universitätbonn

Multigrid Solvers for Complicated Geometries

Lars Ole Schwen

Institute for Numerical Simulation
University of Bonn

joint work with:
Martin Rumpf, University of Bonn
Tobias Preusser, CeVis, University of Bremen
Stefan Sauter, University of Zurich

Bremen, 2007–11–15

Outline

Finite Element Calculations

universitätbonn

Physical problem:

- real object
- physical process

Finite Element Calculations

universitätbonn

Physical problem:

- real object
- physical process

FE Simulation:

1. representation of object, PDE for physical process
2. geometric discretization: mesh (grid points and connectivity)
3. basis functions for discretization of continuous quantity
4. PDE leads to system of equations
5. solve
6. interpret and visualize result

Finite Element Calculations

universitätbonn

Physical problem:

- real object
- physical process

FE Simulation:

1. representation of object, PDE for physical process
2. geometric discretization: mesh (grid points and connectivity)
3. basis functions for discretization of continuous quantity
4. PDE leads to system of equations
5. solve
6. interpret and visualize result

1. Objects / Physical Processes

Representation of objects:

- CAD model
- discrete image / volume dataset (CT, MRT, . . .)
- other levelset description

1. Objects / Physical Processes

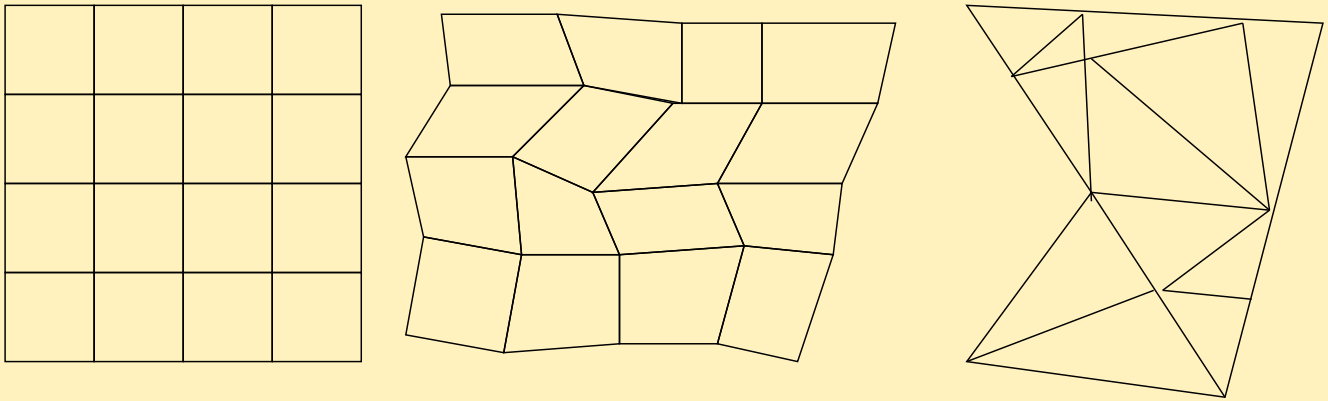
Representation of objects:

- CAD model
- discrete image / volume dataset (CT, MRT, . . .)
- other levelset description

Physical process:

- process typically described by conservation principle
- can derive PDE

2. Grids



Different types of grids:

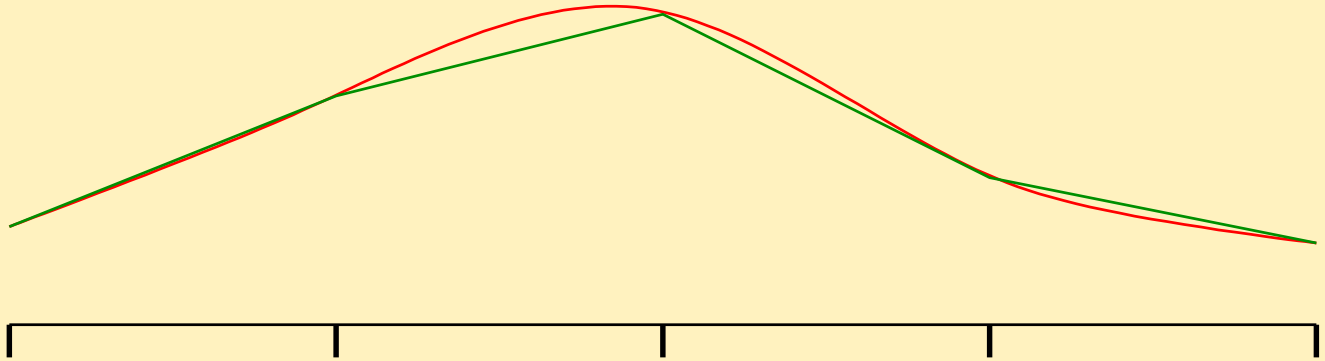
- uniform
- structured
- unstructured

3. Discretization (1D)

Continuous function

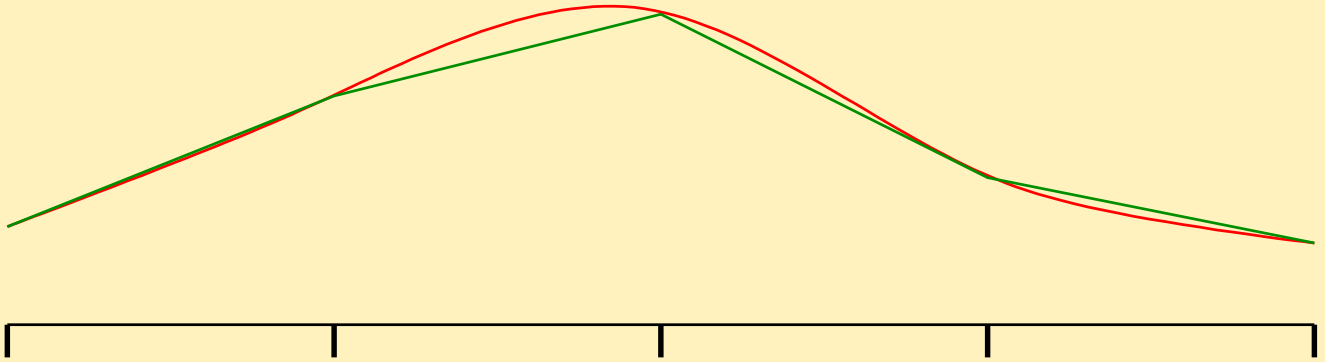
3. Discretization (1D)

Continuous function



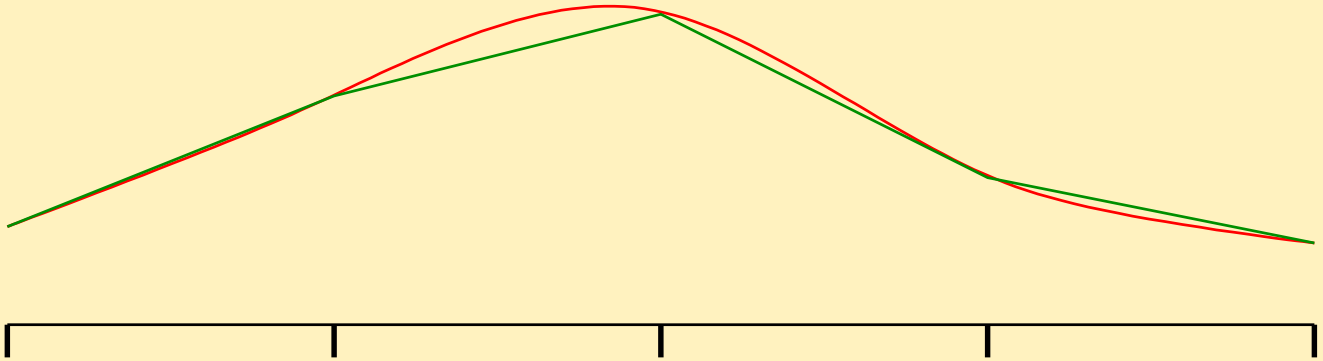
Piecewise linear interpolation on a discrete grid

3. Discretization (1D)

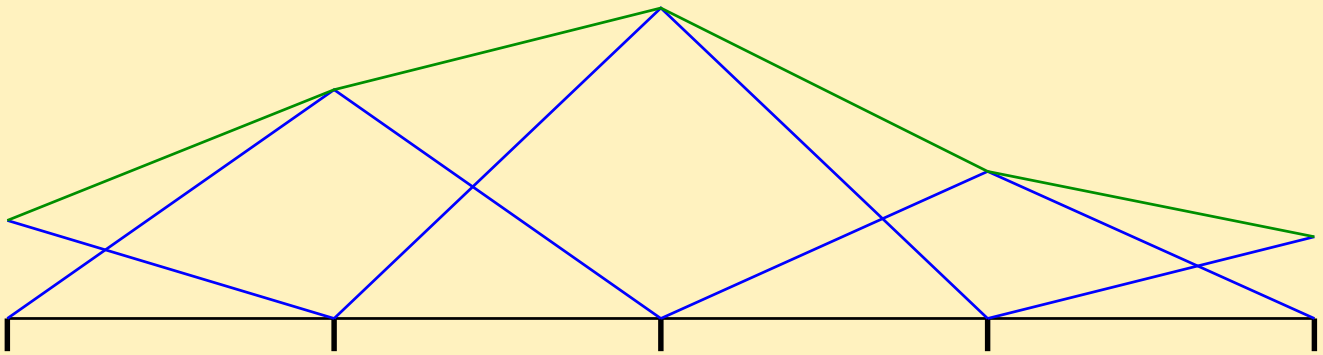


Piecewise linear interpolation on a discrete grid

3. Discretization (1D)

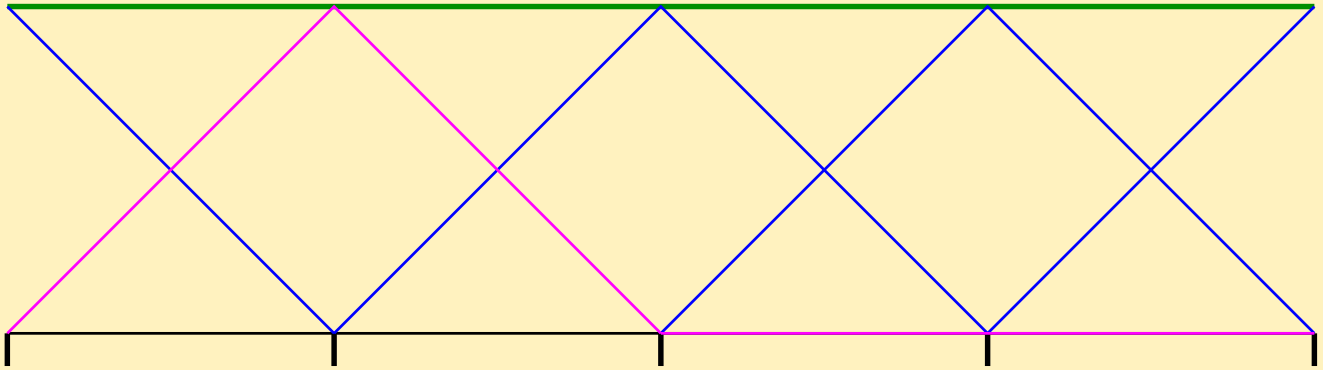


Piecewise linear interpolation on a discrete grid



Discrete interpolation as sum of “hat” functions

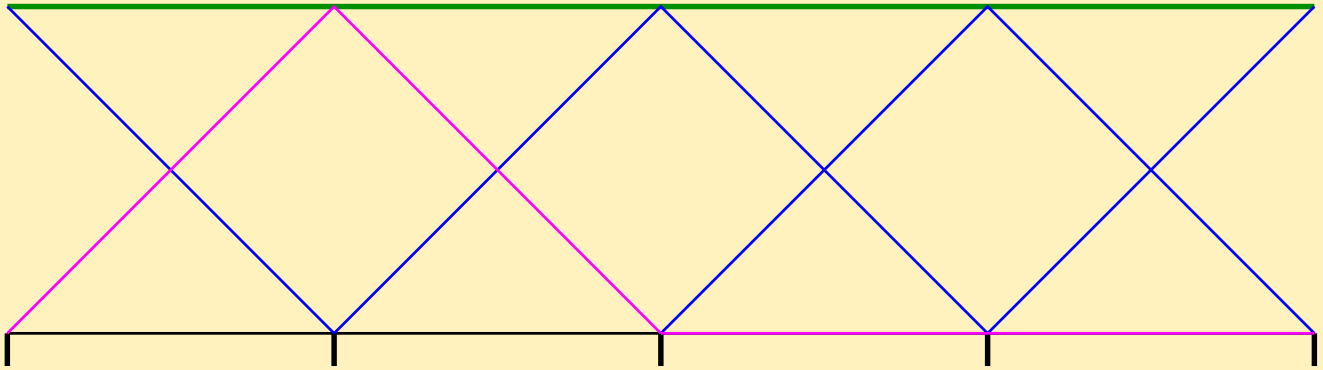
3. Discretization (1D)



The basis of “hat” functions:

- piecewise linear
- nodal
- partition of unity

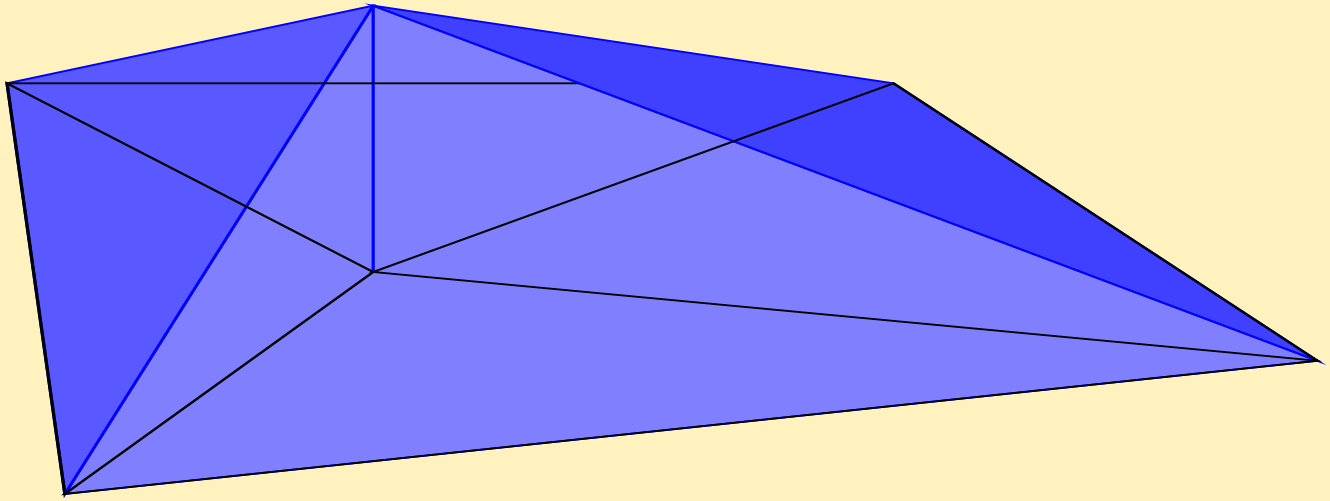
3. Discretization (1D)



The basis of “hat” functions:

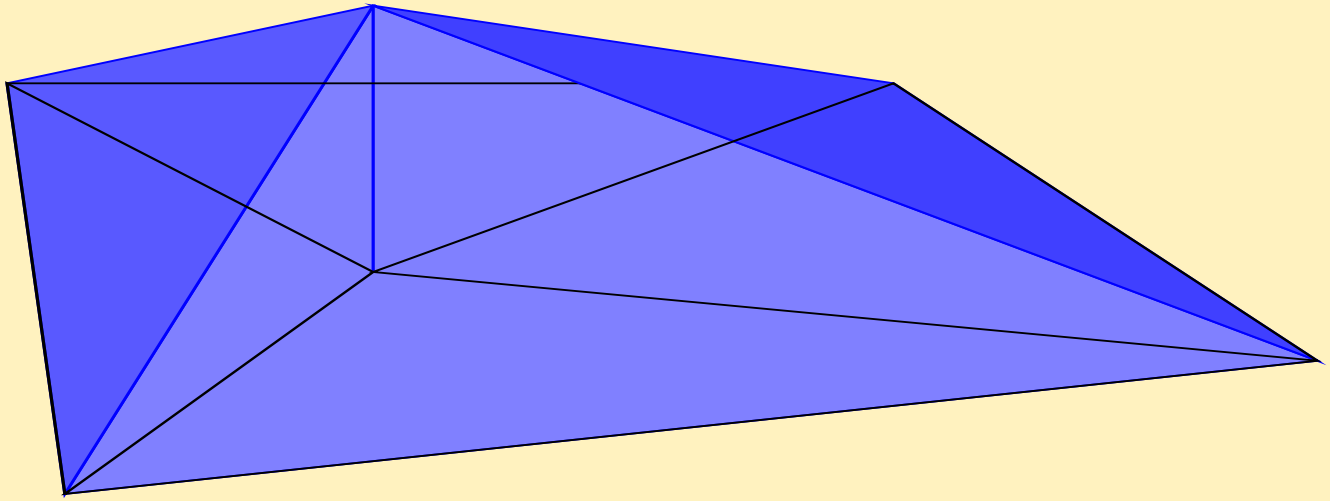
- piecewise linear
- nodal
- partition of unity
- interpolation of continuous function given by coefficient vector

3. Discretization (2D)



For triangular grids, “piecewise linear” can be generalized to 2D.

3. Discretization (2D)



For triangular grids, “piecewise linear” can be generalized to 2D.

