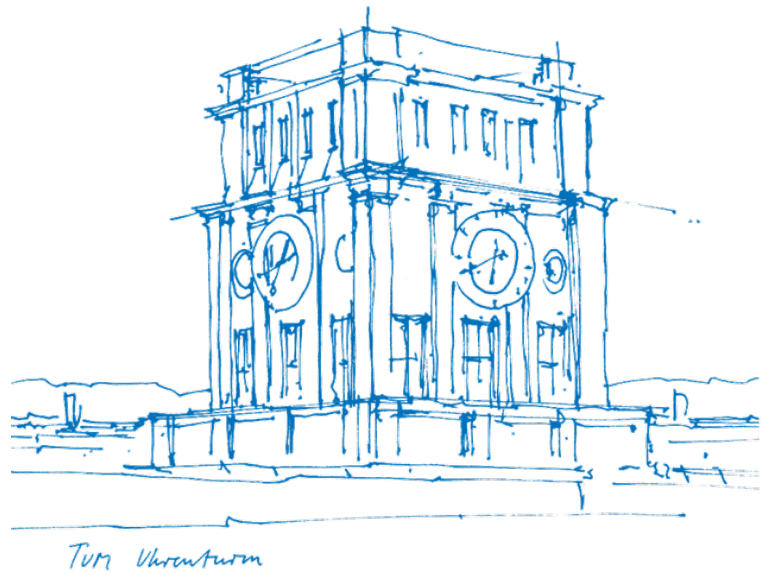
Scientific Computing II

Towards Multigrid Methods

Michael Bader
Technical University of Munich

Summer 2018



Part I

Multigrid Methods

Fundamental Multigrid Ideas
- Nested Iteration
- Coarse-Grid Correction – A Two-Grid Method

Multigrid Schemes
- Multigrid V-Cycle
- Multigrid W-Cycle
- Full Multigrid V-Cycle

Computational Effort
- Costs per Iteration
- Speed of Convergence

Multigrid Idea No. 1

Observation and convergence analysis show:

- “high-frequency error” is relative to mesh size
- on a sufficiently coarse grid, even very low frequencies can be “high-frequency”
  (if the mesh size is big)

“Multigrid” idea:

- use multiple grids to solve the system of equations
- hope that on each grid, a certain range of error frequencies will be reduced efficiently

Nested Iteration

Solve the problem on a coarser grid:

- will be comparably (very) fast
- can give us a good initial guess:
- leads to “poor man’s multigrid”: **nested iteration**

Algorithm:

1. Start on a very coarse grid with mesh size $h = h_0$;
   guess an initial solution x_h
2. Iterate over $A_h x_h = b_h$ using **relaxation** method
   $\Rightarrow$ approximate solution x_h
3. **interpolate** the solution x_h to a finer grid $\Omega_{h/2}$
4. proceed with step 2 (now with mesh size $h := h/2$) using interpolated $x_{h/2}$ as initial solution

Multigrid Idea No. 2

Observation for nested iteration:

- error in interpolated initial guess also includes low frequencies
- relaxation therefore still slow
- can we go “back” to a coarser grid later in the algorithm?

Idea No. 2: use the residual equation
$\Rightarrow$ **coarse-grid correction**:

- solve $Ae = r$ on a coarser grid
- leads to an approximation of the error e
- add this approximation to the fine-grid solution

A Two-Grid Method

Algorithm:

1. **relaxation/smoothing** on the fine level system
   $\Rightarrow$ solution x_h
2. compute the **residual** $r_h = b_h - A_h x_h$
3. **restriction** of r_h to the coarse grid Ω_H
4. compute a **solution** to $A_H e_H = r_H$
5. **interpolate** the coarse grid solution e_H to the fine grid Ω_h
6. add the resulting **correction** to x_h
7. again, **relaxation/smoothing** on the fine grid

Correction Scheme – Components

- **smoother:**
  reduce the high-frequency error components, and get a smooth error
- **restriction:**
  transfer residual from fine grid to coarse grid, for example by
  - injection
  - (full) weighting
- **coarse grid equation:**
  (acts as) discretisation of the PDE on the coarse grid
- **interpolation:**
  transfer coarse grid solution/correction from coarse grid to fine grid

The Multigrid V-Cycle

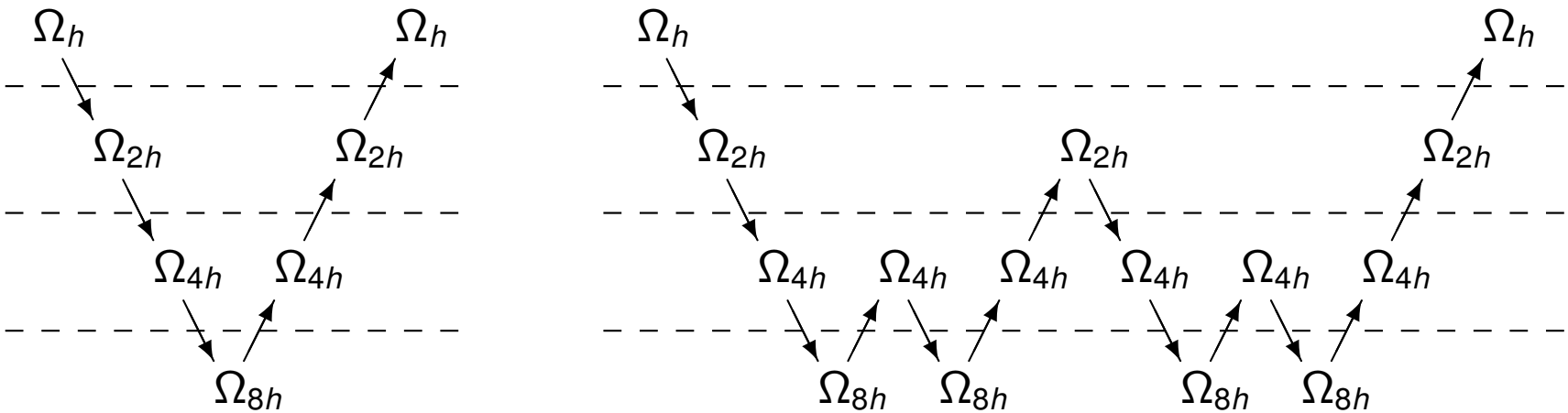
Crucial idea: recursive call of Two-/Multigrid solver on coarse grids

Algorithm:

1. pre-smoothing on the fine level system
   $\Rightarrow$ solution x_l
2. compute the residual $r_l = b_l - A_l x_l$
3. restriction of r_l to the coarse grid Ω_{l-1}
4. solve coarse grid system $A_{l-1} e_{l-1} = r_{l-1}$
   by a **recursive call to the V-cycle algorithm**
5. interpolate the coarse grid solution e_{l-1} to the fine grid Ω_l
6. add the resulting correction to x_l
7. post-smoothing on the fine grid

