Theory of SEM discretisation and code implementation

Niclas Jansson
PDC Centre for HPC

Based on material by:

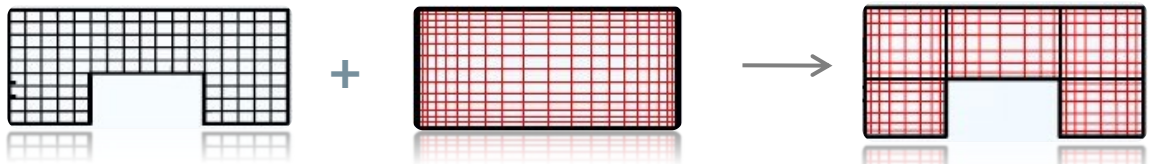
Paul Fischer (UIUC), **Aleks Obabko** (ANL),
Philipp Schlatter (KTH), etc…

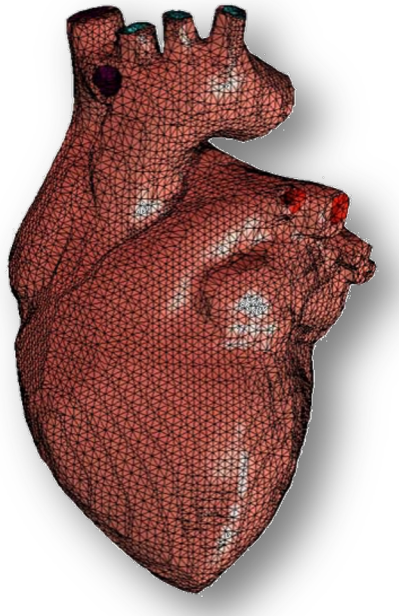
Outline

- Spectral Element Method overview
- Solver outline
- Code implementation

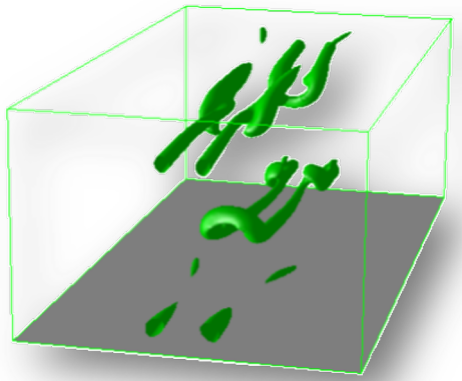
Spectral Element Method

Finite element (FEM) Spectral Spectral element (SEM), Patera 1984

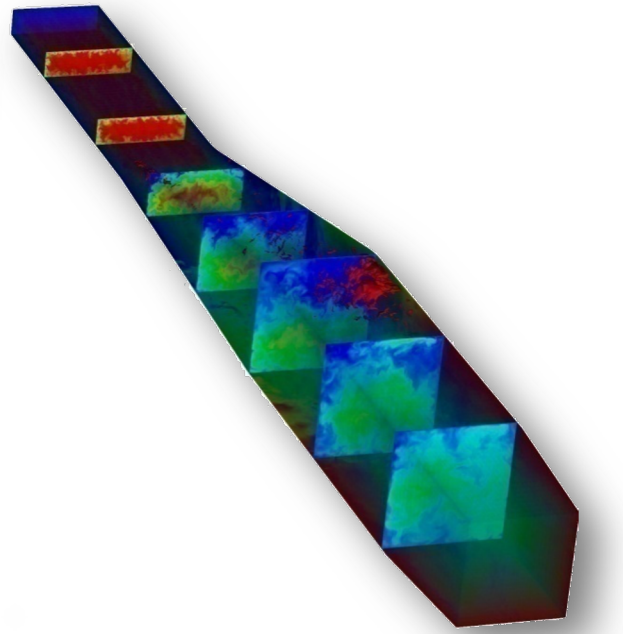




Typically low order



”infinite” order



Locally ”infinite” order

Why high-order?

