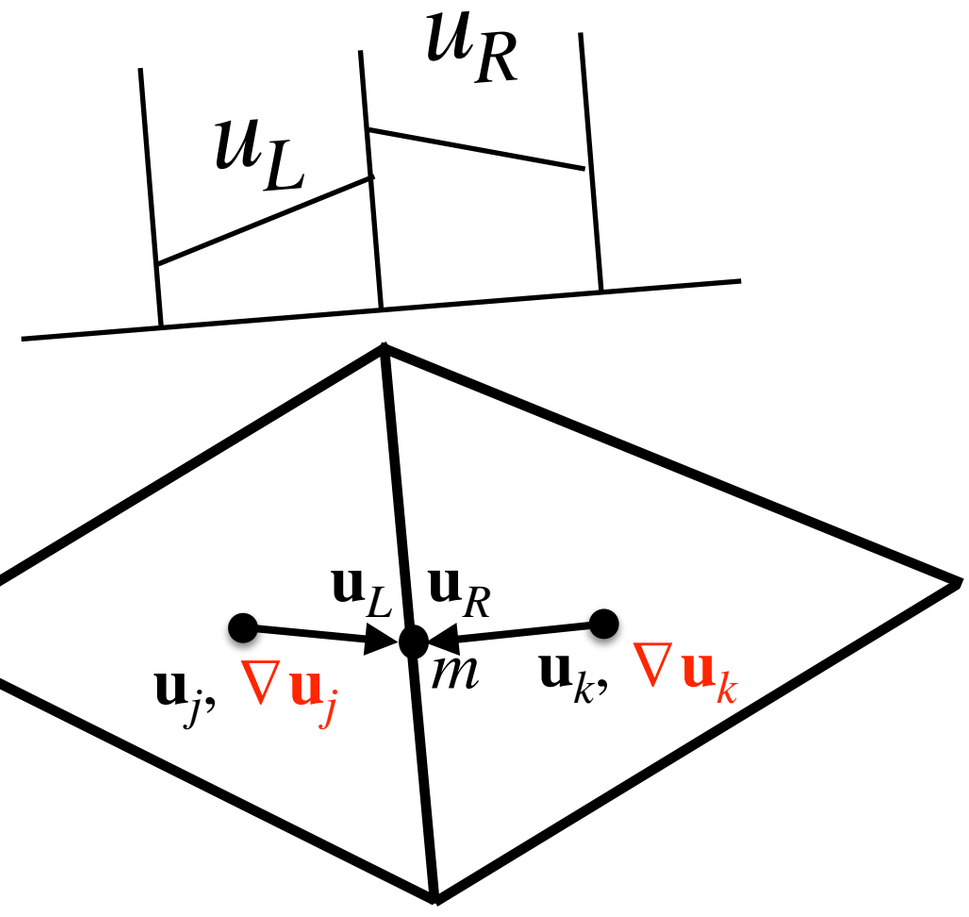


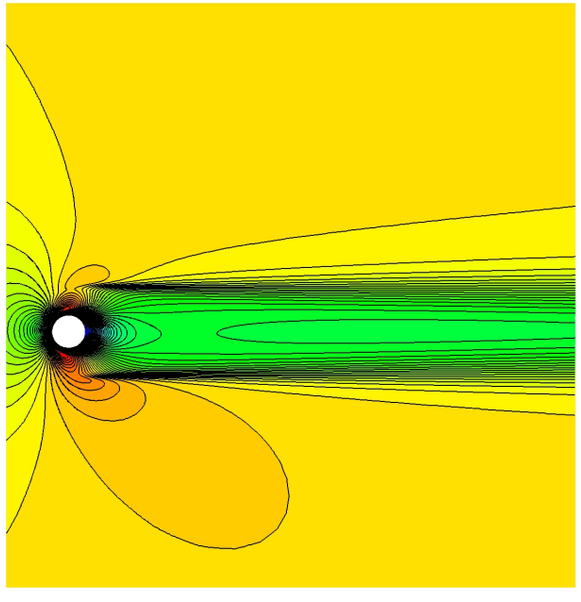
Roe: $\mathbf{\Phi}^{inv} = \frac{1}{2}\left[\mathbf{f}_R^{inv} + \mathbf{f}_L^{inv}\right] - \frac{1}{2}|\hat{\mathbf{A}}|\left(\mathbf{u}_R - \mathbf{u}_L\right)$

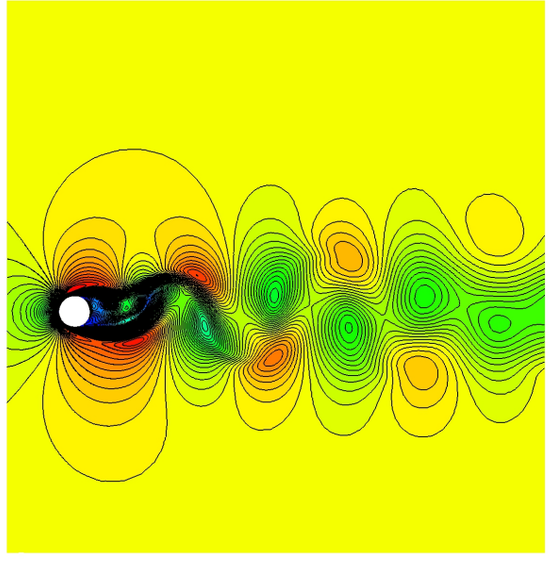
HLL: $\mathbf{\Phi}^{inv} = \frac{S_R \mathbf{f}_L^{inv} - S_L \mathbf{f}_R^{inv}}{S_R - S_L} - \frac{S_R S_L}{S_R - S_L}(\mathbf{u}_R - \mathbf{u}_L)$

2nd-order accuracy



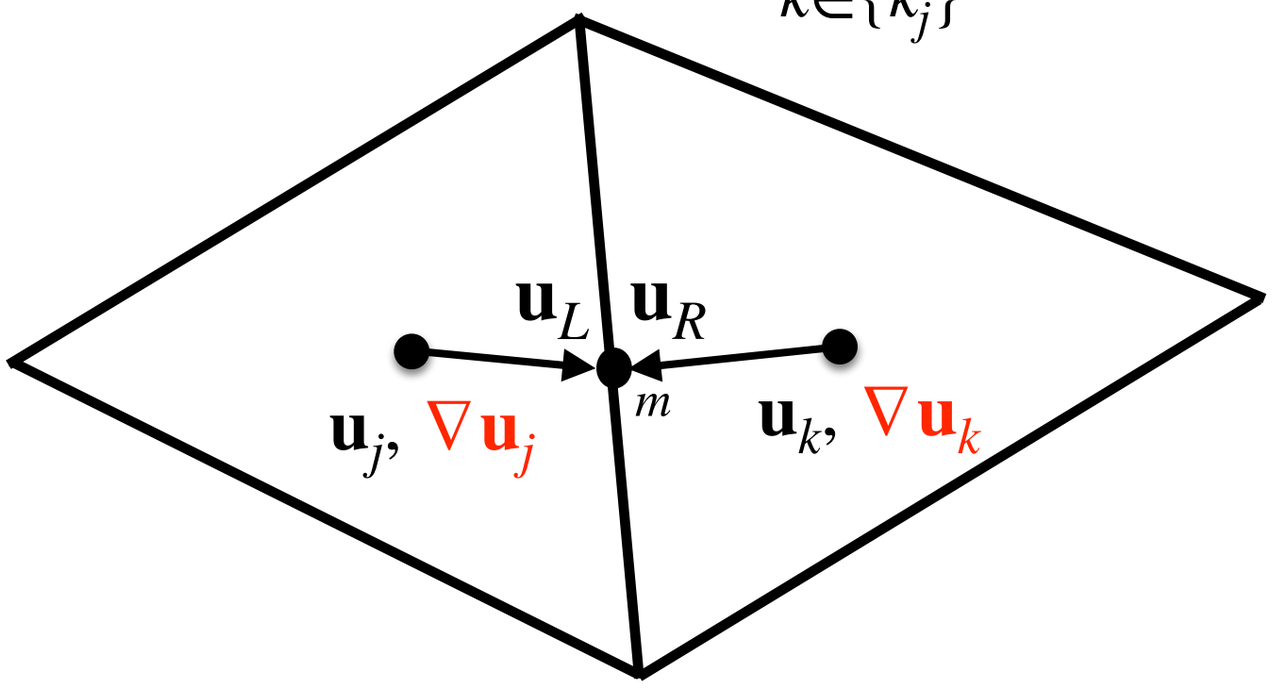


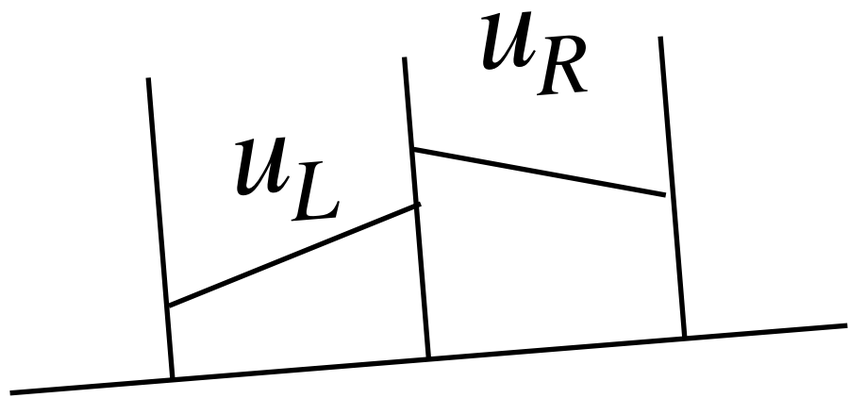




Viscous discretization:

$$\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R, \nabla \mathbf{u}_j, \nabla \mathbf{u}_k) dS$$



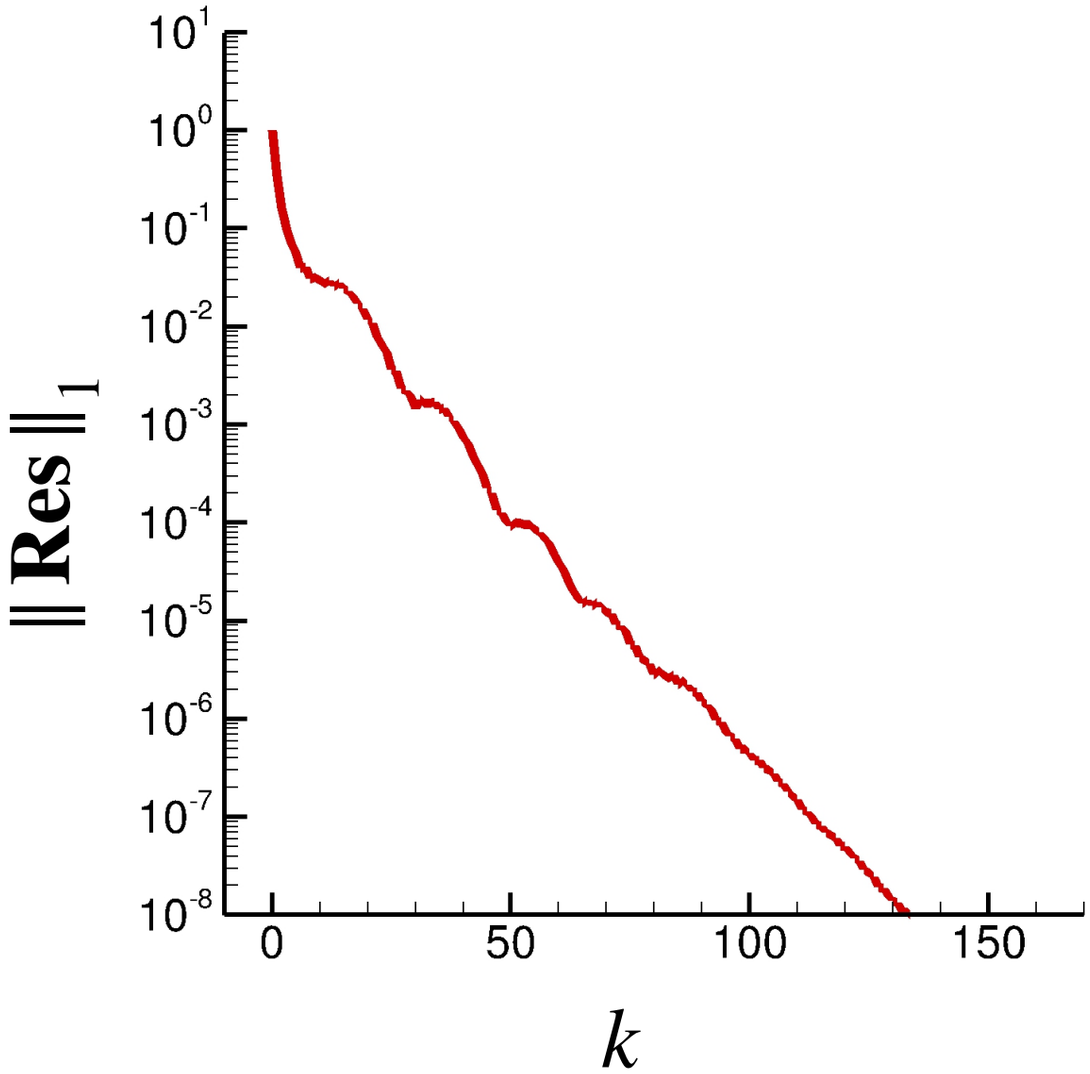


$$\mathbf{\Phi}^{vis} = \mathbf{f}^{vis}(\bar{\mathbf{u}}, \nabla \mathbf{u})$$

Nishikawa, AIAA2010-0593

$$\nabla \mathbf{u} = \frac{1}{2}\left[\nabla \mathbf{u}_j + \nabla \mathbf{u}_k\right] + \frac{(4/3)}{2\left|(\mathbf{x}_k - \mathbf{x}_j) \cdot \hat{\mathbf{n}}_{jk}\right|}\left(\mathbf{u}_R - \mathbf{u}_L\right)$$

Implicit defect-defect-correction solver

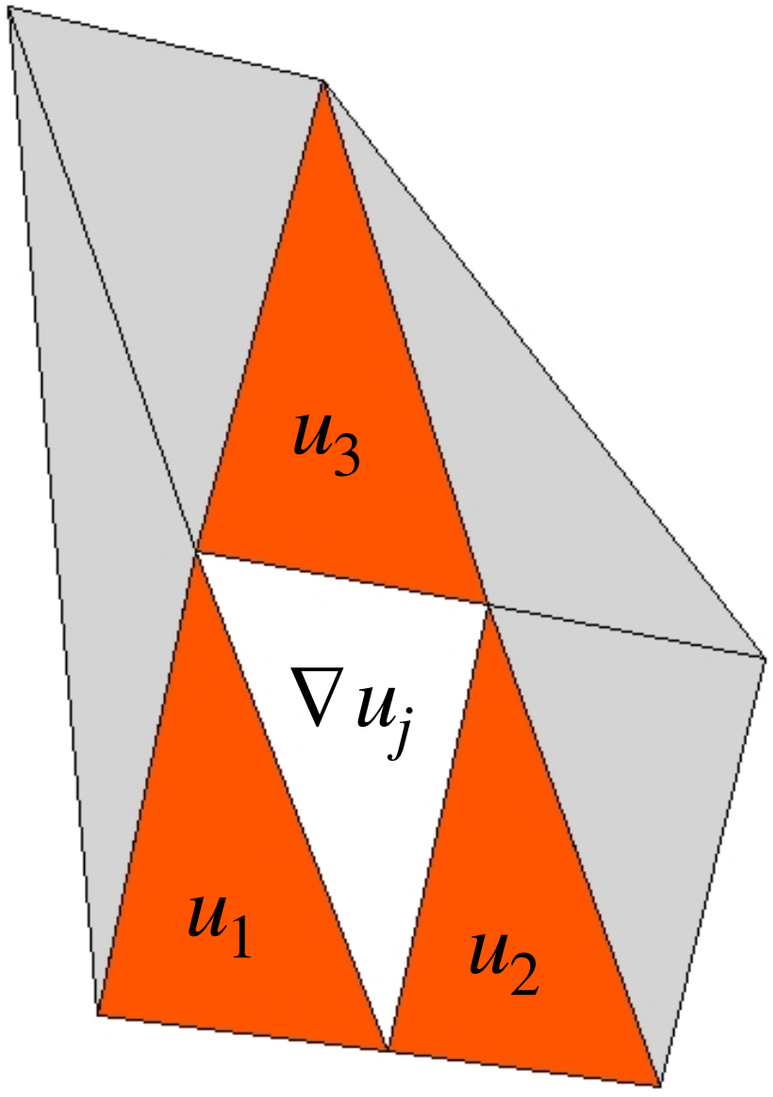


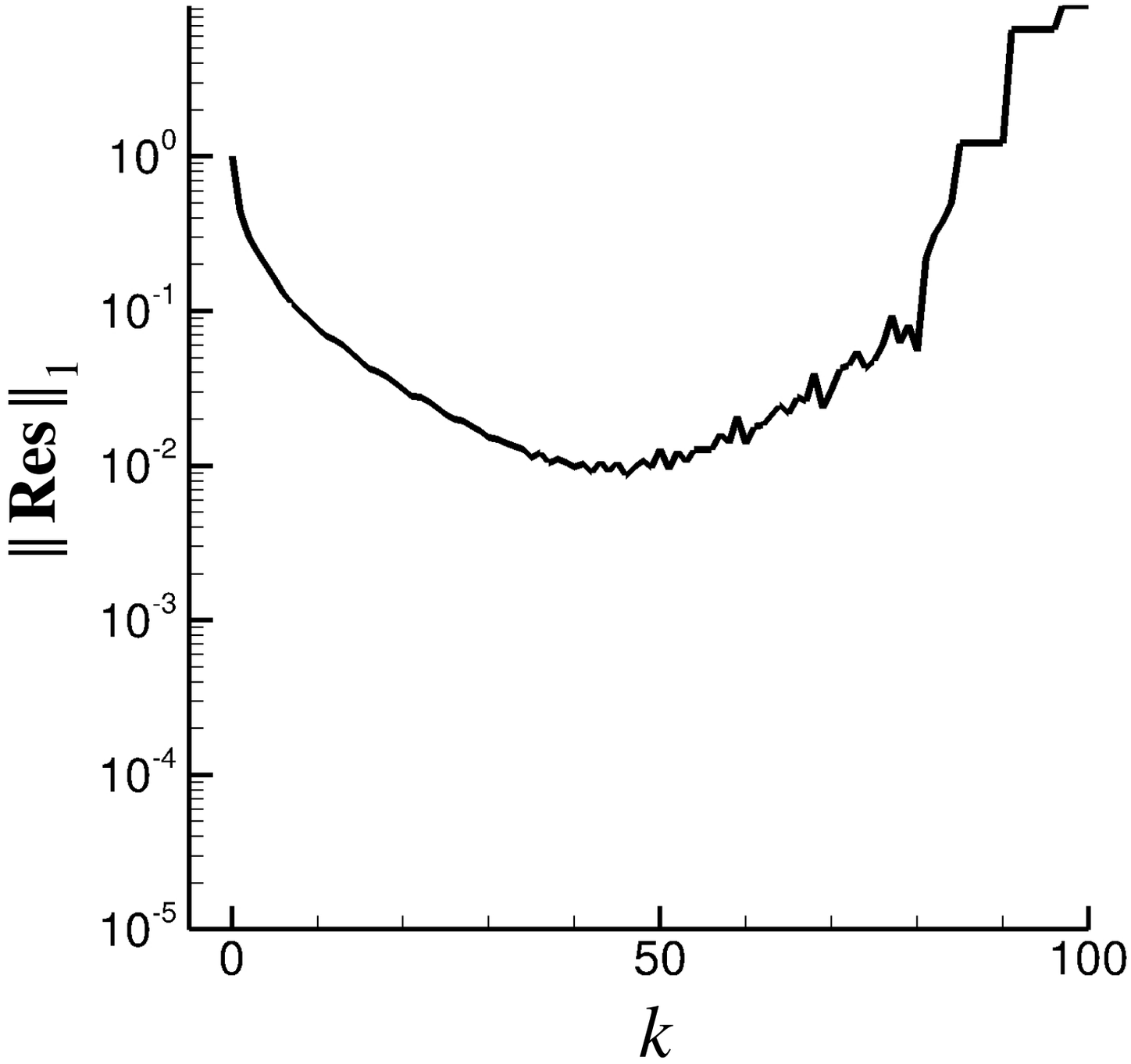
$$\mathbf{U}^{k+1} = \mathbf{U}^{k} + \Delta\mathbf{U}$$

$$\left[\frac{\mathbf{V}}{\Delta\tau} + \frac{\partial \mathbf{Res}^{(1)}}{\partial \mathbf{U}}\right] \Delta\mathbf{U} = -\,\mathbf{Res}(\mathbf{U}^{k})$$

$$\mathbf{Res}^{(1)} = \mathbf{Res}(\nabla\mathbf{U} = 0)$$

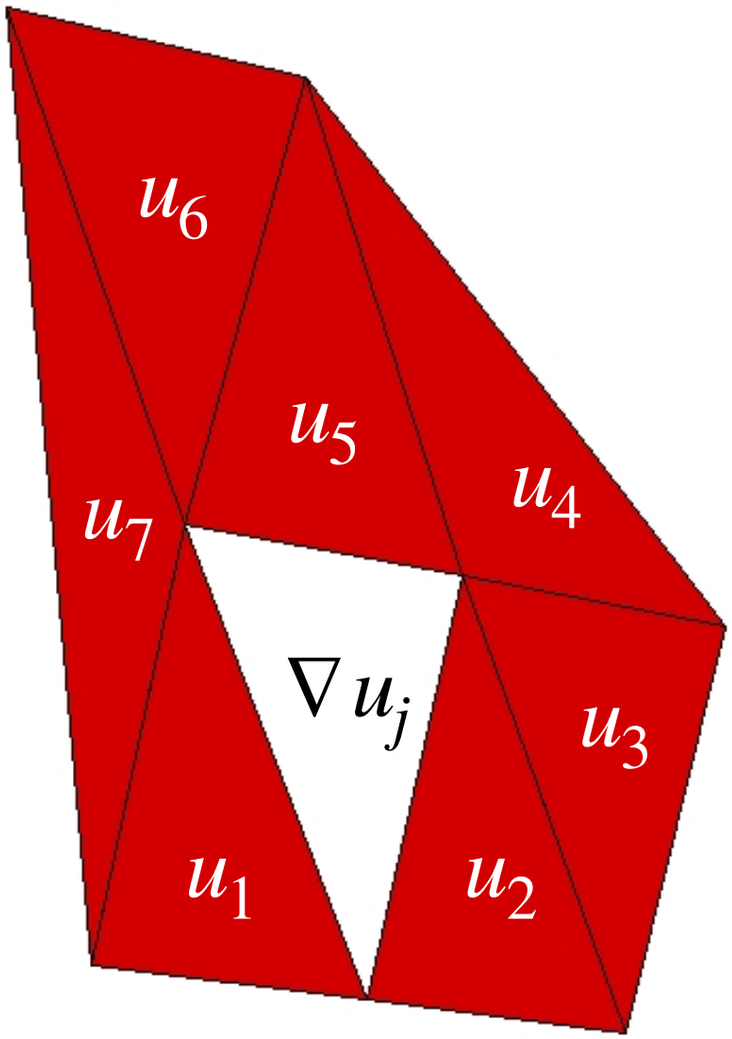
Solver gets unstable if gradients are computed with face-neighbors

