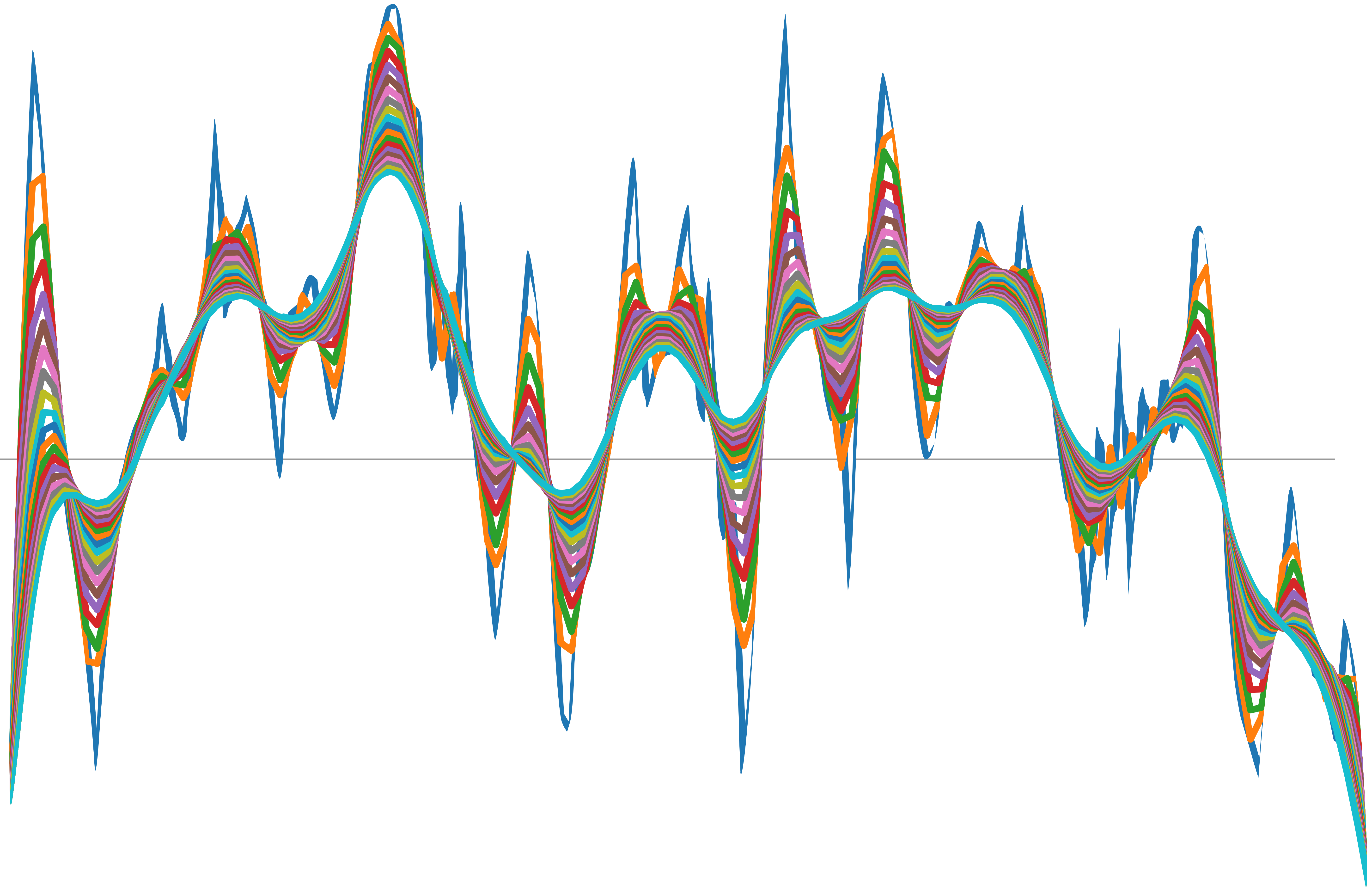
Multigrid Methods

The Basics



2023 Copper Multigrid Conference
April 16, 2023

Hans De Sterck
Department of Applied Mathematics,
University of Waterloo

(slides by Luke Olson, Department of Computer Science, University of Illinois at Urbana-Champaign)

+ Notebooks 6, 7, 8, 9 , slides

What are we trying to do…

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

- Solve problems of the form

$$Ax = b$$

GE : $W = O(n^3)$

MG : $W = O(n)$

- Solve this problem **iteratively**:

$$x_1 \leftarrow x_0 + v$$

- Solve this problem **inexpensively:**
  - The update should be “good” (small number of iterations)
  - Finding the update should be “cheap”

Objectives — high level

- Construct a multigrid method for **your** problem
- Interpret the effectiveness of a multigrid method
- Identify *why* a method works — or — *why* a method does not
- Recognize different forms of multigrid, their pitfalls and their uses

Objectives — this session

- **Create** a two level multigrid method
- **Illustrate** the main components of a multigrid method
- **Calculate** the effectiveness of a multigrid method
- **Highlight** some limitations of multigrid
- **Outline** some approaches to fixing multigrid (and why)
- **Identify** key pieces in moving to unstructured problems

A cautionary example (see slide 3)

Guess and look for and update $x_1 = x_0 + \text{update}$

Updating with the error would be **ideal** $x_1 = x_0 + e_0 = x_0 + (x^* - x_0) = x^*$

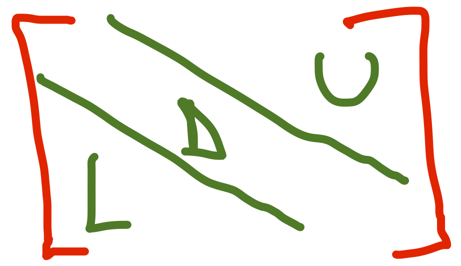
Or in another form $x_1 = x_0 + A^{-1} r_0 = x^*$

1 iteration

Not practical so… $x_1 = x_0 + D^{-1} r_0$

$\boxed{\text{Jacobi} \sim 1\ \text{SpMV}}$ ($W = O(n)$ per iteration)

note: $A = D + L + U$



x^* solution to $Ax = b$

$e_0 = x^* - x_0$ error

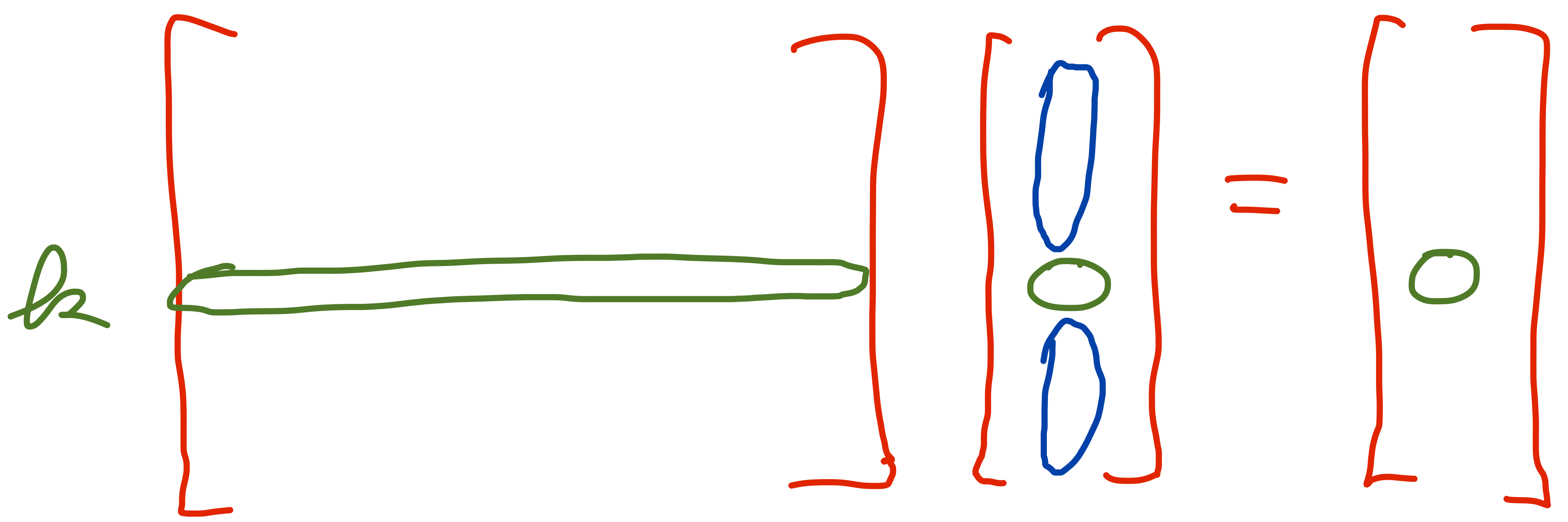
$r_0 = b - Ax_0$ residual

$= Ae_0 = A(x^* - x_0)$

$Ae_0 = r_0$ error equation

Jacobi iteration:

$$A\vec{x} = \vec{b}$$



eq. k: $\sum_j a_{kj} x_j = b_j$

Jacobi: $$x_k^{new} = \frac{1}{a_{kk}} \left(b_k - \sum_{j \neq i} a_{kj} x_j^{old} \right)$$

$$\vec{x}_1 = D^{-1} \left(\vec{b} - (L+U)\vec{x}_0 \right)$$

$$\vec{x}_1 = \vec{x}_0 + D^{-1} \left(\vec{b} - (D+L+U)\vec{x}_0 \right)$$

$$\vec{x}_1 = \vec{x}_0 + D^{-1} \left(\vec{b} - A\vec{x}_0 \right)$$

$$\vec{x}_1 = \vec{x}_0 + D^{-1} \vec{r}_0$$

A reminder — projection methods

- Take a guess

$$x_0$$

- Look for an update that is the "best":

$$x_1 \leftarrow x_0 + u$$

- Minimize over a smaller space

coarse grid

$$\min_{u\in span\{V\}} \|x^* - x_1\| = \|e_1\| = \|\underbrace{x^* - x_0}_{e_0} - u\|$$

$$= \|e_0 - Vy\|$$

- Then $u = Vy$

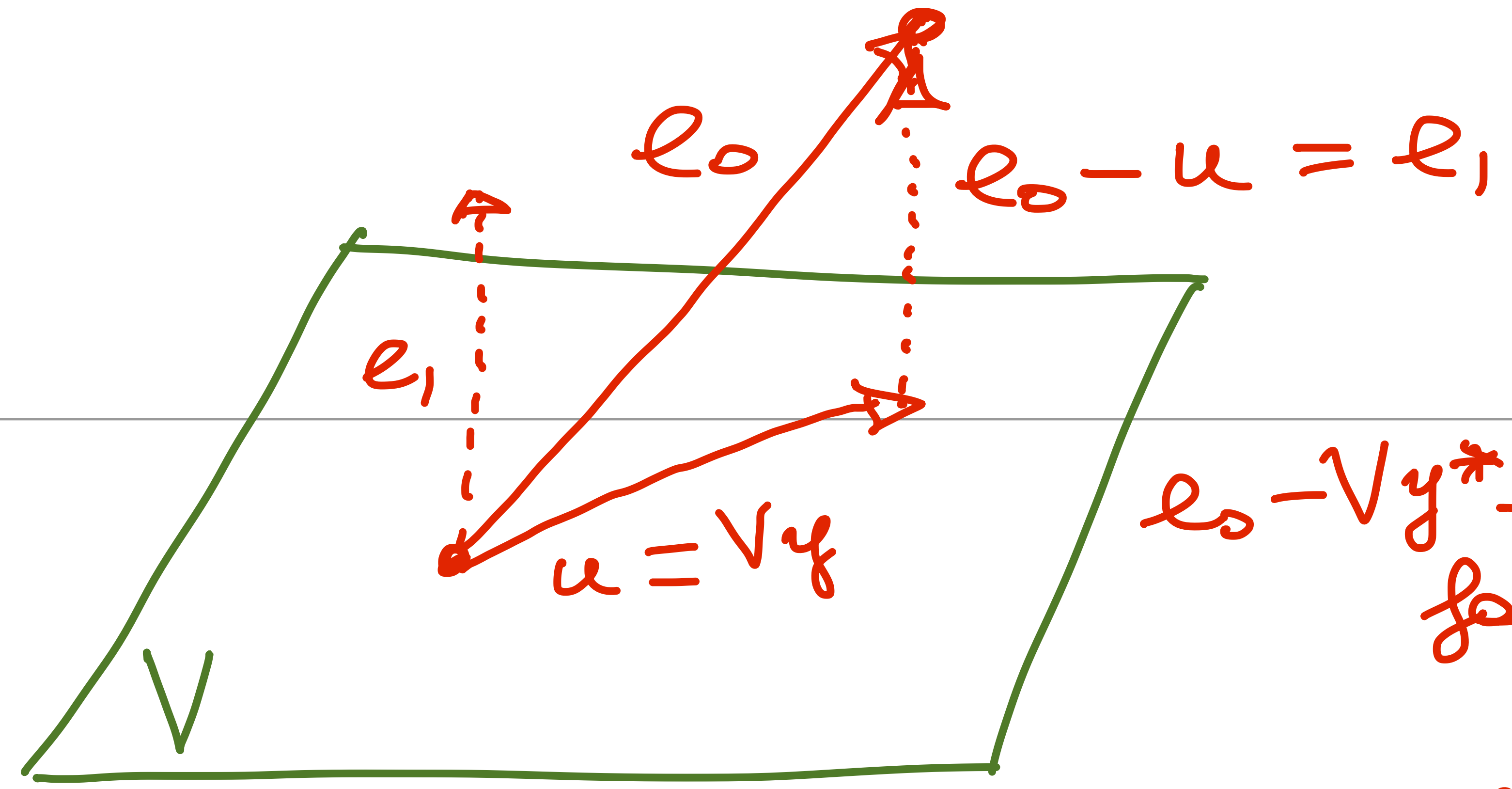
$$V^T V y = V^T e_0$$

normal equations

- So the update looks like

$$x_1 = x_0 + V\underbrace{(V^T V)^{-1} V^T e_0}_{y}$$

(coarse grid correction)



$e_0 - Vy^* \perp Vy$ for all y

$y^T V^T (e_0 - Vy^*) = 0$ for all y

→ optimal u = projection of e_0 on V, orthogonal

→ then remove u from e_0 to get e_1

A reminder — projection methods

- Instead, look at the A-norm (A s.p.d.):

$$\min_{u \in span\{V\}} \|x^* - x_1\|_A$$

- Then $u = Vy$

$$V^T A V y = V^T \underbrace{A e_0}$$

$$V^T A V y = V^T r_0$$

- So that

optimal coarse-grid operator

$$\underbrace{x_1}_{\text{A-optimal coarse-grid correction}} = x_0 + \underbrace{V \overbrace{(V^T A V)}^{}{}^{-1} V^T r_0}_{Vy}$$

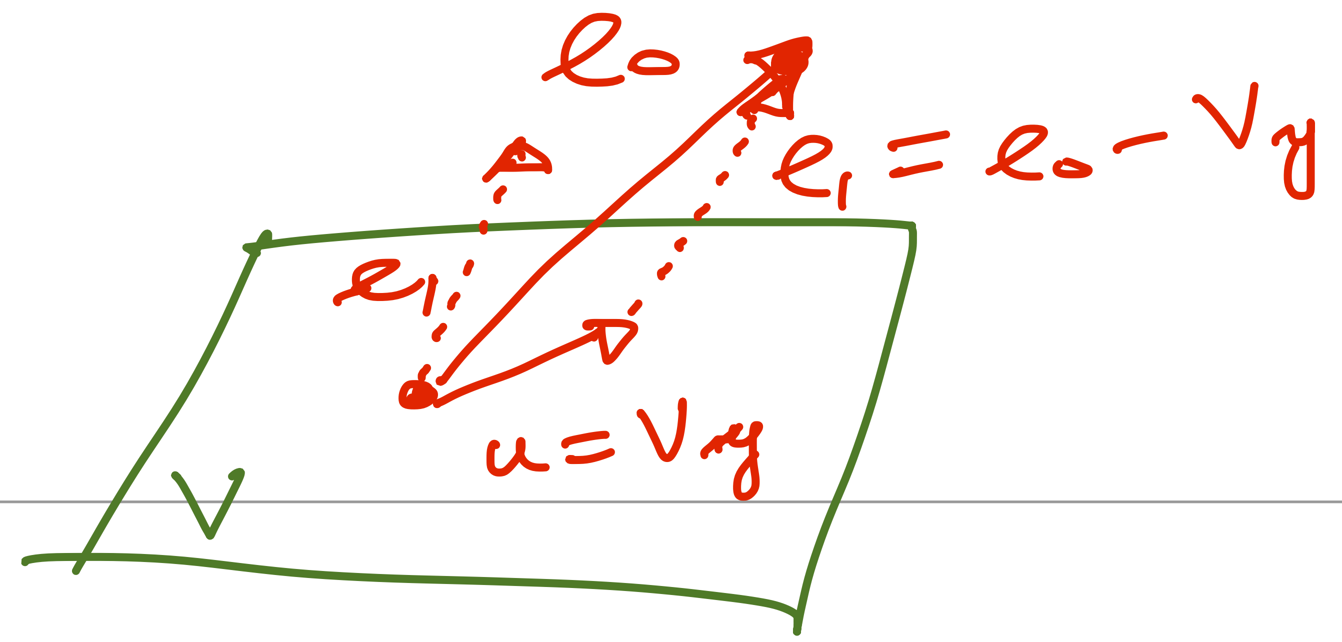
residual is cheaply computable $(O(n))$

A-orthogonal projection:

$$\min_u \|x^* - x_1\|_A = \|x^* - x_0 - u\|_A$$
$$= \|x^* - x_0 - Vy\|_A$$
$$= \|e_0 - Vy\|_A$$

$$\min_y (e_0 - Vy)^T A (e_0 - Vy)$$
$$= e_0^T A e_0 + (Vy)^T A Vy$$
$$- 2 e_0^T A Vy$$
$$\Rightarrow V^T A \underbrace{Vy}_{\text{A-orthogonal projection of } e_0} = V^T A e_0$$

A reminder — projection methods



$$x_1 = x_0 + u$$

$e_1 = e_0 - Vy^* \perp_A Vy$ for all y

$$x_1 = x_0 + V(V^TAV)^{-1}V^Tr_0$$

- What about the error

$$x^* - x_1 = x^* - x_0 - V(V^TAV)^{-1}V^Tr_0$$

$$e_1 = e_0 - V(V^TAV)^{-1}V^TAe_0$$

$$= \left(I - V(V^TAV)^{-1}V^TA\right)e_0$$

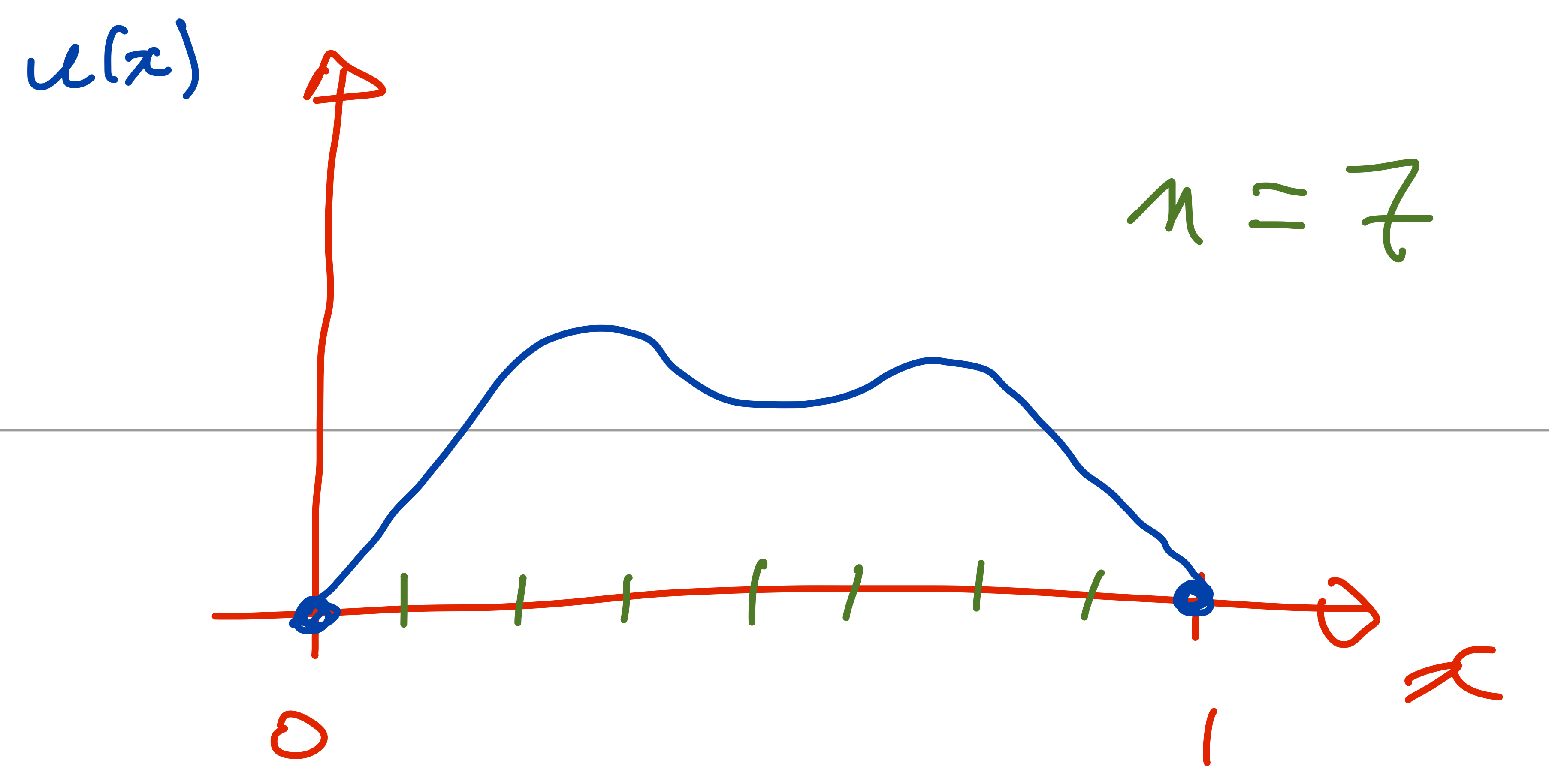
(optimal coarse-grid correction, given V)

A-orthogonal projection onto the range of V

Model Problem

- A model problem

$$-u_{xx} = f$$
$$u(0) = u(1) = 0$$



$n = 7$

$\underbrace{x_0, x_1 \qquad x_n,}_{} x_{n+1}$

n interior points

$h = \frac{1}{n+1} = \frac{1}{8}$

- Finite differences

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i \quad i = 1, \ldots, n \quad u_0 = u_{n+1} = 0$$

- A model matrix problem

$$A = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \end{bmatrix}$$

$A \in \mathbb{R}^{n \times n}$, sparse

$nnz(A) \approx 3n = O(n)$

$A \in \mathbb{R}^{n \times n}$ $h = \frac{1}{n+1}$

Model Problem

- A special matrix:

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & & \\ & & & -1 & 2 \end{bmatrix}$$

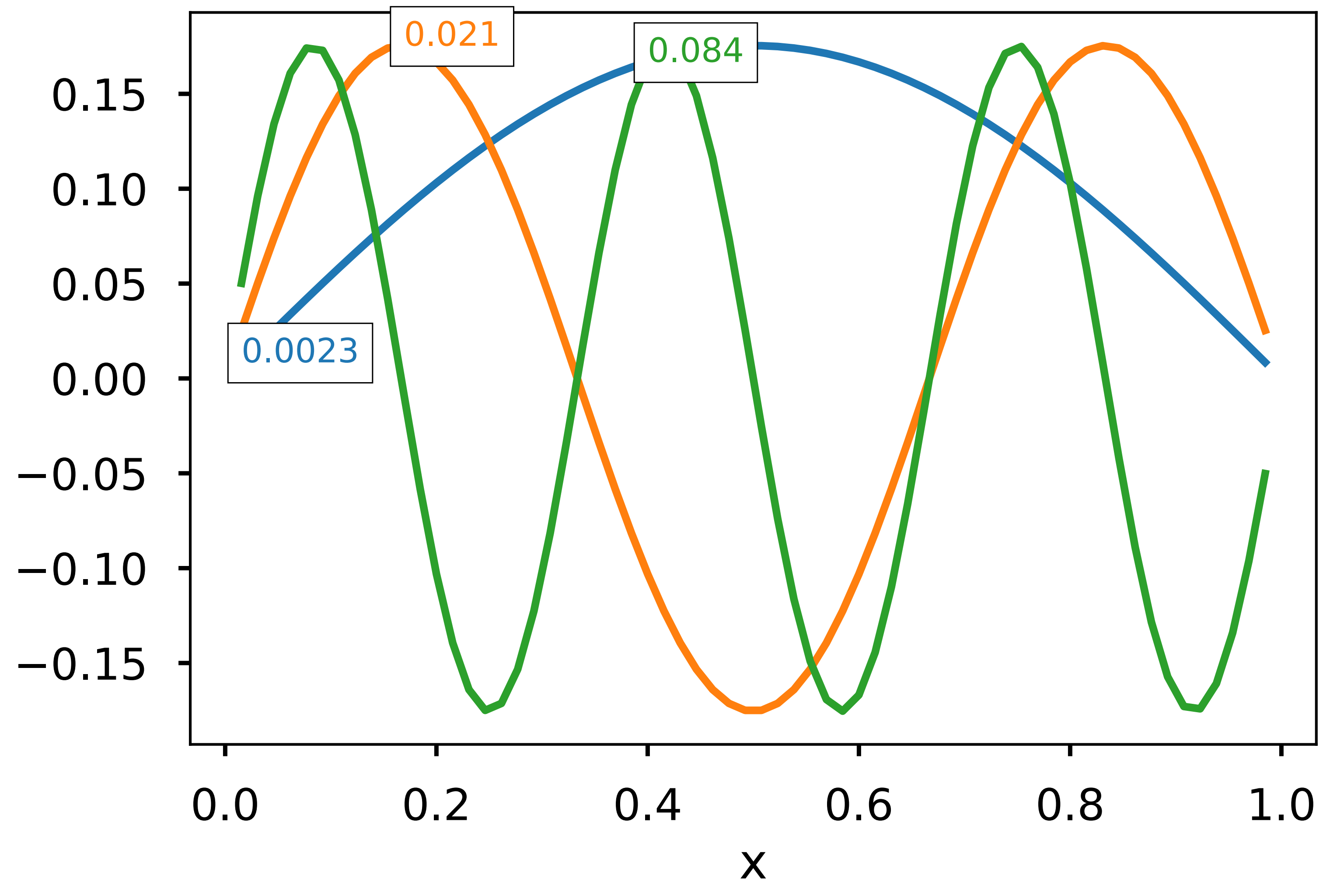
smoothest component = small λ_k $(O(h^2))$

The eigenvalues

$$\lambda_k = 4\sin^2\left(\frac{k\pi}{2(n+1)}\right)$$

$k = 1, \dots, n$

range from (0,4]
(or from c h^2 to ~4)



The eigenvectors are Fourier modes:

$\lambda_1 = 4\sin^2\left(\frac{\pi}{2(n+1)}\right) \approx 4\frac{\pi^2 h^2}{4}$

$$(v_k)_j = \sin\left(\frac{(j+1) * k\pi}{n+1}\right)$$

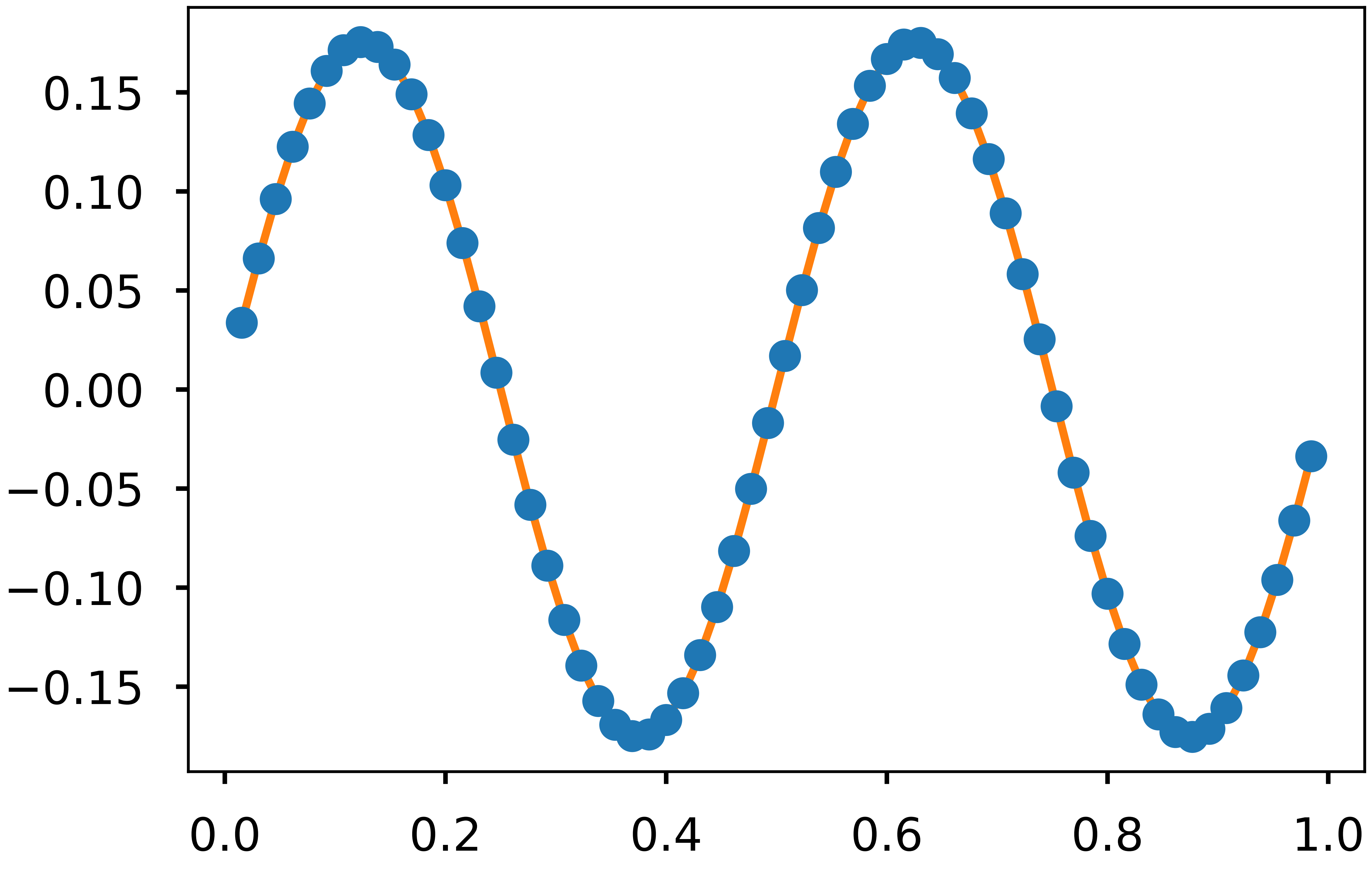
multigrid: (A) relaxation on the fine grid
(B) coarse-grid correction

Take a look: `01-model-problem.ipynb`

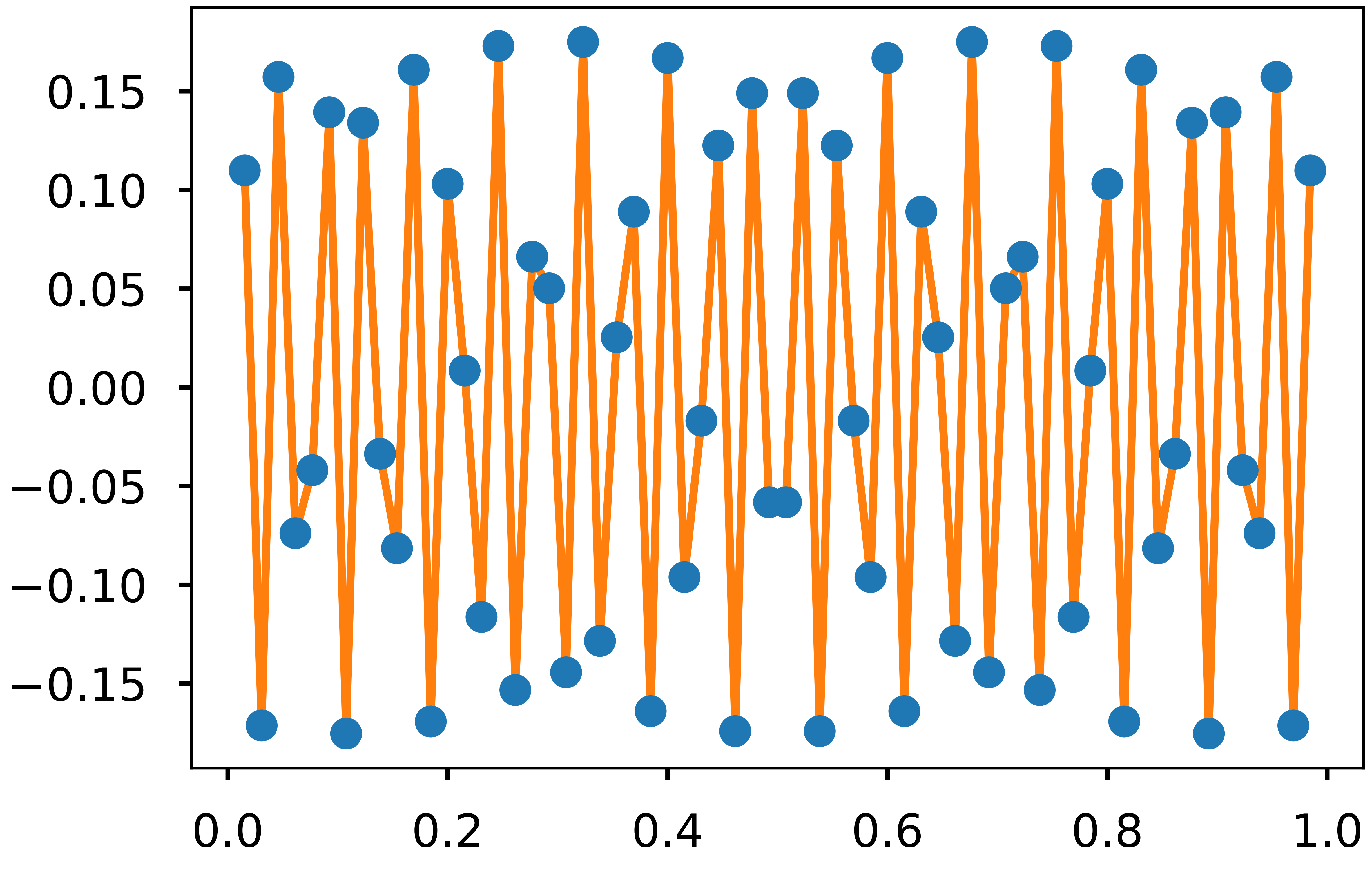
Smooth Mode ~ low Fourier Modes

(for ingredient (A))

- We will talk a lot about "smoothness" — how much variation there is **per** grid point.



