Multigrid as a useful solver/preconditioner for blocks

Iterative Methods for Large-Scale Saddle-Point Problems

Fabio Durastante

Università di Pisa

fabio.durastante@unipi.it

May, 2022

Overview

The Multigrid Idea

☠ A word of caution

These methods **start from a simple idea**, but the confusion of their explanation grows exponentially with the degree of generality and abstraction that one wants to impose.

The Multigrid Idea

☠️ A word of caution

These methods **start from a simple idea**, but the confusion of their explanation grows exponentially with the degree of generality and abstraction that one wants to impose.

Our plan

Therefore we will start explaining them in a completely abstract framework.

The Multigrid Idea

☠ A word of caution

These methods **start from a simple idea**, but the confusion of their explanation grows exponentially with the degree of generality and abstraction that one wants to impose.

Our plan

~~Therefore we will start explaining them in a completely abstract framework.~~ Therefore we will start from the simplest 1D example, and work on that.

The Multigrid Idea

☠ A word of caution

These methods **start from a simple idea**, but the confusion of their explanation grows exponentially with the degree of generality and abstraction that one wants to impose.

Our plan

~~Therefore we will start explaining them in a completely abstract framework.~~ Therefore we will start from the simplest 1D example, and work on that.

Good introductions are contained in the books (Briggs, Henson, and McCormick 2000; Trottenberg, Oosterlee, and Schüller 2001; Vassilevski 2008), for the more theoretically inclined the best high-level presentation is in (Xu and Zikatanov 2017).

Multigrid based on geometry: Poisson

Let us consider the following boundary value problem

$$\begin{cases} -u_{xx}(x) = f(x), & x \in (0,1), \\ u(0) = u(1) = 0. \end{cases}, \; f \in \mathcal{C}^1([0,1]).$$

If we apply **standard centered finite difference** discretization on the grid $\Omega_h = \{x_k\}_{k=0}^{n+2} = \{kh\}$ and $h = 1/(n+2)$, that gives rise to the linear system

$$\frac{K}{h^2} A_n \mathbf{u} = \mathbf{f}, \qquad A_n = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

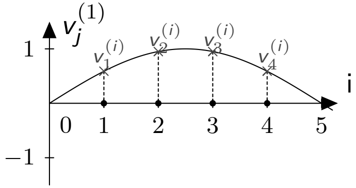
Multigrid based on geometry: Poisson

The matrix A_n is a **very peculiar type of matrix** for which we now everything, specifically:

$$A_n \mathbf{v}^{(i)} = \lambda_i \mathbf{v}^{(i)},$$

with eigenvalues and eigenvectors

$$\lambda_i = 2 - 2\cos\left(\frac{i\pi}{n+1}\right), \quad v_j^{(i)} = \sin\left(\frac{ij\pi}{n+1}\right), \; i,j = 1,\dots,n.$$



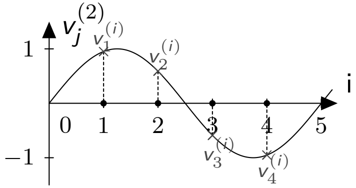
Multigrid based on geometry: Poisson

The matrix A_n is a **very peculiar type of matrix** for which we now everything, specifically:

$$A_n \mathbf{v}^{(i)} = \lambda_i \mathbf{v}^{(i)},$$

with eigenvalues and eigenvectors

$$\lambda_i = 2 - 2\cos\left(\frac{i\pi}{n+1}\right), \quad v_j^{(i)} = \sin\left(\frac{ij\pi}{n+1}\right),\ i,j = 1,\dots,n.$$



- If we **sample the eigenvectors** we get oscillating functions,

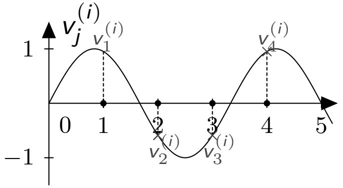
Multigrid based on geometry: Poisson

The matrix A_n is a **very peculiar type of matrix** for which we now everything, specifically:

$$A_n \mathbf{v}^{(i)} = \lambda_i \mathbf{v}^{(i)},$$

with eigenvalues and eigenvectors

$$\lambda_i = 2 - 2\cos\left(\frac{i\pi}{n+1}\right), \quad v_j^{(i)} = \sin\left(\frac{ij\pi}{n+1}\right), \; i,j = 1,\ldots,n.$$



- If we **sample the eigenvectors** we get oscillating functions,
- They represent exactly the *frequency* in the span of the grid function Fourier series.

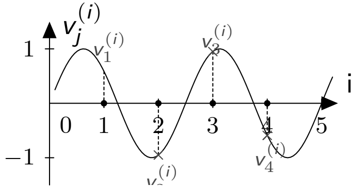
Multigrid based on geometry: Poisson

The matrix A_n is a **very peculiar type of matrix** for which we now everything, specifically:

$$A_n \mathbf{v}^{(i)} = \lambda_i \mathbf{v}^{(i)},$$

with eigenvalues and eigenvectors

$$\lambda_i = 2 - 2\cos\left(\frac{i\pi}{n+1}\right), \quad v_j^{(i)} = \sin\left(\frac{ij\pi}{n+1}\right), \; i,j = 1,\dots,n.$$



- If we **sample the eigenvectors** we get oscillating functions,
- They represent exactly the *frequency* in the span of the grid function Fourier series.