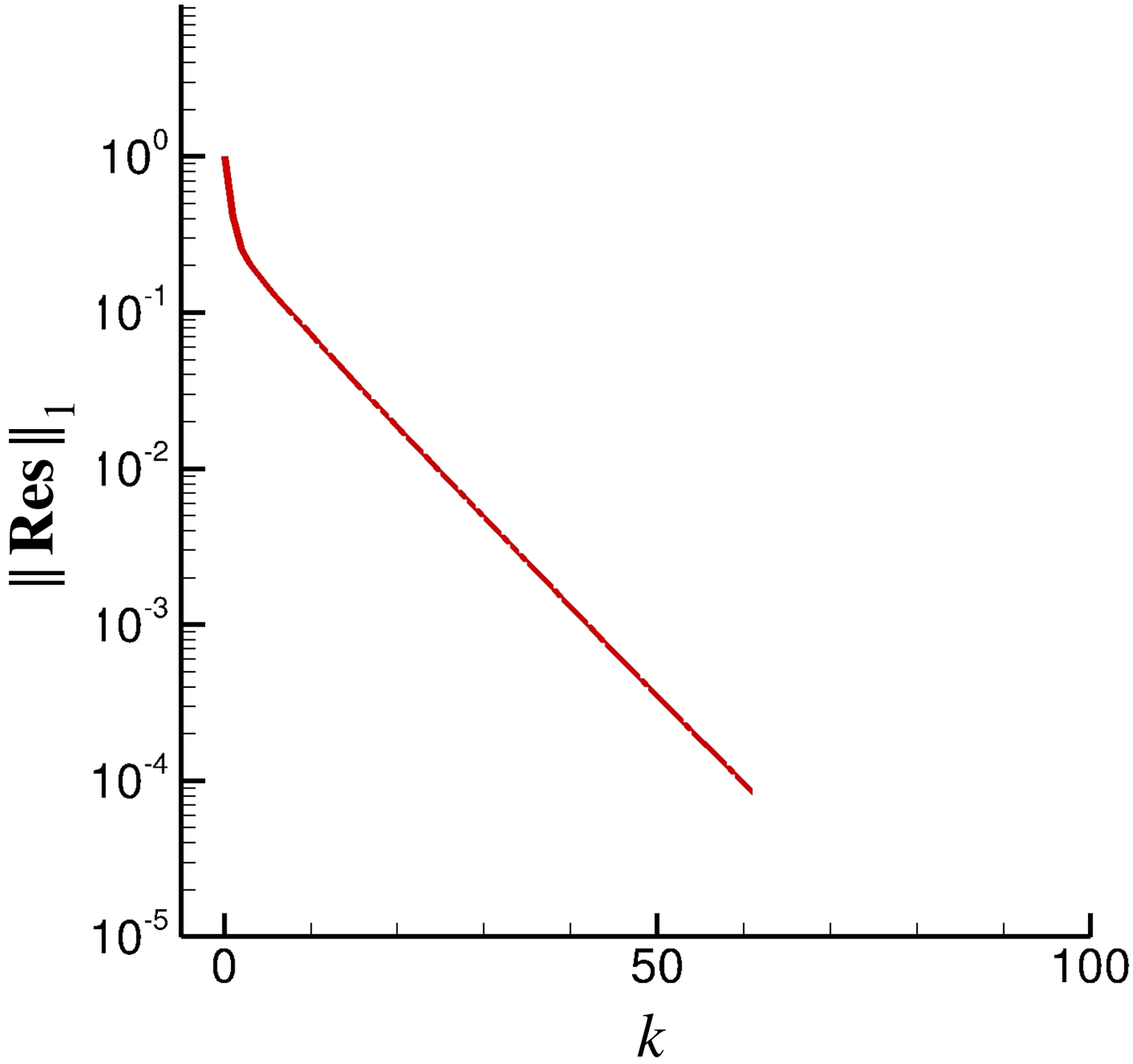




Haider, et al., Numer. Math., Vol. 113, No. 4, 2009
Diskin&Thomas, AIAA J., Vol. 49, No. 4, 2011

More neighbors for stability





Cost of Gradients

$$\begin{bmatrix} \partial_x u_j \\ \partial_y u_j \end{bmatrix} = \sum_k \begin{bmatrix} c_{jk}^x \\ c_{jk}^y \end{bmatrix} (u_k - u_j)$$

Memory

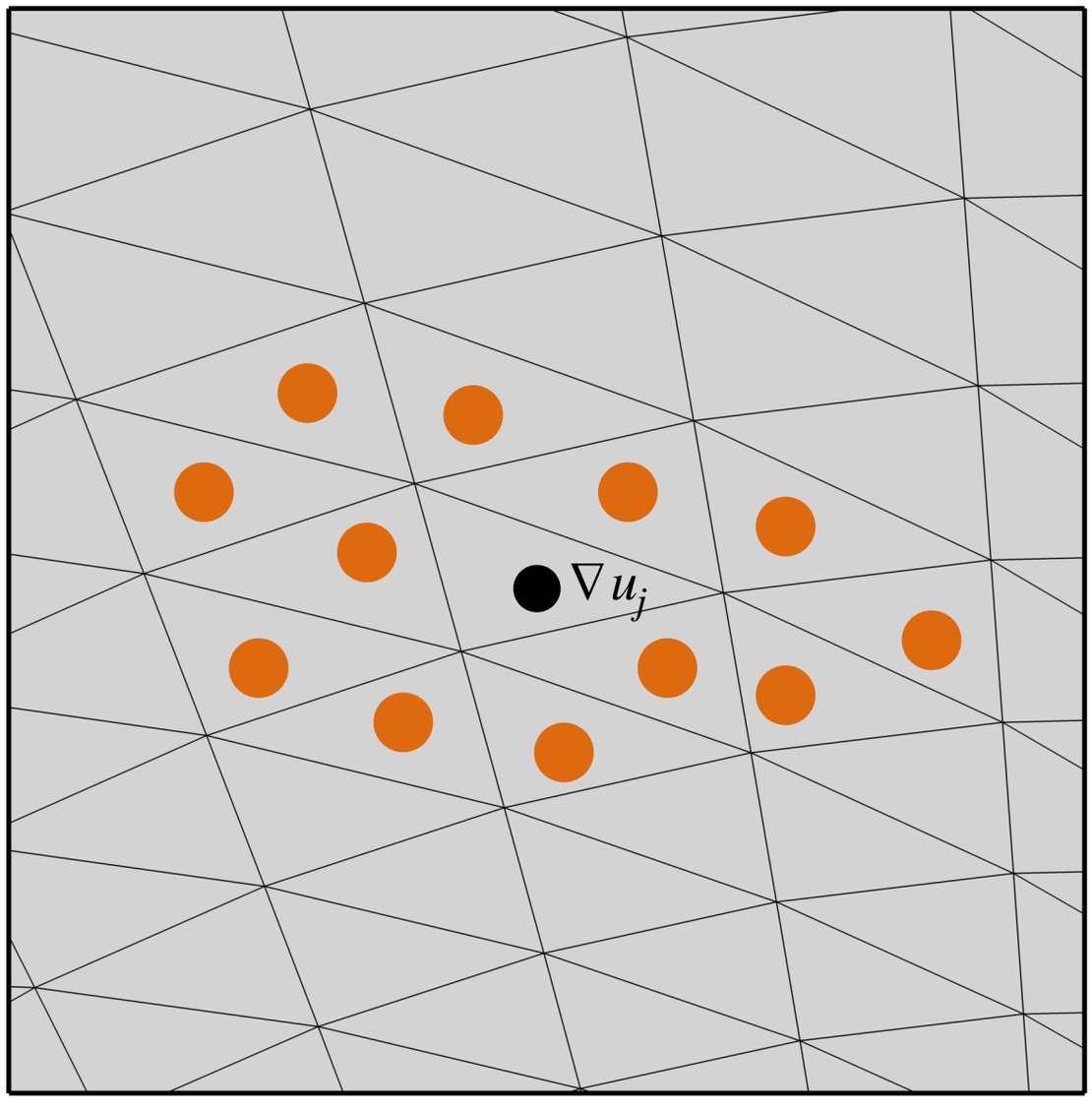
$$\begin{bmatrix} c_{jk}^x \\ c_{jk}^y \end{bmatrix} = ncells \times nnghbrs$$

CPU

```
do j = 1, ncells
 do k = 1, nnghbrs(j)
```

$$\begin{bmatrix} \partial_x u_j \\ \partial_y u_j \end{bmatrix} = \sum_k \begin{bmatrix} c_{jk}^x \\ c_{jk}^y \end{bmatrix} (u_k - u_j)$$

```
 end do
end do
```



$$\rightarrow ncells \times nnghbrs \approx 2nnodes \times 12 = \underline{24 \times nnodes}$$

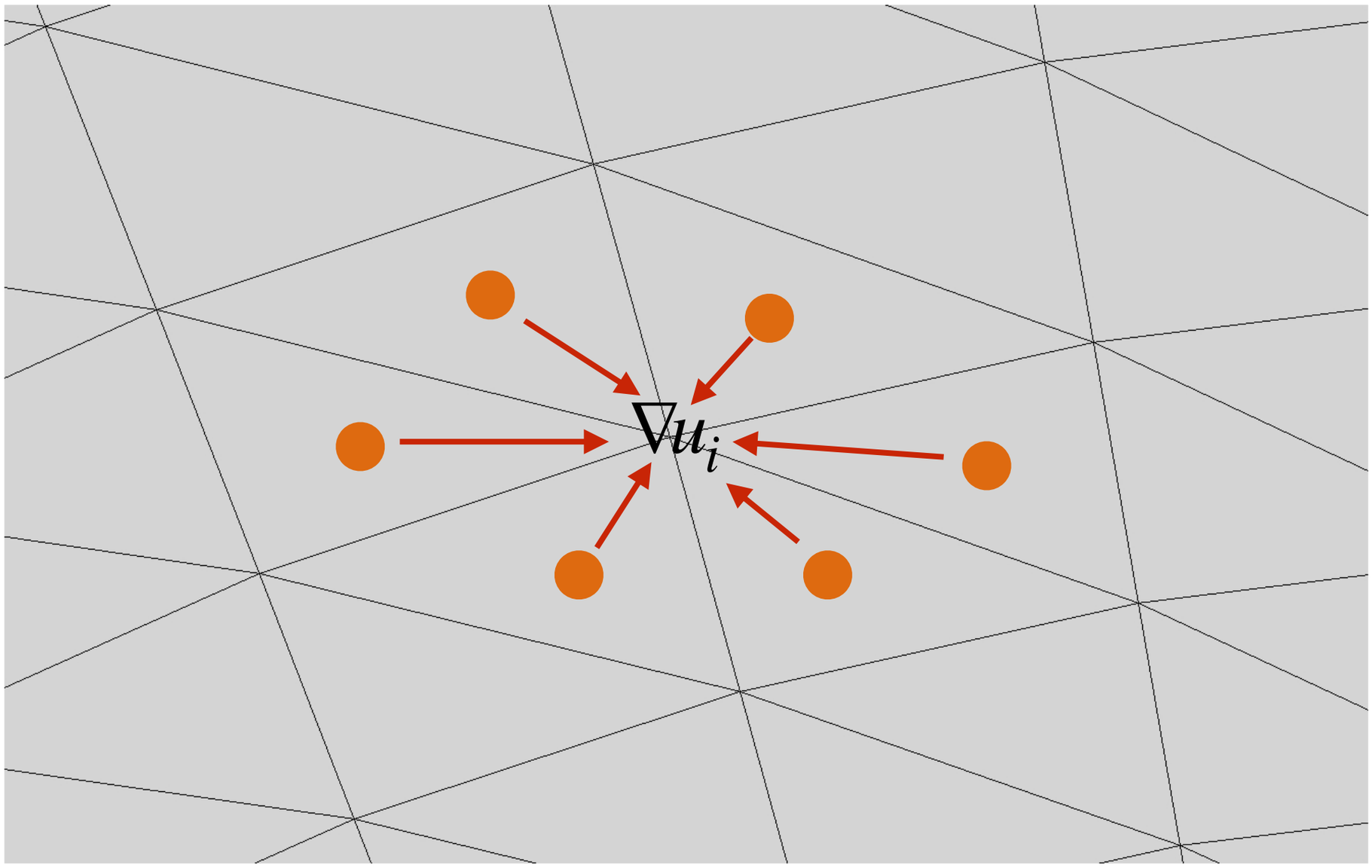
Efficient gradient method by Zhang

Zhang proposed to compute gradients at nodes

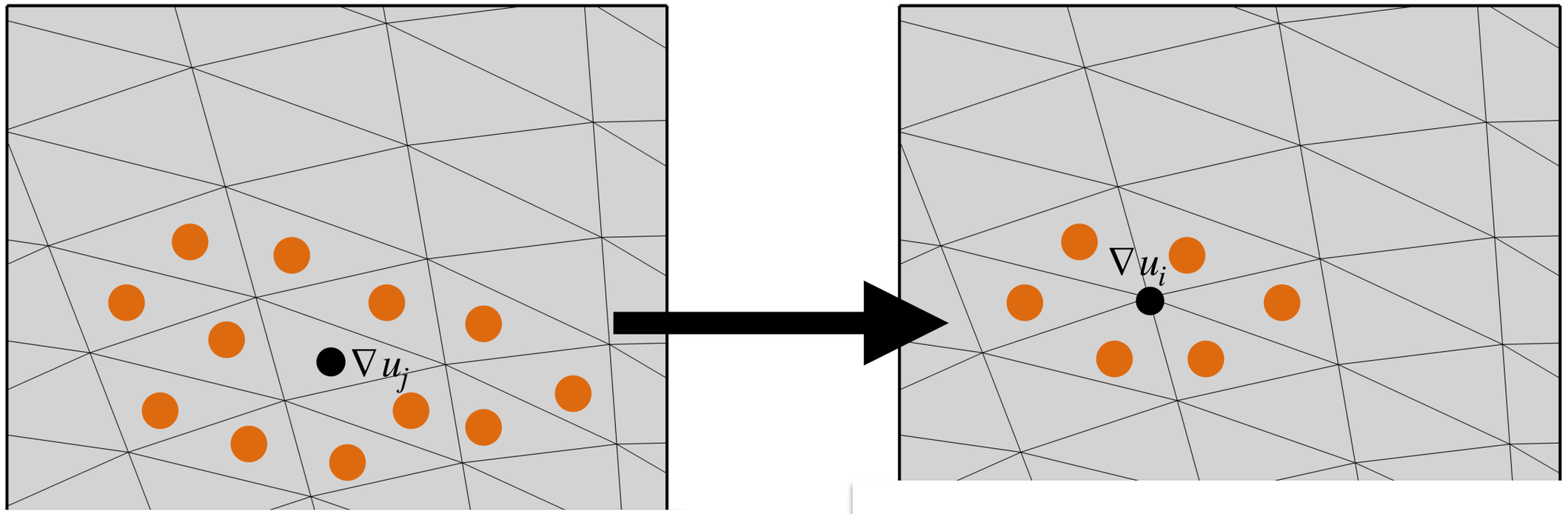
Fan Zhang (2015,2017)

"A vertex-weighted-least-squares gradient reconstruction"

https://arxiv.org/pdf/1702.04518.pdf



75% savings in cost



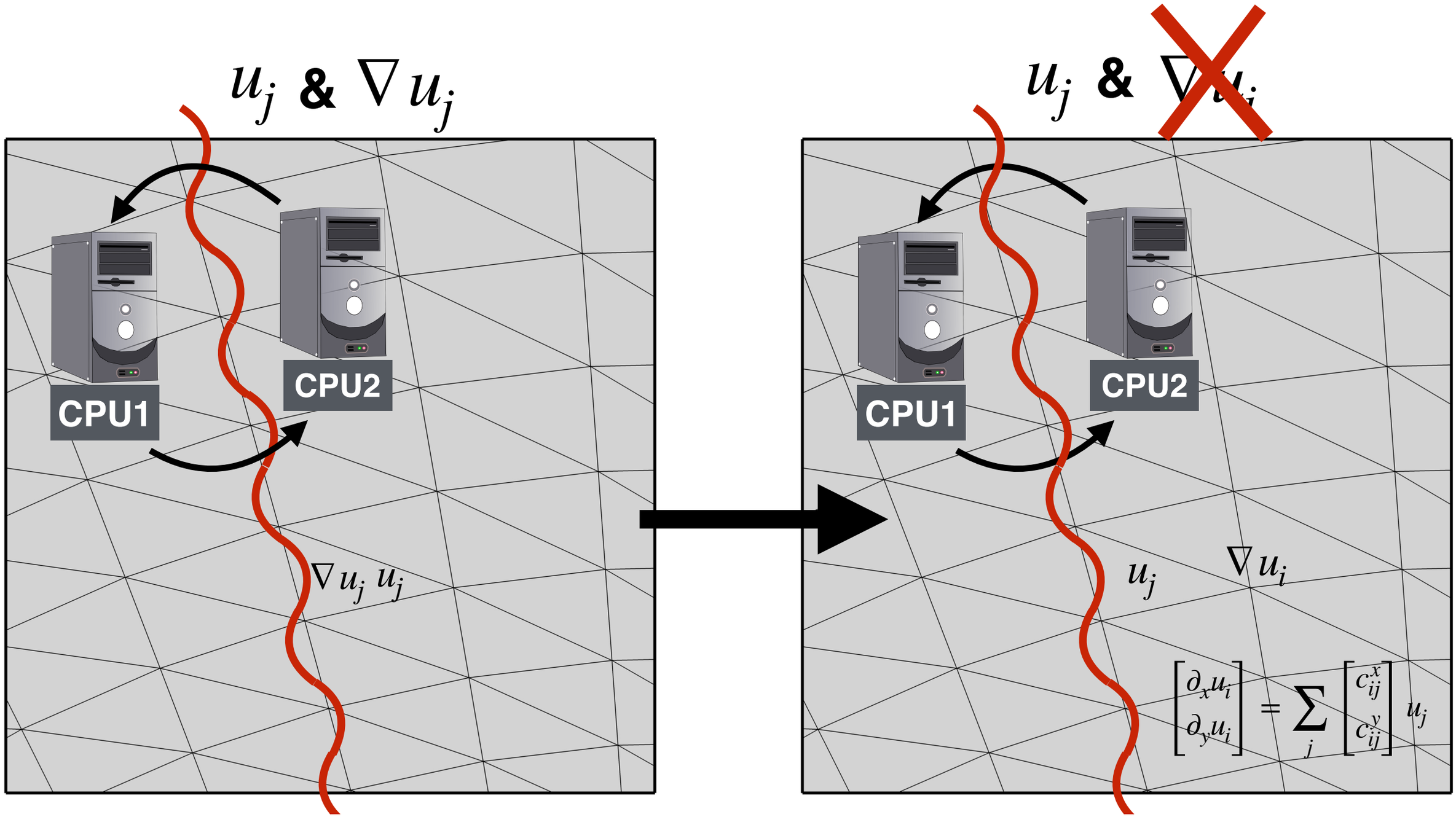
$$\begin{bmatrix} \partial_x u_j \\ \partial_y u_j \end{bmatrix} = \sum_k \begin{bmatrix} c_{jk}^x \\ c_{jk}^y \end{bmatrix} (u_k - u_j)$$

$$\begin{bmatrix} u_i \\ \partial_x u_i \\ \partial_y u_i \end{bmatrix} = \sum_j \begin{bmatrix} c_{ij}^u \\ c_{ij}^x \\ c_{ij}^y \end{bmatrix} u_j$$