For quadrilateral grids, it becomes “piecewise multilinear”.

4. From PDE to Linear Equations

- weak / variational form of PDE problem
- “combination” of discrete basis functions
- integrals of (derivatives of) pairs of basis functions

4. From PDE to Linear Equations

- weak / variational form of PDE problem
- “combination” of discrete basis functions
- integrals of (derivatives of) pairs of basis functions

Consequences:

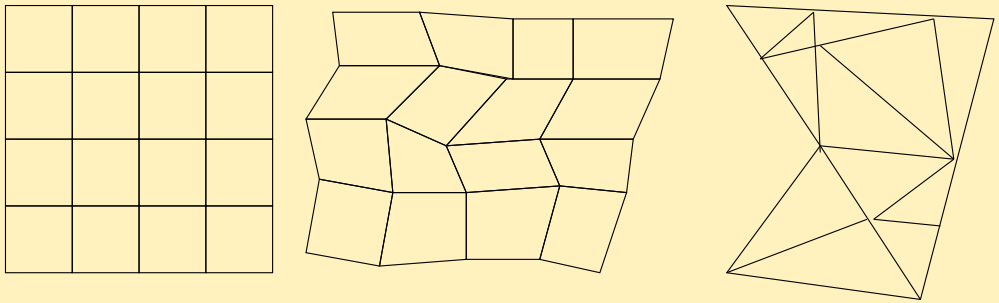
- matrix entry nonzero if supports overlap

4. From PDE to Linear Equations

- weak / variational form of PDE problem
- “combination” of discrete basis functions
- integrals of (derivatives of) pairs of basis functions

Consequences:

- matrix entry nonzero if supports overlap
- matrix is sparse
- sparsity structure reflects connectivity of the grid



Solving a system of equations

$$Ax = b$$

for $A \in \mathbb{R}^{n \times n}$

$$\begin{bmatrix} * & * & \dots & * \\ * & * & & * \\ \vdots & & \ddots & \\ * & * & & * \end{bmatrix} \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \vdots \\ \star \end{bmatrix}$$

Solving a system of equations

$$Ax = b$$

for $A \in \mathbb{R}^{n \times n}$

$$\begin{bmatrix} * & * & \dots & * \\ * & * & & * \\ \vdots & & \ddots & \\ * & * & & * \end{bmatrix} \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \vdots \\ \star \end{bmatrix}$$

Different methods:

- direct
- iterative

Solving a system of equations

$$Ax = b$$

for $A \in \mathbb{R}^{n \times n}$

$$\begin{bmatrix} * & * & \dots & * \\ * & * & & * \\ \vdots & & \ddots & \\ * & * & & * \end{bmatrix} \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \vdots \\ \star \end{bmatrix}$$

Different methods:

- direct
- iterative
- conjugate gradient

Solving a system of equations

universitätbonn

$$Ax = b$$

for $A \in \mathbb{R}^{n \times n}$

$$\begin{bmatrix} * & * & \dots & * \\ * & * & & * \\ \vdots & & \ddots & \\ * & * & & * \end{bmatrix} \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \vdots \\ \star \end{bmatrix}$$

Different methods:

- direct
- iterative
- conjugate gradient
- multigrid

Gaußian Elimination

Simple direct method to compute the solution.

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \end{bmatrix} \rightarrow \begin{bmatrix} 1 & * & * \\ & * & * \\ & * & * \end{bmatrix} \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1 & * & * \\ & 1 & * \\ & & * \end{bmatrix} \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \end{bmatrix} \rightarrow \begin{bmatrix} 1 & * & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \end{bmatrix}$$

Gaußian Elimination

universitätbonn

Properties:

- full matrix storage required, even for sparse matrices: $O(n^2)$ memory

Gaußian Elimination

universitätbonn

Properties:

- full matrix storage required, even for sparse matrices: $O(n^2)$ memory
- $O(n^3)$ algorithmic complexity

Gaußian Elimination

universitätbonn

Properties:

- full matrix storage required, even for sparse matrices: $O(n^2)$ memory
- $O(n^3)$ algorithmic complexity
- propagation of numerical errors

Iterative Methods

$$Ax = b \Leftrightarrow Ax - b = 0$$

Iterative Methods

$$Ax = b \Leftrightarrow Ax - b = 0$$

From some initial guess x^0, compute sequence (x^n).

Iterative Methods

$$Ax = b \Leftrightarrow Ax - b = 0$$

From some initial guess x^0, compute sequence (x^n).

Properties:

- typically no storage of matrix necessary
- can stop if $\|Ax^n - b\|$ "sufficiently small"

Iterative Method: Jacobi

[Jacobi 1844]
Let $D = \text{diag}(A)$ have nonzero entries.

Iterative Method: Jacobi

[Jacobi 1844]
Let $D = \mathrm{diag}(A)$ have nonzero entries.

Jacobi iteration:

$$x^{n+1} = D^{-1}\left(D - A\right)x^{n} + D^{-1}b$$

Iterative Method: Jacobi

[Jacobi 1844]
Let $D = \text{diag}(A)$ have nonzero entries.

Jacobi iteration:

$$x^{n+1} = D^{-1}\left(D - A\right)x^n + D^{-1}b$$

Properties:

- no need to store $A, D, D - A$
- convergence is slow

Iterative Method: Gauß-Seidel

Let

- D be the diagonal part of A with nonzero entries
- L be the strictly lower trianuglar part

Iterative Method: Gauß-Seidel

universitätbonn

Let

- D be the diagonal part of A with nonzero entries
- L be the strictly lower trianuglar part

Gauß-Seidel iteration:

$$x^{n+1} = (D + L)^{-1} (D + L - A) x^n + (D + L)^{-1} b$$

Iterative Method: Gauß-Seidel

universitätbonn

Let

- D be the diagonal part of A with nonzero entries
- L be the strictly lower trianuglar part

Gauß-Seidel iteration:

$$x^{n+1} = (D + L)^{-1} (D + L - A) x^n + (D + L)^{-1} b$$

Properties:

- Jacobi iteration using same memory for x^n and x^{n+1}

Iterative Method: Gauß-Seidel

Let

- D be the diagonal part of A with nonzero entries
- L be the strictly lower trianuglar part

Gauß-Seidel iteration:

$$x^{n+1} = (D + L)^{-1} (D + L - A) x^n + (D + L)^{-1} b$$

Properties:

- Jacobi iteration using same memory for x^n and x^{n+1}
- no need to store $A, L, D + L - A$
- convergence is better than for Jacobi, but still slow

Solving as an Optimization Problem

Let A be symmetric and positive definite (SPD), then

$$Ax = b \Leftrightarrow Ax - b = 0 \Leftrightarrow x \text{ minimizes } \frac{1}{2}Ay \cdot y - b \cdot y$$

Solving as an Optimization Problem

Let A be symmetric and positive definite (SPD), then

$$Ax = b \Leftrightarrow Ax - b = 0 \Leftrightarrow x \text{ minimizes } \frac{1}{2}Ay \cdot y - b \cdot y$$

solving the system is equivalent to a convex minimization problem.

Conjugate Gradient Method

universitätbonn

[Hestenes, Stiefel 1952]
In each step, find optimal descent direction based on old directions.

Conjugate Gradient Method

universitätbonn

[Hestenes, Stiefel 1952]
In each step, find optimal descent direction based on old directions.

Properties:

- direct method (n iterations)
- typically used as iterative method
- convergence depends on condition number $\kappa(A)$

Conjugate Gradient Method

universitätbonn

[Hestenes, Stiefel 1952]
In each step, find optimal descent direction based on old directions.

Properties:

- direct method (n iterations)
- typically used as iterative method
- convergence depends on condition number $\kappa(A)$ and on full spectrum

Condition Number

$$\kappa := \frac{|\text{largest eigenvalue of } A|}{|\text{smallest eigenvalue of } A|}$$

Condition Number

$$\kappa := \frac{|\text{largest eigenvalue of } A|}{|\text{smallest eigenvalue of } A|}$$

Example:

$$\begin{bmatrix} 1 & \\ & 1000 \end{bmatrix} \begin{bmatrix} ? \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \end{bmatrix} \qquad \kappa = 1000$$

Condition Number

$$\kappa := \frac{|\text{largest eigenvalue of } A|}{|\text{smallest eigenvalue of } A|}$$

Example:

$$\begin{bmatrix} 1 & \\ & 1000 \end{bmatrix} \begin{bmatrix} ? \\ ? \end{bmatrix} = \begin{bmatrix} \star \\ \star \end{bmatrix} \qquad \kappa = 1000$$

$$\left\| \begin{bmatrix} 1 & \\ & 1000 \end{bmatrix} \begin{bmatrix} ? \\ ? \end{bmatrix} - \begin{bmatrix} \star \\ \star \end{bmatrix} \right\|$$

Preconditioning

For nonsingular P,

$$Ax = b \Leftrightarrow (P^{-1}A)x = P^{-1}b$$

Preconditioning

For nonsingular P,

$$Ax = b \Leftrightarrow (P^{-1}A)x = P^{-1}b$$

Need

- P “easy to invert”
- $P^{-1}A$ has “small condition number” (for fast CG convergence)

Preconditioning

For nonsingular P,

$$Ax = b \Leftrightarrow (P^{-1}A)x = P^{-1}b$$

Need

- P "easy to invert" $P = \mathbb{1}$
- $P^{-1}A$ has "small condition number" (for fast CG convergence) $P = A^{-1}$

Preconditioning

For nonsingular P,

$$Ax = b \Leftrightarrow (P^{-1}A)x = P^{-1}b$$

Need

- P "easy to invert"
- $P^{-1}A$ has "small condition number" (for fast CG convergence)

Simplest choice: $P = \text{Diag}(A)$, other methods (SSOR, ILU0, ...)

Smoothing by Gauß-Seidel

Consider the Poisson problem (steady state of heat conduction)

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega \end{aligned}$$

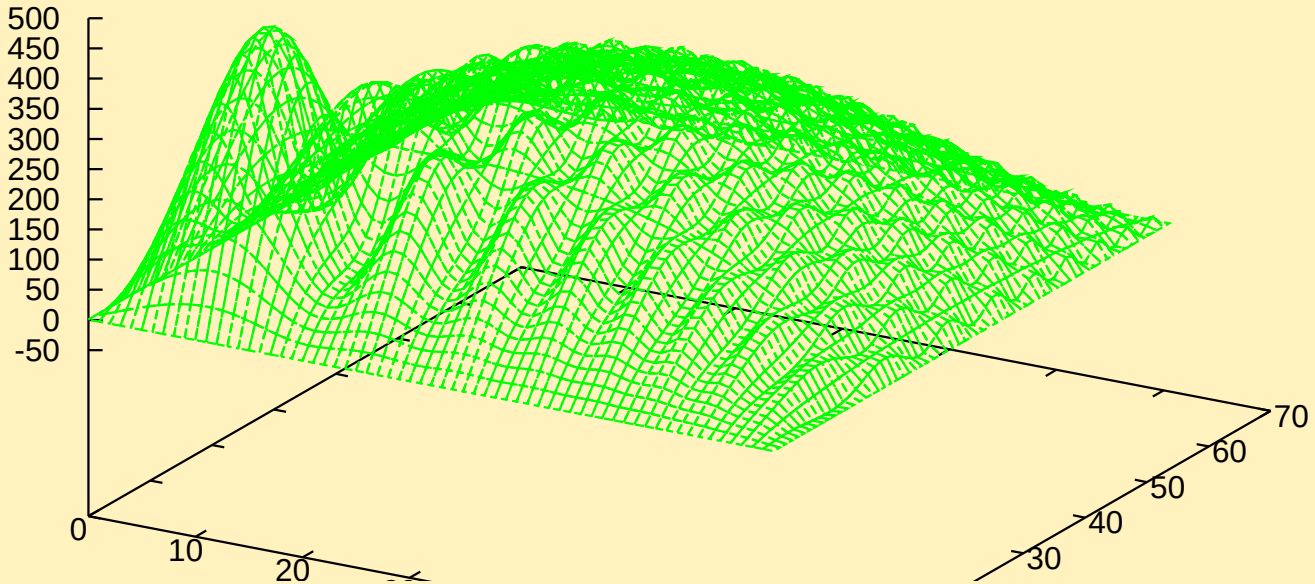
for given source term f.
Again leading to system

$$Ax = b$$

Smoothing by Gauß-Seidel

universitätbonn

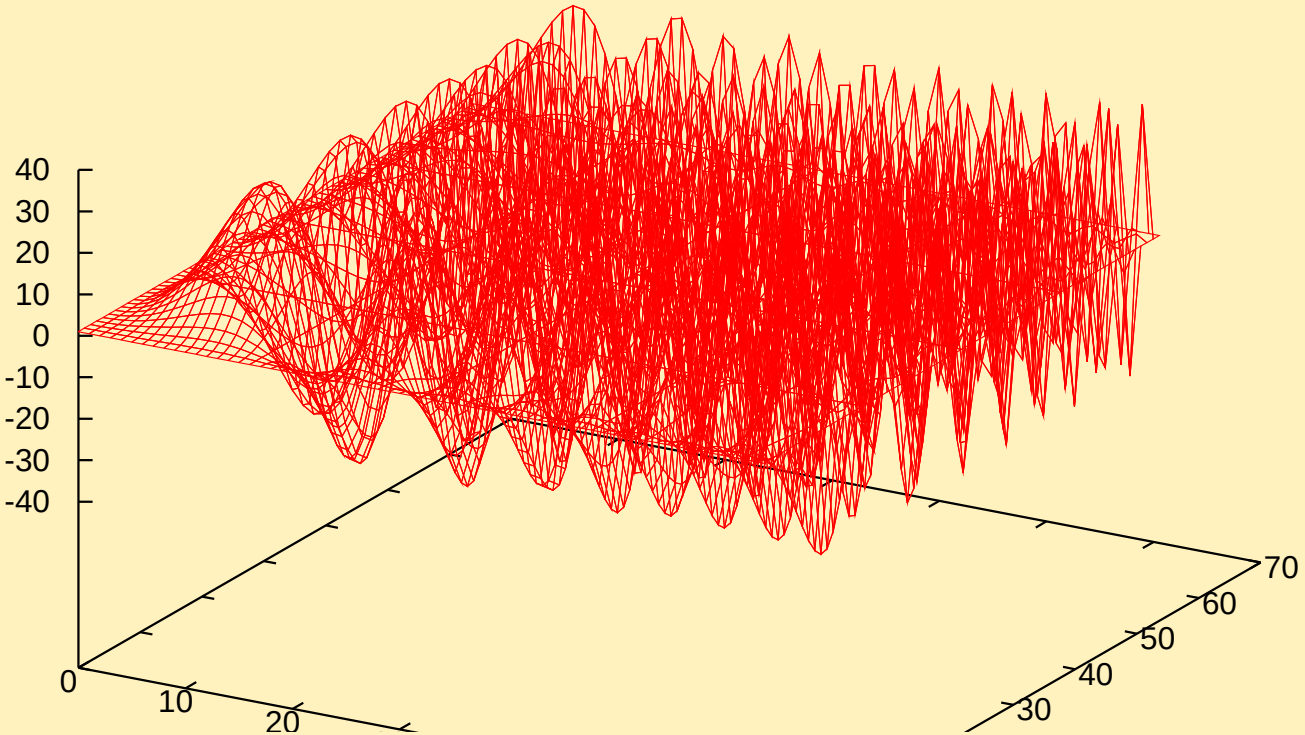
Known solution:

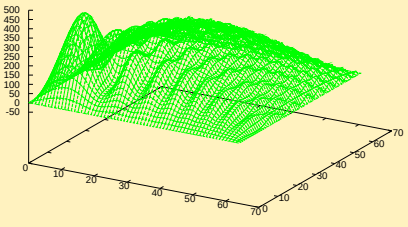


Smoothing by Gauß-Seidel

universitätbonn

Right hand side:

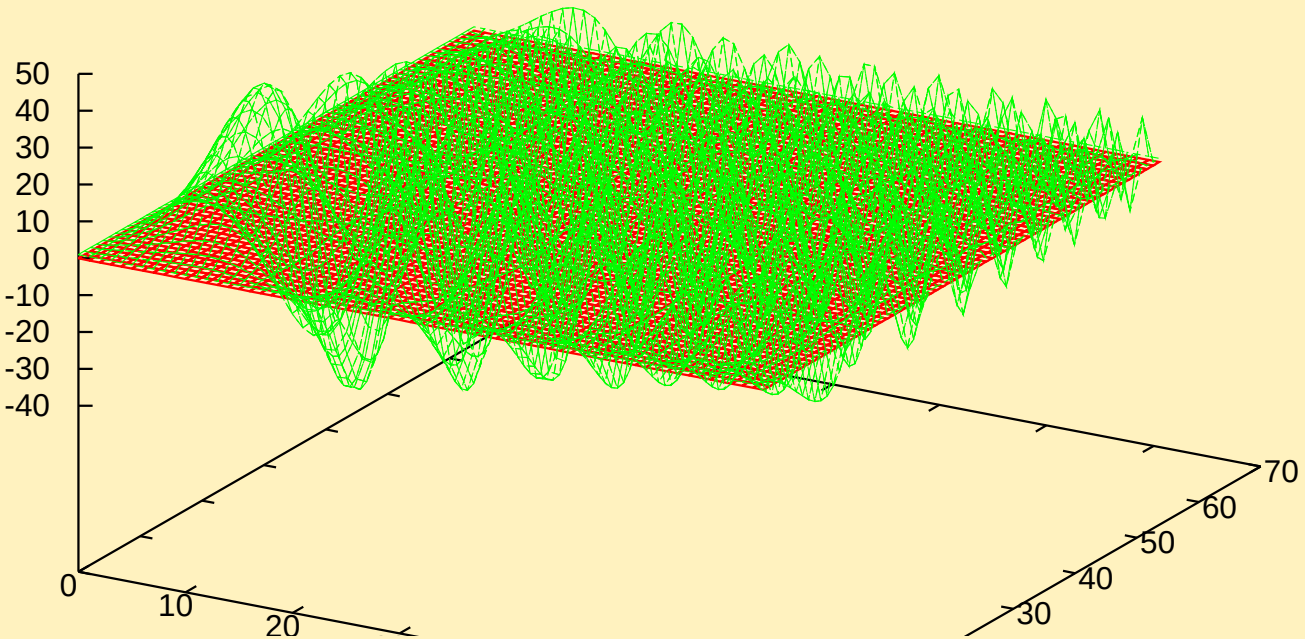


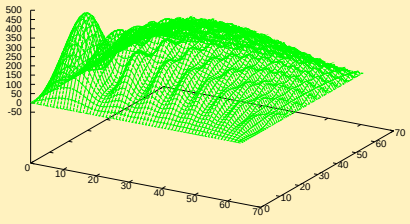


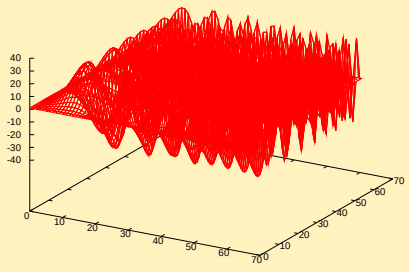
Smoothing by Gauß-Seidel

universitätbonn

Initial guess (red) and “solution” after two Gauß-Seidel iterations:

