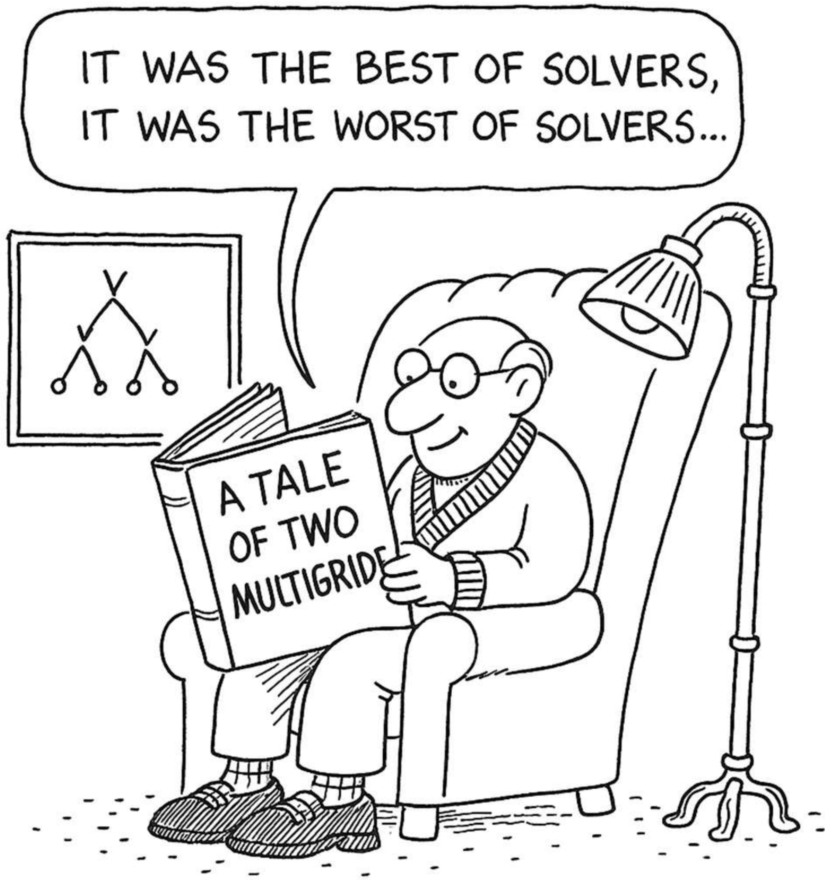


RUHR-UNIVERSITÄT BOCHUM

A TALE OF TWO MULTIGRIDS

A- and P- Multigrid Methods in Vector-Valued PDEs

Abdelrahman Fathy Abdelrahman
CompEng Master Student

Problem Setup

$$Ax = b, \qquad find\ x = ?$$

$$\boldsymbol{A}$$

- **Sparse:**

Direct solvers (e.g., LU) ❌

- **PDE:** (e.g., $Div(\sigma) + b = 0$)

Conventional iterative methods will slow down as these systems grow larger

Problem Setup

So, we need another sparse iterative solvers such that:

- Designed for PDEs (more or less)
- Independent of mesh size (at least in theory)

Luckily, we have:

- Multigrid methods
  - Designed initially for elliptic PDEs
  - Extended to handle other PDEs

julia

AlgebraicMultigrid.jl

(AMG.jl)

But the grass isn‘t always greener on the other side !

The Twist: AMG.jl isn‘t Perfect

Current Problems in AlgebraicMultigrid.jl

- supports only for scalar-valued PDEs (e.g., Poisson’s equation $-\Delta u(x) = f(x)$)
- suffocates when dealing with systems come from higher order basis functions

Proposed Solutions

- Extend AMG.jl to handle vector-valued PDEs
  - Add interface to accept user-defined near null space (nns)
- Develop FerriteMultigrid.jl: an implementation of polynomial multigrid methods
  - Based on:
    - Ferrite.jl
    - AlgebraicMultigrid.jl

Roadmap

- **Multigrid Methods 101**
  - Basic Iterative Methods as Smoothers
  - Error Behaviors Across Grids
  - The Two-Level Method
- **Smoothed aggregation (SA) in Algebraic Multigrid (AMG)**
  - Fixed Near Null Space (NNS)
  - User Defined Near Null Space (NNS)
  - Numerical Experiments
- **FerriteMultigrid.jl: P- Multigrid Extension**
  - Why and How?
  - Coarsening Strategies
  - Package Interface
  - Numerical Experiments
- **Future Work and Possible Extensions**

Roadmap

- **Multigrid Methods 101**
  - Basic Iterative Methods as Smoothers
  - Error Behaviors Across Grids
  - The Two-Level Method
- **Smoothed aggregation (SA) in Algebraic Multigrid (AMG)**
  - Fixed Near Null Space (NNS)
  - User Defined Near Null Space (NNS)
  - Numerical Experiments
- **FerriteMultigrid.jl: P- Multigrid Extension**
  - Why and How?
  - Coarsening Strategies
  - Package Interface
  - Numerical Experiments
- **Future Work and Possible Extensions**

Basic Iterative Methods as Smoothers

$$Ax = b$$

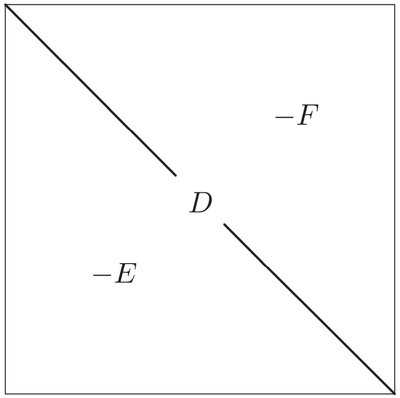
$$A \coloneqq M - N$$

Update eq. : $x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b$

Error eq.: $e^{(k+1)} = Ge^{(k)}, \qquad where\ G = I - M^{-1}A$

NOTE:

- Easy to compute M^{-1}
- E.g.,
  - Jacobi: $M = D$
  - **Forward** Gauss-Seidel: $M = D - E$
  - **Backward** Gauss-Seidel: $M = D - F$



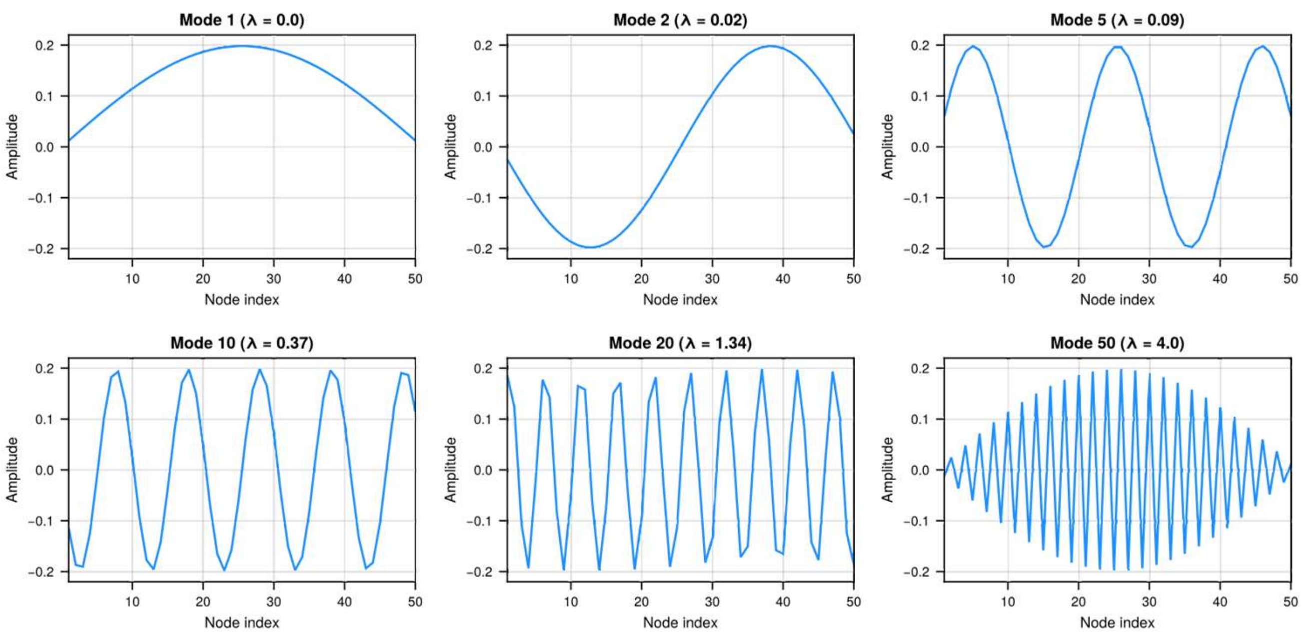
Basic Iterative Methods as Smothers

Error propagation:

- Consider 1D Poisson problem ($-u''(x) = f(x)$) with 50 interior points.
- Selected eigenmodes:

REMEMBER:

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b$$
$$e^{(k+1)} = (I \ - M^{-1}A)e^{(k)}$$