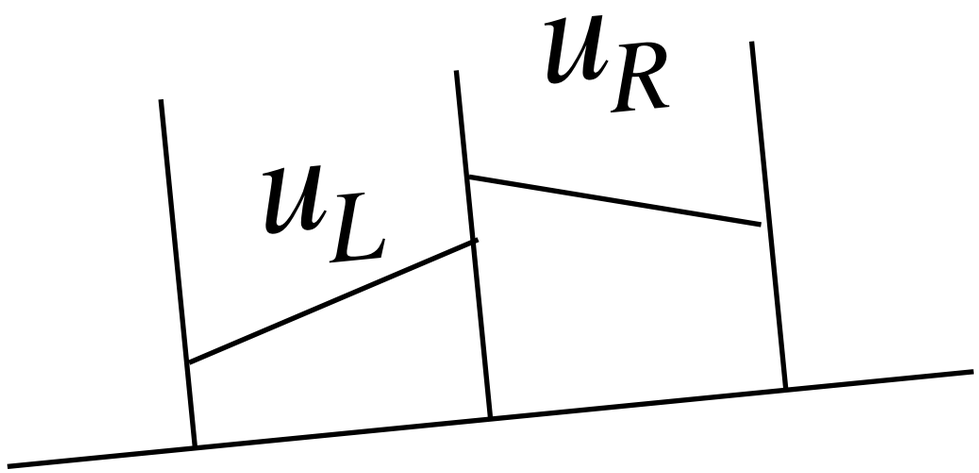
$24 \times nnodes \longrightarrow 6 \times nnodes$

$$\frac{1}{4}$$

Communication not needed

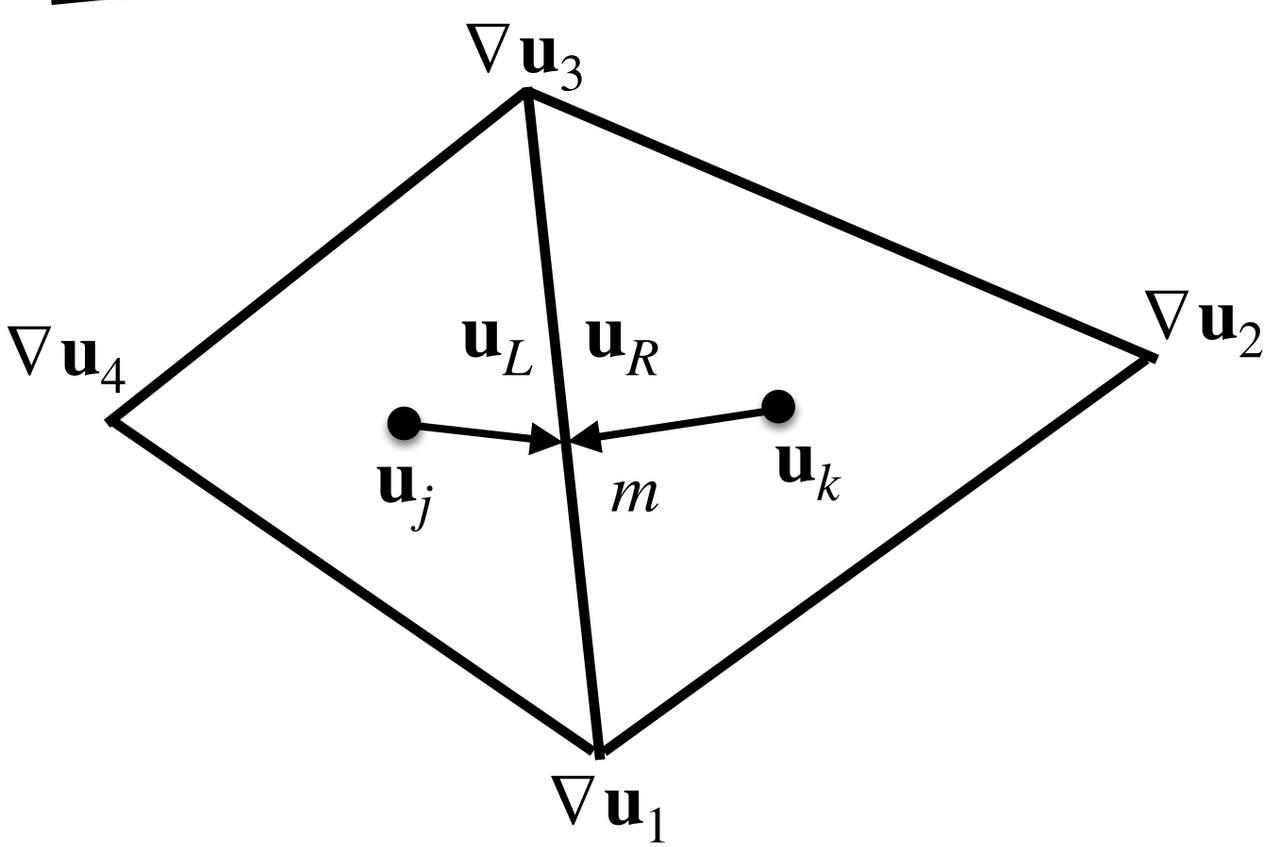


But need cell gradients…
How do we get them?



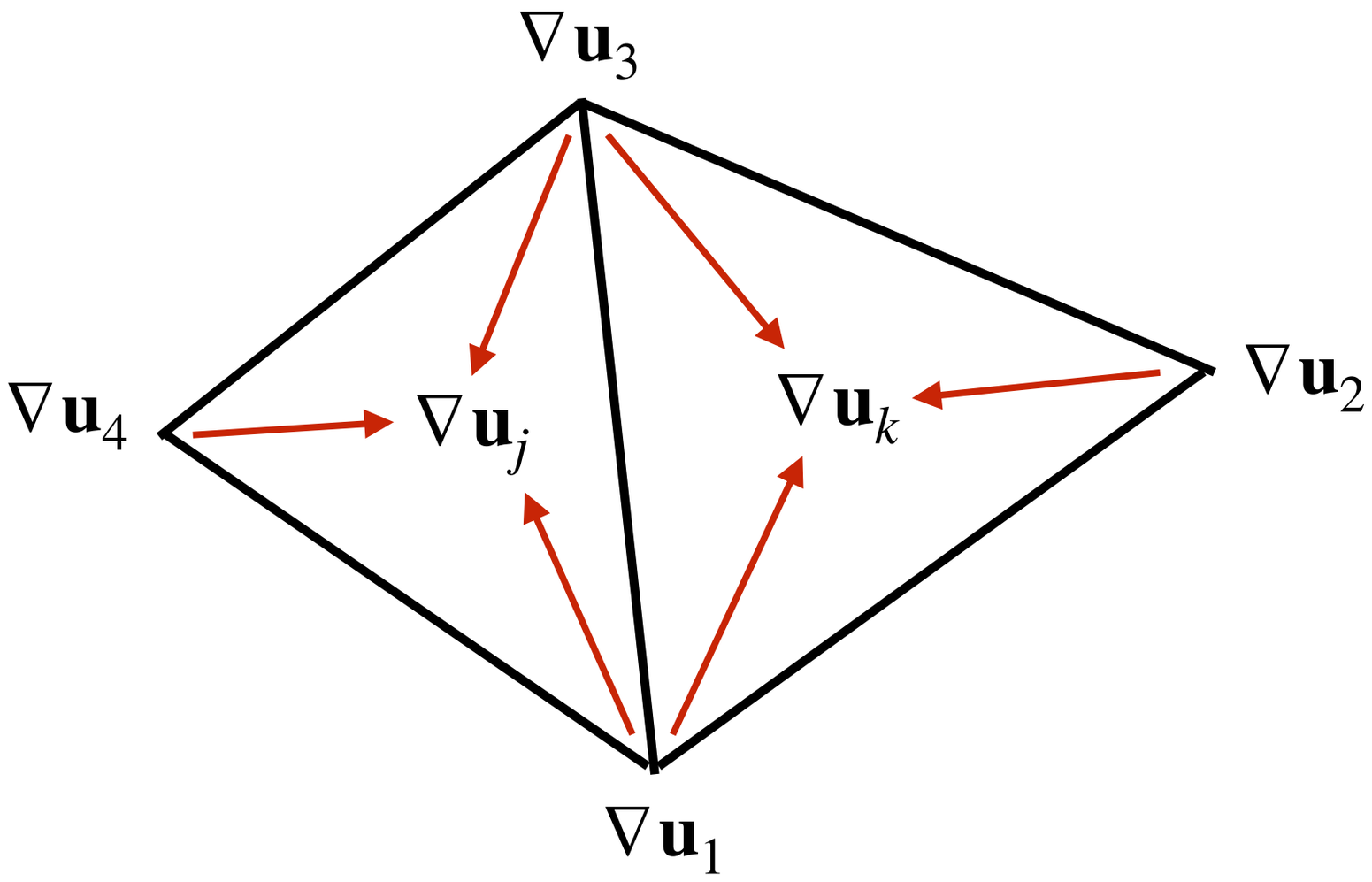
$$\mathbf{u}_L = \mathbf{u}_j + \underline{\nabla \mathbf{u}_j} \cdot (\mathbf{x}_m - \mathbf{x}_j)$$

$$\mathbf{u}_R = \mathbf{u}_k + \underline{\nabla \mathbf{u}_k} \cdot (\mathbf{x}_m - \mathbf{x}_k)$$



Fan Zhang (2015,2017)

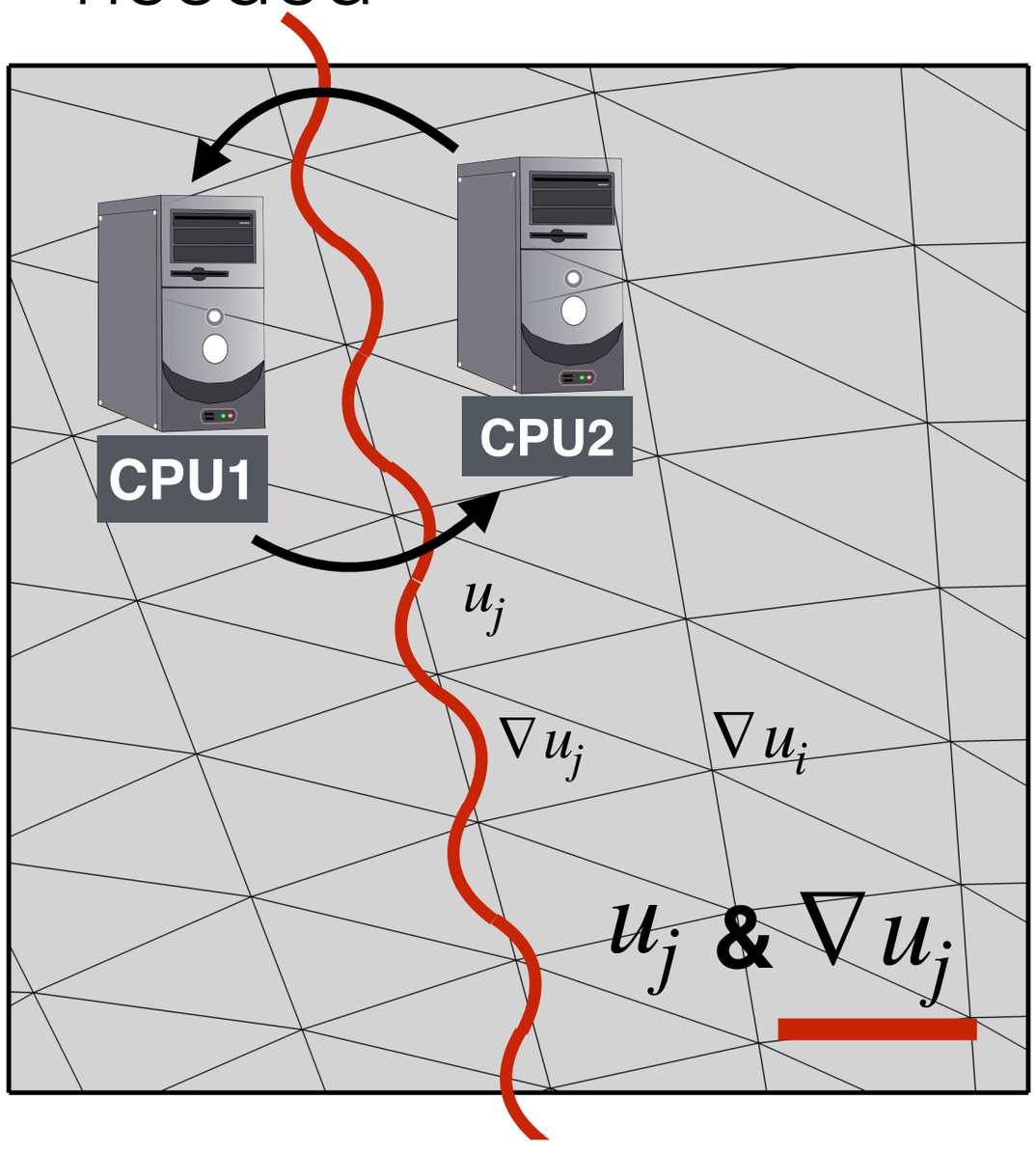
proposes the arithmetic average over each cell



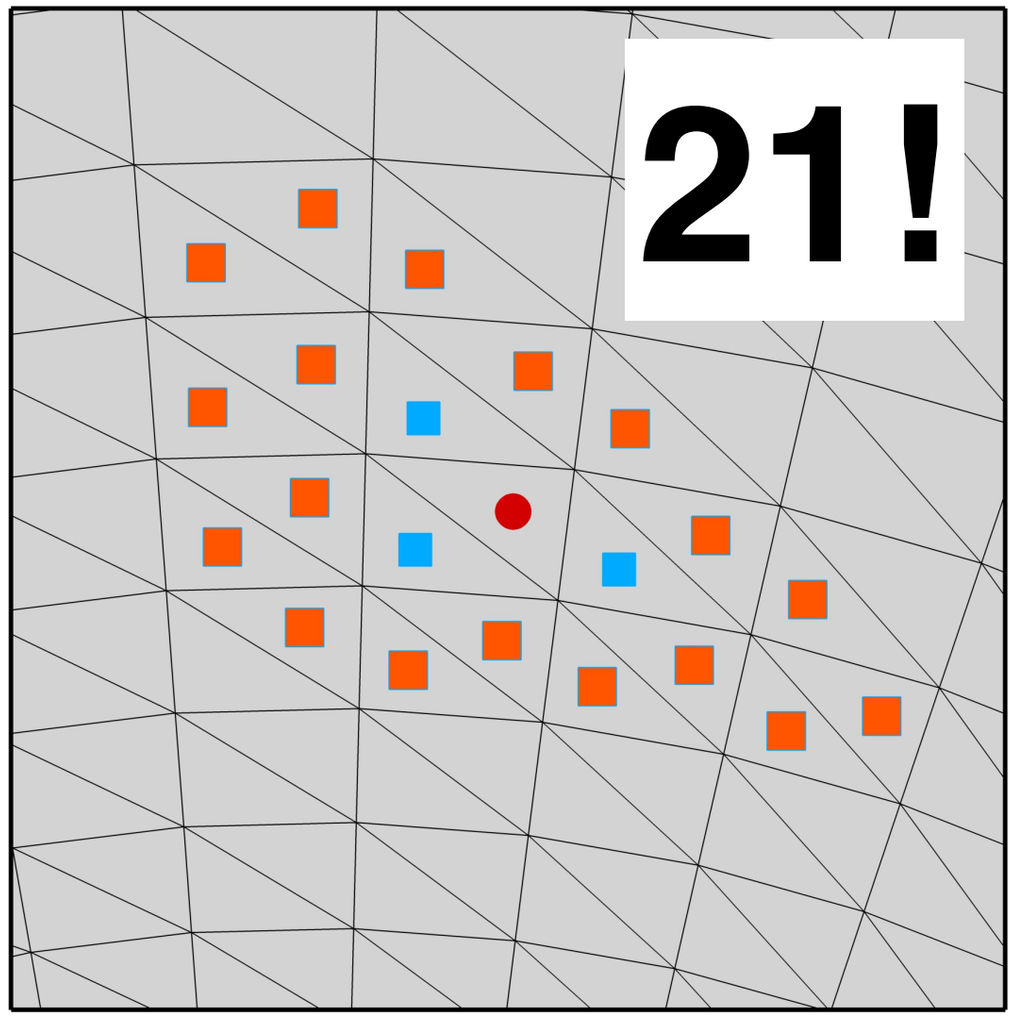
$$\nabla \mathbf{u}_j = \frac{1}{3}\left(\nabla \mathbf{u}_1 + \nabla \mathbf{u}_3 + \nabla \mathbf{u}_4\right) \qquad \nabla \mathbf{u}_k = \frac{1}{3}\left(\nabla \mathbf{u}_1 + \nabla \mathbf{u}_2 + \nabla \mathbf{u}_3\right)$$

Problems:

1. Gradient communication needed



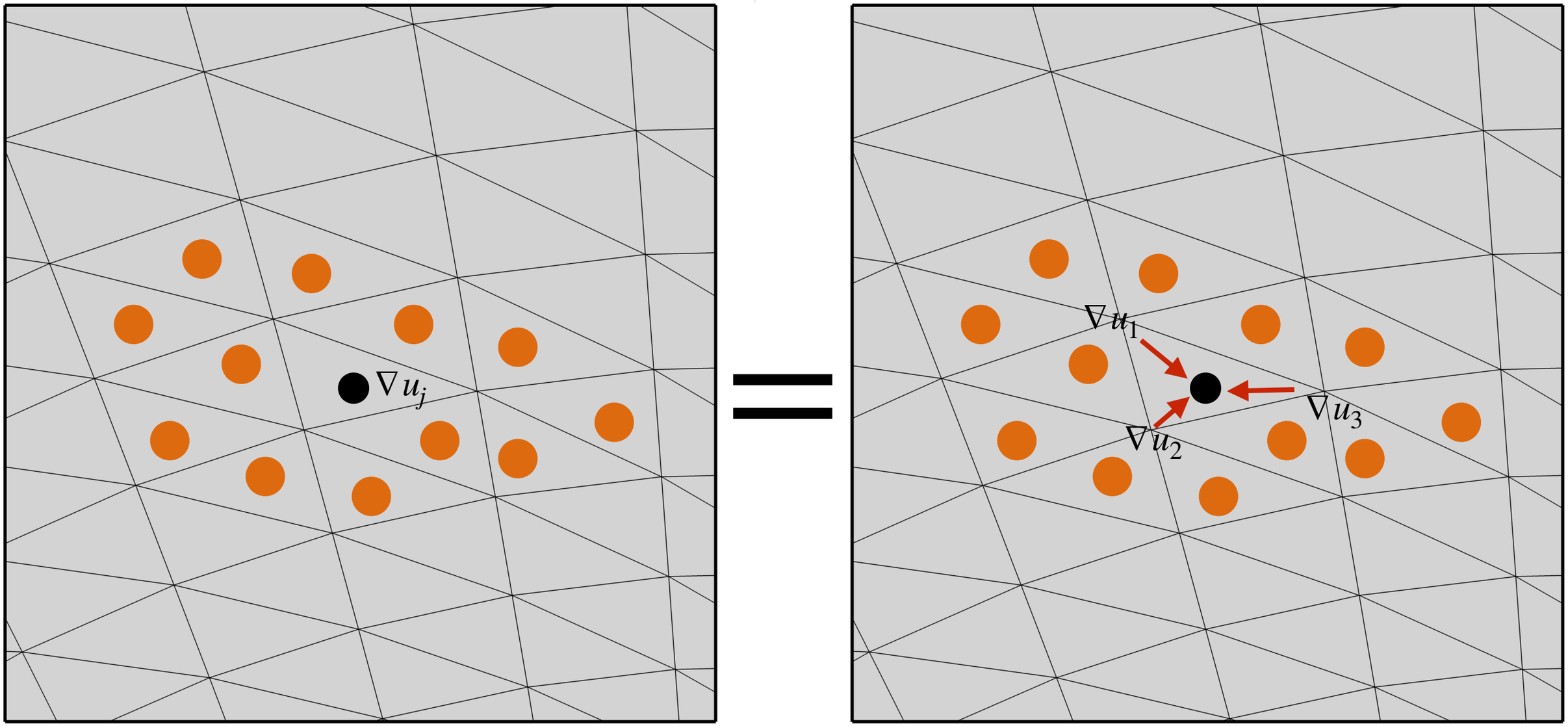
2. Large residual stencil



$$\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R) dS$$

End up with the same gradient stencil

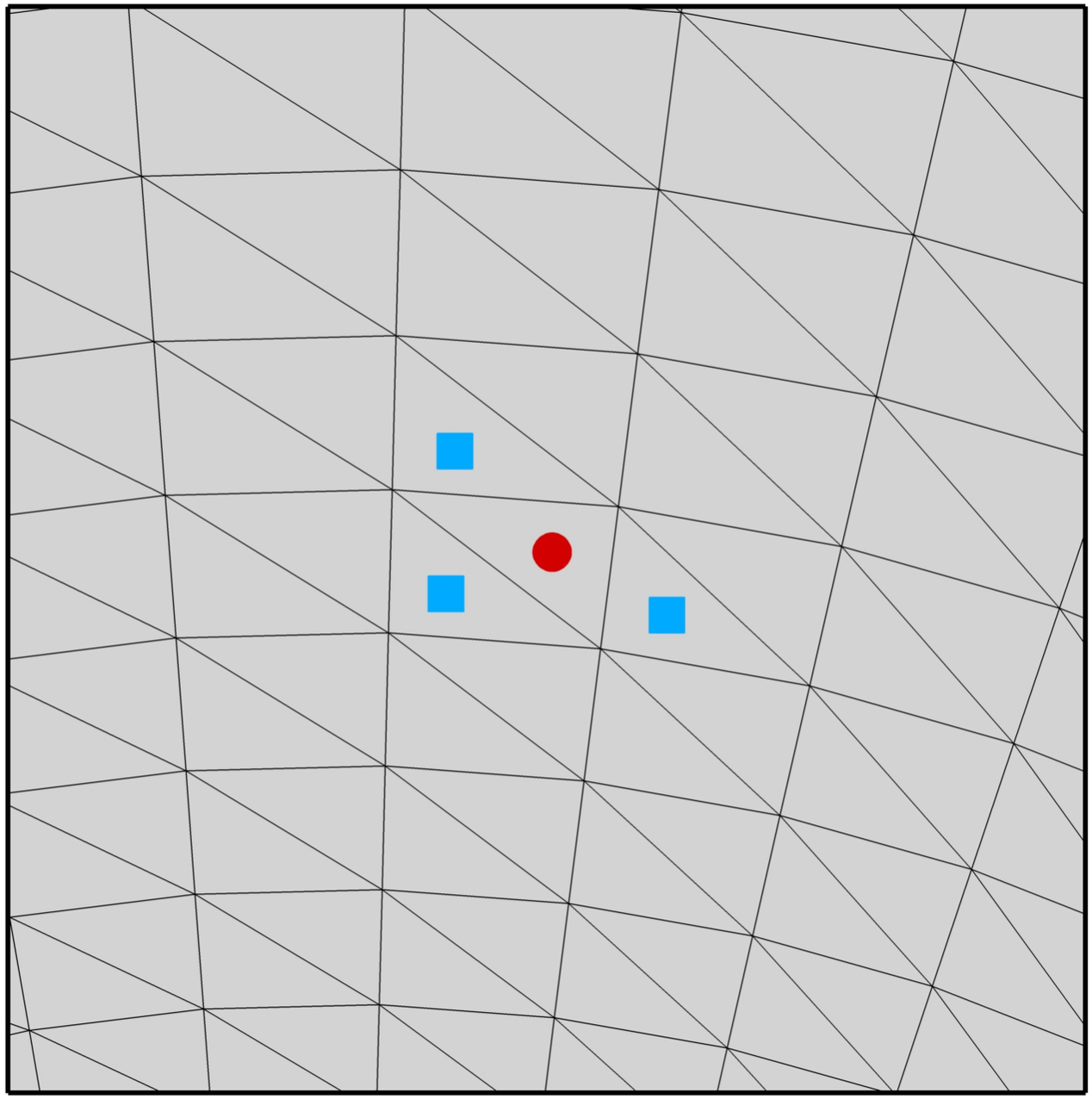
$$\mathbf{u}_L = \bar{\mathbf{u}}_j + \nabla \mathbf{u}_j \cdot (\mathbf{x}_m - \mathbf{x}_j)$$



12

12

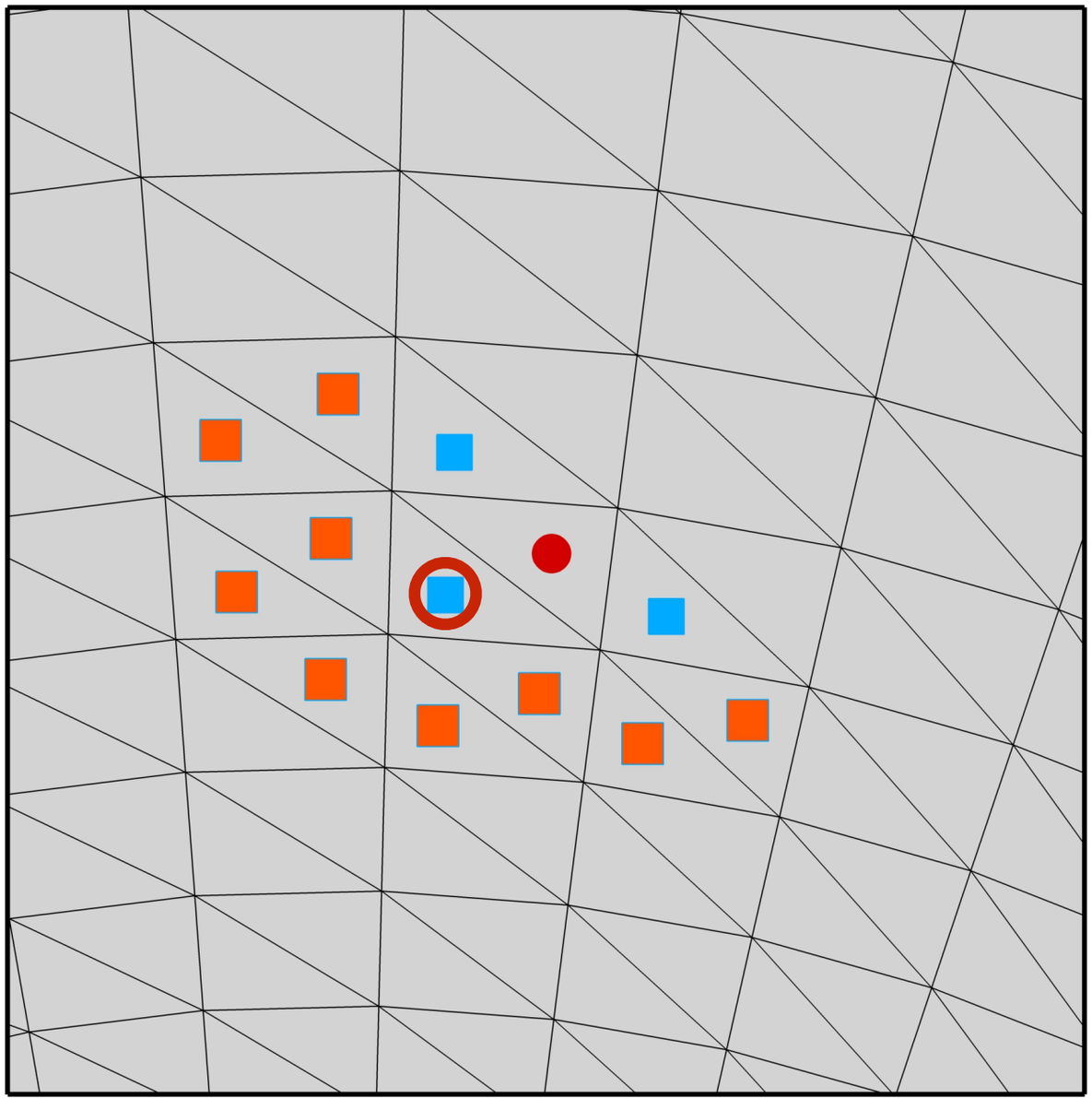
- $\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R) dS$



3

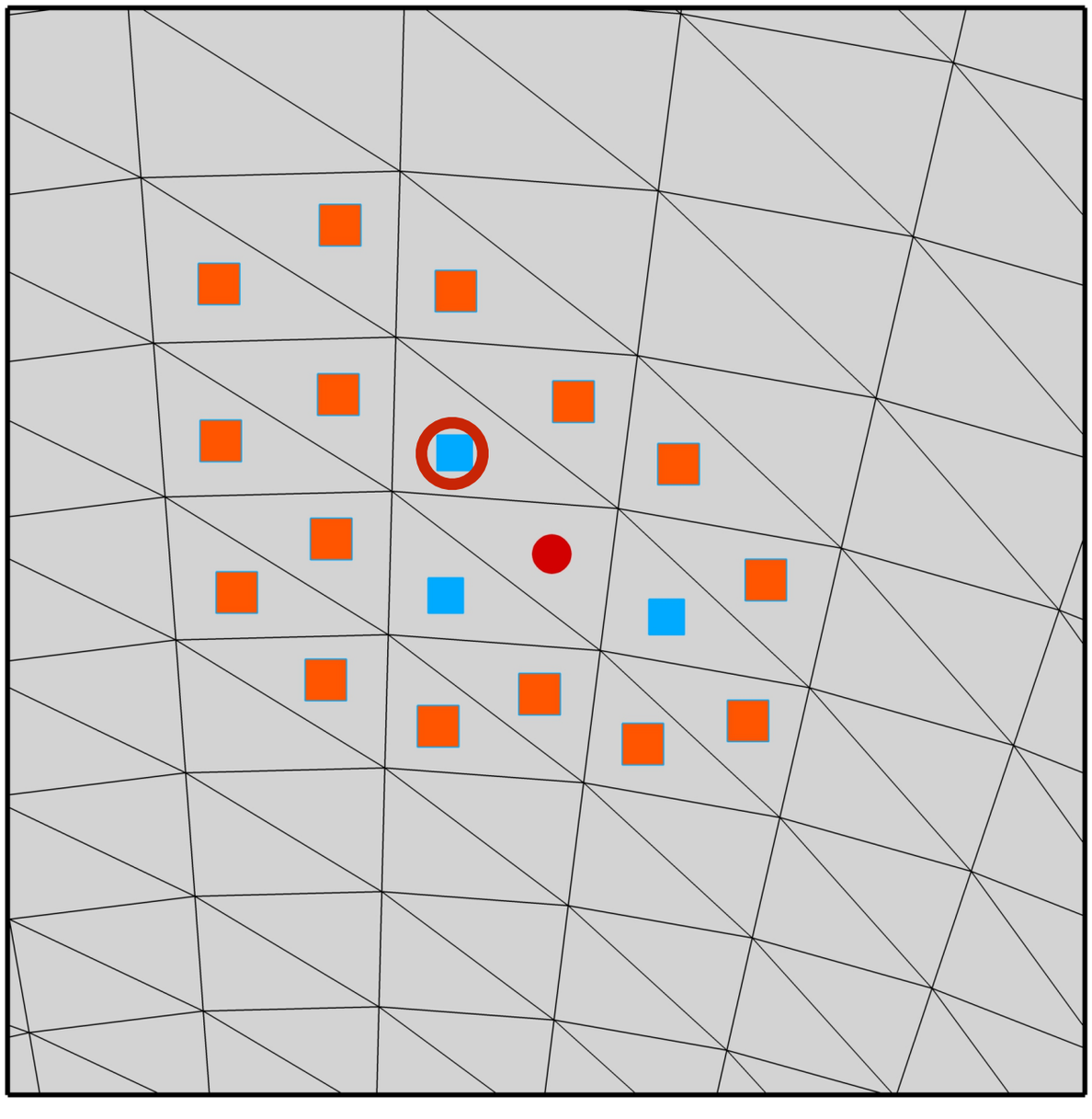
- $\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R) dS$

- $\mathbf{u}_R = \mathbf{u}_k + \nabla \mathbf{u}_k \cdot (\mathbf{x}_m - \mathbf{x}_k)$



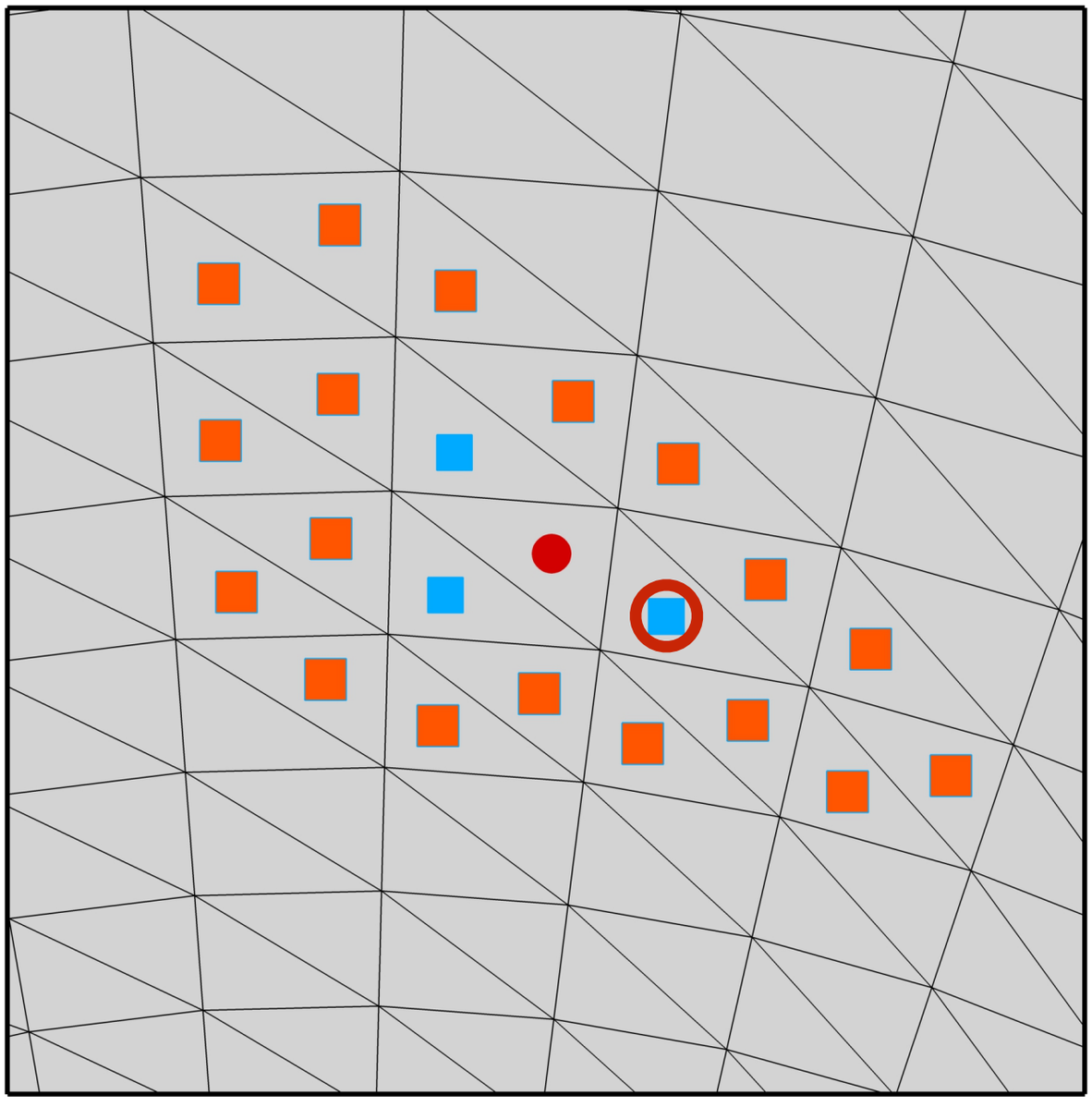
- $\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R) dS$

- $\mathbf{u}_R = \mathbf{u}_k + \nabla \mathbf{u}_k \cdot (\mathbf{x}_m - \mathbf{x}_k)$



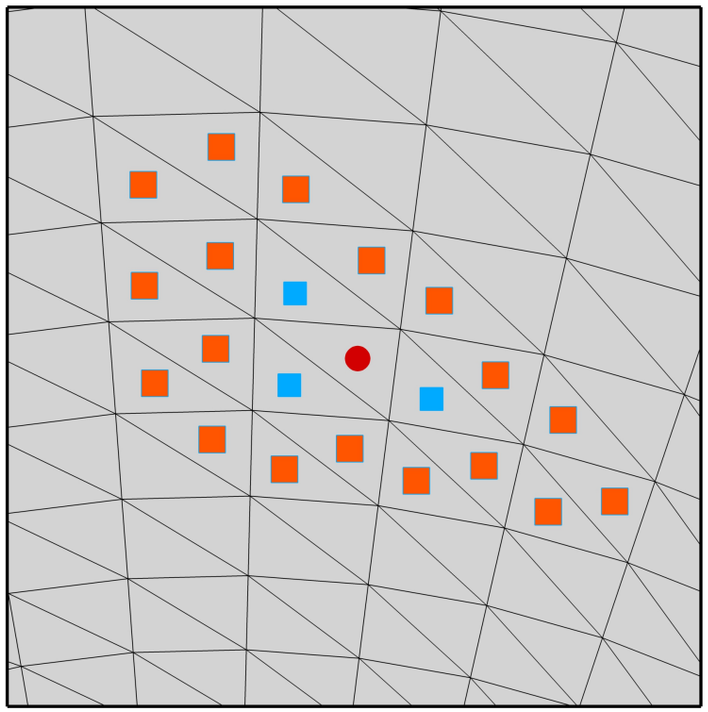
- $\mathbf{Res}_j = \sum_{k \in \{k_j\}} \Phi_n(\mathbf{u}_L, \mathbf{u}_R) dS$

- $\mathbf{u}_R = \mathbf{u}_k + \nabla \mathbf{u}_k \cdot (\mathbf{x}_m - \mathbf{x}_k)$

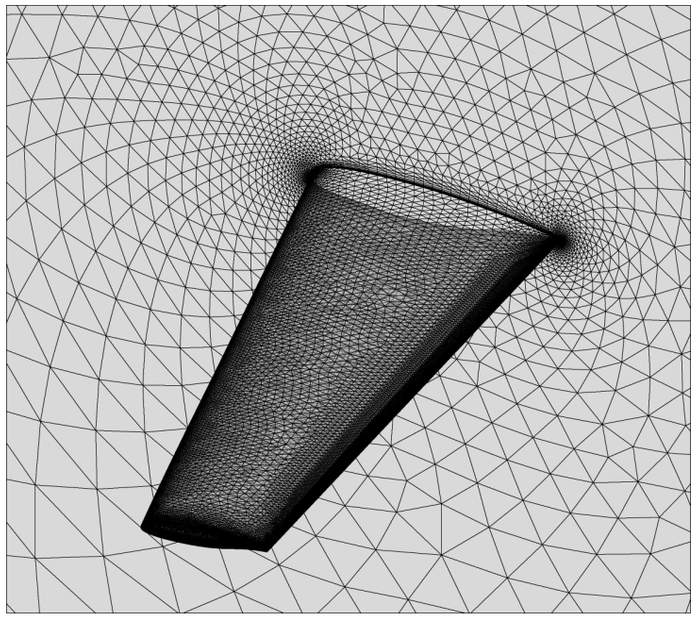


So, the residual stencil is as large as the original cell-centered method…

Large residual stencil = Large Jacobian



$$\frac{\partial \mathbf{Res}}{\partial \mathbf{U}}_{j,k} = (0 \overbrace{\underset{k=1}{\times} \quad \underset{k=2}{\times} \quad \times \quad \cdots \quad \times \quad \times}^{\mathbf{21}} \quad \cdots \quad 0)$$