Multigrid based on geometry: Poisson

The matrix A_n is a **very peculiar type of matrix** for which we now everything, specifically:

$$A_n \mathbf{v}^{(i)} = \lambda_i \mathbf{v}^{(i)},$$

with eigenvalues and eigenvectors

$$\lambda_i = 2 - 2\cos\left(\frac{i\pi}{n+1}\right), \quad v_j^{(i)} = \sin\left(\frac{ij\pi}{n+1}\right),\ i,j = 1,\ldots,n.$$

We divide **arbitrarily** this set of frequencies in two subsets:

$$\begin{array}{ll} \text{Low frequencies} & \left\{\mathbf{v}^{(i)} = \sin(i\mathbf{y}) \,:\, \mathbf{y} = \frac{j\pi}{n+1}, j = 1,\ldots,n\ i = 1,\ldots,n/2-1\right\}, \\ \text{High frequencies} & \left\{\mathbf{v}^{(i)} = \sin(i\mathbf{y}) \,:\, \mathbf{y} = \frac{j\pi}{n+1}, j = 1,\ldots,n\ i = n/2,\ldots,1\right\}. \end{array}$$

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\begin{aligned}\mathbf{u}^{(k+1)} &= \mathbf{u}^{(k)} + D_n^{-1}(\mathbf{f} - A_n\mathbf{u}^{(k)}) = \left(I_n - \frac{1}{2}A_n\right)\mathbf{u}^{(k)} + \frac{1}{2}\mathbf{f},\\ D_n &= \operatorname{diag}(A_n) = 2I_n\end{aligned}$$

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\begin{aligned}\mathbf{u}^{(k+1)} =& \mathbf{u}^{(k)} + D_n^{-1}(\mathbf{f} - A_n\mathbf{u}^{(k)}) = \left(I_n - \frac{1}{2}A_n\right)\mathbf{u}^{(k)} + \frac{1}{2}\mathbf{f},\\ D_n =& \operatorname{diag}(A_n) = 2I_n\end{aligned}$$

- The iteration matrix is then $J_n = I_n - A_n/2$, with the information we have on the spectrum, we observe that: $\rho(J_n) \to 1$ for $n \to +\infty$ so slow convergence! A sorry state of affairs.

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\mathbf{u}^{(k+1)} = \left[(1-\omega)I_n + \omega\left(I_n - \frac{1}{2}A_n\right)\right]\mathbf{u}^{(k)} + \frac{1}{2}\omega\mathbf{f},$$

- The iteration matrix is then $J_n = I_n - A_n/2$, with the information we have on the spectrum, we observe that: $\rho(J_n) \to 1$ for $n \to +\infty$ so slow convergence! A sorry state of affairs.
- Maybe we can **weight the iteration** to make things better

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\mathbf{u}^{(k+1)} = \left[(1-\omega)I_n + \omega\left(I_n - \frac{1}{2}A_n\right)\right]\mathbf{u}^{(k)} + \frac{1}{2}\omega\mathbf{f},$$

- The iteration matrix is then $J_n = I_n - A_n/2$, with the information we have on the spectrum, we observe that: $\rho(J_n) \to 1$ for $n \to +\infty$ so slow convergence! A sorry state of affairs.
- Maybe we can **weight the iteration** to make things better. The iteration matrix is now $J_n^\omega = I_n - \frac{\omega}{2}A_n$. The best spectral conditioning for $\mu_1^\omega = \lambda_{\max}(J_n^\omega)$ is obtained for $\mu_1^1 < \mu_1^\omega \ \forall \omega \in (0,1)$. Therefore, we have only made the convergence worse...

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\mathbf{u}^{(k+1)} = \left[(1-\omega)I_n + \omega\left(I_n - \frac{1}{2}A_n\right)\right]\mathbf{u}^{(k)} + \frac{1}{2}\omega\mathbf{f},$$

We **hold the line**, let us write everything in the **eigenvector basis**:

$$\mathbf{e}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{v}^{(i)}, \quad J_n^\omega = V\Lambda_n^\omega V^T \text{ where } \Lambda_n^\omega = \mathrm{diag}(\mu_i^\omega),$$

at the kth step (for $\bar{\mathbf{u}} = A_n^{-1}\mathbf{f}$ the true solution) we find

$$\begin{aligned}\bar{\mathbf{u}} - \mathbf{u}^{(k)} = \mathbf{e}^{(k)} &= (J_n^\omega)^k \mathbf{e}^{(0)} = V(\Lambda_n^\omega)^k V^T \mathbf{e}^{(0)} = V(\Lambda_n^\omega)^k \alpha \\ &= \sum_{i=1}^{n} \left(1 - 2\omega\sin^2\left(\frac{i\pi}{2(n+1)}\right)\right)^k \alpha_i v^{(i)}.\end{aligned}$$

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\mathbf{u}^{(k+1)} = \left[(1-\omega)I_n + \omega\left(I_n - \frac{1}{2}A_n\right)\right]\mathbf{u}^{(k)} + \frac{1}{2}\omega\mathbf{f},$$

We **hold the line**, let us write everything in the **eigenvector basis**:

$$\bar{\mathbf{u}} - \mathbf{u}^{(k)} = \sum_{i=1}^{n}\left(1 - 2\omega\sin^2\left(\frac{i\pi}{2(n+1)}\right)\right)^k \alpha_i v^{(i)}.$$

Idea!

