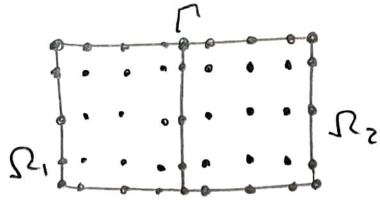
Last time:



$$\nabla^2 u = f \text{ in } \Omega$$
$$u = 0 \text{ on } \partial\Omega$$

① Solve $A_{11} u_1^{source} = f_1$
$A_{22} u_2^{source} = f_2$ ← zero Dirichlet BCs

② Solve $\Sigma u_\Gamma = f_\Gamma - A_{\Gamma 1} u_1^{source} - A_{\Gamma 2} u_2^{source}$

③ Solve $A_{11} u_1^{harmonic} = 0$
$A_{22} u_2^{harmonic} = 0$ ← u_Γ Dirichlet BCs

④ Update: $u_1 = u_1^{source} + u_1^{harmonic}$
$u_2 = u_2^{source} + u_2^{harmonic}$

Σ has fast mat-vec:

$$\Sigma x = (A_{\Gamma\Gamma} - A_{\Gamma 1} A_{11}^{-1} A_{1\Gamma} - A_{\Gamma 2} A_{22}^{-1} A_{2\Gamma})\, x$$
$$= \underbrace{A_{\Gamma\Gamma} x}_{O(n^2)} - \underbrace{D2N_1(x)}_{O(n^2 \log n)} - \underbrace{D2N_2(x)}_{O(n^2 \log n)}$$

$D2N_i(x) =$ • Solve $A_{ii} v_i = 0$ with x Dirichlet BC
• Evaluate normal derivative of v_i on Γ

Need good preconditioner for Σ.

Suppose* we have a fast spectrally accurate subdomain solver $\mathcal{O}(n^2 \log n)$. Then the remaining piece is to solve (2) fast.

Traditional approaches attempt to solve $\Sigma u_\Gamma = z_\Gamma$ using a preconditioned iterative method such as CG, since Σ has a fast matrix-vector multiply:

$$\Sigma x = (A_{\Gamma\Gamma} - \underbrace{A_{\Gamma 1} A_{11}^{-1} A_{1\Gamma}}_{D2N_1} - \underbrace{A_{\Gamma 2} A_{22}^{-1} A_{2\Gamma}}_{D2N_2})\, x$$

$$= A_{\Gamma\Gamma} x - D2N_1(x) - D2N_2(x)$$

where $D2N_i(x) =$
- Solve $A_{ii} v_i = 0$ with x Dirichlet BC
- Evaluate normal derivative of v_i on Γ!

$D2N_i(x)$ takes $\mathcal{O}(n^2 \log n)$ & $A_{\Gamma\Gamma} x$ takes $\mathcal{O}(n^2)$.

We would need a good preconditioner for Σ to get optimal complexity. (This is what is done currently: solve for u_Γ exactly, then u_1 & u_2.)

Instead: Design a preconditioner $A^\dagger \approx A^{-1}$ for the global problem based on approximate subdomain solves $A_{ii}^\dagger \approx A_{ii}^{-1}$ and approximate interface solve $\Sigma^\dagger \approx \Sigma^{-1}$.

("DD-PCG")

For K elements, $\tilde{u} = A^{\dagger} f$ is

for $i = 1, \ldots, K$
- Solve $\tilde{u}_i^{\text{source}} = A_{ii}^{\dagger} f_i$ with zero BCs
- Compute normal derivative $s_i = -A_{\Gamma i} \tilde{u}_i^{\text{source}}$

end
- Compute $z_\Gamma = f_\Gamma + s_1 + \cdots + s_K$
- Solve $\tilde{u}_\Gamma = \Sigma^{\dagger} z_\Gamma$

for $i = 1, \ldots, K$
- Solve $\tilde{u}_i^{\text{harmonic}} = A_{ii}^{\dagger} 0$ with $\tilde{u}_\Gamma$ BCs
- Compute $\tilde{u}_i = \tilde{u}_i^{\text{source}} + \tilde{u}_i^{\text{harmonic}}$

end

It remains to specify $A_{ii}^{\dagger}$ and $\Sigma^{\dagger}$.