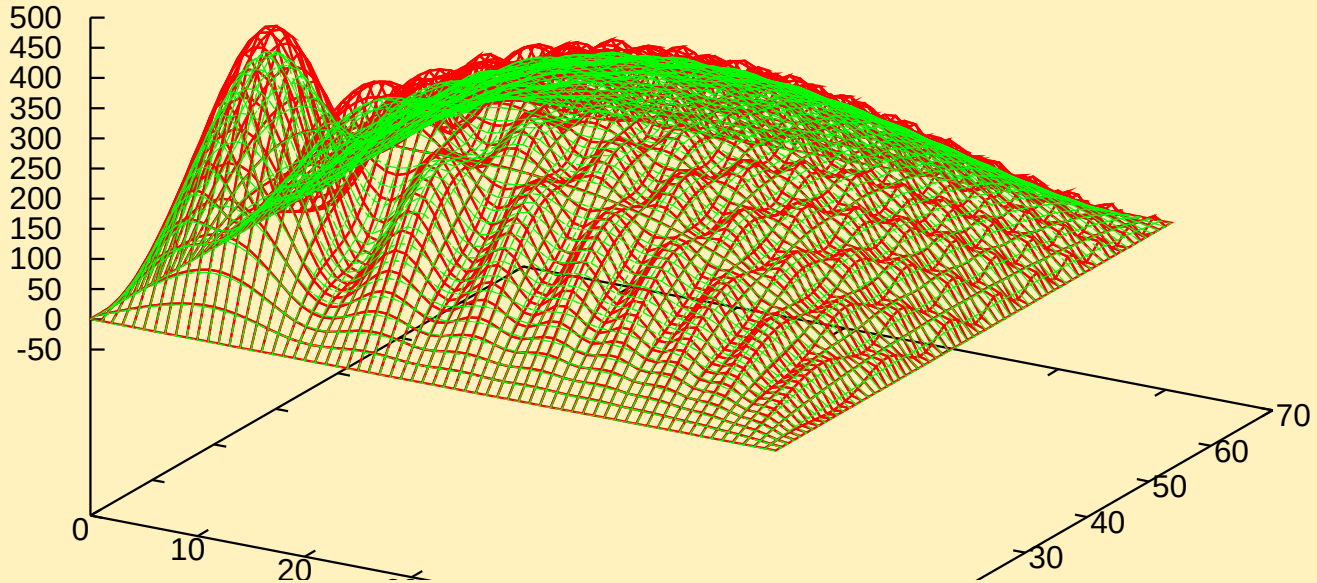


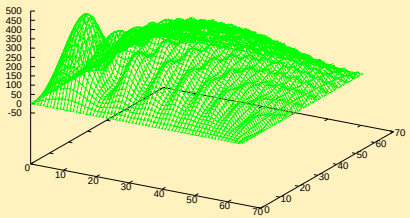


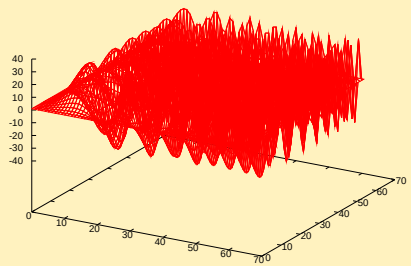


Smoothing by Gauß-Seidel

Initial error (red) and error after two Gauß-Seidel iterations:



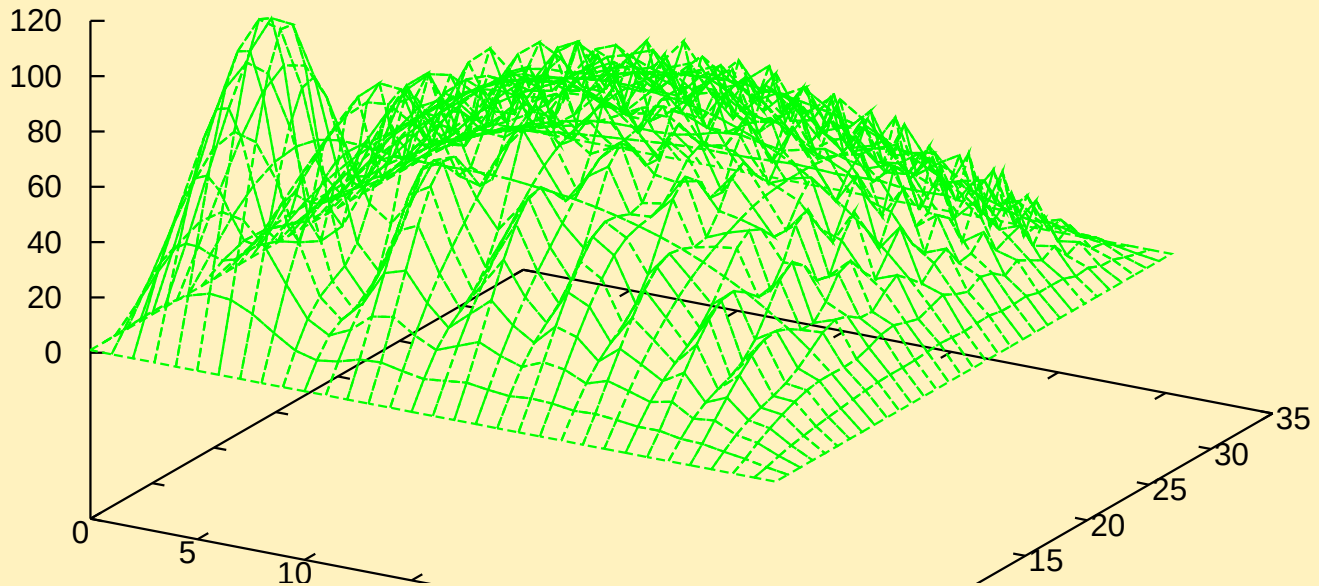




Smoothing by Gauß-Seidel

universitätbonn

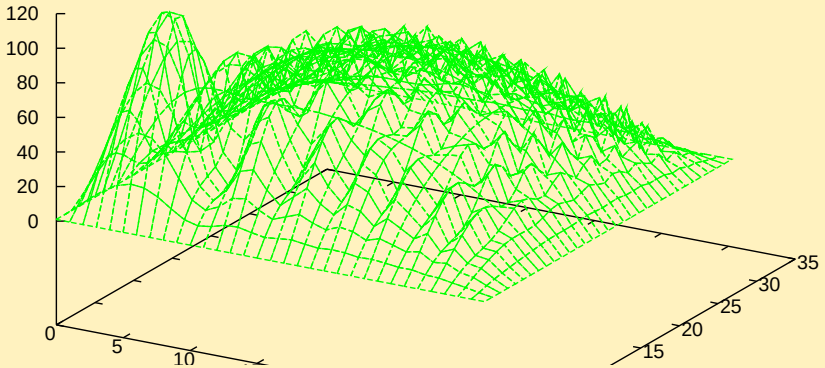
Known solution on coarser grid:

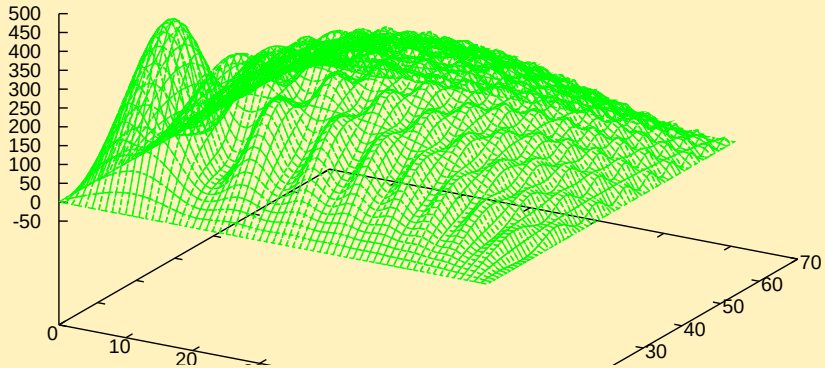


Smoothing by Gauß-Seidel

universitätbonn

Compare coarse and fine grid:

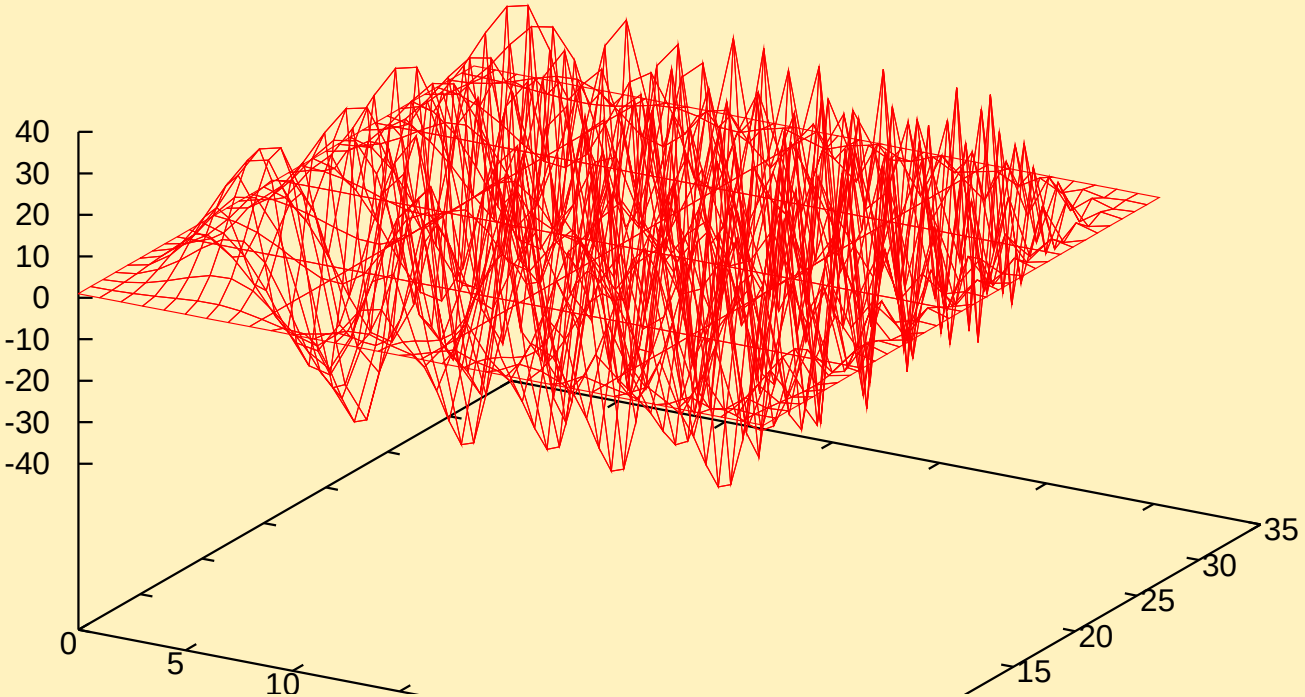


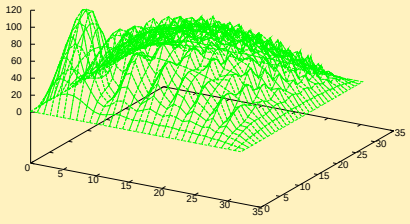


Smoothing by Gauß-Seidel

universitätbonn

Right hand side:

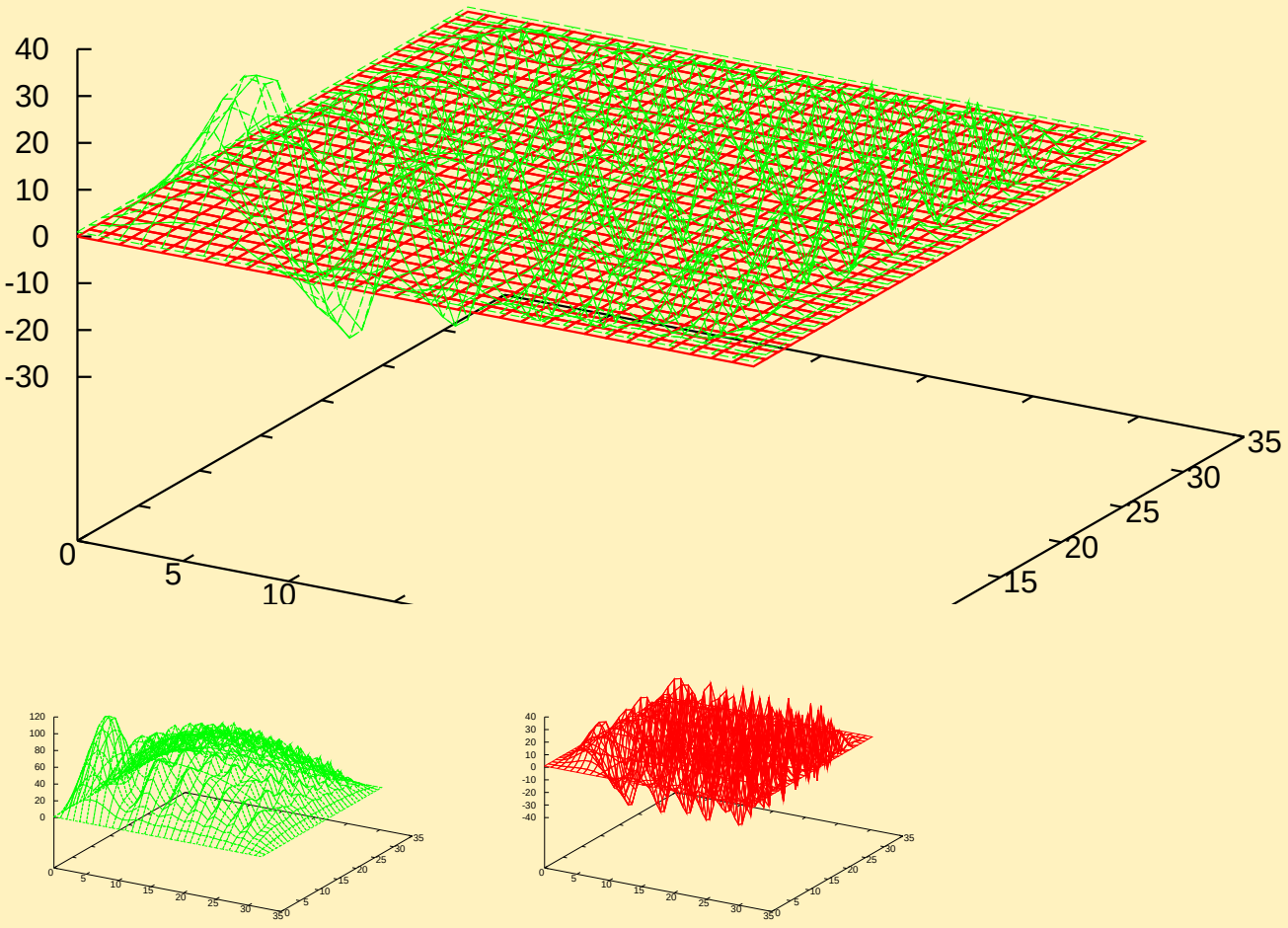




Smoothing by Gauß-Seidel

universitätbonn

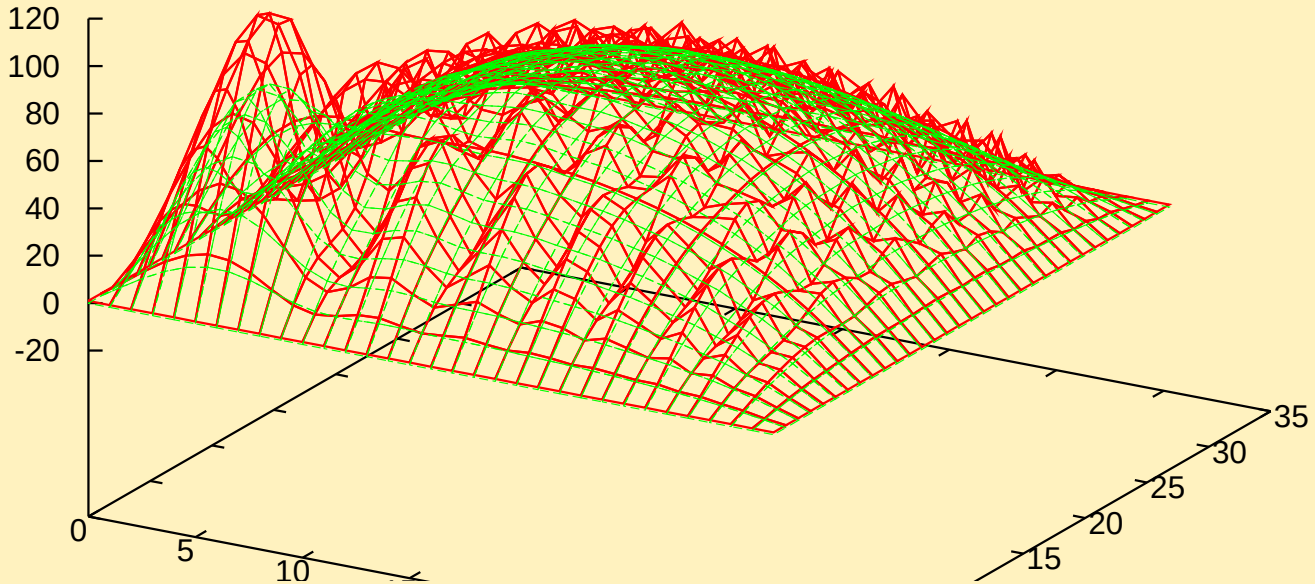
Initial guess (red) and “solution” after two Gauß-Seidel iterations:

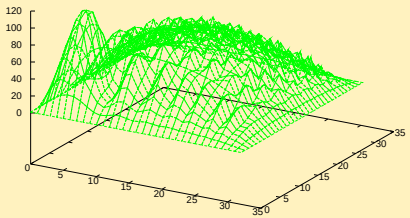


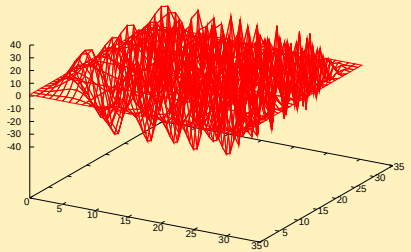
Smoothing by Gauß-Seidel

universitätbonn

Initial error (red) and error after two Gauß-Seidel iterations:







Smoothing by Gauß-Seidel

Observations:

- Gauß-Seidel iterations quickly remove high-frequency components of the error
- low-frequency components are not damped quickly

Smoothing by Gauß-Seidel

Observations:

- Gauß-Seidel iterations quickly remove high-frequency components of the error
- low-frequency components are not damped quickly
- frequency is relative to resolution

Smoothing by Gauß-Seidel

Observations:

- Gauß-Seidel iterations quickly remove high-frequency components of the error
- low-frequency components are not damped quickly
- frequency is relative to resolution

Conclusion:

- Treat different frequencies at different resolutions!

Basic Two-Grid Scheme

universitätbonn

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$

Basic Two-Grid Scheme

universitätbonn

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$

Basic Two-Grid Scheme

universitätbonn

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$
4. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 

Basic Two-Grid Scheme

universitätbonn

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$
4. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 
5. **Prolongate** the result: compute fine e
6. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$

Basic Two-Grid Scheme

universitätbonn

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$
4. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 
5. **Prolongate** the result: compute fine e
6. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$
7. **Postsmooth**: Perform ν_2 Gauß-Seidel iterations 

Basic Two-Grid Scheme

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$
4. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 
5. **Prolongate** the result: compute fine e
6. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$
7. **Postsmooth**: Perform ν_2 Gauß-Seidel iterations 

This requires

Restriction $\mathcal{R}$ (weighted averageing): $\tilde{r} = \mathcal{R}r$

Prolongation $\mathcal{P}$ (interpolation): $e = \mathcal{P}\tilde{e}$

Basic Two-Grid Scheme

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$
4. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 
5. **Prolongate** the result: compute fine e
6. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$
7. **Postsmooth**: Perform ν_2 Gauß-Seidel iterations 

This requires

Restriction $\mathcal{R}$ (weighted averageing): $\tilde{r} = \mathcal{R}r$

Prolongation $\mathcal{P}$ (interpolation): $e = \mathcal{P}\tilde{e}$,
satisfying $\mathcal{P} = \mathcal{R}^T$

Basic Two-Grid Scheme

universitätbonn

[Bakhalov; Fedorenko; Brandt 1977]

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$
4. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 
5. **Prolongate** the result: compute fine e
6. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$
7. **Postsmooth**: Perform ν_2 Gauß-Seidel iterations 

This requires

Restriction $\mathcal{R}$ (weighted averageing): $\tilde{r} = \mathcal{R}r$

Prolongation $\mathcal{P}$ (interpolation): $e = \mathcal{P}\tilde{e}$,
satisfying $\mathcal{P} = \mathcal{R}^T$

System Coarsening $\tilde{A} = \mathcal{R}A\mathcal{P}$

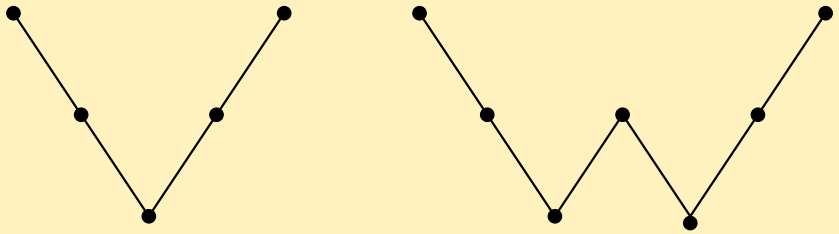
Basic Two-Grid Scheme

universitätbonn

[Bakhalov; Fedorenko; Brandt 1977]

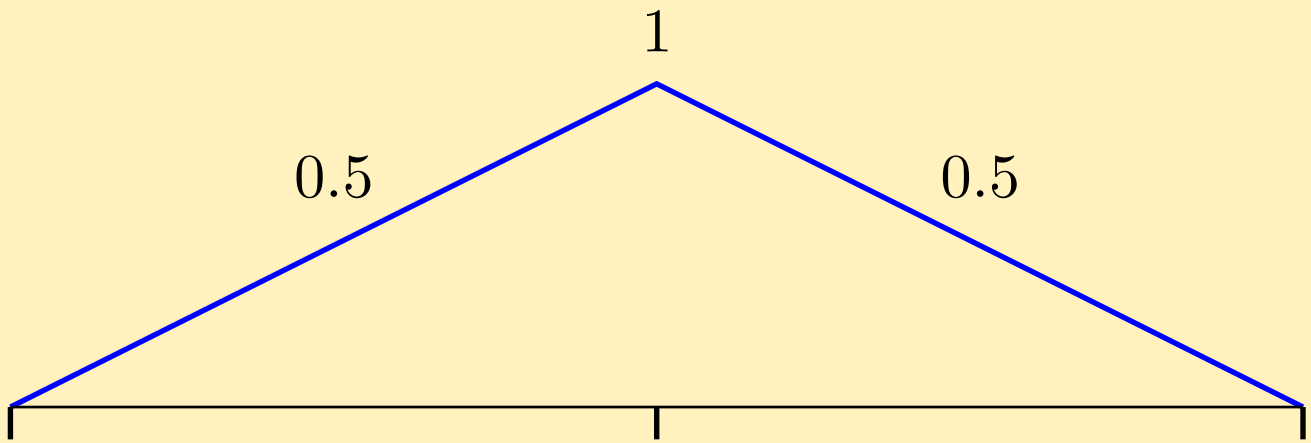
1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$
2. Compute residual: $r = b - Ax^{(1)}$.
3. **Restrict** the residual: compute coarse version $\tilde{r}$
4. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$
5. **Prolongate** the result: compute fine e
6. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$
7. **Postsmooth**: Perform ν_2 Gauß-Seidel iterations

Solve can recursively call this procecdure: Multigrid.
V-cycle vs. W-cycle:



Prolongation

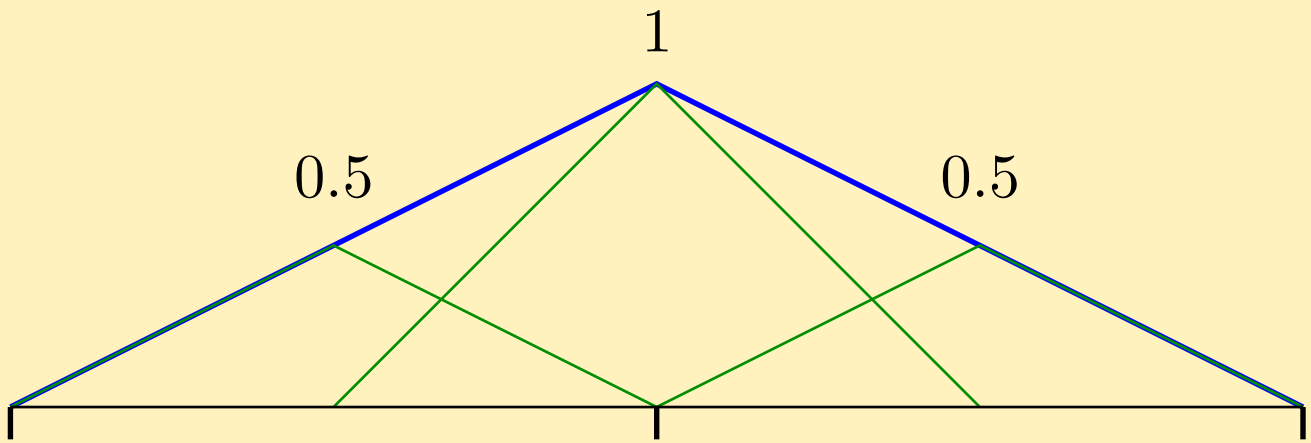
Evaluate coarse grid basis functions at fine grid points.



Can be interpreted as interpolation based on basis functions.

Prolongation

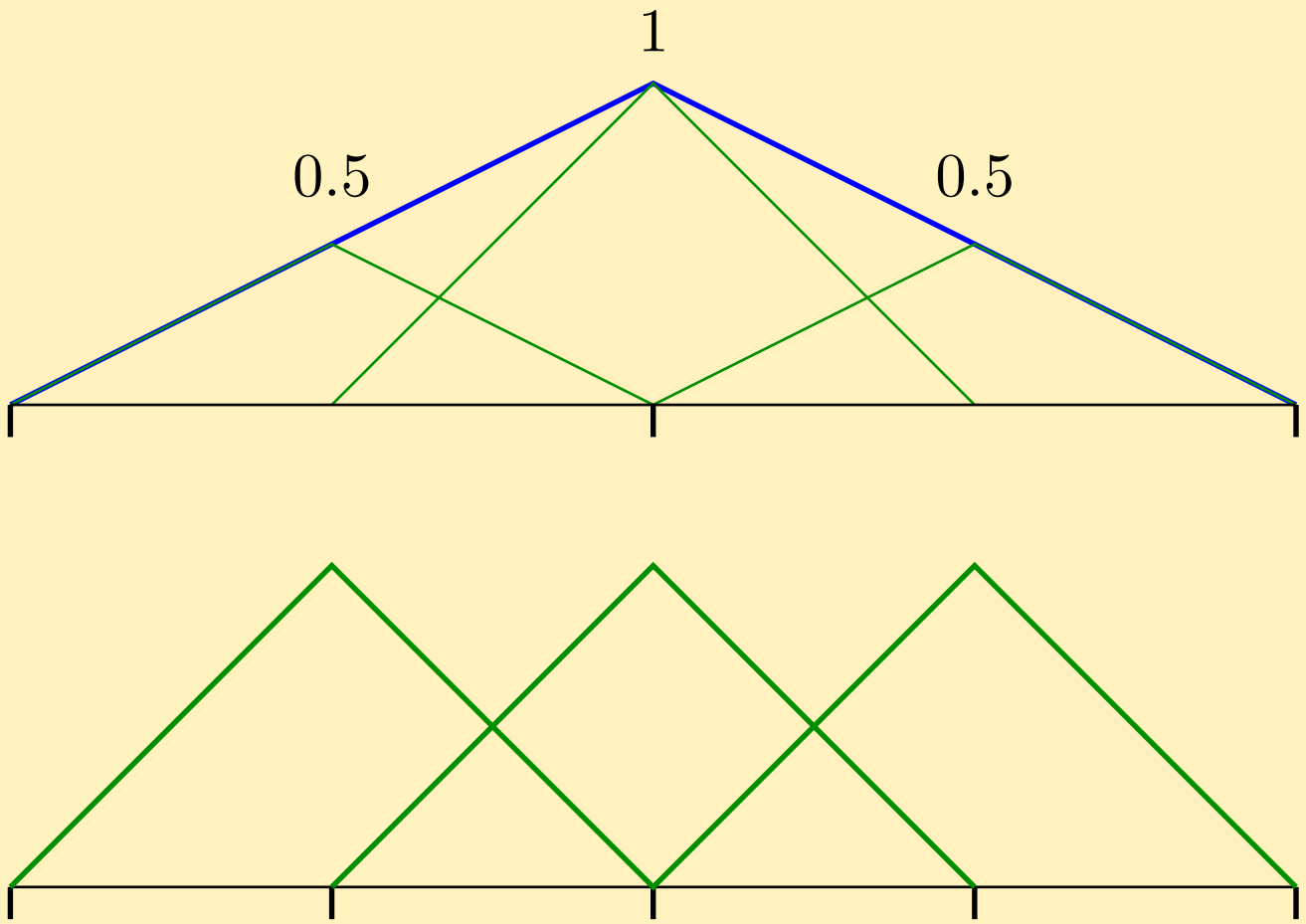
Evaluate coarse grid basis functions at fine grid points.



Can be interpreted as interpolation based on basis functions.

Prolongation

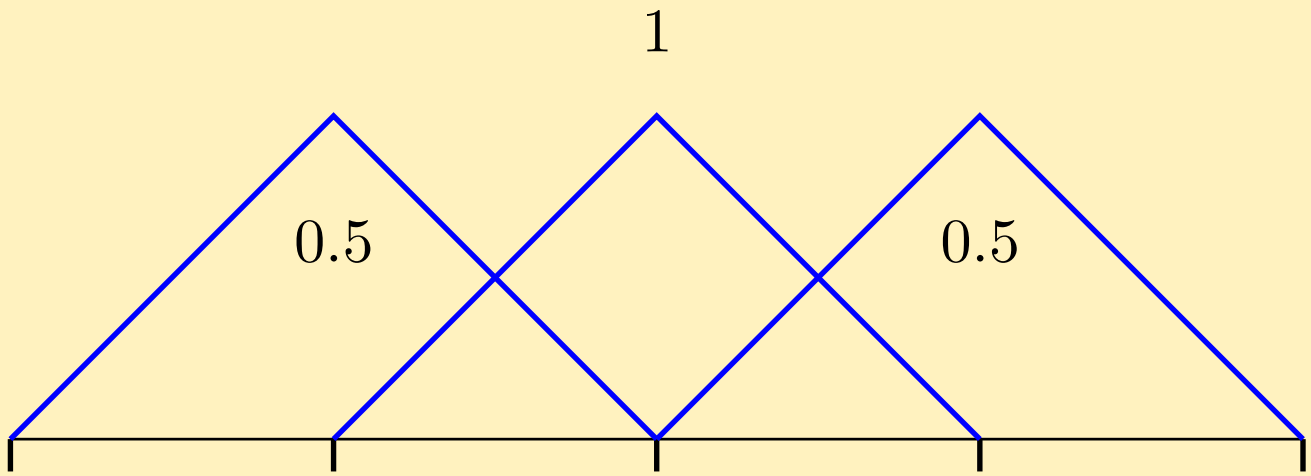
Evaluate coarse grid basis functions at fine grid points.



Can be interpreted as interpolation based on basis functions.

Restriction

Transpose of prolongation.