Low frequency — **High** frequency

Smooth error — Oscillatory error

Basic Iterative Methods as Smoothers

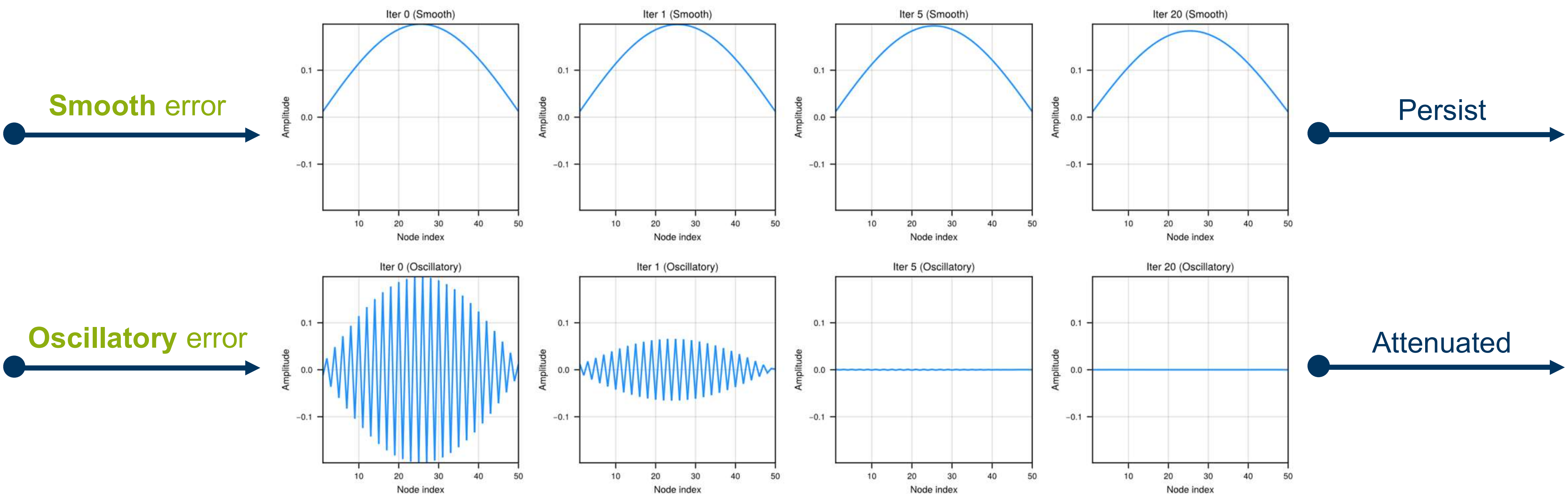
Error propagation:

- Apply Gauss-Seidel error equation multiple times:

REMEMBER:

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b$$

$$e^{(k+1)} = (I - M^{-1}A)e^{(k)}$$

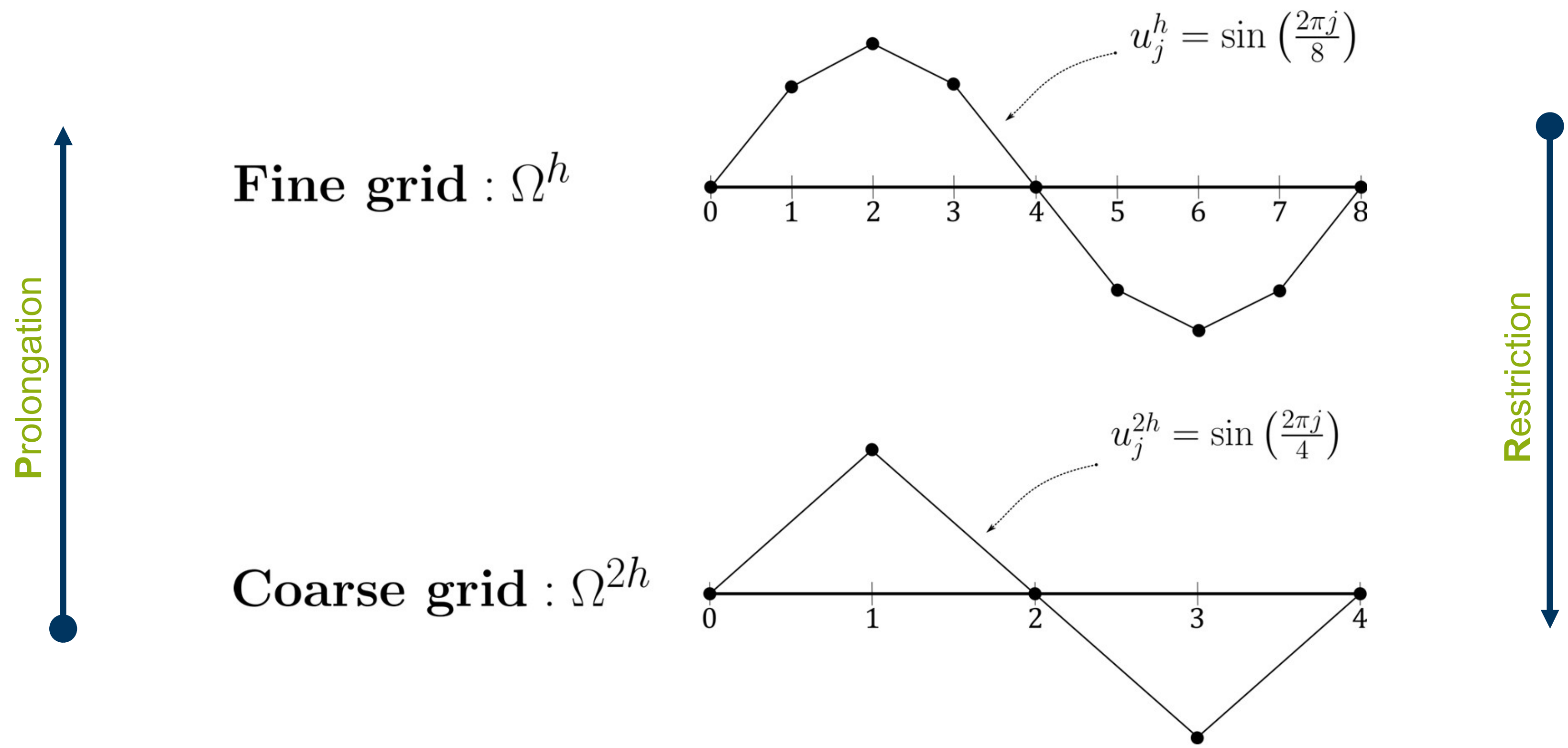


Error Behavior Across Grids

- Compare between the same mode on different grids:
  - E.g., **Second** mode on a **fine** grid ($\boldsymbol{\Omega^h}$, $n = 7$) and a **coarse** grid ($\boldsymbol{\Omega^{2h}}$, $n = 3$)
  - Frequency increases as we go from fine to coarse grid.

REMEMBER:

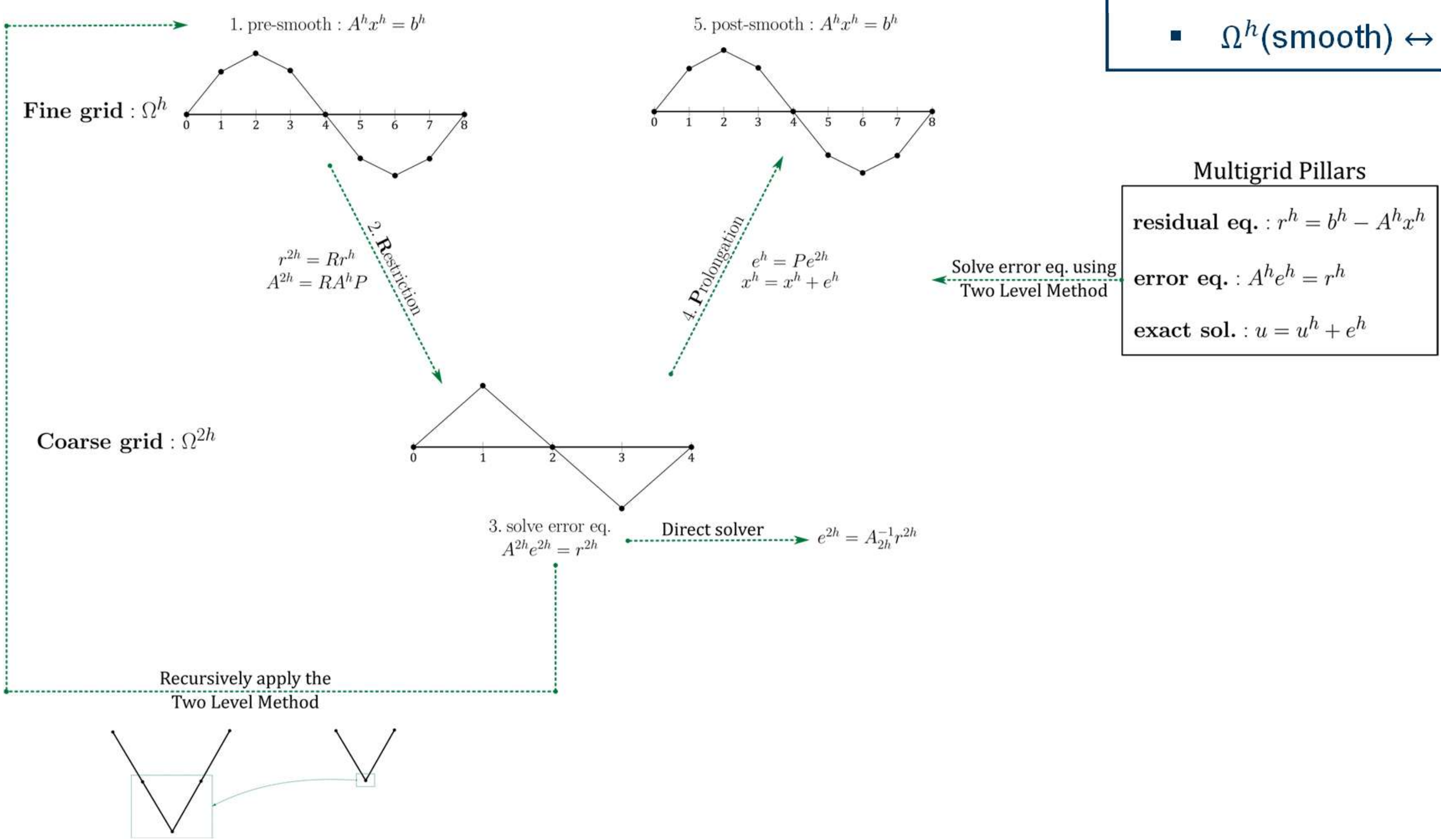
- **By smoothing**
  - **Low** freq.→ persist
  - **High** freq. → attenuated