The ith entry of $\mathbf{e}^{(k)}$ is defined in terms of the ith eigenvalues of J_k^ω:

$$\beta_i = \left(1 - 2\omega\sin^2\left(\frac{i\pi}{2(n+1)}\right)\right)^k \alpha_i \approx \left(1 - \omega\frac{\pi^2}{2}h^2\right)^k \alpha_i.$$

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\mathbf{u}^{(k+1)} = \left[(1-\omega)I_n + \omega\left(I_n - \frac{1}{2}A_n\right)\right]\mathbf{u}^{(k)} + \frac{1}{2}\omega\mathbf{f},$$

We **hold the line**, let us write everything in the **eigenvector basis**:

$$\overline{\mathbf{u}} - \mathbf{u}^{(k)} = \sum_{i=1}^{n}\left(1 - 2\omega\sin^2\left(\frac{i\pi}{2(n+1)}\right)\right)^k \alpha_i v^{(i)}.$$

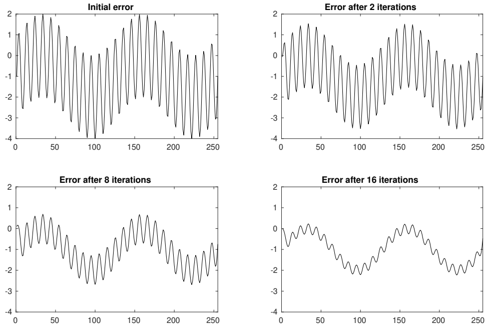
Working only in the high-frequency

We choose an *optimal* ω that minimizes the absolute values of the β_i in the high frequencies, i.e., $\omega_{\text{opt}} = 2/3$.

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\mathbf{u}^{(k+1)} = \left[(1-\omega)I_n + \omega\left(I_n - \frac{1}{2}A_n\right)\right]\mathbf{u}^{(k)} + \frac{1}{2}\omega\mathbf{f},$$



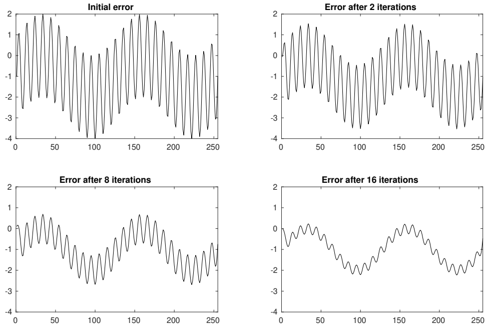
Solver as smoother

In the domain of the high frequencies, for whatever value of k, we find that the error has become a *smooth function*.

Multigrid based on geometry: Jacobi

Now we have put some notation in place, but let us try solving our system with the simplest method we know: Jacobi!

$$\mathbf{u}^{(k+1)} = \left[(1-\omega)I_n + \omega\left(I_n - \frac{1}{2}A_n\right)\right]\mathbf{u}^{(k)} + \frac{1}{2}\omega\mathbf{f},$$



Solver as smoother

In the domain of the high frequencies, for whatever value of k, we find that the error has become a *smooth function*.

But the convergence is always bad...

Multigrid based on geometry: Error equation

- Let us suppose that we have computed an approximation $\widetilde{\mathbf{u}}$ of the solution $\overline{\mathbf{u}}$ through some iterations of the optimally weighted Jacobi.

Multigrid based on geometry: Error equation

- Let us suppose that we have computed an approximation $\widetilde{\mathbf{u}}$ of the solution $\overline{\mathbf{u}}$ through some iterations of the optimally weighted Jacobi.

If only we knew the error...

If we could compute in some way the error $\mathbf{e}$ then the solution of the linear system could be obtained as $\overline{\mathbf{u}} = \widetilde{\mathbf{u}} + \mathbf{e}$

- If we wanted to compute it, we could solve the linear system

$$A_n\mathbf{e} = A_n\overline{\mathbf{u}} - A_n\widetilde{\mathbf{u}} = \mathbf{f} - A_n\widetilde{\mathbf{u}} = \mathbf{r}.$$

Multigrid based on geometry: Error equation

- Let us suppose that we have computed an approximation $\widetilde{\mathbf{u}}$ of the solution $\overline{\mathbf{u}}$ through some iterations of the optimally weighted Jacobi.

If only we knew the error...

If we could compute in some way the error $\mathbf{e}$ then the solution of the linear system could be obtained as $\overline{\mathbf{u}} = \widetilde{\mathbf{u}} + \mathbf{e}$

- If we wanted to compute it, we could solve the linear system

$$A_n\mathbf{e} = A_n\overline{\mathbf{u}} - A_n\widetilde{\mathbf{u}} = \mathbf{f} - A_n\widetilde{\mathbf{u}} = \mathbf{r}.$$

Still no gain?

We are back solving a linear system with the same coefficient matrix. **Nevertheless**, having swapped from the need of computing $\mathbf{u}$ to $\mathbf{e}$ gives us the chance to exploit the information that the error has been smoothed.