small k



large k $\approx$ n

First relaxation scheme: Jacobi

$(h^2 \text{ in it})$

$Au = f$

- The discretization at point i:

d_{ii}

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i$$

- Solving for the variable at this point (eliminating the residual):

$$u_i \leftarrow \frac{1}{2}\left(u_{i-1} + u_{i+1} + h^2 f_i\right)$$

$1/d_{ii}$

- In matrix form:

$$u \leftarrow (I - D^{-1}A)u + h^2 D^{-1} f$$

$$u \leftarrow u + D^{-1} r$$

$$u^* - u \leftarrow u^* - u - D^{-1} A e$$

$$e \leftarrow \underbrace{(I - D^{-1} A)}_{G} e$$

- And the error:

$$e \leftarrow Ge \qquad \boxed{G = I - D^{-1}A}$$

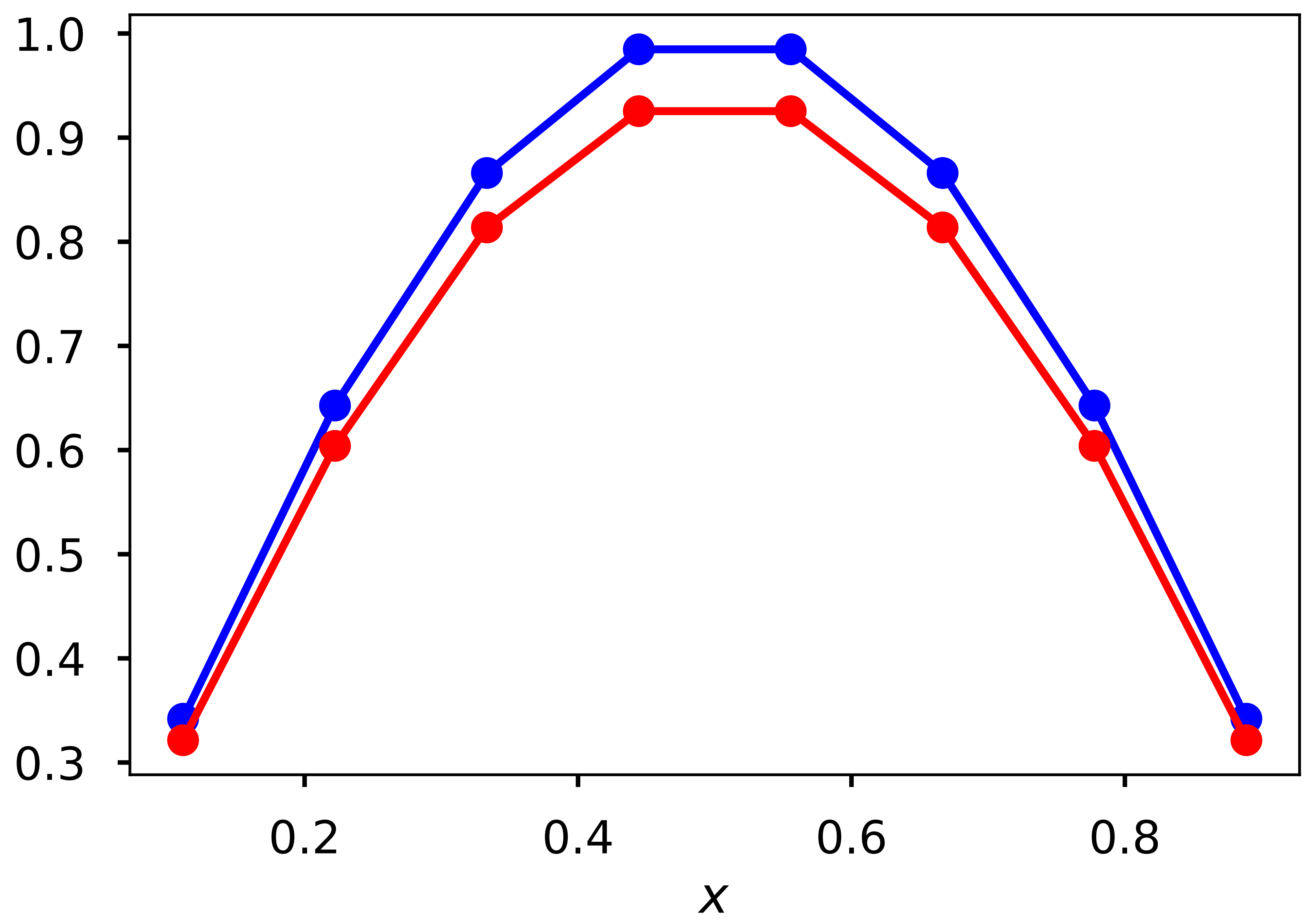
The "error propagation operator"

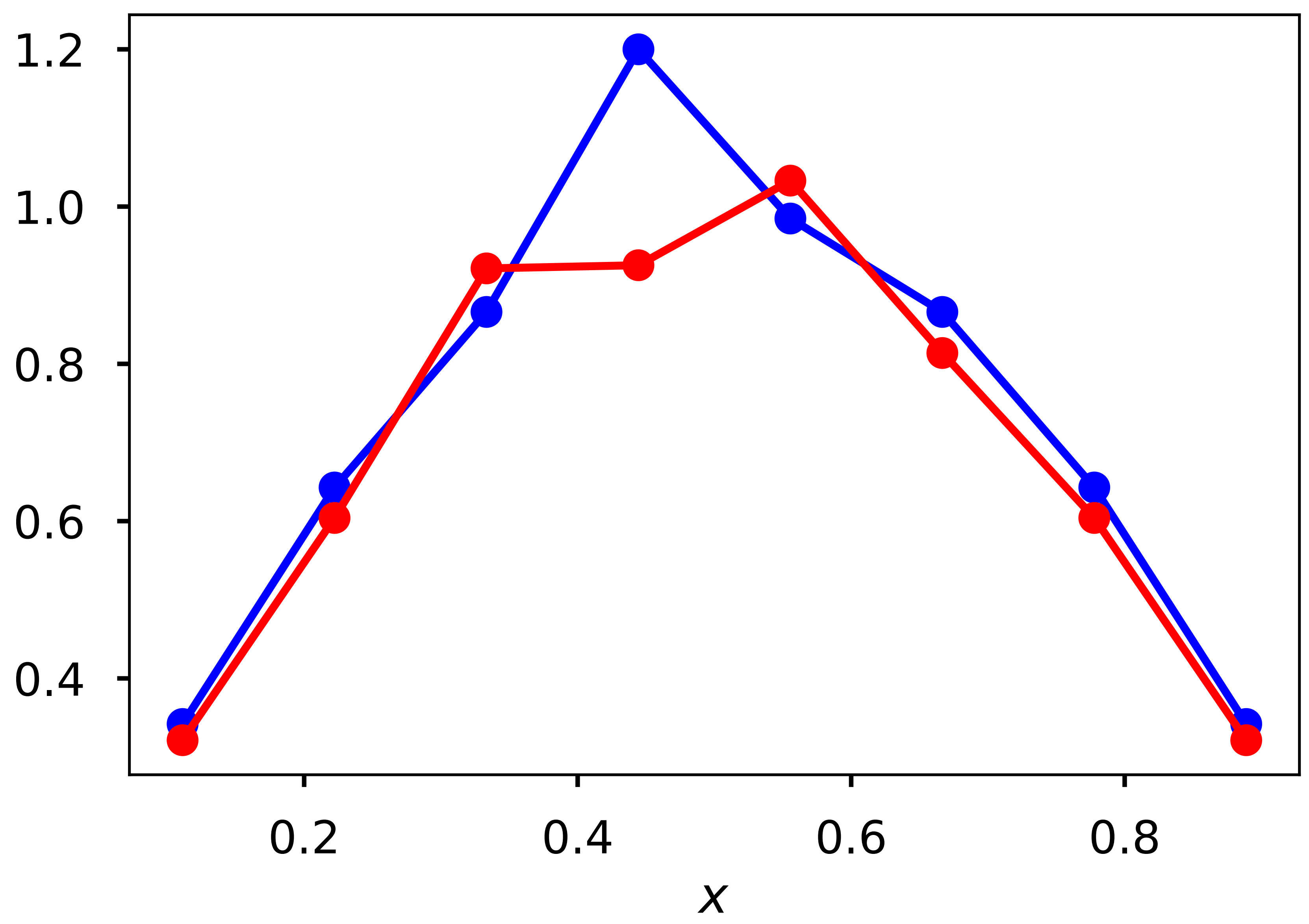
What does Jacobi do to error?

$$\color{red}{e^{new}} \leftarrow (I - D^{-1}A)\color{blue}{e^{old}}$$

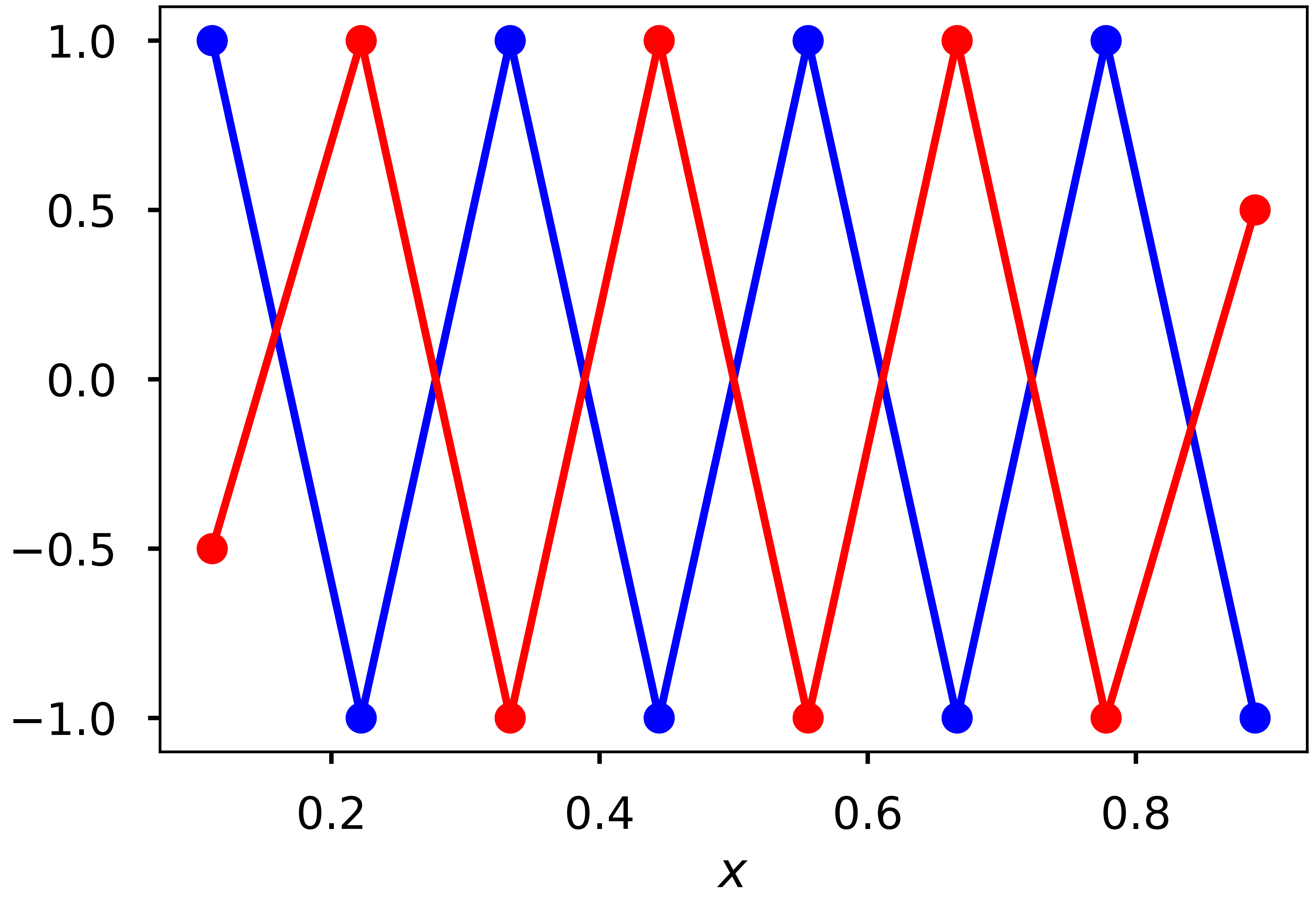
$$\color{red}{e_i^{new}} \leftarrow \frac{1}{2}\left(\color{blue}{e_{i-1}^{old}} + \color{blue}{e_{i+1}^{old}}\right)$$

- It averages…
- Consider smooth and oscillatory error:





But…

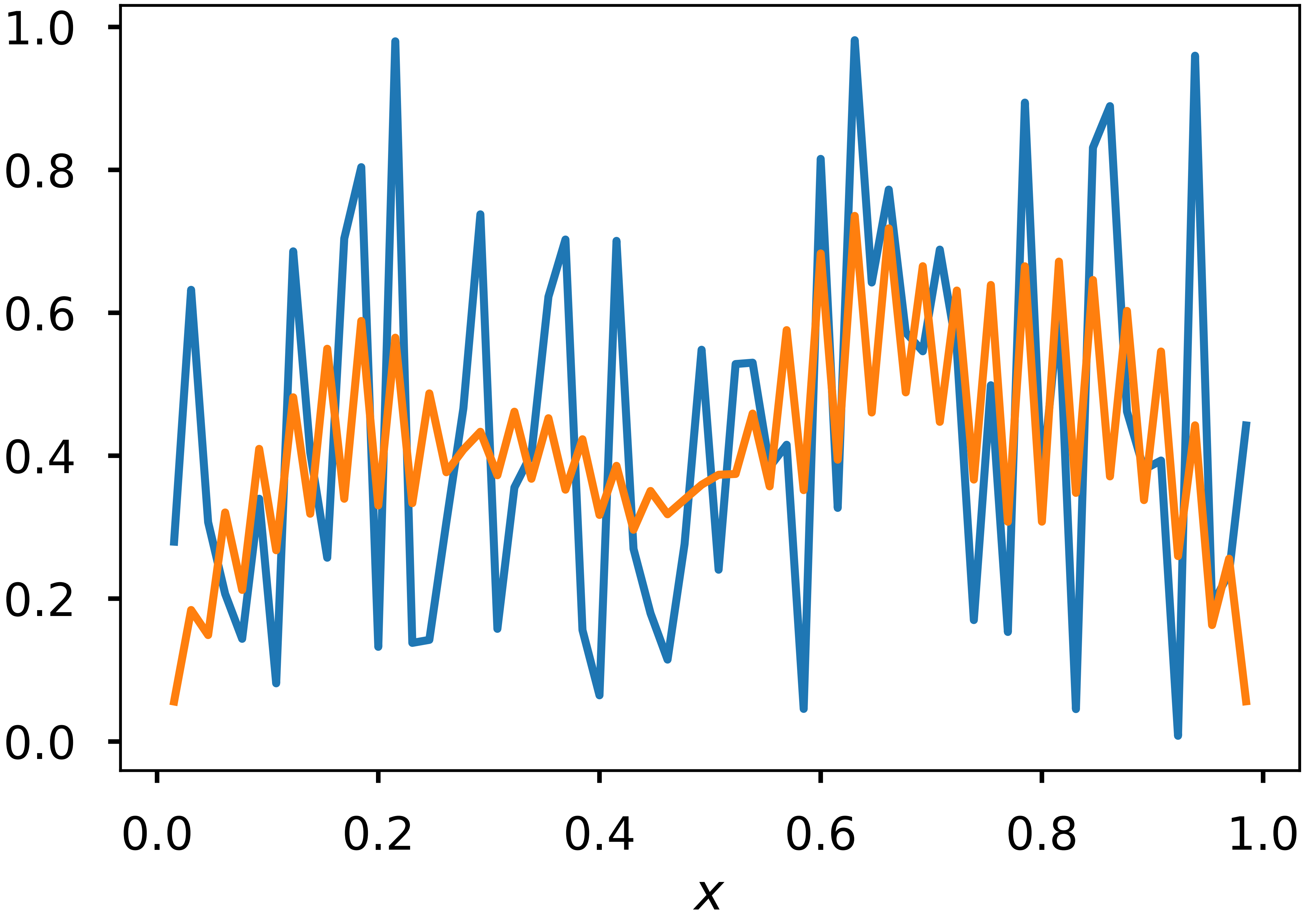


- Jacobi "averages out" certain error very quickly.
- But stagnates on very smooth error or very oscillatory error

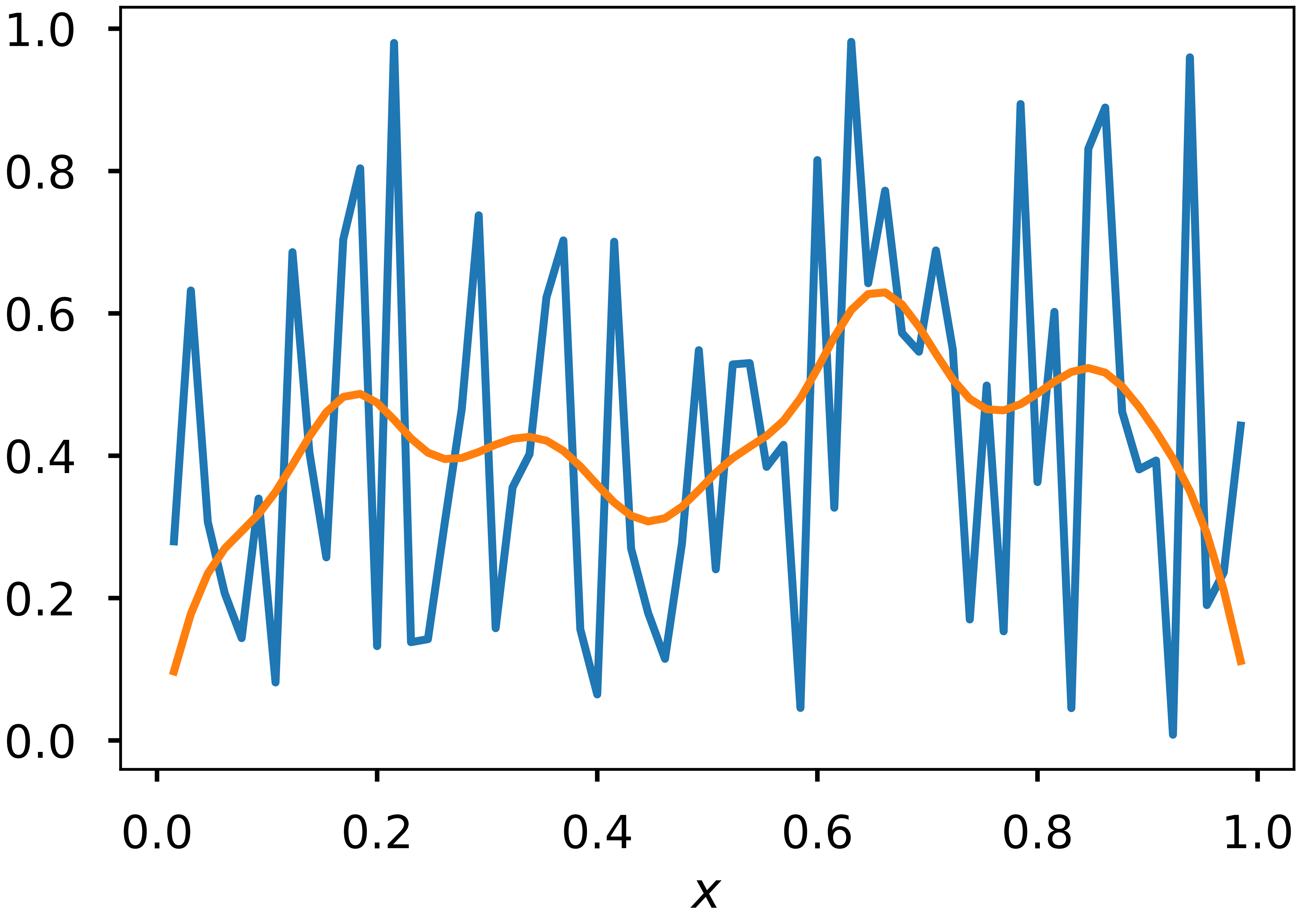
From Jacobi to weighted-Jacobi

- Random initial guess (random error):

$$u \leftarrow u + D^{-1}r$$



$$u \leftarrow u + (2/3)D^{-1}r$$



Weighted Jacobi

- Weighted Jacobi

$$u \leftarrow u + (2/3)D^{-1}r \;=\; \frac{1}{3}u + \frac{2}{3}(u + D^{-1}r)$$

$$= (1-\omega)u + \omega(u + D^{-1}r)$$

- What does this do to other modes?
  - Are smooth modes damped slowly?
  - What about oscillatory modes?

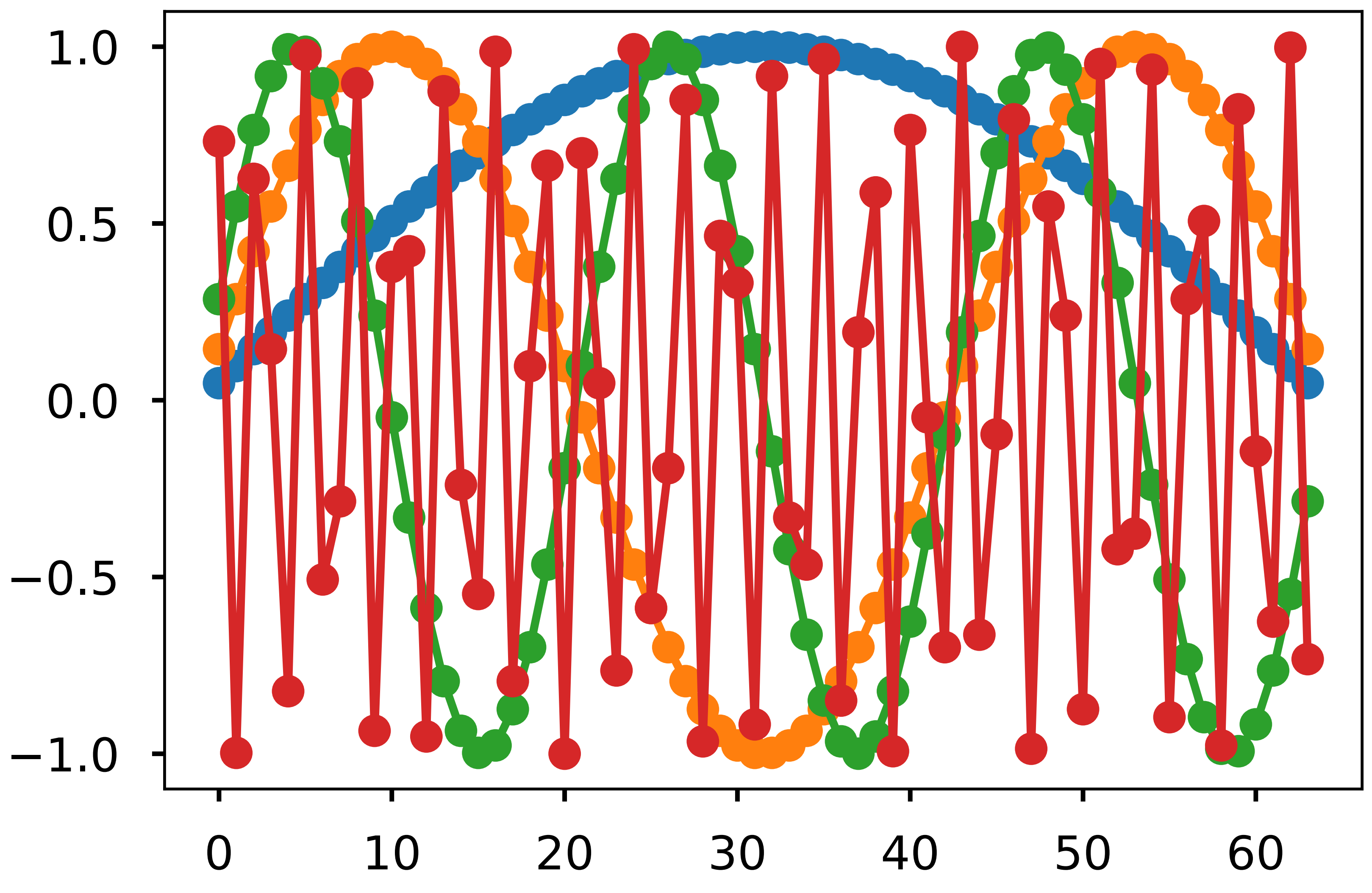
- Why did we pick 2/3 — that seemed like a lucky guess?!

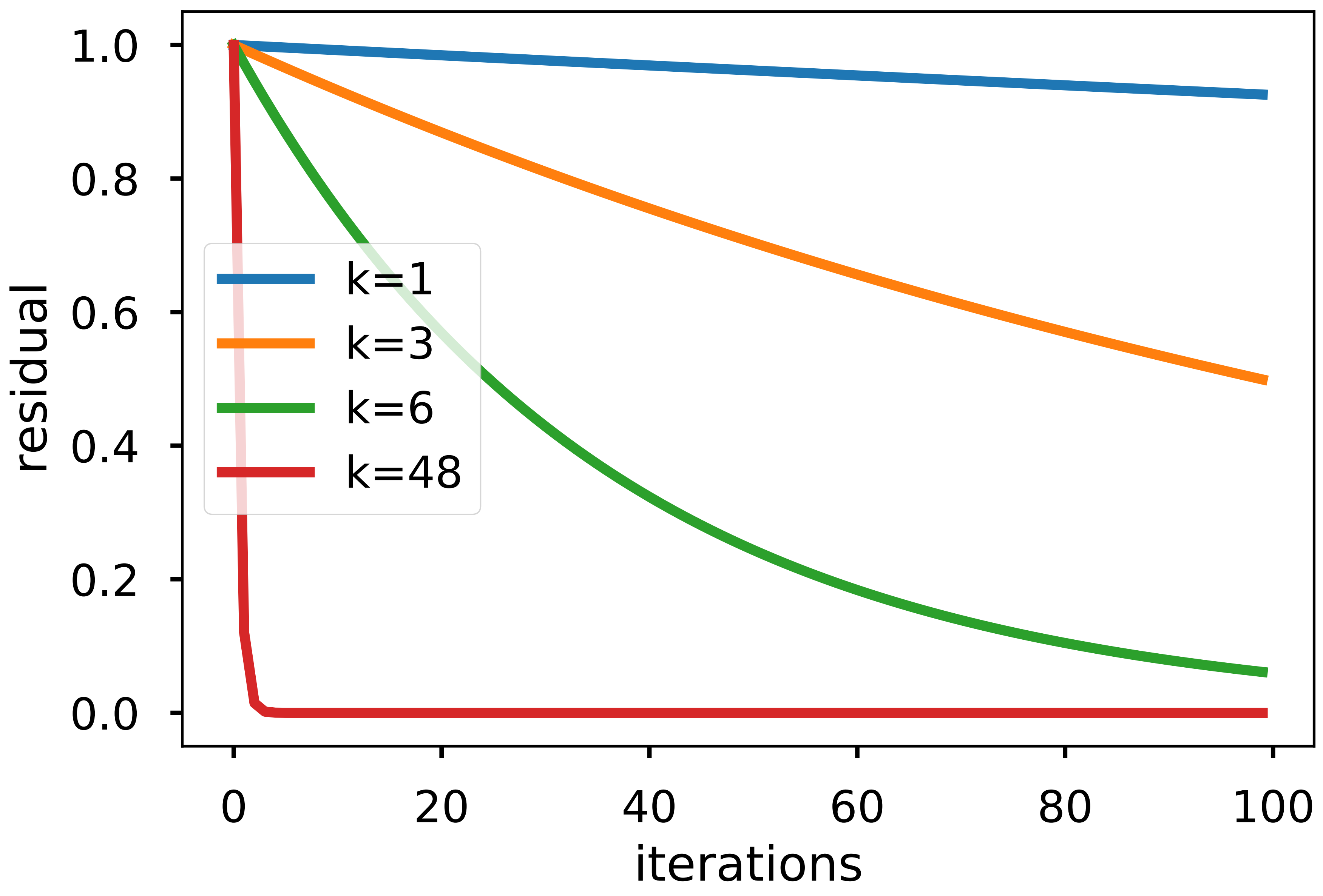
Weighted Jacobi

$n = 64$

$h = \frac{1}{65}$

- If we picked 4 modes, 1, 3, 6, and 48 (out of 64)
- Then smooth modes still dampen less quickly than higher ones





Error Propagation

- Let's consider an initial error

$$e_0 = \sum_{k=1}^{n} c_k v_k$$

- And the weighted Jacobi iteration matrix

$$e \leftarrow (I - \omega D^{-1}A)e = Ge$$

- From ν iterations we have

$$G^{\nu} e_0 = \sum_{k=1}^{n} c_k G^{\nu} v_k$$

$$= \sum_{k=1}^{n} c_k \lambda_k^{\nu} v_k$$

eigenvalues of G

- As a result, mode **k** is reduced by the magnitude of λ_k in every pass

(assume D has constant diagonal)

$Au = f$

$A\hat{u} = \hat{\lambda}\hat{u}$

$G = (I - \omega D^{-1}A)$

then $(I - \omega D^{-1}A)\hat{u}$

$= u - \omega D^{-1}A\hat{\lambda}\hat{u}$

$= (1 - \omega \frac{1}{d}\hat{\lambda})\hat{u}$

so $\lambda = 1 - \frac{\omega}{d}\hat{\lambda}$, $u = \hat{u}$

$k=1$: $\hat{\lambda}_1 = O(h^2)$

$\lambda_1 = O(1)$

Fundamental Theorem of Iteration

$$G = I - M^{-1}A$$

- Convergent ($G^n \to 0$) if and only if $\rho(G) < 1$

if $G = G^T$,

$\|G\| = \rho(G)$

$$\frac{\|e_n\|}{\|e_0\|} \leq \|G^n\| \leq \|G\|^n \approx 10^{-d}$$

$n \log_{10}(\rho(G)) \approx -d$

- How many iterations do we need to guarantee the reduction of the error by *d* digits?

$$n \approx -\frac{d}{\log_{10} \rho(G)}$$

Relaxation

- Convergence **factor**

$$\|G\| \text{ or } \rho(G)$$

- Convergence **rate**

$$-\log_{10}\rho(G)$$

(digits per iteration)

- For Jacobi:

$$G = I - D^{-1}A \longrightarrow \lambda_k = 1 - \frac{1}{2}\cdot 4 \cdot \sin^2\left(\frac{k\pi}{2(n+1)}\right)$$

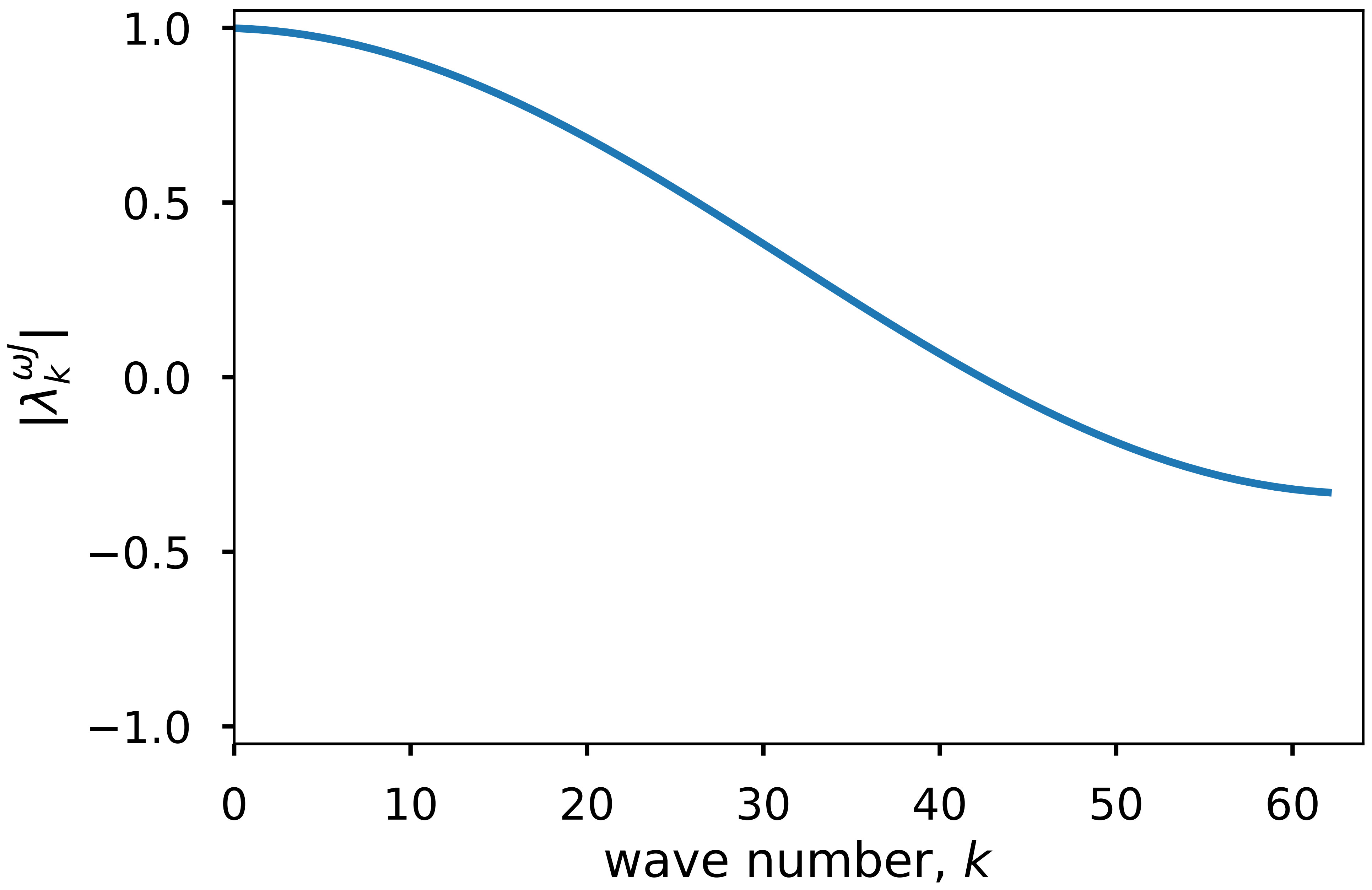
$$G = I - (2/3)D^{-1}A \longrightarrow \lambda_k = 1 - \frac{2}{3}\cdot\frac{1}{2}\cdot 4 \cdot \sin^2\left(\frac{k\pi}{2(n+1)}\right)$$