The Two-Level Method

Ensure Robustness:

- Define near null space B to be equivalent to the smooth error.
- P & R operators must span B



REMEMBER:

- **By smoothing**
  - **Low** freq. → persist
  - **High** freq. → atenuated
- **By coarsening**
  - Ω^h(smooth) ↔ Ω^{2h} (oscillatory)

Roadmap

- **Multigrid Methods 101**
  - Basic Iterative Methods as Smoothers
  - Error Behaviors Across Grids
  - The Two-Level Method
- **Smoothed aggregation (SA) in Algebraic Multigrid (AMG)**
  - Fixed Near Null Space (NNS)
  - User Defined Near Null Space (NNS)
  - Numerical Experiments
- **FerriteMultigrid.jl: P- Multigrid Extension**
  - Why and How?
  - Coarsening Strategies
  - Package Interface
  - Numerical Experiments
- **Future Work and Possible Extensions**

SA - AMG

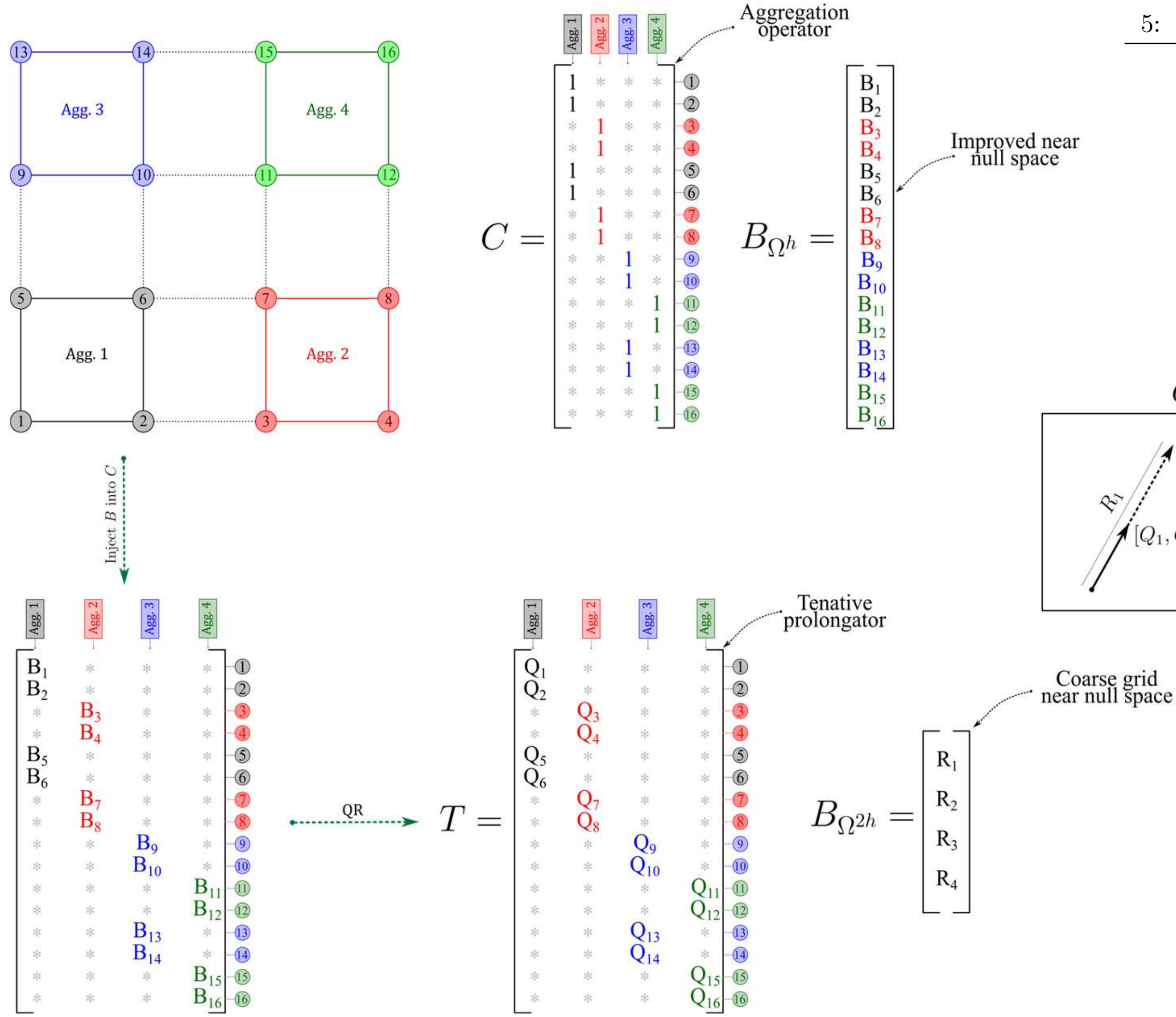
NOTE

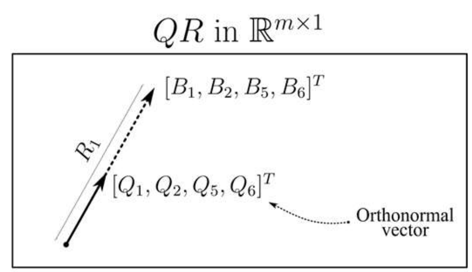
- Two primary ways to construct prolongation and restriction operators:
  - Smoothed Aggregation (our focus ✅)
  - C/F splitting methods (e.g., Ruge-Stuben)

Fixed Near Null Space (NNS)

Algorithm 2 smoothed_aggregation(A)

$B = \texttt{ones}(\texttt{size}(A,1))$
while $\texttt{size}(A) < \texttt{max\_coarse}$ **do**
  $A_{\Omega^{2h}}, B_{\Omega^{2h}}, \texttt{level} = \texttt{extend\_hierarchy}(A, B)$
  $A \leftarrow A_{\Omega^{2h}}, B \leftarrow B_{\Omega^{2h}}$
end while





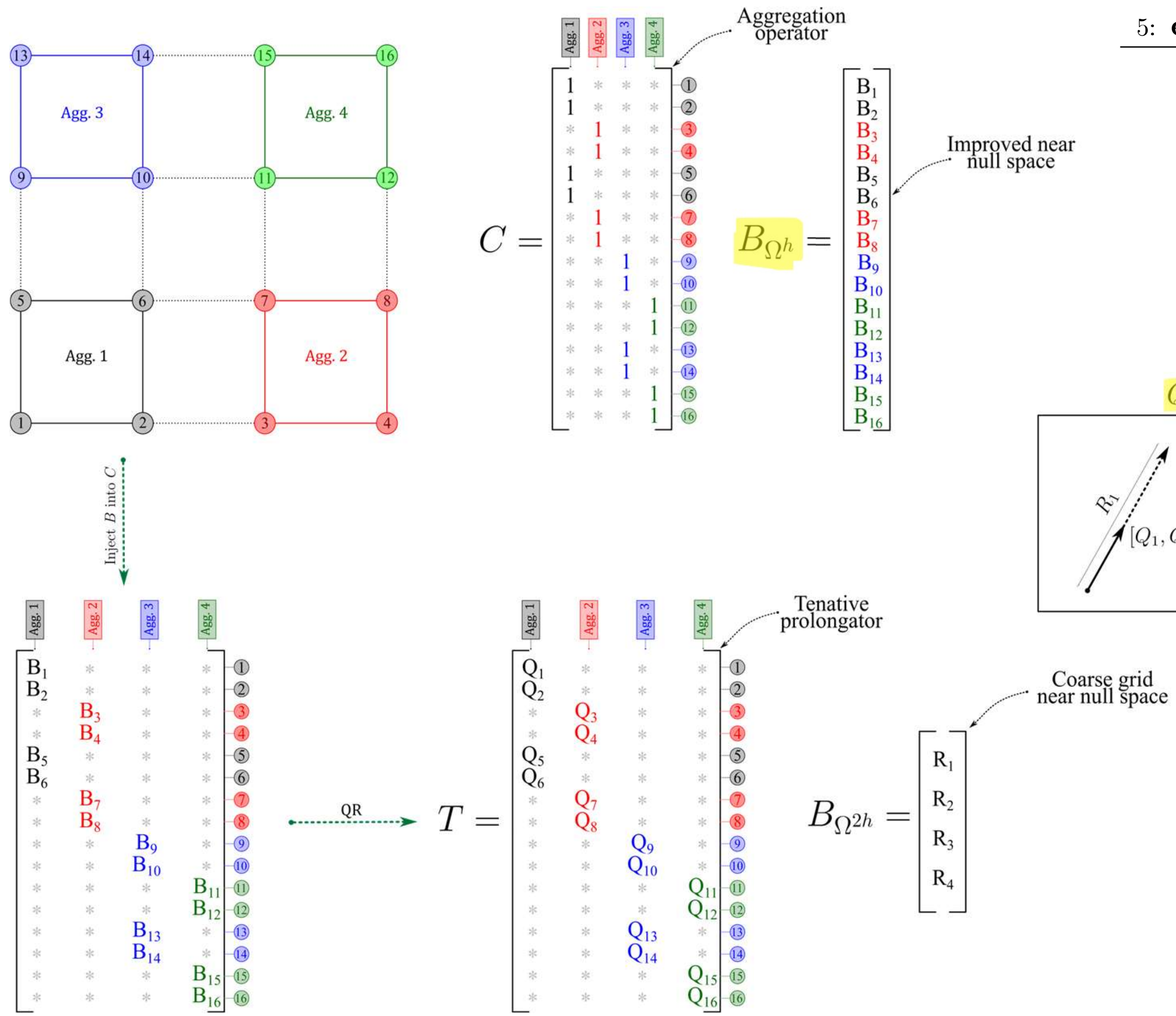
Algorithm 1 extend_hierarchy(A, B)