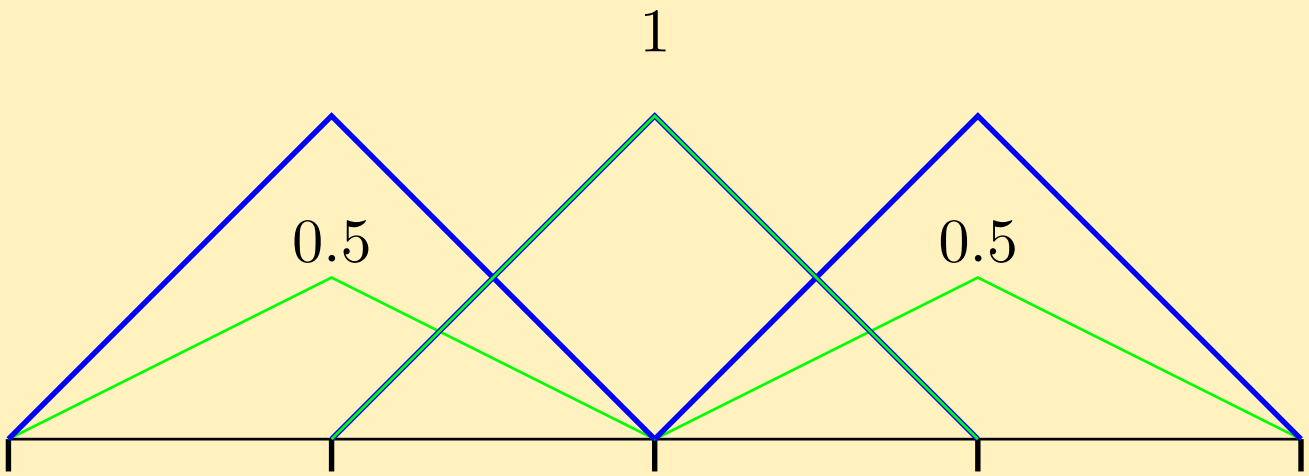


Can be interpreted as weighted averageing

Restriction

universitätbonn

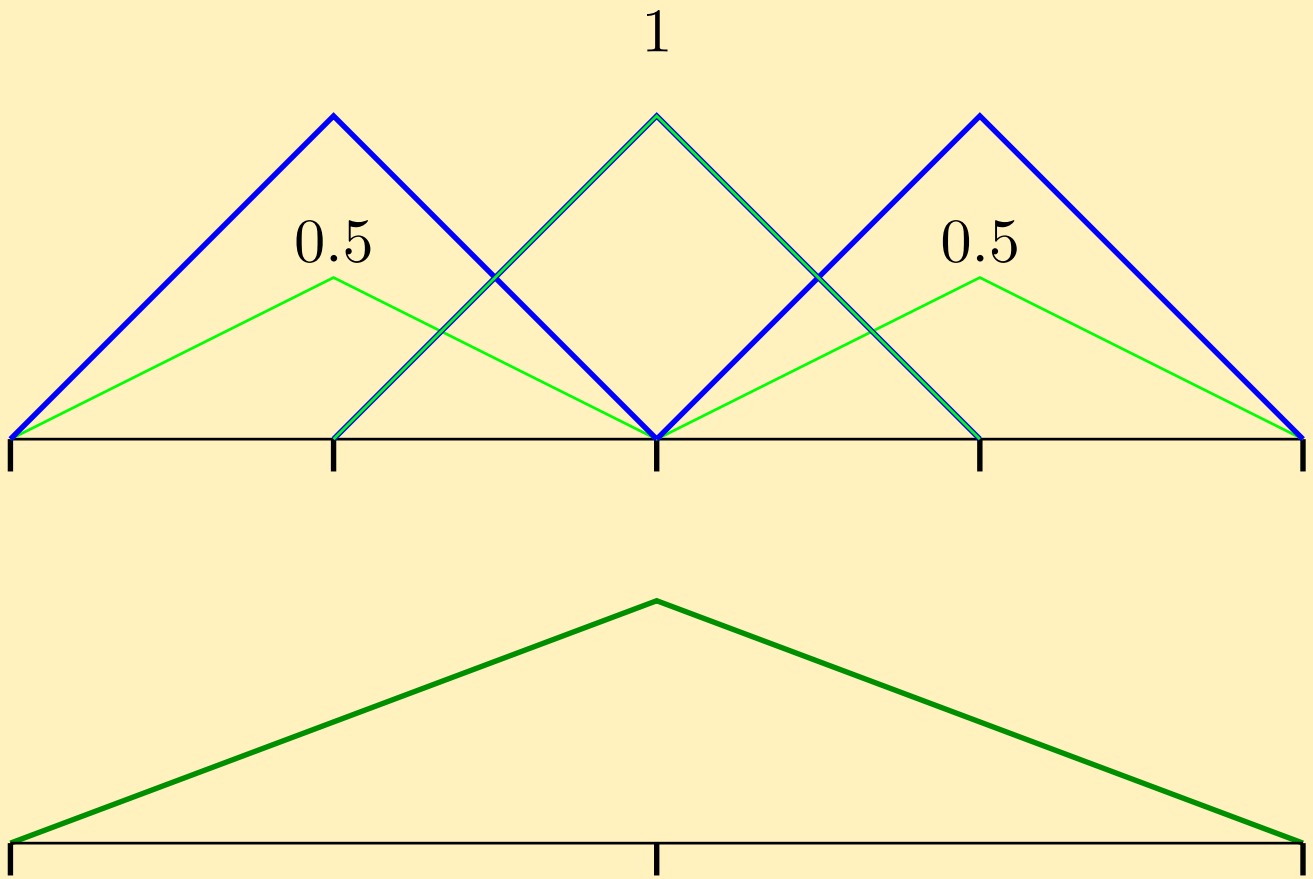
Transpose of prolongation.



Can be interpreted as weighted averageing

Restriction

Transpose of prolongation.



Can be interpreted as weighted averageing

Algorithmic Complexity

For $A \in \mathbb{R}^{n \times n}$ with β nonzero bands, solve

$$Ax = b$$

to accuracy ϵ, e. g.

$$\frac{\|Ax^n - b\|}{\|Ax^0 - b\|} \leq \epsilon$$

Algorithmic Complexity

For $A \in \mathbb{R}^{n \times n}$ with β nonzero bands, solve

$$Ax = b$$

to accuracy ϵ, e. g.

$$\frac{\|Ax^n - b\|}{\|Ax^0 - b\|} \leq \epsilon$$

method	**complexity**	**convergence rate**
Gaußian elimination	$O(n^3)$	—
Jacobi, Gauß-Seidel	$O(\beta n \lvert \log \epsilon \rvert)$	$O\left(\frac{\kappa-1}{\kappa+1}\right)$
conjugate gradient	$O(\beta n \lvert \log \epsilon \rvert)$	$O\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)$
multigrid	$O(\beta n \lvert \log \epsilon \rvert)$	not destroyed by large κ

Algorithmic Complexity

universitätbonn

For $A \in \mathbb{R}^{n \times n}$ with β nonzero bands, solve

$$Ax = b$$

to accuracy ϵ, e. g.

$$\frac{\|Ax^n - b\|}{\|Ax^0 - b\|} \leq \epsilon$$

method	**complexity**	**memory**	
Gaußian elimination	$O(n^3)$	$O(n^2)$	
Jacobi, Gauß-Seidel	$O(\beta n \lvert \log \epsilon \rvert)$	$O(\beta n)$ or less	$\left(\frac{\sqrt{\kappa}}{\sqrt{\kappa}}\right)$
conjugate gradient	$O(\beta n \lvert \log \epsilon \rvert)$	$O(\beta n)$ or less	$\left(\frac{\sqrt{\kappa}}{\sqrt{\kappa}}\right)$
multigrid	$O(\beta n \lvert \log \epsilon \rvert)$	$O(\beta n)$ or less	

Algebraic Multigrid Methods

If no geometric coarse scales available, purely algebraic coarsening can be used:

- sparsity structure of matrix reflects grid connectivity (if any)

Algebraic Multigrid Methods

If no geometric coarse scales available, purely algebraic coarsening can be used:

- sparsity structure of matrix reflects grid connectivity (if any)
- can translate sparsity structure of matrix into graph

Algebraic Multigrid Methods

If no geometric coarse scales available, purely algebraic coarsening can be used:

- sparsity structure of matrix reflects grid connectivity (if any)
- can translate sparsity structure of matrix into graph
- can extract “strongly connected components” from graph

Algebraic Multigrid Methods

If no geometric coarse scales available, purely algebraic coarsening can be used:

- sparsity structure of matrix reflects grid connectivity (if any)
- can translate sparsity structure of matrix into graph
- can extract “strongly connected components” from graph
- can define coarsening only based on this information

Composite FE for Complicated Geometries

universitätbonn

Finite Elements:

Composite FE for Complicated Geometries

universitätbonn

Finite Elements:

- structured cubic grids:
  - efficient
  - natural coarse scales
  - poor geometric flexibility

Composite FE for Complicated Geometries

universitätbonn

Finite Elements:

- structured cubic grids:
  - efficient
  - natural coarse scales
  - poor geometric flexibility
- unstructured triangular/tetrahedral grids
  - high geometric flexibility
  - meshing necessary
  - no natural coarse scales

Composite FE for Complicated Geometries

universitätbonn

Finite Elements:

- structured cubic grids:
  - efficient
  - natural coarse scales
  - poor geometric flexibility
- unstructured triangular/tetrahedral grids
  - high geometric flexibility
  - meshing necessary
  - no natural coarse scales
  - key property: complicated grid with simple basis functions

Composite FE for Complicated Geometries

universitätbonn

Finite Elements:

- structured cubic grids:
  - efficient
  - natural coarse scales
  - poor geometric flexibility
- unstructured triangular/tetrahedral grids
  - high geometric flexibility
  - meshing necessary
  - no natural coarse scales
  - key property: complicated grid with simple basis functions

Composite Finite Elements:

- combine advantages of both

Composite FE for Complicated Geometries

universitätbonn

Finite Elements:

- structured cubic grids:
  - efficient
  - natural coarse scales
  - poor geometric flexibility
- unstructured triangular/tetrahedral grids
  - high geometric flexibility
  - meshing necessary
  - no natural coarse scales
  - key property: complicated grid with simple basis functions

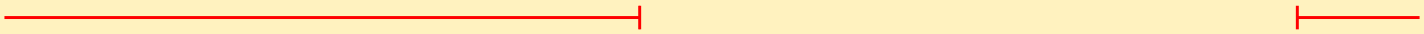
Composite Finite Elements:

- combine advantages of both
- use simple grid with complicated basis functions

Composite FE in 1D

universitätbonn

[Hackbusch, Sauter 1997]



Composite FE in 1D

universitätbonn

[Hackbusch, Sauter 1997]

- equidistant grid for complicated geometry, based on image data

Composite FE in 1D

universitätbonn

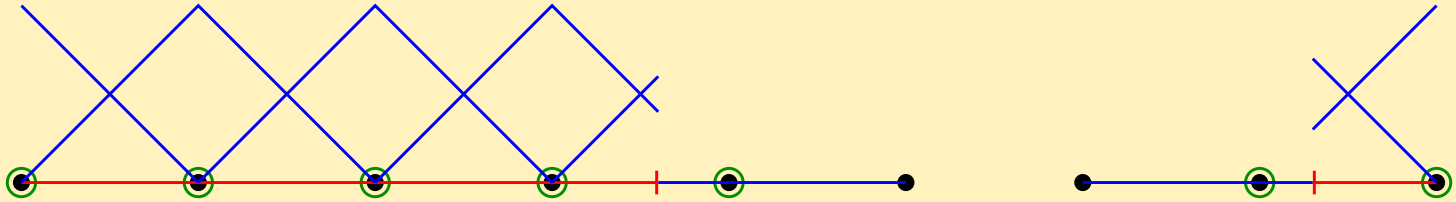
[Hackbusch, Sauter 1997]

- equidistant grid for complicated geometry, based on image data
- one layer of DOF outside object

Composite FE in 1D

universitätbonn

[Hackbusch, Sauter 1997]

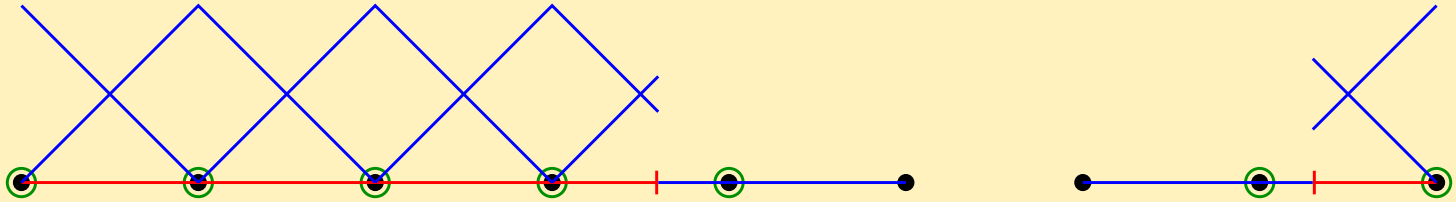


- equidistant grid for complicated geometry, based on image data
- one layer of DOF outside object
- basis functions cut off outside object

Composite FE in 1D

universitätbonn

[Hackbusch, Sauter 1997]

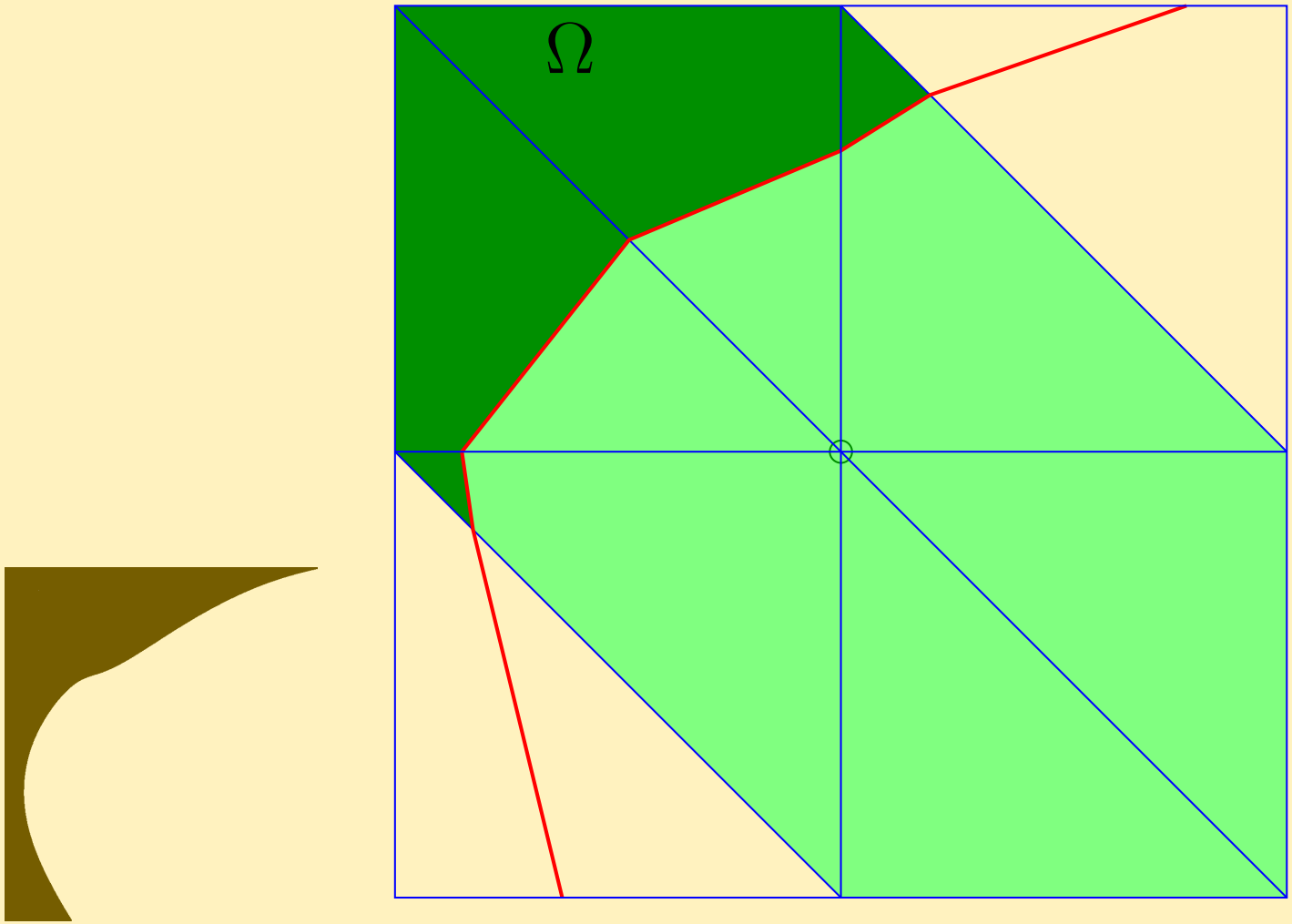


- equidistant grid for complicated geometry, based on image data
- one layer of DOF outside object
- basis functions cut off outside object

CFE basis functions are

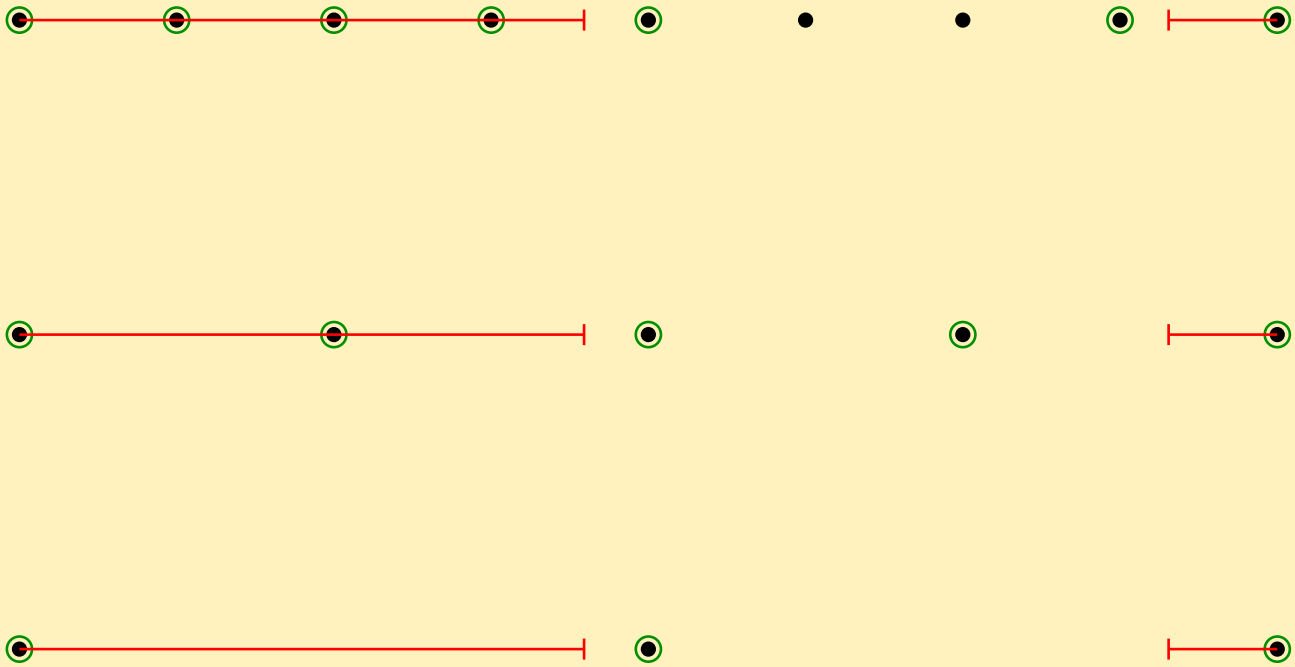
- piecewise linear
- nodal
- partition of unity