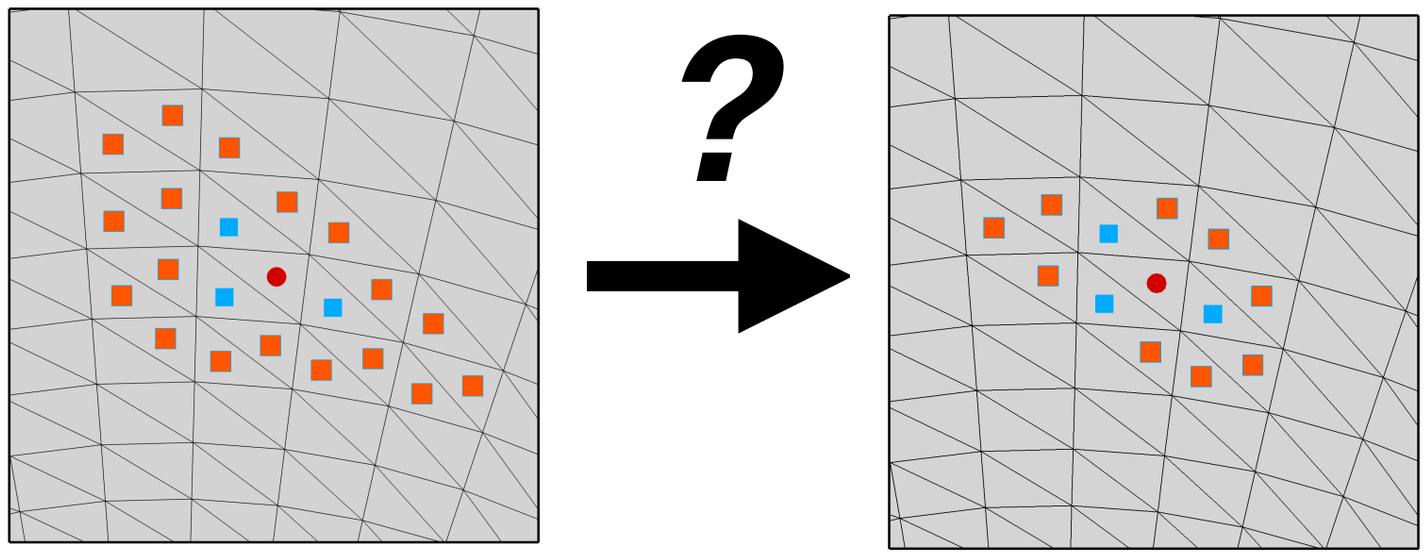


$$\frac{\partial \mathbf{Res}}{\partial \mathbf{U}}_{j,k} = (0 \overbrace{\times \quad \times \quad \times \quad \cdots \quad \times \quad \times}^{\mathbf{\sim 100}} \quad \cdots \quad 0)$$

Better to reduce it for adjoint, Newton solver, etc.

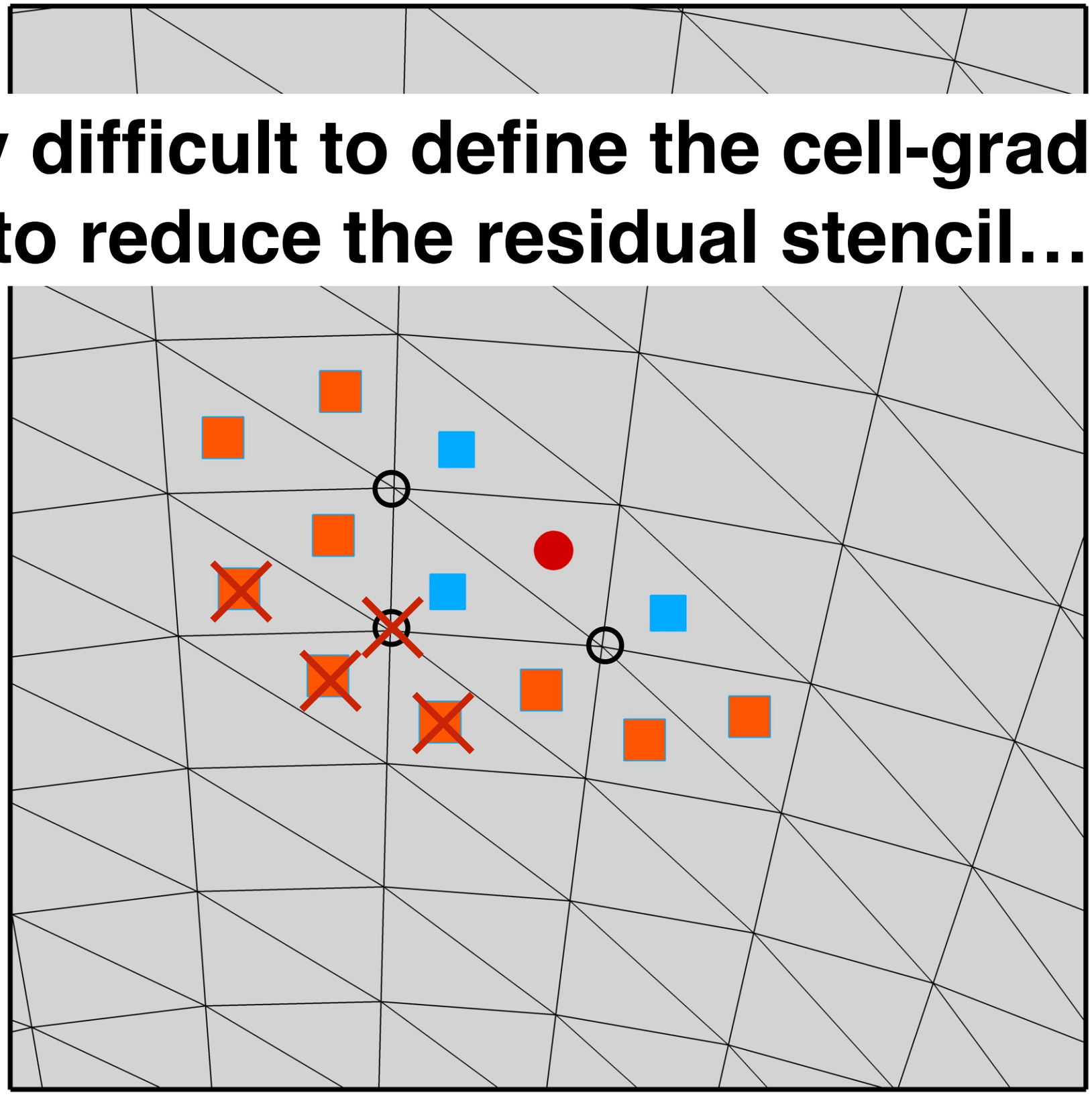
How can we reduce the residual stencil?

- $\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R) dS$



$$\frac{\partial \mathbf{Res}}{\partial \mathbf{U}} = \begin{vmatrix} x\ x\ x\ x\ x\ \ldots x\ x\ x \\ x\ x\ x\ x \ldots\ x\ x\ x \\ x\ x\ x\ \ldots\ x\ x\ x\ x \\ x\ x\ x\ x\ \ldots\ x\ x\ x \\ \ddots \\ x\ x\ x\ x\ \ldots\ x\ x\ x\ x \end{vmatrix} \quad \frac{\partial \mathbf{Res}}{\partial \mathbf{U}} = \begin{vmatrix} x\ x\ \ldots x \\ x\ \ldots x \\ x\ \ldots x\ x \\ x\ x \ldots\ x \\ \ddots \\ x\ x\ \ldots x \end{vmatrix}$$

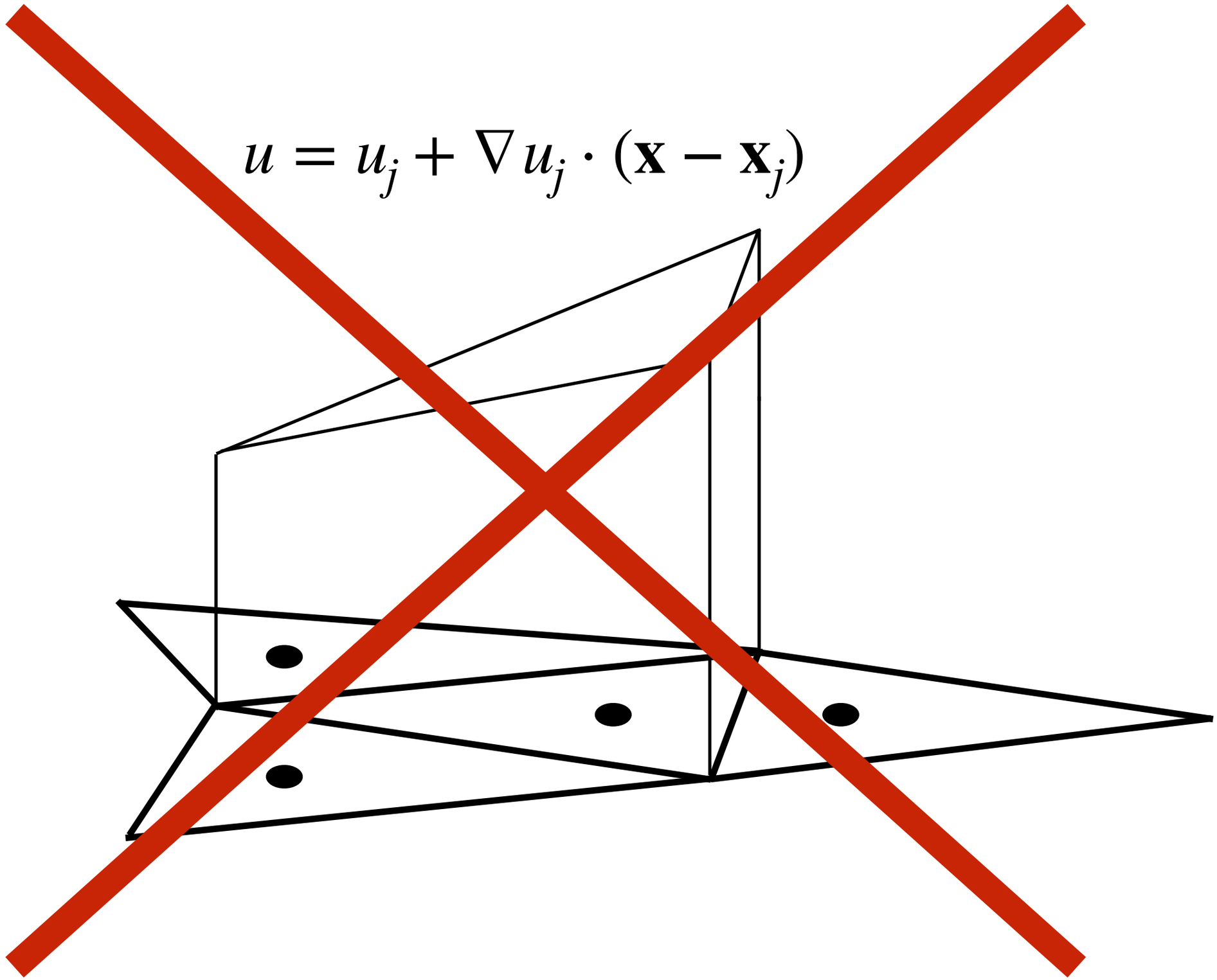
Efficient reconstruction scheme



Very difficult to define the cell-gradient to reduce the residual stencil…

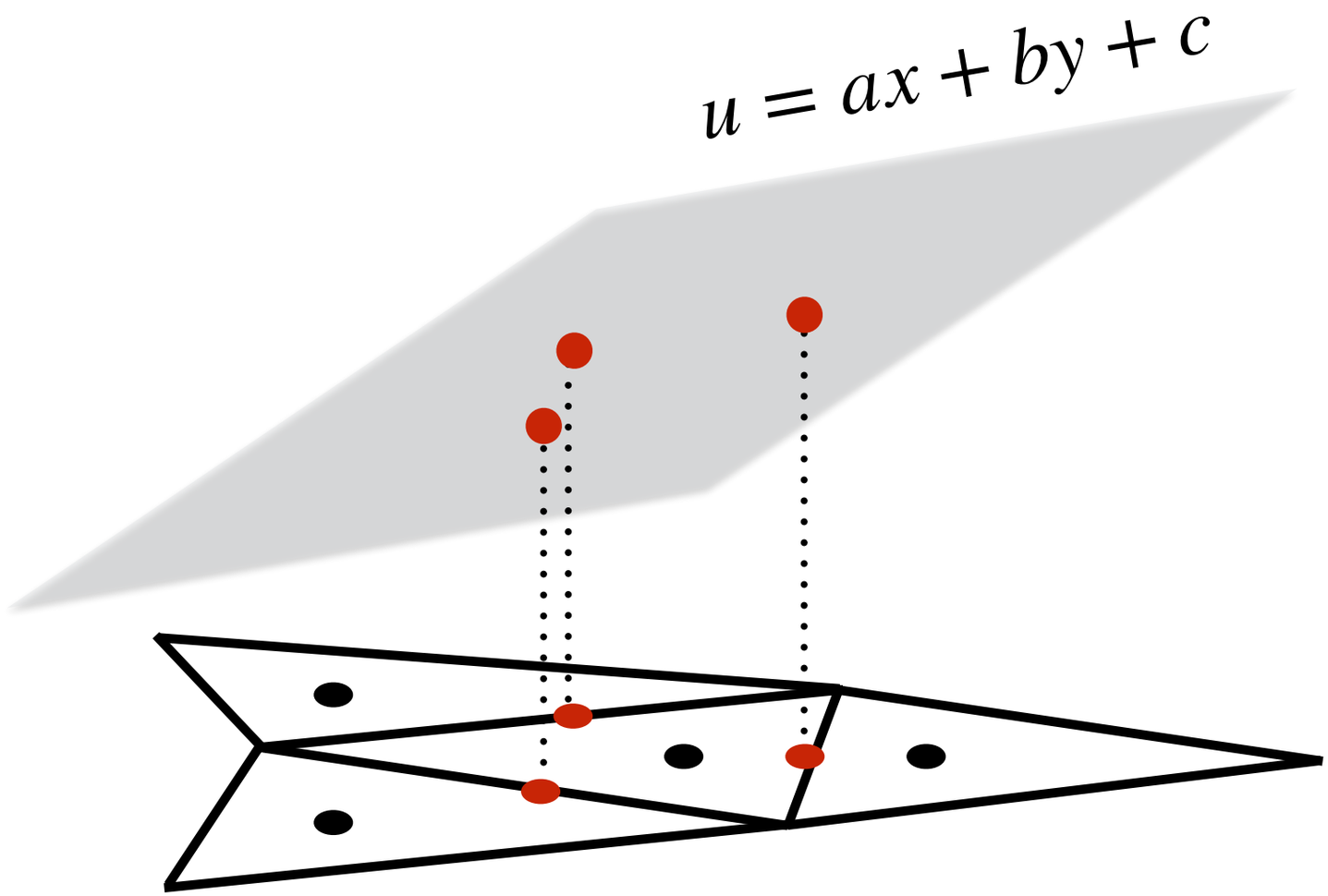
Stop thinking about cell gradients

$$u = u_j + \nabla u_j \cdot (\mathbf{x} - \mathbf{x}_j)$$

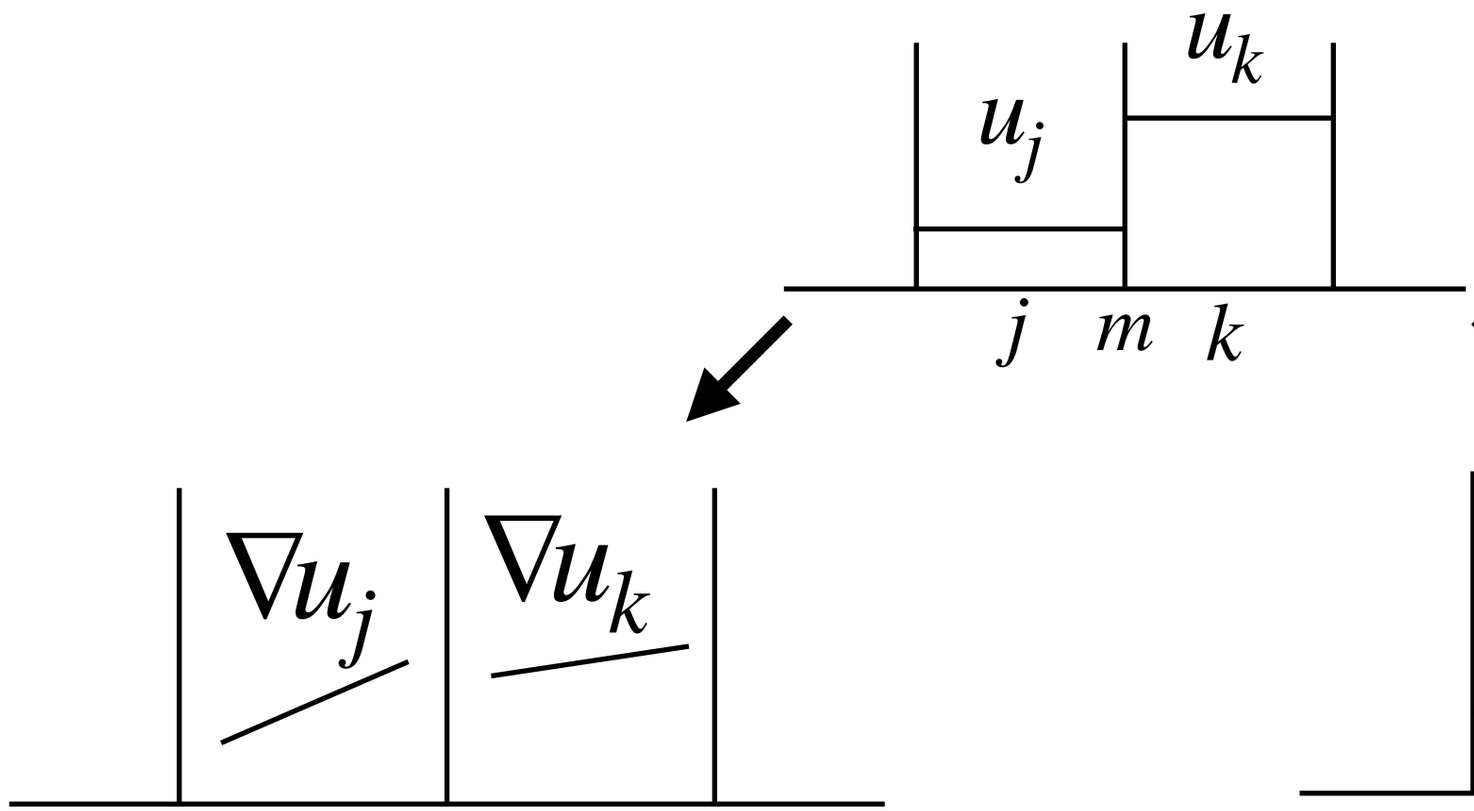


Focus on reconstructed values

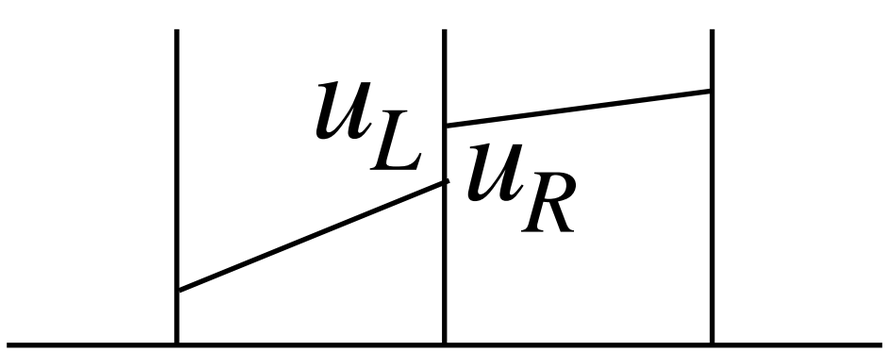
What is essential is to reconstruct the solution such that it is exact for linear functions. The linear polynomial in a cell is not essential.