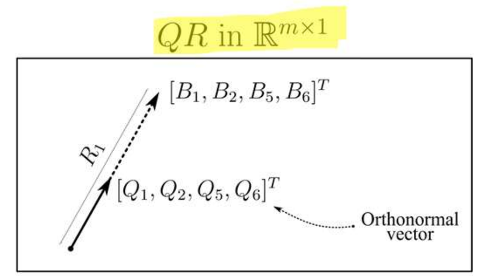
$S = \texttt{strength}(A)$
$C = \texttt{aggregate}(S)$
$b = \texttt{zeros}(\texttt{size}(A,1), \texttt{size}(B,2))$
$B \leftarrow \texttt{improve\_candidates}(A, B, b)$
$T, B_{\Omega^{2h}} = \texttt{fit\_candidates}(C, B)$
$P = \texttt{smooth}(A, T)$
$R = P^T$
$A_{\Omega^{2h}} = RAP$
return $A_{\Omega^{2h}}, B_{\Omega^{2h}}, \texttt{Level}(A, P, R)$

Fixed Near Null Space (NNS)

Algorithm 2 smoothed_aggregation(A)

$B = \texttt{ones}(\texttt{size}(A, 1))$
while $\texttt{size}(A) < \texttt{max\_coarse}$ **do**
  $A_{\Omega^{2h}}, B_{\Omega^{2h}}, \texttt{level} = \texttt{extend\_hierarchy}(A, B)$
  $A \leftarrow A_{\Omega^{2h}}, B \leftarrow B_{\Omega^{2h}}$
end while





Algorithm 1 extend_hierarchy(A, B)

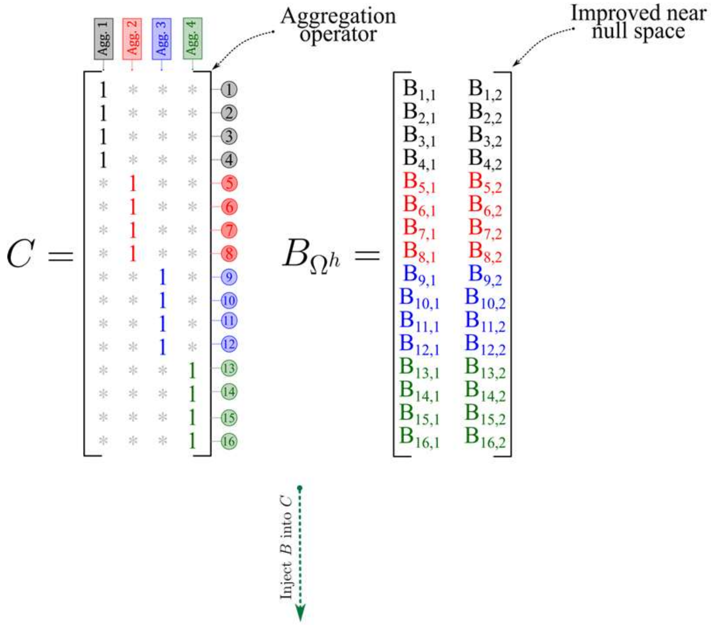
$S = \texttt{strength}(A)$
$C = \texttt{aggregate}(S)$
$b = \texttt{zeros}(\texttt{size}(A, 1), \texttt{size}(B, 2))$
$B \leftarrow \texttt{improve\_candidates}(A, B, b)$
$T, B_{\Omega^{2h}} = \texttt{fit\_candidates}(C, B)$
$P = \texttt{smooth}(A, T)$
$R = P^T$
$A_{\Omega^{2h}} = RAP$
return $A_{\Omega^{2h}}, B_{\Omega^{2h}}, \texttt{Level}(A, P, R)$

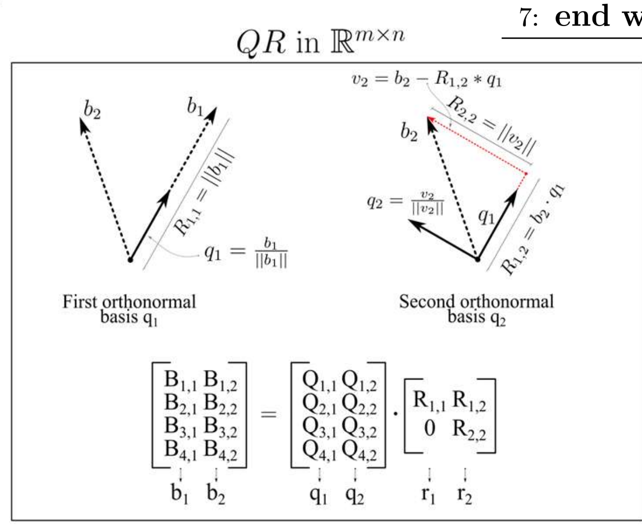
RUHR UNIVERSITÄT BOCHUM

User Defined Near Null Space

Algorithm 3 `smoothed_aggregation`(A; B = nothing)

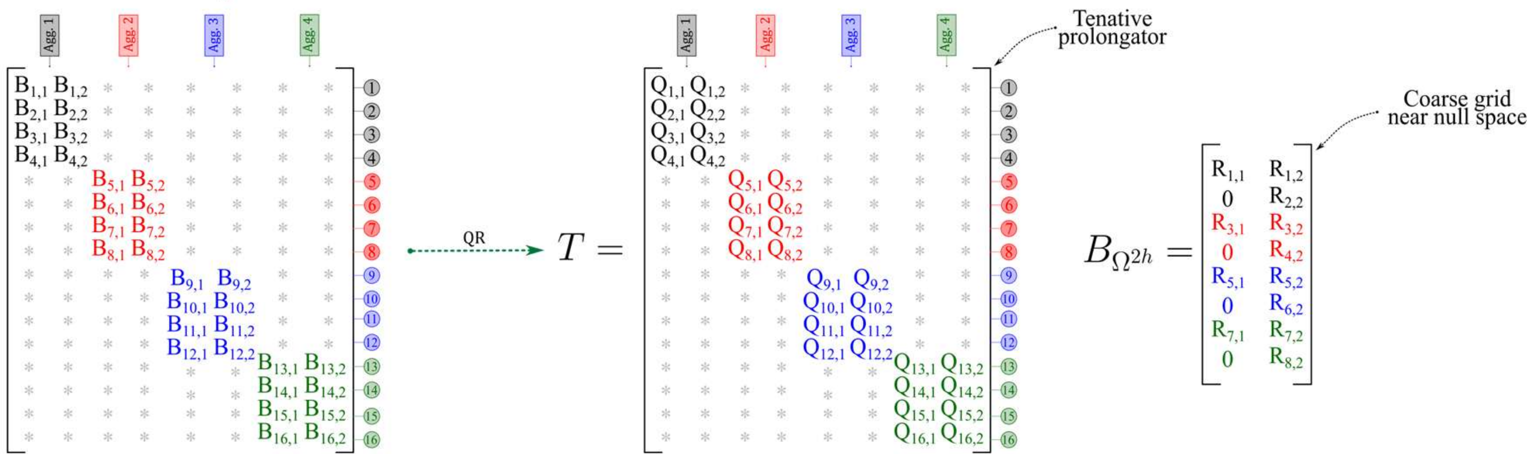
```
if B == nothing then
    B = ones(size(A, 1))
end if
while size(A) < max_coarse do
    A_Ω2h, B_Ω2h, level = extend_hierarchy(A, B)
    A ← A_Ω2h, B ← B_Ω2h
end while
```





Algorithm 1 `extend_hierarchy`(A, B)

```
S = strength(A)
C = aggregate(S)
b = zeros(size(A, 1), size(B, 2))
B ← improve_candidates(A, B, b)
T, B_Ω2h = fit_candidates(C, B)
P = smooth(A, T)
R = P^T
A_Ω2h = RAP
return A_Ω2h, B_Ω2h, Level(A, P, R)
```

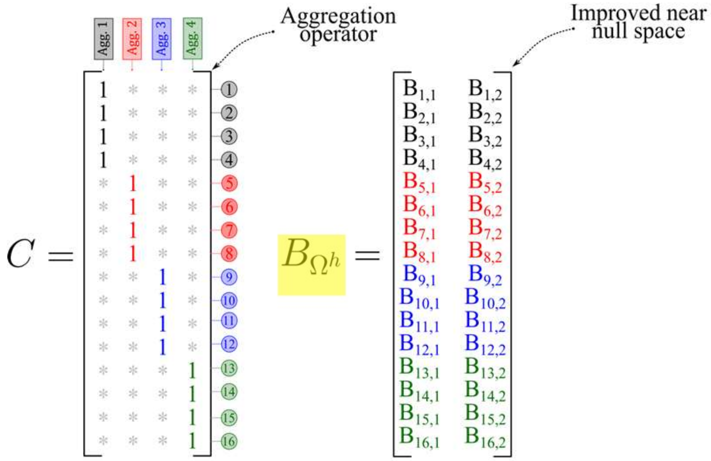


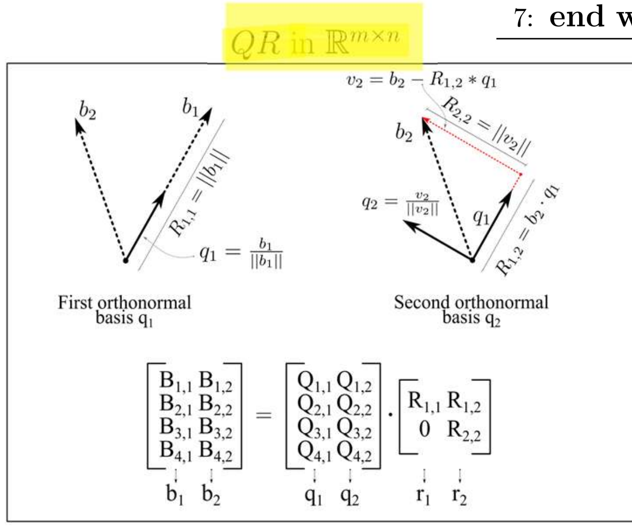
RUHR UNIVERSITÄT BOCHUM RUB

User Defined Near Null Space

Algorithm 3 `smoothed_aggregation`(A; B = nothing)