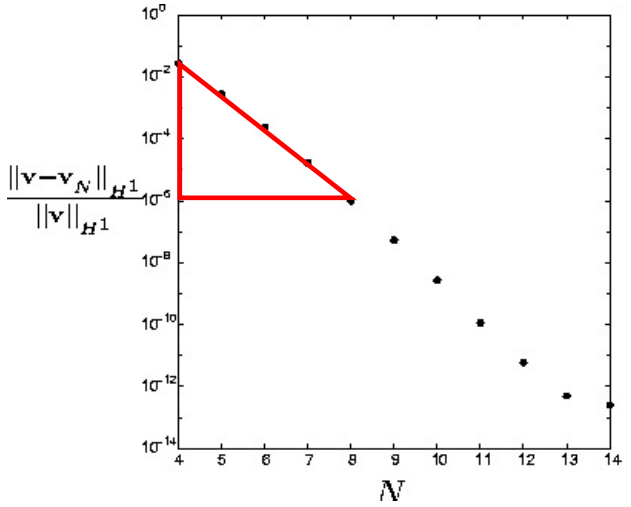
- High-order numerical methods are well suited for accurate, and efficient simulations of turbulent flows
  - **Fast convergence**
  - **Small diffusion and dispersion errors**
  - **Better data volume-over-surface ratio**

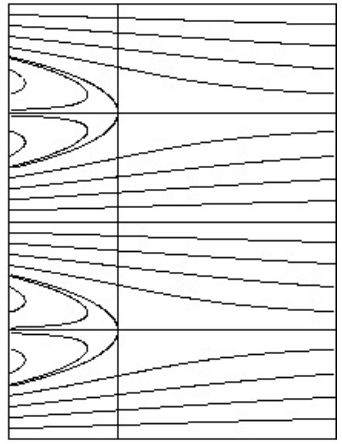
- Spectral elements allows for **complex geometries**

Exponential Convergence

- Converges *exponentially fast* with N for smooth solutions.
- Four orders of magnitude reduction in error when doubling the resolution
- For a given error,
  - Fewer grid points
  - Smaller memory footprint
  - Smaller communication volume

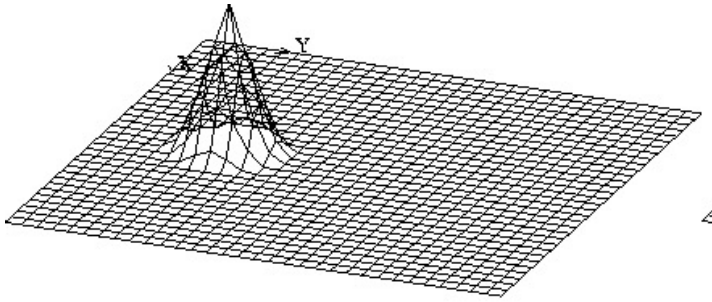
Exact Navier-Stokes Solution (Kovazsnay '48)



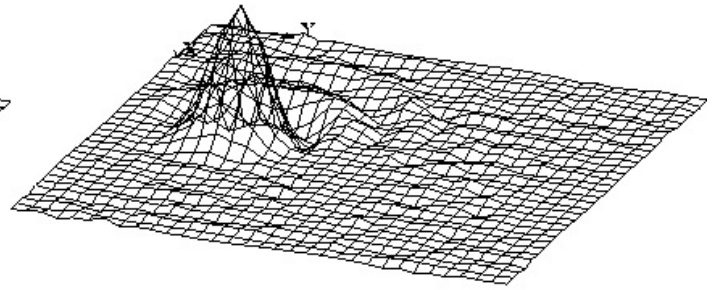


$$
\begin{aligned}
v_x &= 1 - e^{\lambda x}\cos 2\pi y \\
v_y &= \frac{\lambda}{2\pi} e^{\lambda x}\sin 2\pi y \\
\lambda &:= \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}
\end{aligned}
$$

Why high-order?



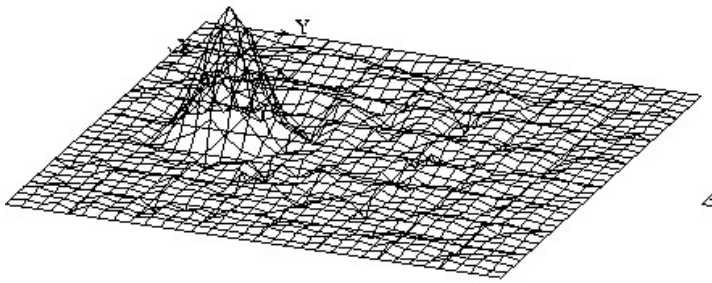
Initial Condition



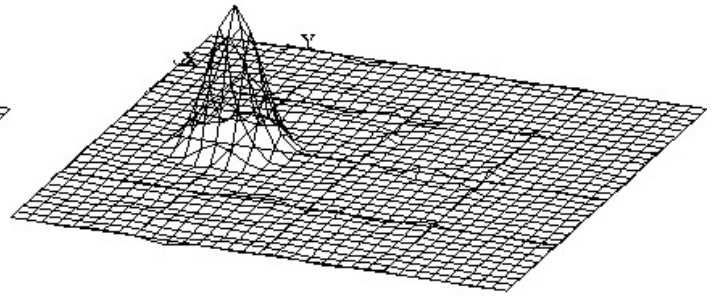
$K_1 = 16,\ N = 2$

Convection of non-smooth data on a 32x32 grid (K_1 x K_1 spectral elements of order N).

