Méthodes Itératives

Carola Kruse, Alena Kopanicakova et Ronan Guivarch

Cours 3, 14/03/2025
Méthodes Multigrilles

Iteration matrix

- The two-level iteration matrix is thus given by

$$S_{2lev} = (I - I_{2h}^{h}(A^{2h})^{-1} I_{h}^{2h} A^{h}) S_{s}^{\nu}$$

- To prove convergence, one could do a rigorous Fourier analysis (see Trottenberg, Osterley et al., Multigrid).
- We will follow an approach of Hackbusch (Multi-grid methods and applications) and show mesh-independent convergence.

Convergence analysis

- The analysis is based on a splitting of S_{2lev} as

$$S_{2lev} = \left(\left(A^h\right)^{-1} - I_{2h}^{h} \left(A^{2h}\right)^{-1} I_{h}^{2h} \right) A^h S_{sm}^{\nu}$$

- Hence,

$$\|S_{2lev}\| \leq \|\left(A^h\right)^{-1} - I_{2h}^{h} \left(A^{2h}\right)^{-1} I_{h}^{2h}\| \cdot \|A^h S_{sm}^{\nu}\|$$

Approximation property Smoothing property

- Approximation property: This is the effect of the coarse grid approximation.
- Smoothing property: The efficiency of the smoothing step.

Smoothing and approximation properties

Definition 1: The matrix S_{sm} is said to possess the smoothing property, if there exist functions $\eta(\nu)$ and $\bar{\nu}(t)$ independent of h, such that

$$\left\| A^h S_{sm}^{\nu} \right\| \leq \eta(\nu) h^{-\alpha}$$

for some number $\alpha > 0$ and for all $1 \leq \nu \leq \bar{\nu}(h)$, with $\eta(\nu) \to 0$ as $\nu \to \infty$

Definition 2: The approximation property holds if there is a constant C_α, which is independent of h, such that

$$\left\| \left(A^h\right)^{-1} - I_{2h}^{h} \left(A^{2h}\right)^{-1} I_{h}^{2h} \right\| \leq C_\alpha h^\alpha$$

with the same α as in the smoothing property.

Convergence of the two-level method

Theorem

Suppose that the smoothing and the approximation property hold. Let $\rho > 0$ be a fixed number. Then there exists a number ν^*, such that

$$\|S_{2lev}\| \leq C_\alpha \eta(\nu) \leq \rho,$$

whenever $\nu \geq \nu^*$.

Proof

Since we required that α is the same for the approximation as well as the smoothing property, we get

$$\|S_{2lev}\| \leq C_\alpha h^\alpha \eta(\nu) h^{-\alpha} = C_\alpha \eta(\nu)$$

Since $\eta(\nu) \to 0$ as $\nu \to \infty$, the right hand side can be smaller than any given ρ, namely as ν is sufficiently large.

Convergence of the two-level method

- Note that the constant $C_\alpha \eta(\nu)$ is independant of the mesh size *h*
- It converges, if sufficiently many smoothing steps are applied
- In practice only a few pre-smoothing steps (1 to 3) are often sufficient

- The estimate above is however often not very tight.

V(2,1)-cycle for 2D Poisson

	$n = 16$				$n = 32$				$n = 64$				$n = 128$			
V-cycle	$\|\mathbf{r}^h\|_h$	ratio	$\|\mathbf{e}\|_h$	ratio	$\|\mathbf{r}^h\|_h$	ratio	$\|\mathbf{e}\|_h$	ratio	$\|\mathbf{r}^h\|_h$	ratio	$\|\mathbf{e}\|_h$	ratio	$\|\mathbf{r}^h\|_h$	ratio	$\|\mathbf{e}\|_h$	ratio
0	6.75e+02		5.45e−01		2.60e+03		5.61e−01		1.06e+04		5.72e−01		4.16e+04		5.74e−01	
1	4.01e+00	0.01	1.05e−02	0.02	1.97e+01	0.01	1.38e−02	0.02	7.56e+01	0.01	1.39e−02	0.02	2.97e+02	0.01	1.39e−02	0.02
2	1.11e−01	0.03	4.10e−04	0.04	5.32e−01	0.03	6.32e−04	0.05	2.07e+00	0.03	6.87e−04	0.05	8.25e+00	0.03	6.92e−04	0.05
3	3.96e−03	0.04	1.05e−04	0.26	2.06e−02	0.04	4.41e−05	0.07	8.30e−02	0.04	4.21e−05	0.06	3.37e−01	0.04	4.22e−05	0.06
4	1.63e−04	0.04	1.03e−04	0.98*	9.79e−04	0.05	2.59e−05	0.59	4.10e−03	0.05	7.05e−06	0.17	1.65e−02	0.05	3.28e−06	0.08
5	7.45e−06	0.05	1.03e−04	1.00*	5.20e−05	0.05	2.58e−05	1.00*	2.29e−04	0.06	6.45e−06	0.91*	8.99e−04	0.05	1.63e−06	0.50
6	3.75e−07	0.05	1.03e−04	1.00*	2.96e−06	0.06	2.58e−05	1.00*	1.39e−05	0.06	6.44e−06	1.00*	5.29e−05	0.06	1.61e−06	0.99*
7	2.08e−08	0.06	1.03e−04	1.00*	1.77e−07	0.06	2.58e−05	1.00*	8.92e−07	0.06	6.44e−06	1.00*	3.29e−06	0.06	1.61e−06	1.00*
8	1.24e−09	0.06	1.03e−04	1.00*	1.10e−08	0.06	2.58e−05	1.00*	5.97e−08	0.07	6.44e−06	1.00*	2.14e−07	0.06	1.61e−06	1.00*
9	7.74e−11	0.06	1.03e−04	1.00*	7.16e−10	0.06	2.58e−05	1.00*	4.10e−09	0.07	6.44e−06	1.00*	1.43e−08	0.07	1.61e−06	1.00*
10	4.99e−12	0.06	1.03e−04	1.00*	4.79e−11	0.07	2.58e−05	1.00*	2.87e−10	0.07	6.44e−06	1.00*	9.82e−10	0.07	1.61e−06	1.00*
11	3.27e−13	0.07	1.03e−04	1.00*	3.29e−12	0.07	2.58e−05	1.00*	2.04e−11	0.07	6.44e−06	1.00*	6.84e−11	0.07	1.61e−06	1.00*
12	2.18e−14	0.07	1.03e−04	1.00*	2.31e−13	0.07	2.58e−05	1.00*	1.46e−12	0.07	6.44e−06	1.00*	4.83e−12	0.07	1.61e−06	1.00*
13	2.33e−15	0.11	1.03e−04	1.00*	1.80e−14	0.08	2.58e−05	1.00*	1.08e−13	0.07	6.44e−06	1.00*	3.64e−13	0.08	1.61e−06	1.00*
14	1.04e−15	0.45	1.03e−04	1.00*	6.47e−15	0.36	2.58e−05	1.00*	2.60e−14	0.24	6.44e−06	1.00*	1.03e−13	0.28	1.61e−06	1.00*
15	6.61e−16	0.63	1.03e−04	1.00*	5.11e−15	0.79	2.58e−05	1.00*	2.30e−14	0.88	6.44e−06	1.00*	9.19e−14	0.89	1.61e−06	1.00*

- 2D Poisson on unit square and homogeneous Dirichlet conditions
- Results taken from Briggs et al, A multigrid tutorial, Table 4.1

The Multigrid Method

Multigrid method

What is the best method to solve the coarse grid equation $A^{2h}e^{2h} = r^{2h}$?

- We could use a direct method. But what if the coarse grid problem is still very large?
- Let us think recursively: The coarse-grid problem is not much different from the original problem.
- We can apply the two-level method to the coarse grid problem, thus going to a grid on Ω^{4h}.
- We can repeat this process until a direct solution of the residual equation is possible.

In the following

- let *L* be the number of levels,
- for simplicity of the notation, we will call the restriction of the residual

$$f^{2^k h} := I_{kh}^{2^k h} r^{kh}.$$

- It is in fact only a new right-hand side.

V-cycle scheme

$v^h \leftarrow V^h(v^h, f^h)$

- Relax ν_1 times on $A^h u^h = f^h$ with initial guess v^h. The result is denoted by v^h.
- Compute $f^{2h} = I_h^{2h} r^h = I_h^{2h}(f^h - A^h v^h)$.
  - Relax ν_1 times on $A^{2h} u^{2h} = f^{2h}$ with initial guess $v^{2h} = 0$. The results is denoted by v^{2h}.
  - Compute $f^{4h} = I_{2h}^{4h} r^{2h} = I_{2h}^{4h}(f^{2h} - A^{2h} v^{2h})$.

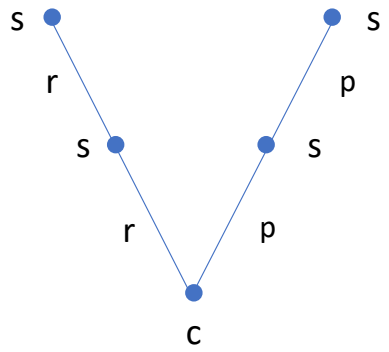
⋮

- Solve $A^{Lh} u^{Lh} = f^{Lh}$

Coarsest grid solve

⋮

  - Correct $v^{2h} := v^{2h} + I_{4h}^{2h} v^{4h}$
  - Apply smoother ν_2 times to $A^{2h} u^{2h} = f^{2h}$ with the initial guess v^{2h}
- Correct $v^h := v^h + I_{2h}^h v^{2h}$
- Apply smoother ν_2 times to $A^h u^h = f^h$ with the initial guess v^h



Remarks

In terms of the error:

- Remember what we said about the frequency components on different grids. Here, we do the following:
  - By applying smoothing iterations on Ω^h, we cause the *h* - high-frequency components to decrease rapidly. The smooth ones stay visible.
  - We then restrict the *h* – low-frequency components to Ω^{2h}, where they become more high-frequent.
  - We then apply again a smoother and damp out the *2h* - high-frequency components.
  - By continuing to the coarsest grid, we obtain a very fast overall reduction of the error.

Furthermore:

- The best number of pre-relaxation and post-relaxation iterations is normally 1 to 3.
- The boundary conditions for the coarse grid problem are 0 (because the coarse-grid variable is the error).
- The initial guess for the coarse-grid solution must be 0.

Multigrid with γ-cycle

- The previous V-cycle scheme is just one possibility to perform a multigrid method.
- It belongs to a family of multigrid methods, the so-called multigrid methods with γ-cycle.
- In these cases, the cycle does not necessarily have a shape of a 'V'.
- It has a compact recursive definition (see next slide).

We use some new notation:

- Number of levels L, thus giving $(L + 1)$ grids.
- Let $k = 0, \dots, L$ be the numbering of the grids. Then $k = 0$ corresponds to the coarsest one, and $k = L$ to the finest one.

Multigrid with γ-cycle

Multigrid Cycle $u_k^{(m+1)} = MGC(k, \gamma, u_k^{(m)}, L_k, f_k, \nu_1, \nu_2)$

1. **Pre smoothing:** Compute $\bar{u}_k^{(m)}$ by applying ν_1 smoothing steps to $u_k^{(m)}$.
2. **Coarse grid correction:**

   Compute the residual and restrict it to the next coarser grid $r_{k-1}^{(m)} = I_k^{k-1}\left(f_k - A_k r_l^{(m)}\right)$

   If $\boldsymbol{k = 1}$:

   We are on the coarsest grid, then solve the problem directly and go to step 3.

   If $\boldsymbol{k > 1}$:

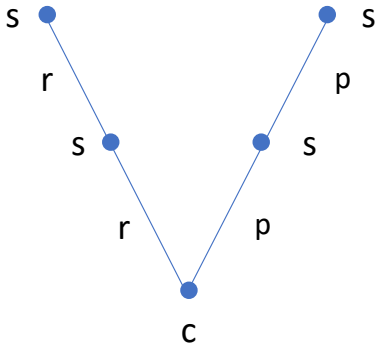
   Apply $\gamma \geq 1$ k – grid cycles using the zero initial guess $v = 0$:

$$v_{k-1}^{(m)} = MGC^{\gamma}(k-1, \gamma, 0, A_{k-1}, r_{k-1}^{(m)}, \nu_1, \nu_2)$$

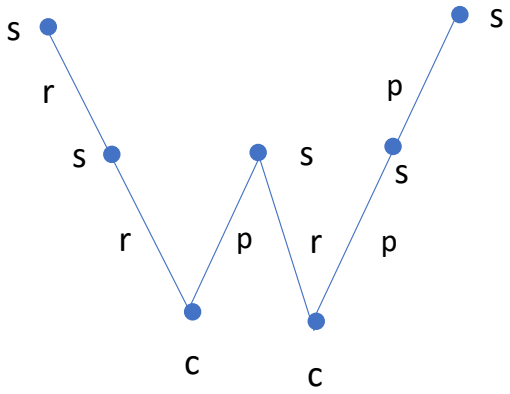
3. **Prolongation and update:** Compute $u_k^{(m)} = \bar{u}_k^{(m)} + I_{k-1}^k v_{k-1}^{(m)}$
4. **Post smoothing:** Apply the smoother ν_2 times on $A_k u_k^m = f^h$ and obtain $u_k^{(m+1)}$.