Composite FE in 2D



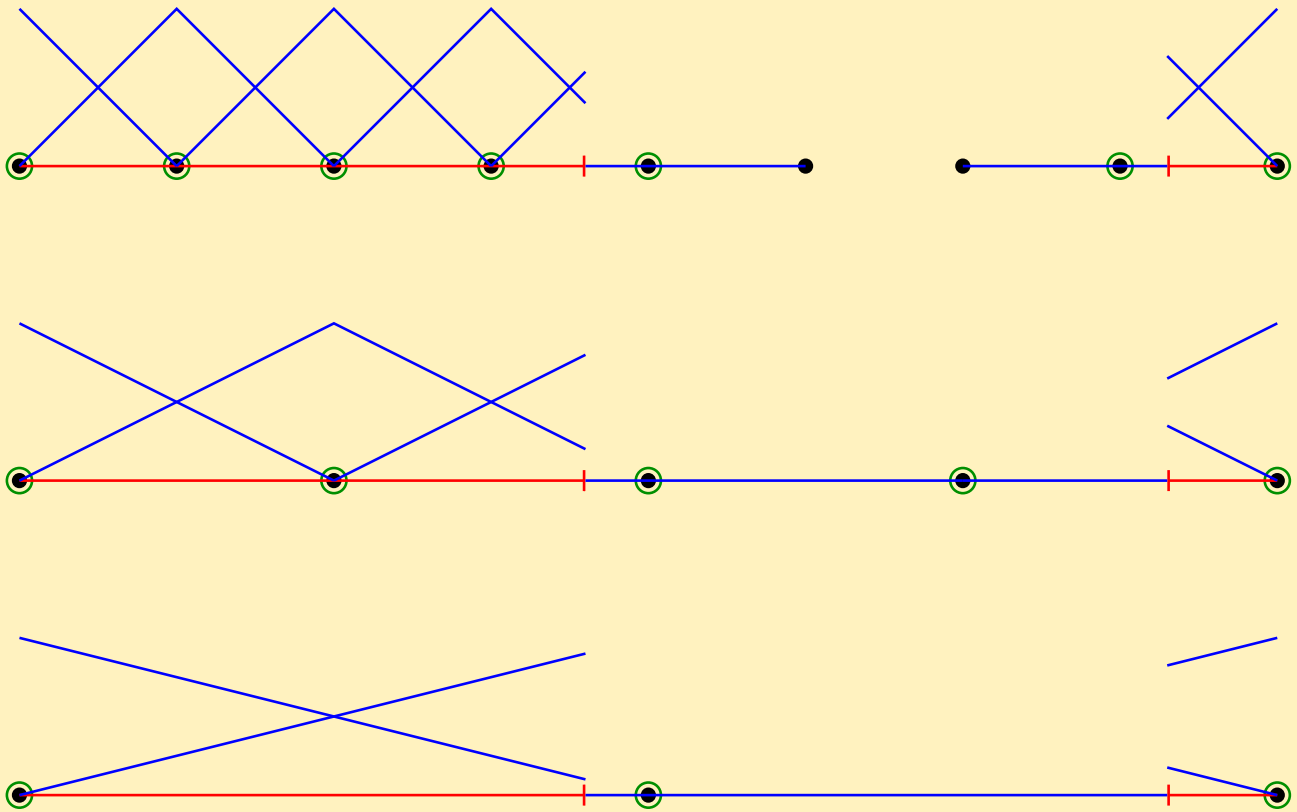
Basis function supported in dark green region and set to zero in light green region

Multigrid: Coarsening in 1D



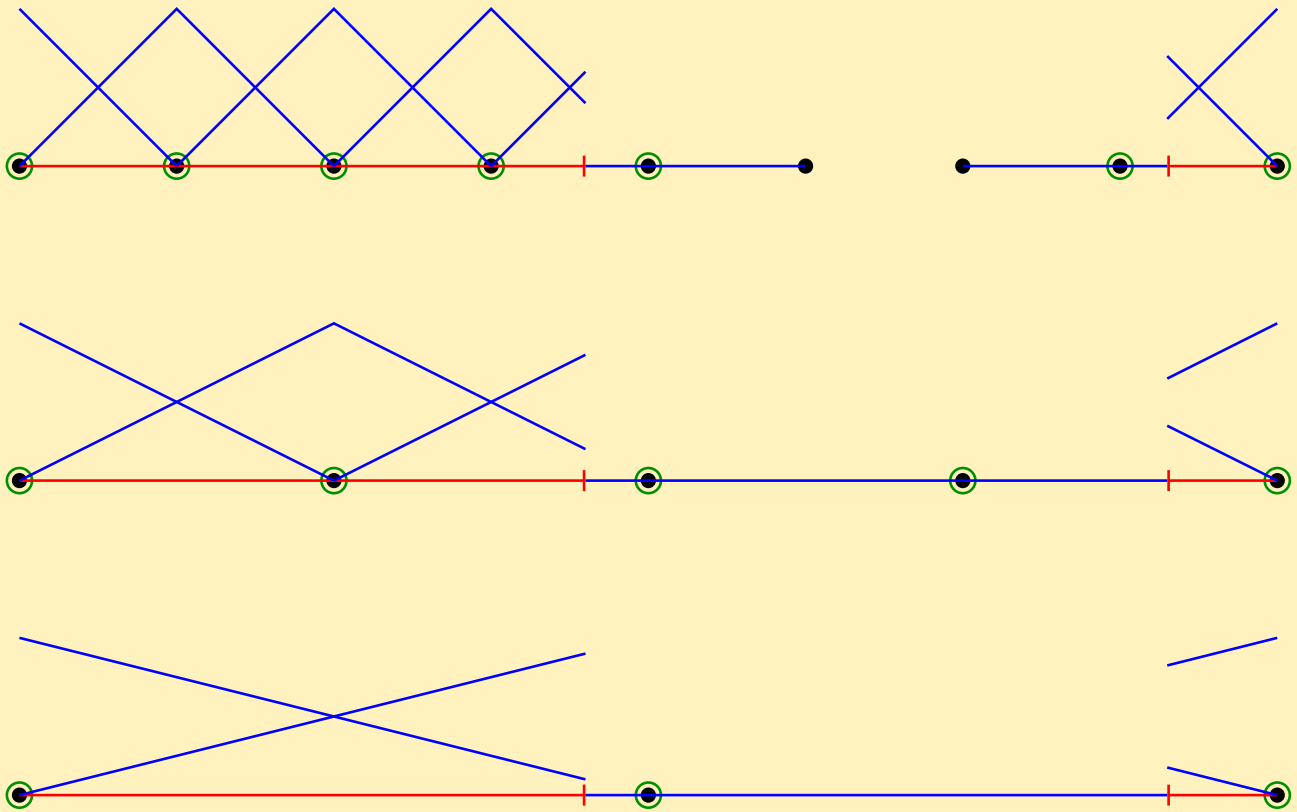
- natural grid hierarchy

Multigrid: Coarsening in 1D



- natural grid hierarchy

Multigrid: Coarsening in 1D

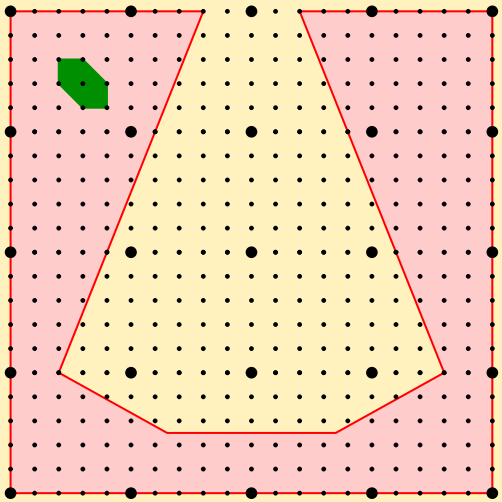


- natural grid hierarchy
- last step questionable: same DOF for separate parts

Multigrid: Artificial Coupling

universitätbonn

"Horseshoe" geometry

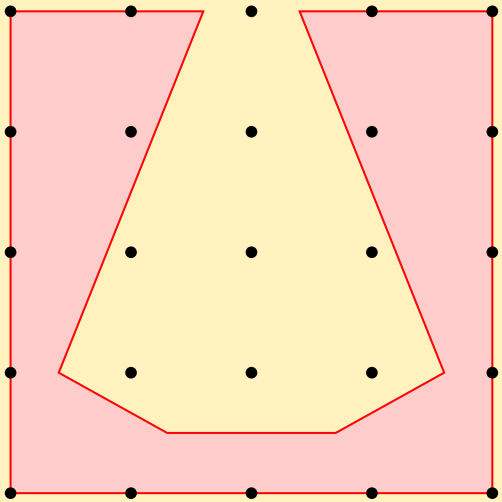


On fine scale: no problem.

Multigrid: Artificial Coupling

universitätbonn

"Horseshoe" geometry

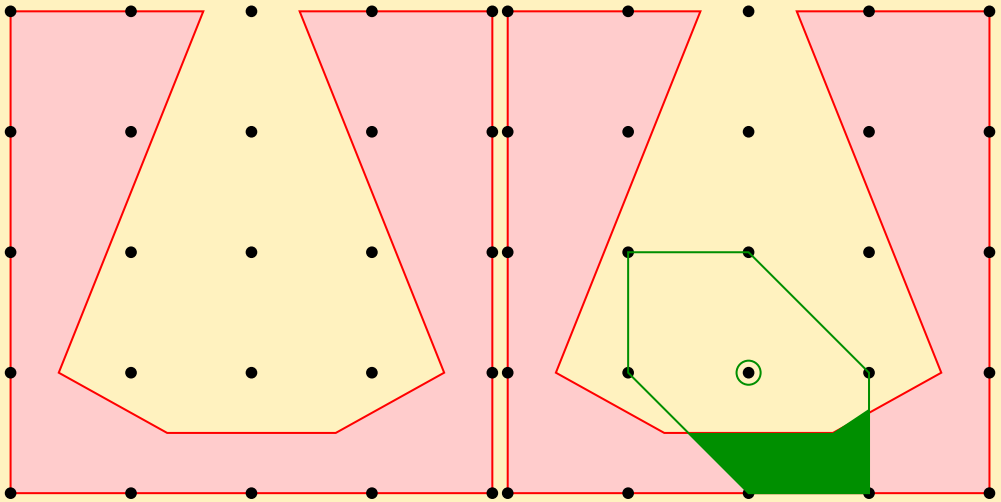


On coarse scale

Multigrid: Artificial Coupling

universitätbonn

"Horseshoe" geometry

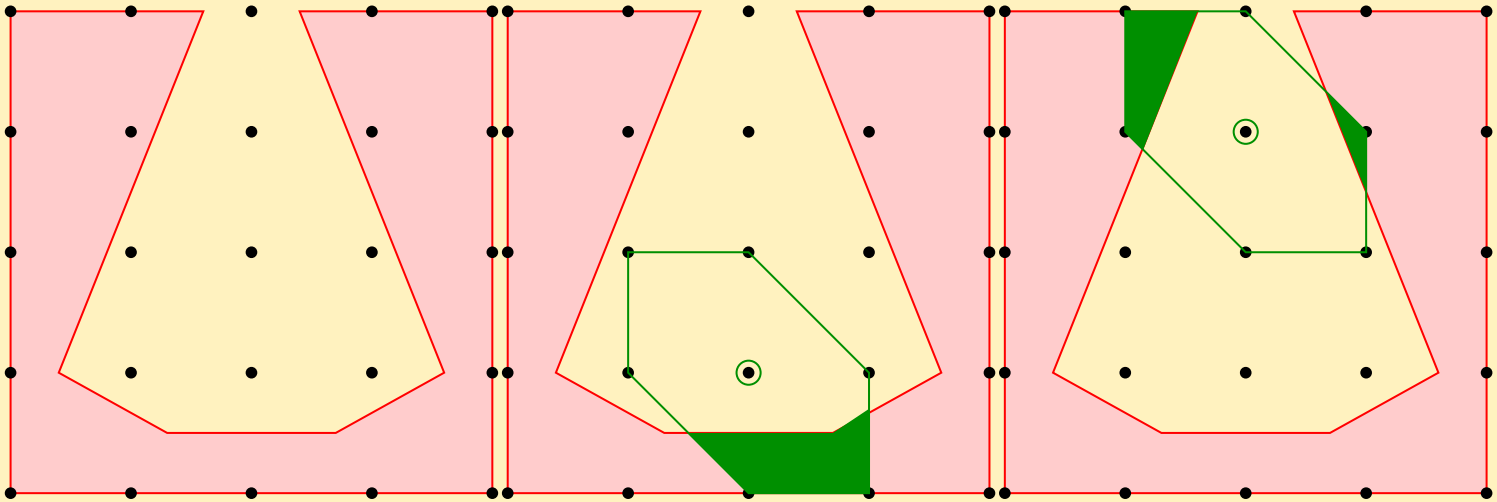


On coarse scale

Multigrid: Artificial Coupling

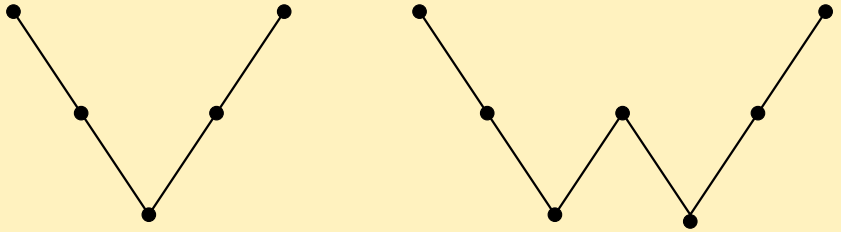
universitätbonn

"Horseshoe" geometry



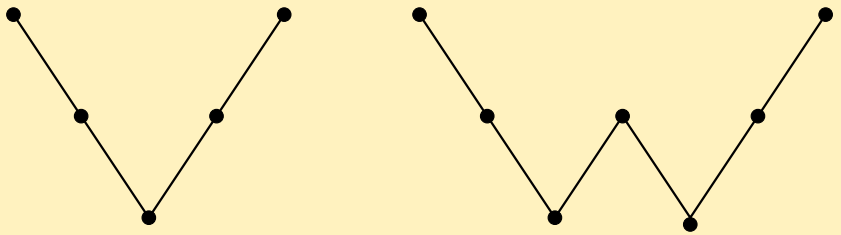
On coarse scale: FE coupling in simulations does not reflect physical coupling.

Truncated Multigrid



Globally stop coarsening process as soon as “horseshoe effect” occurs.

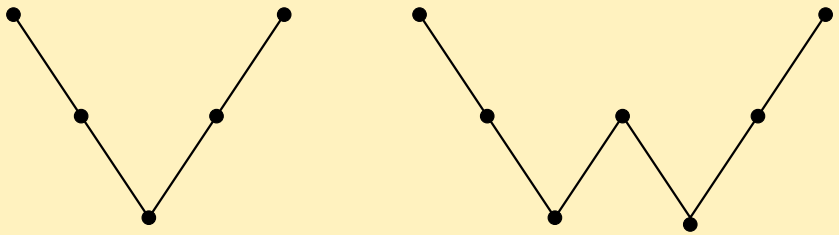
Truncated Multigrid



Globally stop coarsening process as soon as “horseshoe effect” occurs.

- ✔ simple modification
- ✔ good convergence rates

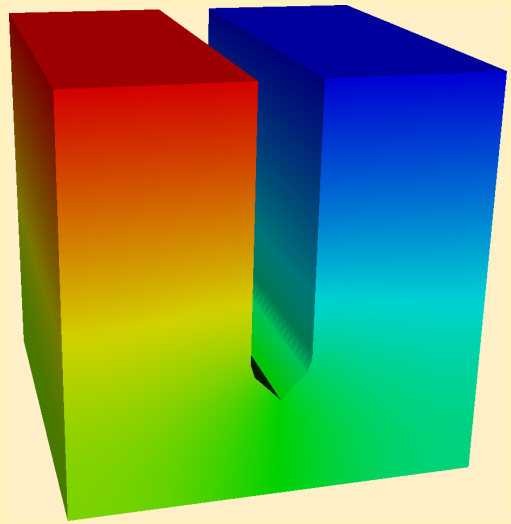
Truncated Multigrid



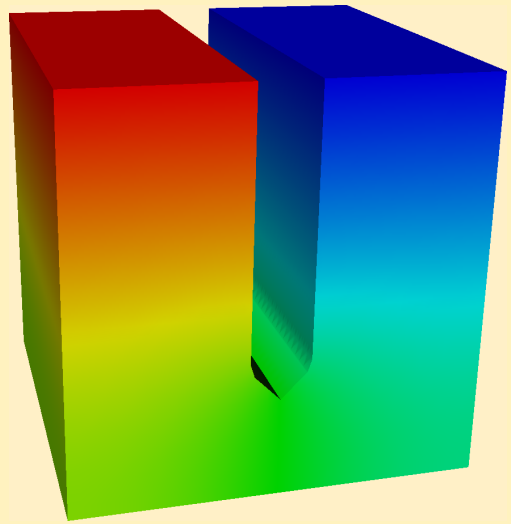
Globally stop coarsening process as soon as “horseshoe effect” occurs.