(cf. Gottlieb & Orszag 77)



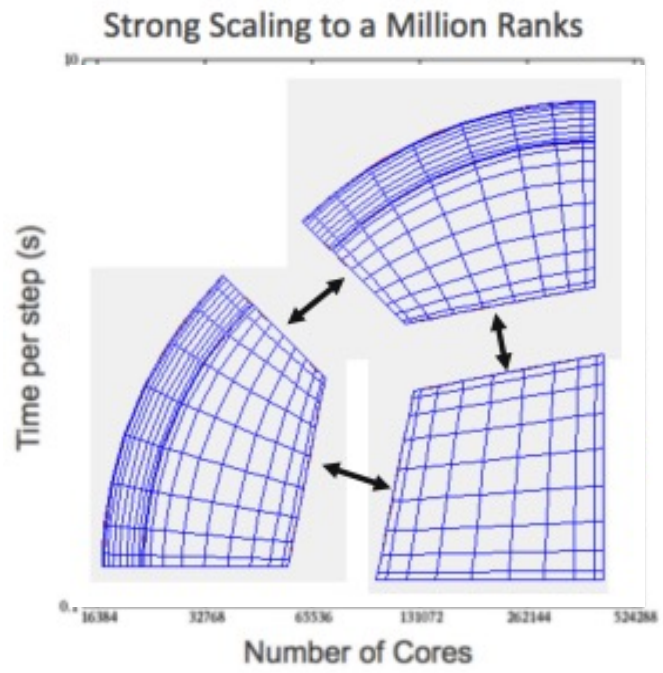
$K_1 = 8,\ N = 4$



$K_1 = 4,\ N = 8$

Highly Scalable Methods

- High-order at low cost
  - Excellent vectorization, dense operators (Level 3 BLAS)
  - Fast, scalable coarse-grid solvers (AMG, XX^T)
  - Domain partitioned into quadrilateral elements
  - Loosely coupled elements (C0 continuity)
- Scales up to 1,000,000 processes on BG/Q
  - 2.95M elements with ninth order polynomials
  - 524 288 cores on ALCF BG/Q Mira;
  - Full scale parallel efficiency 0.6
    - two processes/core, 2000 points/rank



Example – Poisson

Poisson's equation with homogenous Dirichlet boundary conditions,

$$\begin{aligned} -\nabla^2 u &= f && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega. \end{aligned}$$

Find $u \in V \subset H_0^1$ such that,

$$\int_\Omega \nabla u \nabla v \, d\Omega = \int_\Omega f \, v \, d\Omega \quad \forall v \in V.$$

Discretise $\Omega = \bigcup_{e=1}^{E} \Omega^e$, and let V^N be some approximation space where we search for a solution.

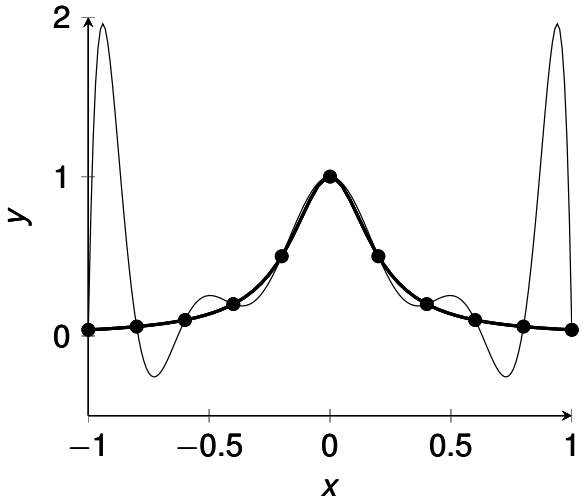
Example – Poisson

- Seek an approximate solution in V^N with "simple" basis functions

$$\varphi_i(x)_j^N = \{\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)\}$$

$$\Rightarrow u(x) = \sum_{i=1}^{n} \xi_i \varphi_i(x)$$

- N-th order polynomials, equidistant points
- We aim for high-order, how large can N be?
- Unstable for high order (Runge's phenomenon)