Multigrid based on geometry: Grids

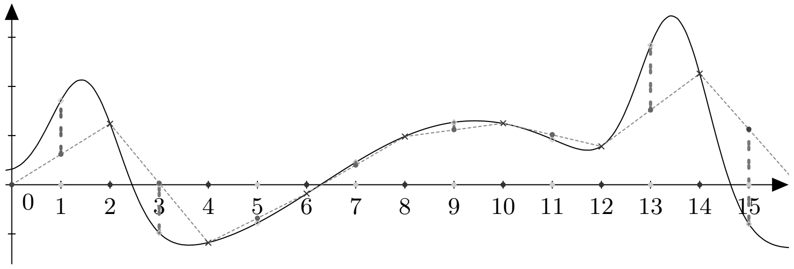
Let us consider the grids (for an odd n):

$$\Omega_h = \left\{ \frac{i\pi}{n+1} \;:\; i = 1, \dots, n \right\},$$
$$\Omega_{2h} = \left\{ \frac{2i\pi}{n+1} \;:\; i = 1, \dots, \frac{n-1}{2} \right\} = \left\{ \frac{i\pi}{\frac{n-1}{2}+1} \;:\; i = 1, \dots, \frac{n-1}{2} \right\},$$

We restrict the matrix and the residual vector on the coarse grid to **solve the error equation**

$$\Omega_h \rightsquigarrow A_n \mathbf{u} = \mathbf{f}, \quad \mathbf{u}, \mathbf{f} \in \mathbb{R}^n,$$
$$\Omega_{2h} \rightsquigarrow A_{\frac{n-1}{2}} \widetilde{\mathbf{e}} = \widetilde{\mathbf{r}}, \quad \widetilde{\mathbf{e}}, \widetilde{\mathbf{r}} \in \mathbb{R}^{\frac{n-1}{2}}.$$

Multigrid based on geometry: Grids



We restrict the matrix and the residual vector on the coarse grid to **solve the error equation**

$$\Omega_h \rightsquigarrow A_n \mathbf{u} = \mathbf{f}, \quad \mathbf{u}, \mathbf{f} \in \mathbb{R}^n,$$

$$\Omega_{2h} \rightsquigarrow A_{\frac{n-1}{2}} \widetilde{\mathbf{e}} = \widetilde{\mathbf{r}}, \quad \widetilde{\mathbf{e}}, \widetilde{\mathbf{r}} \in \mathbb{R}^{\frac{n-1}{2}}.$$

Multigrid based on geometry: the whole idea

1. We apply the **smoother** to smooth the error in the high frequency,
2. We use our **coarsening strategy** for the **error equation**
   2.1 a restriction operator $I_h^{2h} : \Omega_h \to \Omega_{2h}$,
   2.2 a prolongation operator $I_{2h}^{h} : \Omega_{2h} \to \Omega_h$,
   2.3 the discretization matrix at the lower level, i.e., $A_{n-1/2}$.
3. We can make additional distinctions between high and low frequencies for the error equation with respect to the actual grid Ω_{2h} and a coarser grid Ω_{4h} to **iterate our coupling of smoothing iterations and iterative refinement** by coarsening
4. We do something peculiar on the **coarsest grid** in which we face a very small linear system, possibly a single linear scalar equation, that can be solved efficiently by a direct method.

Multigrid based on geometry: the algorithm

```
Data: $\{A_k\}_{k=l}^{0}$, $l$, $\{S_k^{(1)}\}_{k=l}^{0}$, $\{S_k^{(2)}\}_{k=l}^{0}$, $\{I_k^{k-1}\}_{k=l}^{0}$ and $\{I_{k-1}^{k}\}_{k=l}^{0}$, $\mathbf{b}_k{}_{k=l}^{0}$, initial guess $\mathbf{u}^{(j)}$.
Output: Approximation $\mathbf{u}^{(j+1)}$ to the solution of $\mathbf{x}_l$.
Input: $\mathbf{u}_k^{(j+1)} = \mathrm{MGM}(A_k, \mathbf{b}_k, \mathbf{x}_k^{(j)}, k, \nu_1, \nu_2, \gamma)$
$\nu_1$ steps of presmoother $S_k^{(1)}$ applied to $A_k \widetilde{x}_k^{(j)} = \mathbf{b}_k$ ;                // Presmoothing
Compute the residual $\mathbf{r}_k^{(j)} = \mathbf{b}_k - A_k \widetilde{x}_k^{(j)}$ ;                  // Coarse Grid Correction
Restrict the residual $\mathbf{r}_{k-1}^{(j)} = I_k^{k-1} \mathbf{r}_k^{(j)}$;
if $k = 1$ then
    Direct solver for $A_{k-1} \mathbf{e}_{k-1}^{(j)}$;
else
    for $i = 1, \dots, \gamma$ do
        $\mathbf{e}_{k-1}^{(j)} = \mathrm{MGM}(A_{k-1}, \mathbf{r}_{k-1}, 0, k-1, \nu_1, \nu_2, \gamma)$
    end
end
Prolong the error $\mathbf{e}_k^{(j)} = I_{k-1}^{k} \mathbf{e}_{k-1}^{(j)}$;
Update the approximation $\mathbf{x}_k^{(j)} = \widetilde{\mathbf{x}}_k^{(j)} + \mathbf{e}_k^{(j)}$;
$\nu_2$ steps of postsmoother $S_k^{(2)}$ applied to $A_k \widetilde{x}_k^{(j+1)} = \mathbf{b}_k$ with initial guess $\mathbf{x}_k^{(j)}$ ;   // Postsmoothing
```

Multigrid based on geometry: convergence

We express the previous algorithm as the product by an **iteration matrix** M_l:

$$\begin{cases} M_0 = 0, & k = 0, \\ M_k = (S_k^{(1)})^{\nu_1} \left(I_k - I_{k-1}^{k} (I_{k-1} - M_{k-1}^{\gamma}) A_{k-1}^{-1} I_k^{k-1} A_k \right) (S_k^{(2)})^{\nu_2} & k \geq 1. \end{cases}.$$

Idea: it is a stationary method, thus it converges iff $\rho(M_k) < 1$.