Three-grid methods

- In practice, mostly $\gamma = 1$ and $\gamma = 2$ are used.
- The following figures show one iteration step of a multigrid method with 3 grid - levels



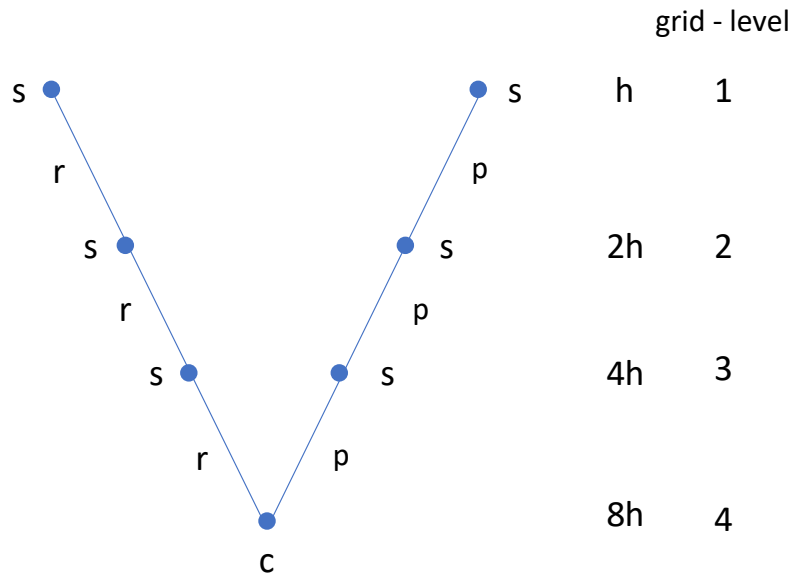
$\gamma = 1$: V-cycle scheme



$\gamma = 2$: W-cycle scheme

Four-grid methods

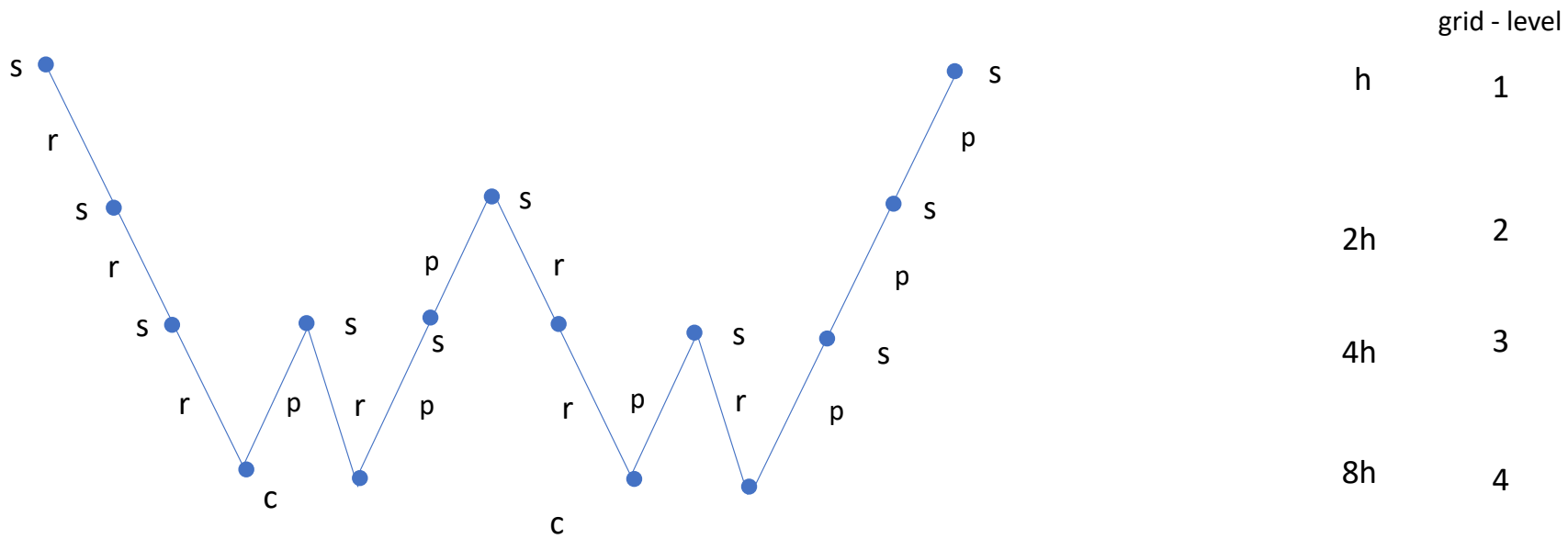
- Here we see a four-grid V-cycle



$\gamma = 1$: V-cycle scheme

Multigrid W-cycle

- Four-grid W-cycle scheme



$\gamma = 2$: W-cycle scheme

Iteration operator of γ-cycle

Remeber that the two-grid iteration operator from grid *h* to *2h* is given by

$$(S_{2lev})_h^{2h} = (I - I_{2h}^h \left(A^{2h}\right)^{-1} I_h^{2h} A^h) S_{sm}^{\nu}$$

What changes in the γ-cycle multigrid method?

$\left(A^{2h}\right)^{-1}$ is now an recursive loop of two-level operators.

Let L be the number of levels and $k = L, \dots, 0$ the grid level. The multigrid iteration operator M_k is given for $k = 1, .., L$ by

$$M_0 = 0$$
$$M_k = S_k^{\nu_2}\left(I - I_{k-1}^k (I_{k-1} - (M_{k-1})^{\gamma})(A_{k-1})^{-1} I_k^{k-1} A_k\right) S_k^{\nu_1}$$

The γ –cycle multigrid method also shows mesh independent convergence as the two-level method.

Computational work

Computational work of multigrid cycle

- Earlier, we have seen that the convergence speed of the two-level method does not depend on the mesh size of the finest grid (mesh-independent convergence).
- The same is true for the multigrid algorithm.
- But just knowing that its convergence is independent of the mesh size says nothing about its efficiency.
- We thus want to estimate the computational work of a multigrid method.
- We follow the description in the book ‘Multigrid’ of Trottenberg et al.

Computational costs of V-Cycle

- From the recursive definition of a multigrid cycle, it follows that the computational work W_l per multigrid cycle on Ω_l is given recursively by

$$W_1 = W_1^0 + W_0, \qquad W_{k+1} = W_{k+1}^k + \gamma W_k, \qquad k = 1, \dots, l-1$$

- Here, W_{k+1}^k denotes the computational work of one (h_{k+1}, h_k) two-grid cycle, excluding the work needed to solve the residual equation.
- W_0 denotes the work needed to compute the exact solution on the coarsest grid Ω_0.
- By computational work, we mean some reasonable measure, typically the number of arithmetic operations.

We obtain for level $l \geq 1$

$$W_l = \sum_{k=1}^{l} \gamma^{l-k} W_k^{k-1} + \gamma^{l-1} W_0$$

Computational costs

- We use standard coarsening in 2D, i.e. $h \rightarrow 2h \rightarrow 4h \rightarrow$ …, which gives in terms of grid points

$$N_k = 4N_{k-1}, \qquad (k = 1,2, \dots, l)$$

  with equality up to lower order terms (boundary effects).

- We assume that the multigrid components (relaxation, computation of residuals, fine-to-coarse and coarse-to-fine grid transfer) require a number of arithmetic operations per point of the respective grids, which is bounded by a small constant *C*, independent of *k:*

$$W_k^{k-1} \leq C\ N_k, \qquad (k = 1, 2, \dots, k),$$

  with, again, $'\leq'$ up to lower order terms.

The constant C

W_k^{k-1} or, in other words the constant C, is determined by the computational work of the multigrid components of the (h_k, h_{k-1}) two grid method, namely

$$W_k^{k-1} = \big((\nu_1 + \nu_2)w_0 + w_1 + w_2\big)N_k$$

Here, w_0, w_1, w_2 are measures for the computational work per grid point of Ω_k needed for the single components

- w_0: one smoothing step on Ω_k
- w_1: computation of the residual and its transfer to Ω_{k-1}
- w_2: interpolation of the correction to Ω_k and its addition to the previous approximation.
- ν_1, ν_2: number of pre- and postsmoothing steps

In practice, interpolation and restriction are often negligible and only the cost of one smoothing step in used.

Computational costs

- We then obtain

$$W_l \leq \begin{cases} \frac{4}{3} C N_l, & \text{for } \gamma = 1 \\ 2\, C N_l & \text{for } \gamma = 2 \\ 4\, C N_l & \text{for } \gamma = 3 \\ \mathcal{O}(N_l \log N_l) & \text{for } \gamma = 4 \end{cases}$$

- This estimate shows that the computational work needed for one 2D multigrid cycle is proportional to the number of grid points on the finest grid for $\gamma \leq 3$ and standard coarsening.
- Together with the h- independent convergence, this means that multigrid methods achieve a fixed reduction of the error in $\mathcal{O}(N)$ operations.
- The constant C depends on γ, the type of coarsening and the other multigrid components.

Full Multigrid

Approximation and algebraic error

- The error that we do is twofold:
  - Discretization error
  - Algebraic error

- Let u be the exact solution, $\tilde{u}^h$ the solution that we obtain after the solution of the linear system, and u^h the exact solution of the linear system.

- How good can we approximate the exact solution?

$$\|u - \tilde{u}^h\| = \|u - u^h + u^h - \tilde{u}^h\| \leq \|u - u^h\| + \|u^h - \tilde{u}^h\| = \mathcal{O}(h^2) + \epsilon$$

Algebraic error:
Depends on the required precision of the iterative solver ϵ

Discretization error:
Depends on the mesh size h

- Solving for an accuracy better than the one limited by the discretization error, does not make sense in many cases.
- The iterative solution may thus get cheaper (less iterations are needed).

Full Multigrid V-cycle

- Remember that one way to improve the algorithms is to provide a good initial guess.
- We could thus precede such multigrid V-cycle by one coarse grid solve to have a good initial guess.
- This is called Full Multigrid V-cycle scheme.

It has the fundamental properties:

1. An approximation of the u_h^{FMG} of the discrete solution u_h can be computed up to an error $|| u^h - u_h^{FMG} ||$, which is approximately equal to the discretization error $|| u - u_h ||$.
2. FMG is an asymptotically optimal method, i.e. the number of arithmetic operations required to compute ${u_h}^{FMG}$, is proportional to the number of grid points of Ω_h (with only a small constant of proportionality).

- For many problems, FMG with just a single V-cycle per level suffices to reduce the error below dicretization error level. In this case, only $\mathcal{O}(N)$ operations are required overall.

What does it do?