High-order spatial discretisation

- N-th order Legendre-Lagrange polynomials $h_i(\xi)$,

$$h_i(\xi) = \frac{N(N+1)^{-1}(1-\xi^2)L'_N(\xi_i)}{(\xi-\xi_i)L_N(\xi_i)} \text{ for } \xi \in [-1,1]$$

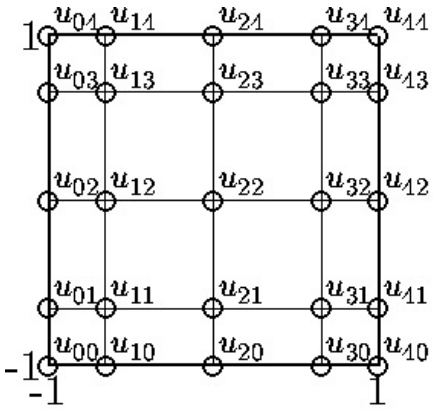
  - Gauss-Lobatto-Legendre quadrature points ξ_i
  - N-th Legendre polynomials L_N

- New ansatz, tensor product polynomials

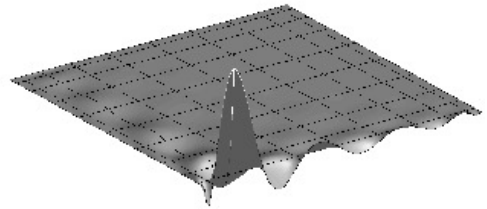
$$u(x,y) = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij}\underbrace{h_i(x)h_j(y)}_{a_{ij}}$$

- Linear system

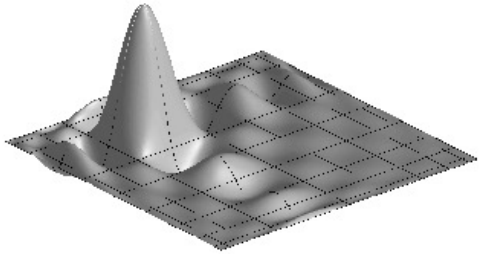
$$Au = f$$

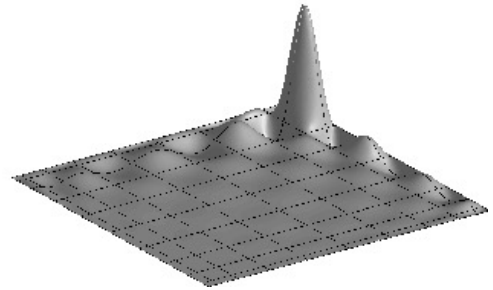


N=4



N=10



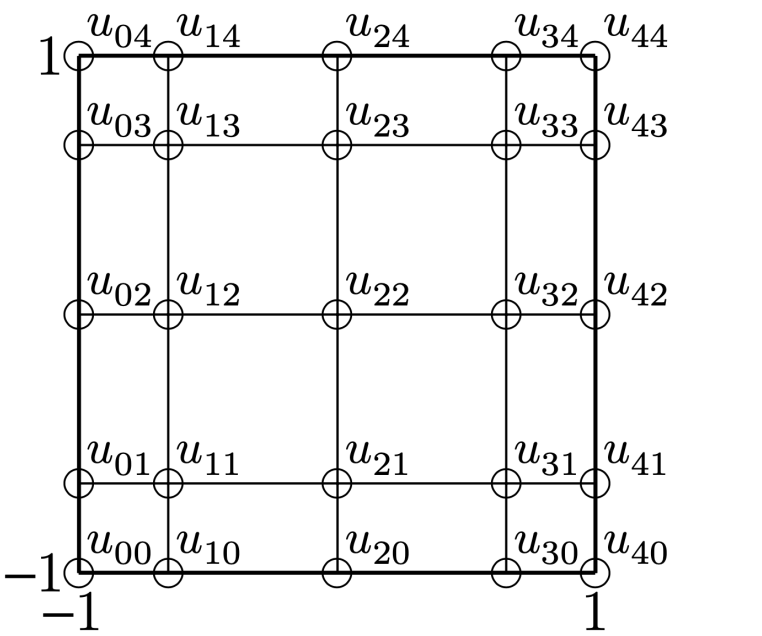


High-order spatial discretisation

- Assembly of Stiffness matrix, add element stiffness matrices $A^k_{i,j}$ to the global stiffness matrix A

for $k \in T$
  for $i = 1 \ \ldots n$
    for $j = 1 \ \ldots n$
      $A_{\mathbb{I}(i),\mathbb{I}(j)} \mathrel{+}= A^k_{i,j}$