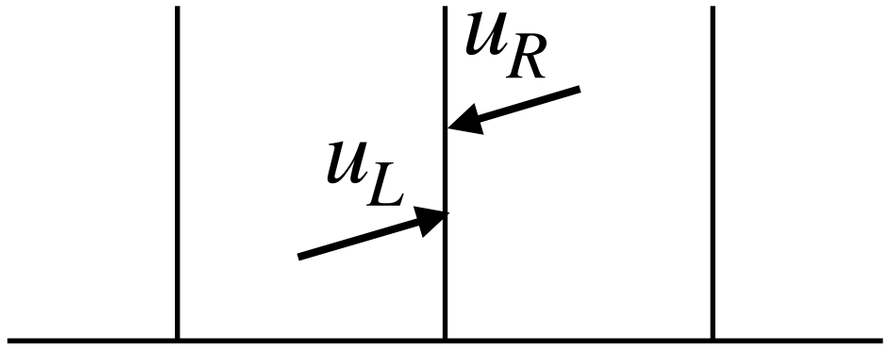


Define the gradient at the face not at the cell



This can still be exact for linear functions



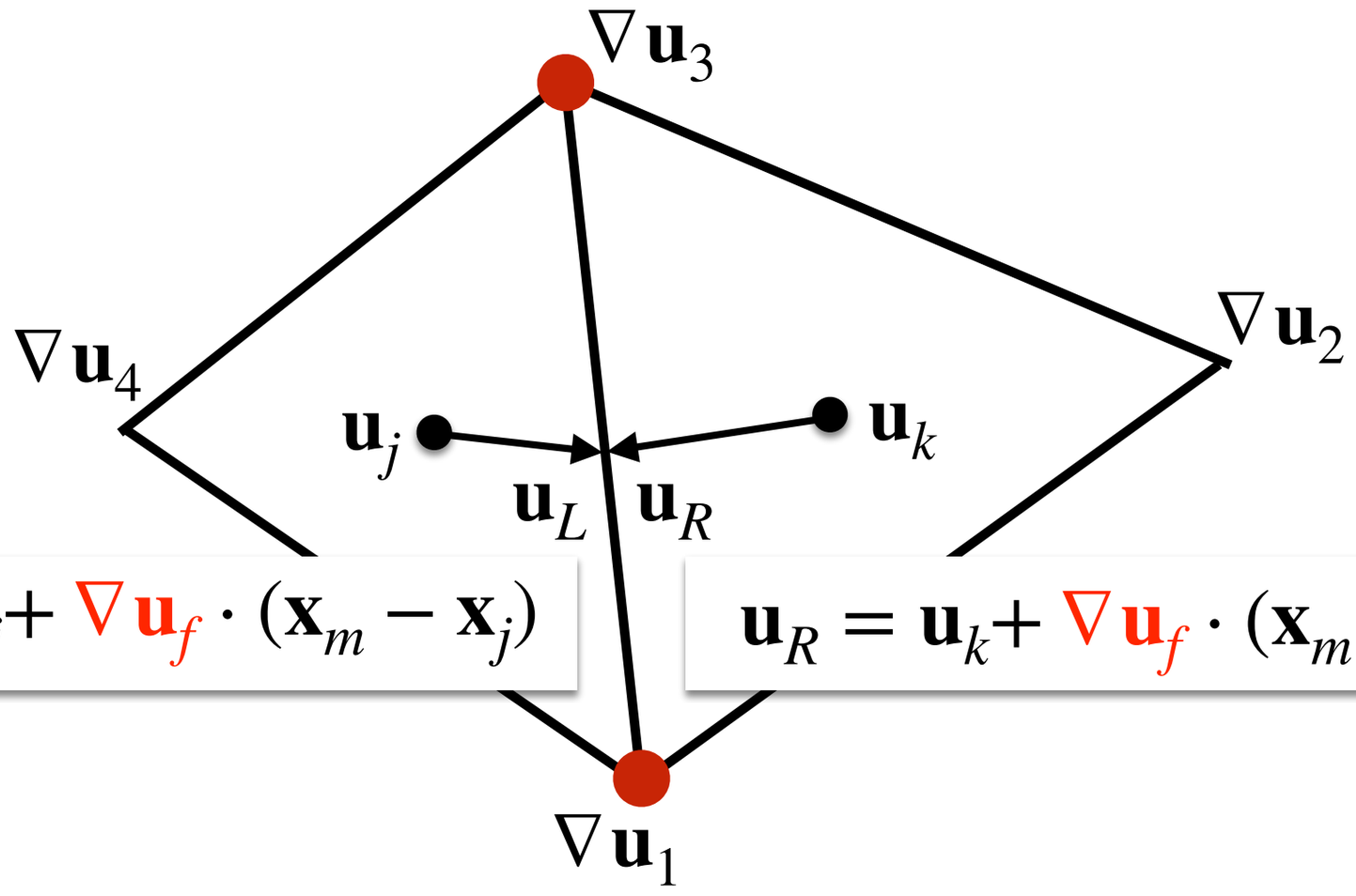


$$u_L = u_j + \nabla u_j \cdot (\mathbf{x}_m - \mathbf{x}_j)$$
$$u_R = u_k + \nabla u_k \cdot (\mathbf{x}_m - \mathbf{x}_k)$$

$$u_L = u_j + \nabla u_f \cdot (\mathbf{x}_m - \mathbf{x}_j)$$
$$u_R = u_k + \nabla u_f \cdot (\mathbf{x}_m - \mathbf{x}_k)$$

Reconstruction with common gradients

We propose to define the face gradient by averaging and use it in both reconstructions from the left and right cells

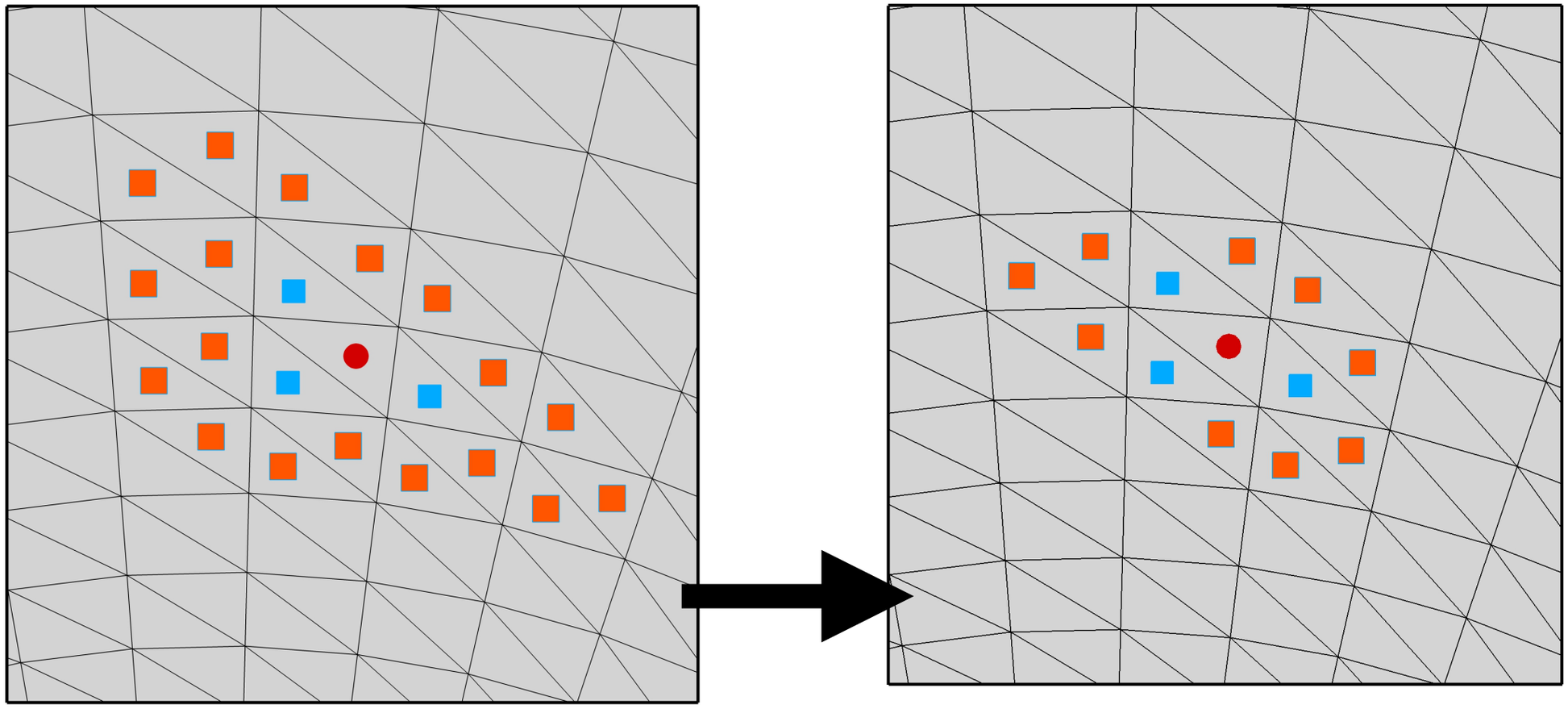


F-ANG : ***F**ace-**A**veraged **N**odal **G**radient*

$$\nabla \mathbf{u}_f = \frac{1}{2}\left(\nabla \mathbf{u}_1 + \nabla \mathbf{u}_3\right)$$

Residual stencil is dramatically reduced

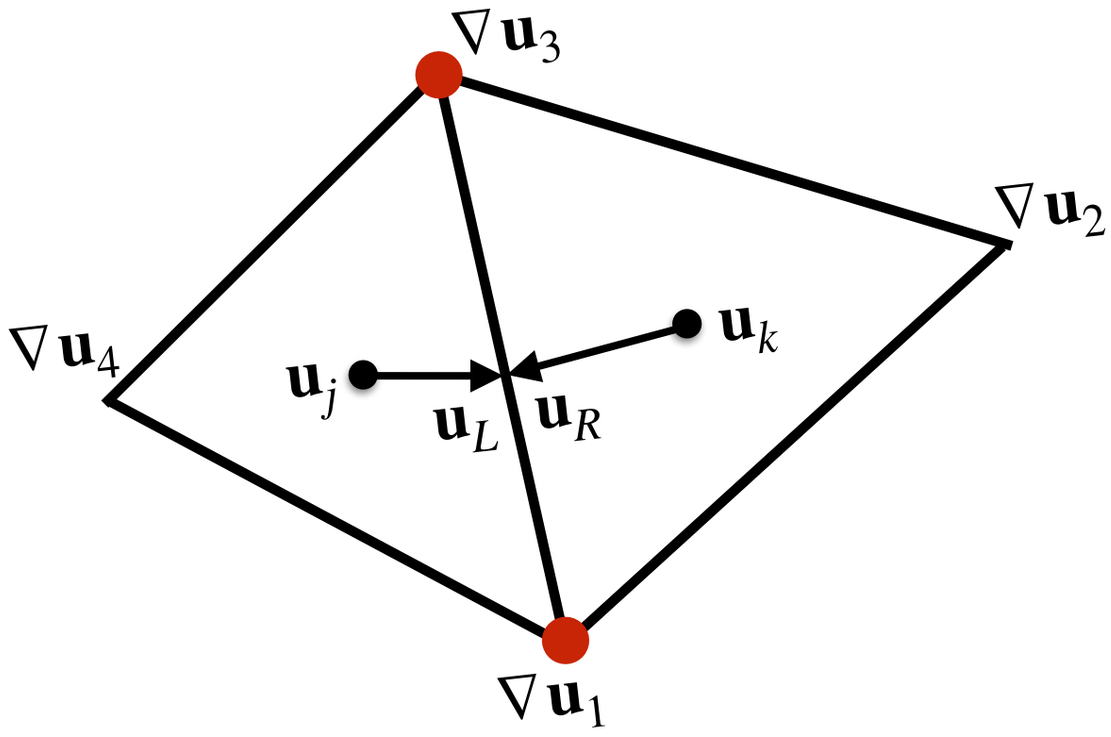
$$\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_n(\mathbf{u}_L, \mathbf{u}_R) dS$$

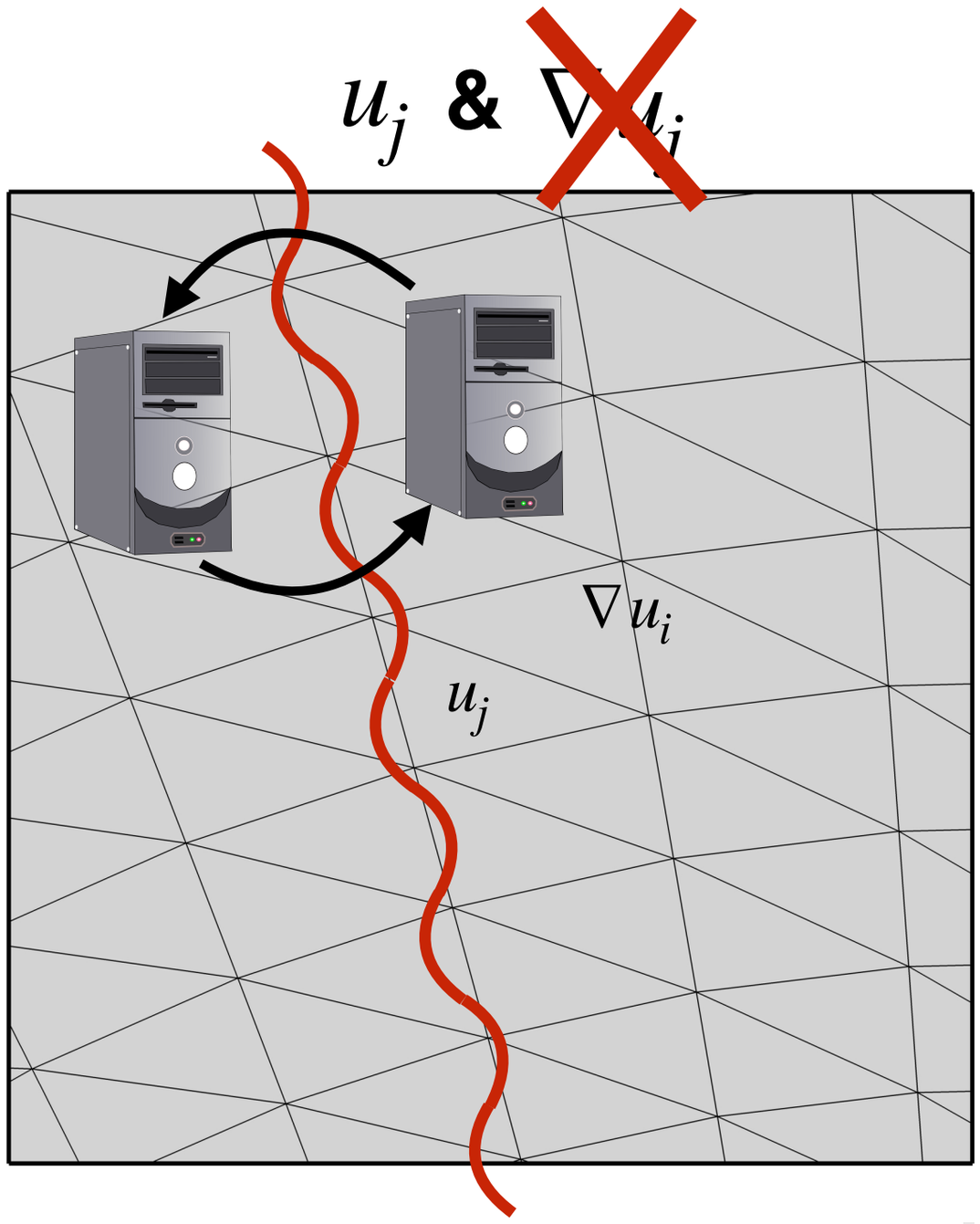


21

12 !

Communication is not needed for gradients



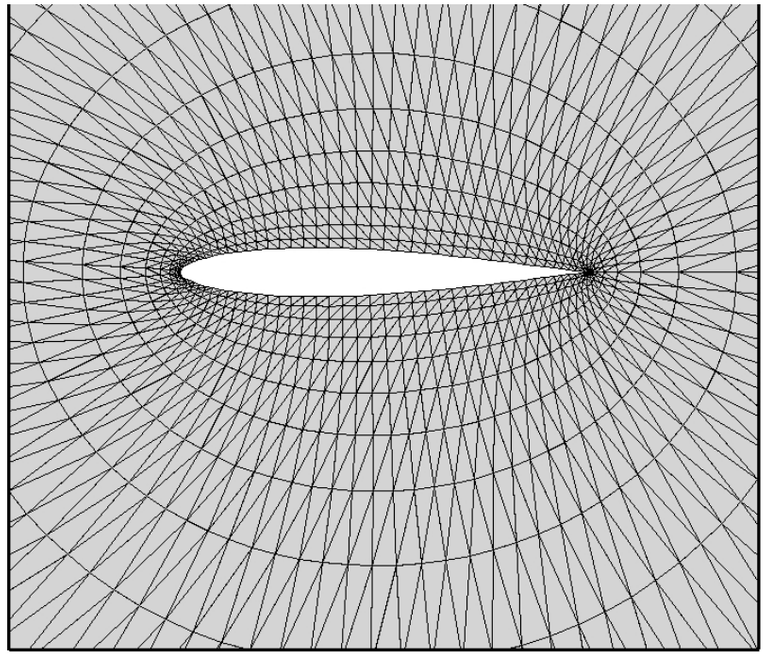


Results

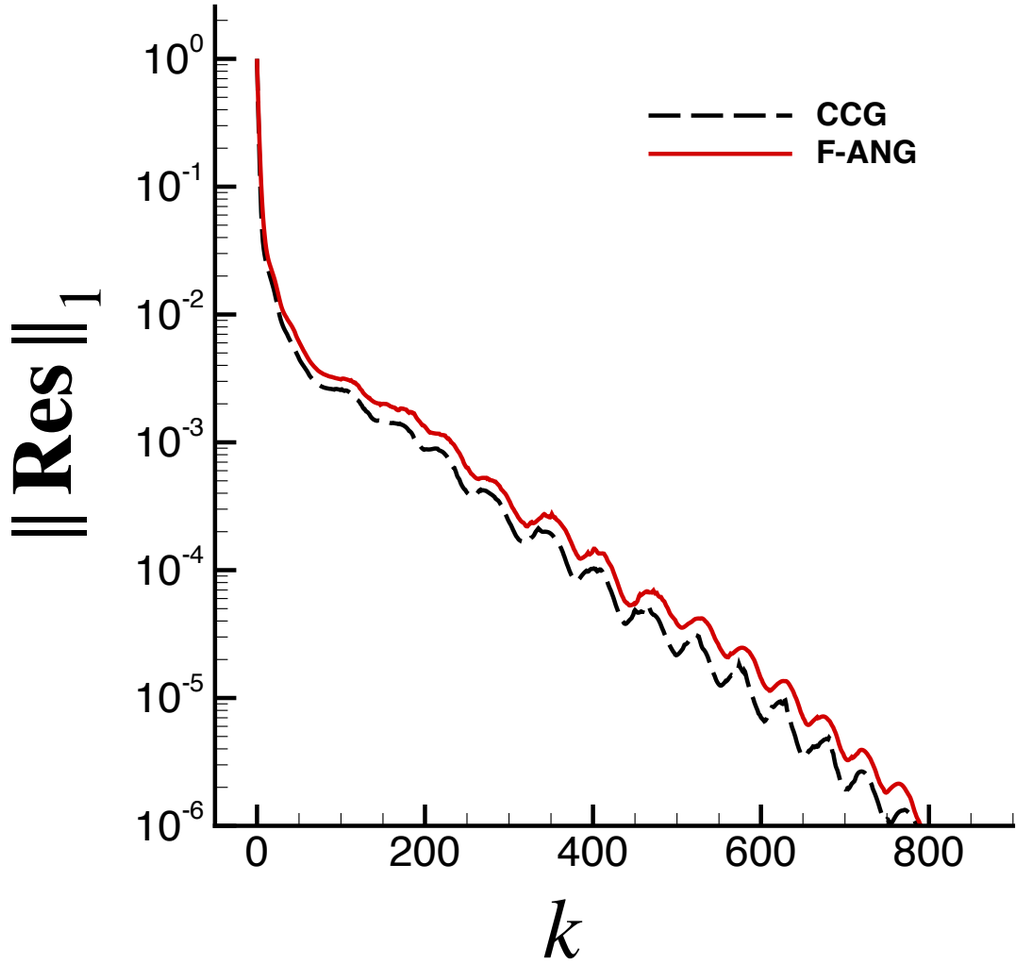
Inviscid: Transonic flow over an airfoil