Multigrid based on geometry: convergence

Convergence theorem

Let $A \in \mathbb{R}^{n\times n}$ be SPD. Assume that the prolongation operators I_{k-1}^{k} have full rank and that the Galerkin conditions holds

$$I_{k-1}^{k} = (I_{k}^{k-1})^{T}, \quad A_{k-1} = I_{k}^{k-1} A_k I_{k-1}^{k}, \; \forall\, k = l-1, \ldots, 0,$$

Furthermore, if the orthogonal projector $\Pi_k = I - I_{k+1}^{k} A_{k+1}^{-1} I_{k}^{k+1} A_k$, satisfies

$$\forall\, \mathbf{e}_k \, \exists\, \delta_1 > 0 \,:\, \|S_k^{(2)} \mathbf{e}_k\|_A^2 \leq \|\mathbf{e}_k\|_A^2 - \delta_1 \|\Pi_k \mathbf{e}_k\|_A^2,$$

independently of $\mathbf{e}_k$ and k, then the multigrid method with $\gamma = 1$, $\nu_1 = 0$ and $\nu_2 \geq 1$ (no *pre–smoother*), has a converge factor bounded above by $\sqrt{1-\delta_1}$ with $\delta_1 \leq 1$.

Multigrid based on geometry: convergence

Convergence theorem

Let $A \in \mathbb{R}^{n\times n}$ be SPD. Assume that the prolongation operators I_{k-1}^{k} have full rank and that the Galerkin conditions holds

$$I_{k-1}^{k} = (I_{k}^{k-1})^{T}, \quad A_{k-1} = I_{k}^{k-1} A_k I_{k-1}^{k},\ \forall\, k = l-1,\dots,0,$$

Furthermore, if the following condition holds

$$\forall\, \mathbf{e}_k\, \exists\, \delta_2 > 0\ :\ \|S_k^{(1)}\mathbf{e}_k\|_A^2 \leq \|\mathbf{e}_k\|_A^2 - \delta_2 \|\Pi_k S_k^{(1)} \mathbf{e}_k\|_A^2,$$

independently of $\mathbf{e}_k$ and k, then the multigrid method based with $\gamma = 1$, $\nu_1 \geq 1$ and $\nu_2 = 0$ (no *post–smoother*), has a converge factor bounded above by $1/\sqrt{1+\delta_2}$.

Multigrid based on geometry: convergence

Convergence theorem

Let $A \in \mathbb{R}^{n\times n}$ be SPD. Assume that the prolongation operators I_{k-1}^{k} have full rank and that the Galerkin conditions holds

$$I_{k-1}^{k} = (I_{k}^{k-1})^{T}, \quad A_{k-1} = I_{k}^{k-1} A_k I_{k-1}^{k}, \, \forall\, k = l-1, \ldots, 0,$$

Finally, if both estimate

$$\forall\, \mathbf{e}_k \,\exists\, \delta_1 > 0 \,:\, \|S_k^{(2)}\mathbf{e}_k\|_A^2 \leq \|\mathbf{e}_k\|_A^2 - \delta_1 \|\Pi_k \mathbf{e}_k\|_A^2,$$
$$\forall\, \mathbf{e}_k \,\exists\, \delta_2 > 0 \,:\, \|S_k^{(1)}\mathbf{e}_k\|_A^2 \leq \|\mathbf{e}_k\|_A^2 - \delta_2 \|\Pi_k S_k^{(1)} \mathbf{e}_k\|_A^2,$$

holds, for a pre– and post–smoother an estimate of the convergence factor is given by $\sqrt{1-\delta_1/1+\delta_2}$.

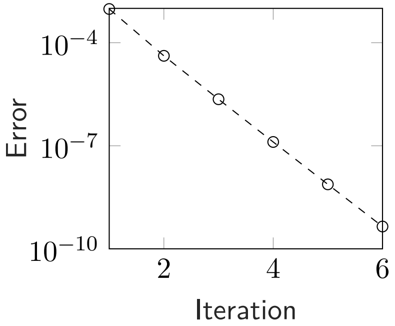
Concluding the example

We test the simple *recursive* implementation from

E6-SimpleGMG/ex_toepmultigrid.m

We consider the 1D Poisson problem with

- Jacobi smoother with optimal parameter,
- $nu_1 = \nu_2 = 2$ smoother steps,
- Use **linear interpolation**,
- Impose Galerkin conditions,
- Use **multigrid as a solver**.



Concluding the example

We test the simple *recursive* implementation from

`E6-SimpleGMG/ex_toepmultigrid.m`

We consider the 1D Poisson problem with

- Jacobi smoother with optimal parameter,
- $nu_1 = \nu_2 = 2$ smoother steps,
- Use **linear interpolation**,
- Impose Galerkin conditions,
- Use **multigrid as a solver**.

Convergence result on finer meshes

```
Size 63 Iteration 5 Time 6.72e-03
Size 127 Iteration 5 Time 5.61e-03
Size 255 Iteration 6 Time 1.91e-02
Size 511 Iteration 5 Time 5.40e-02
Size 1023 Iteration 6 Time 1.42e-01
Size 2047 Iteration 6 Time 5.85e-01
Size 4095 Iteration 6 Time 2.36e+00
```

Concluding the example

We test the simple *recursive* implementation from

`E6-SimpleGMG/ex_toepmultigrid.m`

We consider the 1D Poisson problem with

- Jacobi smoother with optimal parameter,
- $nu_1 = \nu_2 = 2$ smoother steps,
- Use **linear interpolation**,
- Impose Galerkin conditions,
- Use **multigrid as a solver**.