The spectral radius is almost the same, but…

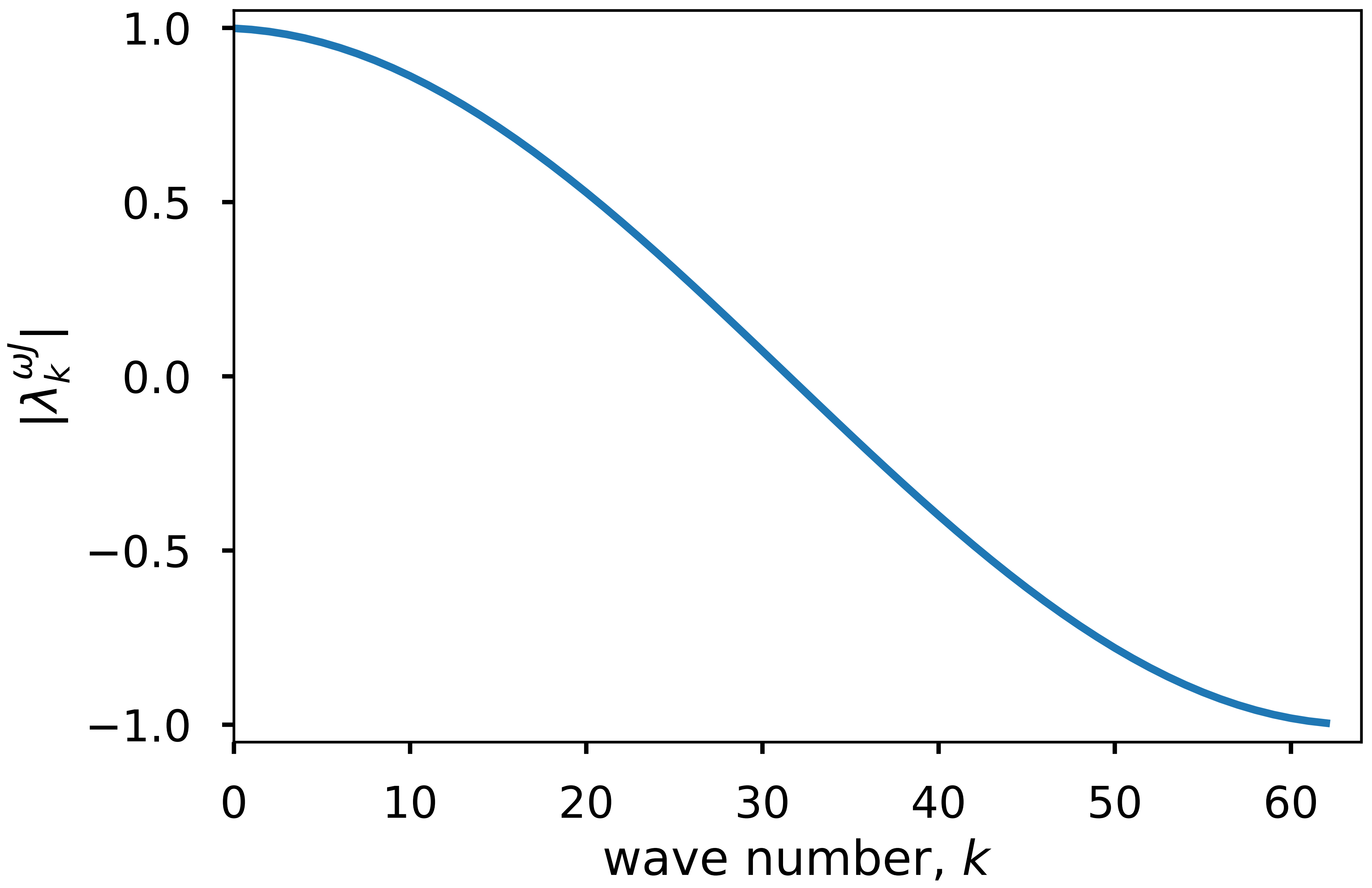
Weighted Jacobi

- If we look at the spectrum:
    - Weighted Jacobi dampens modes that are highly oscillatory

$$\omega = 2/3$$



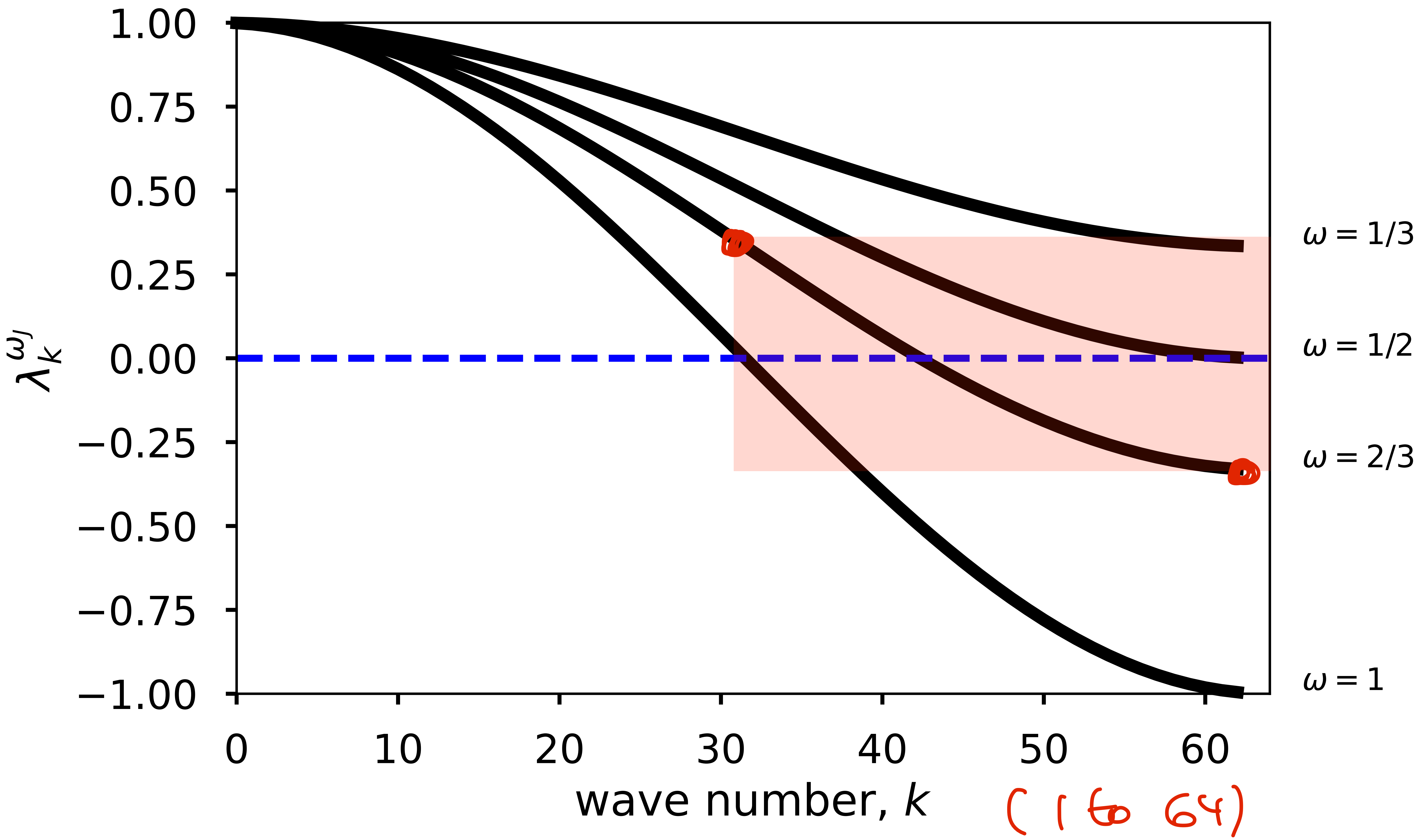
$$\omega = 1$$



Take a look: `02-relaxation-Jacobi.ipynb`

Weighted Jacobi

- Selecting 2/3 balances the reduction in error in the **high** modes

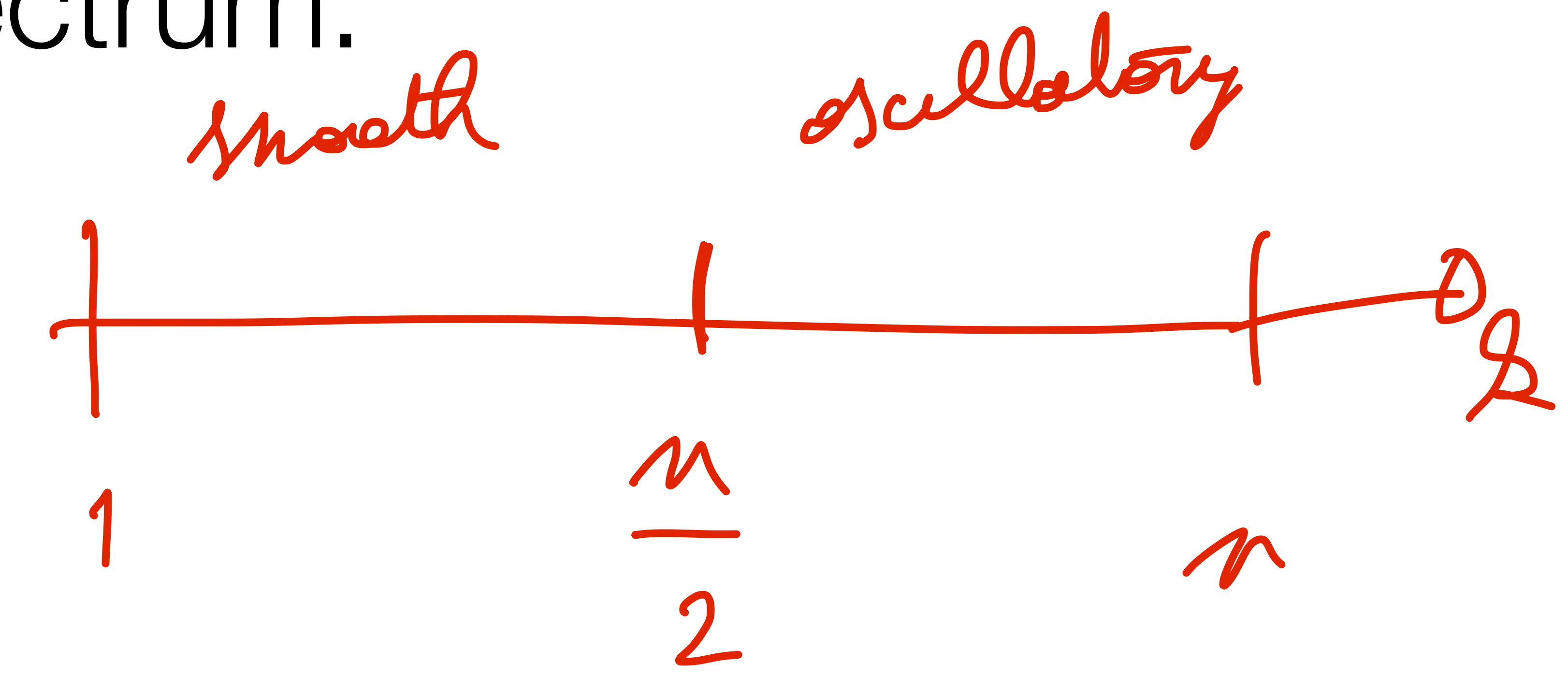


$(n = 64)$

The multigrid smoothing factor

- The **smoothing factor** of relaxation method G is the maximum magnitude of the upper half of the spectrum:

$$\max_{k \in [n/2, n]} |\lambda_k^G|$$



- A desired feature for relaxation:

$$\boxed{\begin{array}{c}\textbf{Oscillatory}\text{ modes are quick to converge}\\ \textbf{Smooth}\text{ modes are slow to converge}\end{array}}$$

Multigrid Step #1: pick a smoother

(A) relaxation on the fine grid

- For $\omega = 2/3$

$$|\lambda_{n/2}| = |\lambda_n| = 1/3$$

- For $\omega = 1$

$$|\lambda_{\cancel{n/2}}| = |\lambda_n| = 1$$

1

$$|\lambda_{\frac{n}{2}}| \approx 0$$

- Jacobi is not a smoother (weighted *is*)

Take a look: `03-smooth-error.ipynb`

An important observation on "smoothness"

- So far, we've mainly looked at

$$Au = 0$$

- In general we need to consider

$$Au = f$$

- If we **smooth** with

$$u \leftarrow u + \omega D^{-1} r$$

$e_1 = G e_0$

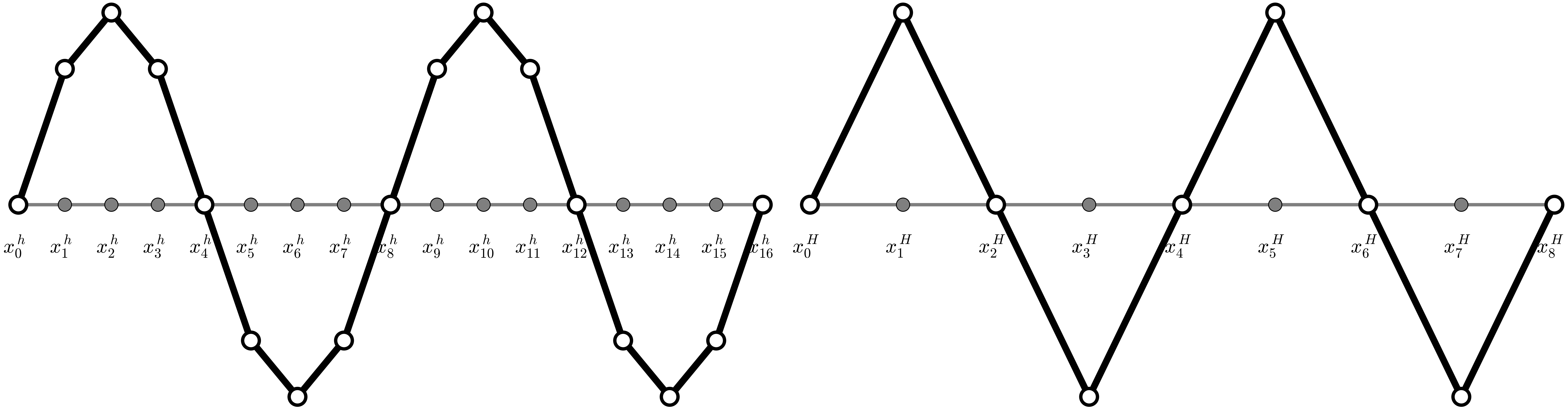
then the **error** is smooth, not (necessarily the solution).

Coarse Grids

B use coarse-grid correction to remove smooth fine-level error modes

- Smooth modes look like oscillatory modes when sampled on a coarse grid
- 4-mode of 15 **versus** 4-mode of 7

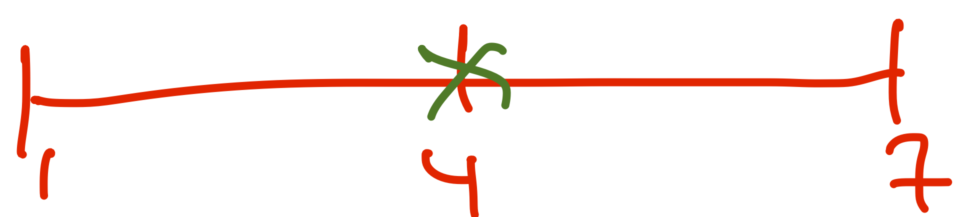
→ use relaxation on coarse grid ∇



$k=4,\ n=15$ $k=4,\ n=7$

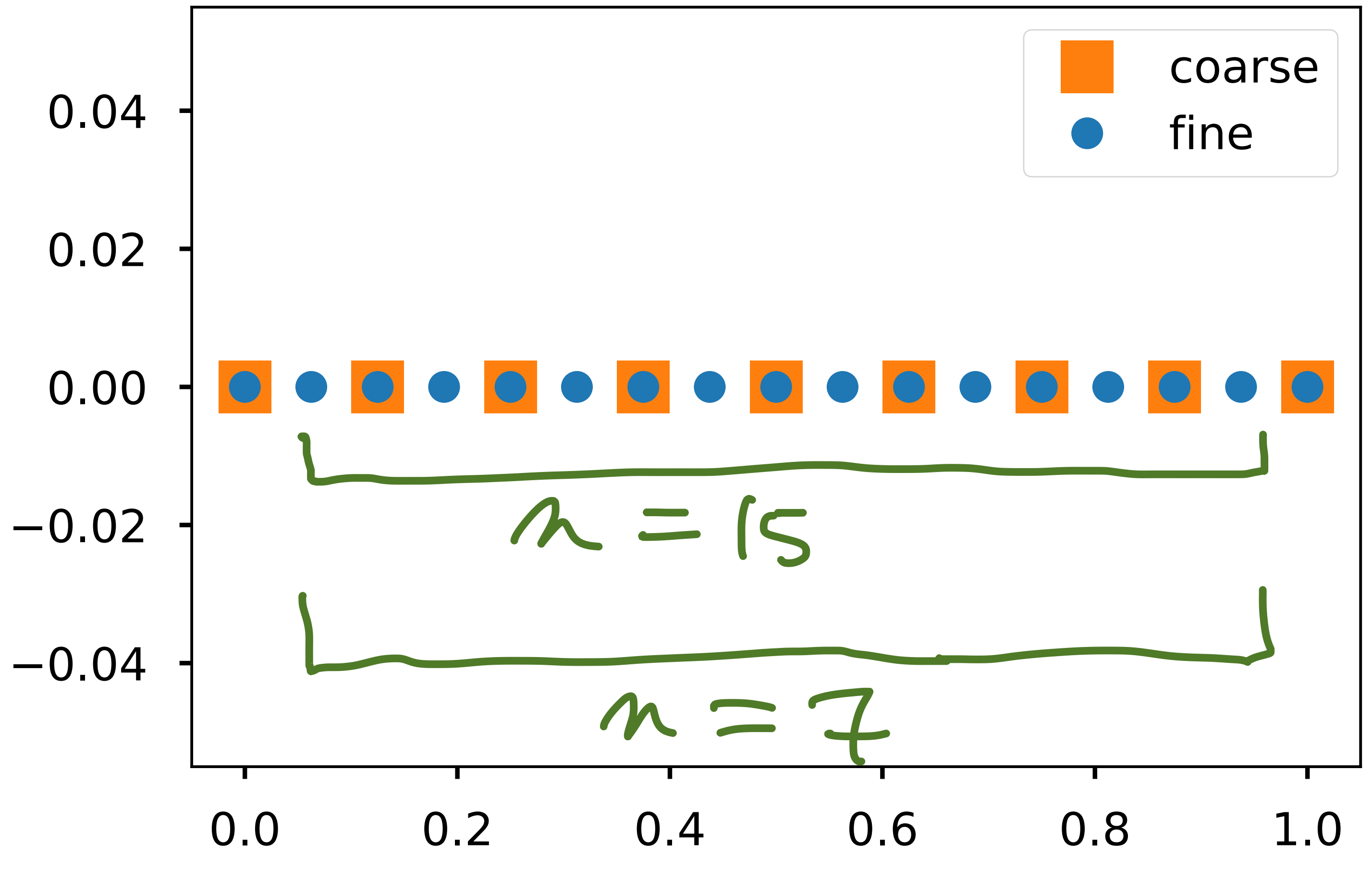
This looks like a “smooth” mode

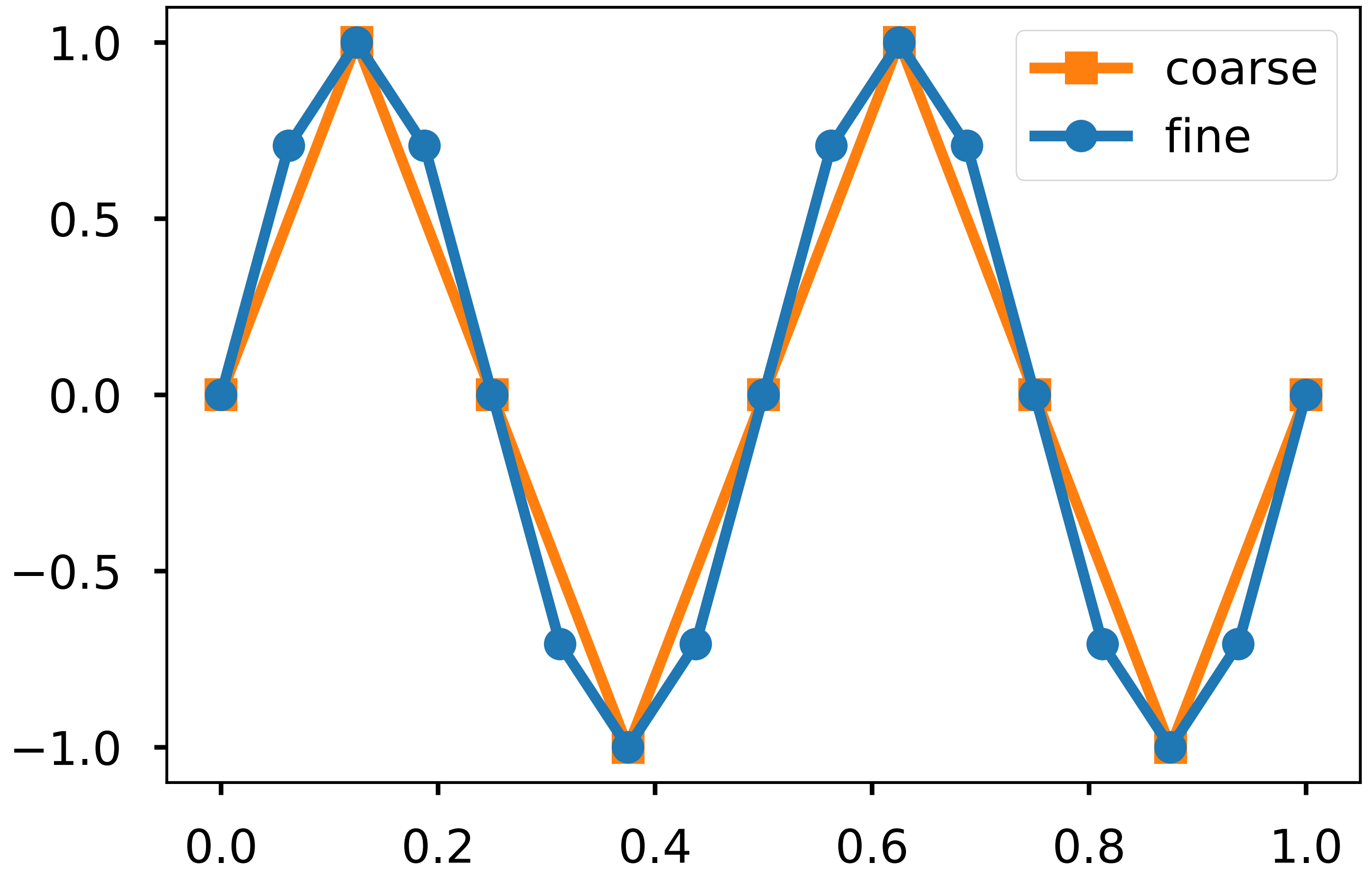
This looks like an “oscillatory” mode



Coarse modes $\left(k \in \left[1, \frac{n-1}{2}\right]\right)$

$h = \frac{1}{n+1}$





4 fine

4 coarse

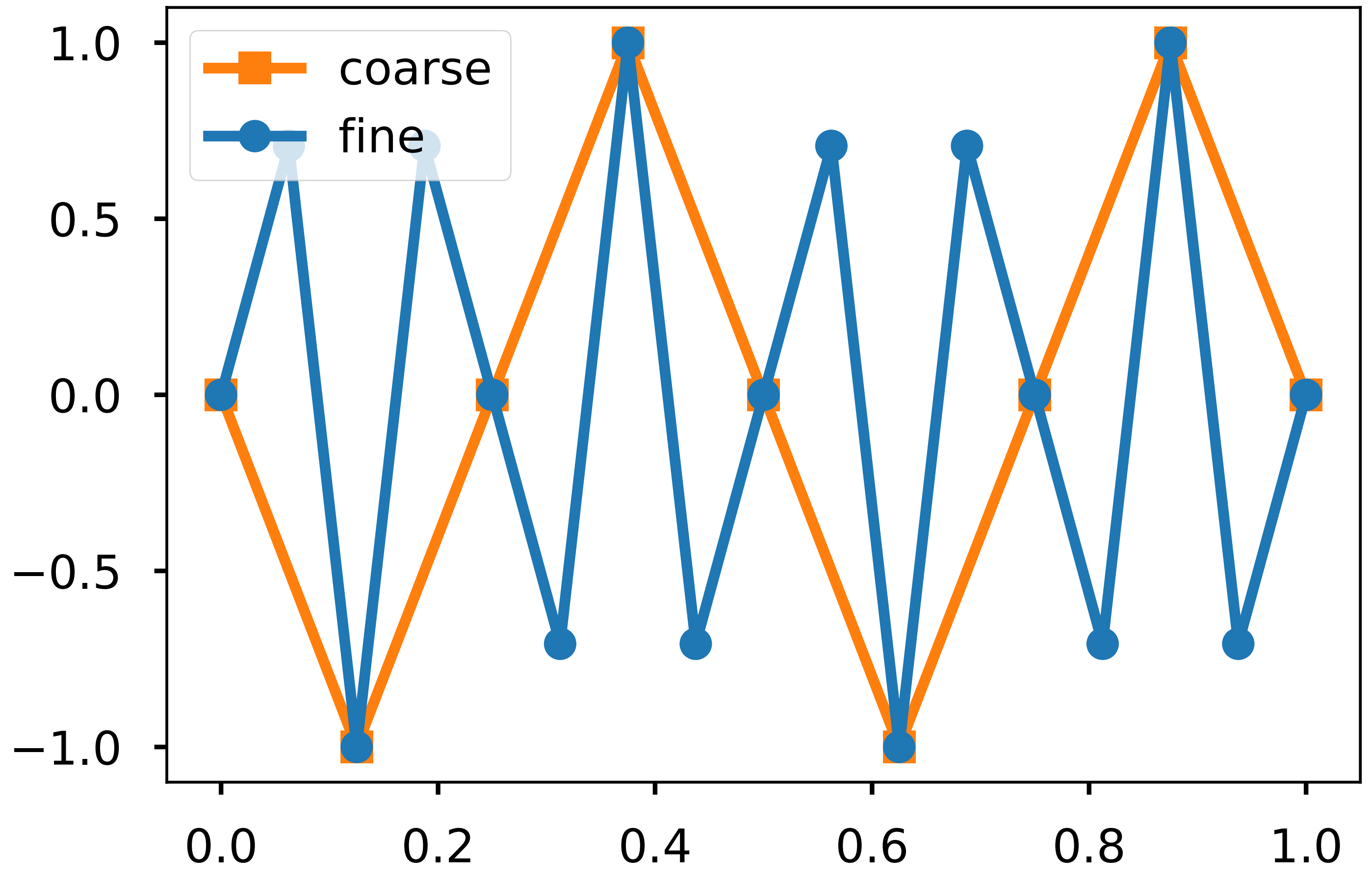
mode 4 of 15

$$(v_k)_j = \sin\frac{jk\pi}{n+1}$$ Fine mode

$$(v_k)_{2j} = \sin\frac{2jk\pi}{n+1}$$ Fine mode (every other)

$$= \sin\frac{jk\pi}{(n+1)/2}$$ Coarse mode

$$= (\hat{v}_k)_j$$



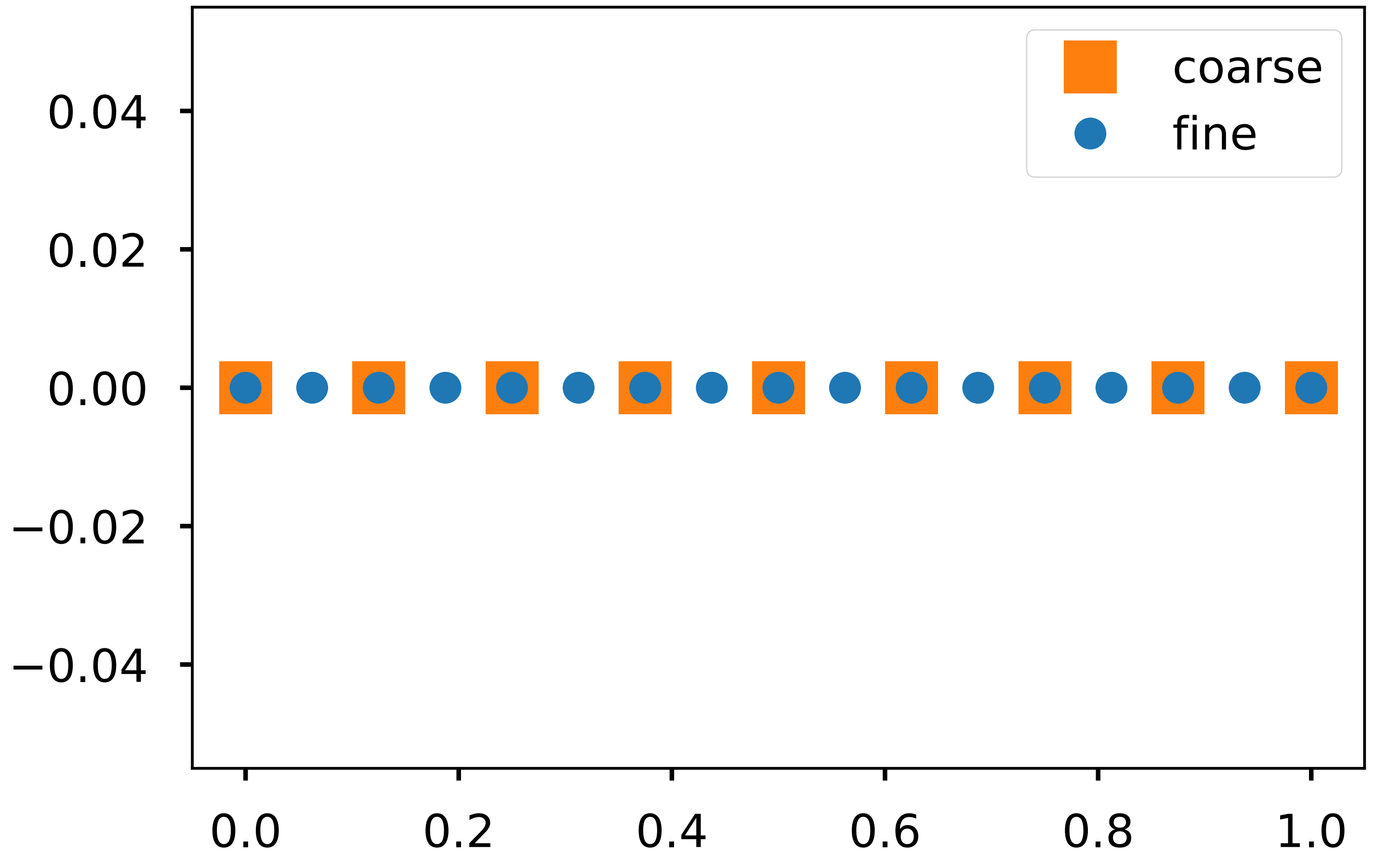
12 fine becomes −4 coarse

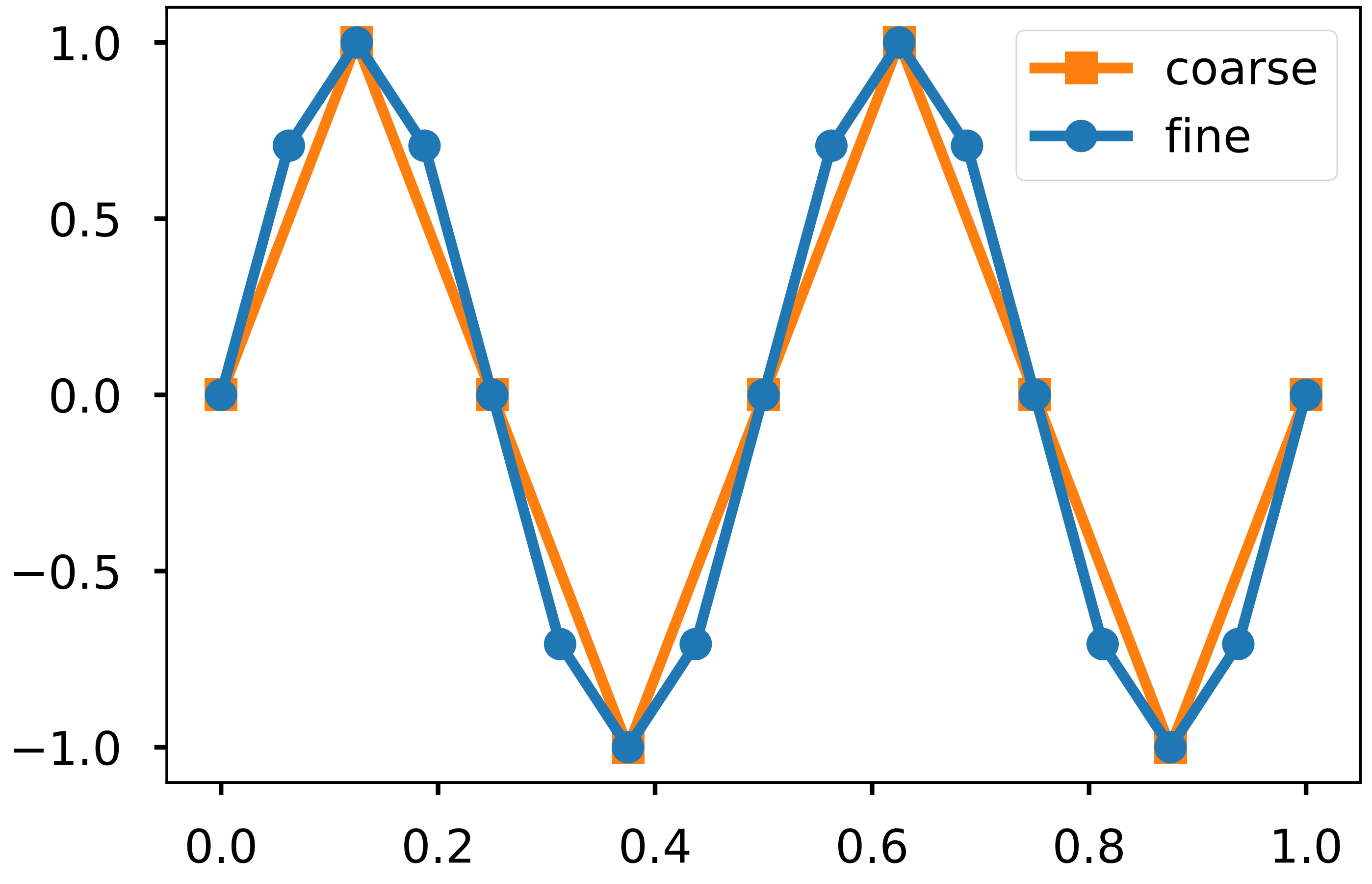
mode 12 of 15

For low modes, k-modes are preserved

Take a look: `04-coarse-modes.ipynb`

Coarse modes $\left(k \in \left[\frac{n+1}{2}, n\right]\right)$





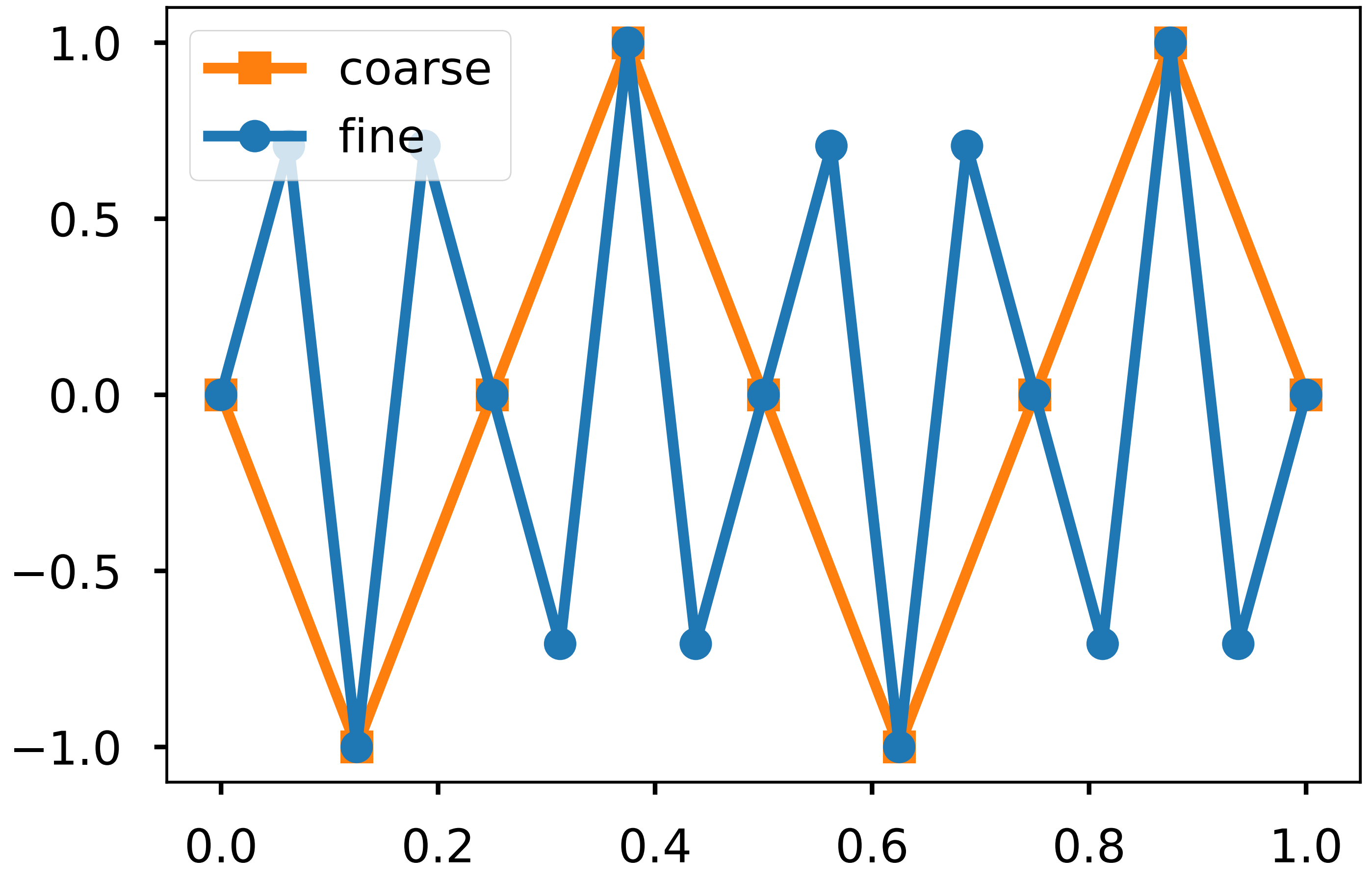
mode 4 of 15

$$(v_k)_{2j} = \sin\frac{2jk\pi}{n+1}$$ Fine mode

$$= -\sin\frac{2j(n-k)\pi}{n+1}$$ Fine mode (every other)