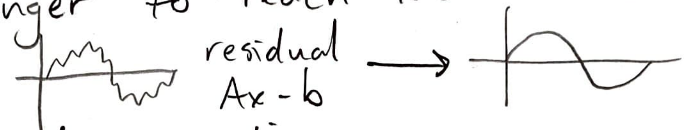
$A_{ii}^{\dagger}$ is "easy" = "approximate fast Poisson solve"

If we assume discretization is SPD (such as Legendre-Galerkin), can use <u>multigrid</u> method.

<u>A multigrid solver for the interface</u>

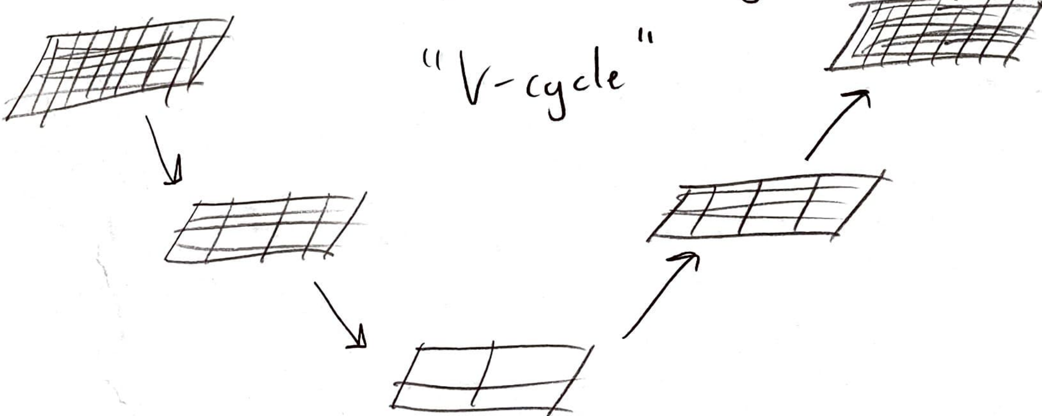
Multigrid methods rely on two ingredients: <u>smoothing</u> & <u>coarse-grid correction</u>.

A smoother (e.g. Jacobi, Gauss-Seidel) is an iterative scheme that <u>locally</u> satisfies a linear system. Thus, it kills <u>high-freq.</u> error quickly, but takes longer to reach the true solution.



"What local change to x will kill high-freq error in residual?"

Coarse-grid correction notes that if we run the smoother on a <u>coarser</u> "grid", then the <u>local</u> action of the smoother now has a more <u>global</u> effect, and the <u>low frequency</u> modes of error that were a problem on the fine grid are "higher" frequency here.



Jacobi: $A = D + L + U$, $Ax = b$ "Matrix splitting"

Iterate $x^{(i+1)} = D^{-1}(b - (L+U)x^{(i)})$

G-S: $A = D + L + U$, $Ax = b$

Iterate $x^{(i+1)} = (D+L)^{-1}(b - Ux^{(i)})$

To use either of these on Σ, need to have formed Σ explicitly, which is not optimal complexity due to A_{ii}^{-1}!

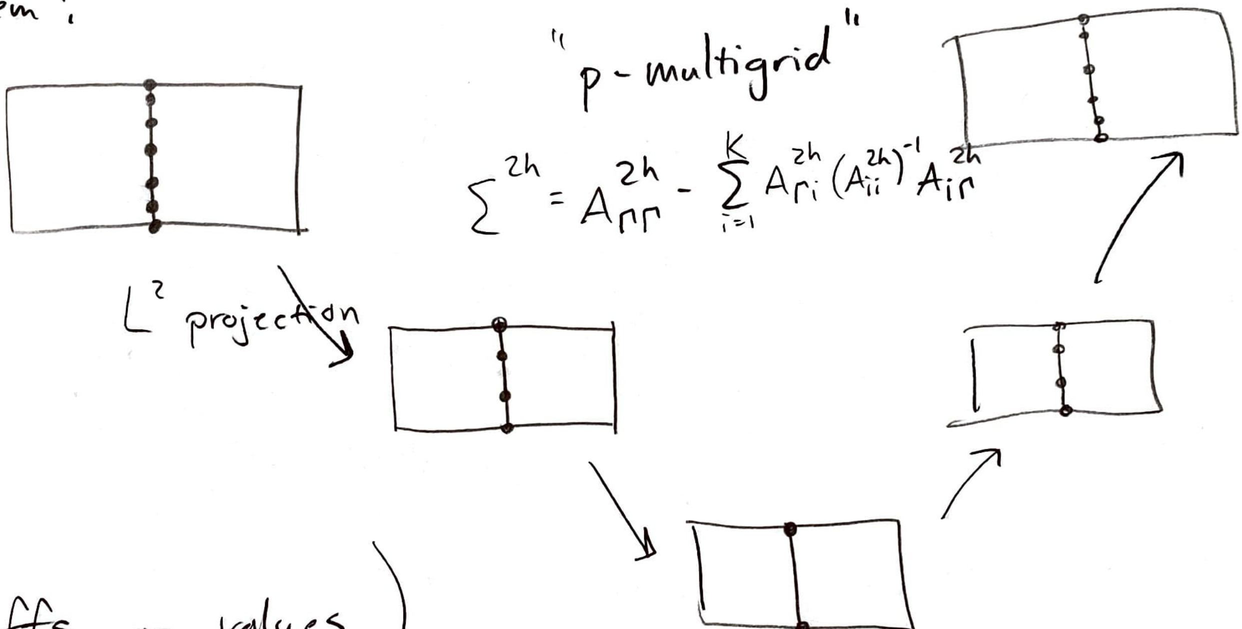
<u>Trick</u>: If Σ is SPD, can prove the following (fixed pt) iteration will always converge:

$$\left(\Sigma = A_{\Gamma\Gamma} - \sum_{i=1}^{K} A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma} \right)$$

$$A_{\Gamma\Gamma} u_\Gamma^{(j+1)} = f_\Gamma + \sum_{i=1}^{K} A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma} u_\Gamma^{(j)} \quad \text{(Assembly-free!)}$$

Need to invert $A_{\Gamma\Gamma}$. (Fast?) $\left(\begin{array}{l} \text{c.f. } \Sigma = D - L - U \text{ Jacobi} \\ D u_\Gamma^{(j+1)} = f_\Gamma + (L+U) u_\Gamma^{(j)} \end{array} \right)$

Then, what are the coarse grids for the interface problem?



(coeffs or values)

So, Σ^{+} = "approximate interface solve"
= run a few V-cycles of multigrid.

Proof: $\underbrace{A_{\Gamma\Gamma}}_{S_1} u_\Gamma^{(j+1)} = f_\Gamma + \underbrace{\sum_{i=1}^{K} A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}}_{S_2} u_\Gamma^{(j)}$

$$\Rightarrow S_1 u_\Gamma^{(j+1)} = f_\Gamma + S_2 u_\Gamma^{(j)} \quad (1)$$

Let u_Γ^* be the fixed point, so

$$S_1 u_\Gamma^* = f_\Gamma + S_2 u_\Gamma^* \quad (2)$$

Then the error $e^{(j)} = u_\Gamma^{(j)} - u_\Gamma^*$ is given by $(1) - (2)$:

$$S_1 e^{(j+1)} = S_2 e^{(j)}$$

$$\Rightarrow e^{(j+1)} = S_1^{-1} S_2 e^{(j)}$$

This will converge iff $\rho(S_1^{-1}S_2) < 1$. ($\rho \nearrow \lambda_{max}$)

Note
- $\Sigma = S_1 - S_2 \Rightarrow S_1 = \Sigma + S_2$
- $S_1^{-1}S_2 \sim S_2^{1/2}(S_1^{-1}S_2)S_2^{-1/2} = S_2^{1/2} S_1^{-1} S_2^{1/2}$

  ($S_2^{1/2}$ exists since S_2 is real symmetric)

So $\rho(S_1^{-1}S_2) = \rho(S_2^{1/2}S_1^{-1}S_2^{1/2}) = \rho((S_2^{-1/2}S_1S_2^{-1/2})^{-1})$

$$= \rho((S_2^{-1/2}(\Sigma + S_2)S_2^{-1/2})^{-1})$$

$$= \rho((S_2^{-1/2}\Sigma S_2^{-1/2} + I)^{-1})$$

$$= \frac{1}{\lambda_{min}(S_2^{-1/2}\Sigma S_2^{-1/2} + I)}$$

$$= \frac{1}{\lambda_{min}(S_2^{-1/2}\Sigma S_2^{-1/2}) + 1}$$

Since Σ is SPD and $S_2^{-1/2}$ is symmetric, $S_2^{-1/2}\Sigma S_2^{-1/2}$ is SPD, so: $\lambda_{min}(S_2^{-1/2}\Sigma S_2^{-1/2}) > 0$.

Thus, $\rho(S_1^{-1}S_2) < 1$.