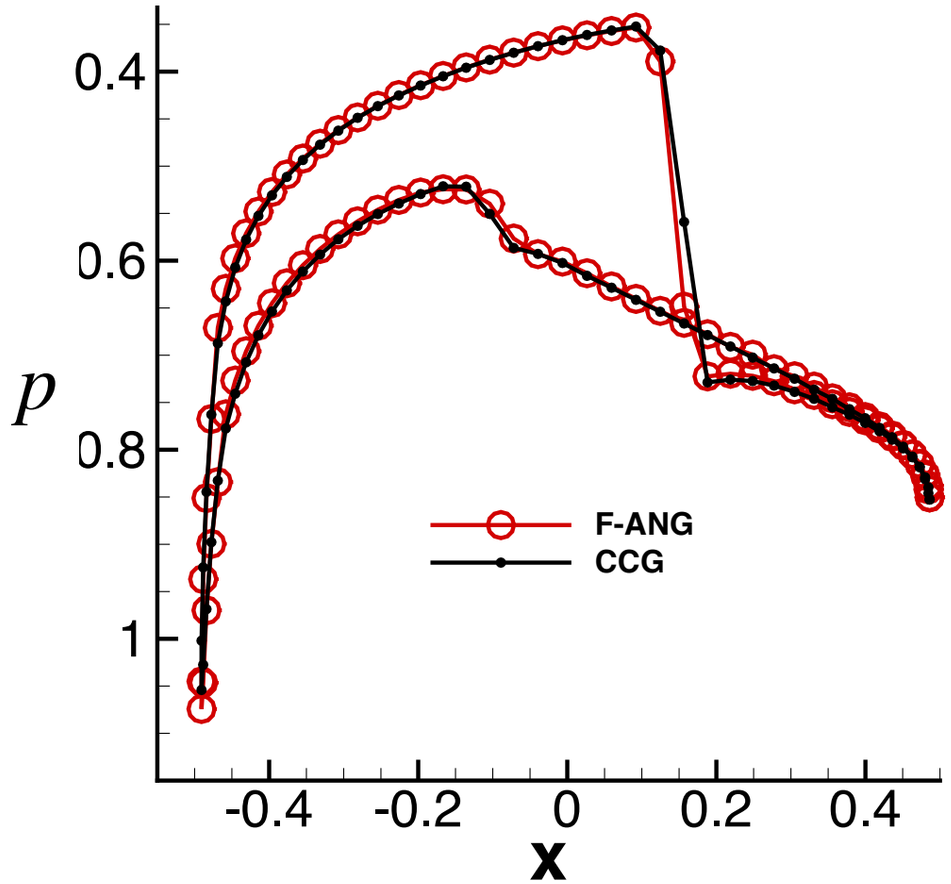
$$M_\infty = 0.85$$

$$\alpha = 1.25$$

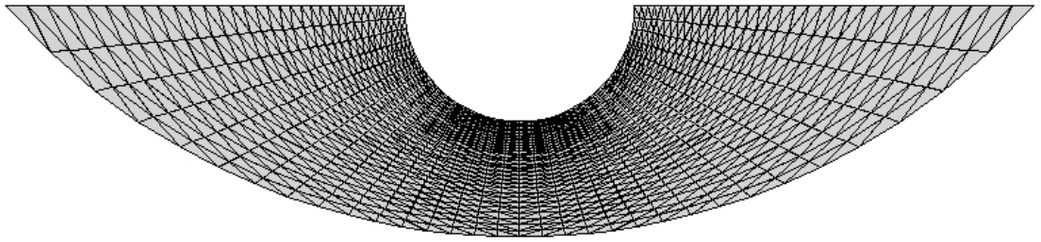


F-ANG works equally well



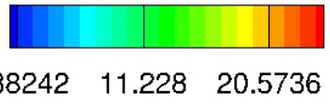


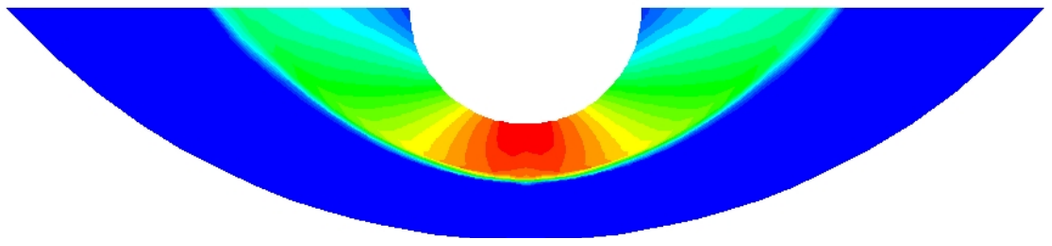
Inviscid: hypersonic flow over a cylinder