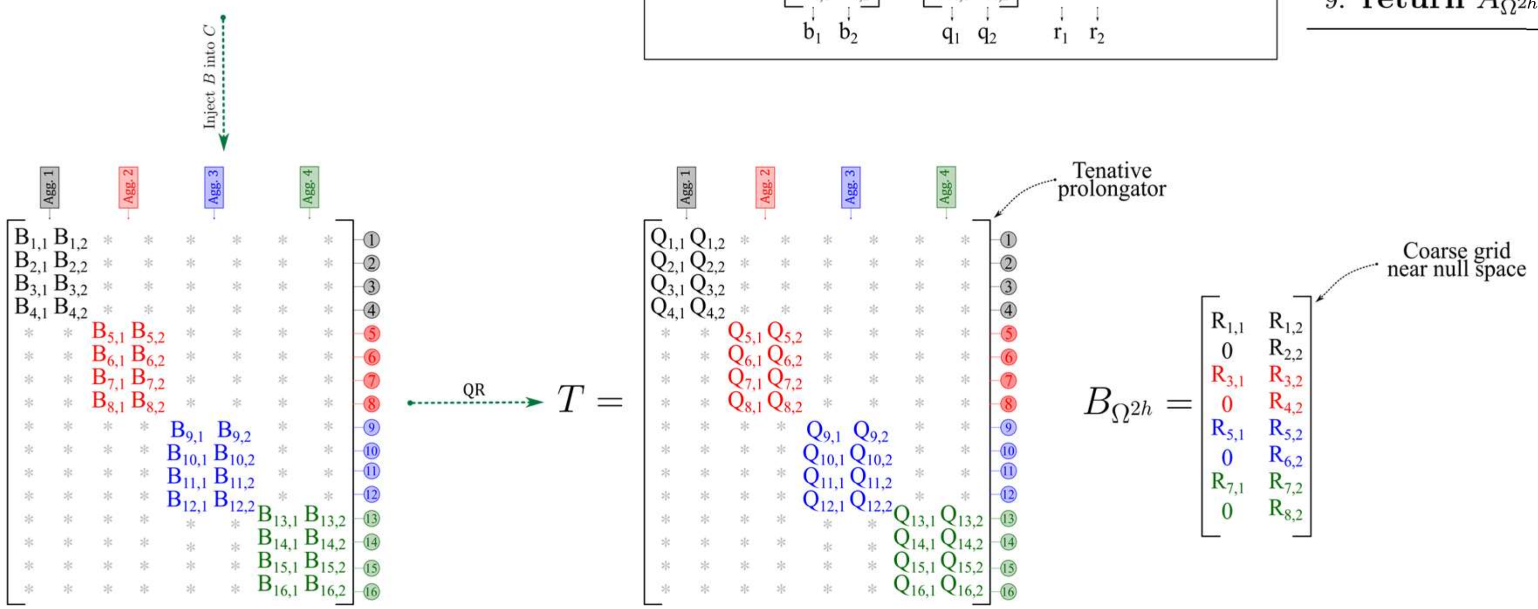
if $B ==$ `nothing` **then**
    $B =$ `ones(size(`$A, 1$`))`
end if
while `size`$(A) <$ `max_coarse` **do**
    $A_{\Omega^{2h}}, B_{\Omega^{2h}},$ `level` $=$ `extend_hierarchy`(A, B)
    $A \leftarrow A_{\Omega^{2h}}, B \leftarrow B_{\Omega^{2h}}$
end while





Algorithm 1 `extend_hierarchy`(A, B)

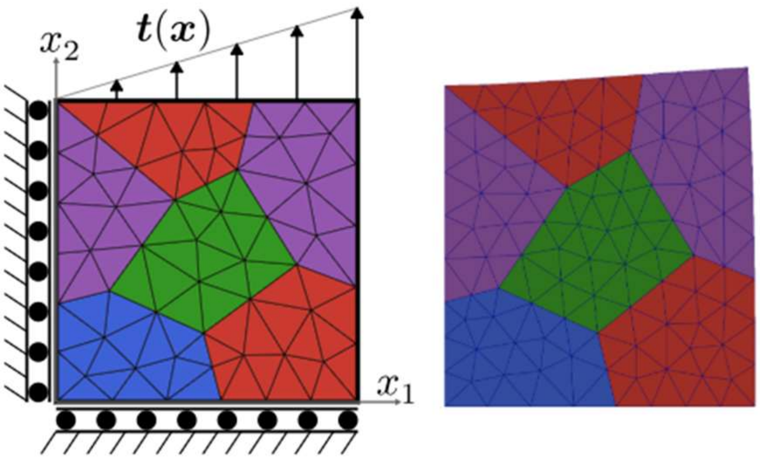
$S =$ `strength`(A)
$C =$ `aggregate`(S)
$b =$ `zeros(size(`$A, 1$`), size(`$B, 2$`))`
$B \leftarrow$ `improve_candidates`(A, B, b)
$T, B_{\Omega^{2h}} =$ `fit_candidates`(C, B)
$P =$ `smooth`(A, T)
$R = P^T$
$A_{\Omega^{2h}} = RAP$
return $A_{\Omega^{2h}}, B_{\Omega^{2h}},$ `Level`(A, P, R)



Numerical Experiments

2D Linear Elasticity:

- NNS (B) represents the rigid body modes:
  - Translation in the $x-$ direction
  - Translation in the $y-$ direction
  - Rotation about $z-$ axis



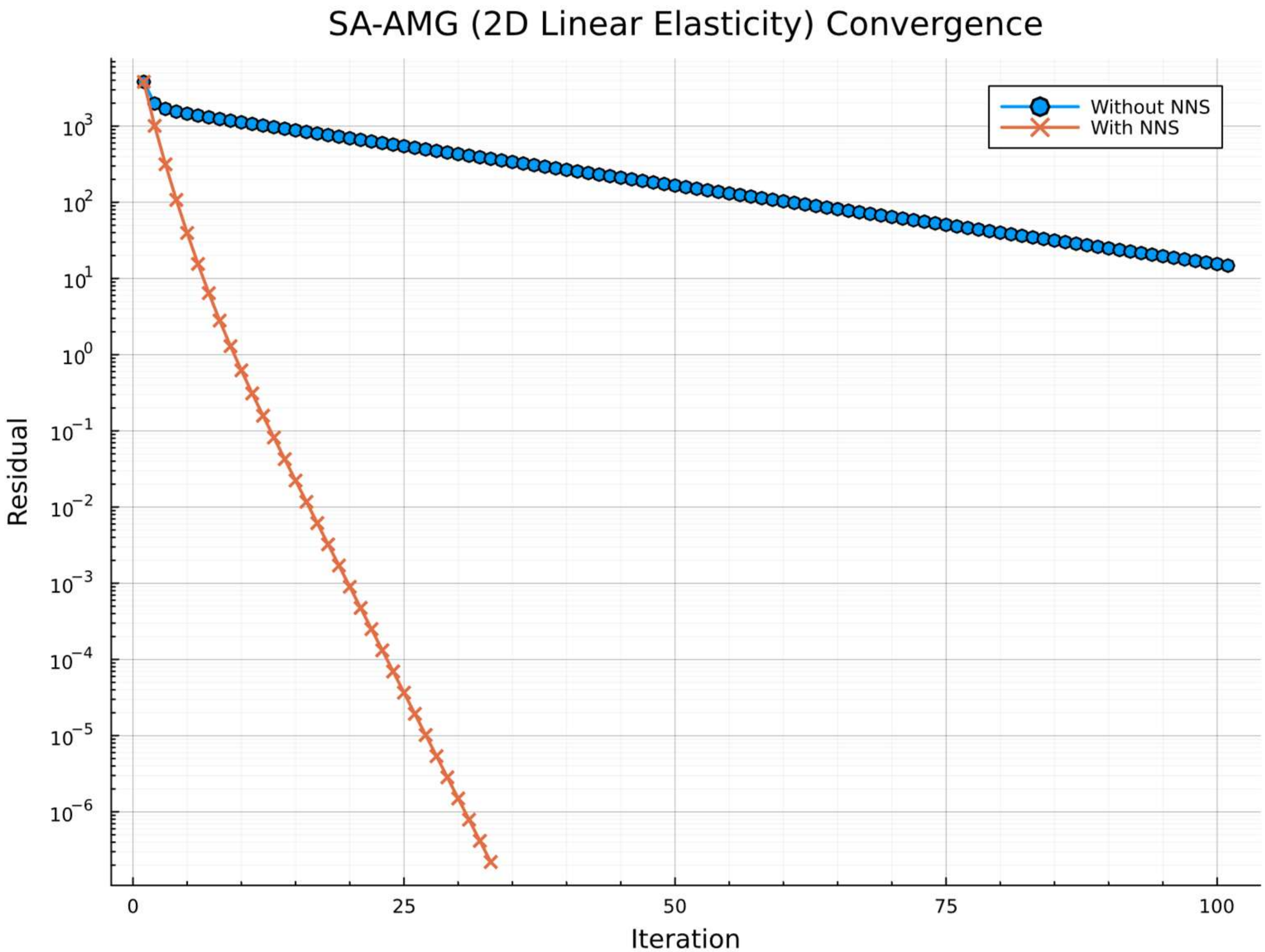
$$B = \begin{bmatrix} 1 & 0 & -y_1 \\ 0 & 1 & x_1 \\ 1 & 0 & -y_2 \\ 0 & 1 & x_2 \\ \vdots & \vdots & \vdots \\ 1 & 0 & -y_n \\ 0 & 1 & x_n \end{bmatrix} \in \mathbb{R}^{2nx3}$$

```
using AlgebraicMultigrid
x_nns, residuals_nns = solve(A, b, SmoothedAggregationAMG(), log=true, reltol=1e-10;B=B)
x_wonns, residuals_wonns = solve(A, b, SmoothedAggregationAMG(), log=true, reltol=1e-10)
```