The W-Cycle

- perform **two** coarse grid correction steps instead of one



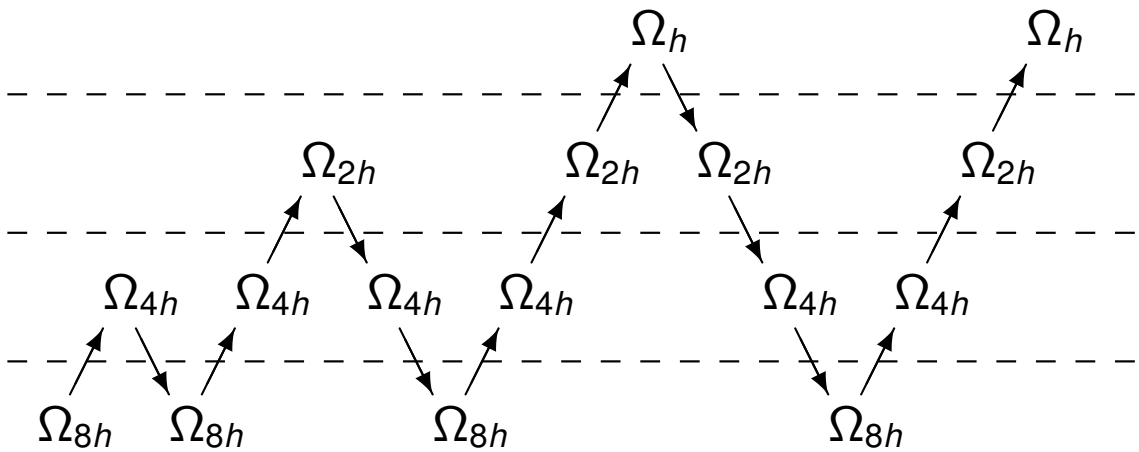
(V-cycle and W-cycle)

- more expensive
- useful in situations where the coarse grid correction is not very accurate

The Full Multigrid V-Cycle (FMV- or F-Cycle)

Recursive algorithm:

- combines nested iteration and V-cycle
- (recursively!) perform an **FMV-cycle** on the next coarser grid to get a good initial solution
- interpolate this initial guess to the current grid
- perform a **V-cycle** to improve the solution



V-Cycle – Computational Costs

Further algorithmic details:

- on the coarsest grid: direct solution;
  but: number of unknowns small, ideally $\mathcal{O}(1)$
- number of smoothing steps is typically very small (1 or 2)
  and must not depend on problem size

Computational Costs (storage and computing time):

- 1D: $c \cdot n + c \cdot n/2 + c \cdot n/4 + \ldots \leq 2c \cdot n$
- 2D: $c \cdot n + c \cdot n/4 + c \cdot n/16 + \ldots \leq 4/3c \cdot n$
- 3D: $c \cdot n + c \cdot n/8 + c \cdot n/64 + \ldots \leq 8/7c \cdot n$
- overall costs are dominated by the costs of the finest grid
  (n the number of grid points on the finest grid: typicall $n = h^{-D}$)

Thus: runtime $\mathcal{O}(n)$ per iteration, but how many iterations necessary?

Speed of Convergence

- fastest method around (if all components are chosen carefully), but: best-possible convergence often hard to obtain
- **“textbook multigrid efficiency”**:

$$\left\| e^{(m+1)} \right\| \leq \gamma \left\| e^{(m)} \right\|,$$

  where convergence rate $\gamma < 1$ (esp. $\gamma \ll 1$) is independent of the number of unknowns
  - $\Rightarrow$ constant number of multigrid steps to obtain a given number of digits
  - $\Rightarrow$ overall computational work increases only linearly with the number of unknowns
- see exercises: analysis of two-grid convergence

Speed of Convergence (2)

For the “Model Problem” (i.e., Poisson Problem):

- $\mathcal{O}(n)$ to solve up to “reduce error by a factor of . . . ” (10^{-8}, e.g.)
- wanted: solve to the **“level of truncation”**
  $\rightarrow$ depending on discretisation error; for example $\mathcal{O}(h^2)$
- $\mathcal{O}(n)$ up to “level of truncation” achieved by FMV-Cycle;
  ideal case: 1 cycle; after each V-cycle, the “level of truncation” error is achieved on that grid on that grid

For Other Problems:

- OK for strongly elliptic problems
- multigrid variants for non-linear problems, parabolic/hyperbolic, . . .
- **every component may fail, leading to slow or no convergence:**
  smoother, interpolation/restriction, coarse-grid operator
- achieving “textbook efficiency” usually a demanding task

Part II

Components of Multigrid Methods

Interpolation
Restriction
Coarse Grid Operator
Smoothers

Interpolation (aka “Prolongation”)

For Poisson problem:

- (bi-)linear interpolation:
  in 1D: resembles homogeneous ($f = 0$) solution
- constant (in general too small approximation order):
  sometimes used for cell-based coarsening (unknowns located in cell centers)
- quadratic, cubic, etc.:
  often too costly, more smoothing steps are cheaper and can eliminate the disadvantage of a lower-order interpolation
- **but:** in FMV-cycle interpolation to finer grid (after a completed V-cycle) should be higher-order (to limit the introduced error)

Interpolation – Matrix Notation

For linear interpolation (1D):

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(0 + x_1) \\ x_1 \\ \frac{1}{2}(x_1 + x_2) \\ x_2 \\ \frac{1}{2}(x_2 + x_3) \\ x_3 \\ \frac{1}{2}(x_3 + 0) \end{pmatrix}$$

Notation: $I_{2h}^h x_{2h} = x_h$ or $P_{2h}^h x_{2h} = x_h$

Note: disregards boundary values (here: 0-Dirichlet condition assumed)

Interpolation – Convection-Diffusion

Example problem: 1D convection-diffusion equation

$$-\epsilon u_{xx} + c u_x = f, \qquad 0 < \epsilon \ll c$$

Operator-dependent Interpolation:

- consider homogeneous problem ($f = 0$)
  with Dirichlet boundaries: $u(0) = 1$, $u(1) = 0$
- exact solution for this case:

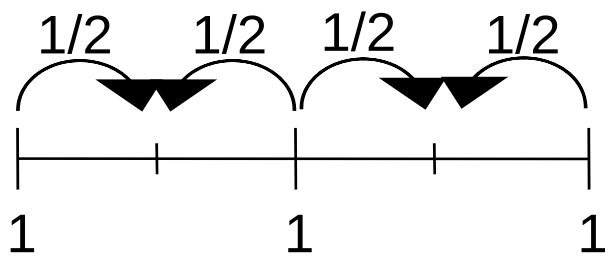
$$u(x) = \frac{1}{1 - e^{c/\epsilon}} \left(e^{cx/\epsilon} - e^{c/\epsilon} \right) = 1 - \frac{1 - e^{cx/\epsilon}}{1 - e^{c/\epsilon}}$$

- interpolate at $x = \frac{1}{2}$: $u\left(\frac{1}{2}\right) = 1 - \frac{1-e^{c/2\epsilon}}{1-e^{c/\epsilon}} =: 1 - \frac{1-z}{1-z^2}$;
  for large $z = e^{\frac{c}{2\epsilon}}$, we have $u\left(\frac{1}{2}\right) \approx 1 - \frac{1}{z} \approx 1$
- thus: linear interpolation inappropriate (and leads to slow convergence)
  $\rightarrow$ interpolation should be **operator-dependent**