- Let u_k^{FMG} denote the solution defined on Ω_k and u_k^h the exact solution to the discrete problem. We want the algebraic error in this solution to be at worst comparable to the discretizaton error

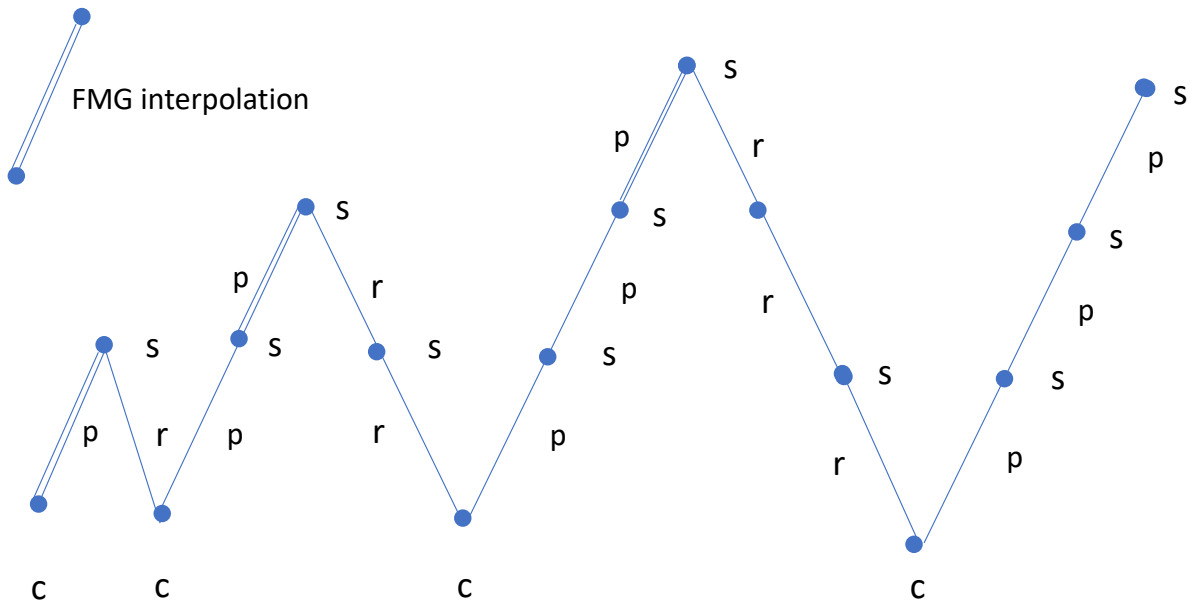
$$\left\| u_k^h - u_k^{FMG} \right\| \leq \beta \left\| u - u_k^h \right\|.$$

- This then immediately implies

$$\left\| u - u_k^{FMG} \right\| = \left\| u - u_k^h \right\| + \left\| u_k^h - u_k^{FMG} \right\| \leq (1 + \beta) \left\| u - u_k^h \right\|$$

- We see that working hard to reduce β below 1 does not improve the quality of the approximation. The effort is better spent on reducing the discretization error by using a finer grid.

Full Multigrid V-cycle



Full multigrid V-cycle

Mesh size	Grid - Level
h	1
2h	2
4h	3
8h	4

Full Multigrid

- For $k = 0$:
  - Solve $L_0 u_0 = f_0$, providing $u_0^{FMG} = u_0$

For $k = 1, 2, \ldots, l$:

- $u_k^0 = \prod_{k-1}^{k} u_{k-1}^{FMG}$
- $u_k^{FMG} = MGC^r(k+1, \gamma, u_k^0, A_k, f_k, \nu_1, \nu_2)$

- r is the number of repetitions of a chosen MG cycle
- $\prod_{k-1}^{k}$ from Ω_{k-1} to Ω_k is the FMG interpolation operator, that transfers approximations of the solution to the fine grid.

FMG computational work

Computational work for standard coarsening in 2D

$$W_l^{FMG} \leq \frac{4}{3} r W_l + \frac{4}{3} W_{l-1}^{INT}$$

- Here, W_{l-1}^{INT} denotes the work needed for the FMG interpolation process from grid Ω_{l-1} to Ω_l and W_l is the work required for one multigrid cycle on the finest grid level Ω_l.
- Both W_l and W_{l-1}^{INT} are $\mathcal{O}(N)$. Thus the FMG cycle only requires $\mathcal{O}(N)$ operations.

Full Multigrid V-cycle

N	FMG(1,0) $\|\mathbf{e}\|_h$	ratio	FMG(1,1) $\|\mathbf{e}\|_h$	ratio	FMG(2,1) $\|\mathbf{e}\|_h$	ratio	FMG(1,1) WU	V(2,1) cycles	V(2,1) WU
2	5.86e−03		5.86e−03		5.86e−03				
4	5.37e−03	0.917	2.49e−03	0.424	2.03e−03	0.347	7/2	3	12
8	2.78e−03	0.518	9.12e−04	0.367	6.68e−04	0.328	7/2	4	16
16	1.19e−03	0.427	2.52e−04	0.277	1.72e−04	0.257	7/2	4	16
32	4.70e−04	0.395	6.00e−05	0.238	4.00e−05	0.233	7/2	5	20
64	1.77e−04	0.377	1.36e−05	0.227	9.36e−06	0.234	7/2	5	20
128	6.49e−05	0.366	3.12e−06	0.229	2.26e−06	0.241	7/2	6	24
256	2.33e−05	0.359	7.35e−07	0.235	5.56e−07	0.246	7/2	7	28
512	8.26e−06	0.354	1.77e−07	0.241	1.38e−07	0.248	7/2	7	28
1024	2.90e−06	0.352	4.35e−08	0.245	3.44e−08	0.249	7/2	8	32
2048	1.02e−06	0.351	1.08e−08	0.247	8.59e−09	0.250	7/2	9	36

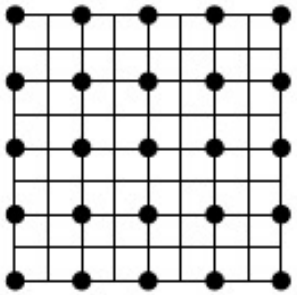
- 2D Poisson on unit square and homogeneous Dirichlet conditions
- Results taken from Briggs et al, A multigrid tutorial, Table 4.1

Practical aspects

Coarsening

Coarse grids – standard coarsening

- So far, we have only talked about standard coarsening, i.e. we assume a number of 2^n elements and divide these for each coarser grid by 2.



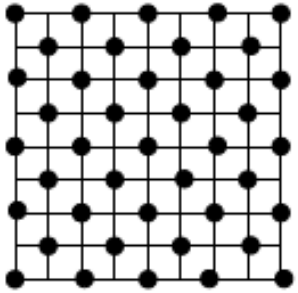
- In *d* dimensions, the relation between the number of grid points (neglecting boundary effects) is thus given by

$$\# \, \Omega_{2h} \approx \frac{1}{2^d} \# \, \Omega_h$$

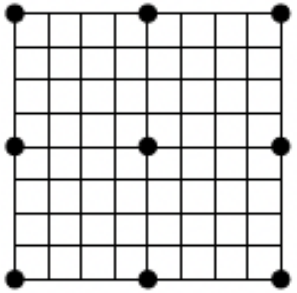
- There are however other choices…

Coarse grids – red-black and 4h

- Also of interest:



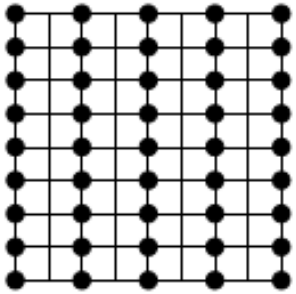
Red-black (or checkerboard) coarsening



4h-coarsening

Coarse grids – semi-coarsening

- For some aninsotropic problems, it might be more appropriate to coarsen only in the *x*- or *y*-direction (we will see an example next). This is called semi-coarsening.



- The mesh size is doubled in one direction only, and is left constant in the other, i.e. $H = (2h_x, h_y)$ or $H = (h_x, 2h_y)$
- The number of degrees of freedom do not decrease as fast as for standard coarsening. Some more levels might be needed to have a small coarse-grid problem.
- In this case, we have

$$\# \, \Omega_{2h} \approx \frac{1}{2} \# \, \Omega_h$$

Anisotropic Poisson equation

- Let us assume that we have a model problem with $0 < \epsilon \ll 1$

$$-u_{xx} - \epsilon u_{yy} = f \qquad \text{+ Bdy conditions}$$

- This leads to the same five-point as the Poisson equation ($\epsilon = 1$)

$$A^h = \frac{1}{h^2}\begin{pmatrix} & -\epsilon & \\ -1 & 2+2\epsilon & -1 \\ & -\epsilon & \end{pmatrix}$$

- We have only a weak connection in y-direction. This is best visible in the limiting case $\epsilon = 0$.

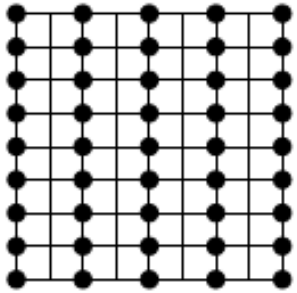
$$A^h = \frac{1}{h^2}\begin{pmatrix} & 0 & \\ -1 & 2 & -1 \\ & 0 & \end{pmatrix} \quad \leftarrow \quad \text{This is the 1D-Poisson 3-Point stencil in x-direction}$$

- Thus, Gauss-Seidel or weighted Jacobi would smooth well in x-direction (they do so in 1D), but each line is not well connected to the x-line above.

How can we overcome this problem?

Semi-coarsening and point relaxation

- For this problem, it makes sense to use semi-coarsening.



- Point relaxation (standard Gauss-Seidel or weighted Jacobi that acts on each point), will not smoothen well in y-direction. So it makes sense to keep every point also on the coarse grid to have a good resolution of the error.

- We thus only coarsen in x-direction, where the smoother works well as in 1D.

- Interpolation and restriction are done in a one-dimensional way.

Full coarsening and line relaxation

- If we order the unknowns in the direction of constant *x*, i.e. the direction of strong coupling, we obtain

$$A^h = \begin{pmatrix} \widetilde{D} & -cI & & & \\ -cI & \widetilde{D} & -cI & & \\ & -cI & \widetilde{D} & \ddots & \\ & & \ddots & \ddots & -cI \\ & & & -cI & \widetilde{D} \end{pmatrix} \quad \text{with} \quad \widetilde{D} = \begin{pmatrix} 2+2\epsilon & -1 & & & \\ -1 & 2+2\epsilon & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2+2\epsilon \end{pmatrix} \quad \text{and} \quad c = \frac{\epsilon}{h^2}$$

- Each block is associated with an individual horizontal grid line.
- We can apply weighted Jacobi in the sense of a block-matrix, i.e.

$$D^{-1} = \begin{pmatrix} \widetilde{D}^{-1} & & & \\ & \widetilde{D}^{-1} & & \\ & & \ddots & \\ & & & \widetilde{D}^{-1} \end{pmatrix}$$

- The matrices $\widetilde{D}$ are tridiagonal and can thus be efficiently solved using some kind of Gaussian elimination.
- This method is called line-relaxation, as it treats one line together.

Practical aspects

Parallelization

Parallel properties of smoothers - Jacobi

- With growing problem sizes, we are interested in parallel implementations of the multigrid method. One important ingredient are the smoothers.

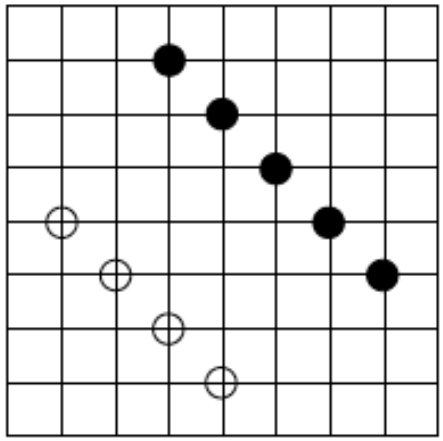
Weighted Jacobi:

- This smoother is fully parallel, as we need for the next iterate only the values of the previous iterate.
- The new values do not depend on each other, thus it can be applied to all grid points simultaneously.
- The degree of parallelism is the number of grid points, thus

$$\text{Par-deg}(\omega\text{-JAC}) = \sharp\, \Omega_h$$

Parallel properties of smoothers – Gauss-Seidel

- The classical Gauss-Seidel smoother is not parallel, as for each new value in a line, we access the already computed ones.
- However, there are special cases.

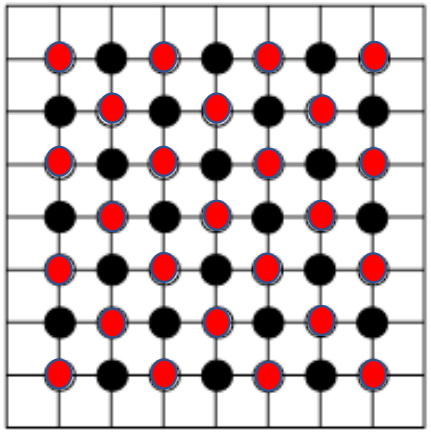


(These are the grid-points, not the matrix entries.)

- For the 2D Poisson problem with a 5-point stencil, the entries on a diagonal are independent from each other. These can be relaxed in parallel.
- From one diagonal to the other is sequential.
- The number of points in one diagonal varies.
- The degree of parallelism is restricted by

$$\text{Par-deg(GS)} = (\#\ \Omega_h)^{\frac{1}{2}}$$

Red-black Gauss-Seidel smoother



- The red-black Gauss-Seidel smoother consiste of two half-steps:
  1. All red grid points are treated simultaneously
  2. All black grid points are treated, using the updated red grid point values.