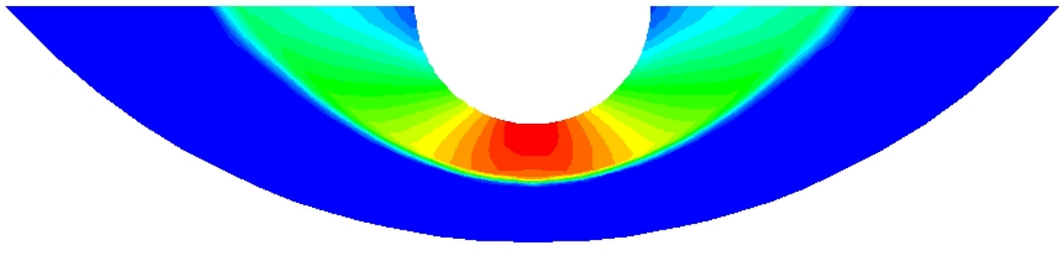
$$M_\infty = 5.2$$

$$\alpha = 0$$

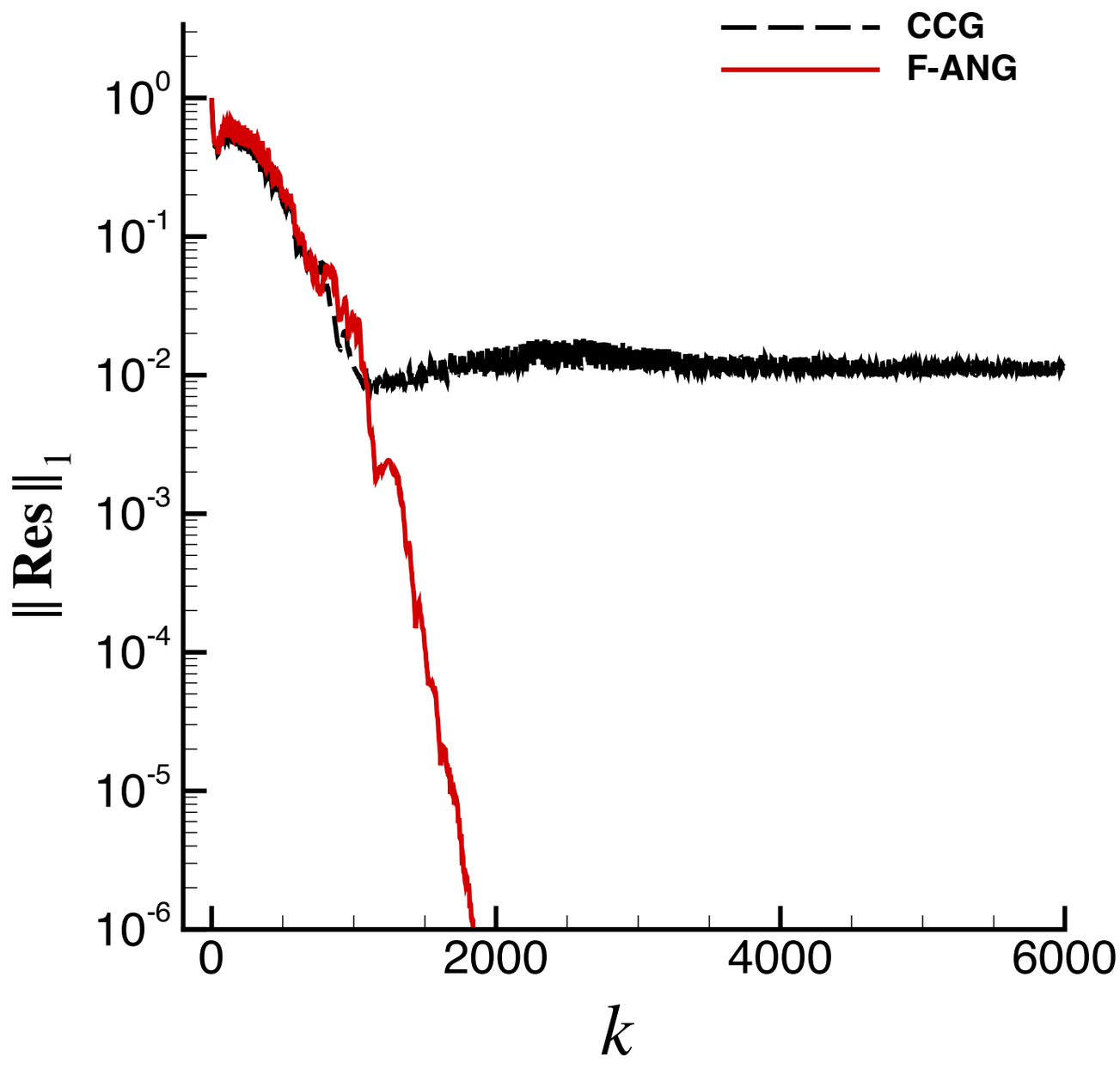




CCG - HLL



F-ANG - HLL

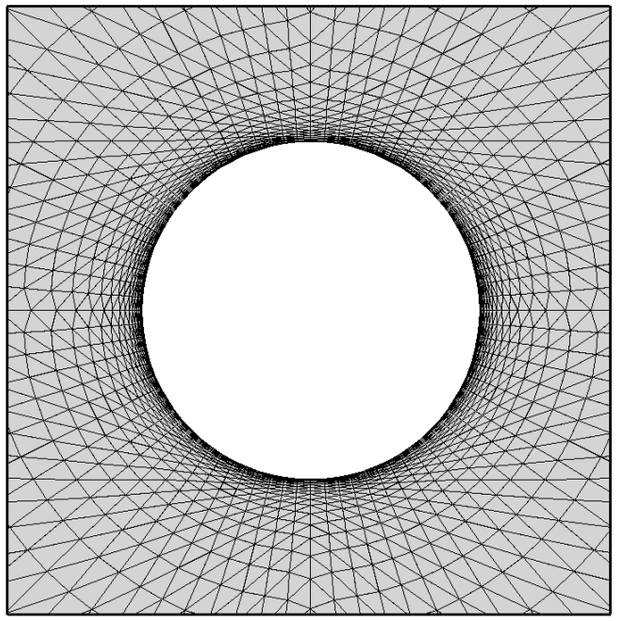


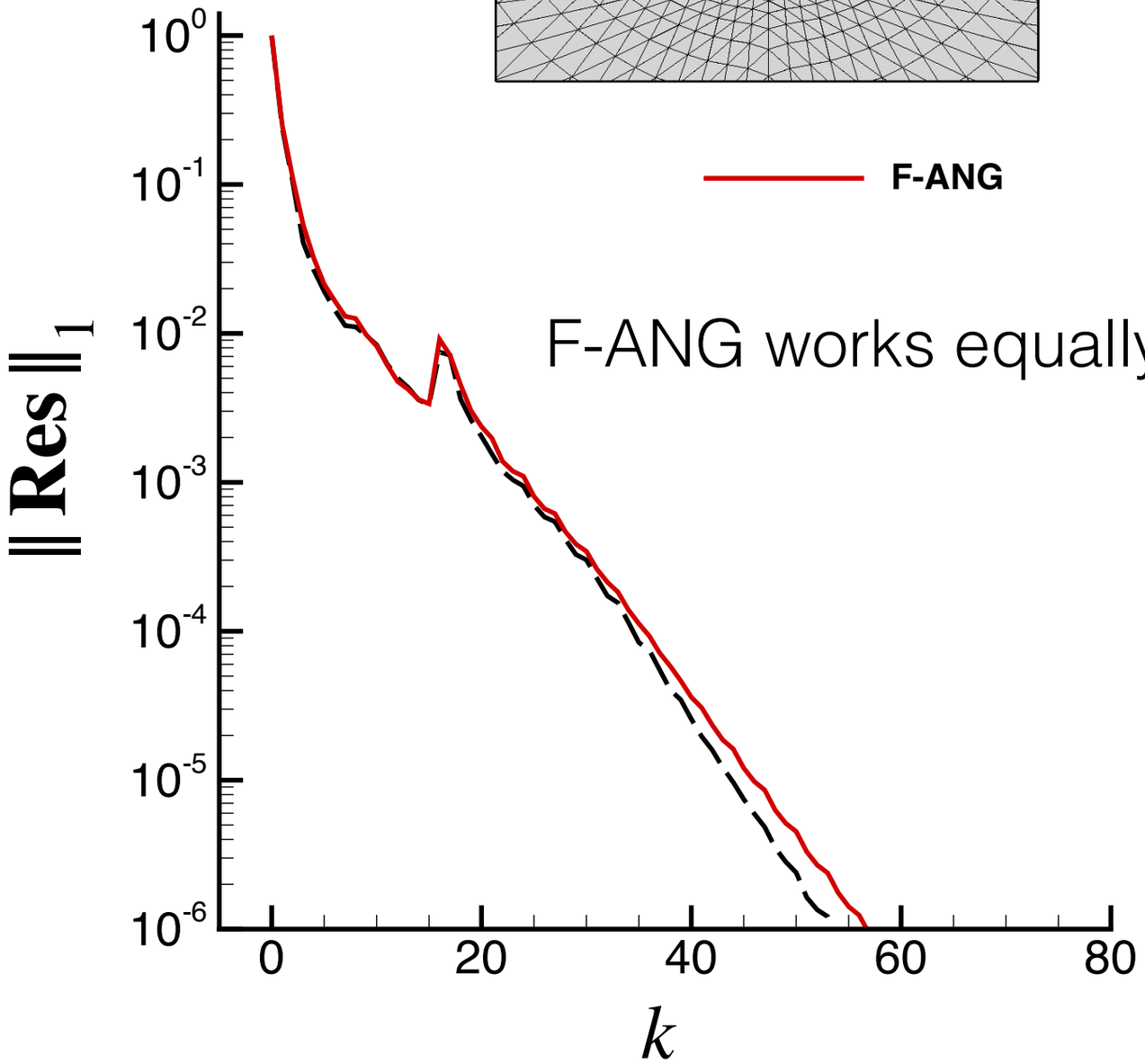
Better convergence with F-ANG

Viscous flow over a cylinder

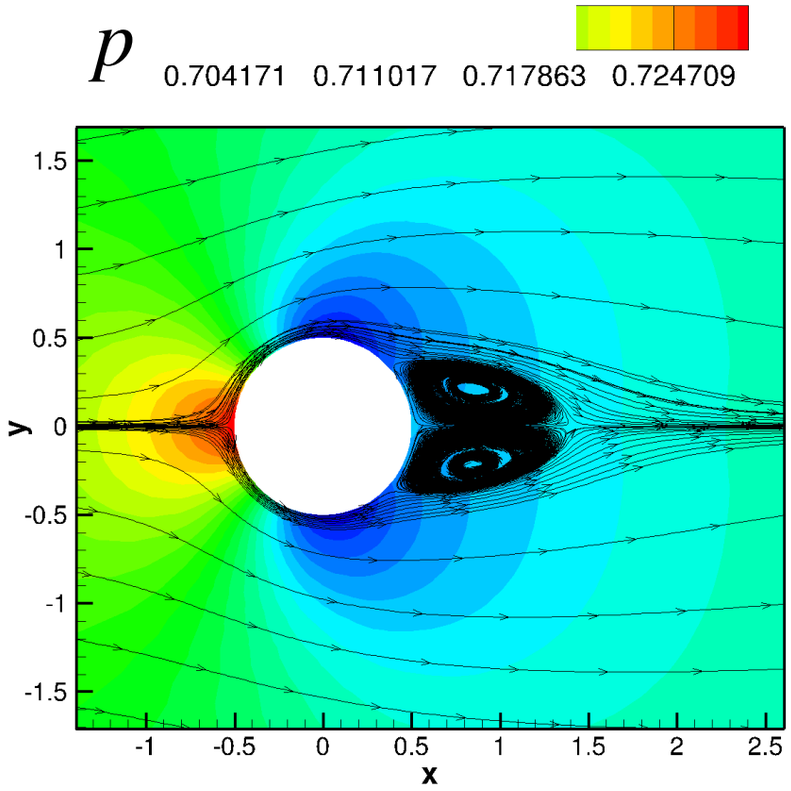
$$M_\infty = 0.3$$

$$Re_\infty = 40$$

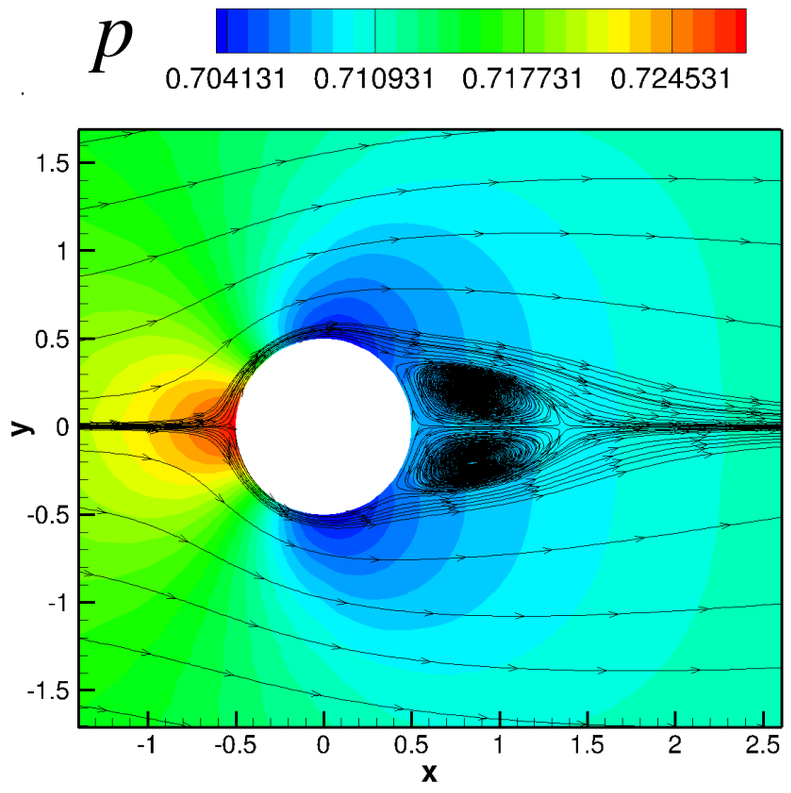




CCG



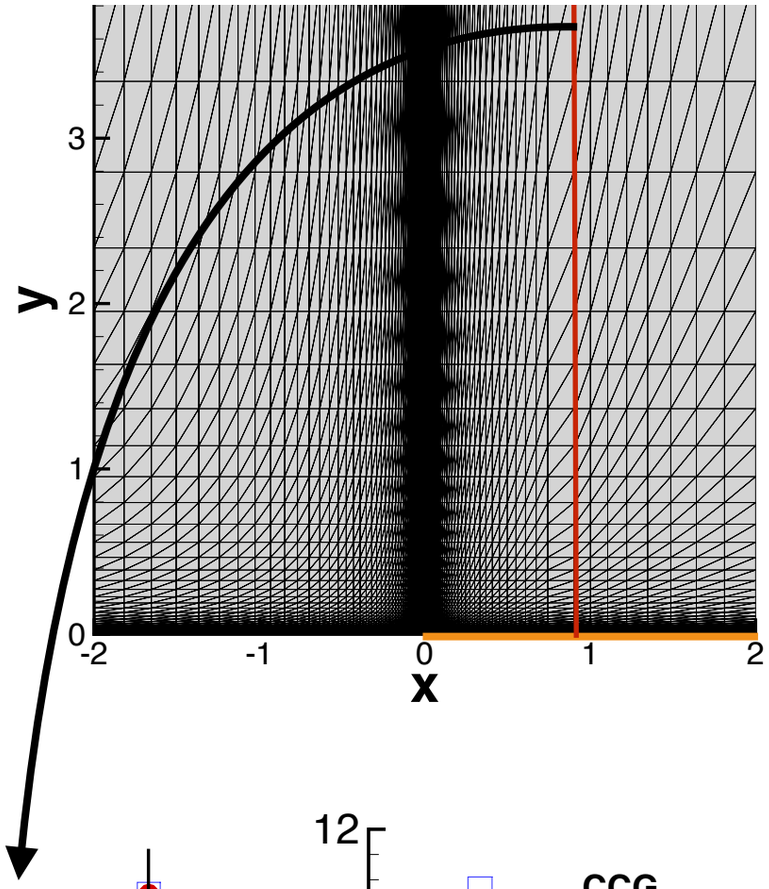
F-ANG

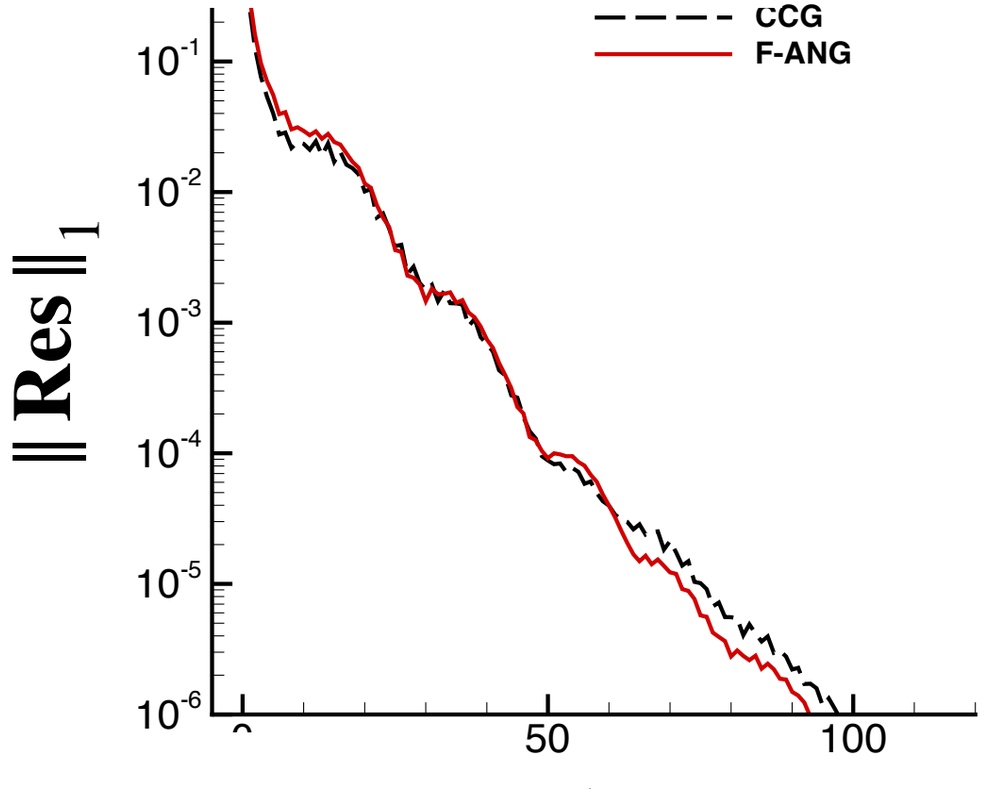


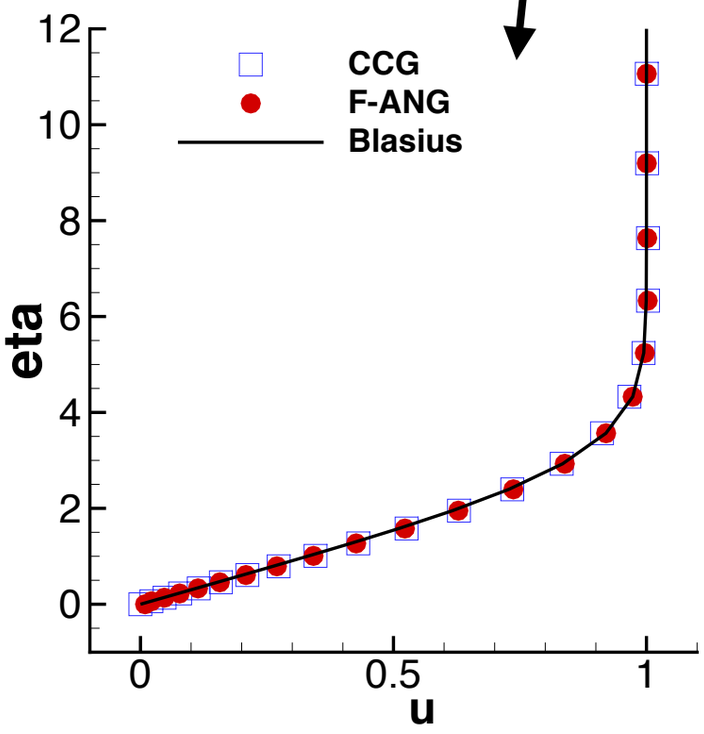
Viscous flow over a flat plate

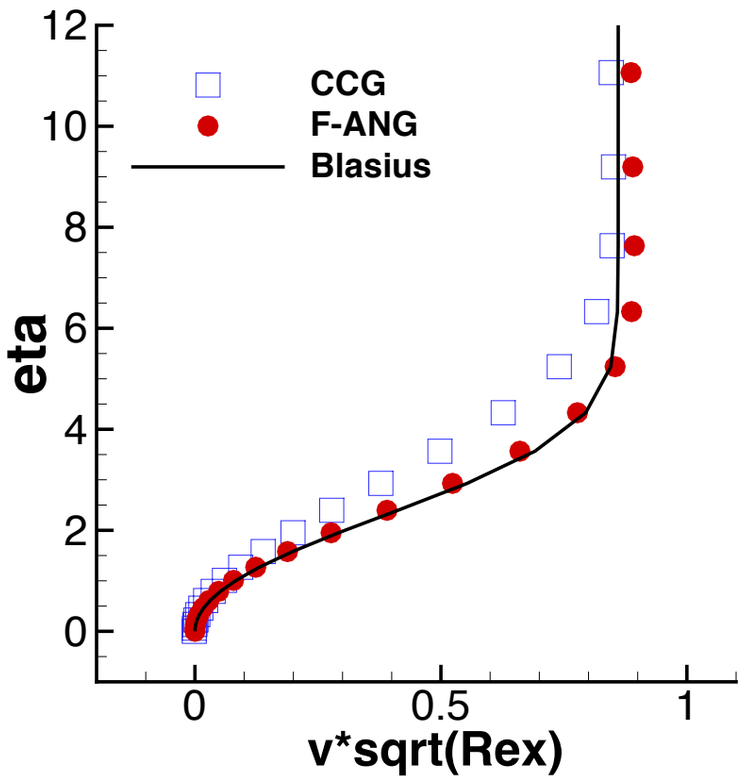
$$M_\infty = 0.15$$

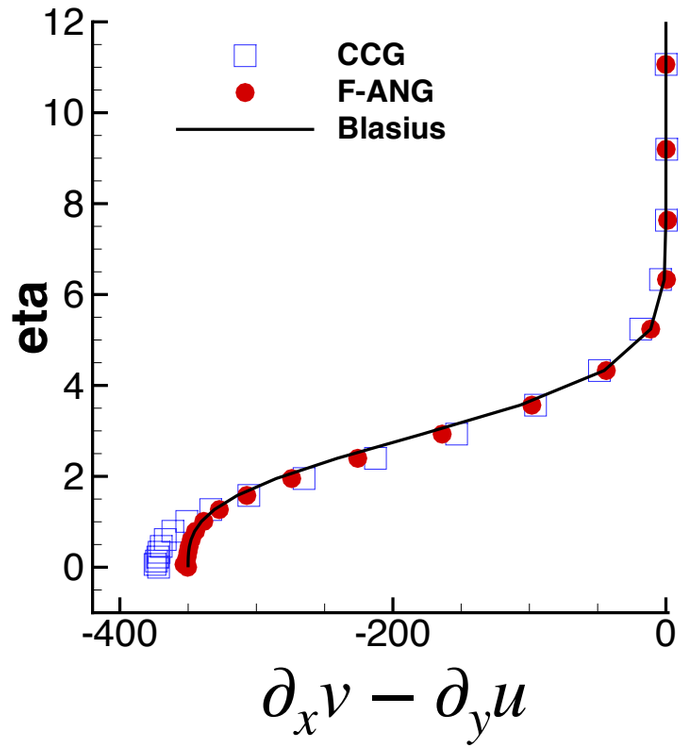
$$Re_\infty = 10^6$$



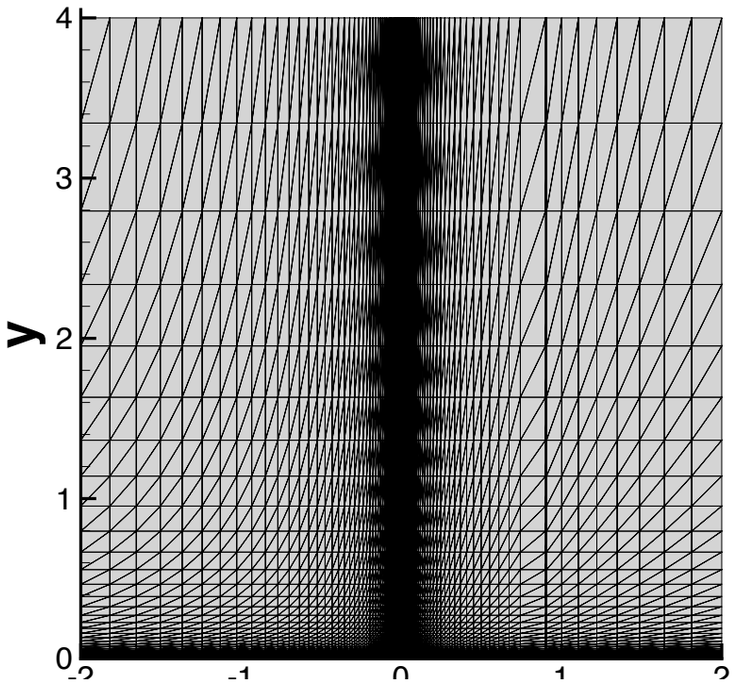








F-ANG converges better and produces more accurate solutions



$[c^x_{jk}, c^y_{jk}]$

	Total	Min	Mean	Max
CCG(vn)	133,554(1.00)	3	11.0	12
F-ANG	34,040(0.25)	3	5.0	6

-75%

$$\frac{\partial \mathbf{Res}}{\partial \mathbf{U}}_{j,k}$$

	Total	Min	Mean	Max
CCG(vn)	243,792(1.00)	7	21.5	22
F-ANG	144,896(0.59)	4	12.8	13

-40%

21.5: x x x x x ...x x x

↓

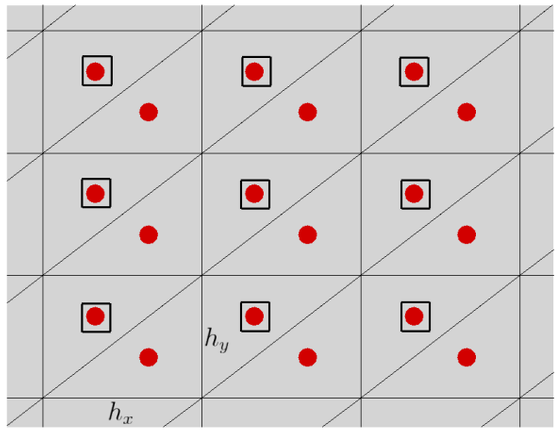
12.8: x x ..x

Cost of gradients and Jacobian size are greatly reduced

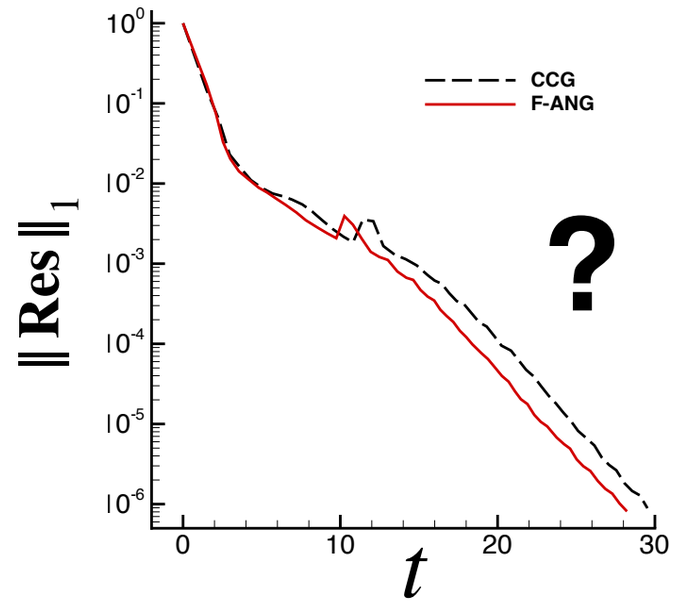
Future work

The work presented is just a preliminary study to investigate its potential; there is much to be done:

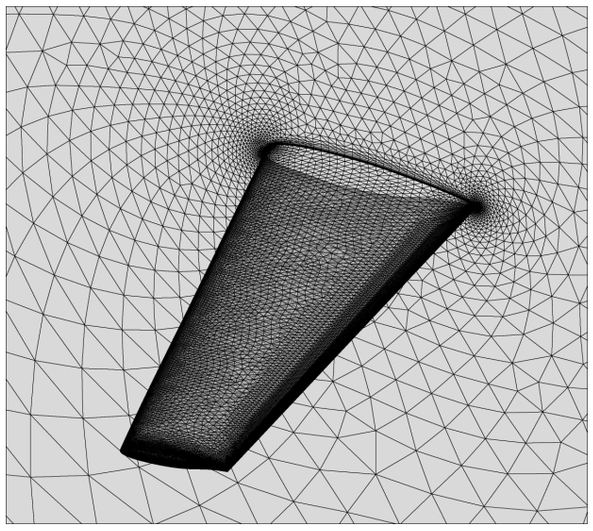
- Proof of convergence
- Actual saving in CPU time

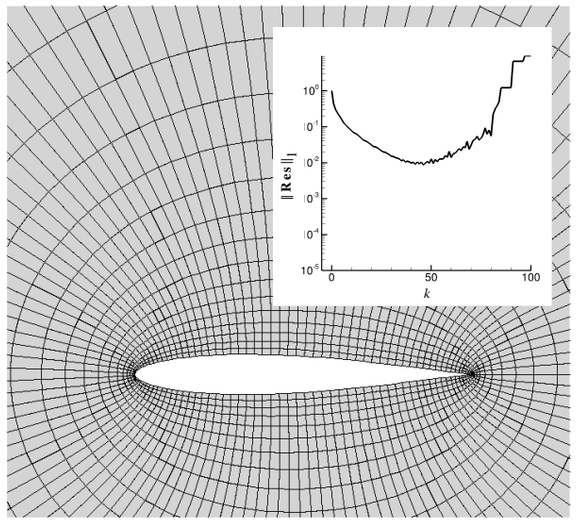


$$\rho(\mathbf{I} - \mathbf{J}^{-1}\mathbf{R}) < 1?$$

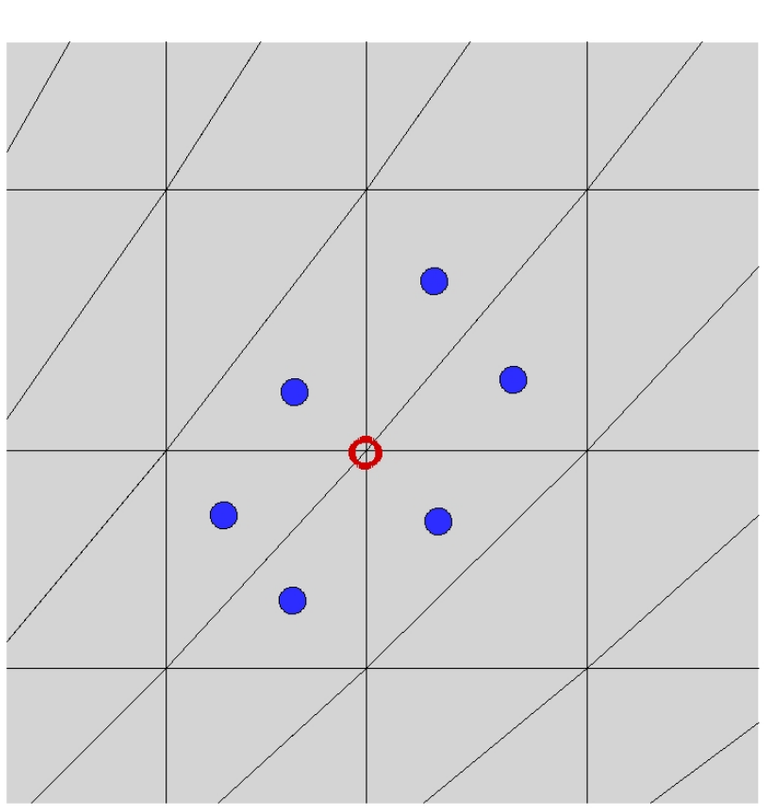
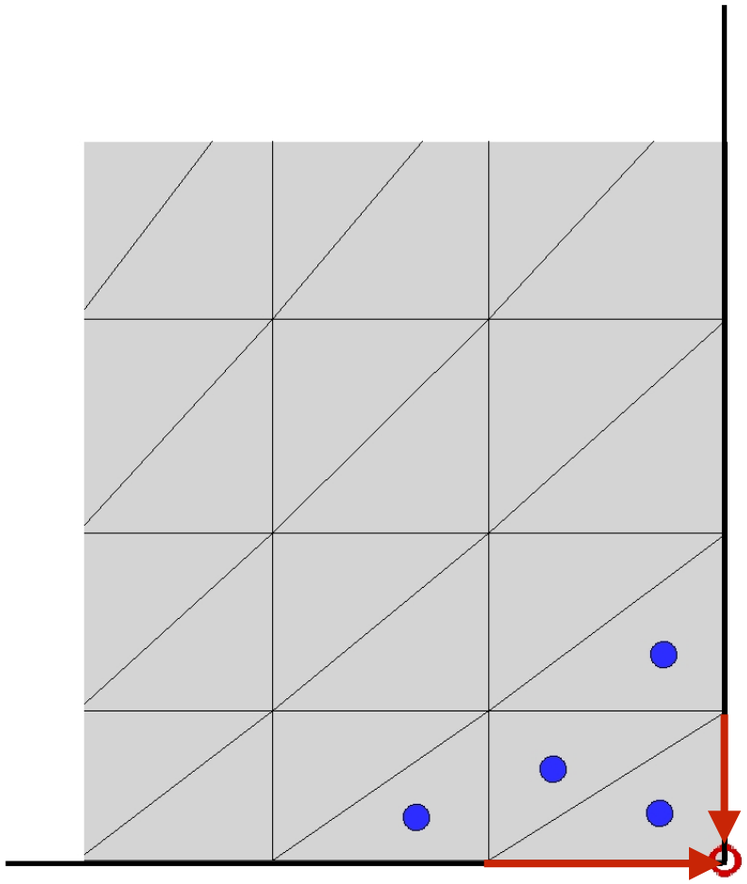
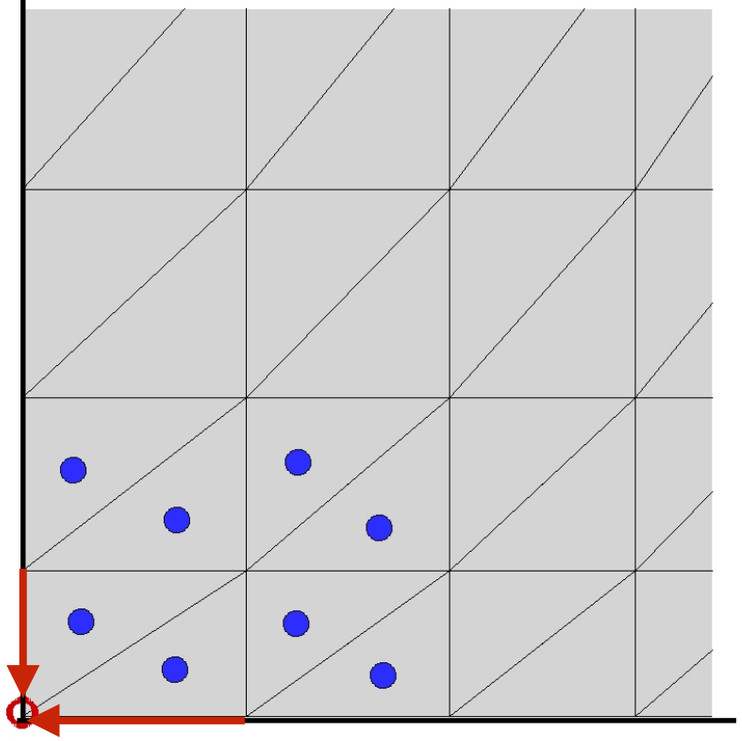


- 3D tetrahedral grids
- Extensions to mixed grids





Extra slide:

$$\begin{bmatrix} u_i \\ \partial_x u_i \\ \partial_y u_i \end{bmatrix} = \sum_k \begin{bmatrix} c_{ij}^x \\ c_{ij}^x \\ c_{ij}^y \end{bmatrix} \bar{u}_j$$

$$nnghbrs \geq 3$$