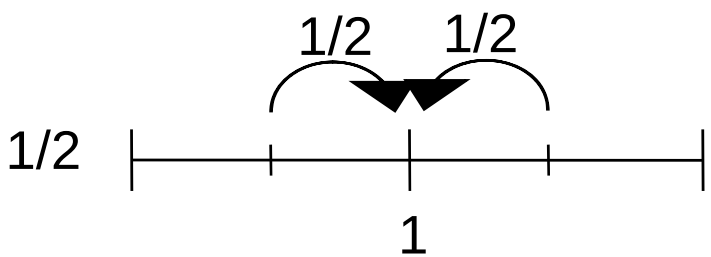
Restriction

For Poisson problem:

- “injection”: pick values at corresp. coarse grid points
- “full weighting” = transpose of bilinear interpolation (safer, more robust convergence), see illustration below for the 1D case



linear interpolation



full weighting

Restriction – Matrix Notation

For full weighting (1D):

$$\begin{pmatrix} \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(x_1 + 2x_2 + x_3) \\ \frac{1}{2}(x_3 + 2x_4 + x_5) \\ \frac{1}{2}(x_5 + 2x_6 + x_7) \end{pmatrix}$$

Notation: $I_h^{2h} x_h = x_{2h}$ or $R_h^{2h} x_h = x_{2h}$

Coarse Grid Operator

Two main options:

1. discretise PDE on grid Ω_h to obtain A_h
2. "Galerkin approach": $A_{2h} := R_h^{2h} A_h P_{2h}^h$
   $\rightarrow$ compare effect on vector x_{2h}:

$$A_{2h} x_{2h} := R_h^{2h} A_h P_{2h}^h x_{2h}$$

   $\rightarrow$ evaluate from right to left:
   - interpolate $x_2 h$ to $\hat{x}_h := P_{2h}^h x_{2h}$
   - apply fine-grid operator A_h to interpolated $\hat{x}_h$
   - restrict resulting matrix-vector product to Ω_{2h}

Exercise:

- Compute $A_{2h} := R_h^{2h} A_h P_{2h}^h$ for $A_h := \frac{1}{h^2}\text{tridiag}(-1, 2, -1)$

A Matrix-Oriented View of Coarse-Grid Correction

1. given a system of equations $A_h x_h = b_h$
2. pre-smoothing leads to approximate solution $x_h^{(i)}$
   and resp. error $x_h = x_h^{(i)} + e_h^{(i)}$
3. compute residual $r_h^{(i)} = b_h - A_h x_h^{(i)}$;
   respective residual equation: $A_h e_h^{(i)} = r_h^{(i)}$
4. restriction of residual equation: $R_h^{2h} A_h e_h^{(i)} = R_h^{2h} r_h^{(i)}$
5. approximate error on coarse grid: $e_h^{(i)} \approx P_{2h}^h e_{2h}^{(i)}$
   $\rightarrow$ leads to Galerkin coarsening $R_h^{2h} A_h P_{2h}^h e_{2h}^{(i)} = R_h^{2h} r_h^{(i)}$
6. compute error $e_{2h}^{(i)}$ from $(R_h^{2h} A_h P_{2h}^h) e_{2h}^{(i)} = A_{2h} e_{2h}^{(i)} = b_{2h} := R_h^{2h} r_h^{(i)}$
7. interpolate coarse-grid error, $e_h^{(i)} \approx P_{2h}^h e_{2h}^{(i)}$,
   and apply as coarse-grid correction: $x_h^{(i+1)} = x_h^{(i)} + P_{2h}^h e_{2h}^{(i)}$
8. post-smoothing on approximate solution $x_h^{(i+1)}$

Galerkin Coarse Grid Operator: $A_{2h} := R_h^{2h} A_h P_{2h}^h$

- assume linear Interpolation and full weighting restriction:

$$R_h^{2h} = \begin{pmatrix} \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \end{pmatrix} \qquad P_{2h}^h = (R_h^{2h})^T$$

- for Poisson equation with stencil $\frac{1}{h^2}[-1 \quad 2 \quad -1]$ (see exercises)
  $\rightarrow$ coarse-grid stencil reproduces: $\frac{1}{(2h)^2}[-1 \quad 2 \quad -1]$
- for pure convection and central differencing, stencil $\frac{1}{h}[1 \quad 0 \quad -1]$,
  $\rightarrow$ coarse-grid stencil reproduces: $\frac{1}{2h}[1 \quad 0 \quad -1]$ (check!)
  $\rightarrow$ but leads to unstable discretisation
- question: result for upwind discretisation: stencil $\frac{1}{h}[-1 \quad 1 \quad 0]$?

Galerkin Coarsening and Convection

- coarsening with linear Interpolation, full weighting restriction, and upwind stencil $\frac{1}{h}[-1 \quad 1 \quad 0]$
- leads to coarse-grid stencil $\frac{1}{h}\left[-\frac{3}{4} \quad \frac{1}{2} \quad \frac{1}{4}\right]$
  $\rightarrow$ not a diagonal-dominant matrix
  $\rightarrow$ unstable discretisation
- remedy: use “downwind” interpolation and “upwind” restriction:

$$R_h^{2h} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \qquad P_{2h}^h = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

- result: upwind discretisation stencil reproduced: $\frac{1}{2h}[-1 \quad 1 \quad 0]$!

Matrix-Dependent Interpolation/Restriction

- assume general 1D 3-point discretisation stencil: $[s_l \quad s_c \quad s_r]$ where $s_c = -(s_l + s_r) > 0$
- use the following 3-point interpolation stencil: $\left]-\frac{s_l}{s_c} \quad 1 \quad -\frac{s_r}{s_c}\right[$ and $P_{2h}^h = (R_h^{2h})^T$, thus:

$$R_h^{2h} = \begin{pmatrix} -\frac{s_l}{s_c} & 1 & -\frac{s_r}{s_c} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{s_l}{s_c} & 1 & -\frac{s_r}{s_c} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{s_l}{s_c} & 1 & -\frac{s_r}{s_c} \end{pmatrix}$$

- Galerkin coarsening $A_{2h} := R_h^{2h} A_h P_{2h}^h$ leads to stencil

$$\frac{1}{s_c}\left[-s_l^2 \quad s_l^2 + s_r^2 \quad -s_r^2\right] \qquad \rightarrow \textbf{check this!}$$

- remains diagonal dominant $\rightarrow$ stable coarse-grid discretisation
- try as exercise:
  compute coarse-grid operator for convection-diffusion equation;
  compare operator- and matrix-dependent interpolation/restriction

Matrix-Dependent Interpolation/Restriction (2)

- take a closer look at the matrix multiplication $R_h^{2h} A_h$:

$$\begin{pmatrix} \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ \cdots & 0 & -\frac{s_l}{s_c} & 1 & -\frac{s_r}{s_c} & 0 & \cdots \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \ddots & \ddots & \ddots & & & & \\ 0 & s_l & s_c & s_r & 0 & \cdots & \\ \cdots & 0 & s_l & s_c & s_r & 0 & \cdots \\ & \cdots & 0 & s_l & s_c & s_r & 0 \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}$$

- **equivalent to performing row operations as in Gaussian elimination:**

$$\begin{pmatrix} \ddots & \ddots & \ddots & & & & \\ 0 & s_l & s_c & s_r & 0 & \cdots & \\ \cdots & 0 & s_l & s_c & s_r & 0 & \cdots \\ & \cdots & 0 & s_l & s_c & s_r & 0 \\ & & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{matrix} \\ \cdot(-\frac{s_l}{s_c}) \\ \cdot 1 \\ \cdot(-\frac{s_r}{s_c}) \\ \\ \end{matrix}$$

- red entries become 0 $\Rightarrow$ coarse-grid unknown no longer depends on fine-grid unknowns
- similar for multiplication $A_h P_{2h}^h$, but with column operations

Smoothers

Efficient smoothers for the Poisson problem (see tutorials):

- Gauss-Seidel
- red-black Gauss-Seidel
- damped ($\omega = \frac{2}{3}$) Jacobi
  $\rightarrow$ why $\omega = \frac{2}{3}$?
  $\rightarrow$ examine smoothing factors (dependent on wave number k)

How about ...