$$= -\sin\frac{j(n-k)\pi}{(n+1)/2}$$ Coarse mode

$$= -(\hat{v}_{n-k})_j$$

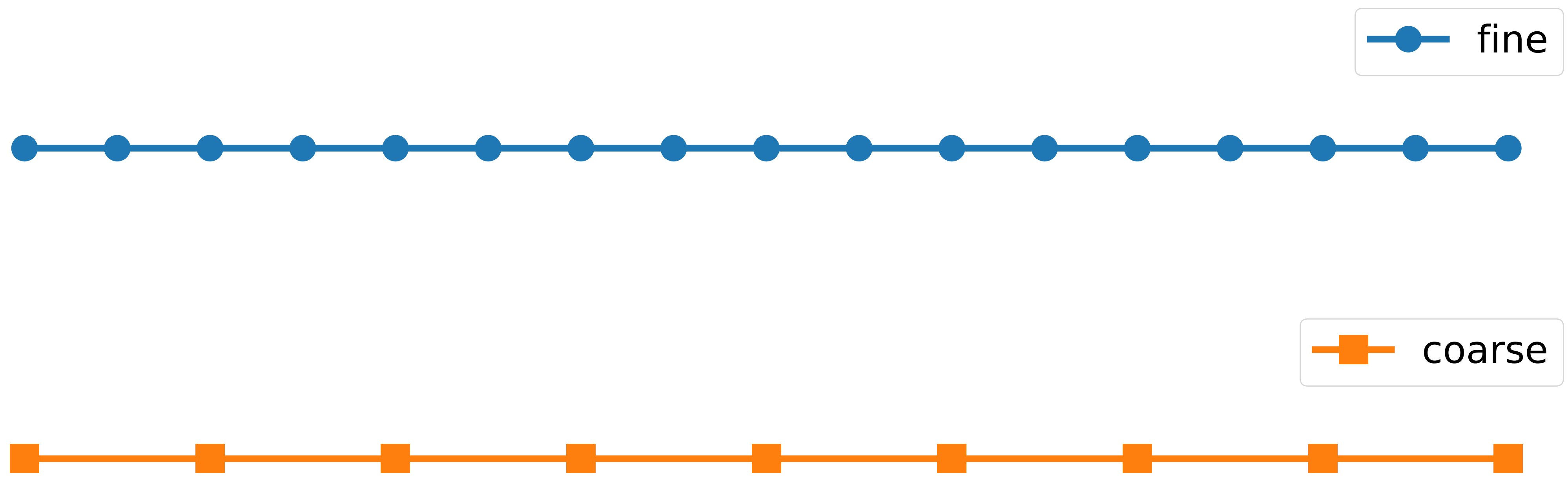


12 fine becomes −4 coarse

mode 12 of 15

For high modes, k-modes are *aliased*

Questions to resolve…



- How to transfer between **fine** and **coarse**?

(we want to use relaxation on coarse grid to correct fine grid)

- What do we do to "solve" on a coarse grid?

(relax to efficiently remove slow-to-converge fine-grid error)

What to use to transfer…

- Return to our projection problem: (optimal coarse-grid correction)

$$x_1 = \overbrace{x_0}^{\text{after five relaxation}} + \underbrace{V}_{15\times 7}\underbrace{(V^T A V)^{-1}}_{7\times 7}\underbrace{V^T}_{7\times 15}\underbrace{r_0}_{15\times 1}$$

- If we have a smoothing property, then smoothing a few times will leave smooth error.
- coarse-grid correction: project e_0 onto range of V, and subtract the projection
- Let’s construct V from say continuous piecewise linears

Interpolation P: from coarse to fine

- Consider coarse grid

$$\Omega^{2h}$$

- and fine grid

$$\Omega^{h}$$

- Construct an operator

$$P : \Omega^{2h} \to \Omega^{h}$$

$$P = \begin{array}{cc} & 7 \\ 15 & \begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix} \end{array}$$

- Such that

$$Pv^{2h}$$

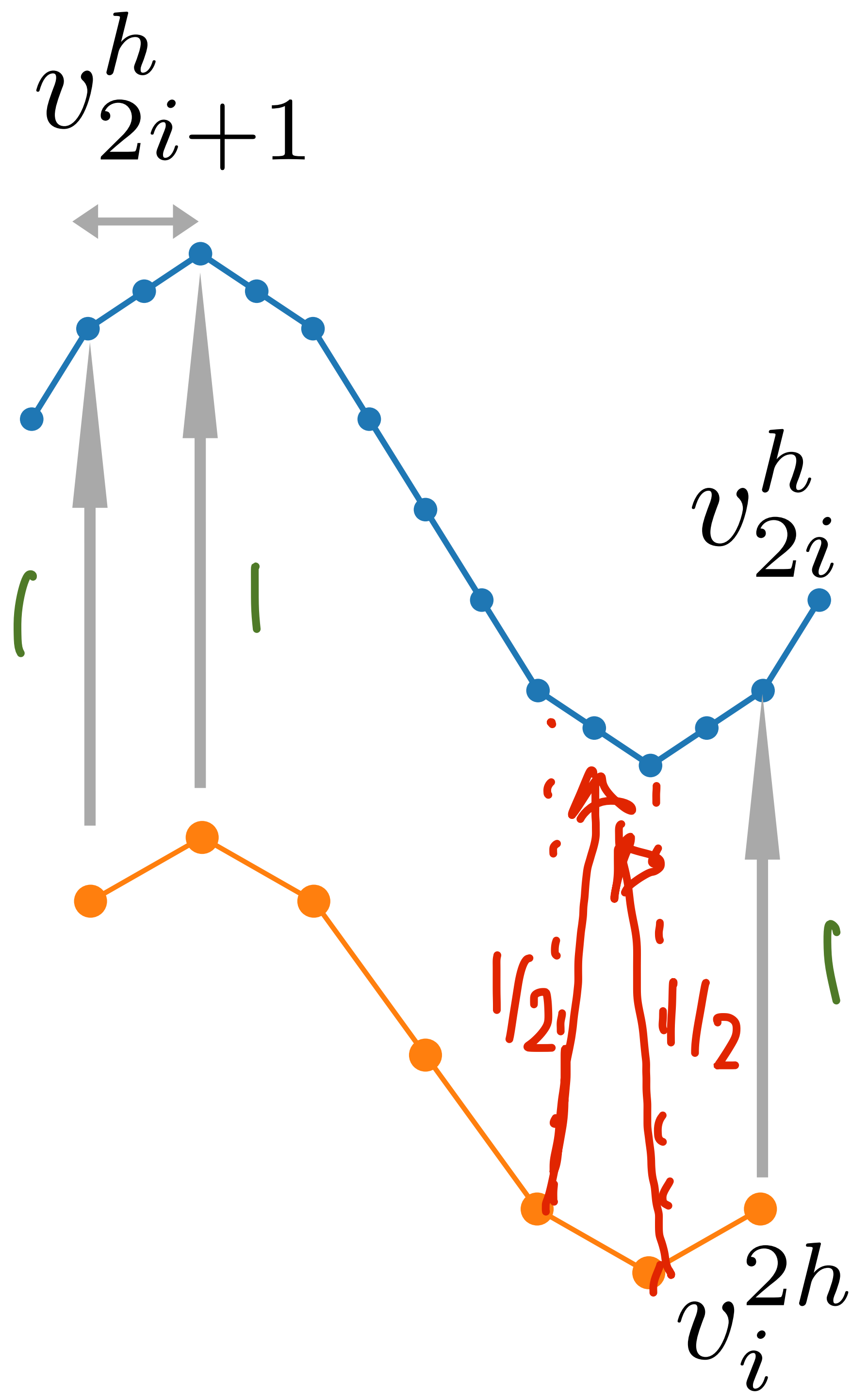
is continuous and piecewise linear

$(P = V)$

Interpolation P: from coarse to fine

$$v^h_{2i} = v^{2h}_i$$ Injection

$$v^h_{2i+1} = \frac{1}{2}(v^{2h}_i + v^{2h}_{i+1})$$ Average (linear interp)



15

4

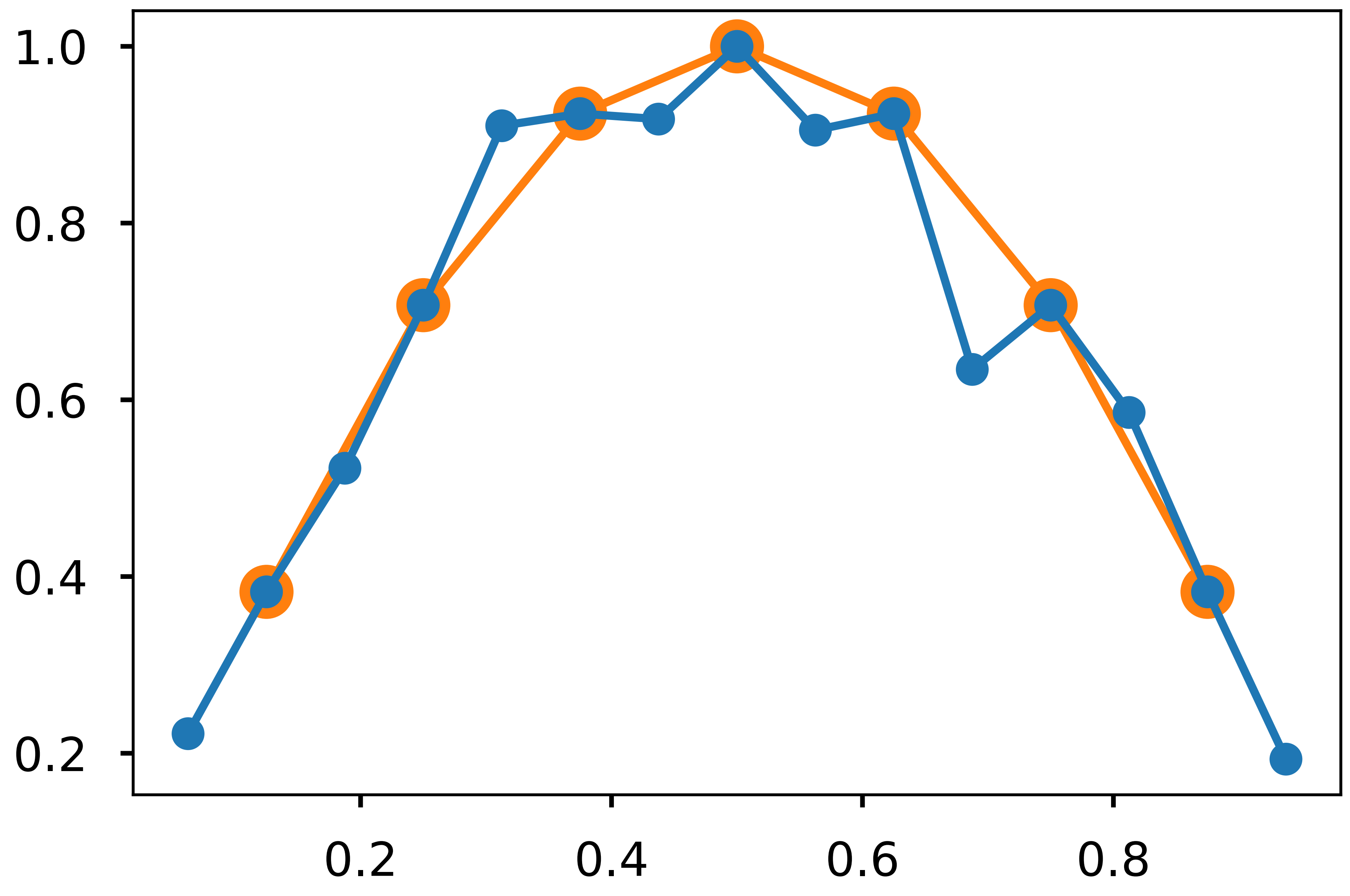
7

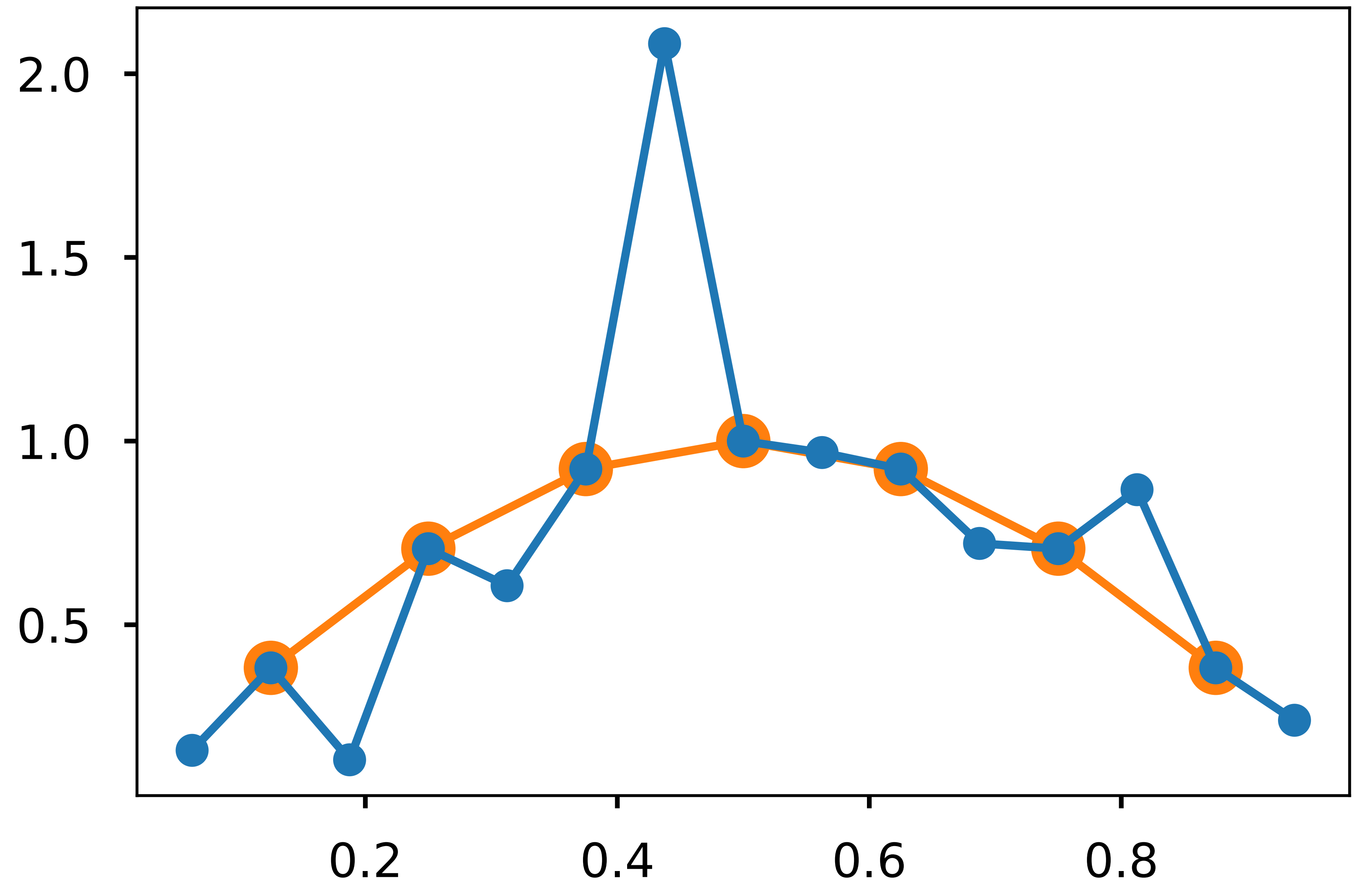
- Values at points common to both grids are reused (injected)

Interpolation

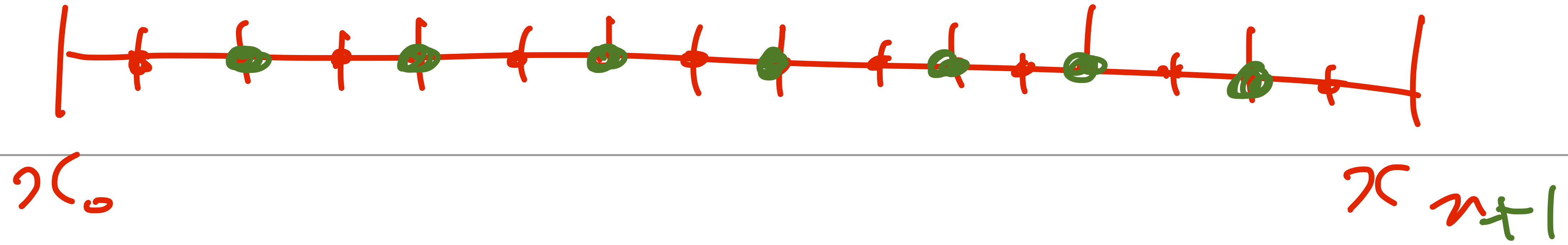
- interpolation gives smooth result

(we want to correct smooth error on the fine grid)

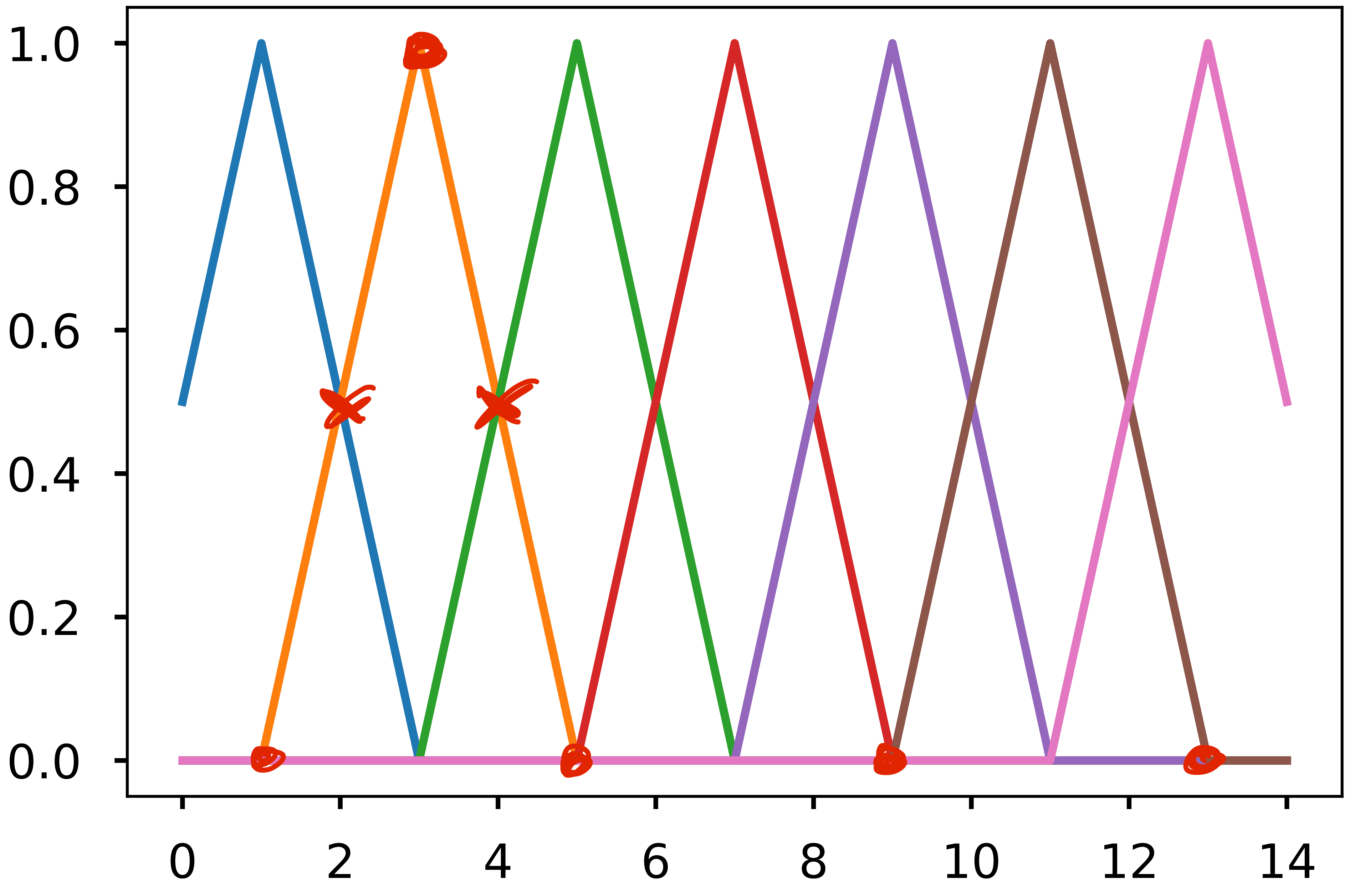




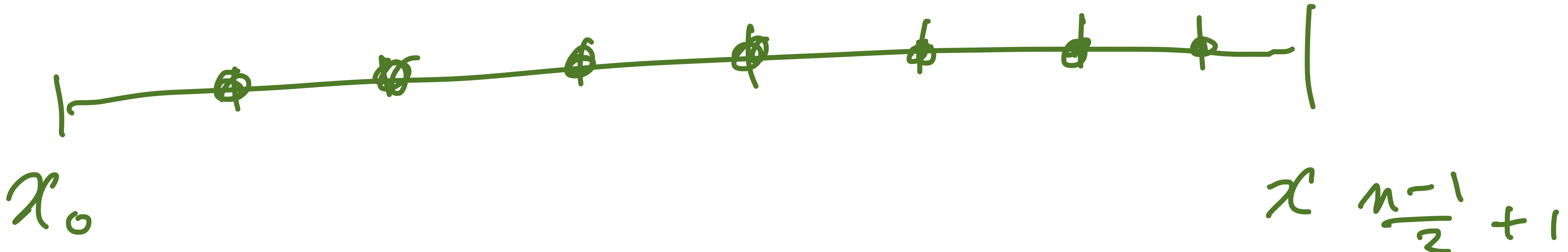
In matrix form



$$P = \frac{1}{2}\begin{bmatrix}
1. & & & & & & \\
2. & & & & & & \\
1. & 1. & & & & & \\
 & 2. & & & & & \\
 & 1. & 1. & & & & \\
 & & 2. & & & & \\
 & & 1. & 1. & & & \\
 & & & 2. & & & \\
 & & & 1. & 1. & & \\
 & & & & 2. & & \\
 & & & & 1. & 1. & \\
 & & & & & 2. & \\
 & & & & & 1. & 1. \\
 & & & & & & 2. \\
 & & & & & & 1.
\end{bmatrix}$$



- The columns of P are basis functions (right)
- The fine grid vectors, Pv, are linear combinations of these functions
- Notice: P is full rank!



The coarse grid operator

$$x_1 = x_0 + V(V^T AV)^{-1} V^T r_0$$

(sub)space defined by P

$x_1 = x_0 + P(P^T AP)^{-1} P^T r_0$

- If V is defined by $span\{P\}$ —or just P,

- Then V^T defines **restriction** as P^T

- And the **coarse level operator** is defined by $P^T AP$

└ optimal coarse-grid correction, given V
(minimizes $\|e_1\|_A$)

Take a look: `06-multigrid-two-level.ipynb`

Two level method

$$x_1 = x_0 + V(V^T A V)^{-1} V^T r_0$$

$$\downarrow$$

$$x_1 = x_0 + \underbrace{P(P^T A P)^{-1} P^T r_0}_{\hat{e}_c}$$

• Given	x_0		• What should this be?
• Smooth a few times	$x_0 \leftarrow x_0 + \omega D^{-1} A r_0$		• How many times
• Form residual	$r_0 = b - A x_0$		
• Restrict the residual	$P^T r_0$		
• Solve the coarse problem	$P^T A P \hat{e}_c = P^T r_0$	$A_c \hat{e}_c = r_c$	• What does this mean?
• Interpolate the approx error	$P\hat{e}_c$		• What are we interpolating?
• Correct the initial guess	$x_0 + P\hat{e}_0$		• Are we done?

$x_1 = x_0 + e_{CGC}$

find "good" e_{CGC} (to remove smooth fine-level error)

note $A e_0 = r_0$ (e_0 is smooth after fine-level relaxation)

$(x^* = x_0 + e_0)$

find coarse approximation of current e_0

$P^T A e_0 = P^T r_0$ let $e_0 \simeq P \hat{e}_c$

solve $\boxed{P^T A P \hat{e}_c = P^T r_0}$

(choose P such that its range contains smooth fine-level errors)

correct $x_1 \leftarrow x_0 + P \hat{e}_c$

$A_c = P^T A P$ or: $A_c = A_{2h}$

Notes on variants

- An alternative to restriction, is injection

$$\begin{bmatrix}
0 & 1 & 0 & & & & & & & & & & & & \\
 & & 0 & 1 & 0 & & & & & & & & & & \\
 & & & & 0 & 1 & 0 & & & & & & & & \\
 & & & & & & 0 & 1 & 0 & & & & & & \\
 & & & & & & & & 0 & 1 & 0 & & & & \\
 & & & & & & & & & & 0 & 1 & 0 & & \\
 & & & & & & & & & & & & 0 & 1 &
\end{bmatrix}$$

- Or a weighted transpose of linear interpolation (to restrict constants exactly)

$$\frac{1}{4}\begin{bmatrix}
1 & 2 & 1 & & & & & & & & & & & & \\
 & & 1 & 2 & 1 & & & & & & & & & & \\
 & & & & 1 & 2 & 1 & & & & & & & & \\
 & & & & & & 1 & 2 & 1 & & & & & & \\
 & & & & & & & & 1 & 2 & 1 & & & & \\
 & & & & & & & & & & 1 & 2 & 1 & & \\
 & & & & & & & & & & & & 1 & 2 & 1
\end{bmatrix}$$

- An alternative to $A_c = P^T A_h P$ is to rediscretize $A_c = A_{2h}$

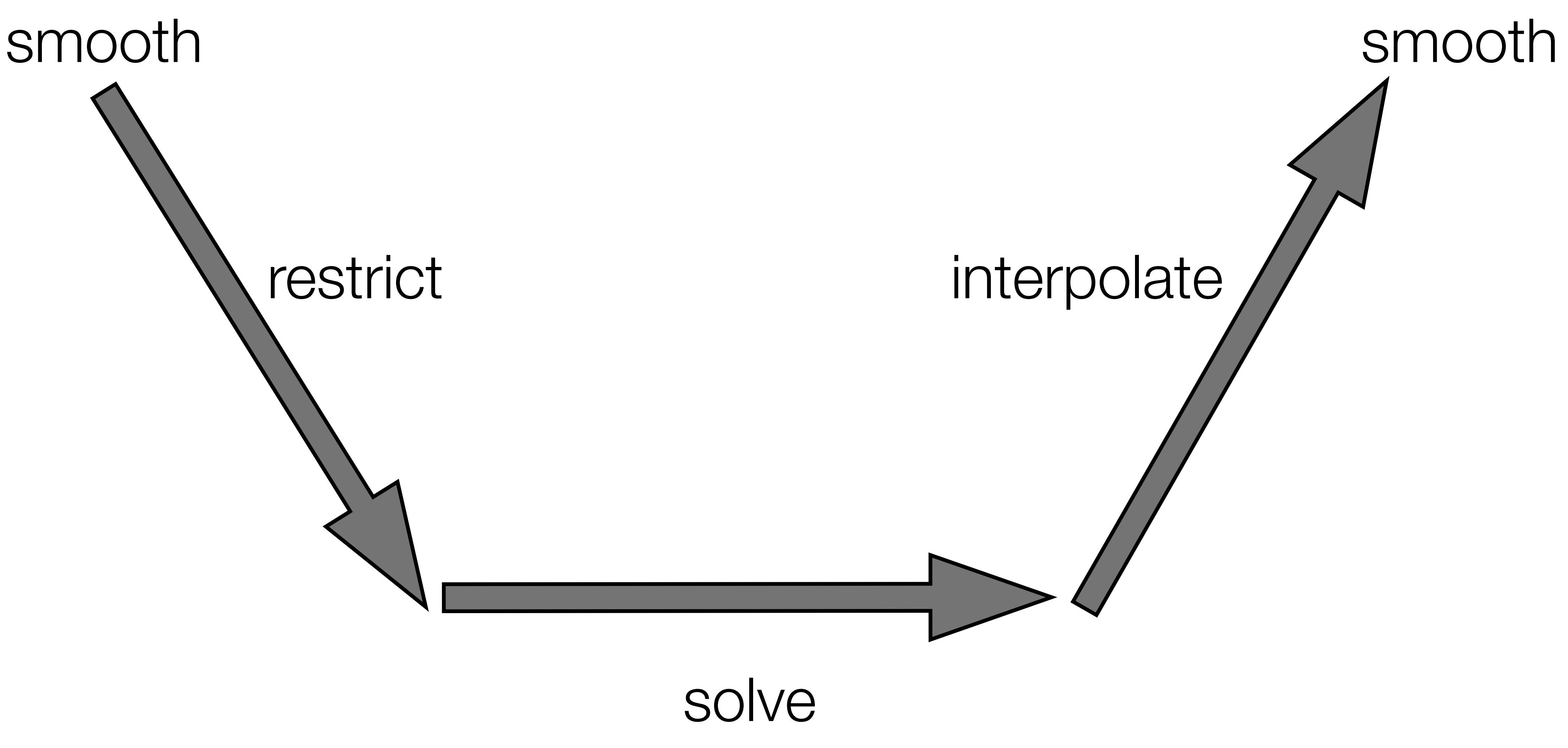
$$x_1 \leftarrow x_0 + P A_{2h}^{-1} R r_0$$

Take a look: `07-multigrid-two-level-matrix-form.ipynb`

Algorithm: two-level multigrid

Input: initial guess $P^T = R$

1. Smooth ν_{pre} times on $Au = f$
2. Compute $r = f - Au$
3. Compute $r_c = Rr$
4. Solve $A_c e_c = r_c$
5. Interpolate $\hat{e} = P e_c$
6. Correct $u \leftarrow u + \hat{e}$
7. Smooth ν_{post} times on $Au = f$



A two-level “V” cycle

How Accurate is Multigrid?

- Consider the exact solution to the PDE u^*

$$-u'' = f$$

- The exact solution to the **discrete** problem u_h^*

$$Au = b$$

- The approximate **discrete** solution $u_h \approx u_h^*$

- Define

$u^* - u_h^*$ Discretization error

$u_h^* - u_h$ Algebraic error

How Accurate should Multigrid aim to be?