- ✔ simple modification
- ✔ good convergence rates
- ✘ slow, because large problems need to be solved directly

Truncated Multigrid, Example



Truncated Multigrid, Example



slot width	coarsest grid	convergence rate
1/7	1/16	0.117
1/7	1/8	0.451
1/14	1/64	0.107
1/14	1/32	0.171
1/14	1/16	0.650

Adaptively Stopped Coarsening

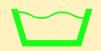
Recall basic multigrid cycle:

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2.
3. Compute Residual: $r = b - Ax^{(1)}$.
4. **Restrict** the residual: compute coarse version $\tilde{r}$
5. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 
6. **Prolongate** the result: compute fine e
7. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$
8.
9. **Postsmooth**: Perform ν_2 Gauß-Seidel iterations 

Adaptively Stopped Coarsening

universitätbonn

Distinguish between "standard" nodes and "horseshoe" nodes:

1. **Presmooth**: Perform ν_1 Gauß-Seidel iterations: $x^{(1)}$ 
2. Perform α_1 additional Gauß-Seidel iterations for "horseshoe" nodes only 
3. Compute Residual: $r = b - Ax^{(1)}$.
4. **Restrict** the residual: compute coarse version $\tilde{r}$ ignoring "horseshoe" nodes
5. **Solve** $\tilde{A}\tilde{e} = \tilde{r}$ 
6. **Prolongate** the result: compute fine e ignoring "horseshoe" nodes
7. Add the coarse grid correction: $x^{(2)} = x^{(1)} + e$
8. Perform α_2 additional Gauß-Seidel iterations for "horseshoe" nodes only 
9. **Postsmooth**: Perform ν_2 Gauß-Seidel iterations 

Adaptively Stopped Coarsening

- ✔ easy to implement
- ✘ minor or no improvement of convergence
- theoretical implications for convergence?

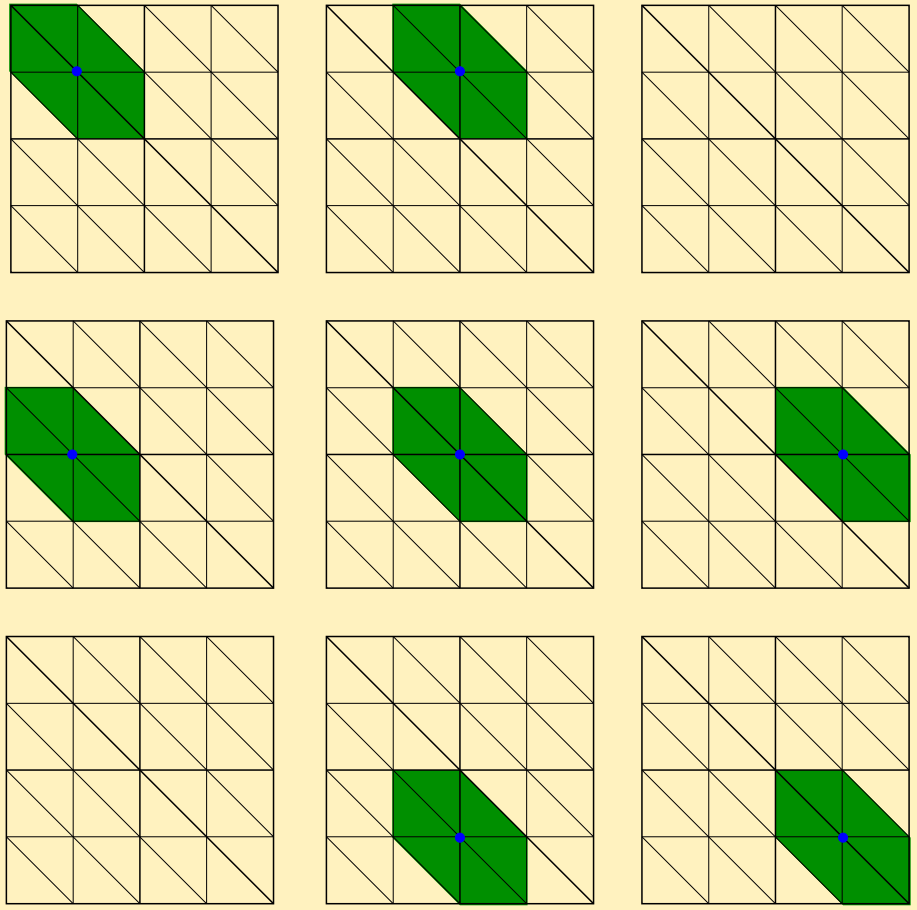
Adaptively Stopped Coarsening

- ✔ easy to implement
- ✘ minor or no improvement of convergence
- theoretical implications for convergence?

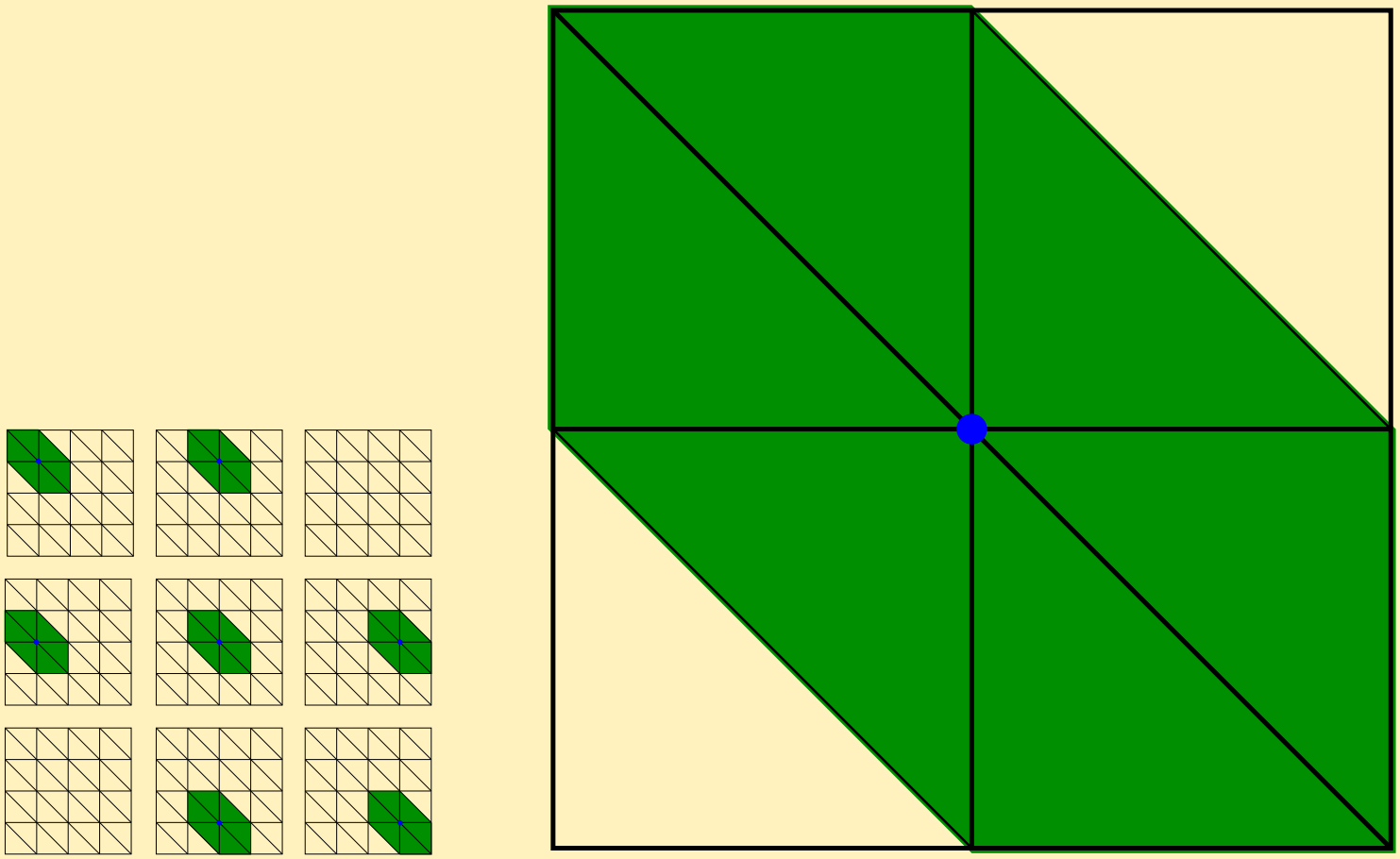
slot width	coarsest grid	conv. rate std MG	adapted MG
1/7	1/16	0.117	
1/7	1/8	0.451	0.416
1/14	1/64	0.107	
1/14	1/32	0.171	
1/14	1/16	0.650	0.816

Review: Standard Grid Coarsening (2D)

universitätbonn

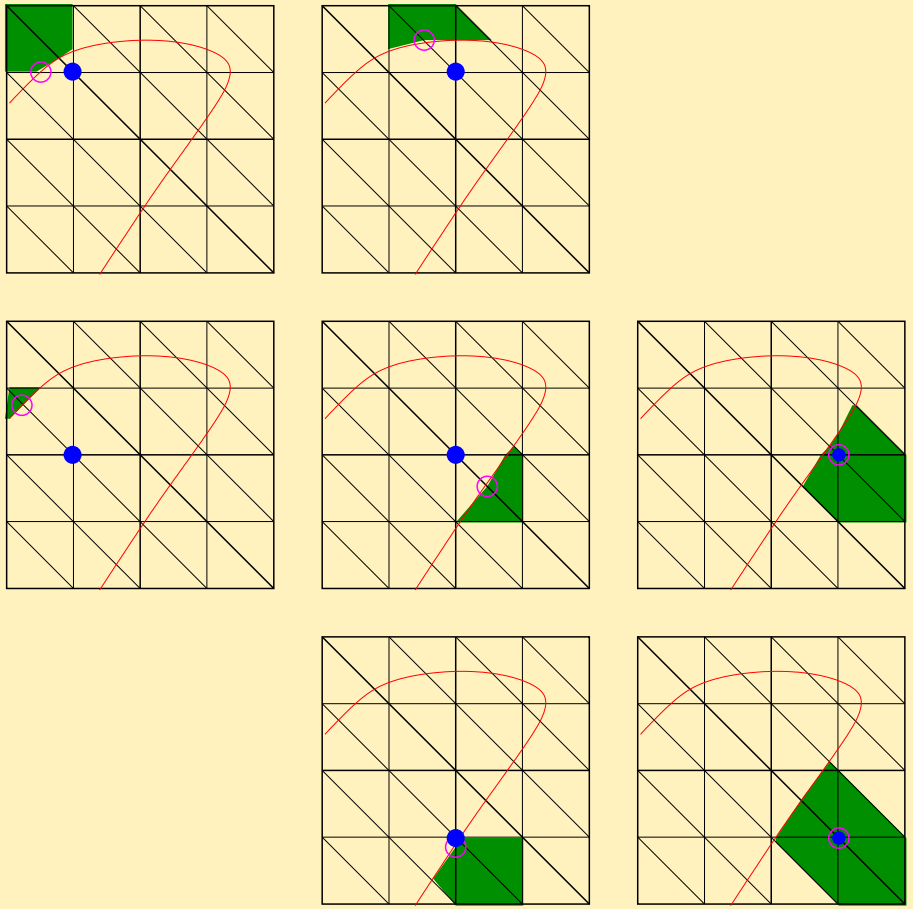


Review: Standard Grid Coarsening (2D)

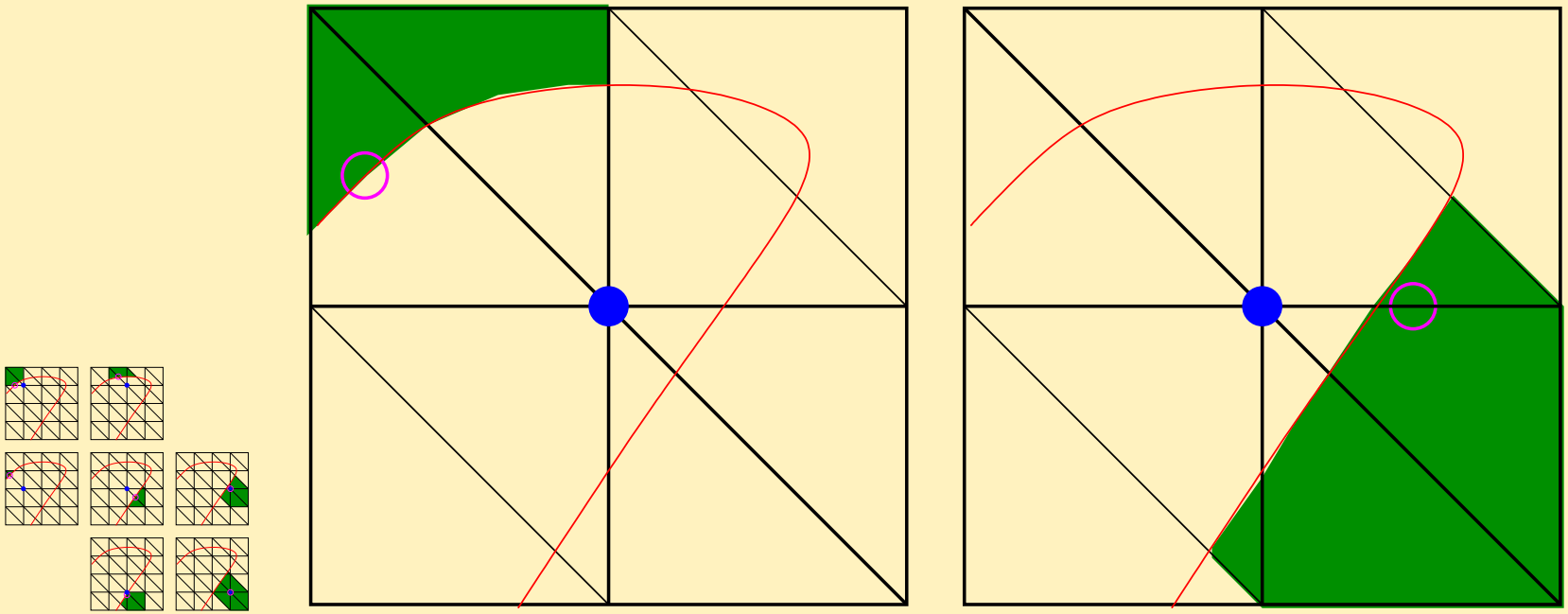


One fine basis function and its neighbors are combined to a coarse basis function.

Coarsening by Hierarchical Partitioning (2D)

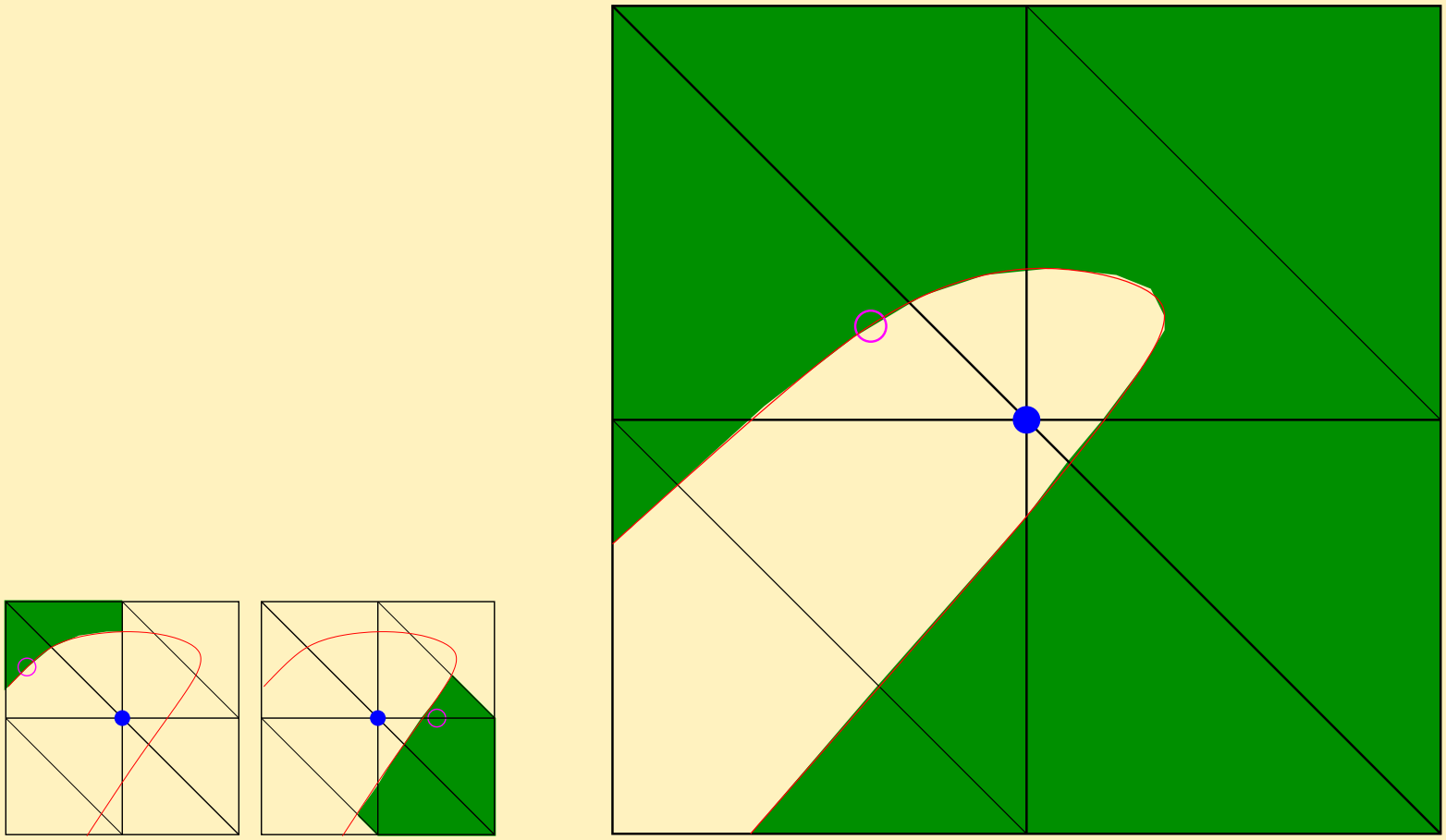


Coarsening by Hierarchical Partitioning (2D)