- Sparse system with linear elements
- How large is $A^k_{i,j}$ with our ansatz?
  - Quite large with $O(N^4)$ non-zeros ($O(N^6)$ in 3D!)
- **Not feasible to form A!**



Matrix-Free Operator Evaluation

- Always work with the unassembled matrix A_L

$$\underbrace{\begin{pmatrix} A^1 & & & & & \\ & A^2 & & & & \\ & & \ddots & & & \\ & & & A^e & & \\ & & & & \ddots & \\ & & & & & A^E \end{pmatrix}}_{A_L} \underbrace{\begin{pmatrix} u^1 \\ u^2 \\ \vdots \\ u^3 \\ \vdots \\ u^E \end{pmatrix}}_{u_L}$$

- Data replicated in u_L (shared points)
- We only need the action $A_L u_L$ inside iterative solvers (why?)
- Can we afford to compute A_L? (memory?, compute wise?)

Matrix-Free Operator Evaluation

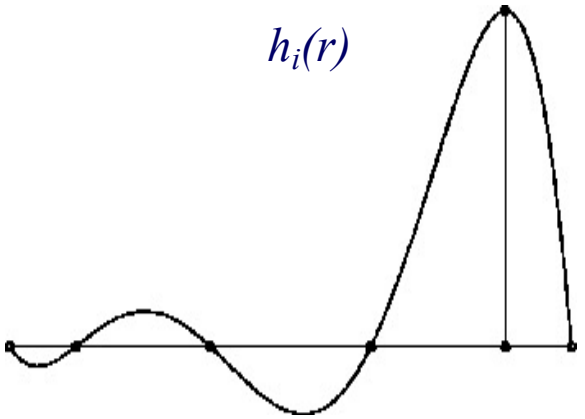
- Poisson's element stiffness matrix: $\int_e \nabla u \nabla v \, de$
- Tensor product polynomials

$$u(x,y) = \sum_{i=0}^{N} \sum_{j=0}^{N} u_{ij} \, h_i(x) h_j(y)$$

- Partial derivatives evaluated as matrix-matrix products

$$\left.\frac{\partial u}{\partial r}\right|_{\xi_i \xi_j} = \sum_{p=0}^{N} u_{pj} \left.\frac{\partial u}{\partial r}\right|_{\xi\_i} = \sum_p \underbrace{\widehat{D}_{ip} u_{pj}}_{\text{mxm}} = D_r u$$

- Precompute D for all directions
- Can we reuse the same partial derivative for all elements?

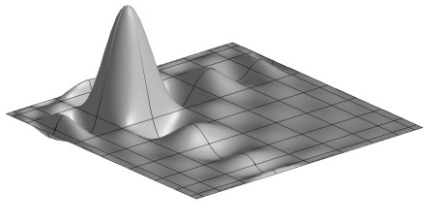


$$\begin{pmatrix} D & & & & \\ & D & & & \\ & & \ddots & & \\ & & & D & \\ & & & & \ddots \\ & & & & & D \end{pmatrix} \begin{pmatrix} u^1 \\ u^2 \\ \vdots \\ u^3 \\ \vdots \\ u^E \end{pmatrix}$$

Mapped geometries

- All evaluations performed on the reference element
  - Invertible map from physical domain to reference domain
- Geometry, same tensor-product as the solution

$$\mathrm{x}(r,s) = \sum_{i=0}^{N} \sum_{j=0}^{N} x_{ij}\, h_i(r) h_j(s)$$



2D basis function, N=10

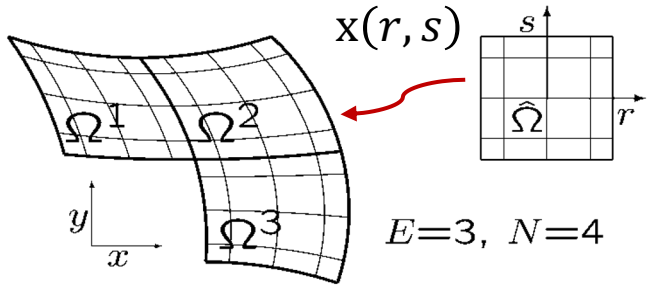
- Partial derivatives of u using the chain rule (in $\mathbb{R}^d$)

$$\frac{\partial u}{\partial x_k} = \sum_{k=1}^{d} \frac{\partial u}{\partial r_i} \frac{\partial r_i}{\partial x_k}$$