RUHR UNIVERSITÄT BOCHUM RUB

Numerical Experiments

2D Linear Elasticity:

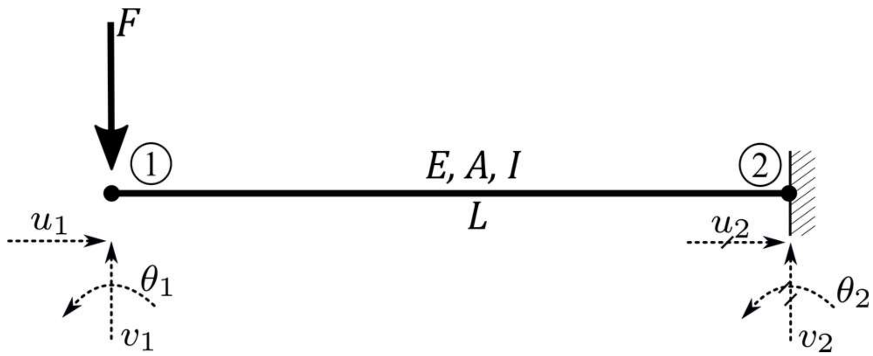


RUHR UNIVERSITÄT BOCHUM RUB

Numerical Experiments

Cantilever Beam:

- NNS (B) represents the rigid body modes:
  - Translation in the $x-$ direction
  - Translation in the $y-$ direction
  - Rotation about $z-$ axis



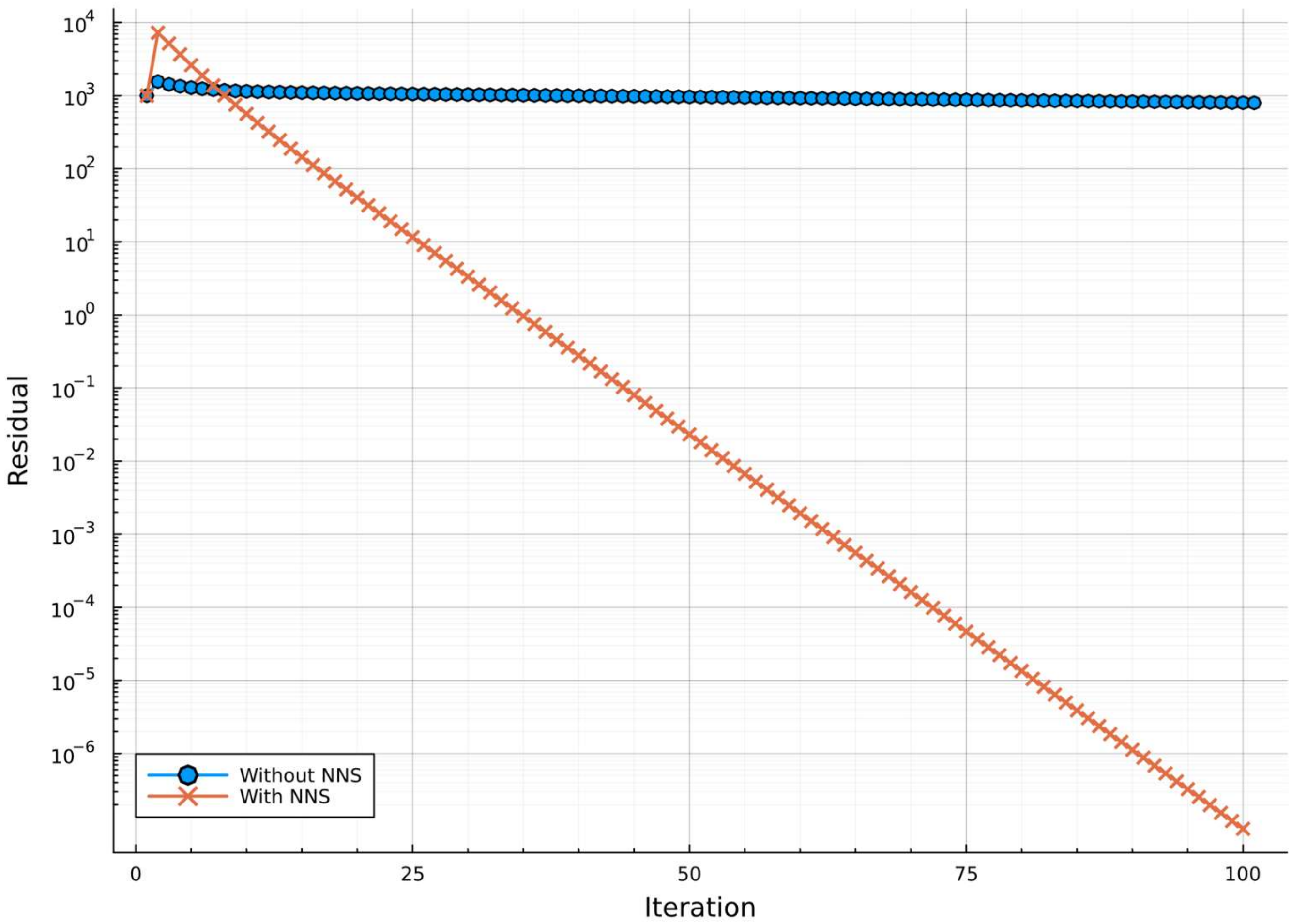
$$B = \begin{bmatrix} 1 & 0 & -y_1 \\ 0 & 1 & x_1 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 0 & -y_n \\ 0 & 1 & x_n \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3nx3}$$

```
using AlgebraicMultigrid
x_nns, residuals_nns = solve(A, b, SmoothedAggregationAMG(), log=true, reltol=1e-10;B=B)
x_wonns, residuals_wonns = solve(A, b, SmoothedAggregationAMG(), log=true, reltol=1e-10)
```