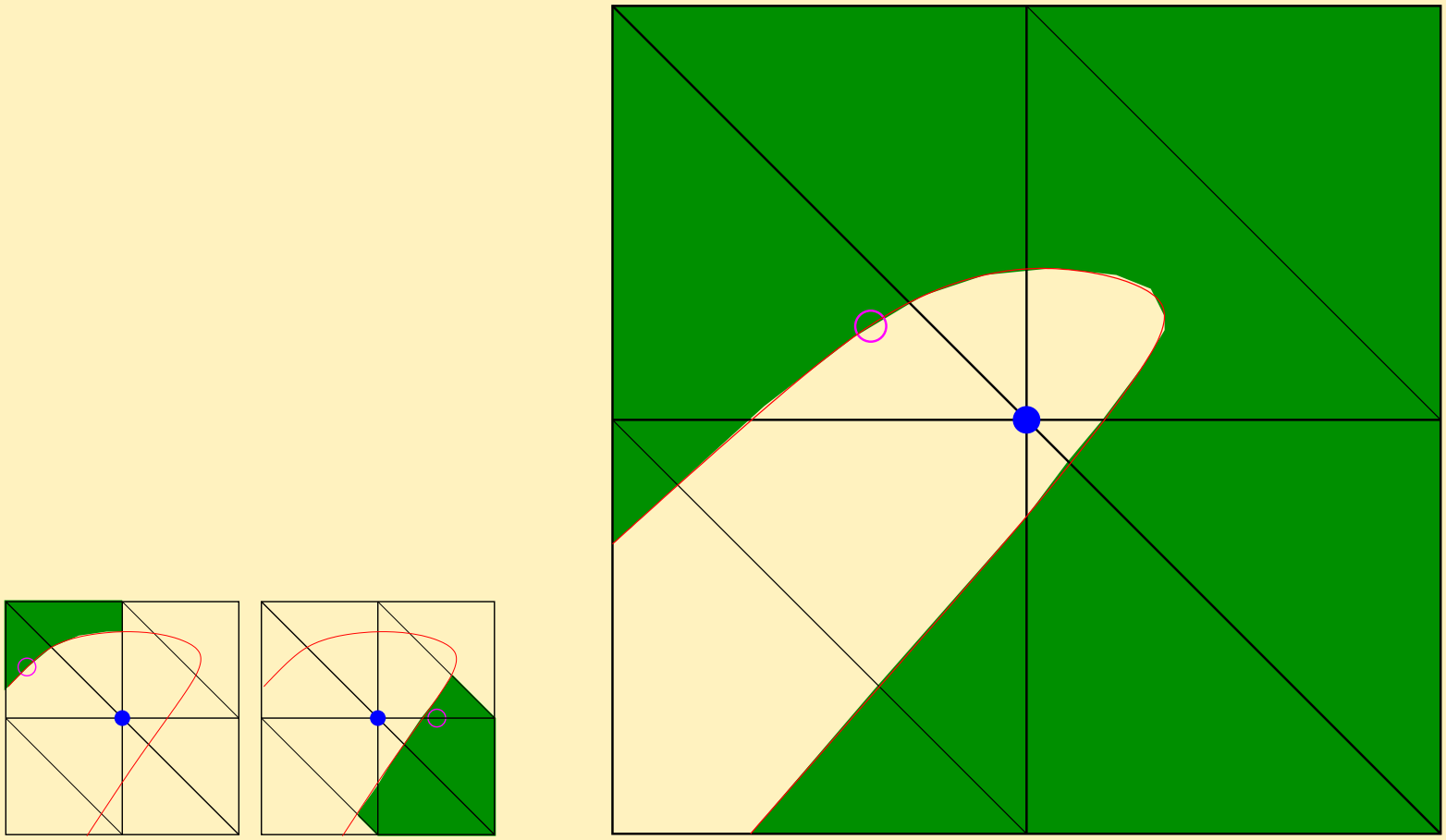
Fine DOFs and basis functions combined to *two* intermediate ones.

Coarsening by Hierarchical Partitioning (2D)



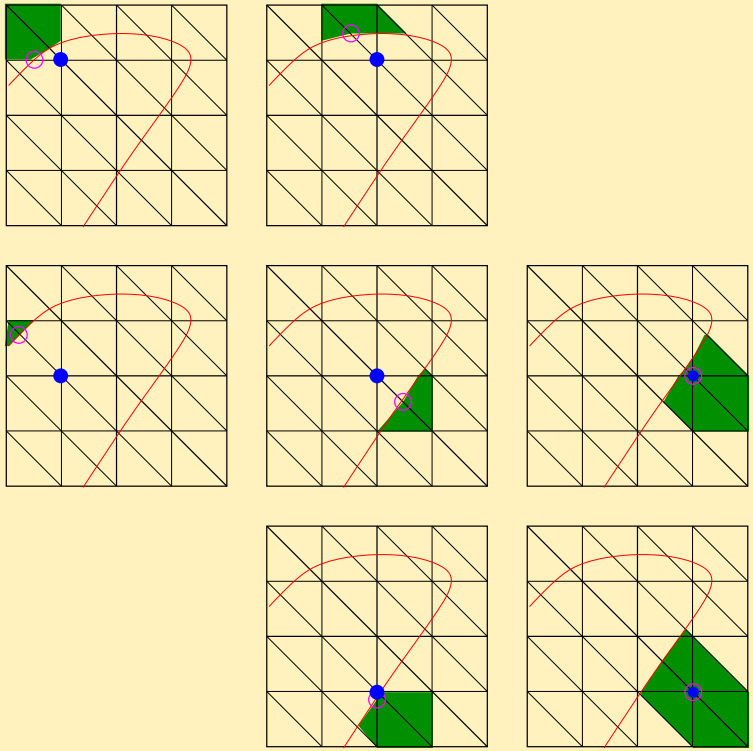
Intermediate DOFs and basis functions recombined to *same* coarse ones.

Coarsening by Hierarchical Partitioning (2D)



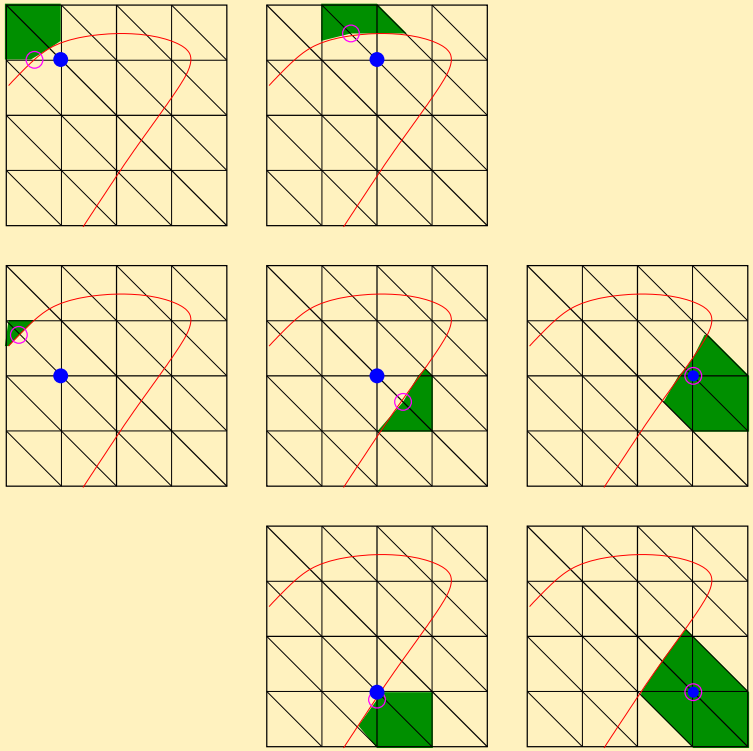
Partitioning must not be more expensive than $O(n)$ (at least not much more . . .)

Hierarchical Partitioning



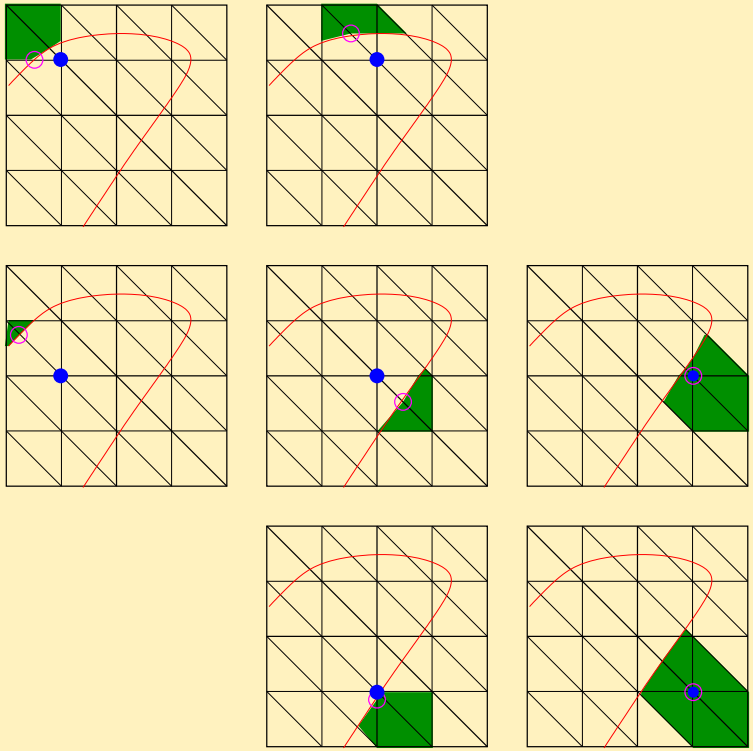
- assign “seed” to each DOF

Hierarchical Partitioning



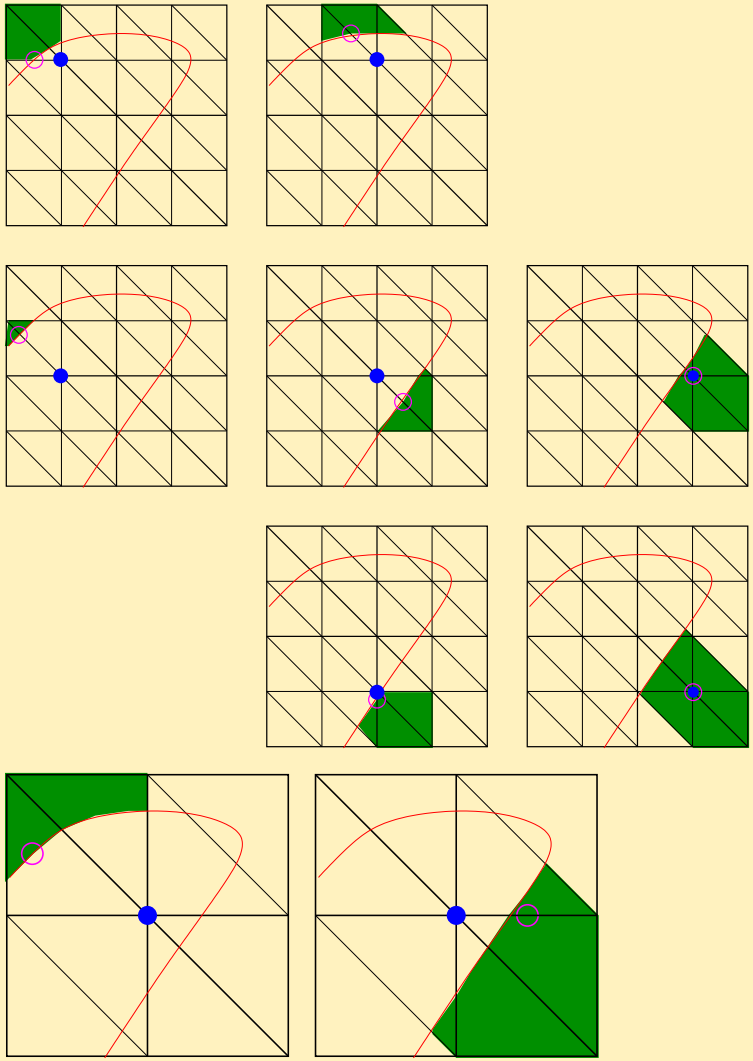
- assign “seed” to each DOF
- compute geodesic distances in “neighborhoods” of seeds

Hierarchical Partitioning



- assign “seed” to each DOF
- compute geodesic distances in “neighborhoods” of seeds
- combine “neighboring” seeds to “components”

Hierarchical Partitioning

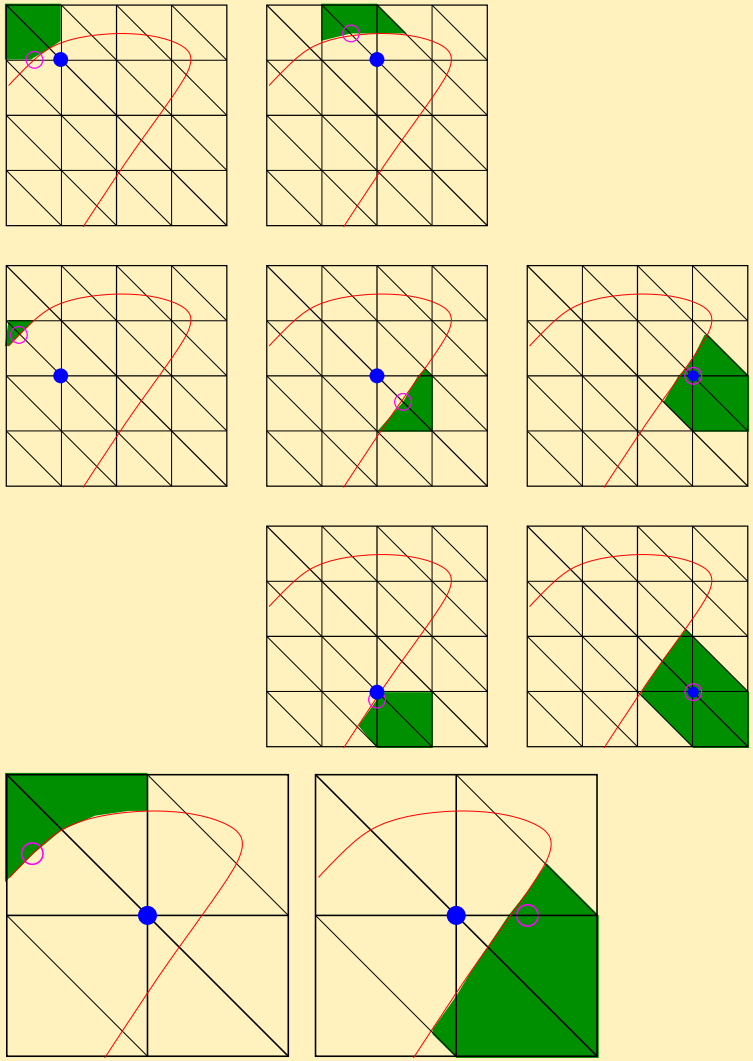


- assign “seed” to each DOF
- compute geodesic distances in “neighborhoods” of seeds
- combine “neighboring” seeds to “components”

Component given by

- grid level
- DOF
- seed
- children

Hierarchical Partitioning



- assign “seed” to each DOF
- compute geodesic distances in “neighborhoods” of seeds
- combine “neighboring” seeds to “components”

Component given by

- grid level
- DOF
- seed
- children

Now grow those neighborhoods with distance information and continue . . .

Back to Multigrid

Goal: Use this hierarchy to define $\mathcal{R}$estriction and $\mathcal{P}$rolongation.

Back to Multigrid

Goal: Use this hierarchy to define $\mathcal{R}$estriction and $\mathcal{P}$rolongation.

Issues:

- (cache-) efficient algorithmical treatment of “cloned” DOFs
- keep track of different components on different levels
- overall efficiency?

Summary

1. FE Simulations and Systems of Equations
2. Solvers for Systems of Equations
   - Direct and Iterative Methods
   - Conjugate Gradient Solvers
3. Multigrid Methods
   - Smoothing Properties of Iterative Solvers
   - Multigrid for Uniform Grids
   - Algebraic Multigrid Methods
4. Multigrid Methods for Complicated Geometries
   - Composite Finite Elements
   - Multigrid for CFE
   - Improvement of the Multigrid Method
   - Hierarchical Partitioning

References

- F. Liehr, T. Preusser, M. Rumpf, S. Sauter and O. Schwen: *Composite Finite Elements for 3D Image Based Computing*, submitted to Computing and Visualization in Science, 2007
- T. Preusser, M. Rumpf and O. Schwen: *Finite Element Simulation of Bone Microstructures*, Proceedings of the 14th Finite Element Workshop, Ulm 2007

Contact:

`ole.schwen@ins.uni-bonn.de`

`http://numod.ins.uni-bonn.de`