- Group all geometric factors $\frac{\partial r_i}{\partial x_k}$

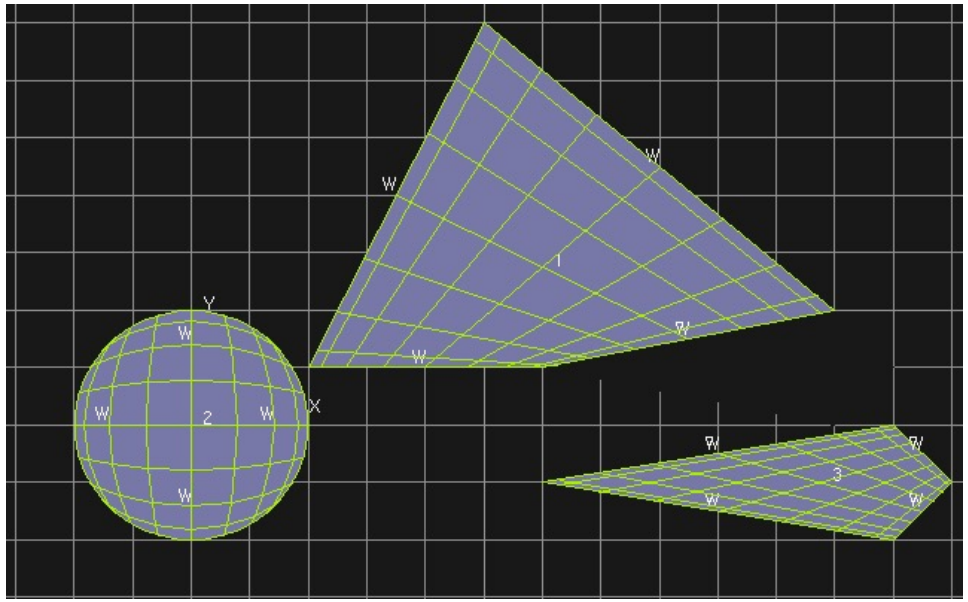
$$G_{ij}(r) = \sum_{k=1}^{d} \frac{\partial r_i}{\partial x_k} \frac{\partial r_j}{\partial x_k} J(r)$$