RUHR UNIVERSITÄT BOCHUM RUB

Numerical Experiments

Cantilever Beam



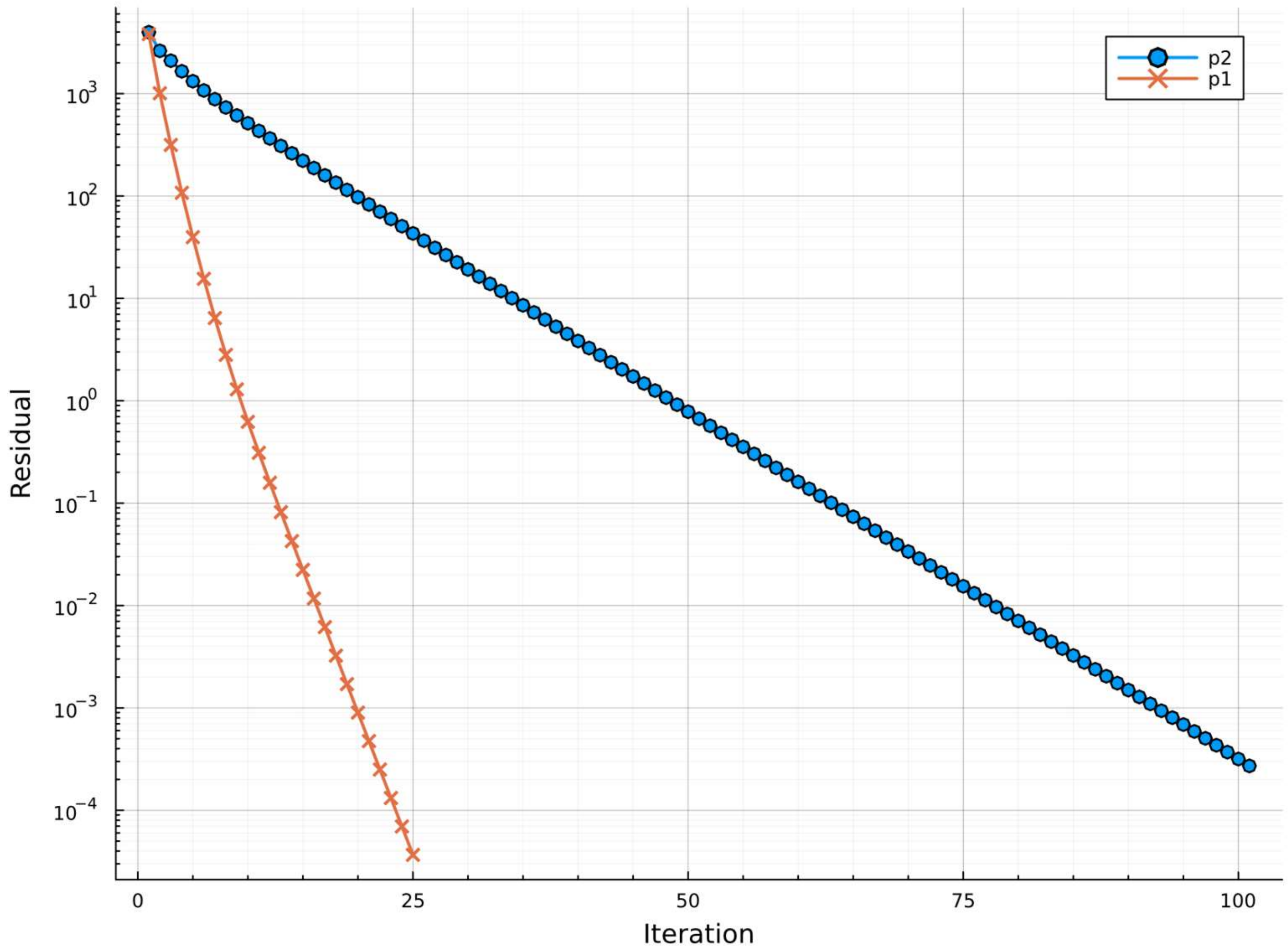
RUHR UNIVERSITÄT BOCHUM

Roadmap

- **Multigrid Methods 101**
  - Basic Iterative Methods as Smoothers
  - Error Behaviors Across Grids
  - The Two-Level Method
- **Smoothed aggregation (SA) in Algebraic Multigrid (AMG)**
  - Fixed Near Null Space (NNS)
  - User Defined Near Null Space (NNS)
  - Numerical Experiments
- **FerriteMultigrid.jl: P- Multigrid Extension**
  - Why and How?
  - Coarsening Strategies
  - Package Interface
  - Numerical Experiments
- **Future Work and Possible Extensions**

Why and How?

Why FerriteMultigrid.jl?

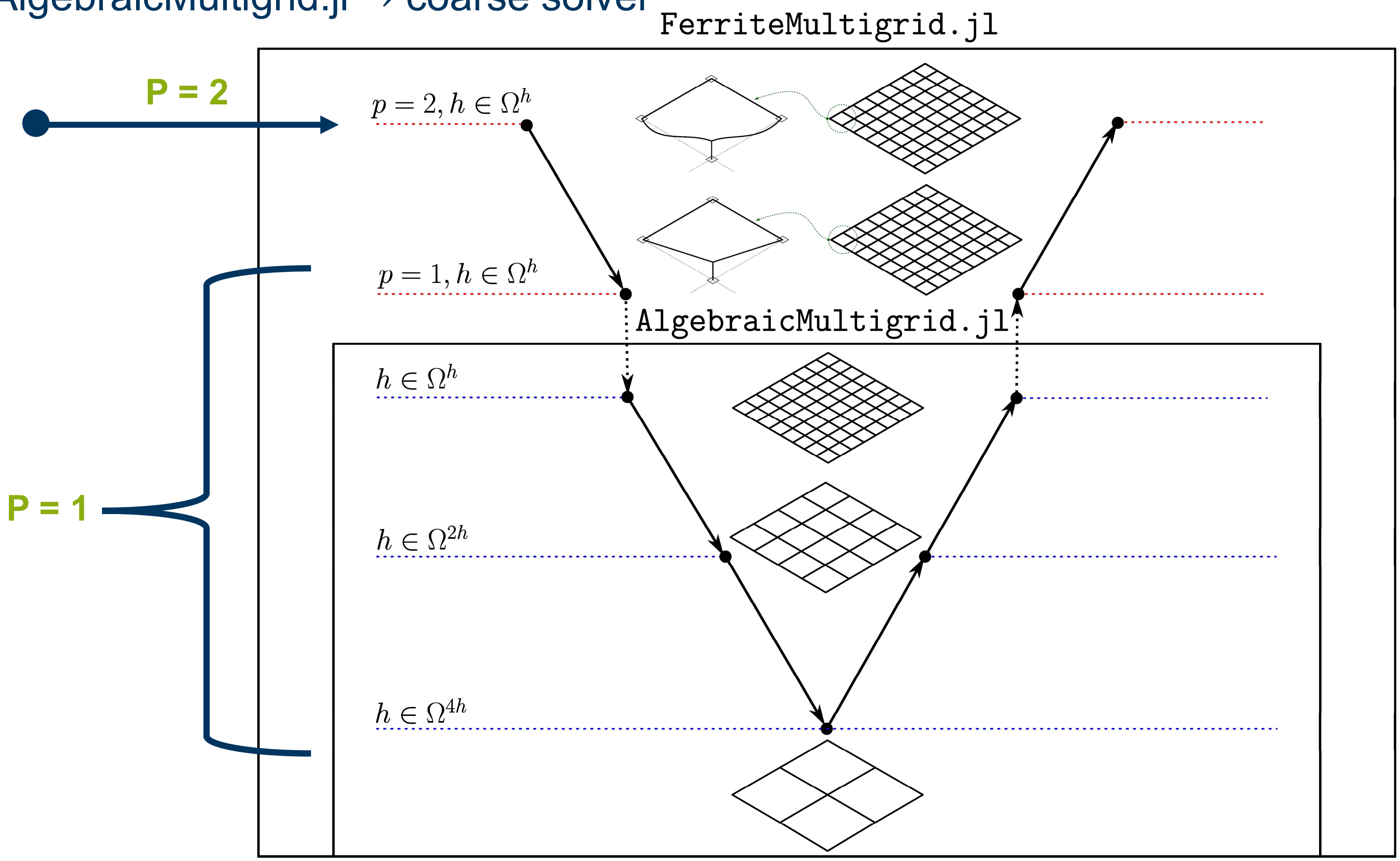


RUHR UNIVERSITÄT BOCHUM RUB

Why and How?

How FerriteMultigrid.jl?

- Ferrite.jl → FEM infrastructure
- AlgebraicMultigrid.jl → coarse solver



Coarsening Strategies

1. Galerkin Projection

$$A_{h,p-1} = \mathcal{I}_{p}^{p-1} A_{h,p} \mathcal{I}_{p-1}^{p},$$

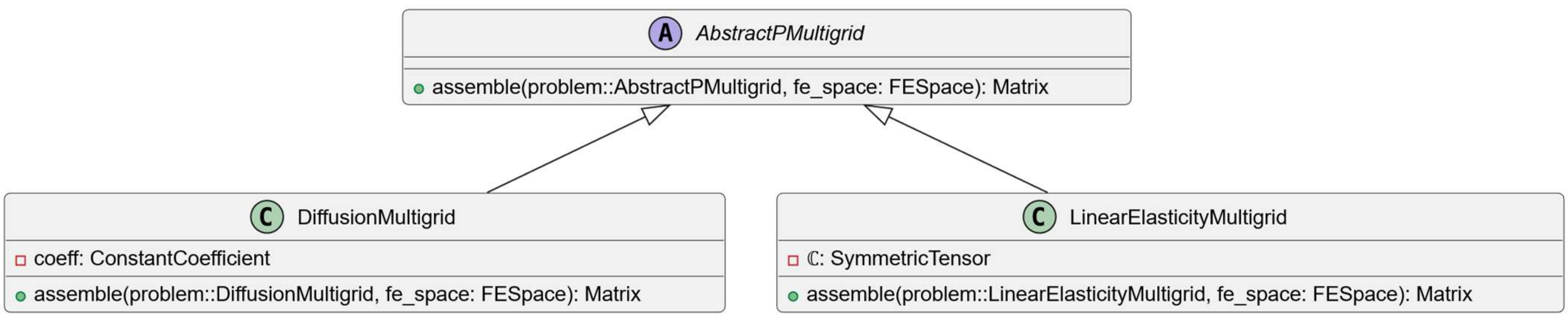
$$\mathcal{I}_{p-1}^{p}(\mathbf{v}_{p-1}) = (\mathbf{M}_p)^{-1} \mathbf{P}_{p-1}^{p}\, \mathbf{v}_{p-1},$$

$$(\mathbf{M}_p)_{ij} := \int_{\Omega} \Phi_{i,p}\, \Phi_{j,p}\, \mathrm{d}\Omega, \qquad (\mathbf{P}_{p-1}^{p})_{ij} := \int_{\Omega} \Phi_{i,p}\, \Phi_{j,p-1}\, \mathrm{d}\Omega.$$

NOTE:

- I_{p-1}^{p}: prolongation operator
- I_{p}^{p-1}: restriction operator
- $\boldsymbol{M}_p$: mass matrix on fine grid
- $\boldsymbol{P}_{p-1}^{p}$: projection from coarse to fine grid
- Φ_p: shape functions at poly. p