- The degree of parallelism is thus

$$\text{Par-deg(RB-GS)} = \frac{1}{2}\left(\#\ \Omega_h\right)$$

Smoothing vs Parallelism

Relaxation	Smoothing factor	Smoothing	Parallel degree
ω-Jacobi, $\omega = 1$	1	No	N
ω-Jacobi, $\omega = 0.5$	0.75	Unsatisfactory	N
ω-Jacobi, $\omega = 2/3$	0.33	Very good	N
Gauss-Seidel	0.5	Good	$\leq \sqrt{N}$
GS-RB	0.25	Very good	$\frac{1}{2}N$

Measure multigrid convergence factor

- In order to analyze a multigrid iteration, one often wants to determine its convergence factor ρ empirically.
- The only quantities that are available for the determination of ρ are the residuals on each level.
- For example, we can compute

$$q^{(m)} := \frac{\left\| r_h^{(m)} \right\|_2}{\left\| r_h^{(m-1)} \right\|_2}$$

Or as a product of all previous residuals

$$\hat{q}^{(m)} := \sqrt[m]{q^{(m)} q^{(m-1)} \dots q^{(1)}} = \sqrt[m]{\frac{\left\| r_h^m \right\|_2}{\left\| r_h^0 \right\|_2}}$$

- The quantity $\hat{q}^{(m)}$ represents an average defect reduction factor over m iterations.
- In general, for $r_h^0 \neq 0$, we have $\hat{q}^{(m)} \to \rho$ for m sufficiently large.
- The convergence history for $q^{(m)}$ might also be of interest.

Final remarks

Take-away message

- Smoothing handles high frequencies and multigrid the low ones.
- Multigrid has mesh-independent convergence.
- Usually $\mathcal{O}(N)$ or $\mathcal{O}\ (N\ \log N)$ operations are needed.
- The importance of such an efficient method is becoming greater the greater the problem size and also as computers grow stronger.

Multigrid is not the answer to everything!

It works well for: Sparse, low dimension, large, stiff, elliptic PDE, structured, smoothly varying coefficients, symmetric positive define, …

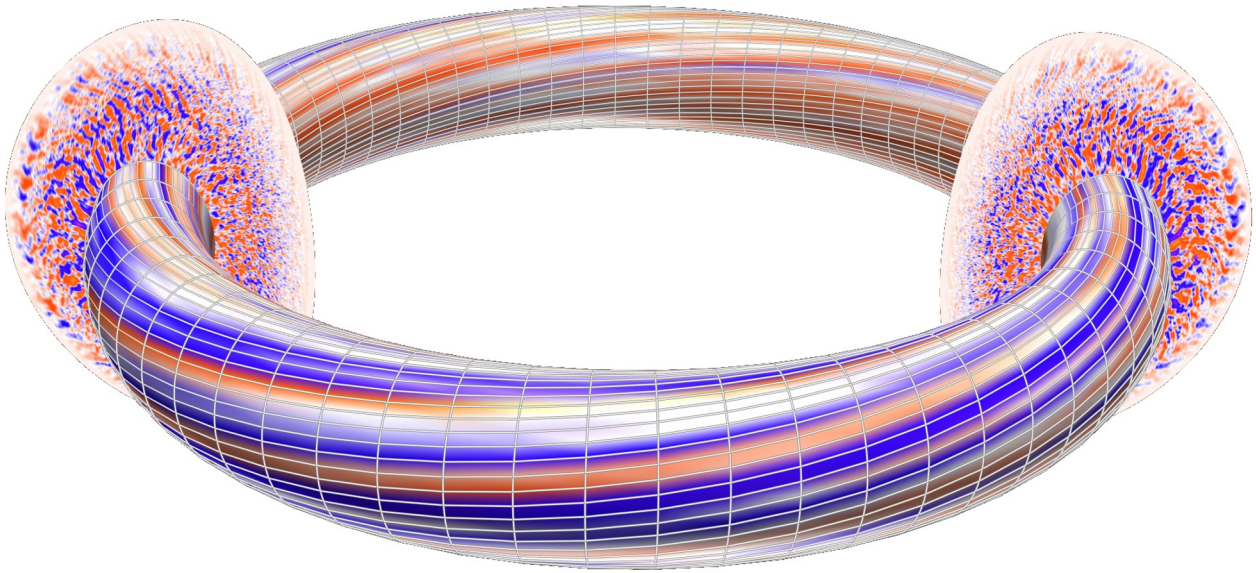
Sometimes yes, sometimes no: nonlinear, disordered, anisotropic, discontinuous coefficients, non-elliptic PDE, PDE systems, non-symmetric, indefinite, ...

Not appropriate for: dense, high-dimensional, small, singe-scale, …

Skills required for developing efficient MG algorithms

1. Choosing a good local iteration,
2. Choosing appropriate coarse-scale variables,
3. Choosing inter-scale transfer operators,
4. Constructing coarse-scale approximations to the fine-scale problem.

Case study



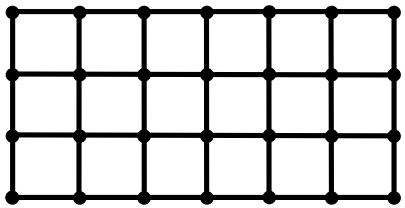
GmgPolar

Solver for the gyrokinetic Poisson equation

Domains

- Simple cases until now...