Convergence result on finer meshes

```
Size 63 Iteration 5 Time 6.72e-03
Size 127 Iteration 5 Time 5.61e-03
Size 255 Iteration 6 Time 1.91e-02
Size 511 Iteration 5 Time 5.40e-02
Size 1023 Iteration 6 Time 1.42e-01
Size 2047 Iteration 6 Time 5.85e-01
Size 4095 Iteration 6 Time 2.36e+00
```

Tests and extensions

The code contains other test problems with which you can play around. To get **better performances** you could re-implement the algorithm in a *non recursive way*.

More general geometries

One of the **major question** is now:

> "How do we find interpolation operators and smoother satisfying the convergence theorem?"

More general geometries

One of the **major question** is now:

"How do we find interpolation operators and smoother satisfying the convergence theorem?"

- For structured matrices (Toeplitz, Circulant, τ-algebra, *etc.*) we can discharge the problem on the properties of some functions describing the spectrum. Unfortunately, this is usually possible only when the matrix is obtained from the discretization of a PDE on a **structured** or **uniform** grid.

More general geometries

One of the **major question** is now:

"How do we find interpolation operators and smoother satisfying the convergence theorem?"

- For structured matrices (Toeplitz, Circulant, τ-algebra, *etc.*) we can discharge the problem on the properties of some functions describing the spectrum. Unfortunately, this is usually possible only when the matrix is obtained from the discretization of a PDE on a **structured** or **uniform** grid.
- In general, our linear system could be coming from an **optimization problem**, being the **Laplacian of a graph**, being the **discretization** of a differential operator on an **unstructured grid**

The way forward

We will reformulate the algorithm to use only purely algebraic properties of the matrix.

Revisiting the components: A-convergent smoothers

To build the “source agnostic” Multigrid we start by revisiting the constitutive components.

Theorem (A-convergent spliting)

Let A be SPD. Assume that for a given M the iteration matrix $I - M^{-1}A$ has an A-norm less than one, or, equivalently that

$$\|I - A^{1/2}M^{-1}A^{1/2}\| < 1.$$

The symmetrization $\overline{M} = M(M + M^T - A)^{-1}M^T$ satisfies

(i) $I - \overline{M}^{-1}A = (I - M^{-T}A)(I - M^{-1}A)$,

(ii) $\overline{M} - A$ is Symmetric Positive Semidefinite

(iii) $\|I - A^{1/2}M^{-1}A^{1/2}\| = \|I - A^{1/2}\overline{M}^{-1}A^{1/2}\|$,

(iv) $\|I - A^{1/2}M^{-1}A^{1/2}\| < 1 \Leftrightarrow M + M^T - A$ SPD

Revisiting the components: 2×2-block factorization

Given $A \in \mathbb{R}^{n \times n}$ SPD, we let J and P be two **rectangular matrices** with n rows so that we can consider the 2×2-block factorization:

$$A = \begin{bmatrix} \mathcal{A} & \mathcal{R} \\ \mathcal{L} & \mathcal{B} \end{bmatrix}, \quad \mathcal{A} = J^T A J, \quad \mathcal{B} = P^T A P.$$

We now suppose having two matrices $\mathcal{M} \approx \mathcal{A}$ and $\mathcal{D} \approx \mathcal{B}$ with $\mathcal{D}$ SPD, where "$\approx$" (usually) means that $\mathcal{M}$ and $\mathcal{D}$ are $\mathcal{A}/\mathcal{B}$-convergent splitting.

Revisiting the components: 2×2-block factorization

Given $A \in \mathbb{R}^{n \times n}$ SPD, we let J and P be two **rectangular matrices** with n rows so that we can consider the 2×2-block factorization:

$$A = \begin{bmatrix} \mathcal{A} & \mathcal{R} \\ \mathcal{L} & \mathcal{B} \end{bmatrix}, \quad \mathcal{A} = J^T A J, \quad \mathcal{B} = P^T A P.$$

We now suppose having two matrices $\mathcal{M} \approx \mathcal{A}$ and $\mathcal{D} \approx \mathcal{B}$ with $\mathcal{D}$ SPD, where "$\approx$" (usually) means that $\mathcal{M}$ and $\mathcal{D}$ are $\mathcal{A}/\mathcal{B}$-convergent splitting.

```
Let $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$;
Use method $\mathcal{M}$ for $(J^T A J)\mathbf{x} = J^T \mathbf{r}_0,$;
Compute the residual $\mathbf{r}_m = b - A\mathbf{u}_m = b - A\mathbf{u}_0 - AJ\mathbf{x}_m = (I - AJ\mathcal{M}^{-1}J^T)\mathbf{r}_0$;
```

Algorithm 1: Product iteration method

Revisiting the components: 2×2-block factorization

Given $A \in \mathbb{R}^{n \times n}$ SPD, we let J and P be two **rectangular matrices** with n rows so that we can consider the 2×2-block factorization:

$$A = \begin{bmatrix} \mathcal{A} & \mathcal{R} \\ \mathcal{L} & \mathcal{B} \end{bmatrix}, \quad \mathcal{A} = J^T A J, \quad \mathcal{B} = P^T A P.$$