- Jacobi (non-weighted)?
  $\rightarrow$ does not work (zig-zag pattern prevents smoothing)
- SOR?
  $\rightarrow$ typically does not work well for Poisson model problem
  (does not smooth high frequencies efficiently)
  $\rightarrow$ can help for other problems using a tailored ω

Smoothers – Anisotropic Problems

Model problems and observation:

- anisotropic Poisson equation: $u_{xx} + \epsilon u_{yy} = f$ with $\epsilon \ll 1$
  (or similar: $\epsilon \gg 1$)
- Poisson equation on stretched grids:

$$\frac{1}{h_x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + \frac{1}{h_y^2}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) = f_{ij}$$

  with $h_x \ll h_y$ (or similar: $h_x \gg h_y$)
- Strong dependency in x-direction, weak dependency in y-direction
  (or vice versa)
- Good smoothing of the error only in x-direction
  (or in y-direction):

$$u_{i,j} = \frac{1}{2h_x^2 + 2h_y^2}\left(-h_x^2 h_y^2 f_{ij} + h_y^2(u_{i+1,j} + u_{i-1,j}) - h_x^2(u_{i,j+1} + u_{i,j-1})\right)$$

Smoothers – Anisotropic Problems (2)

Semi-Coarsening:

- situation on coarser grid for example $H_x = 2h_x$, $H_y = h_y$:

$$\frac{1}{4h_x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + \frac{1}{h_y^2}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) = f_{ij}$$

- thus: anisotropy has weakened on the semi-coarsened grid

Line Smoothers:

- perform a column-wise (or row-wise) Jacobi/Gauss-Seidel relaxation
  $\rightarrow$ solve each column (or row) simultaneously:

$$u_{i-1,j}^{(n+1)} - (2 + 2\epsilon)u_{ij}^{(n+1)} + u_{i+1,j}^{(n+1)} = h^2 f_{ij} - \epsilon(u_{i,j-1}^{(n)} + u_{i,j+1}^{(n)})$$

- use direct, tridiagonal solver for each “line” (i.e., row or column)

Smoothers – Convection-Diffusion

- example: 1D Convection-Diffusion equation

$$-\epsilon u_{xx} + u_x = f, \qquad \epsilon \ll 1$$

- "upwind discretisaton":

$$-\frac{\epsilon}{h^2}(u_{n-1} - 2u_n + u_{n+1}) + \frac{1}{h}(u_n - u_{n-1}) = f_n$$

- (weighted) Jacobi and red-black Gauss-Seidel?
  $\rightarrow$ no smoothing, basically updates one grid point per iteration
- Gauss-Seidel (relaxation from "left to right")?
  $\rightarrow$ almost an exact solver

Smoothers – More Complicated Situations

Problems:

- anisotropic Poisson with space-dependent $\epsilon = \epsilon(x, y)$,
  or more general:
  $$-\nabla\big(D(x, y)\nabla u(x, y)\big) = f(x, y)$$
- convection-diffusion with variable convection:

$$-\epsilon u_{xx} + v(x)u_x = f \qquad -\epsilon\Delta u + v(x, y)\nabla u(x, y) = f(x, y)$$

Approaches for Smoothing:

- alternating line smoothers, “plane smoothers” in 3D
- “Zebra” line smoothers (similar to red-black-GS)
- Gauss-Seidel smoothing in “downwind” order
  $\rightarrow$ difficult to do for complicated flows in 2D and 3D

Part III

Finite Elements, Hierarchical Bases, and Multilevel Methods

FEM Main Ingredients
- Test and Shape Functions
- 1D Poisson Equation and Nodal Basis

Hierarchical Bases and Generating Systems
- Hierarchical Basis
- FEM with Hierarchical vs. Nodal Bases
- Hierarchical Basis Transformation
- FEM and Hierarchical Basis Transform

Generating Systems and Multigrid
- Hierarchical Generating System and FEM

Remember: Finite Elements – Main Ingredients

1. solve *weak form* of PDE to reduce regularity properties

$$u'' = f \quad \longrightarrow \quad \int v' u' \, \mathrm{dx} = \int v f \, \mathrm{dx}$$

   $\rightarrow$ allows additional *weak* solutions

2. compute a *function* as numerical solution
   $\rightarrow$ search in a function space W_h:

$$u_h = \sum_j u_j \varphi_j(x), \qquad \mathrm{span}\{\varphi_1, \ldots, \varphi_J\} = W_h$$

3. find weak solutions of simple form:
   for example piecewise linear functions and choose basis functions with *local support* (“hat functions”)
   $\rightarrow$ leads to system of linear equations

Test and Shape Functions

- consider a general PDE $Lu = f$ on some domain Ω
- search for solution functions u_h of the form

$$u_h = \sum_j u_j \varphi_j(x)$$

  the $\varphi_j(x)$ are typically called **shape** or **ansatz** functions
- the basis functions $\varphi_j(x)$ build a vector space (i.e., a **function space**) W_h

$$\text{span}\{\varphi_1, \ldots, \varphi_J\} = W_h$$

- insert into weak formulation

$$\int vL\left(\sum_j u_j \varphi_j(x)\right) \, \mathrm{dx} = \int vf \, \mathrm{dx} \quad \forall v \in V$$

Test and Shape Functions (2)

- choose a basis $\{\psi_i\}$ of the *test* space V_h
  typically defined on some discretisation grid Ω_h
- then: if all basis functions ψ_i satisfy

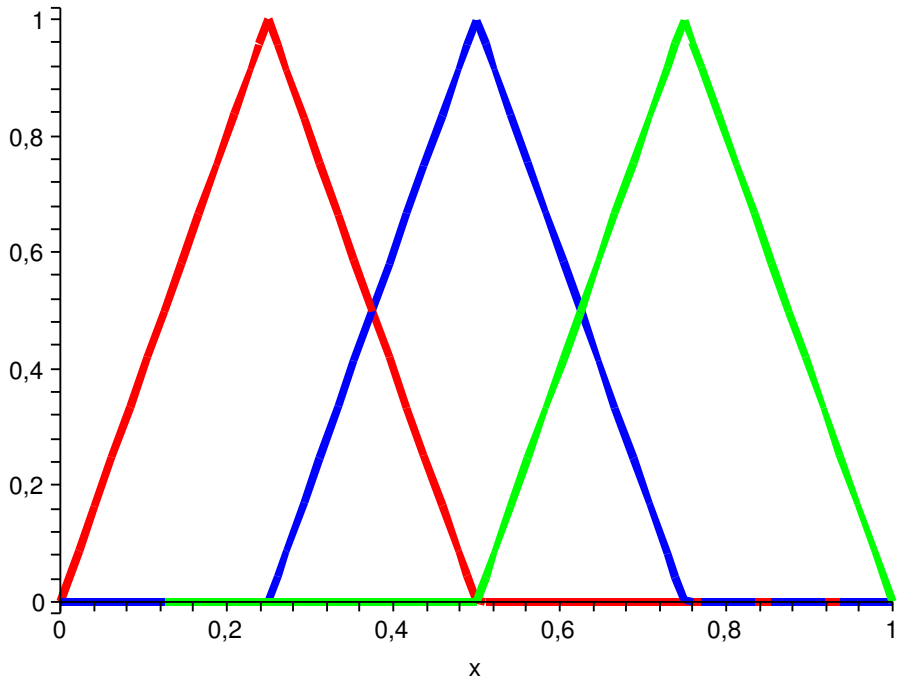
$$\int \psi_i(x) L\left(\sum_j u_j \varphi_j(x)\right) \,\mathrm{dx} = \int \psi_i(x) f(x) \,\mathrm{dx} \quad \forall \psi_i$$