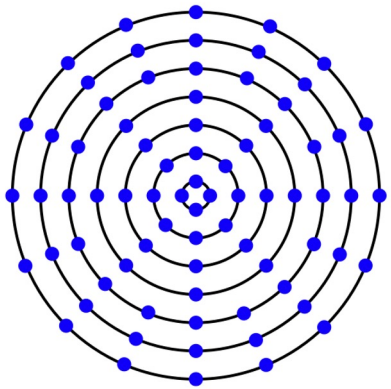
1D



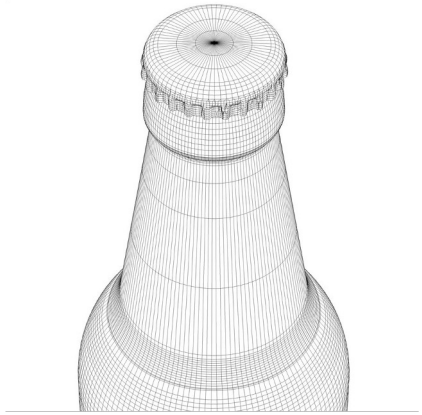
2D

- But more complex domains are frequent

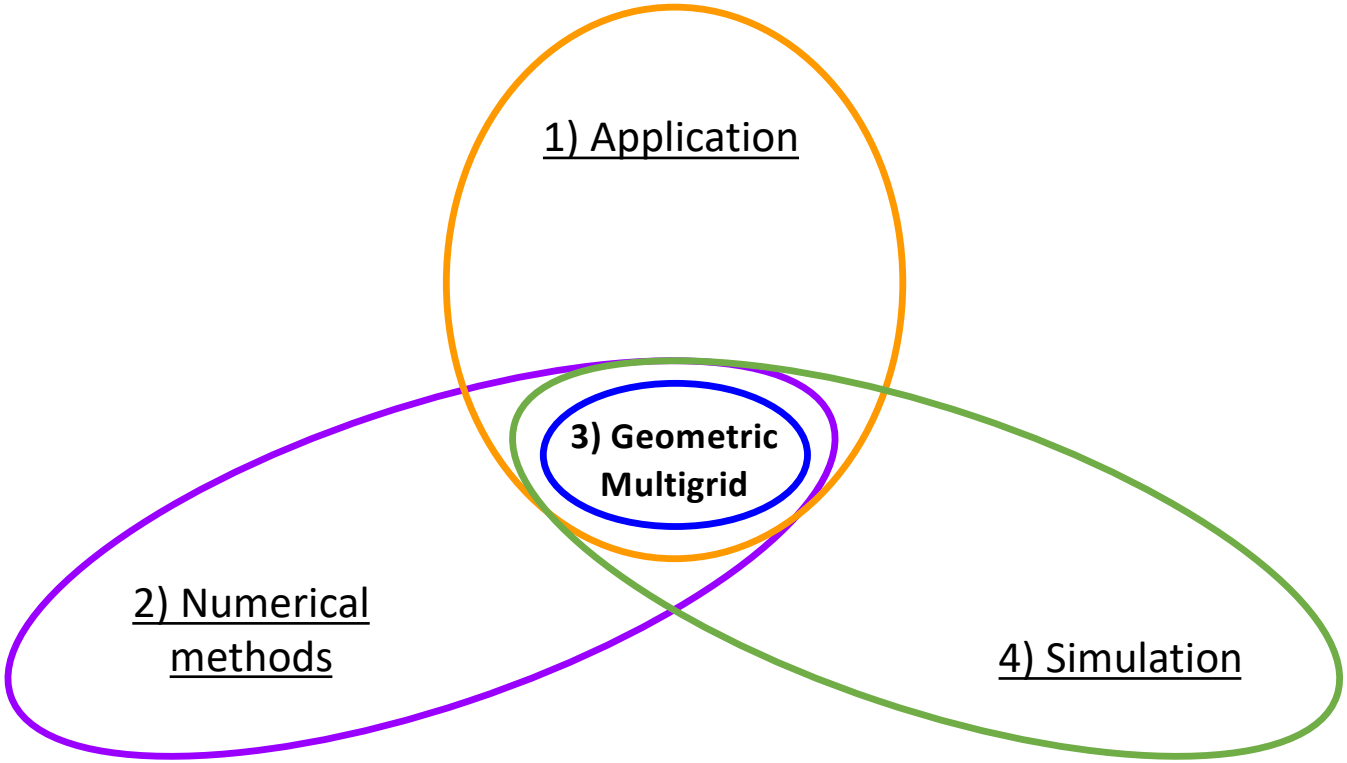
Unstructured

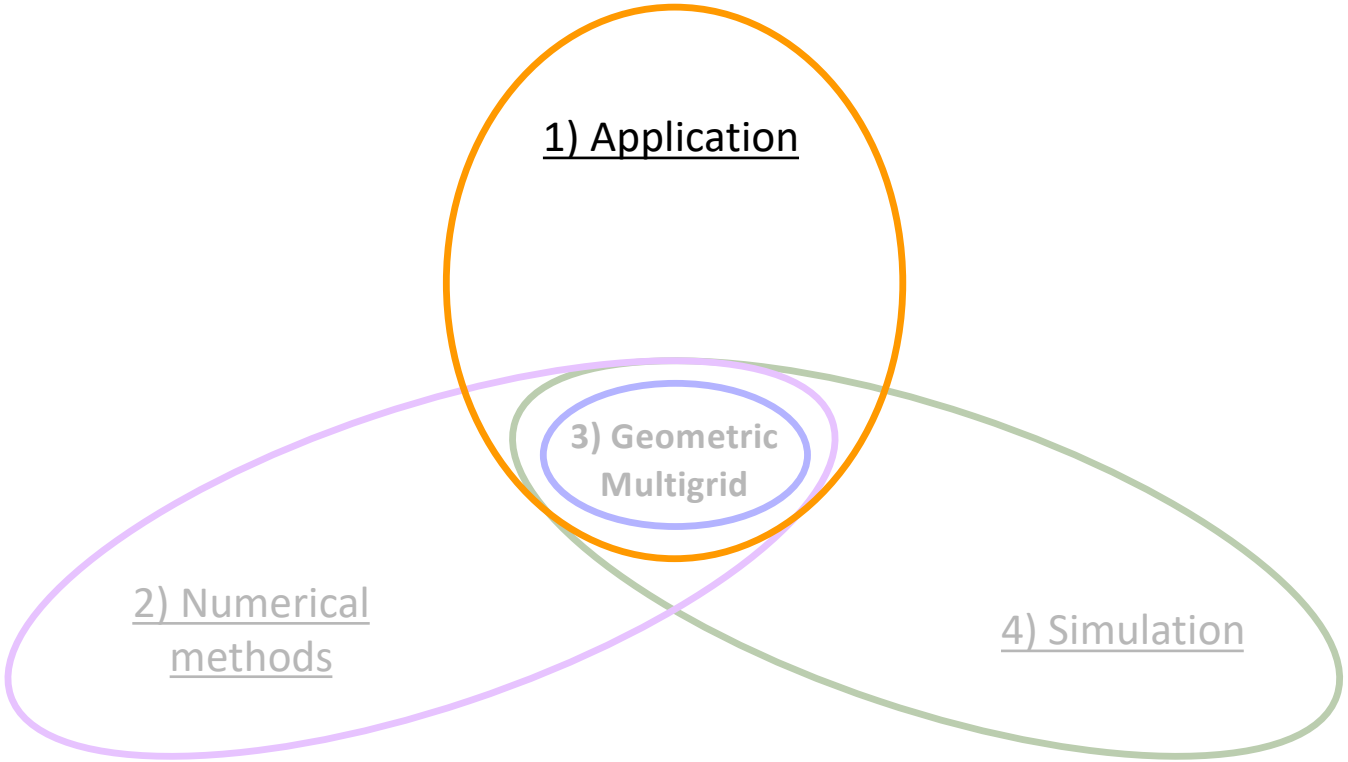


Circular



3D


1) Application
3) Geometric Multigrid
2) Numerical methods
4) Simulation


1) Application
3) Geometric Multigrid
2) Numerical methods
4) Simulation

Application: Plasma simulation

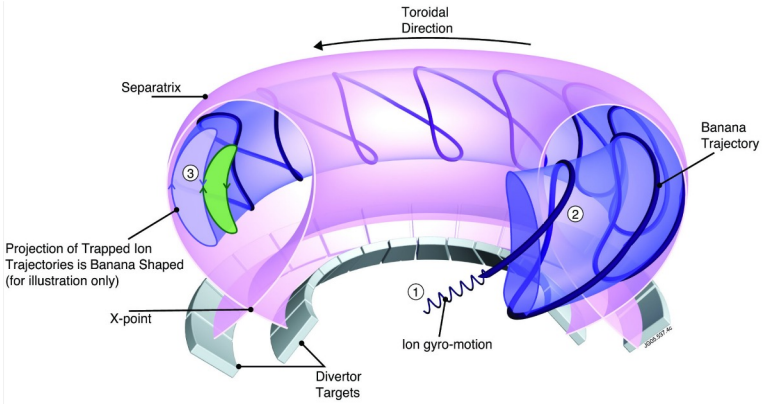
- Fusion reaction for energy in Tokamaks (e.g. ITER project)
  - Best hope for future "clean" energy
  - Real-life experiments from hard to impossible...

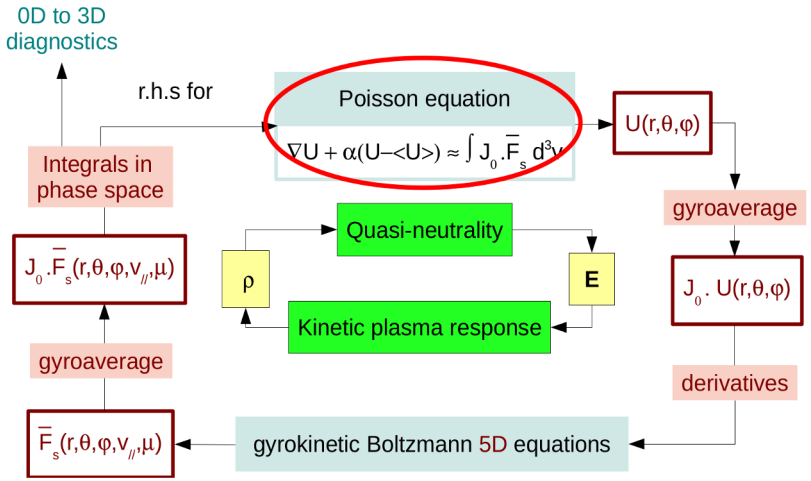
    ⇒ Numerical simulation !
- Plasma simulation
  - Solution to a 6D Vlaslov equation to solve (do not ask)
  - coupled with a linear 3D quasi-neutrality equation:

$$-\nabla(\alpha \cdot \nabla u) + \beta(u - < u >) = f$$

- Simulation code Gysela
  - Gyrokinetic: 6D → 5D
  - Issue: Linear system = slowest part of the code

- Task is part of the project Energy oriented center of excellence





How to solve the 3D equation

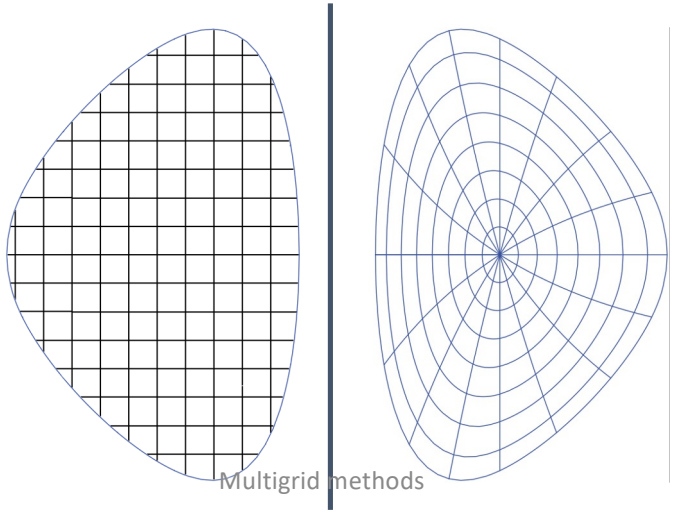
1. Projection onto the Fourier space: 3D → 2D
2. Simplify the PDE equation:

$$\begin{aligned} -\nabla(\alpha \cdot \nabla u) &= f, && \text{in } \Omega \\ u &= 0, && \text{in } \partial\Omega \end{aligned}$$

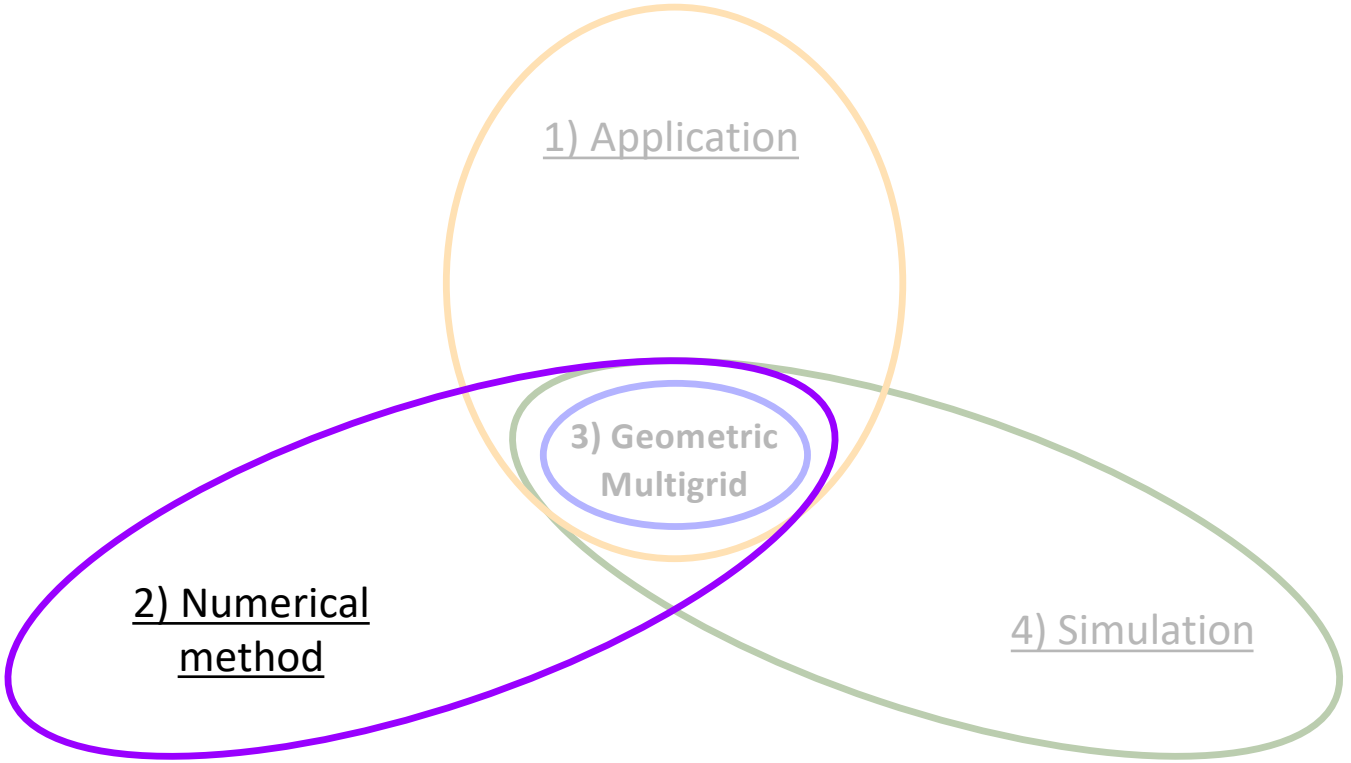
   - Often done in linear algebra ! Increase gradually the difficulty…
   - Each 2D problem corresponds to a disk-like cross section of the Tokamak

3. Solve these 2D "Poisson-like" equations. We have 2 possibilities for the grid:

- Cartesian
  - ➢ more out-of-the book
  - ➢ un-natural
  - ➢ (efficient smoother ?)

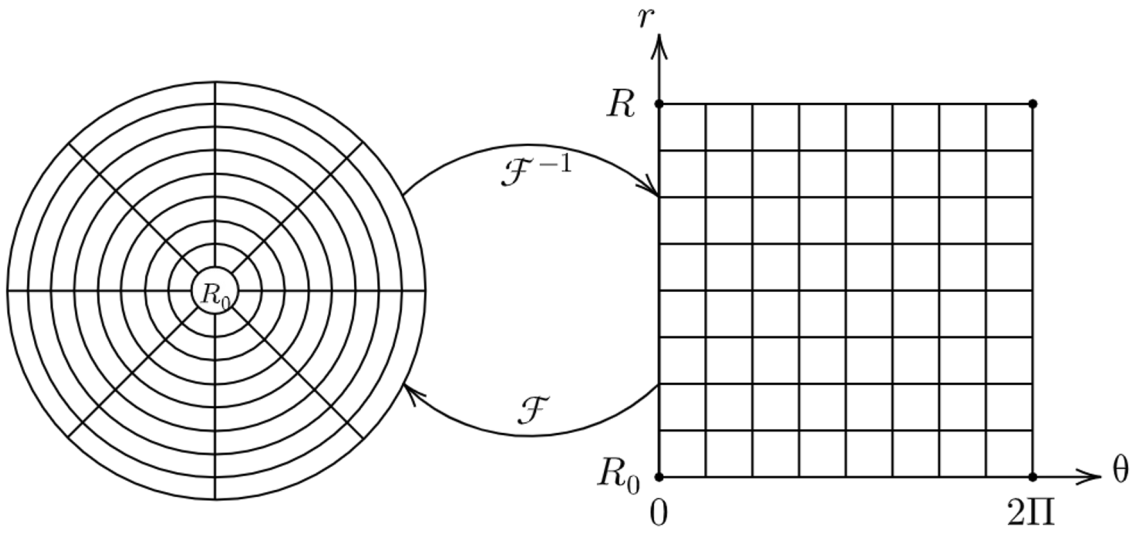


- Polar
  - ➢ very natural
  - ➢ still structured ! How ?
  - ➢ Can midpoint be handled?


1) Application
3) Geometric Multigrid
2) Numerical method
4) Simulation

Geometry handling

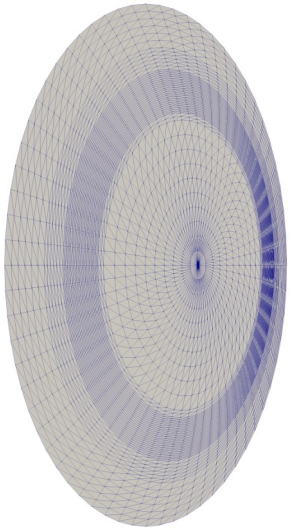
Coordinate transformation from standard polar to cartesian coordinates.