- Would like the total error bounded

$$\begin{aligned}\|u^* - u_h\| &\le \overbrace{\|u^* - u_h^*\|}^{\text{disc.}} + \overbrace{\|u_h^* - u^h\|}^{\text{alg.}} \\ &\le \varepsilon\end{aligned}$$

- To achieve this,

  —force the discretization (grid space) so that

$$\|u^* - u_h^*\| \le ch^2 \le \frac{\varepsilon}{2}$$

  —and the algebraic error (convergence) up to the discretization error

$$\|u_h^* - u^h\| \le \frac{\varepsilon}{2}$$

Multigrid convergence

- Convergence factor of a cycle — the factor by which the error (residual) is reduced (in some norm) in each iteration

$$\gamma$$

$$\gamma^m = c\, n^{-2}$$

…assume this is independent of n

$$m \log \gamma = \log c - 2 \log n$$

- Wish to have m cycles such that

$$\gamma^m \sim \mathcal{O}(n^{-2}) \leq \frac{\varepsilon}{2}$$

- Then we need

$$m \sim \mathcal{O}(\log n)$$

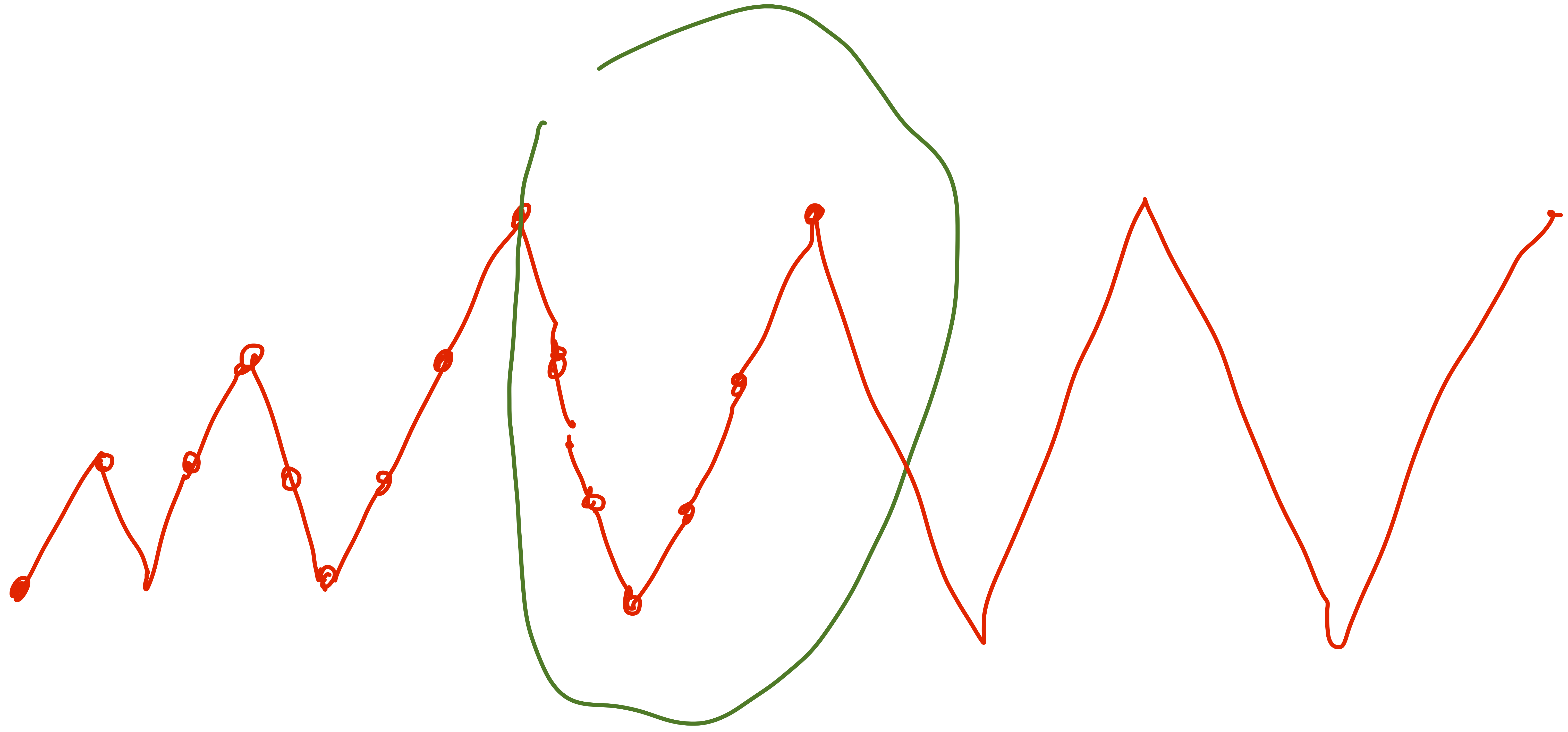
$$W = O(n \log n)$$

cycles

want $W = O(n)$

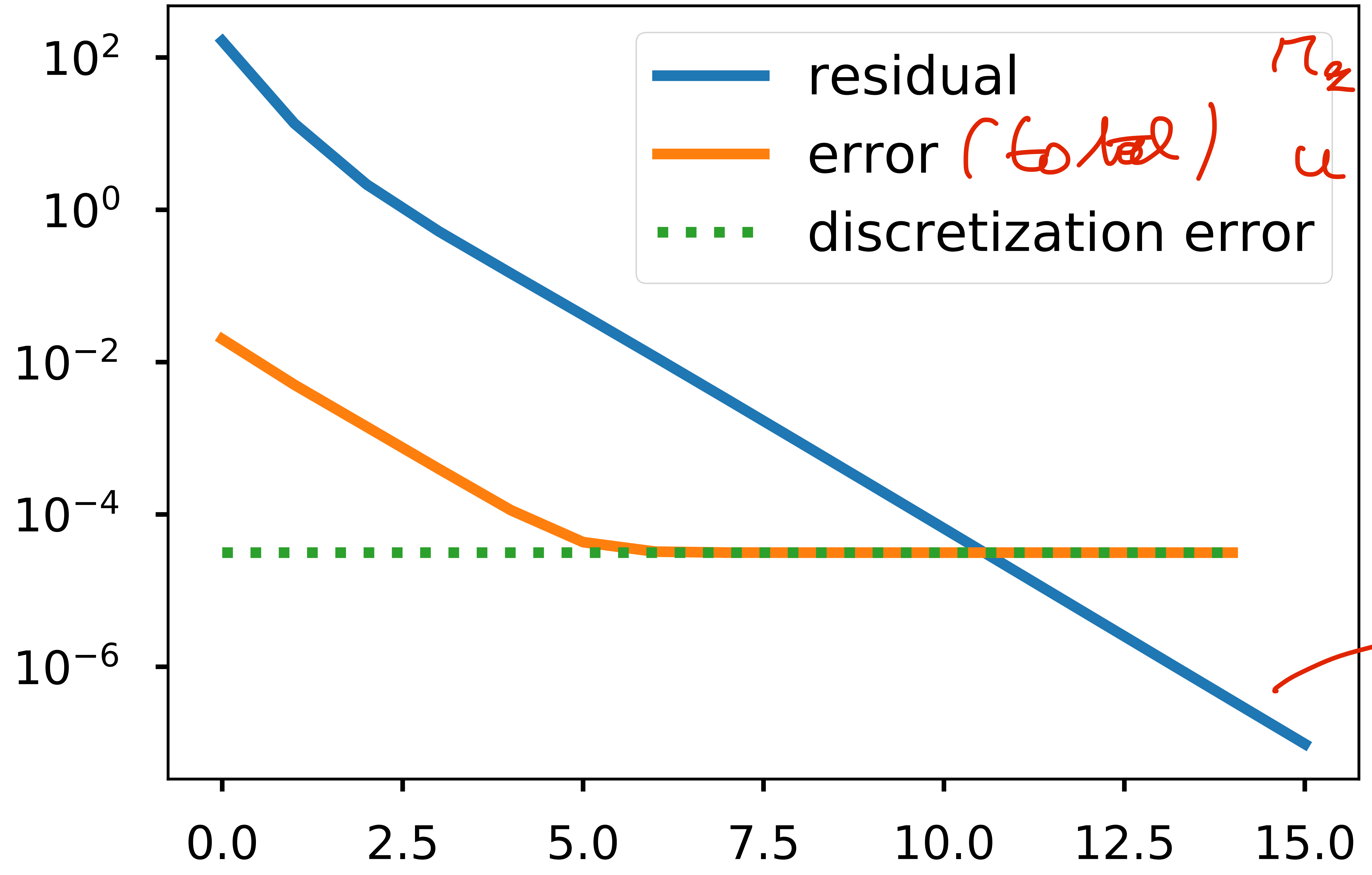
Full Multigrid (FMG)

$W = O(n)$



Take a look: `08-multigrid-v-cycle.ipynb`

Algebraic error



$r_\ell = b - A x_\ell$

$u^* - u_\ell$

$u^* - u^*_\ell$

$A(u^*_\ell - u_\ell)$

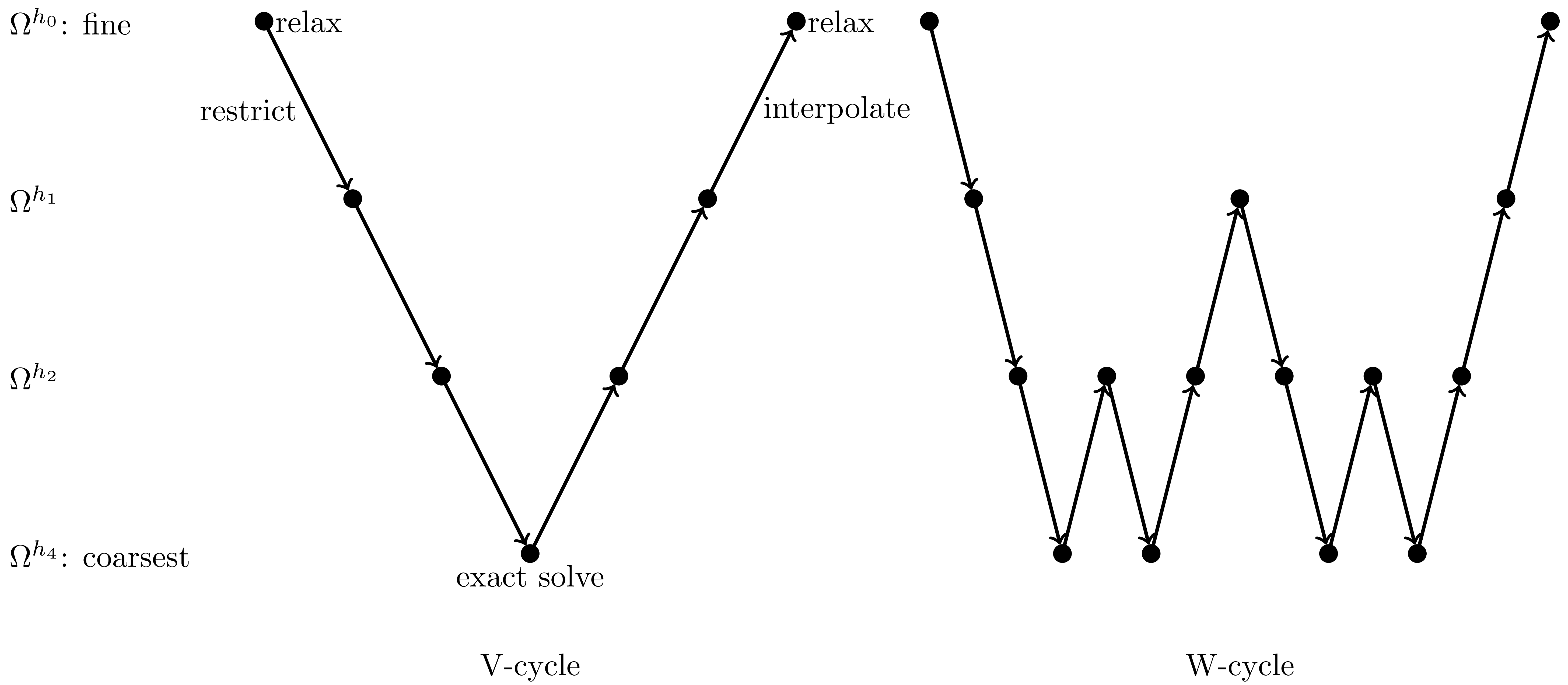
u^*

u^*_ℓ

u_ℓ

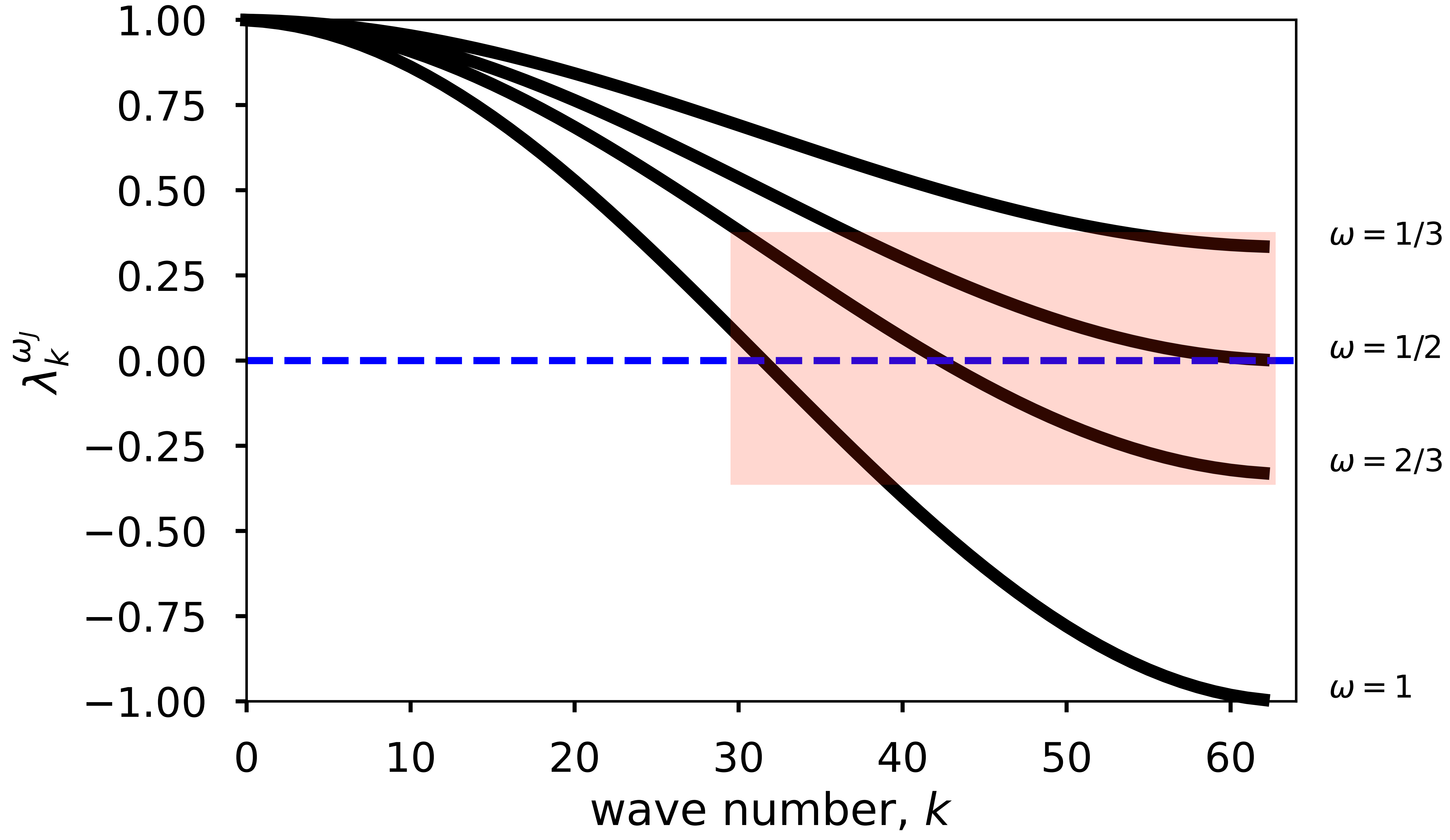
- The total error is limited by the discretization error

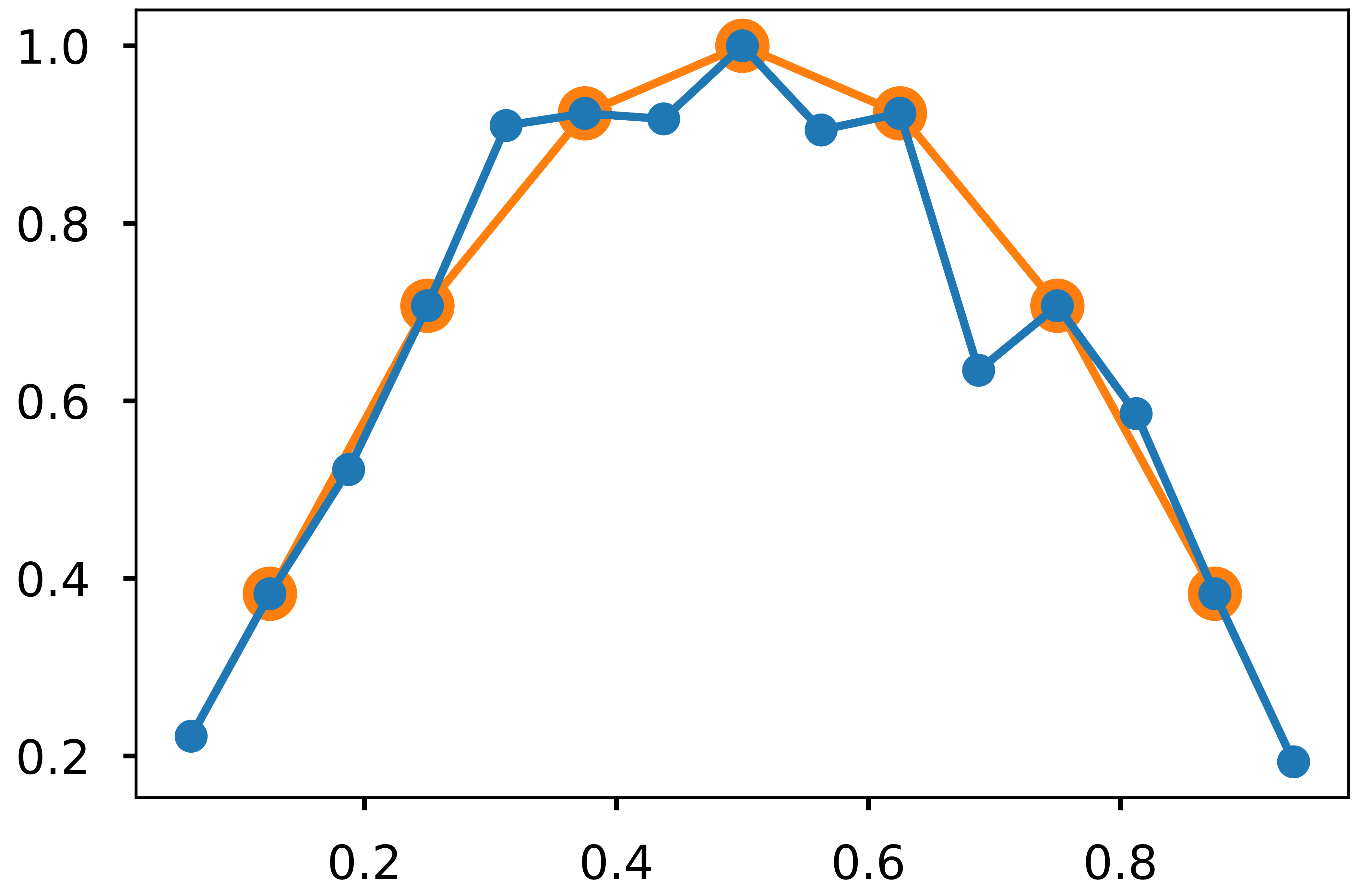
The Multigrid V-Cycle and W-Cycle



- W-Cycle can account for inadequate coarse level solves

Recap…



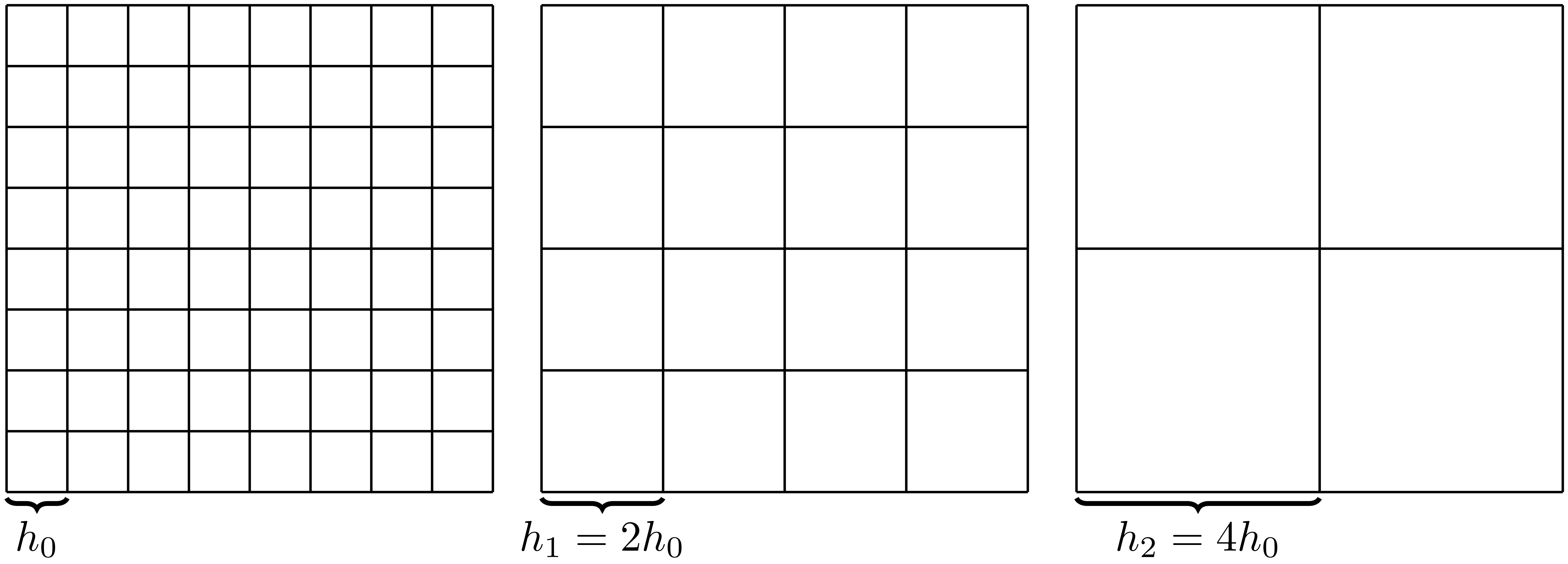


- Smoothing:
  Reduce high frequency error

- Coarse-grid correction:
  Reduce smooth-ish things in the range of interpolation

$$e_1 = e_0 - P(P^TAP)^{-1}P^TA\underbrace{Ge}_{e_0}$$

Multigrid in 2D



- Again, assume we have a sequence of uniform grids
- Relaxation remains the same (what is ω?)

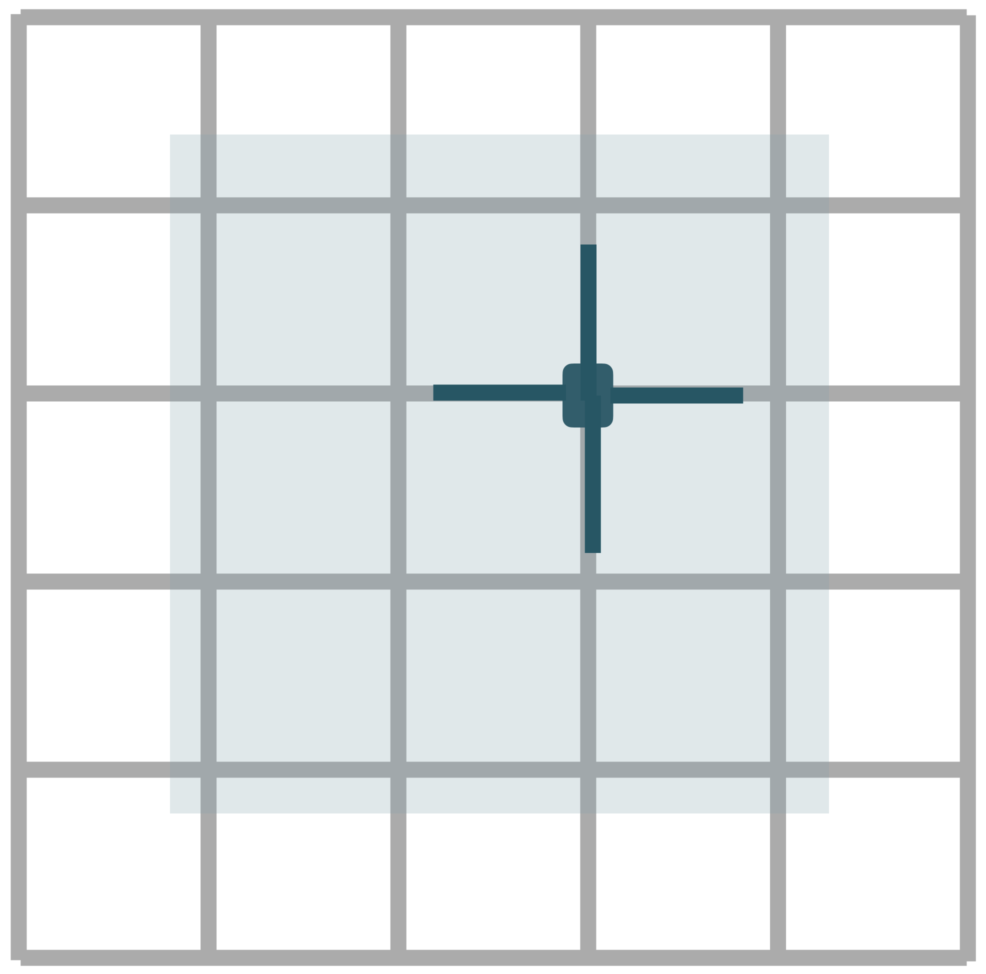
Multigrid in 2D

- Model problem

$$
\begin{aligned}
-u_{xx} - u_{yy} &= f \\
u &= 0 \qquad \text{on boundary}
\end{aligned}
$$

nnz $\approx$ 5n

- Results in the stencil / matrix

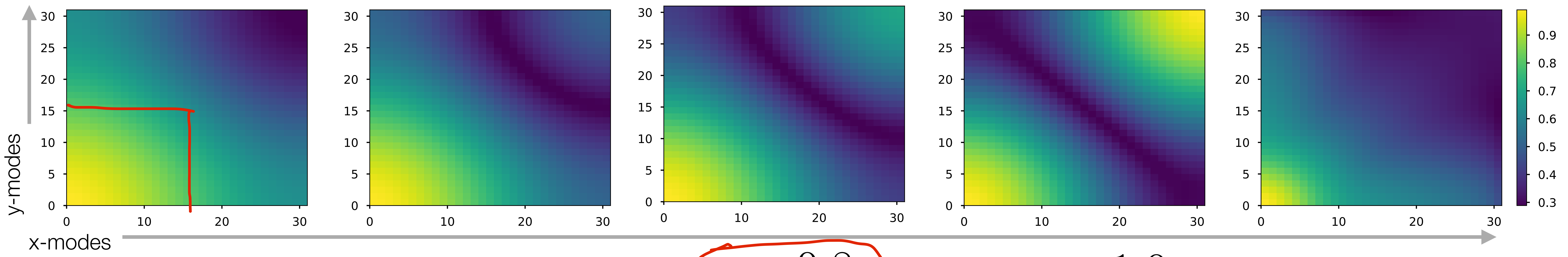


$$
\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}
\rightarrow
\frac{1}{h^2}
\begin{bmatrix}
4 & -1 & & & -1 & & & & & & & & & & & \\
-1 & 4 & -1 & & & -1 & & & & & & & & & & \\
& -1 & 4 & -1 & & & -1 & & & & & & & & & \\
& & -1 & 4 & & & & -1 & & & & & & & & \\
-1 & & & & 4 & -1 & & & -1 & & & & & & & \\
& -1 & & & -1 & 4 & -1 & & & -1 & & & & & & \\
& & -1 & & & -1 & 4 & -1 & & & -1 & & & & & \\
& & & -1 & & & -1 & 4 & & & & -1 & & & & \\
& & & & -1 & & & & 4 & -1 & & & -1 & & & \\
& & & & & -1 & & & -1 & 4 & -1 & & & -1 & & \\
& & & & & & -1 & & & -1 & 4 & -1 & & & -1 & \\
& & & & & & & -1 & & & -1 & 4 & & & & -1 \\
& & & & & & & & -1 & & & & 4 & -1 & & \\
& & & & & & & & & -1 & & & -1 & 4 & -1 & \\
& & & & & & & & & & -1 & & & -1 & 4 & -1 \\
& & & & & & & & & & & -1 & & & -1 & 4
\end{bmatrix}
$$

Relaxation in 2D

$$\sin\left(\frac{k_x i\pi}{n+1}\right)\sin\left(\frac{k_y j\pi}{n+1}\right)$$

Convergence factor over 10 sweeps



$\omega = 0.5$ $\quad \omega = 0.667$ $\quad \omega = 0.8$ $\quad \omega = 1.0$ $\quad$ GS

$$I - \omega D^{-1}A \qquad\qquad I - (D-E)^{-1}A$$

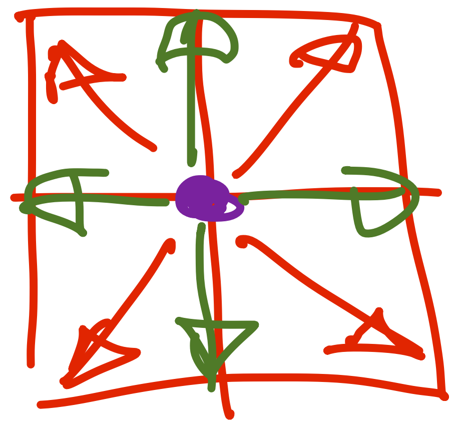
- weighted Jacobi: Same issue — need to select a parameter
- Gauss-Seidel improved
- Red-Black Gauss-Seidel and other schemes even more effective

Take a look: `09-multigrid-2d.ipynb`

Interpolation in 2D

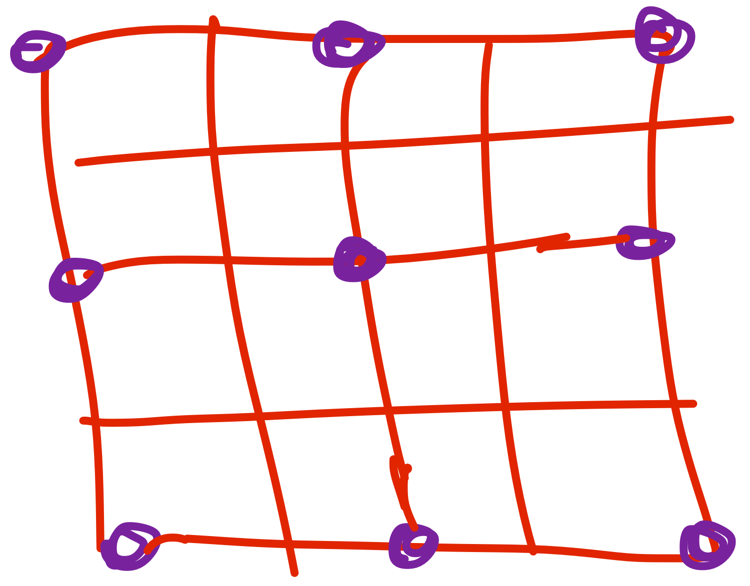
- Bilinear interpolation, tensor product of 1D interpolation

$$\frac{1}{2}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \longrightarrow \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



• is coarse

- Example 3x3 -> 7x7



$$\begin{bmatrix} 0.5 & & \\ 1 & & \\ 0.5 & 0.5 & \\ & 1 & \\ & 0.5 & 0.5 \\ & & 1 \\ & & 0.5 \end{bmatrix} \otimes \begin{bmatrix} 0.5 & & \\ 1 & & \\ 0.5 & 0.5 & \\ & 1 & \\ & 0.5 & 0.5 \\ & & 1 \\ & & 0.5 \end{bmatrix} \longrightarrow \frac{1}{4}\begin{bmatrix} 1 & & & & & \\ 2 & & & & & \\ 1 & 1 & & & & \\ & 2 & & & & \\ & 1 & 1 & & & \\ & & 2 & & & \\ & & 1 & & & \\ 2 & & & & & \\ 4 & & & & & \\ 2 & 2 & & & & \\ & 4 & & & & \\ & 2 & 2 & & & \\ & & 4 & & & \\ & & 2 & & & \\ 1 & & & 1 & & \\ 2 & & & 2 & & \\ 1 & 1 & & 1 & 1 & \\ & 2 & & & 2 & \\ & 1 & 1 & & 1 & 1 \\ & & 2 & & & 2 \\ & & 1 & & & 1 \\ & & & 2 & & \\ & & & 4 & & \\ & & & 2 & 2 & \\ & & & & 4 & \\ & & & & 2 & 2 \\ & & & & & 4 \\ & & & & & 2 \\ & \vdots & & \vdots & & \end{bmatrix}$$

A few observations so far: **One**

- Let's consider a V(1,1) cycle — weighted Jacobi, etc.

smooth

smooth

- The **error** propagation for this looks like

$$e_1 = \underbrace{G\overbrace{(I - P(P^TAP)^{-1}P^TA)}^{\text{CGC}}G}_{M}e_0 \qquad G = I - \omega D^{-1}A$$

before smoothing

$$Me_k \not\to 0?$$

- One thing we can do, is consider bounds on each operation.