Jacobian from mapping



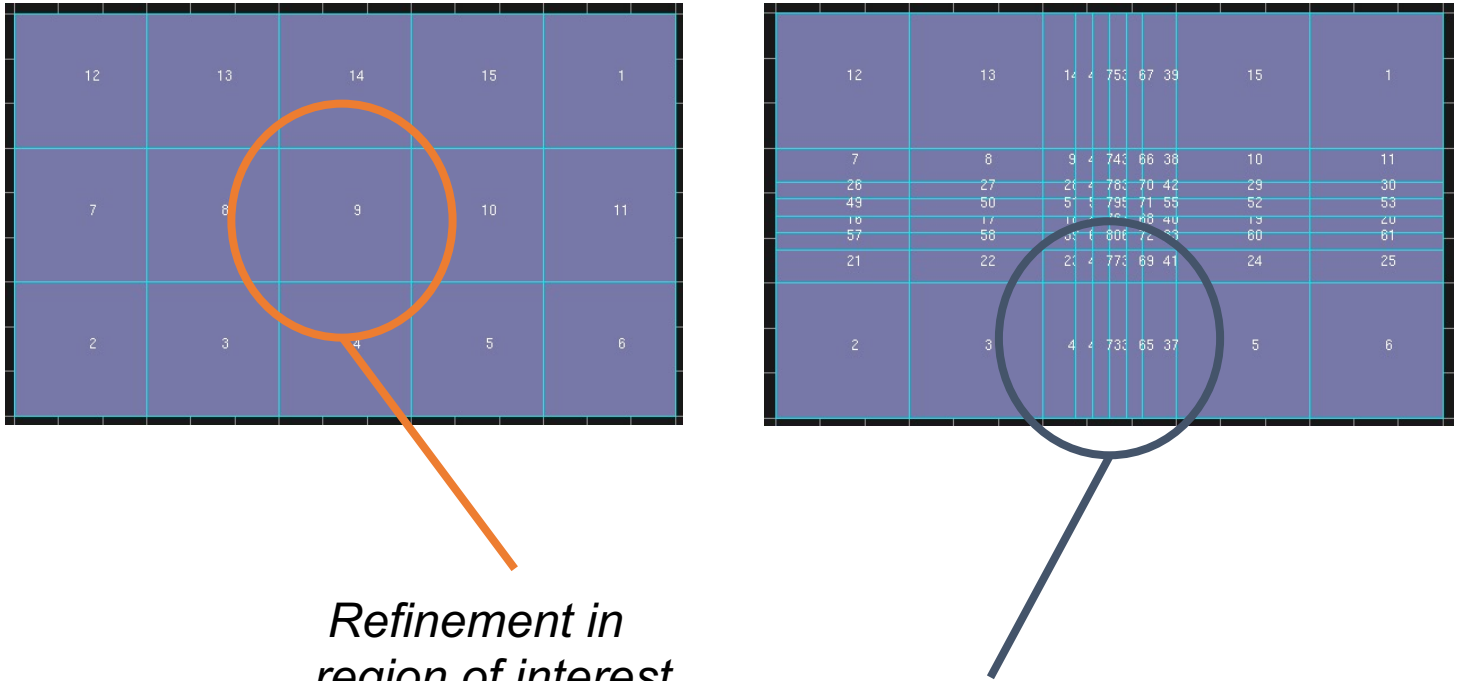
Mapped geometries

- Best to use affine (i.e., linear) transformations in order to preserve underlying GLL spacing for stability and accurate quadrature.
- Avoid singular corners - $\sim 180^{o}$ or $\sim 0^{o}$
- Avoid high-aspect-ratio cells, if possible



Mesh anisotropy

A common refinement scenario (somewhat exaggerated):



Refinement in region of interest…

yields unwanted high aspect-ratio cells in the far field

Refinement propagation leads to

- unwanted elements in far-field
- high aspect-ratio cells that are detrimental to iterative solver performance (F. JCP'97)

Matrix-Free Operator Evaluation

- Generalized element stiffness matrix

$$\int_e \nabla u \nabla v \, de \approx \begin{pmatrix} D_R v^e \\ D_s v^e \end{pmatrix}^T \begin{pmatrix} G_{rr} & G_{rs} \\ G_{rs} & G_{ss} \end{pmatrix} \begin{pmatrix} D_r u^e \\ D_s u^e \end{pmatrix}$$

$$= (v^e)^T \begin{pmatrix} D_R \\ D_s \end{pmatrix}^T \begin{pmatrix} G_{rr} & G_{rs} \\ G_{rs} & G_{ss} \end{pmatrix} \begin{pmatrix} D_r \\ D_s \end{pmatrix} u^e$$

$$= (v^e)^T D^T G^e D u^e$$

- The entire bilinear form for Poisson

$$\int_\Omega \nabla u \nabla v \, d\Omega = \sum_{e=1}^{E} (v^e)^T D^T G^e D u^e = \sum_{e=1}^{E} (v^e)^T A^e u^e$$

- Recap: through use of chain rule + GLL quadrature:
  - ***Matrix-free operator*** *evaluation.*
  - *Operation count is* ***only*** $\boldsymbol{O(N^4)}$ ***not*** $\boldsymbol{O(N^6)}$ *(in 3D) using* $D_R = (I \otimes I \otimes D)$
  - *Memory access is 7n* (G_{rr}, G_{rs}, *etc., are diagonal)*
  - *Work is dominated by matrix-matrix products involving* D_r, D_s, *etc.*

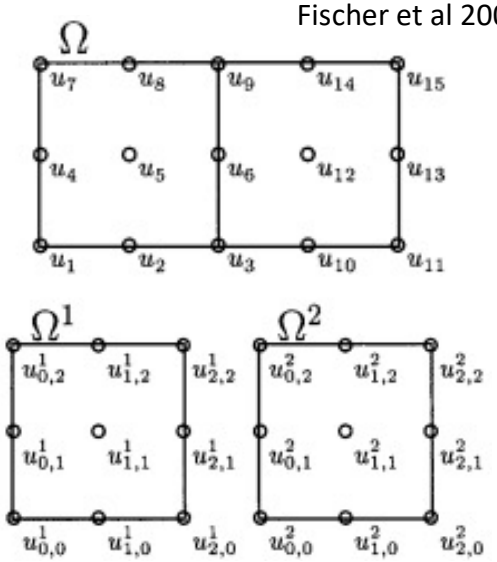
Direct stiffness summation

- How to ensure continuity?
  - Data replicated in $u_L = \{u^e\}_{e=1}^E$
- Boolean gather Q^T and scatter matrix Q
  - Mapping from local to global