We now suppose having two matrices $\mathcal{M} \approx \mathcal{A}$ and $\mathcal{D} \approx \mathcal{B}$ with $\mathcal{D}$ SPD, where "$\approx$" (usually) means that $\mathcal{M}$ and $\mathcal{D}$ are $\mathcal{A}/\mathcal{B}$-convergent splitting.

```
Let $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$;
Use method $\mathcal{M}$ for $(J^T A J)\mathbf{x} = J^T \mathbf{r}_{0,}$;
Compute the residual $\mathbf{r}_m = b - A\mathbf{u}_m = b - A\mathbf{u}_0 - AJ\mathbf{x}_m = (I - AJ\mathcal{M}^{-1}J^T)\mathbf{r}_0$;
Use method $\mathcal{D}$ for $(P^T A P)\mathbf{w} = P^T \mathbf{r}_m$;
Compute the residual $\mathbf{r}_w = \mathbf{b} - A\mathbf{u}_w = \mathbf{b} - A\mathbf{u}_m - AP\mathbf{w} = (I - AP\mathcal{D}^{-1}P^T)(I - AJ\mathcal{M}^{-1}J^T)\mathbf{r}_0$;
```

Algorithm 2: Product iteration method

Revisiting the components: 2×2-block factorization

Given $A \in \mathbb{R}^{n \times n}$ SPD, we let J and P be two **rectangular matrices** with n rows so that we can consider the 2×2-block factorization:

$$A = \begin{bmatrix} \mathcal{A} & \mathcal{R} \\ \mathcal{L} & \mathcal{B} \end{bmatrix}, \quad \mathcal{A} = J^T A J, \quad \mathcal{B} = P^T A P.$$

We now suppose having two matrices $\mathcal{M} \approx \mathcal{A}$ and $\mathcal{D} \approx \mathcal{B}$ with $\mathcal{D}$ SPD, where "$\approx$" (usually) means that $\mathcal{M}$ and $\mathcal{D}$ are $\mathcal{A}/\mathcal{B}$-convergent splitting.

```
Let $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$;
Use method $\mathcal{M}$ for $(J^T A J)\mathbf{x} = J^T \mathbf{r}_{0,}$;
Compute the residual $\mathbf{r}_m = b - A\mathbf{u}_m = b - A\mathbf{u}_0 - AJ\mathbf{x}_m = (I - AJ\mathcal{M}^{-1}J^T)\mathbf{r}_0$;
Use method $\mathcal{D}$ for $(P^T A P)\mathbf{w} = P^T \mathbf{r}_m$;
Compute the residual $\mathbf{r}_w = \mathbf{b} - A\mathbf{u}_w = \mathbf{b} - A\mathbf{u}_m - AP\mathbf{w} = (I - AP\mathcal{D}^{-1}P^T)(I - AJ\mathcal{M}^{-1}J^T)\mathbf{r}_0$;
Use method $\mathcal{M}$ for $(J^T A J)\mathbf{x} = J^T \mathbf{r}_{w,}$;
The new residual is
  $\mathbf{r}_{\text{new}} = \mathbf{b} - A\mathbf{u}_{\text{new}} = \mathbf{b} - A\mathbf{u}_w - AJ\mathbf{x} = (I - AJ\mathcal{M}^{-T}J^T)(I - AP\mathcal{D}^{-1}P^T)(I - AJ\mathcal{M}^{-1}J^T)\mathbf{r}_0$
```

Algorithm 3: Product iteration method

Revisiting the components: 2×2-block factorization

Given $A \in \mathbb{R}^{n\times n}$ SPD, we let J and P be two **rectangular matrices** with n rows so that we can consider the 2×2-block factorization:

$$A = \begin{bmatrix} \mathcal{A} & \mathcal{R} \\ \mathcal{L} & \mathcal{B} \end{bmatrix}, \quad \mathcal{A} = J^T A J, \quad \mathcal{B} = P^T A P.$$

We now suppose having two matrices $\mathcal{M} \approx \mathcal{A}$ and $\mathcal{D} \approx \mathcal{B}$ with $\mathcal{D}$ SPD, where "$\approx$" (usually) means that $\mathcal{M}$ and $\mathcal{D}$ are $\mathcal{A}/\mathcal{B}$-convergent splitting.

Residual iteration matrix

The residual iteration E_r is therefore given by:

$$E_r = \mathbf{b} - A\mathbf{u}_{\text{new}} = \mathbf{b} - A\mathbf{u}_w - AJ\mathbf{x} = (I - AJ\mathcal{M}^{-T}J^T)(I - AP\mathcal{D}^{-1}P^T)(I - AJ\mathcal{M}^{-1}J^T),$$

thus $\mathbf{u} - \mathbf{u}_0 = A^{-1}\mathbf{r}_0 \mapsto \mathbf{u} - \mathbf{u}_{\text{new}} = A^{-1}\mathbf{r}_{\text{new}}$ since $AE = E_r A$

$$E = (I - J\mathcal{M}^{-T}J^T A)(I - P\mathcal{D}^{-1}P^T A)(I - J\mathcal{M}^{-1}J^T A) = A^{-1}E_r A.$$

Revisiting the components: 2×2-block factorization

Lemma

If $\mathcal{M}$ and $\mathcal{D}$ are convergent smoother then $\|E\mathbf{e}\|_A \leq \|\mathbf{e}\|_A$.

Revisiting the components: 2×2-block factorization

Lemma

If $\mathcal{M}$ and $\mathcal{D}$ are convergent smoother then $\|E\mathbf{e}\|_A \leq \|\mathbf{e}\|_A$.