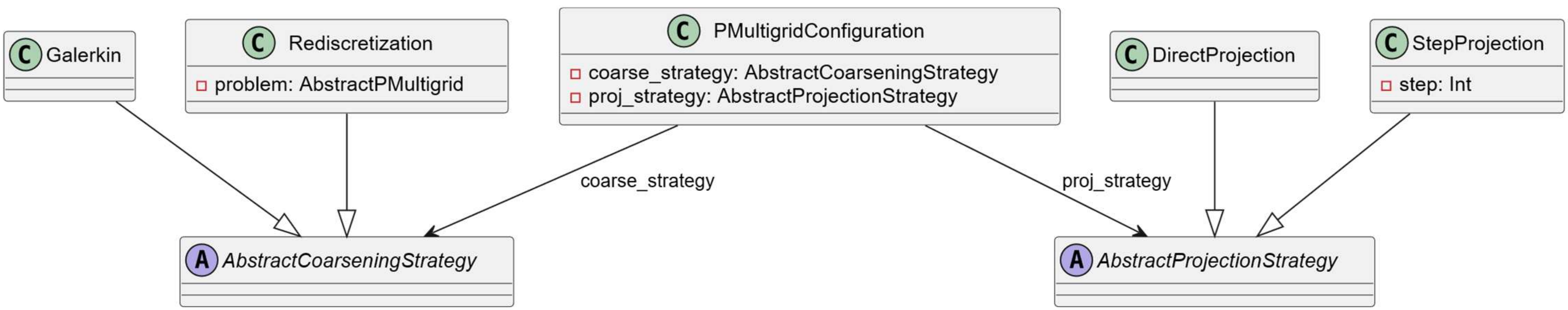
2. Rediscretization



RUHR UNIVERSITÄT BOCHUM RUB

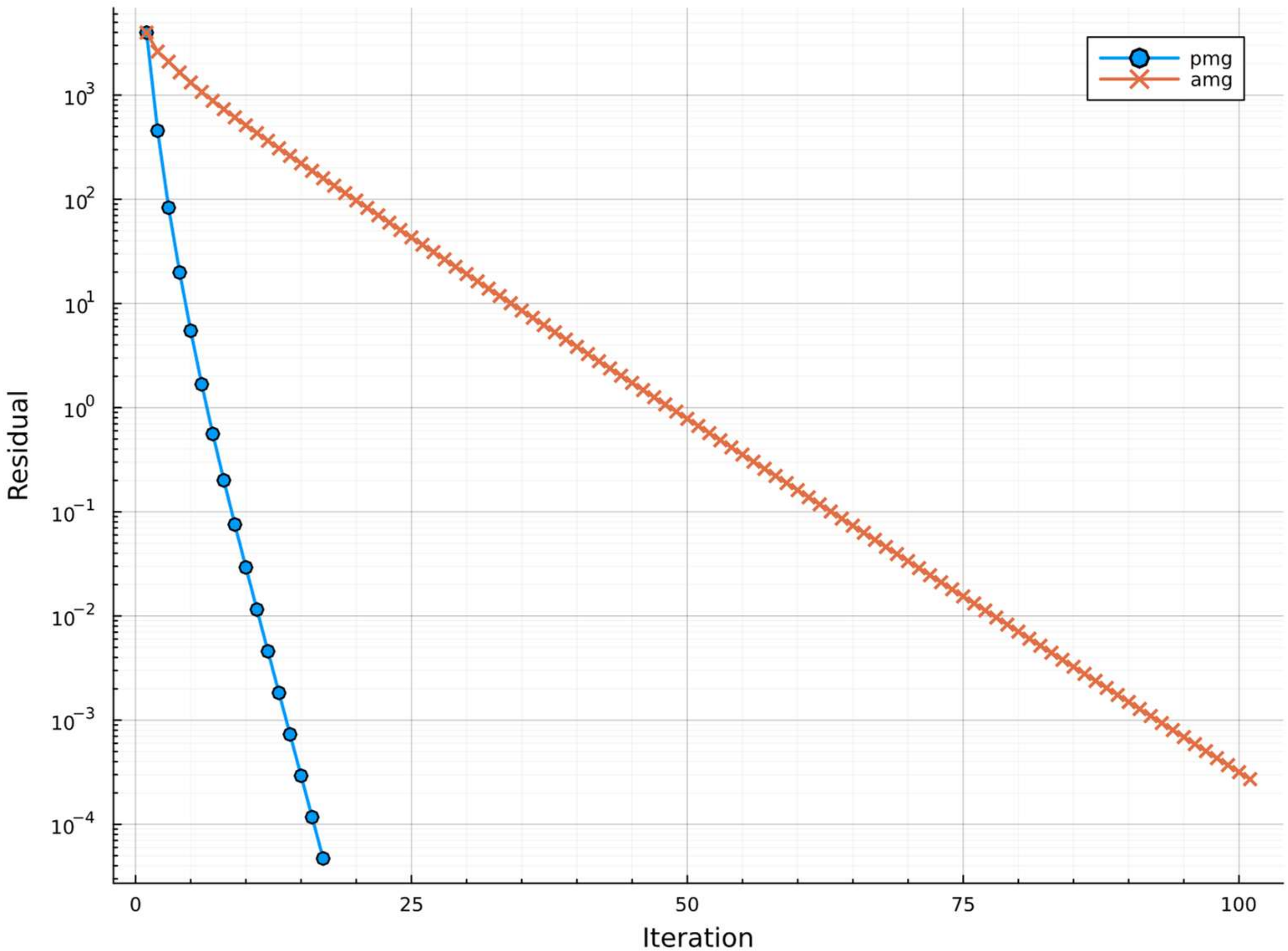
Package Interface

```julia
using FerriteMultigrid

# Define a 1D diffusion problem with p = 2 and 3 quadrature points.
K, f, fe_space = poisson(1000, 2, 3)

# Define a p-multigrid configuration
config = pmultigrid_config() # default config (galerkin as coarsening strategy and direct projection (i.e., from p to 1 directly))

# Solve using the p-multigrid solver
x, res = solve(K, f, fe_space, config; log = true, rtol = 1e-10)
```



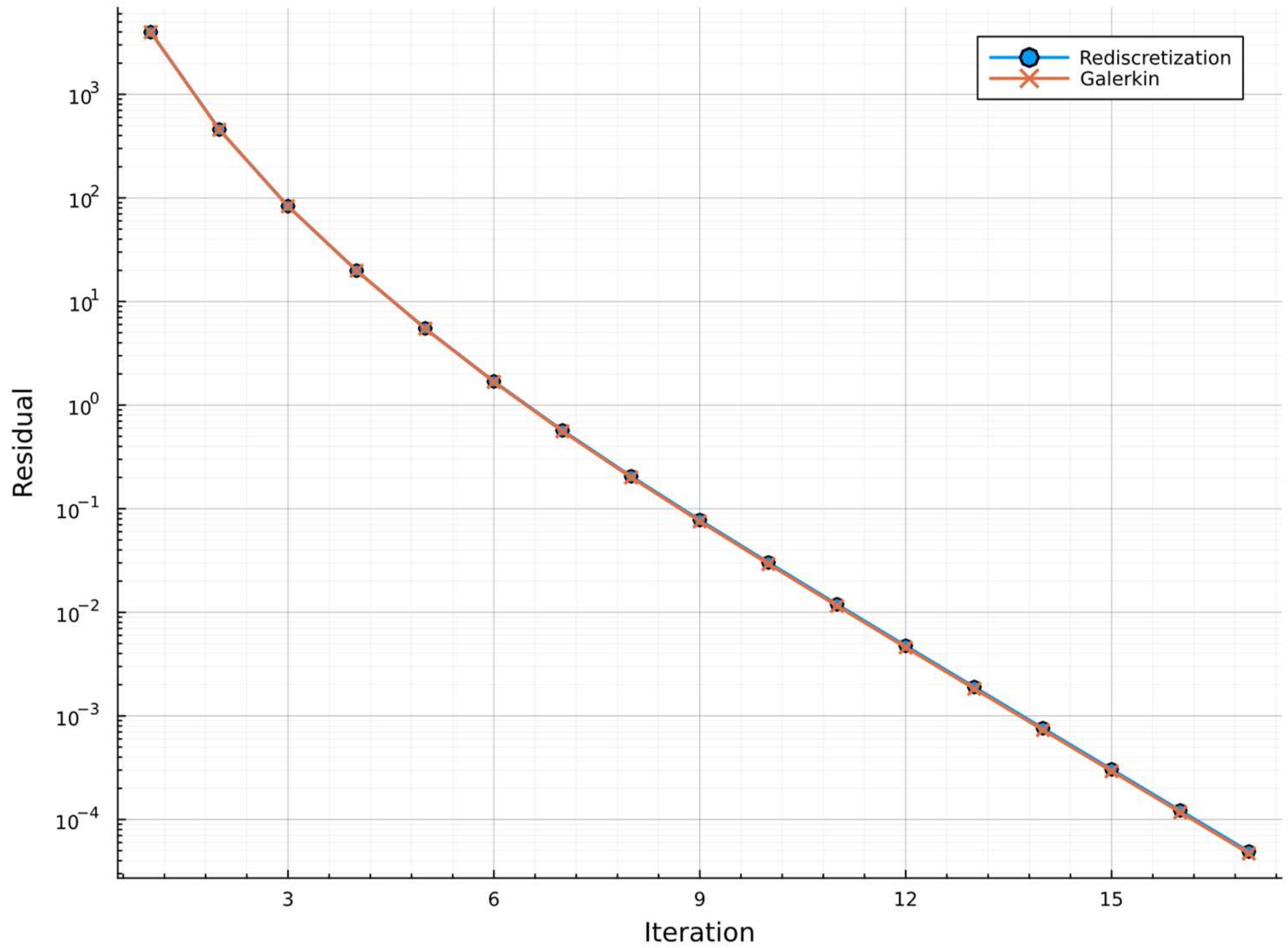
Results

SA-AMG vs P-Multigrid (2D Linear Elasticity – Quadratic element approximation)



Results

Galerkin vs Rediscretization (2D Linear Elasticity)



Future Work and Possible Extensions

- **Isogeometric analysis**
  - Theoretical foundation: https://www.sciencedirect.com/science/article/pii/S0045782520305326?via%3Dihub
- **Learning-based AMG coarsening**
  - Theoretical foundation: https://openreview.net/pdf?id=xXYjxli-2i
  - GitHub issue: https://github.com/JuliaLinearAlgebra/AlgebraicMultigrid.jl/issues/84
- **Support for classical AMG with NNS**
  - Theoretical foundation: https://onlinelibrary.wiley.com/doi/10.1002/nla.688
  - GitHub issue: https://github.com/JuliaLinearAlgebra/AlgebraicMultigrid.jl/issues/80


END OF PRESENTATION
THANK YOU FOR LISTENING
makeameme.org