  then all $v \in V_h$ satisfy the equation
- the $\{\psi_i\}$ are therefore often called **test functions**
- we obtain a system of equations for unknowns u_j:
  one equation for each test function ψ_i
- V_h is often chosen to be identical to W_h (**Ritz-Galerkin method**)
  $\rightarrow$ we then have as many equations as unknowns
- leads to system of linear equations $Au = b$ where

$$(Au)_i := \sum_j u_j \underbrace{\int \psi_i(x) L\varphi_j(x) \,\mathrm{dx}}_{=A_{ij}} = \int \psi_i(x) f(x) \,\mathrm{dx} =: b_i$$

Example: Nodal Basis

$$\varphi_i(x) := \begin{cases} \frac{1}{h}(x - x_{i-1}) & x_{i-1} < x < x_i \\ \frac{1}{h}(x_{i+1} - x) & x_i < x < x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

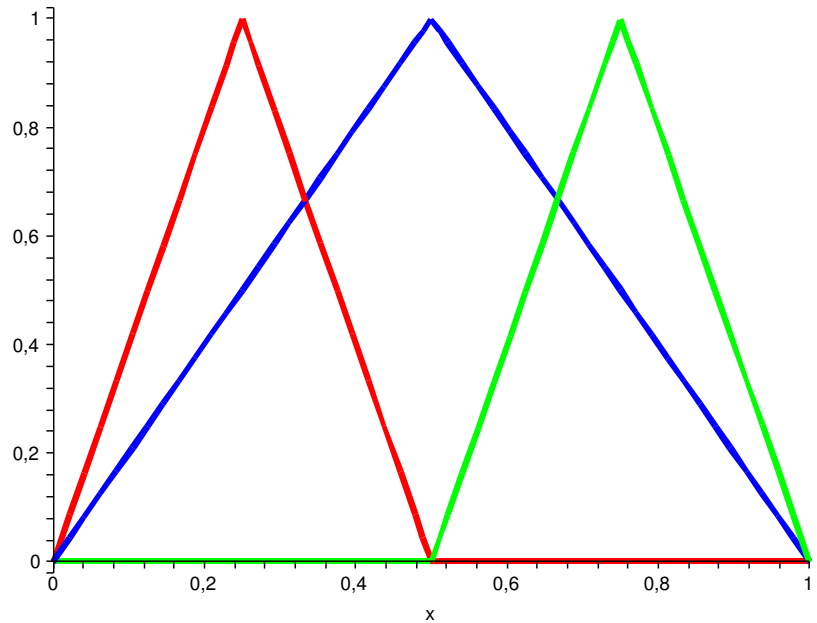


Example Problem: 1D Poisson

- in 1D: $-u''(x) = f(x)$ on $\Omega = (0,1)$,
  hom. Dirichlet boundary cond.: $u(0) = u(1) = 0$
- weak form:
$$\int_0^1 v'(x) \cdot u'(x)\ \mathrm{dx} = \int_0^1 v(x) f(x)\ \mathrm{dx} \quad \forall v$$
- grid points $x_i = ih$, (for $i = 1, \ldots, n-1$); mesh size $h = 1/n$
- $V_h = W_h$: piecewise linear functions (on intervals $[x_i, x_{i+1}]$)
- leads to stiffness matrix:
$$\frac{1}{h} \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$

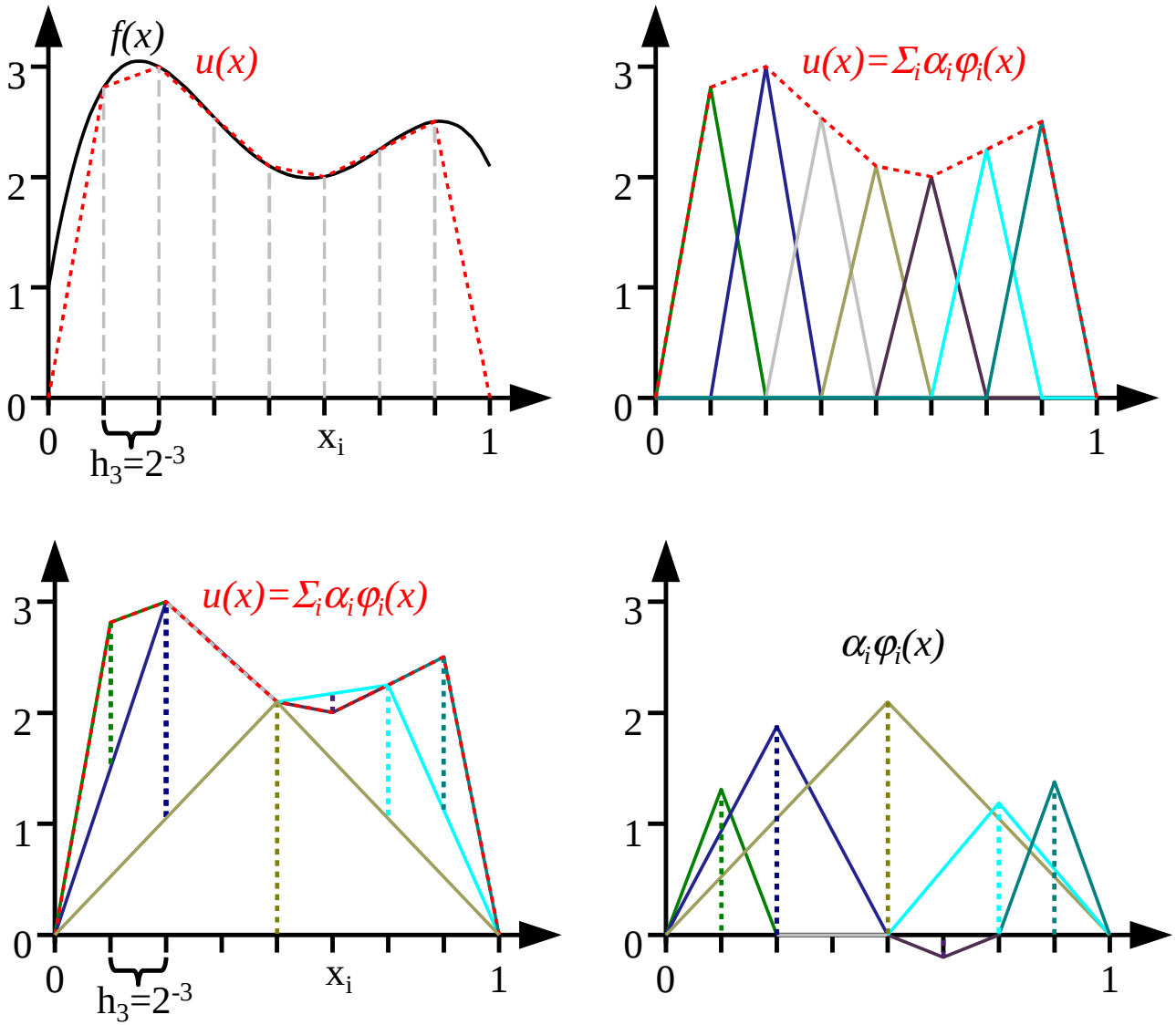
Hierarchical Basis

- hat functions with multi-level resolution



- FEM solution identical for hierarchical and nodal basis (same function space!)
- known from Scientific Computig I: **diagonal stiffness matrix** for 1D Poisson!