A few observations so far: **One**

- Take the operator

$$M = G(I - P(P^T AP)^{-1} P^T A)G$$

- And makes some bounds

$$\underbrace{\|I - P(P^T AP)^{-1} P^T A\|}_{\text{approximation property}} \underbrace{\|G\|^2}_{\text{smoothing property}}$$

- General $\|G\| \le 1$

1D over [n/2,n]: $\|G\| \le \dfrac{1}{3}$

2D over [n/2,n]: $\|G\| \le \dfrac{3}{5}$

"Multigrid tutorial"

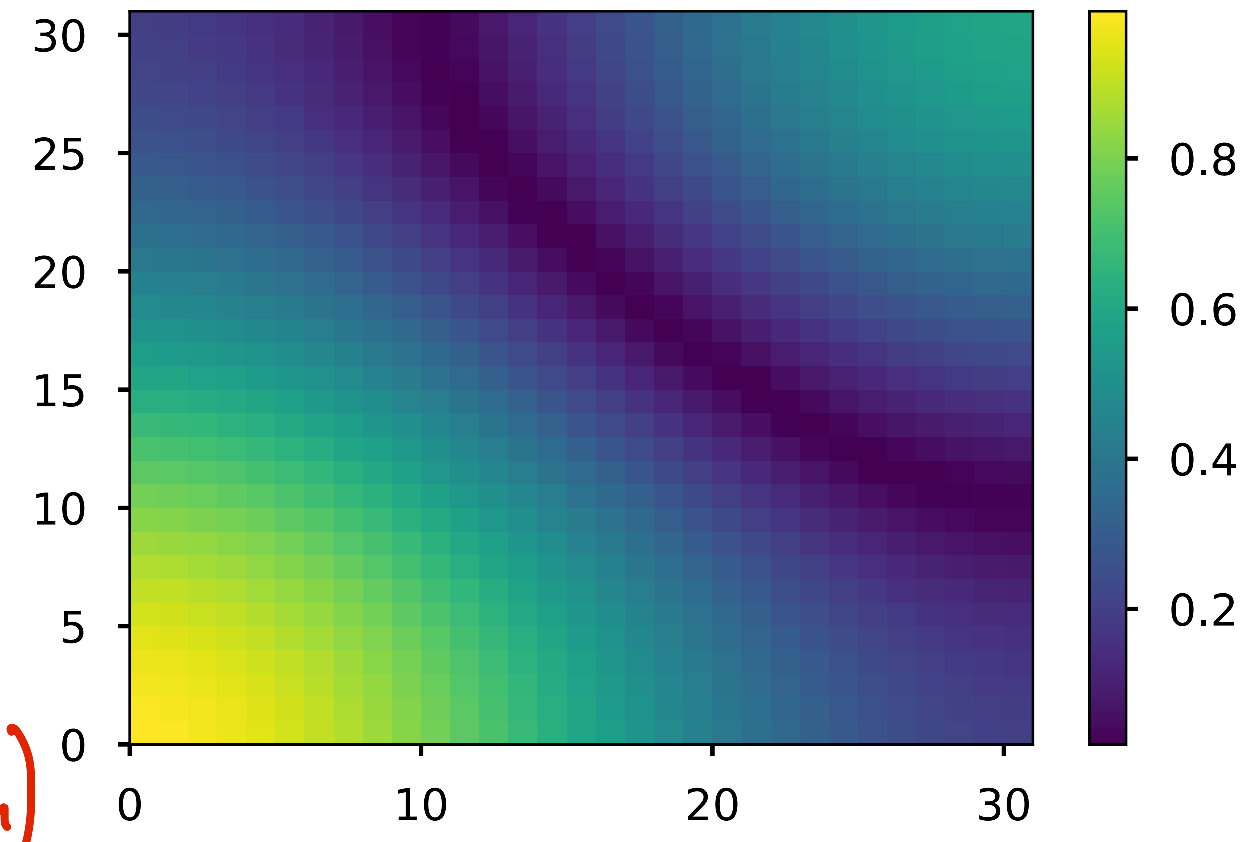
A few observations so far: **One**

$$\underbrace{\|I - P(P^TAP)^{-1}P^TA\|}_{\text{approximation property}}\underbrace{\|G\|^2}_{\text{smoothing property}}$$

- General $\|G\| \leq 1$

1D over [n/2,n]: $\|G\| \leq \frac{1}{3}$

2D over [n/2,n]: $\|G\| \leq \frac{3}{5}$



- Also, if w s.t. $Aw \in \mathcal{N}(P^T)$ (null space of restriction)

then $(I - P(P^TAP)^{-1}P^TA)w = w$

$\| \cdot \| \geq 1$

L error

A few observations so far: **Two**

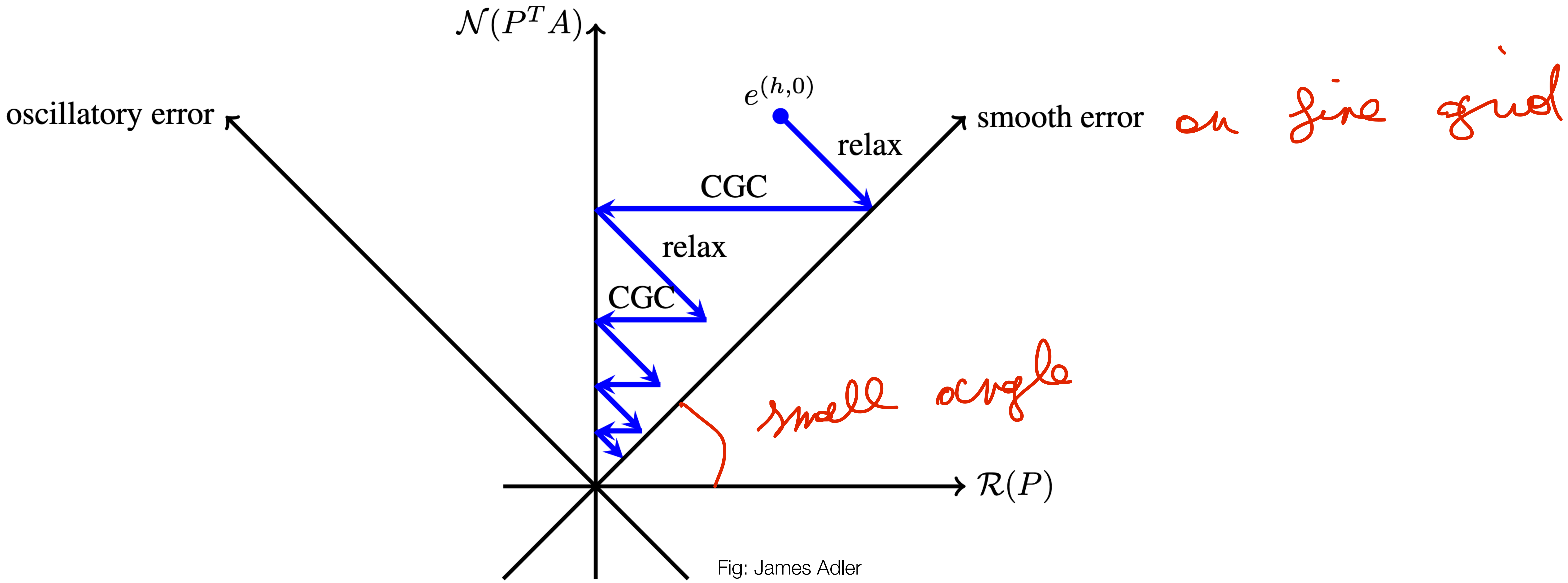


Fig: James Adler

- Complementary processes:
  - **relaxation:** targets (Fourier) smoothing
  - **coarse grid correction:** targets things in the range of interpolation

A few observations so far: **Three**

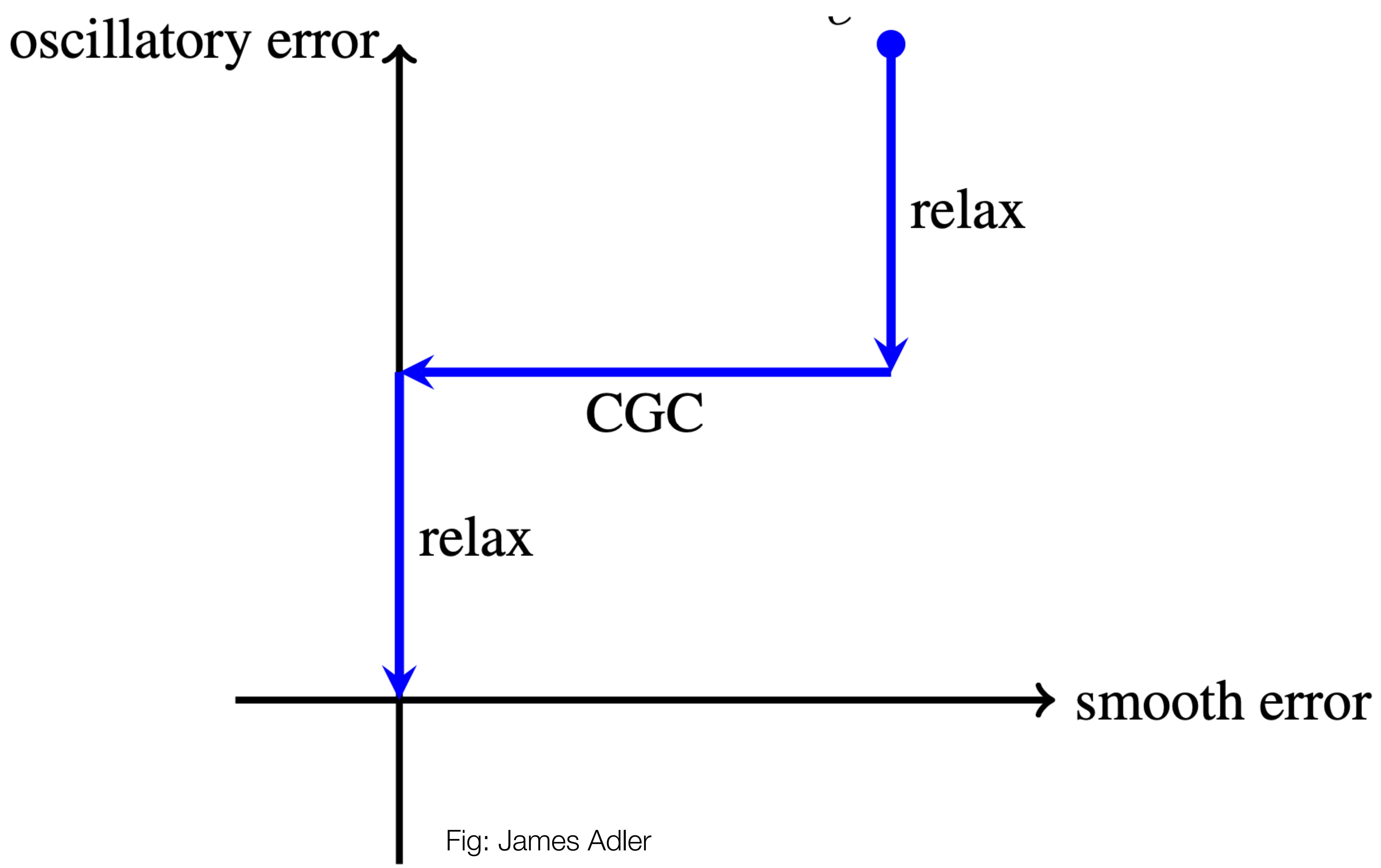


Fig: James Adler

$$e_1 \leftarrow (I - P(P^TAP)^{-1}P^TA)Ge_0$$

$$Ge_0 \in \mathcal{R}(P) \quad \Rightarrow \quad e_1 = 0$$

interpolation should capture what relaxation misses

Take a look: `10-multigrid-anisotropy.ipynb`

What can go wrong?!

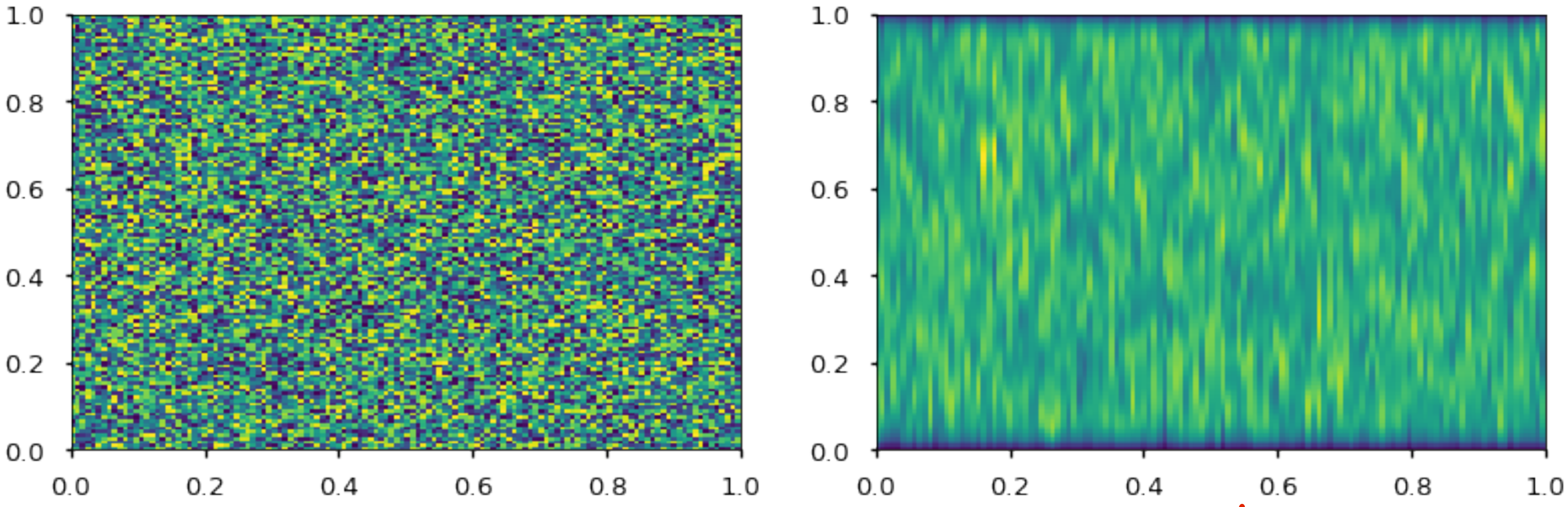
- Consider an *anisotropic problem*

after relaxation: smooth in y, not in x

$$-\varepsilon u_{xx} - u_{yy} = f$$

$$u = 0 \qquad \text{on boundary}$$

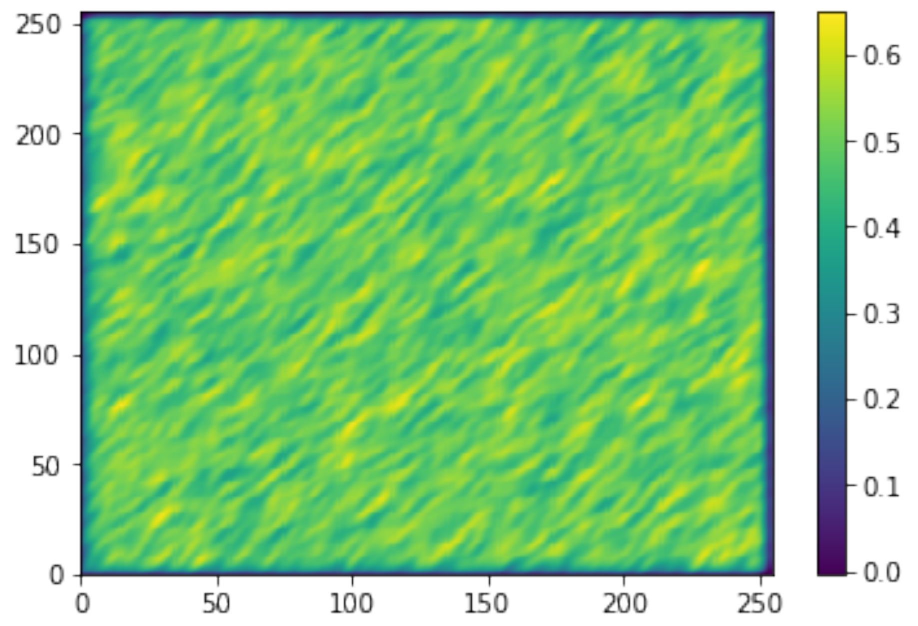
$$\begin{bmatrix} & -1 & \\ -\varepsilon & 2+2\varepsilon & -\varepsilon \\ & -1 & \end{bmatrix}$$

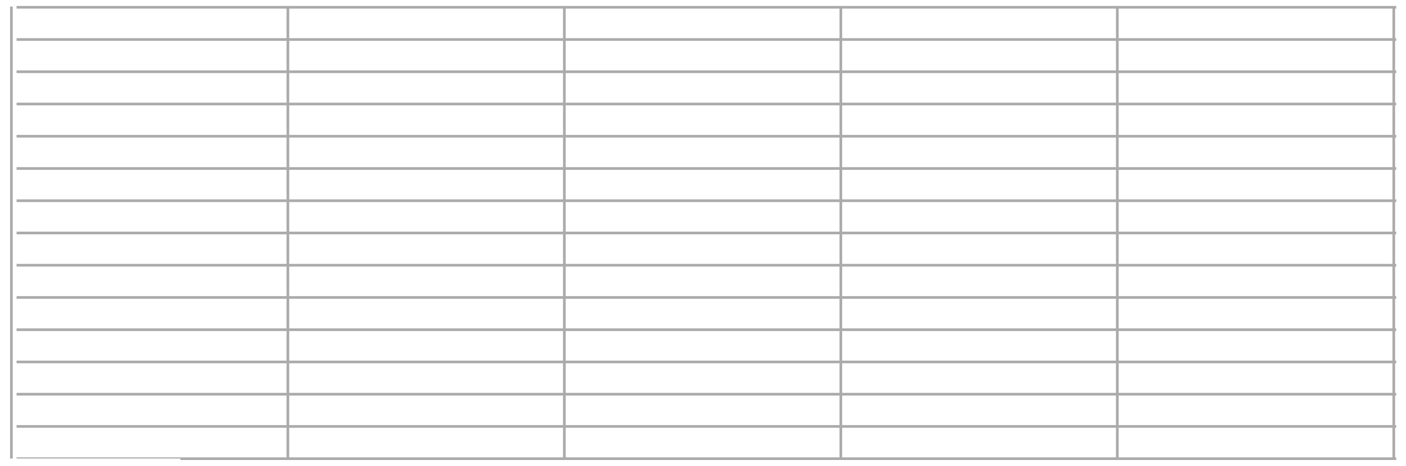


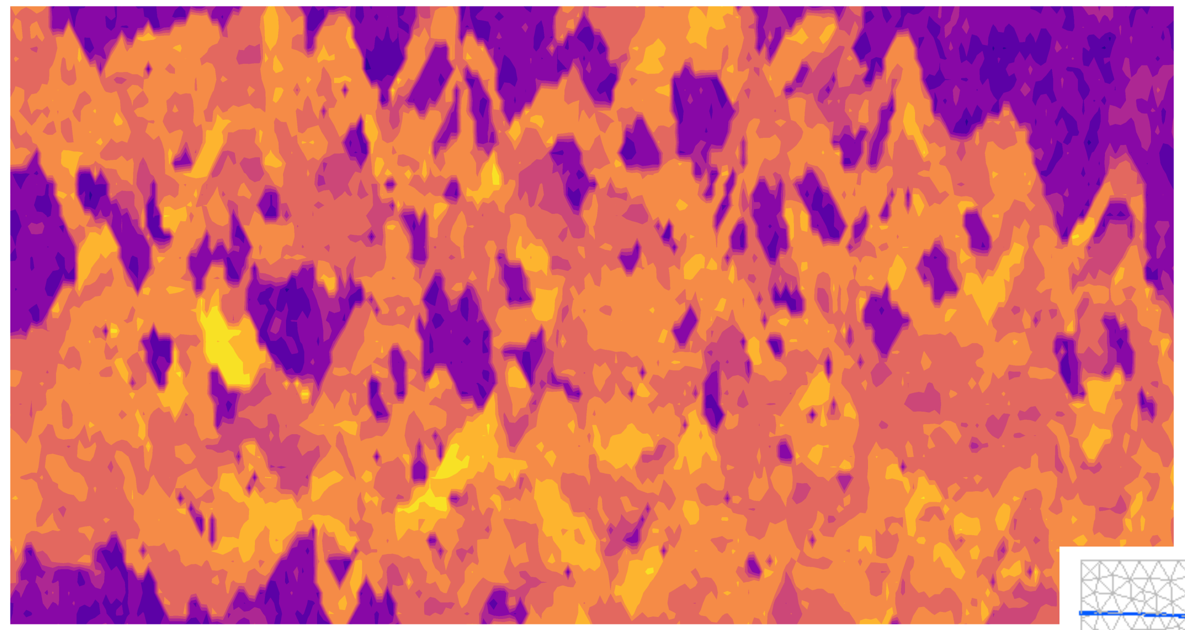
we cannot coarsen in x !

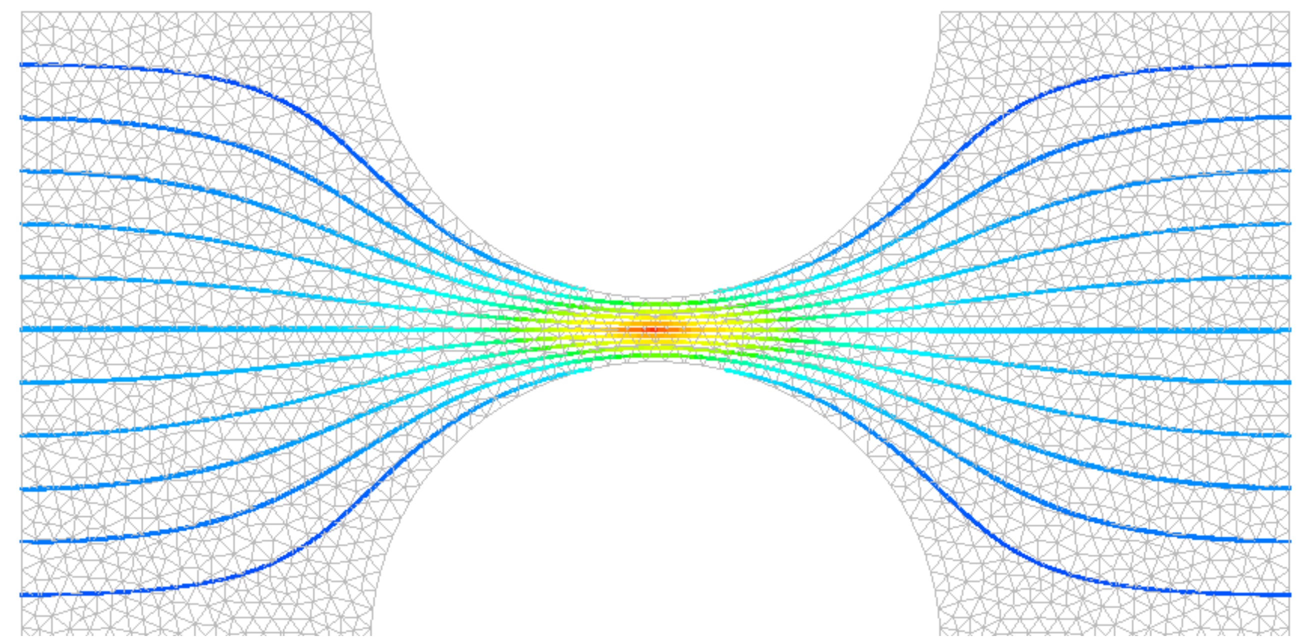
What can go wrong?!

- Anisotropy
- Mesh stretching
- Jumping coefficients
- Non-elliptic







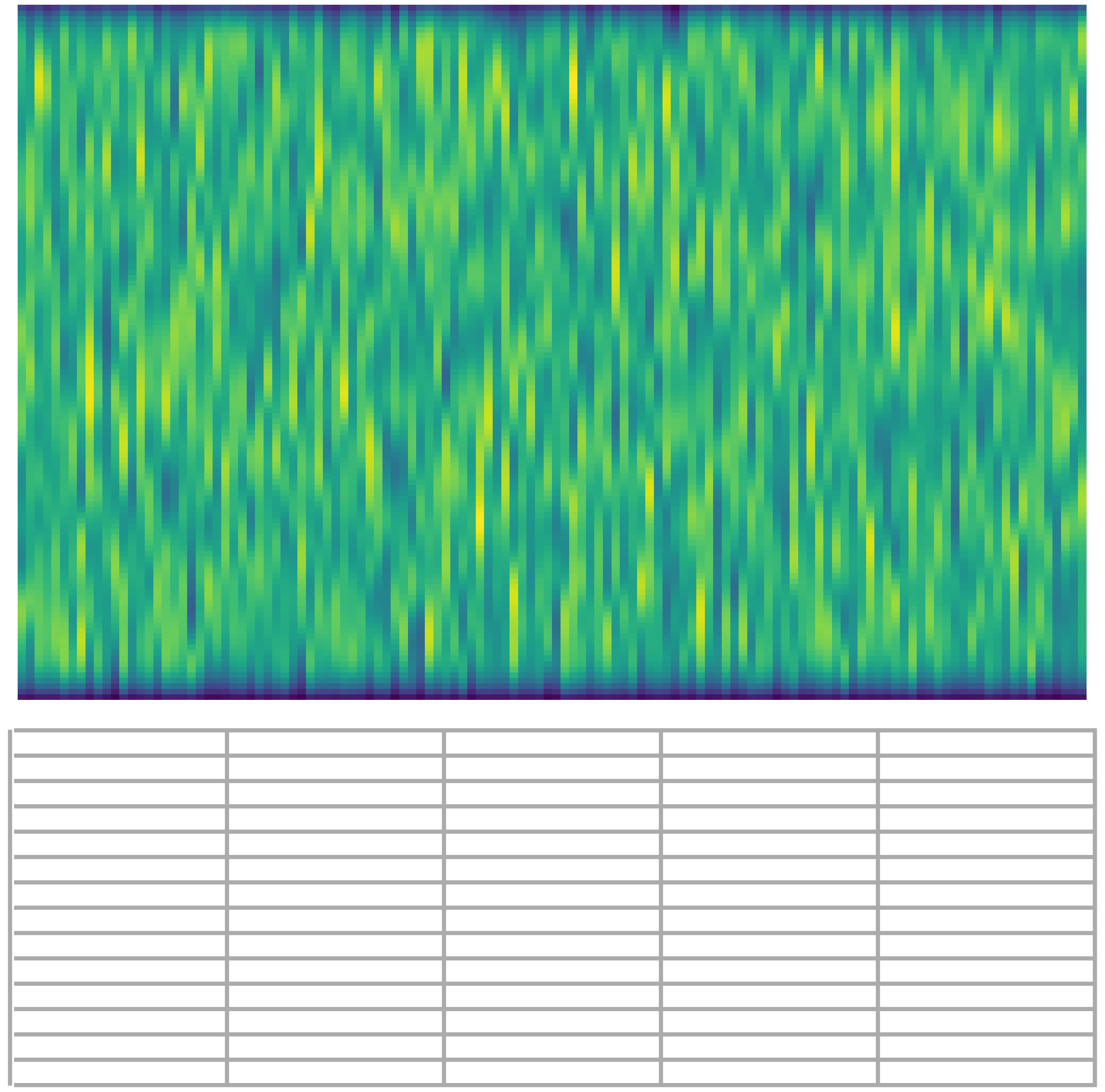


Options for more robust Multigrid

$-\varepsilon u_{xx} - u_{yy} = f$

- **Semicoarsening:** Coarsen in the direction of smoothness

(change coarsening)

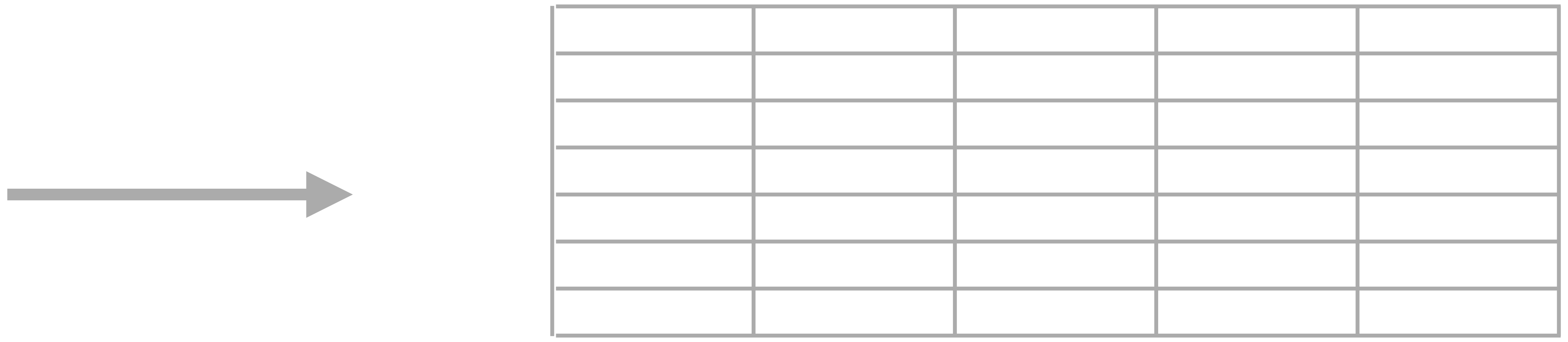


don't coarsen in x

only coarsen in y

(then CGC will be OK)

(on special coarse grid)



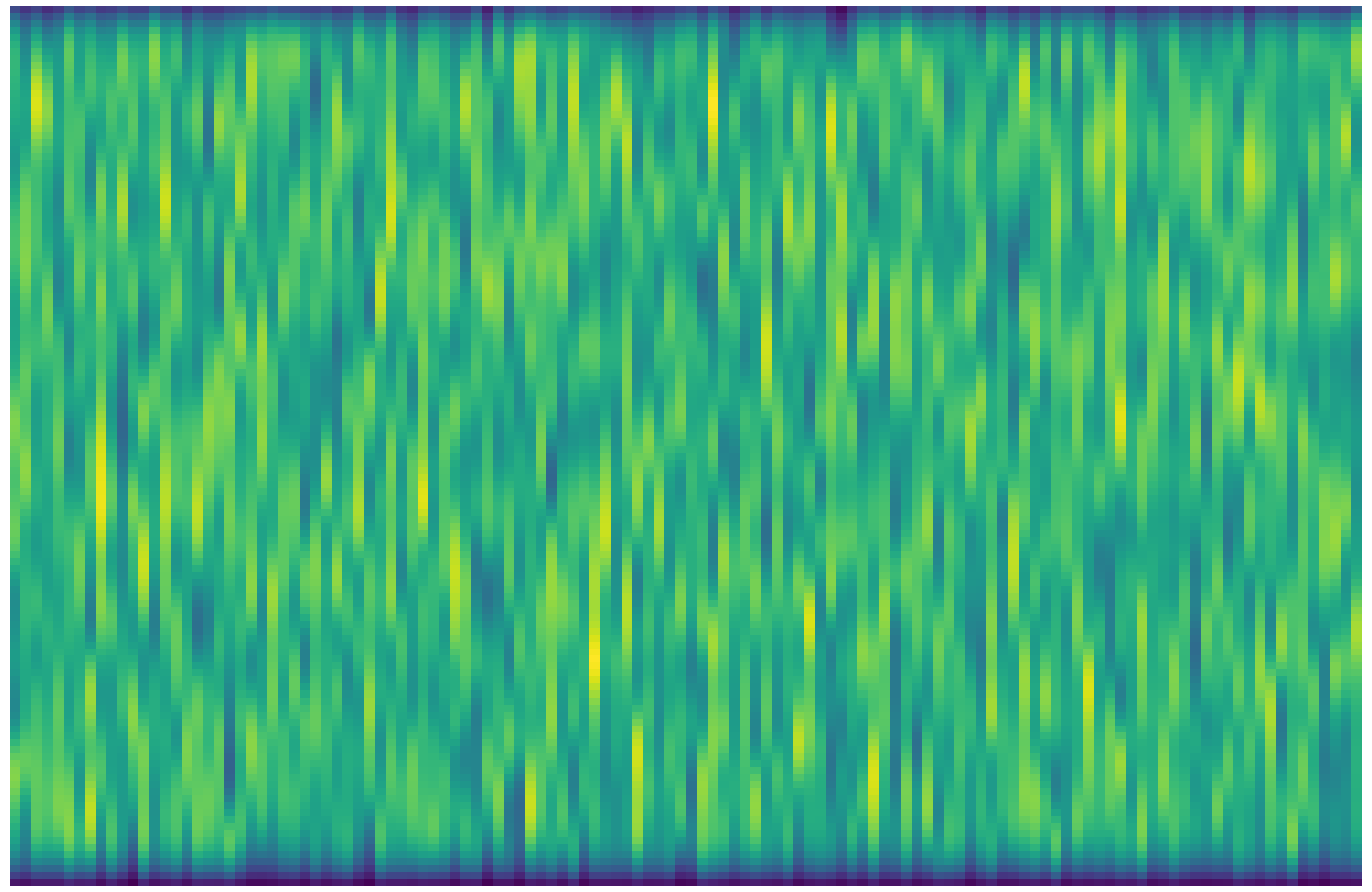
- Downside: if anisotropy varies in another direction, we need a different grid

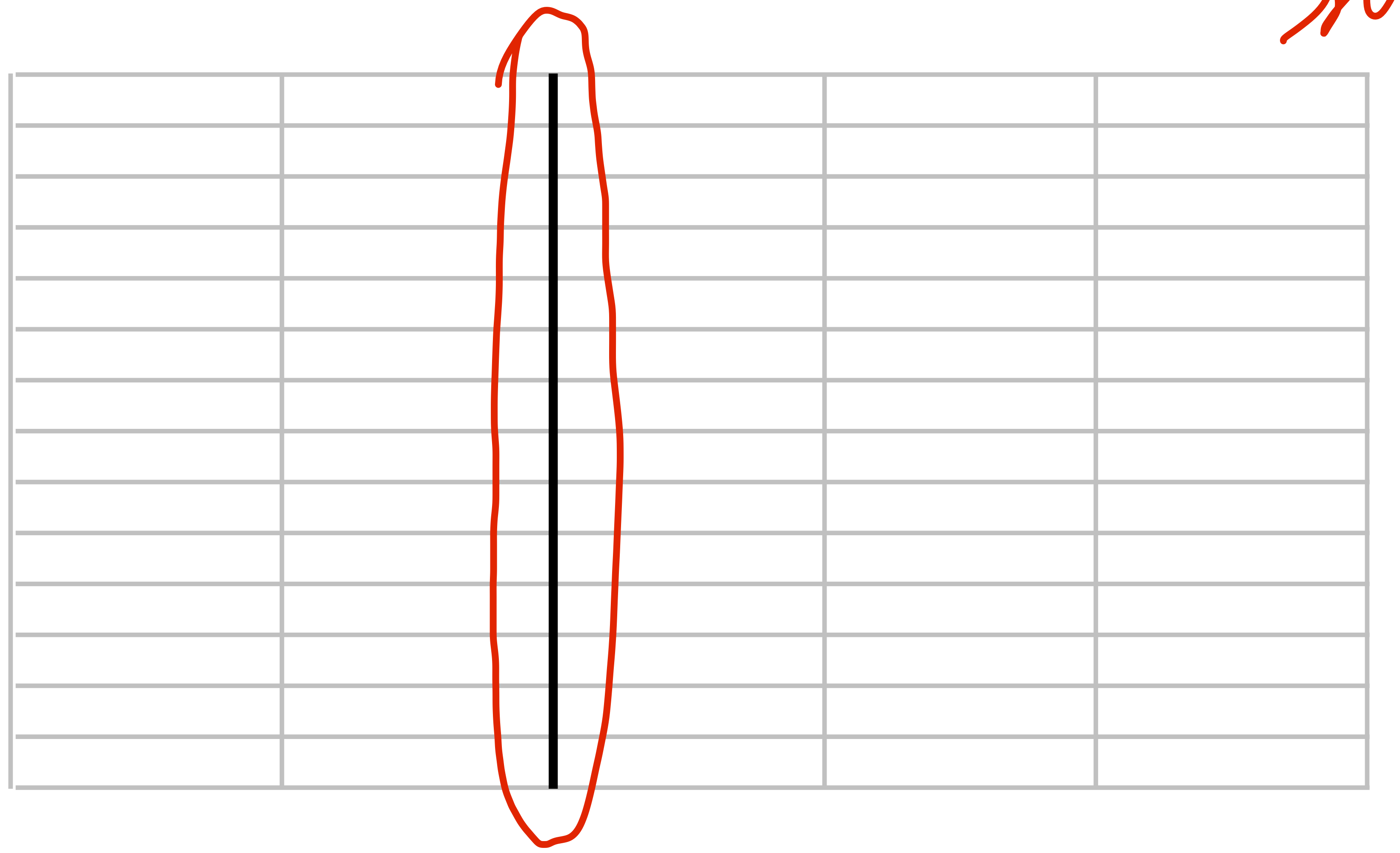
$-\varepsilon u_{xx} + u_{yy} = f$

Options for more robust Multigrid

- **Line/plane relaxation** (or: change relaxation)
  Perform relaxation in groups (in a line)

smooth these together (direction of strong connections)