Attention: $R_0 > 0$ for a bijective mapping.

$$x = r\cos\theta$$
$$y = r\sin\theta$$

Disk-like geometry



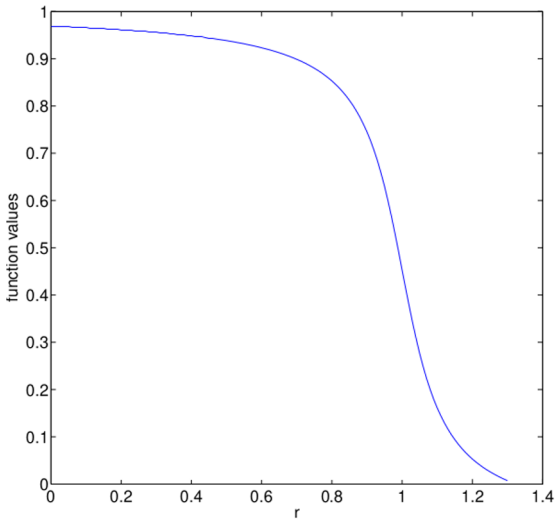
Stretched ellipse

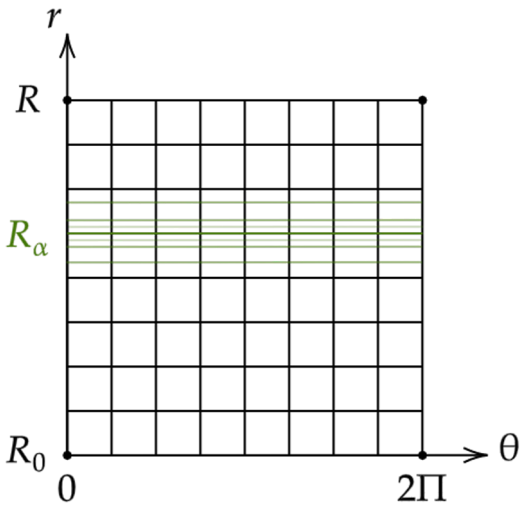
$$x = (1-\kappa)r\cos\theta - \delta r^2$$
$$y = (1-\kappa)r\sin\theta$$

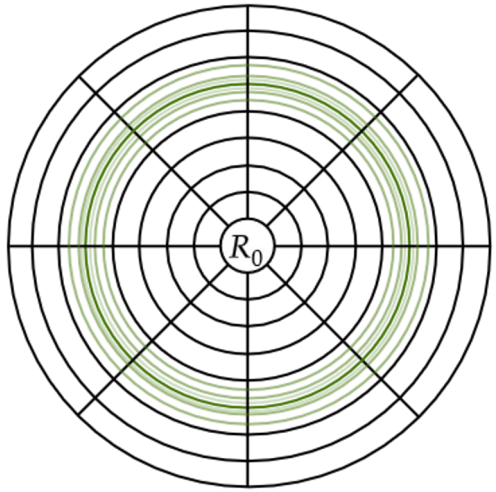
About the grid

$$-\nabla(\alpha\nabla u) = f \quad \text{in } \Omega$$
$$u = 0, \quad \text{on } \partial\Omega$$

- $R_0 > 0$ with Dirichlet boundary conditions
- α uniform in θ-direction, so mesh uniform too
- α has a steep decent in *r*-direction, so mesh anisotropic in *r*







$$\alpha(r) = \frac{2}{2.6 + 3.14}\left(1.3 + arctan\left(\frac{1-r}{0.09}\right)\right)$$

Poisson-like problem in polar coordinates

- The 2D Poisson-like problem in generalized polar coordinates:

$$-\nabla(\alpha(r)\nabla u) = -\frac{\partial \alpha(r)}{\partial r}\nabla u - \alpha(r)\nabla u = f$$

$$\nabla u = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial r^2}$$

- Straight forward finite difference discretization in r and θ would lead to a non-symmetric matrix.

- Transform the problem into a minimization problem

$$\min J(u) \Longleftrightarrow \frac{\partial}{\partial u} J(u) = 0$$

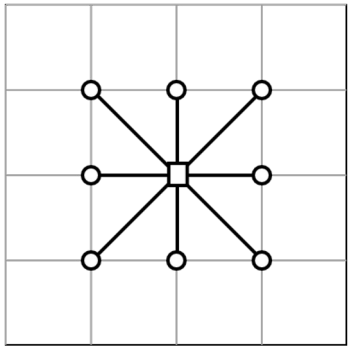
- Energy functional $J(u)$ is a self-adjoint operator ---> natural symmetry of the discretization

Symmetric discretization

- The energy functional is given by

$$J(u) := \int_{\Omega} \alpha \nabla u^T \nabla u - f u dx$$

- We discretize the problem using a 9-point stencil
- We obtain the symmetric linear system $Ax = b$ with $m = n_r\, n_\theta$ unknowns



$$u_{s+1,t} : (*9)_{s+1,t} = -\frac{k_t + k_{t-1}}{h_s} \frac{a^{rr}(s,t) + a^{rr}(s+1,t)}{2}, \quad u_{s+1,t+1} : (*9)_{s+1,t+1} = \frac{a^{r\theta}(s+1,t) + a^{r\theta}(s,t+1)}{4},$$

$$u_{s-1,t} : (*9)_{s-1,t} = -\frac{k_t + k_{t-1}}{h_{s-1}} \frac{a^{rr}(s-1,t) + a^{rr}(s,t)}{2}, \quad u_{s+1,t-1} : (*9)_{s+1,t+1} = \frac{a^{r\theta}(s,t-1) + a^{r\theta}(s+1,t)}{4},$$

$$u_{s,t+1} : (*9)_{s,t+1} = -\frac{h_s + h_{s-1}}{k_t} \frac{a^{\theta\theta}(s,t) + a^{\theta\theta}(s,t+1)}{2}, \quad u_{s-1,t+1} : (*9)_{s-1,t+1} = \frac{a^{r\theta}(s-1,t) + a^{r\theta}(s,t+1)}{4},$$

$$u_{s,t-1} : (*9)_{s,t-1} = -\frac{h_s + h_{s-1}}{k_{t-1}} \frac{a^{\theta\theta}(s,t-1) + a^{\theta\theta}(s,t)}{2}, \quad u_{s-1,t-1} : (*9)_{s-1,t-1} = \frac{a^{r\theta}(s-1,t) + a^{r\theta}(s,t-1)}{4},$$

$$u_{s,t} : (*9)_{s,t} = -\left[(*9)_{s+1,t} + (*9)_{s-1,t} + (*9)_{s,t+1} + (*9)_{s,t-1}\right],$$

No details here but note:

- scaling factors for the anisotropy
- $a^{rr}/a^{r\theta}/a^{\theta\theta}$: Jacobian of the previous mapping
- detDFinv also from mapping

Multigrid components

Standard coarsening, i.e. divide by 2.

- Then the degrees of freedom reduce by a factor of 4, $m_{l+1} \approx \frac{m_l}{4}$

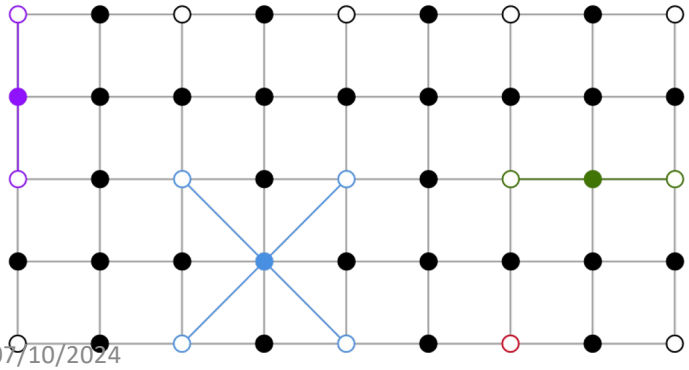
The coarse grid matrices are built by a discretization on each level