$$a(u,v) = \sum_{e=1}^{E} (v^e)^T A^e u^e = (\boldsymbol{Q}v)^T A_L \boldsymbol{Q}u = v^T A u.$$

- Q nor Q^T formed, only the action QQ^T is used
  - Matrix-vector product $w = Au \Rightarrow w_L = QQ^T A_L u_L$
  - Combined gather-scatter operation
  - Local u^e_{ijk} corresponds to global index $u_{\hat{\imath}}$
- **Unit depth stencil for all N!**

Fischer et al 2002



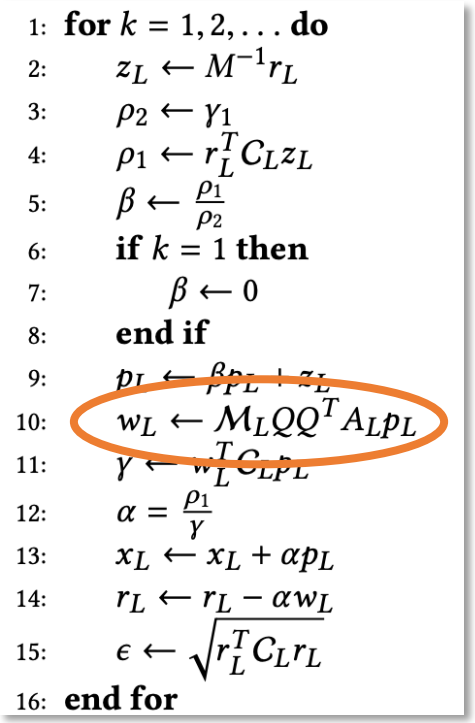
Algorithm 1 The gather-scatter kernels QQ^T.

Function $u = Q^T u_L$	**Function** $u_L = Qu$
for $e = 1, 2 \ldots, E$ **do**	**for** $e = 1, 2 \ldots, E$ **do**
for $i, j, k = 1, 2 \ldots, n$ **do**	**for** $i, j, k = 1, 2 \ldots, n$ **do**
$\hat{\imath} \leftarrow loctglb(i, j, k, e)$	$\hat{\imath} \leftarrow loctglb(i, j, k, e)$
$u_{\hat{\imath}} \leftarrow u_{\hat{\imath}} + u^e_{ijk}$	$u^e_{ijk} \leftarrow u_{\hat{\imath}}$
end for	**end for**
end for	**end for**

Scalable linear solvers

- Key considerations:
  - Bounded iteration counts as $N \to \infty$
  - Cost that does not scale prohibitively with number of processors, P

- Our choice:
  - CG or GMRES, preconditioned with multilevel additive Schwarz
  - Coarse-grid solve:
    - XX^T projection-based solver
    - single V-cycle of well-tuned AMG (in house or using HYPRE)



- Projection in time – extract available temporal regularity in $\{\underline{p}^{n-1}, \underline{p}^{n-2}, \ldots, \underline{p}^{n-k}\}$

Time stepping

- Navier-Stokes time advancement:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\nabla P + \frac{1}{Re} \nabla^2 \vec{u}$$

$$-\nabla . \vec{u} = 0$$

- Nonlinear term: *explicit*
  - k th-order backward difference formula / extrapolation
  - characteristics (Pironneau '82, MPR '90)

Stiffness matrix (Laplacian) Mass matrix

$$H\phi = (h_1 A + h_2 B)[\phi]$$

$$Hu = \left(\frac{1}{Re} A - \frac{b_0}{\Delta t} B\right)[u] = f_u \; (h_1 = 1, h_2 = -\frac{b_0}{\Delta t})$$

$$Hp = (A)[p] = f_p \; (h_1 = 1, h_2 = 0)$$

- Stokes problem: pressure/viscous decoupling:
  - 3 Helmholtz solves for velocity *("easy" w/ Jacobi-precond. CG)*
  - (consistent) Poisson equation for pressure *(computationally dominant)*