Hierarchical vs. Nodal Basis



Hierarchical Basis and Multigrid

- What happens, if we use FEM on hat function bases with different resolutions?
- Define “mother of all hat functions”

$$\phi(x) := \max\{1 - |x|, 0\}$$

- consider mesh size $h_n = 2^{-n}$ and grid points $x_{n,i} = i \cdot h_n$
- nodal basis then $\Phi_n := \{\phi_{n,i}, 0 \leq i \leq 2^n\}$ with

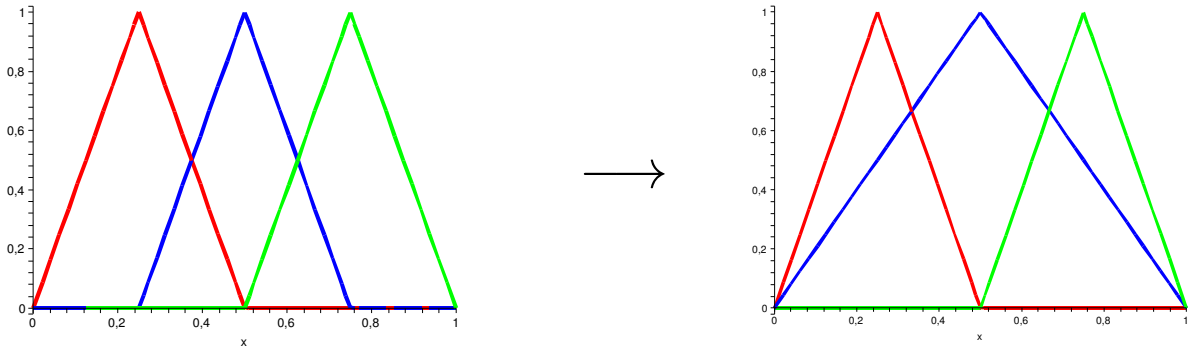
$$\phi_{n,i}(x) := \phi\left(\frac{x - x_{n,i}}{h_n}\right)$$

- hierarchical basis combines $\widehat{\Phi}_n := \{\phi_{n,i}, i = 1, 3, \ldots, 2^n - 1\}$ (only odd indices) and defines basis as

$$\Psi_n := \bigcup_{l=1}^{n} \widehat{\Phi}_l$$

Hierarchical Basis Transformation

(or: How to represent functions on a coarser grid?)



- represent hat functions $\phi_{n-1,i}(x)$ via fine-level functions $\phi_{n,j}(x)$

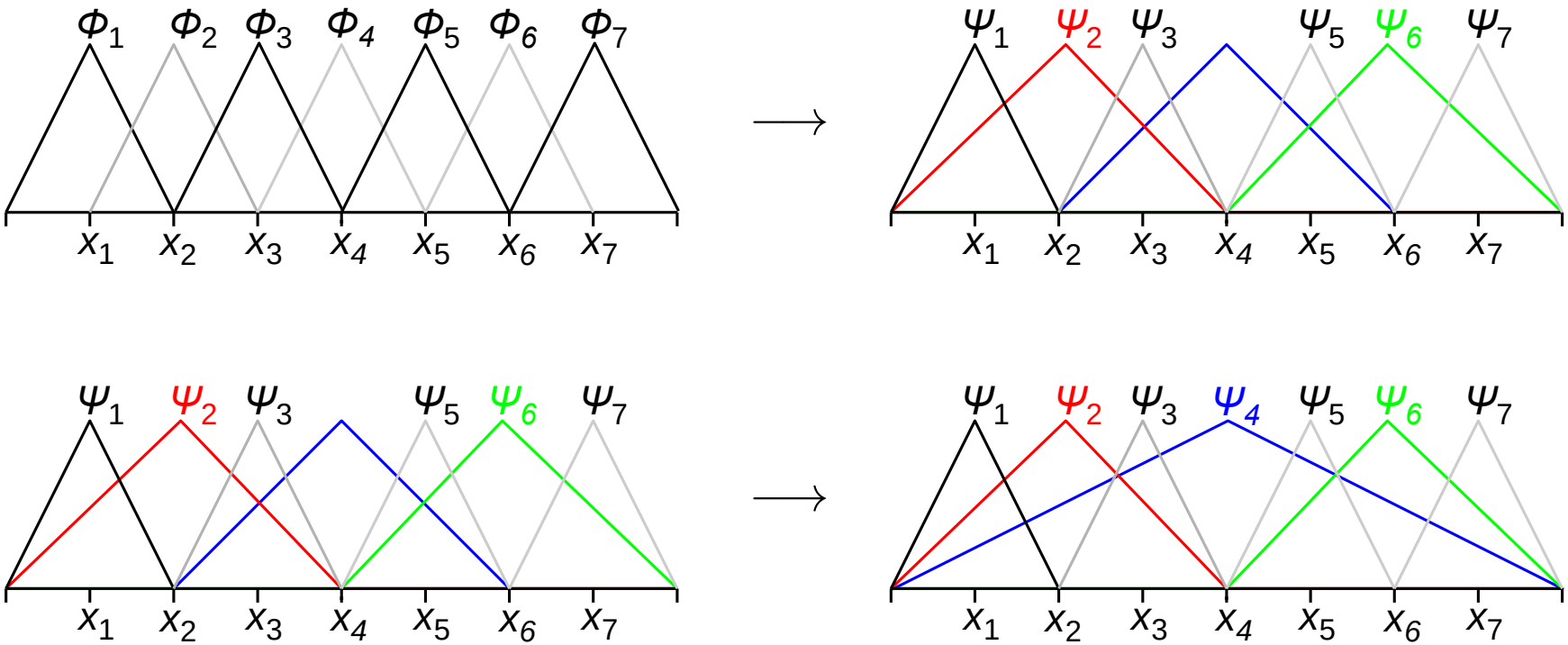
$$\phi_{n-1,i}(x) = \tfrac{1}{2}\phi_{n,2i-1}(x) + \phi_{n,2i}(x) + \tfrac{1}{2}\phi_{n,2i+1}(x)$$

- hierarchical-basis transformation as matrix-vector product:

$$\begin{pmatrix} \psi_{n,i-1}(x) \\ \psi_{n,i}(x) \\ \psi_{n,i+1}(x) \end{pmatrix} := \begin{pmatrix} \phi_{n,2i-1}(x) \\ \phi_{n-1,i}(x) \\ \phi_{n,2i+1}(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \phi_{n,2i-1}(x) \\ \phi_{n,2i}(x) \\ \phi_{n,2i+1}(x) \end{pmatrix}$$

Hierarchical Basis Transformation (2)

Level-by-level algorithm for hierarchical transform:



Remark: allows to implement transform in $\mathcal{O}(N)$

Hierarchical Basis Transformation (3)

- hierarchical basis transformation: $\psi_{n,i}(x) = \sum_j H_{i,j}\phi_{n,j}(x)$
- transform can be written as matrix-vector product: $\vec{\psi}_n = H_n\vec{\phi}_n$
- step-wise transform from each grid level to the next similar to

$$H_3^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

cmp. restriction operator!