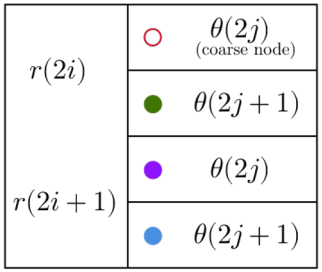
Prolongation operator

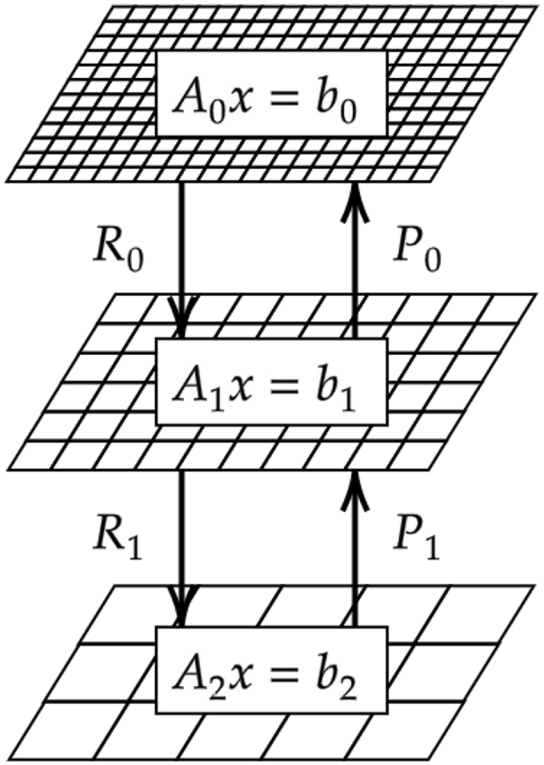
- Take average of neighboring nodes

Restriction operator

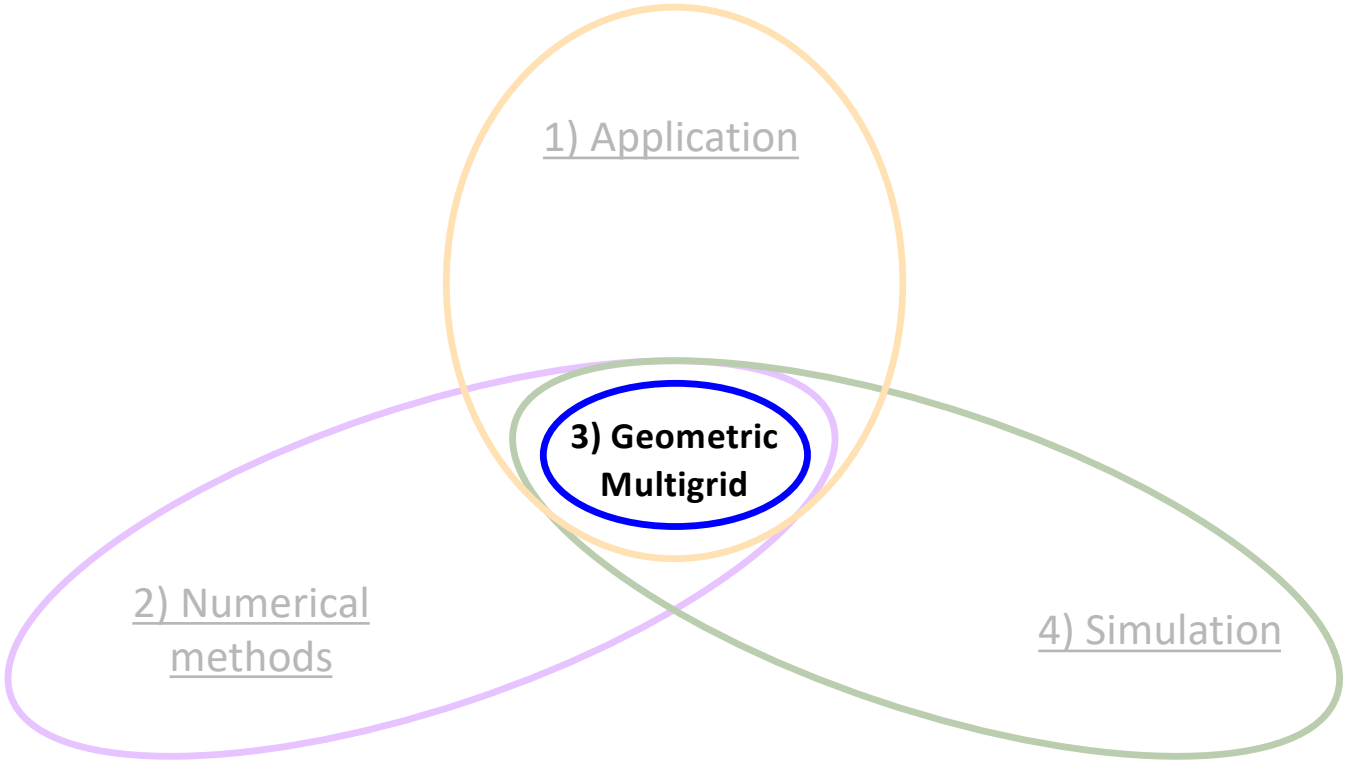
- Define it as $R_l = P_l^T$





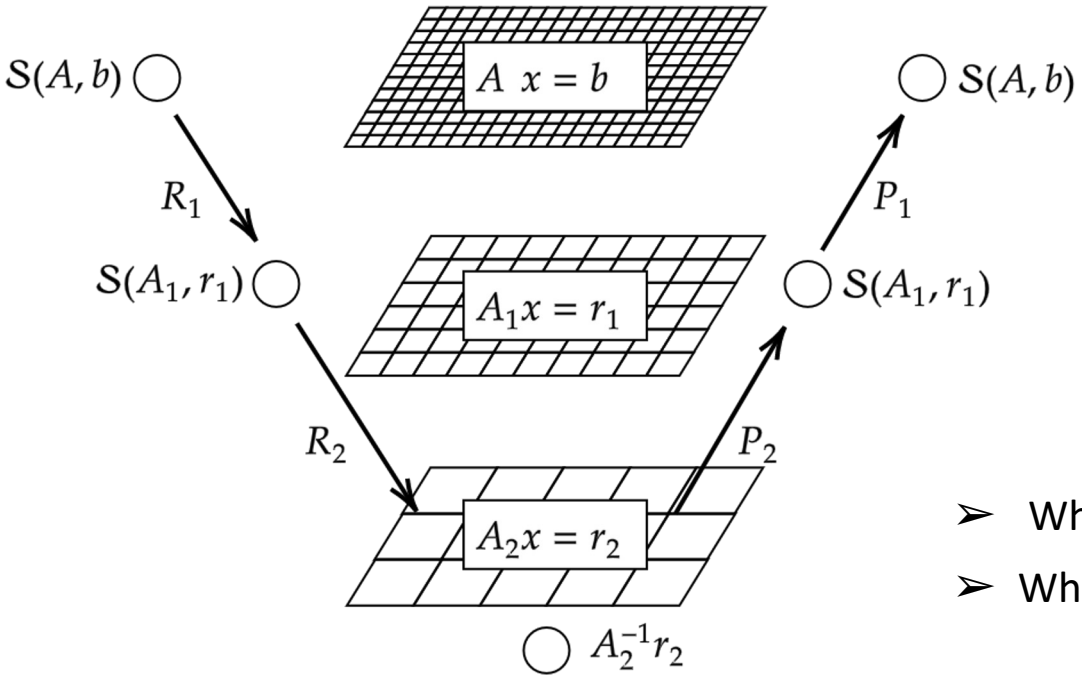


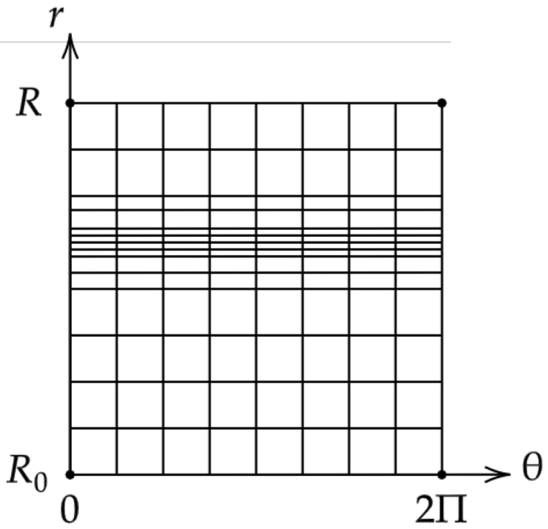
About this lecture



Geometric multigrid solver

✅ Grids ✅ Linear systems ✅ Prolongation/Restriction operator





- What could be a good smoother?
- What about the issue of the anisotropy in the *r* -direction?

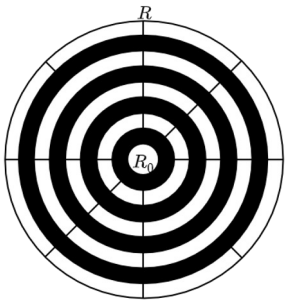
Handling the anisotropy

The convergence of multigrid solvers can be deteriorated by an anisotropy in the problem

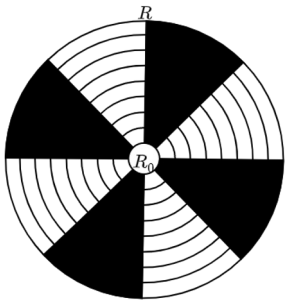
There are two typical possibilities to address this issue:

- Semi-coarsening (coarsen less in the direction of the anisotropy)
  - But here: How to coarsen when the anisotropy varies w.r.t. radius?
- Use a block smoother / line relaxation

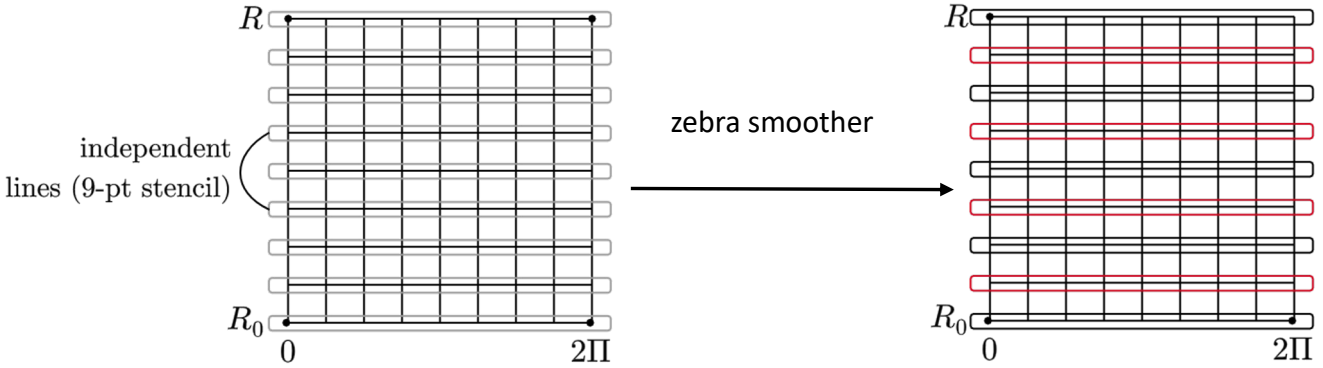
Zebra smoothers:



Circle



Radial

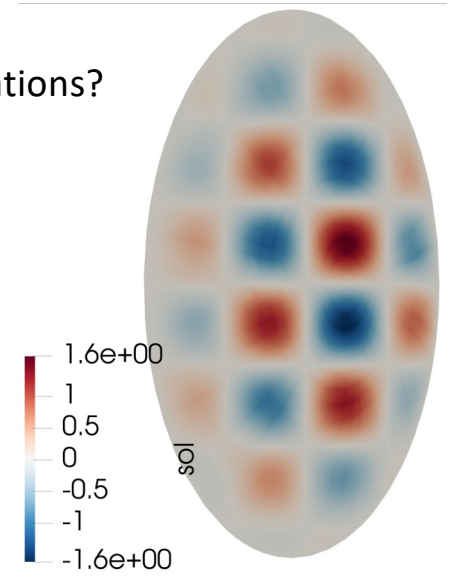


Experiment: GmgPolar with line relaxation

- We use as test problem the ellipse geometry with the manufactured solution

$$u(x,y) = (R^2 - r^2(x,y))\cos(2\pi x)\sin(2\pi x)$$

  - This solution is not aligned with polar grid. ---> Can the method capture these variations?
  - The solution is oscillating. –-> Does the method get penalized by this?



- Initial guess $u_0 = 0$ for MG solver. We monitor
- The residual $\mu = \dfrac{\|b - Au\|_2}{\|b - Au_0\|_2}$ and use $\mu < 10^{-8}$ as stopping tolerance
- The error $\|u - u^{(k)}\|$
- The residual reduction $\rho = \sqrt[it]{\dfrac{\|r_{it}\|}{\|r_0\|}}$

	circle smoothing				*radial smoothing*			
$n_r \times n_\theta$	its	$\widehat{\rho}$	$\|err\|_{\ell_2}$	$\|err\|_\infty$	its	$\widehat{\rho}$	$\|err\|_{\ell_2}$	$\|err\|_\infty$
49×64	150	0.98	7.6e-02	1.5e-01	150	0.95	7.1e-02	1.5e-01
97×128	150	0.98	3.4e-02	1.4e-01	150	0.97	1.8e-02	4.1e-02
193×256	150	0.98	3.0e-02	1.4e-01	150	0.97	4.7e-03	1.5e-02
385×512	150	0.98	2.9e-02	1.4e-01	150	0.97	1.6e-03	1.5e-02

Slow, slow, slow convergence (ρ≃1)

Why is the convergence so slow ?

All is caused by the effiency of the smoother

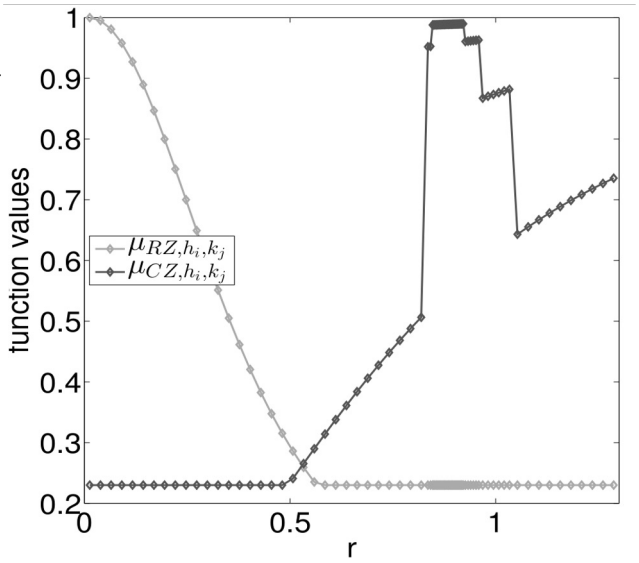
- The convergence of the zebra-smoother is highly dependent on (r, θ)
- In Barros 1988, the following convergence factors are given for annulus $(r_i, r_i + h_i) \times [0, 2\,\pi)$:

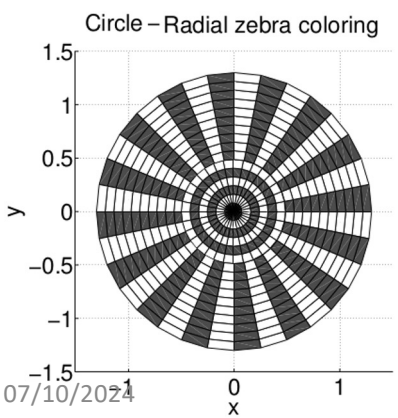
$$\mu_{CZ,h_i,k_j} = \max_{r_i \le r \le r_i + h_i} \left\{ \left(\frac{q_{i,j}^2 r^2}{1 + q_{i,j}^2 r^2} \right)^2, C_C \right\}$$

$$\mu_{RZ,h_i,k_j} = \max_{r_i \le r \le r_i + h_i} \left\{ \left(\frac{1}{1 + q_{i,j}^2 r^2} \right)^2, C_R \right\}$$

with

- $C_C \in \{0.23, 0.34\}, C_R \in \{0.23\}$
- $h_i = r_{i+1} - r_i$
- $k_j = \theta_{j+1} - \theta_j$
- $q_{ij} = \frac{k_i}{h_i}$





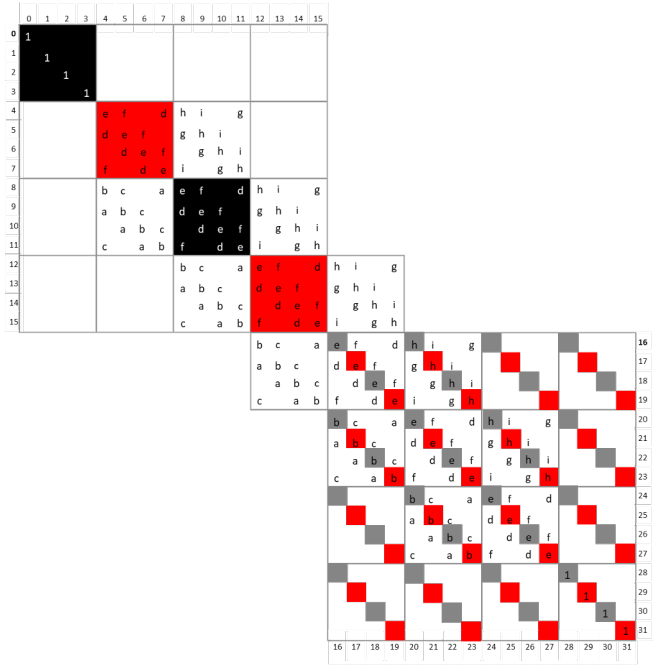
Use a hybrid Zebra Circle-Radial line smoother

- Non-overlapping decomposition
- Switch, when $q_{i,j}^2 r^2 > 1 \quad \Leftrightarrow \quad \frac{k_j}{h_i} r_i > 1$

The final algorithm - GmgPolar

MG(l, f, u)

- **Pre-smoothing** (block Gauss-Seidel)
  - For smoother in {circle, radial} do
    - For color in {black, White} do
    - $u_{sc} = A_{sc}^{-1}(f_{sc} - A_{sc}^{\perp}u)$
- **Coarse grid correction**
  - Compute and restrict residual $r = P^T(f - Au)$
  - If $l = 1$:
    coarsest grid, then solve problem directly
  - If $l > 1$: **MG**(l-1, f, u)
- **Prolongation and update**: $u = u + Pe$
- **Post-smoothing** (as pre-smoothing)



Notation:

- A is matrix on level l.
- A_{sc} and f_{sc} are the set of nodes of A and f for type and color.
- $A_{sc}^{\perp}$ all the other nodes.