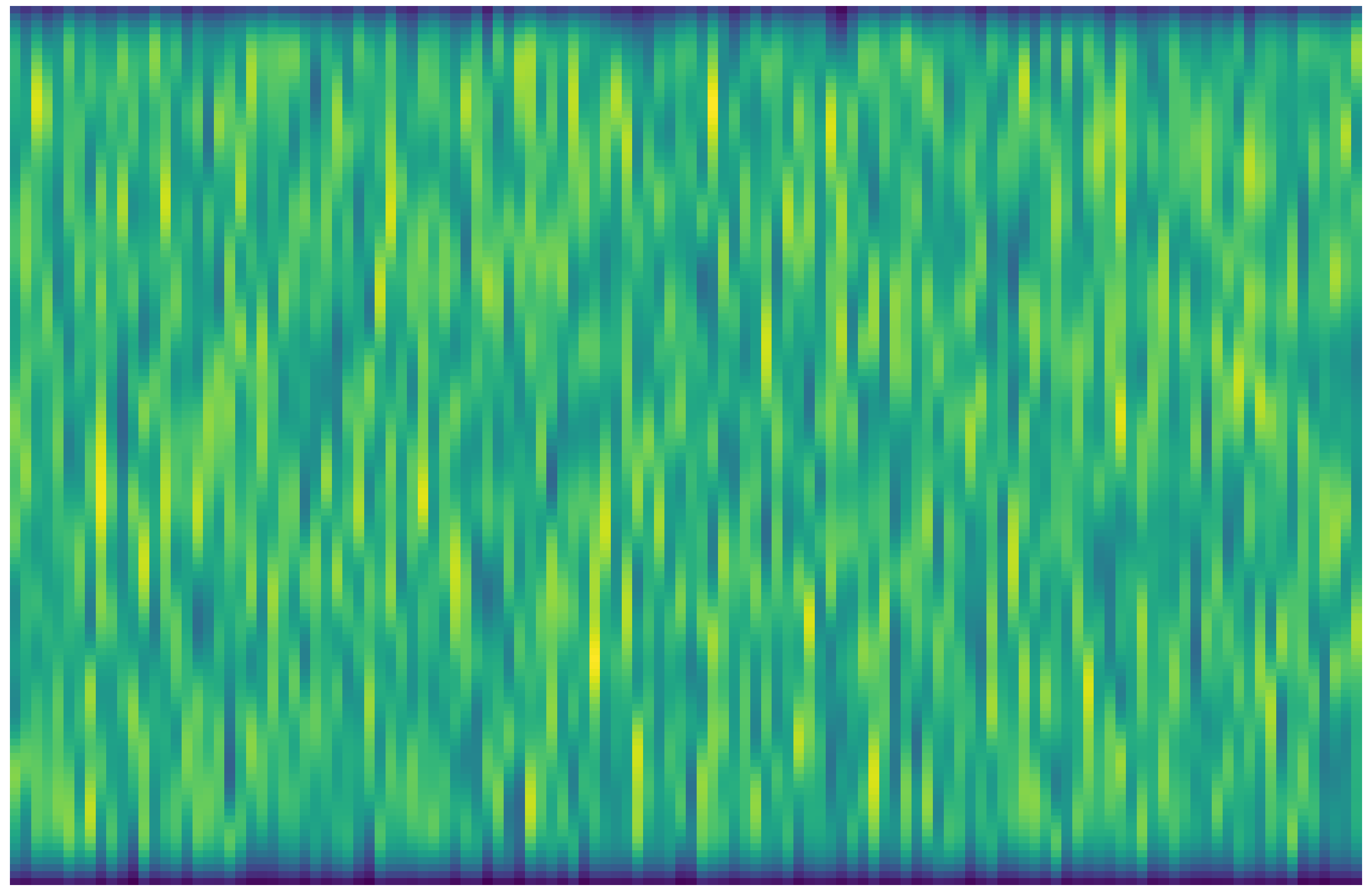


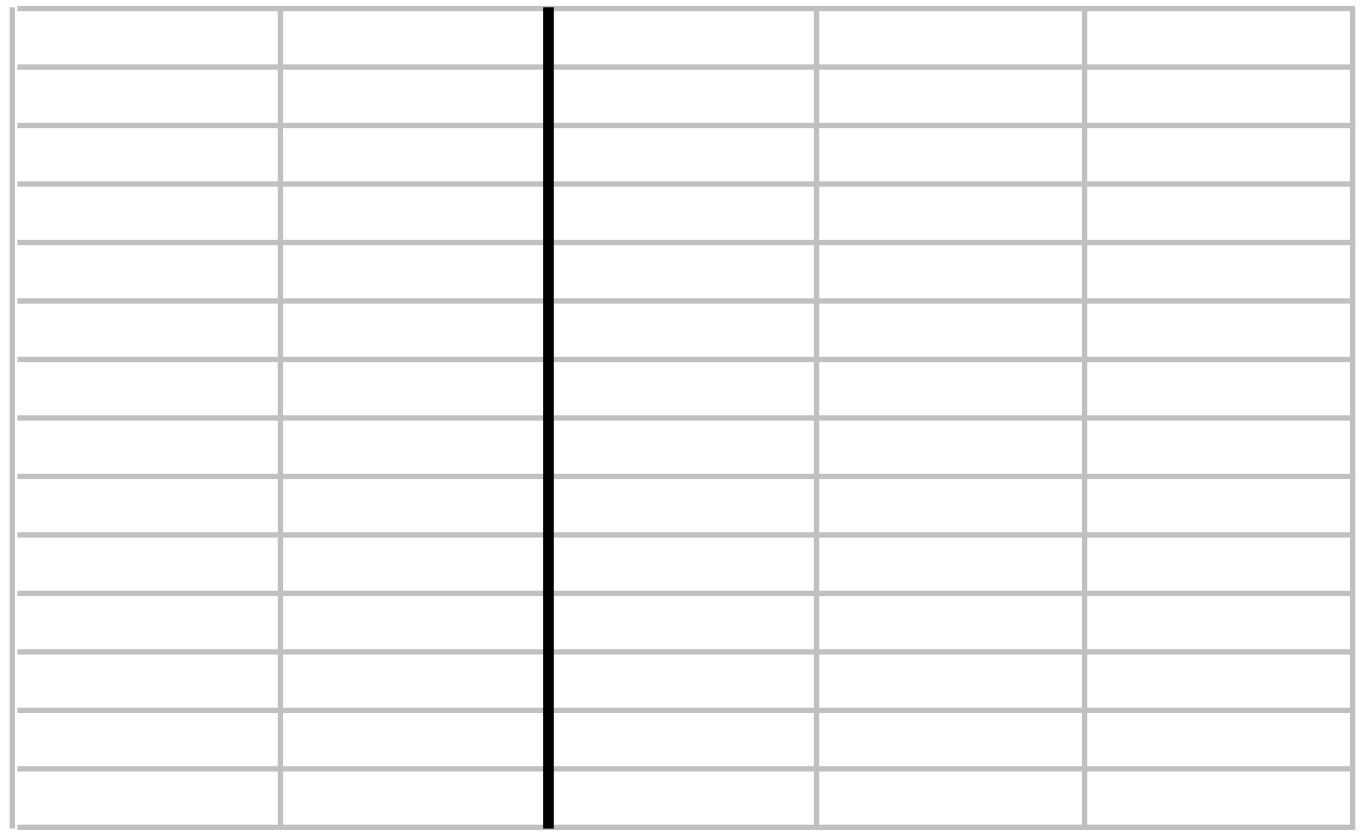
- Example:
  - relax error in block in the y-direction (constant x-lines)
  - no (block) smoothing in the x-direction (constant y-lines)

(then CGC will be OK on fully coarsened grid)

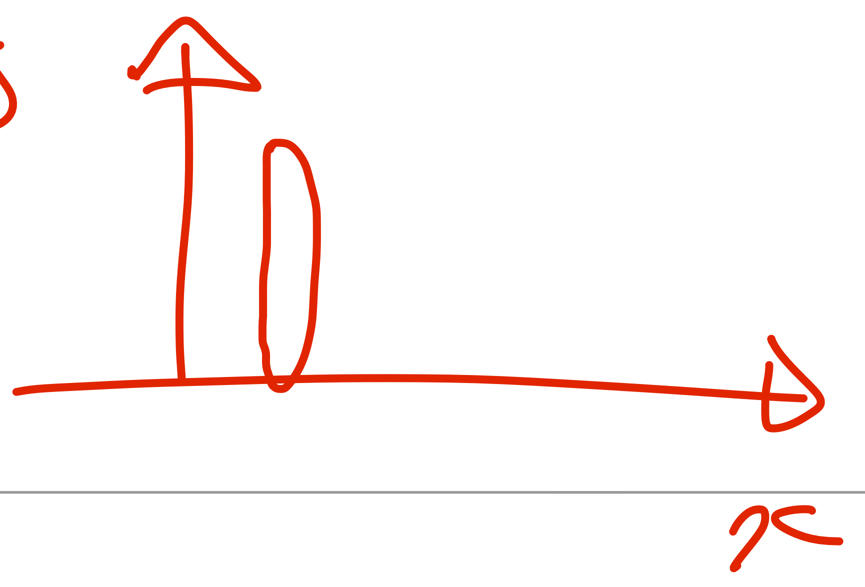
Options for more robust Multigrid

- **Line/plane relaxation**
- For each x-line (lines of strong anisotropy):
  - Eliminate the residual on the entire line
  - (Gauss-Seidel, by lines)





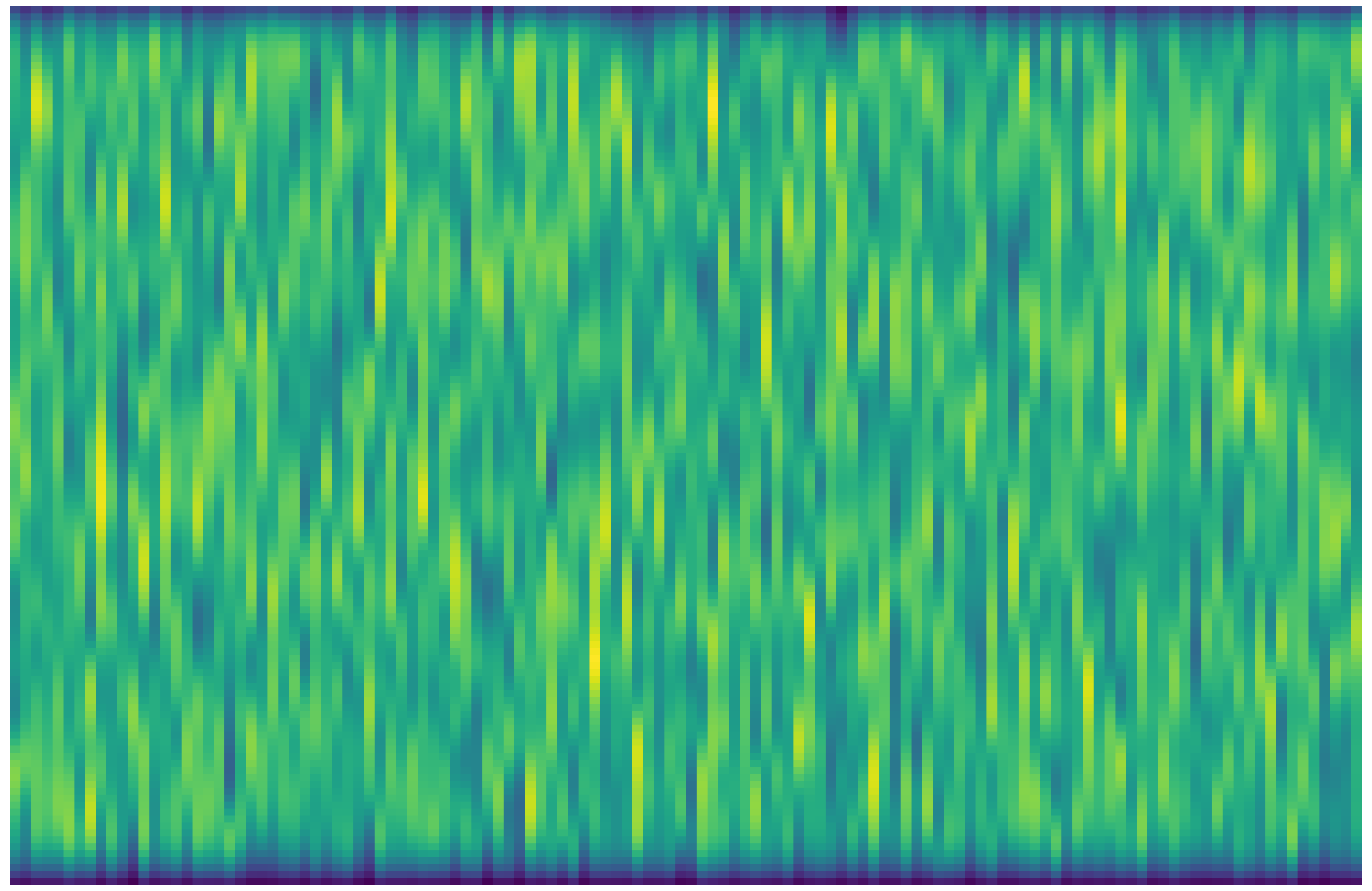
$-\varepsilon u_{xx} + u_{yy} = f$



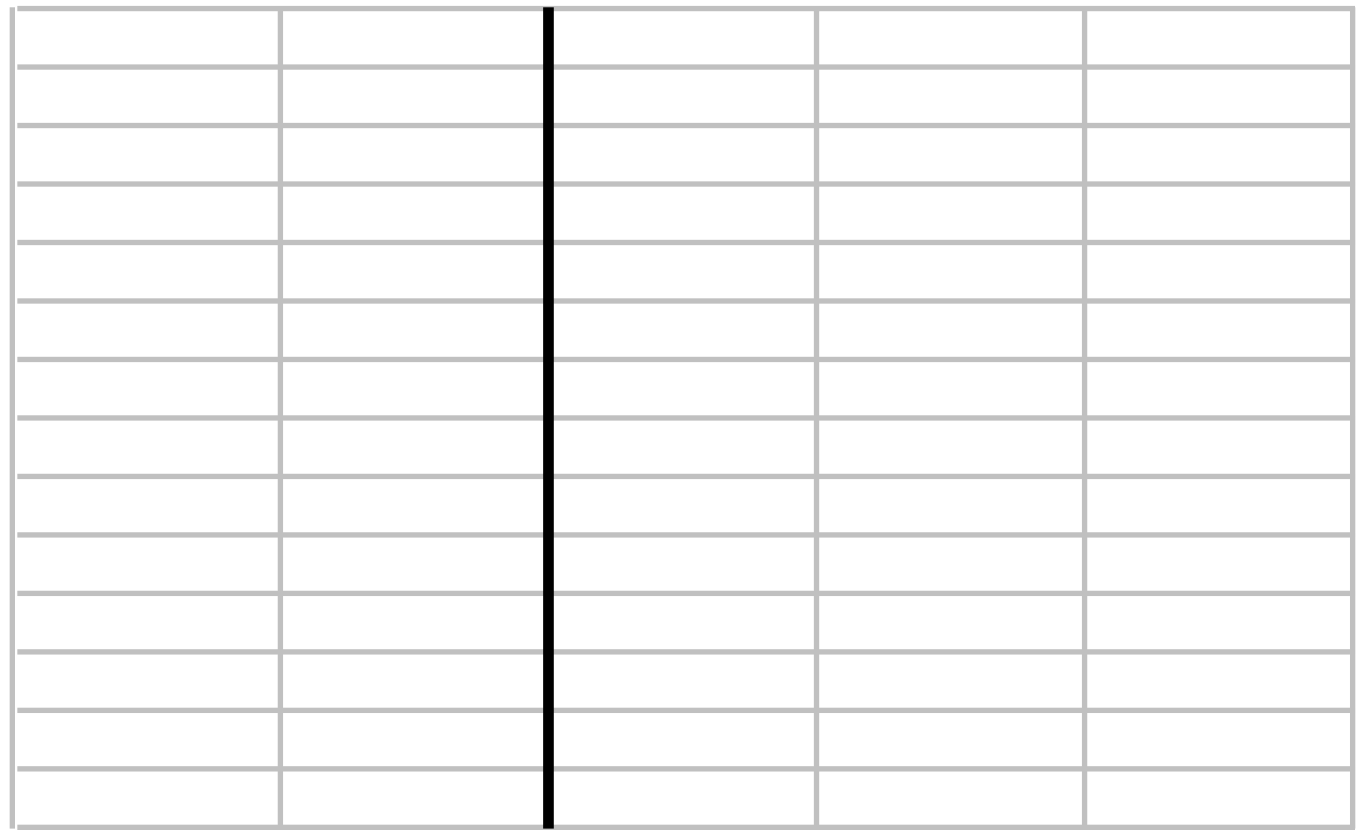
column - lexicographic

Options for more robust Multigrid

- **Line/plane relaxation**
- $$A = \frac{1}{(n+1)^2}\begin{bmatrix} A_{1D} & -\varepsilon I & & & \\ -\varepsilon I & A_{1D} & -\varepsilon I & & \\ & -\varepsilon I & A_{1D} & -\varepsilon I & \\ & & \ddots & & \\ & & & -\varepsilon I & A_{1D} \end{bmatrix}$$



$$A_{1D} = \frac{1}{(n+1)^2}\begin{bmatrix} 2+2\varepsilon & -1 & & \\ -1 & 2+2\varepsilon & -1 & \\ & & \ddots & \\ & & -1 & 2+2\varepsilon \end{bmatrix}$$



Take a look: `11-multigrid-2d-line-relaxation.ipynb`

Options for more robust Multigrid

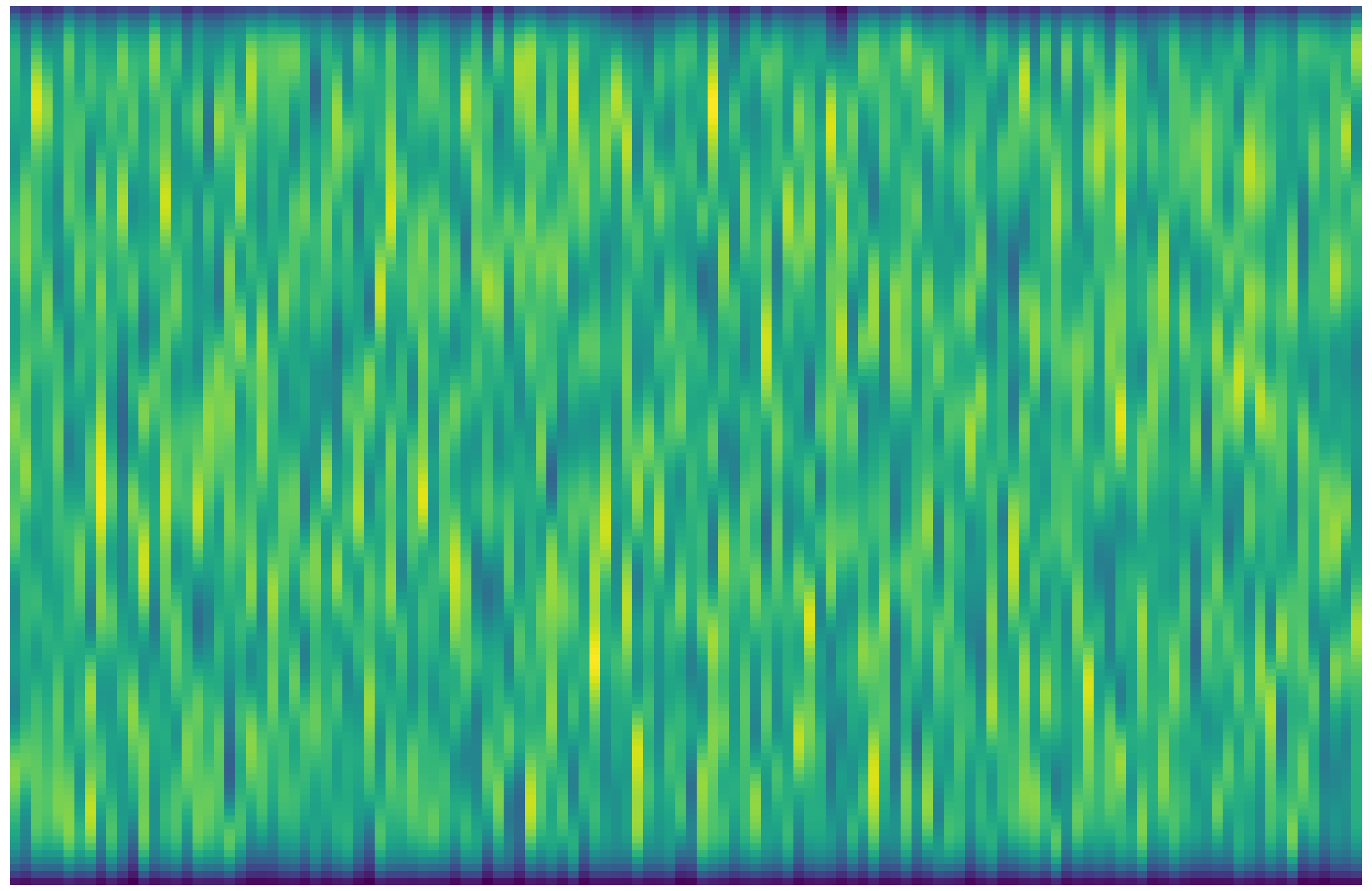
- **Line/plane relaxation**
- For each line, solve

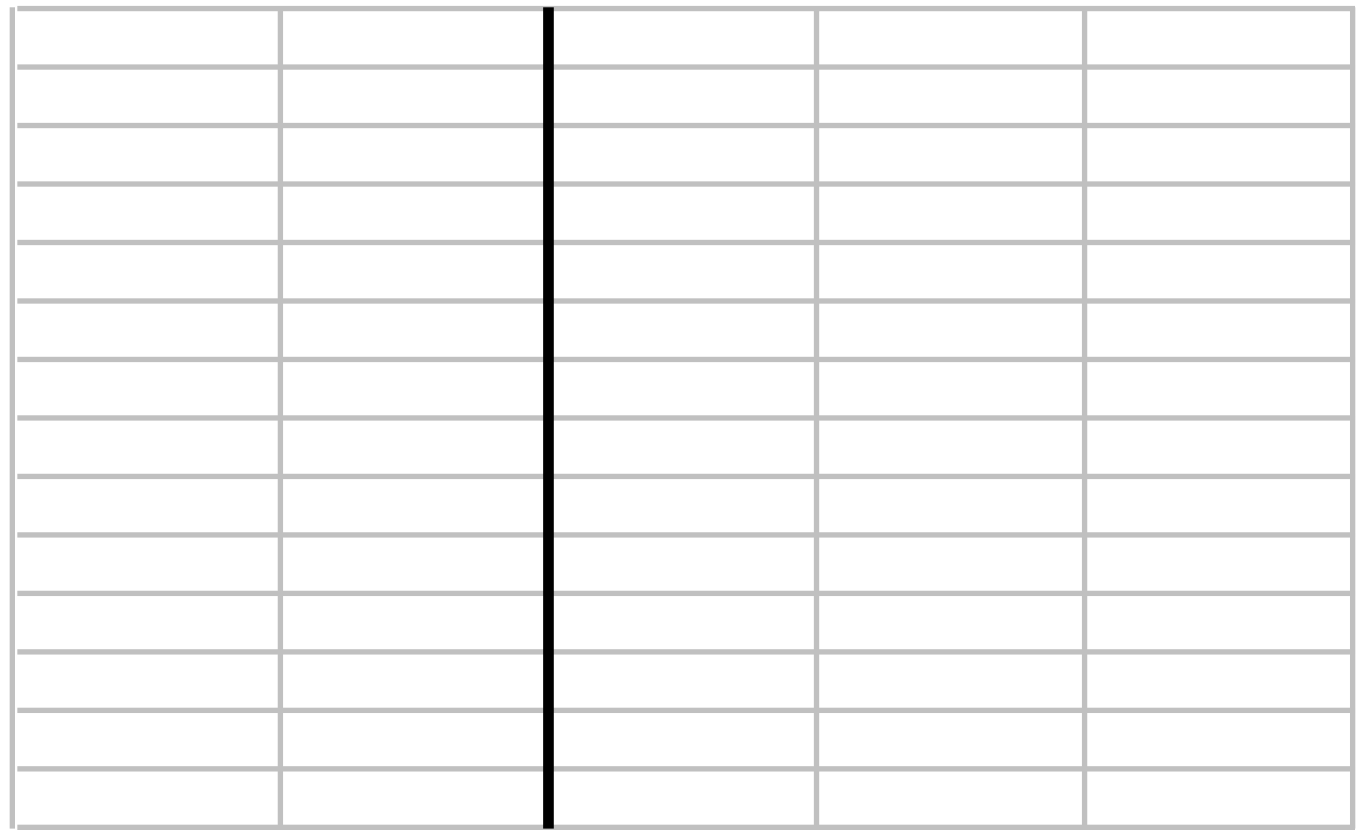
$$A_{1D} v_k = g_k$$

L x constant

$$v_k = \begin{bmatrix} \vdots \\ v_{k,j-1} \\ v_{k,j} \\ v_{k,j+1} \\ \vdots \end{bmatrix} \quad g_k = \begin{bmatrix} \vdots \\ f_{k,j-1} + \varepsilon\left(v_{k-1,j-1} + v_{k+1,j-1}\right) \\ f_{k,j} + \varepsilon\left(v_{k-1,j} + v_{k+1,j}\right) \\ f_{k,j+1} + \varepsilon\left(v_{k-1,j+1} + v_{k+1,j+1}\right) \\ \vdots \end{bmatrix}$$

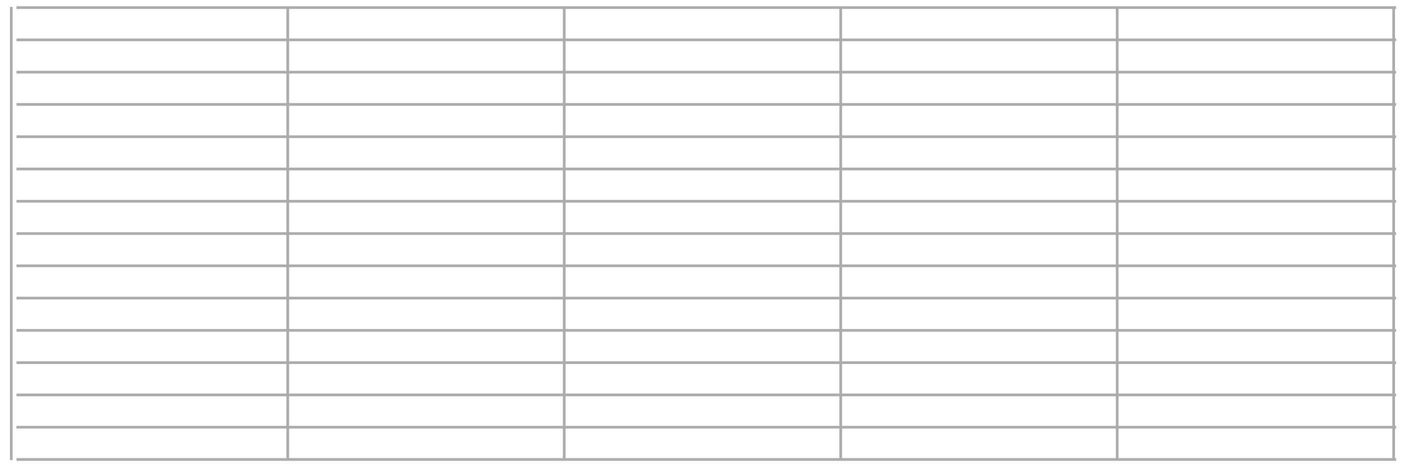
- In 3D, lines become planes…

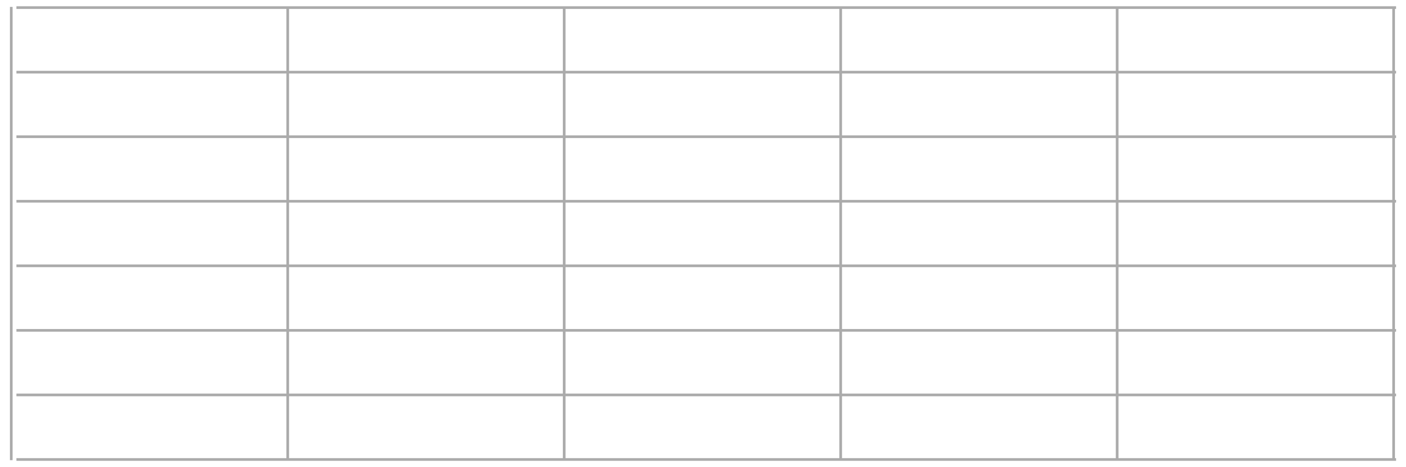




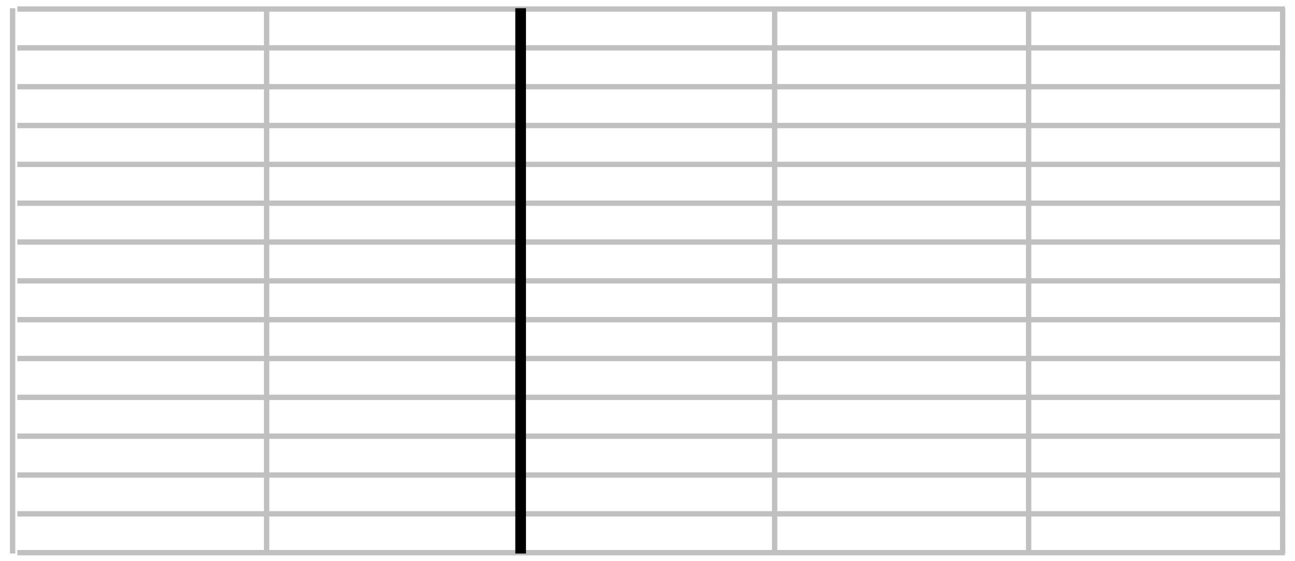
Options for more robust Multigrid

- **Semicoarsening**
  Coarsen in the direction of smoothness





- **Line/plane relaxation**
  Perform relaxation in groups (in a line)



- **Operator based Interpolation** (e.g. BoxMG)

$$Ae = 0$$

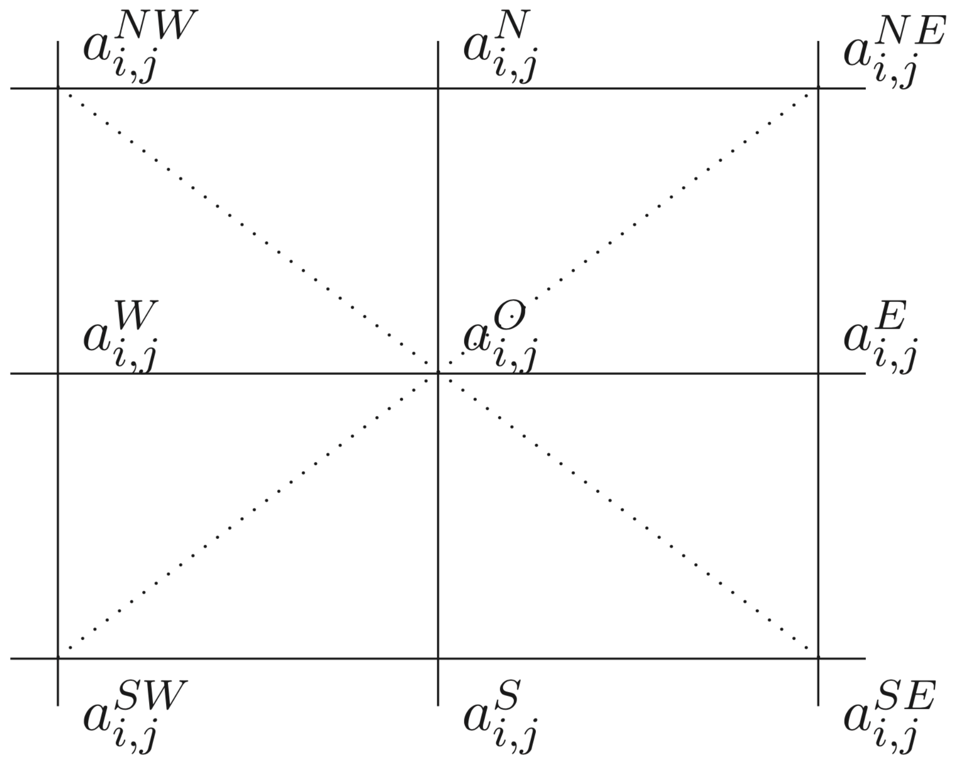
J. E. Dendy, Black box multigrid, J. Comput. Phys., 1982

J. E. Dendy and J. D. Moulton, Black box multigrid with coarsening by a factor of three, J. Numer. Lin. Alg. App., 2010

BoxMG

Math and figures from:
Robust and adaptive multigrid methods: comparing structured and algebraic approaches, MacLachlan, Moulton, Chartier

- Node (i,j) stencil:

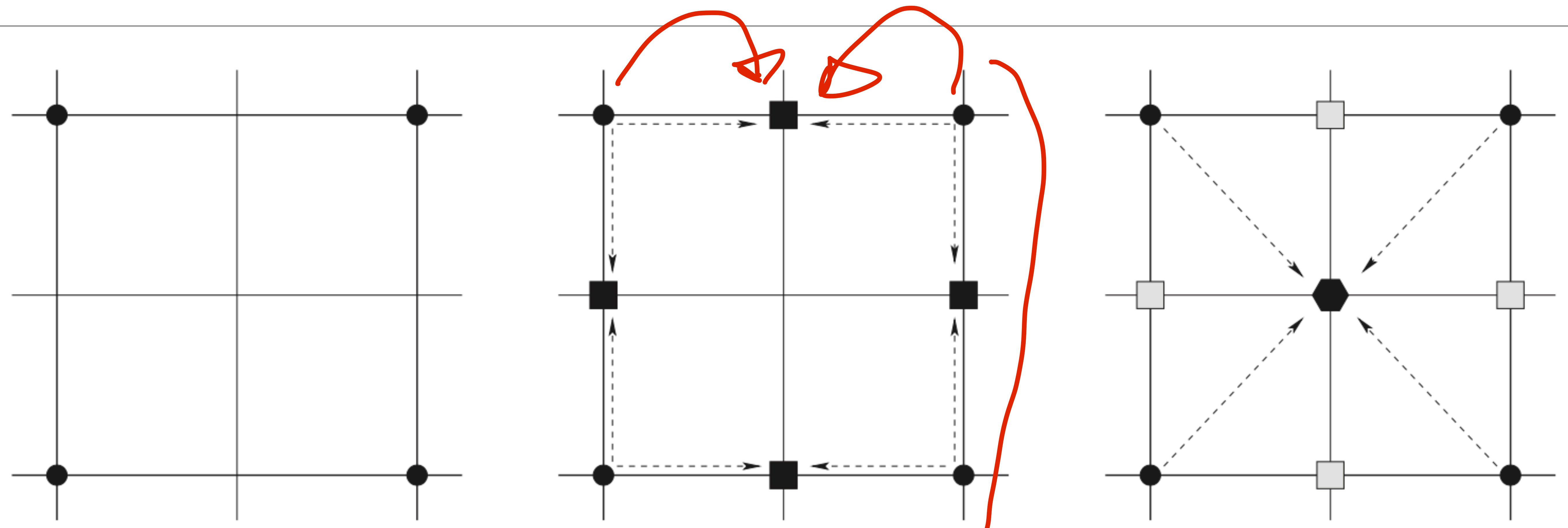


$$Ax = b$$

$$a_{i,j}^{SW} x_{i-1,j-1} + a_{i,j}^{S} x_{i,j-1} + a_{i,j}^{SE} x_{i+1,j-1} + a_{i,j}^{W} x_{i-1,j} + a_{i,j}^{O} x_{i,j} + a_{i,j}^{E} x_{i+1,j} + a_{i,j}^{NW} x_{i-1,j+1} + a_{i,j}^{N} x_{i,j+1} + a_{i,j}^{NE} x_{i+1,j+1} = b_{i,j}$$