- H_n then a sequence of level-to-next-level transforms:

$$H_n = H_n^{(1)}\, H_n^{(2)}\, \ldots\, H_n^{(n-2)}\, H_n^{(n-1)}$$

Hierarchical Coordinate Transformation

- consider function $f(x) \approx \sum_i a_i \psi_{n,i}(x)$ represented via hier. basis
- wanted: corresponding representation in nodal basis

$$\sum_k b_k \phi_{n,k}(x) = \sum_i a_i \psi_{n,i}(x) \approx f(x)$$

- with $\psi_{n,i}(x) = \sum_j H_{i,j} \phi_{n,j}(x)$ we obtain

$$\sum_k b_k \phi_{n,k}(x) = \sum_i a_i \sum_j H_{i,j} \phi_{n,j}(x) = \sum_j \sum_i a_i H_{i,j} \phi_{n,j}(x)$$

- compare coordinates (identify indices j and k) and get

$$b_j = \sum_i H_{i,j} a_i = \sum_i \left(H^T\right)_{j,i} a_i$$

- written in vector notation: $b = H^T a$

FEM and Hierarchical Basis Transform

- FEM discretisation with hierarchical test and shape functions:

$$\int \psi_i(x) L\left(\sum_j u_j \psi_j(x)\right) \,\mathrm{dx} = \int \psi_i(x) f(x) \,\mathrm{dx} \quad \forall \psi_i$$

- leads to respective stiffness matrix $A^{\mathrm{HB}}_{i,j}$:

$$\int \psi_i(x) L\left(\sum_j u_j \psi_j(x)\right) \,\mathrm{dx} = \sum_j u_j \int \psi_i(x) L\psi_j(x) \,\mathrm{dx} = \sum_j u_j A^{\mathrm{HB}}_{i,j}$$

- vs. stiffness matrix with nodal basis as shape functions:

$$\int \psi_i(x) L\left(\sum_j v_j \phi_j(x)\right) \,\mathrm{dx} = \sum_j v_j \int \psi_i(x) L\phi_j(x) \,\mathrm{dx} = \sum_j v_j A^{*}_{i,j}$$

- Note: $\sum_j u_j A^{\mathrm{HB}}_{i,j}$ and $\sum_j v_j A^{*}_{i,j}$ are both equal to $\int \psi_i(x) f(x) \,\mathrm{dx}$
- Therefore: $(A^{\mathrm{HB}} u)_i = \sum_j u_j A^{\mathrm{HB}}_{i,j} = \sum_j v_j A^{*}_{i,j} = (A^{*} v)_i$ and $v = H^T u$

FEM and Hierarchical Basis Transform (2)

- status: FEM with hierarchical test and nodal shape functions

$$\int \psi_i(x) L\left(\sum_j v_j \phi_j(x)\right) \,\mathrm{dx} = \int \psi_i(x) f(x) \,\mathrm{dx}$$

- represent test functions via nodal basis:

$$\int \sum_k H_{i,k} \phi_k(x) L\left(\sum_j v_j \phi_j(x)\right) \,\mathrm{dx} = \int \sum_k H_{i,k} \phi_k(x) f(x) \,\mathrm{dx}$$

$$\sum_k H_{i,k} \int \phi_k(x) L\left(\sum_j v_j \phi_j(x)\right) \,\mathrm{dx} = \sum_k H_{i,k} \int \phi_k(x) f(x) \,\mathrm{dx}$$

- leads to new system of equations: $HA^{\mathrm{NB}}\, v = H\, b^{\mathrm{NB}}$
  where A^{NB} and b^{NB} stem from nodal-basis FEM discretisation!
- with $v = H^T u$ we obtain $H\, A^{\mathrm{NB}}\, H^T u = H\, b$ as system of equations, thus: $A^{\mathrm{HB}} = H\, A^{\mathrm{NB}}\, H^T$ ($\rightsquigarrow$ **Galerkin coarsening**)

FEM and Hierarchical Basis Transform – Summary

- in general: FEM with nodal test and shape functions

$$\int \phi_i(x) L\left(\sum_j v_j \phi_j(x)\right) \,\mathrm{dx} = \int \phi_i(x) f(x) \,\mathrm{dx} \qquad \rightsquigarrow A^{\mathrm{NB}} u^{\mathrm{NB}} = b^{\mathrm{NB}}$$

  changed to different test and shape functions:

$$\int \psi_i(x) L\left(\sum_j v_j \psi_j(x)\right) \,\mathrm{dx} = \int \psi_i(x) f(x) \,\mathrm{dx} \qquad \rightsquigarrow A^{??} u^{??} = b^{??}$$

- change from nodal to hierarchical basis:

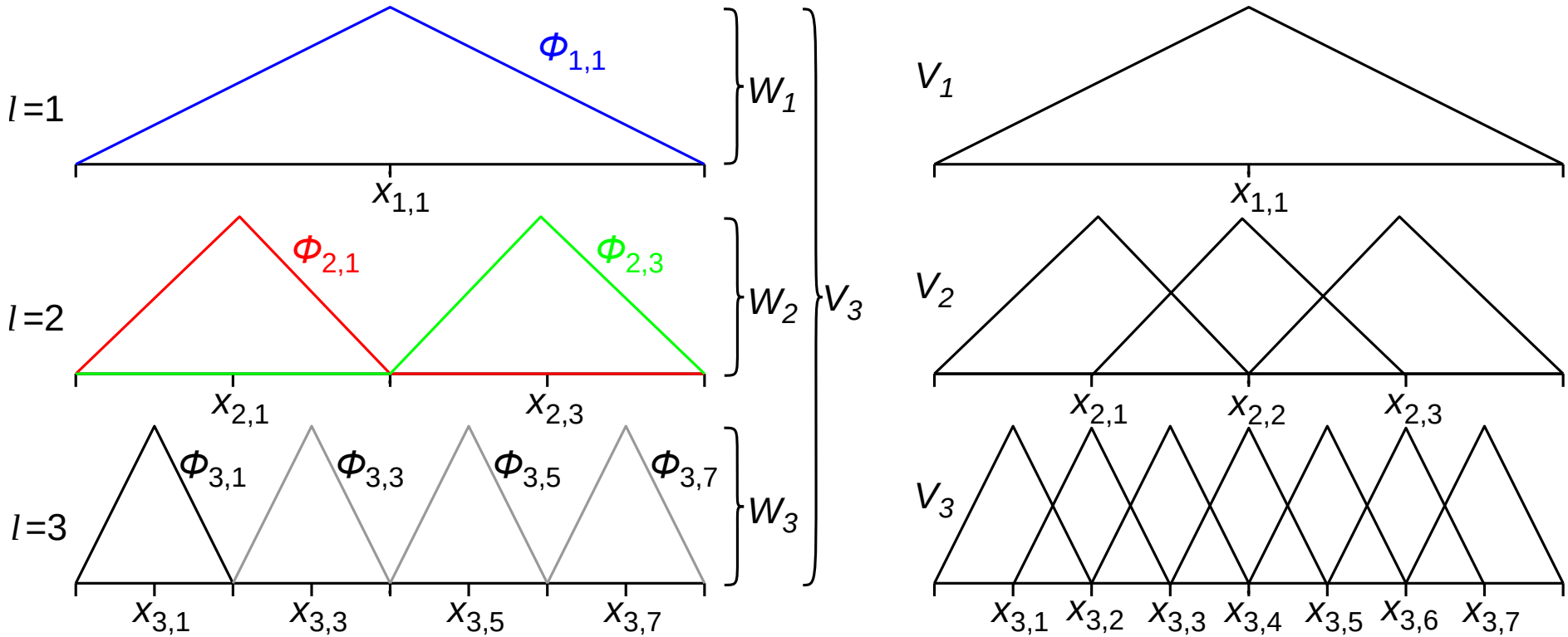
  $A^{\mathrm{HB}} u^{\mathrm{HB}} = H A^{\mathrm{NB}} H^T u^{\mathrm{HB}} = H b^{\mathrm{NB}} = b^{\mathrm{HB}}$