Numerical experiments: GmgPolar with Circle-Radial smoother

- Only circle or radial smoothing (seen earlier):

	circle smoothing				*radial smoothing*			
$n_r \times n_\theta$	its	$\widehat{\rho}$	$\|err\|_{\ell_2}$	$\|err\|_\infty$	its	$\widehat{\rho}$	$\|err\|_{\ell_2}$	$\|err\|_\infty$
49×64	150	0.98	7.6e-02	1.5e-01	150	0.95	7.1e-02	1.5e-01
97×128	150	0.98	3.4e-02	1.4e-01	150	0.97	1.8e-02	4.1e-02
193×256	150	0.98	3.0e-02	1.4e-01	150	0.97	4.7e-03	1.5e-02
385×512	150	0.98	2.9e-02	1.4e-01	150	0.97	1.6e-03	1.5e-02

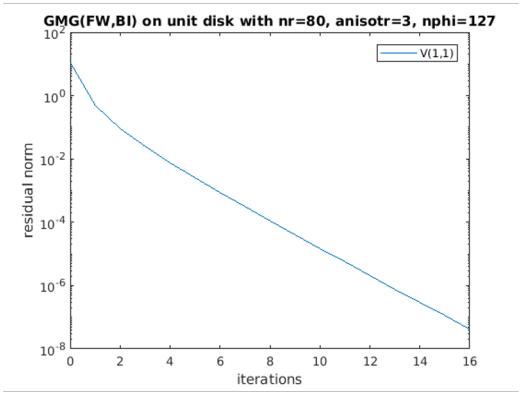
- Hybrid circle-radial smoother

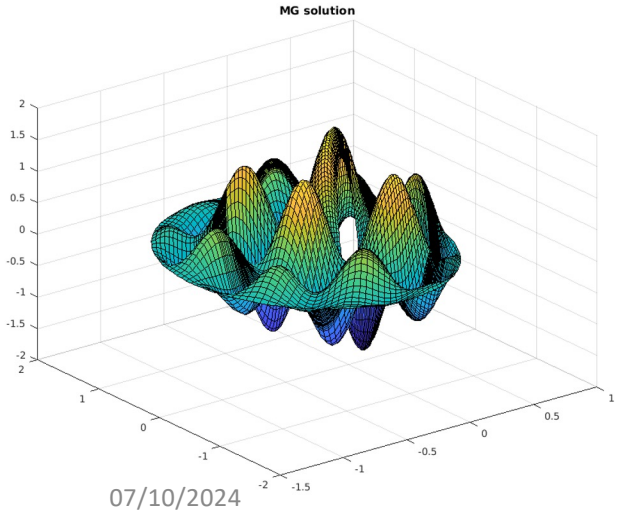
	optimized smoothing			
$n_r \times n_\theta$	its	$\widehat{\rho}$	$\|err\|_{\ell_2}$	$\|err\|_\infty$
49×64	46	0.67	7.1e-02	1.5e-01
97×128	45	0.66	1.8e-02	4.1e-02
193×256	44	0.66	4.5e-03	1.1e-02
385×512	44	0.65	1.1e-03	2.6e-03

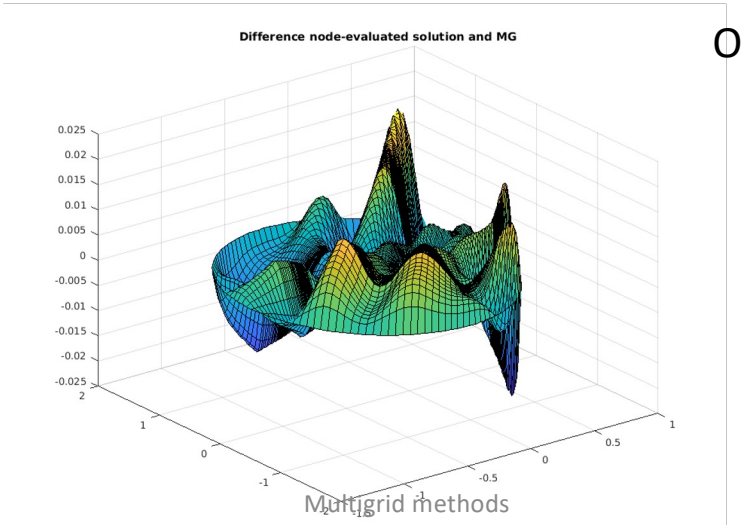
Much better !

In practice: GmgPolar with Circle-Radial smoother

	optimized smoothing			
$n_r \times n_\theta$	its	$\widehat{\rho}$	$\|err\|_{\ell_2}$	$\|err\|_{\infty}$
49×64	46	0.67	7.1e-02	1.5e-01
97×128	45	0.66	1.8e-02	4.1e-02
193×256	44	0.66	4.5e-03	1.1e-02
385×512	44	0.65	1.1e-03	2.6e-03



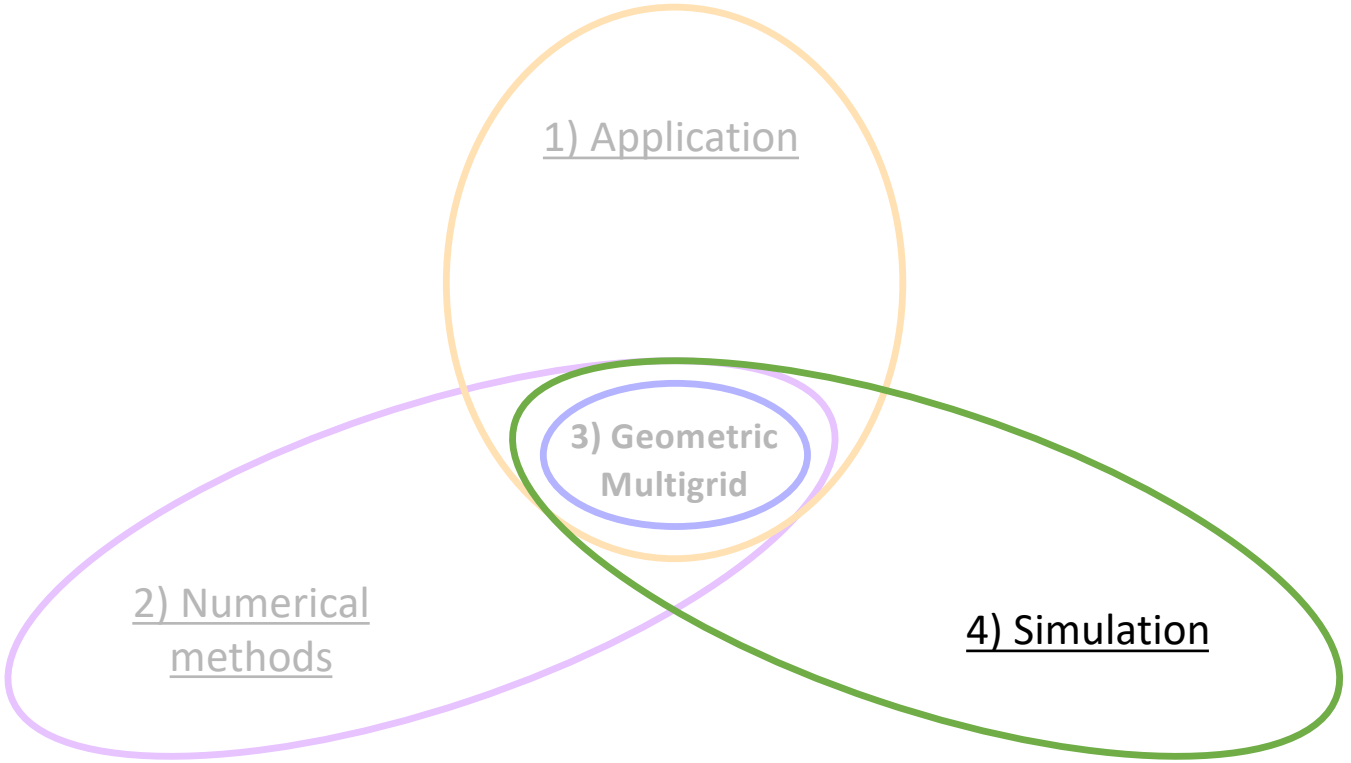




Order of approximation:

- $ord = \dfrac{log(e(k-1)/e(k))}{log(\sqrt{m(k)/m(k-1)})}$
- Expresses how close we solve the actual PDE problem using the MG solve: err = $\mathcal{O}(h^{ord})$

Gmgpolar gives ord = 2


1) Application
3) Geometric Multigrid
2) Numerical methods
4) Simulation

Optimal complexity

Multigrid solvers are said to be of optimal complexity, if

- The computational complexity of 1 iteration is linear w.r.t. the size of the problem, i.e. $\mathcal{O}(m)$,
- Their convergence is mesh independent.

For GmgPolar: We have seen empirically that we have mesh-independent convergence.

Consider again MG(l, b, u):

- $u = S(A_l, b, u)$
- $r_c = R_l(b - A_l u)$
- $e_c = A_L^{-1} r_c$, if l = L-1
- $MG(l+1, r_c)$, else
- $u = S(A_l, b, u + P e_c)$

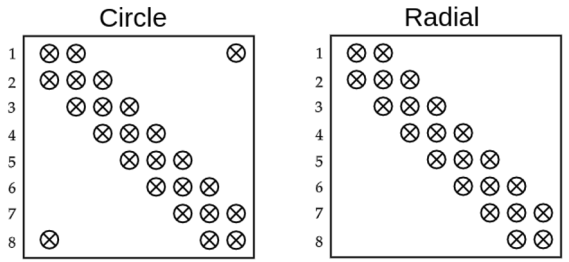
S (A_l, b, u):

- for *(s, c)* in *{Circle, Radial}* × *{Black, White}*
- $r_{sc} = (f_{sc} - A_{sc}^{\perp} u)$
- $u_{sc} = A_{sc}^{-1} r_{sc}$

- Matrices: A_l ($nz_r = 9$), P_l ($nz_r = 7/2$), A_{sc} ($nz_r = 3$), $A_{sc}^{\perp}$ ($nz_r = 6$)
  - Construction per element: A_l: ≈7 flops, P_l: ≈2 flops, A_{sc}: ≈14 flops, $A_{sc}^{\perp}$: ≈8 flops
  - Application per element: 2 flops

  ⇒ total=nz_r(cost+2)m=$\mathcal{O}$(m)
- Coarsest grid solve A_L^{-1} using Cholesky decomposition: $\mathcal{O}(m_L^3/3)$... BUT $m_L \approx m/4^L \ll m$!! (neglectable with enough levels)
- Gauss Seidel's $A_{sc}^{-1} r_{sc}$: $\mathcal{O}(12 m_{sc})$ for Circle and $\mathcal{O}(8 m_{sc})$ for Radial

Optimal complexity



- The matrices A_{sc} have specific structures:
  - Radial: tridiagonal structure (known linear)
  - Circle: tridiagonal + periodicity
- Example: How to factorise $A_{Circle,\ c} = LU$? $\Rightarrow \mathcal{O}(8m_{sc})$ flops

It can be shown after many calculations that

$$W(MG) = \frac{4iter}{3}[(\nu_1 + \nu_2)\mathfrak{F}(smoother\ apply) + \mathfrak{F}(residual)] \times m,$$

with $\mathfrak{F}(smoother\ apply)$ and $\mathfrak{F}(residual)$ being constant.

Conclusion: beyond ?