BoxMG

Math and figures from:
Robust and adaptive multigrid methods: comparing structured and algebraic approaches, MacLachlan, Moulton, Chartier



1. Inject coarse points (left)
2. Assume the error is constant along x-lines (and y-lines)
3. Infer interpolation from the edges (right)

constant

Example: Assuming Ae=0 and constant y-lines:

$$(a^S + a^O + a^N)e = -(a^{SW} + a^W + a^{NW})e^c_{w}$$
$$- (a^{SE} + a^E + a^{NW})e^c_{E}$$

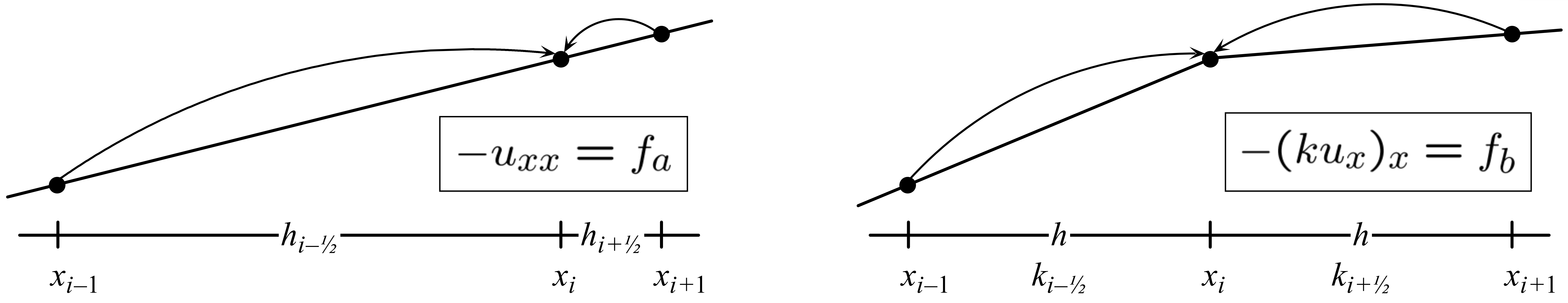
Interpolation based on entries in A

Algebraic Multigrid (AMG) uses matrix coefficients

- Geometric information alone is not sufficient

Linear Interpolation　　Operator-Dependent Interpolation

Slide credit: R. Falgout, 2/1/2016



- AMG does not use geometric information, but captures both linear & operator-dep interpolation

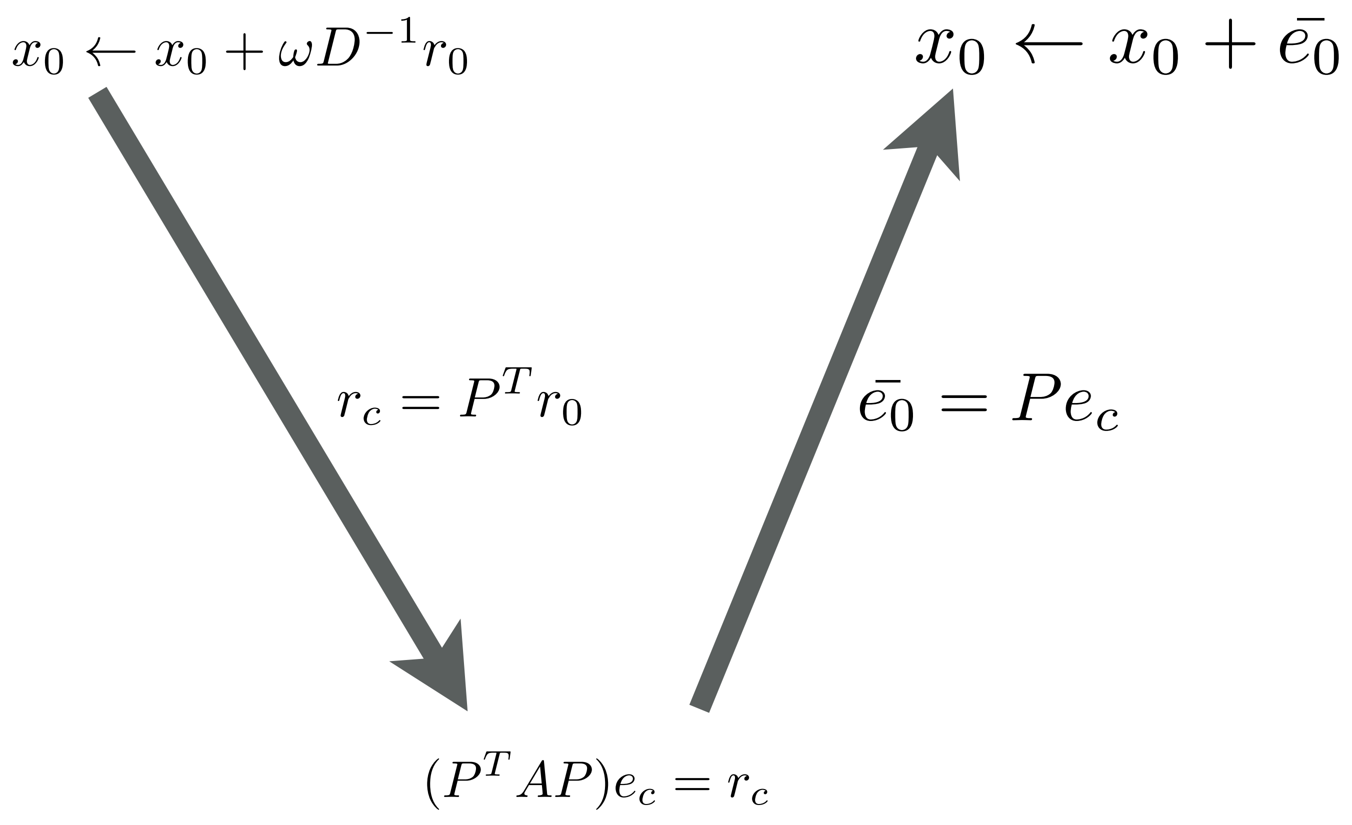
C　F　C

$$(A\mathbf{u})_i = a_{i,i-1}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1} \approx 0$$

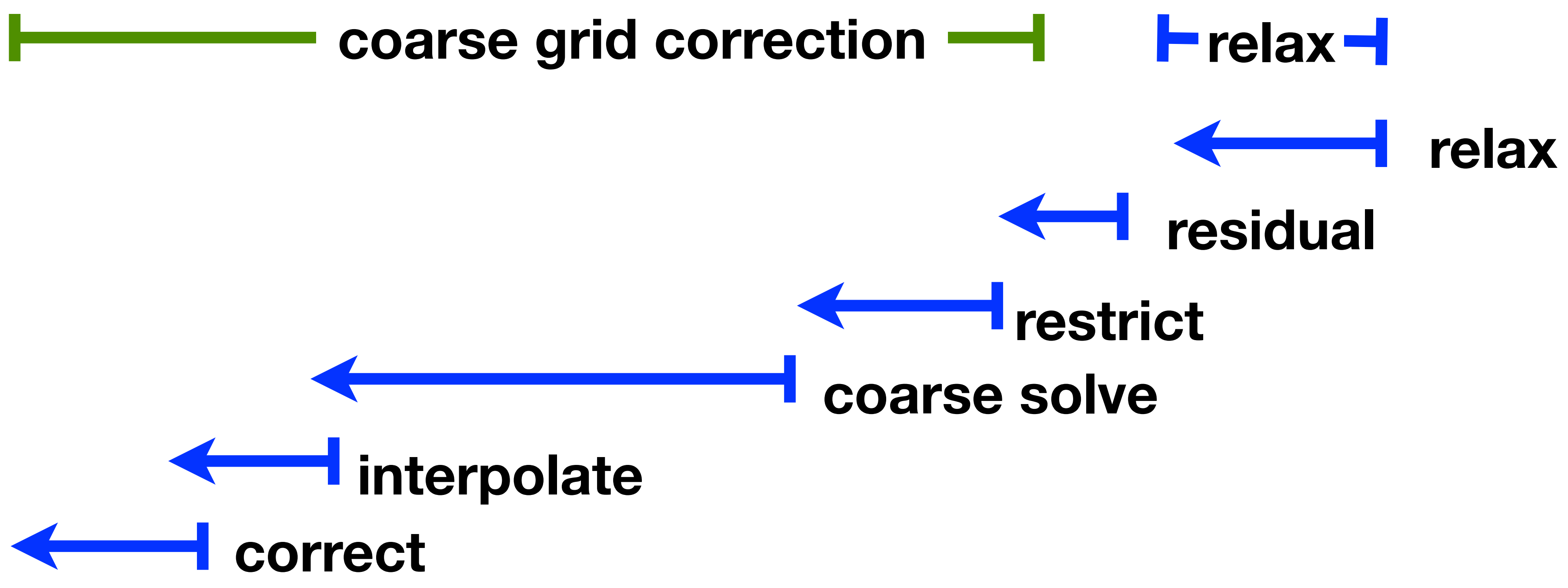
(slow modes are close to constant)

$$u_i = \left(-\frac{a_{i,i-1}}{a_{i,i}}\right) u_{i-1} + \left(-\frac{a_{i,i+1}}{a_{i,i}}\right) u_{i+1}$$

Algebraic Observation



$$e_1 \leftarrow (I - P(P^T A P)^{-1} P^T A) G e_0$$



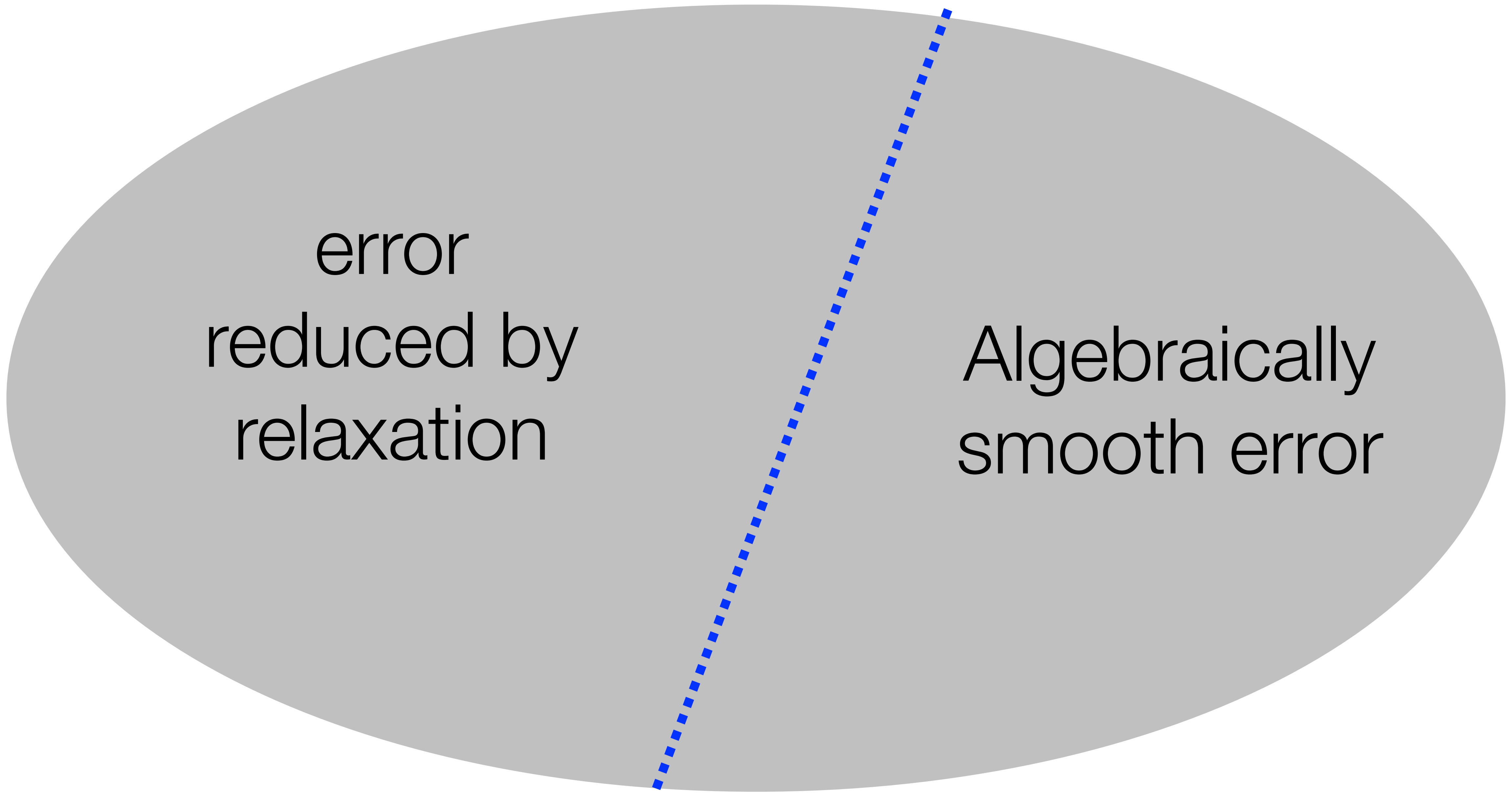
$$Ge_0 \in \mathcal{R}(P) \quad \Rightarrow \quad e_1 = 0$$

interpolation should capture what relaxation misses

determine coarse grid and interpolation based on A

Algebraically Smooth Error

- **"Algebraically smooth" error may not be geometrically smooth**

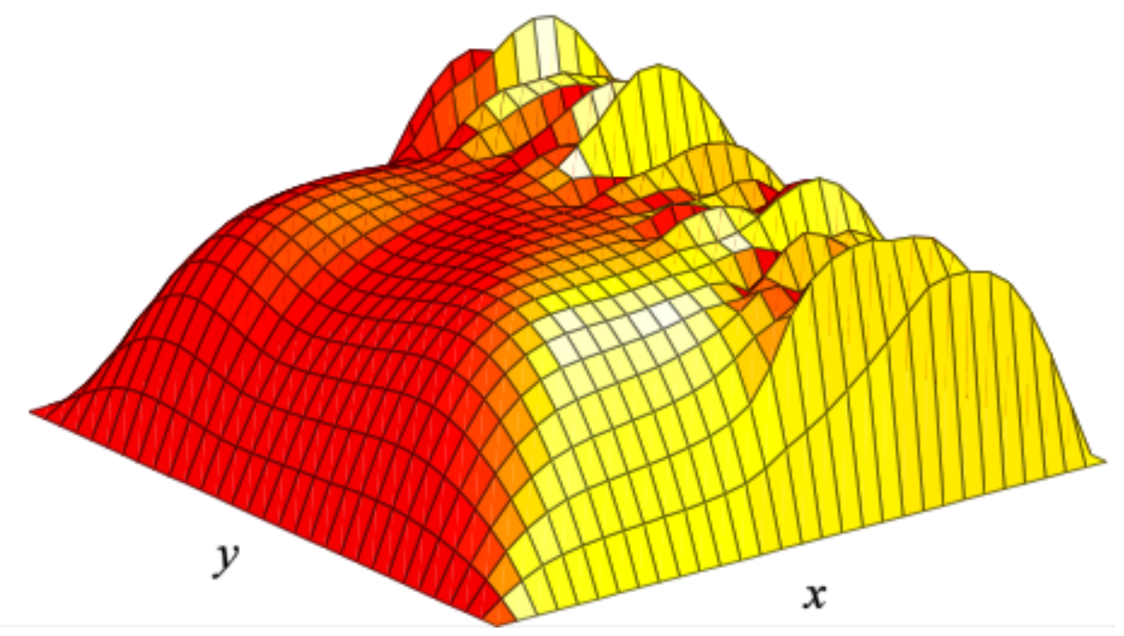


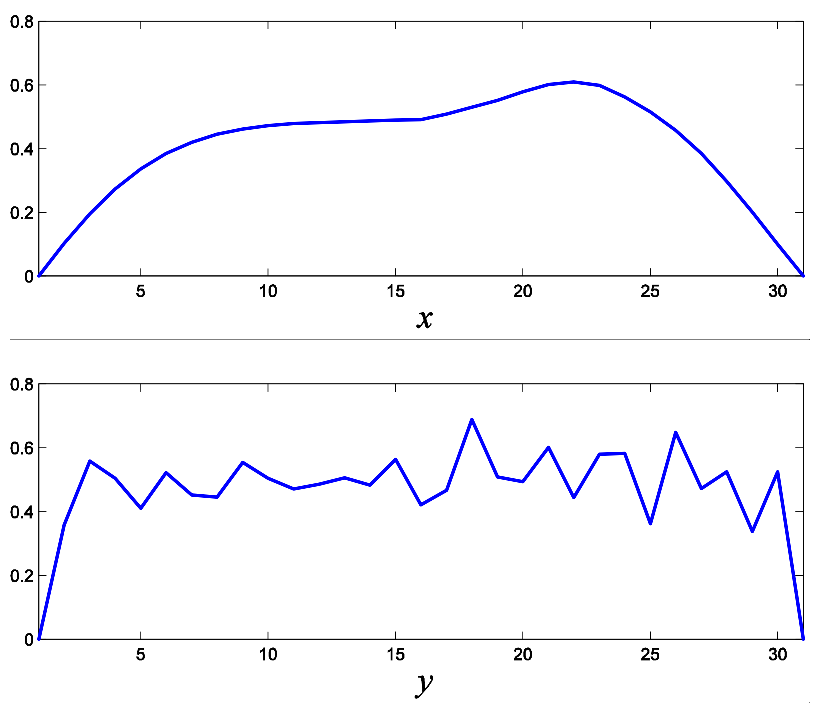
Error left by relaxation can be geometrically oscillatory

- 7 GS sweeps on

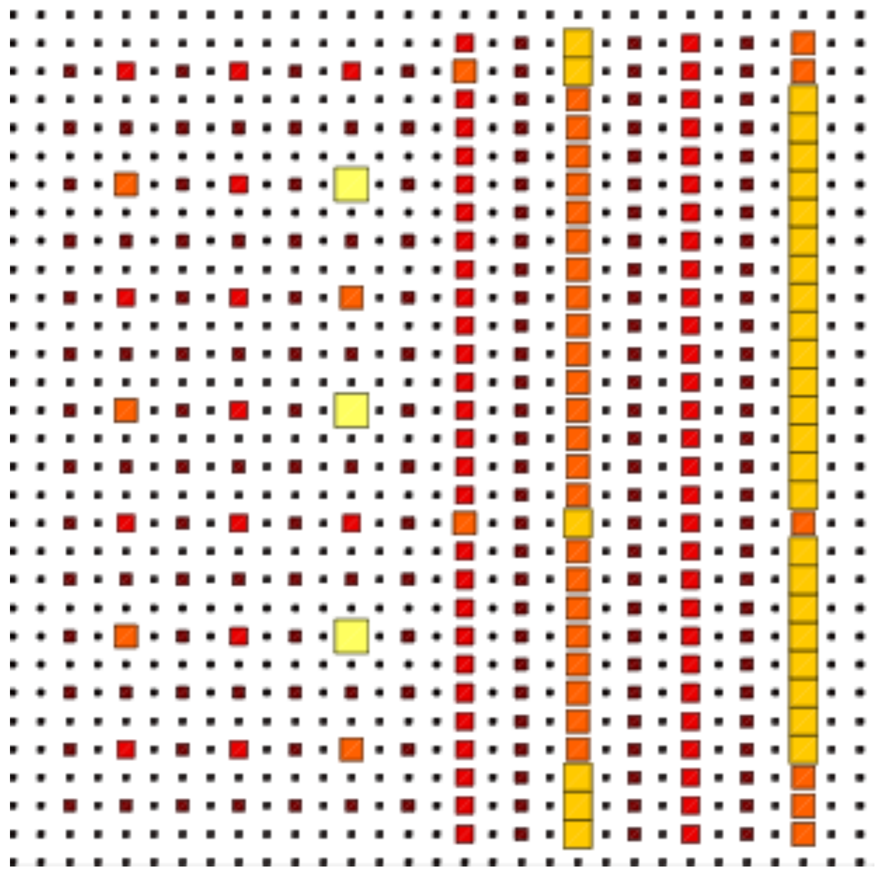
$$-au_{xx} - bu_{yy} = f$$

$a = b$	$a \gg b$





- Caution: this example
  - targets geometric smoothness
  - uses pointwise smoothers



don't coarsen in y, because b is very small, error is oscillatory in y

Slide credit: R. Falgout, 2/1/2016

AMG coarsens grids in the direction of geometric smoothness

"coarsen in the direction of strong connections"

Main idea: Algebraically smooth error

- Take a relaxation scheme such as w-Jacobi

$$e \leftarrow (I - M^{-1}A)e$$

- If relaxation stagnates, then the remaining error exhibits poor convergence, so

error component with small eigenvalues of A converge slowly

- Formally (characterized by small eigenvalues)

$$(I - M^{-1}A)e \approx e \Rightarrow M^{-1}Ae \approx 0 \Rightarrow r \approx 0$$

(small residual)

$$\langle Ae, e \rangle << 1$$

(small eigenvalues of A)

(if $\|e\| = 1$)

Main idea: Algebraically smooth error

- We then have

$$
\begin{aligned}
\langle Ae, e\rangle &= \sum_i e_i(A_{ii}e_i + \sum_{j\neq i} A_{ij}e_j) \\
&= \sum_i e_i \left(\sum_{j\neq i} -A_{ij}(e_i - e_j) \right) \\
&= \sum_{i<j} -A_{ij}\cdot e_i \cdot (e_i - e_j) + \sum_{i>j} -A_{ij}\cdot e_i \cdot (e_i - e_j) \\
&= \sum_{i<j} -A_{ij}\cdot e_i \cdot (e_i - e_j) - \sum_{i<j} -A_{ij}\cdot e_j \cdot (e_i - e_j) \\
&= \sum_{i<j} -A_{ij}\cdot (e_i - e_j)^2 \ll 1
\end{aligned}
$$

assume zero row sum $A_{ii} = -\sum_{i\neq j} A_{ij}$

swap i, j

A_{ji}

- Ok, so smooth error varies **slowly** in the direction of large matrix coefficients

when A_{ij} is large,
$e_i \simeq e_j$ after relaxation

Briggs, William L. and Henson, Van Emden and McCormick, Steve F., A Multigrid Tutorial (2Nd Ed., 20000

Take a look: `12-AMG-coarse-mesh.ipynb`

Main idea: Algebraically smooth error

- We have assumed **geometric** smoothness to show

$$\mathbf{e}^T A \mathbf{e} = \sum_{i<j} (-a_{ij})(e_i - e_j)^2 \ll 1$$

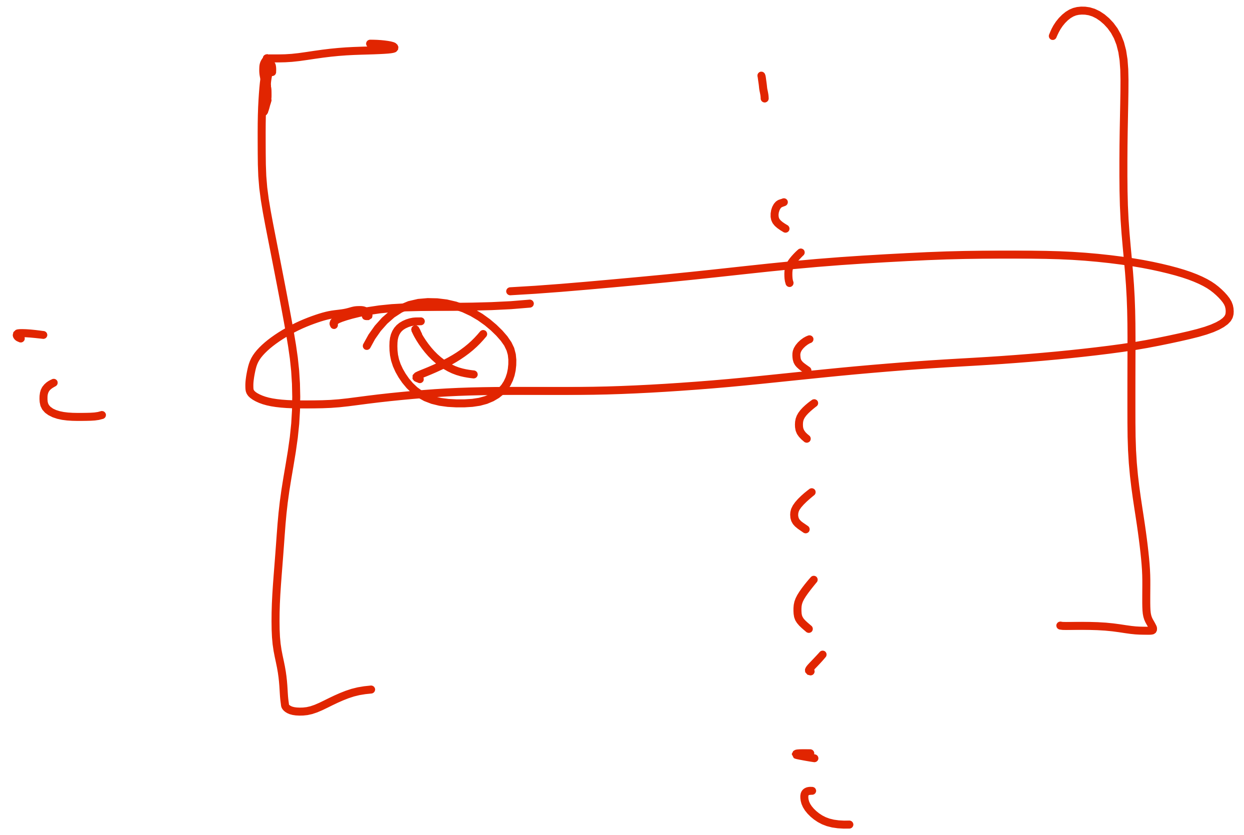
assume $a_{ij} < 0$ if $i \neq j$

- **CF AMG**: Smooth error varies slowly in the direction of "large" matrix coefficients
- **Strength of connection**: Given a threshold $0 < \theta \leq 1$, we say that variable u_i strongly depends on variable u_j if

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}$$

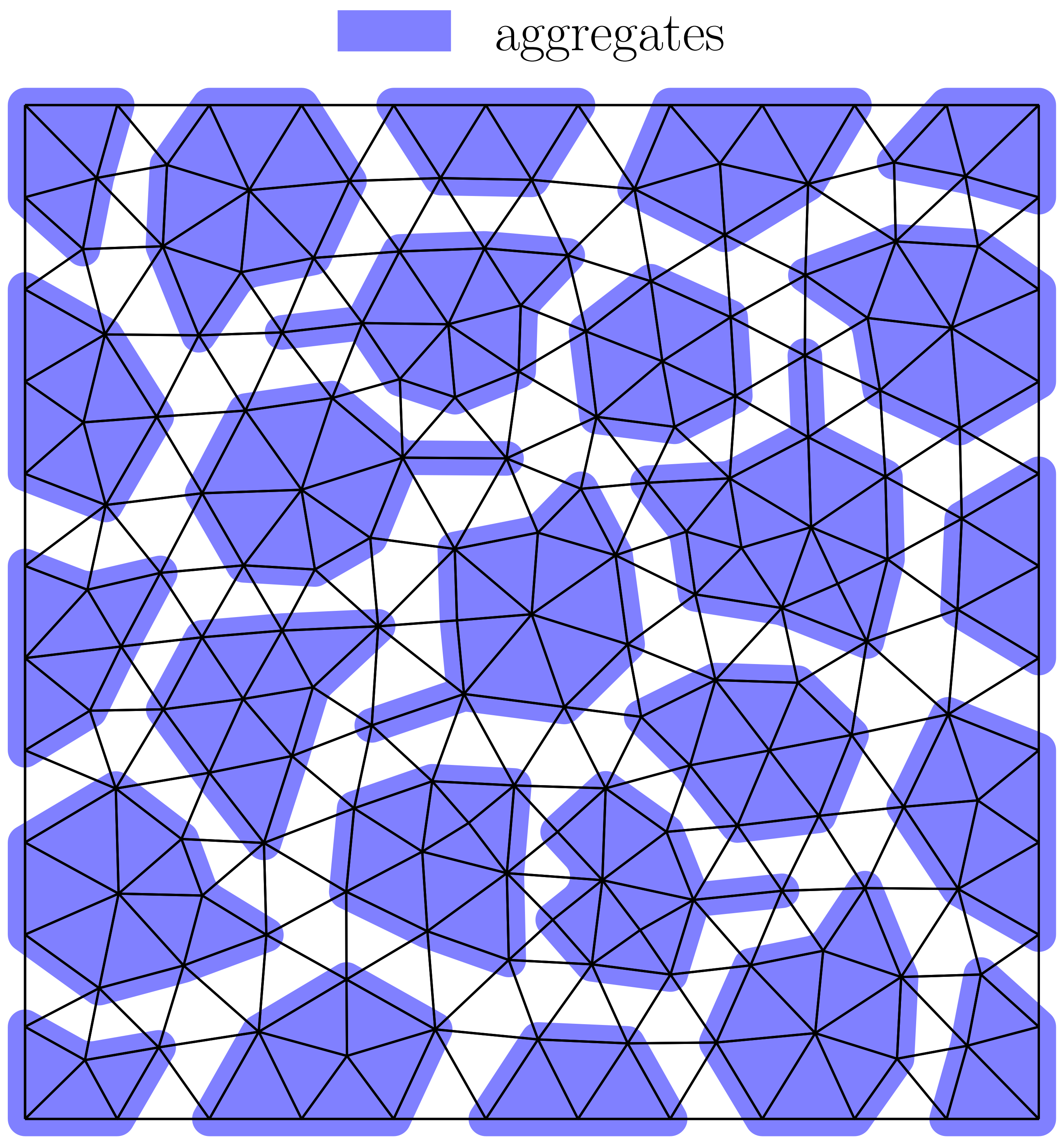
$<$ in row i

- Often positive off-diagonals are treated as weak
- This definition of strength of connection is not symmetric

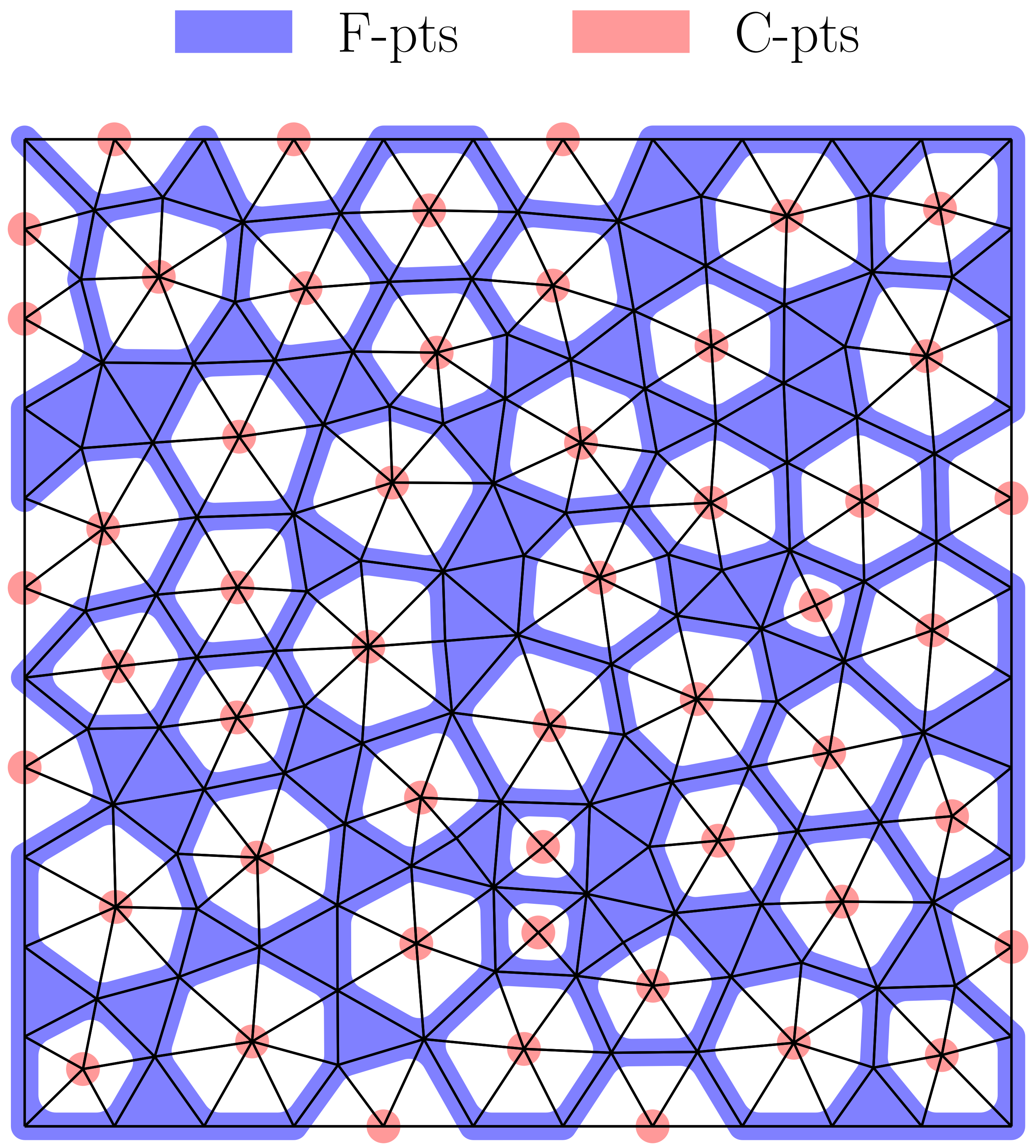


Slide credit: R. Falgout, 2/1/2016

Two (general) forms of AMG



- Smoothed Aggregation AMG (SA-AMG)
- Interpolation constructed from candidate vectors
- Clear approach to *optimize* interpolation



- Coarse-Fine AMG (CF-AMG) or Ruge-Stüben
- Coarse grid points are a subset of the fine grid points
- Edge-wise construction of interpolation, allowing straightforward control of sparsity
- Incorporating near-nullspace is not straightforward