- change from nodal to coarser-level nodal basis:

  $A^{\mathrm{C}} u^{\mathrm{C}} = R A^{\mathrm{NB}} R^T u^{\mathrm{C}} = R b^{\mathrm{NB}} = R^C$

- **Note:** R, R^T as in Galerkin coarsening; strongly related to H, H^T

Hierarchical Generating System



- hierarchical basis combines $\widehat{\Phi}_n := \{\phi_{n,i}, i = 1, 3, \ldots, 2^n - 1\}$ (only odd indices)
- generating system combines nodal bases of all levels: $\bigcup_{l=1}^{n} \Phi_l$

FEM and Hierarchical Generating Systems

- solution function u_h represented as

$$u_h = \sum_{l=1}^{n} \sum_{j=1}^{2^l-1} v_{l,j} \phi_{l,j}(x)$$

  $\rightarrow$ non-unique “multi-grid” representation!
- FEM discretisation with test and shape functions from hierarchical generating system:

$$\int \phi_{k,i}(x) L\left(\sum_l \sum_j v_{l,j} \phi_{l,j}(x) \right) \, \mathrm{dx} = \int \phi_{k,i}(x) f(x) \, \mathrm{dx} \quad \forall \phi_{k,i} \in \bigcup_{k=1}^{n} \Phi_k$$

- assume order of test/shape functions:

$$\phi_{n,1}, \ldots, \phi_{n,2^n-1}, \; \phi_{n-1,1}, \ldots, \phi_{n-1,2^{n-1}-1}, \ldots, \phi_{2,1}, \phi_{2,2}, \phi_{2,3}, \; \phi_{1,1}$$

- leads to (linearly dependent!) system of linear equations:

$$A^{\mathrm{GS}} u_h = b^{\mathrm{GS}}$$

Multi-Level System of Equations

- system matrix A^{GS} given as (for example)

$$\begin{pmatrix} A_h & A_h^{2h} & A_h^{4h} \\ A_{2h}^h & A_{2h} & A_{2h}^{4h} \\ A_{4h}^h & A_{4h}^{2h} & A_{4h} \end{pmatrix}$$

- A_h, A_{2h}, and A_{4h} are the nodal-basis stiffness matrices for resolution h, $2h$, and $4h$
- consider submatrix $A_{2h}^h \rightarrow$ computed as

$$\int \phi_{2h,i}(x) L\left(\sum_j v_{h,j}\phi_{h,j}(x)\right) \,\mathrm{dx} = \int \phi_{2h,i}(x) f(x) \,\mathrm{dx} \quad \forall \phi_{2h,i} \in \Phi_{2h}$$

- test functions $\phi_{2h,i}(x) = \frac{1}{2}\phi_{h,2i-1}(x) + \phi_{h,2i}(x) + \frac{1}{2}\phi_{h,2i+1}(x)$
- not a hierarchical transform, but a **restriction** operation R_h^{2h}:

$$\vec{\phi}_{2h} = R_h^{2h}\vec{\phi}_h \quad \text{thus:} \quad A_{2h}^h = R_h^{2h} A_h$$

Multi-Level System of Equations (2)

- system of linear equations $A^{GS} v^{GS} = b^{GS}$ thus given as

$$\begin{pmatrix} A_h & A_h P_{2h}^h & A_h P_{4h}^h \\ R_h^{2h} A_h & A_{2h} & A_{2h} P_{4h}^{2h} \\ R_h^{4h} A_h & R_{2h}^{4h} A_{2h} & A_{4h} \end{pmatrix} \begin{pmatrix} v_h \\ v_{2h} \\ v_{4h} \end{pmatrix} = \begin{pmatrix} b_h \\ R_h^{2h} b_h \\ R_h^{4h} b_h \end{pmatrix}$$

  with restriction and prolongation operators: $R = P^T$
- matrix is singular(!) due to obviously linearly dependent rows
- however: all solutions lead to same piecewise linear function
- result of a relaxation method on this system of equations?

  symmetric Gauss-Seidel sweep corresponds to one multigrid V-Cycle!

  (Griebel, 1994)

Symmetric Gauss-Seidel on $A^{GS}v^{GS} = b^{GS}$

- pick out second block row of the system $A^{GS}v^{GS} = b^{GS}$:

$$R_h^{2h} A_h v_h + A_{2h} v_{2h} + A_{2h} P_{4h}^{2h} v_{4h} = R_h^{2h} b_h$$

- Gauss-Seidel $\rightarrow$ thus solve for unknowns in v_{2h}:

$$A_{2h} v_{2h} + A_{2h} P_{4h}^{2h} v_{4h}^{(\text{old})} = R_h^{2h} b_h - R_h^{2h} A_h v_h^{(\text{new})} = R_h^{2h} \left(b_h - A_h v_h^{(\text{new})} \right)$$

- observation #1: relaxation works on restricted fine-grid residual $R_h^{2h} \left(b_h - A_h v_h^{(\text{new})} \right)$
- observation #2: relaxation considers prolongated coarse-grid correction from previous iteration: $A_{2h} P_{4h}^{2h} v_{4h}^{(\text{old})}$

$\Rightarrow$ **FEM on hierarchical generating systems matches V-Cycle Multigrid with Galerkin coarsening**

Literature/References – Multigrid

- Briggs, Henson, McCormick: *A Multigrid Tutorial* (2nd ed.), SIAM, 2000.
- Trottenberg, Oosterlee, Schüller: *Multigrid*, Elsevier, 2001.
- Shapira: *Matrix-Based Multigrid: Theory and Applications*, Springer, 2008.
- Hackbusch: *Iterative Solution of Large Sparse Systems of Equations*, Springer 1993.
- Brandt, Livne: *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, Revised Edition, SIAM
- M. Griebel: *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen*, Teubner Skripten zur Numerik, 1994

  M. Griebel: *Multilevel algorithms considered as iterative methods on semidefinite systems*, SIAM Int. J. Sci. Stat. Comput. 15(3), 1994.