Block-factorizations and product iteration methods

We implicitly define the product iteration method

$$I - B^{-1}A = (I - J\mathcal{M}^{-T}J^TA)(I - P\mathcal{D}^{-1}P^TA)(I - J\mathcal{M}^{-1}J^TA).$$

Revisiting the components: 2×2-block factorization

Lemma

If $\mathcal{M}$ and $\mathcal{D}$ are convergent smoother then $\|E\mathbf{e}\|_A \leq \|\mathbf{e}\|_A$.

Block-factorizations and product iteration methods

We implicitly define the product iteration method

$$I - B^{-1}A = (I - J\mathcal{M}^{-T}J^TA)(I - P\mathcal{D}^{-1}P^TA)(I - J\mathcal{M}^{-1}J^TA).$$

Theorem

Let $\overline{\mathcal{M}} = \mathcal{M}(\mathcal{M} + \mathcal{M}^T - \mathcal{A})^{-1}\mathcal{M}^T$, given the following block-factored matrix

$$\hat{\mathcal{B}} = \begin{bmatrix} \mathcal{M} & O \\ P^T\mathcal{A}J & I \end{bmatrix} \begin{bmatrix} (\mathcal{M} + \mathcal{M}^T - \mathcal{A})^{-1} & O \\ O & \mathcal{D} \end{bmatrix} \begin{bmatrix} \mathcal{M}^T & J^TAP \\ O & I \end{bmatrix}$$

we express explicitly the operator as

$$B^{-1} = [J, P]\hat{\mathcal{B}}[J, P]^T = J\overline{\mathcal{M}}^{-1}J^T + (I - J\overline{\mathcal{M}}^{-T}J^TA)P\mathcal{D}^{-1}P^T(I - \mathcal{A}J\mathcal{M}^{-1}J^T).$$

Revisiting the components: 2×2-block factorization

What did we just prove?

1. We can build a block-factorization preconditioner B^{-1} as $[J, P]\hat{\mathcal{B}}[J, P]^T$,
2. The matrix $\hat{\mathcal{B}}$ is obtained from the approximate block-factorization of the two-by-two block matrix $\hat{\mathcal{A}} = [J, P]^T A[J, P]$,
3. The stationary matrix iteration $I - B^{-1}A$ can be expressed as the product

$$(I - J\mathcal{M}^{-T}J^TA)(I - P\mathcal{D}^{-1}P^TA)(I - J\mathcal{M}^{-1}J^TA),$$

   that act on $\mathrm{Range}(J)$, $\mathrm{Range}(P)$, and $\mathrm{Range}(J)$.

Revisiting the components: 2×2-block factorization

What did we just prove?

1. We can build a block-factorization preconditioner B^{-1} as $[J, P]\hat{\mathcal{B}}[J, P]^T$,
2. The matrix $\hat{\mathcal{B}}$ is obtained from the approximate block-factorization of the two-by-two block matrix $\hat{\mathcal{A}} = [J, P]^T A[J, P]$,
3. The stationary matrix iteration $I - B^{-1}A$ can be expressed as the product

$$(I - J\mathcal{M}^{-T}J^TA)(I - P\mathcal{D}^{-1}P^TA)(I - J\mathcal{M}^{-1}J^TA),$$

that act on $\mathrm{Range}(J)$, $\mathrm{Range}(P)$, and $\mathrm{Range}(J)$.

We have written, using block factorization, a **method of the type we saw in our geometric example** on the 1D Laplacian. The **high** and **low frequency** spaces are then represented as the **images** and **kernels** of the J and P maps.

Two-grid preconditioner

For $J = I$ then $[J, P]^T$ has full column rank since $[I, P][I, P]^T = I + PP^T$ is SPD.

Two-grid preconditioner

Given a smoother M for A and an interpolation matrix P, and let $\mathcal{D}$ be an SPD approximation to $\mathcal{B} = P^T AP$, such that

- $M + M^T - A$ is SPD (equivalently, $\|I - A^{1/2}M^{-1}A^{1/2}\| < 1$).
- $\mathcal{D} - \mathcal{B}$ is SPD.

Then, given the block matrix

$$\hat{\mathcal{B}} = \begin{bmatrix} M & O \\ P^T A & I \end{bmatrix} \begin{bmatrix} (M^T + M - A)^{-1} & O \\ O & \mathcal{D} \end{bmatrix} \begin{bmatrix} M^T & AP \\ O & I \end{bmatrix}$$

we define the preconditioner $B_{TG}^{-1} = [I, P]\hat{\mathcal{B}}[I, P]^T$ or, equivalently,

$$B_{TG}^{-1} = \overline{M}^{-1} + (I - AM^{-1})^T P\mathcal{D}^{-1}P^T(I - M^{-1}A).$$

Two-grid preconditioner: convergence

What can we say about the **convergence properties** of such method?

Two-grid preconditioner: convergence

What can we say about the **convergence properties** of such method?

Convergence constant

We would like to estimate the best constant

$$\mathbf{v}^T A\mathbf{v} \leq \mathbf{v}^T B_{TG}\mathbf{v} \leq K_{TG}\mathbf{v}^T A\mathbf{v}.$$

Two-grid preconditioner: convergence

What can we say about the **convergence properties** of such method?

Convergence constant

We would like to estimate the best constant

$$\mathbf{v}^T A \mathbf{v} \le \mathbf{v}^T B_{TG} \mathbf{v} \le K_{TG} \mathbf{v}^T A \mathbf{v}.$$

Theorem

Assume that J and P are such that any vector $\mathbf{v}$ can be decomposed as $\mathbf{v} = J\mathbf{w} + P\mathbf{x}$. We introduce the projectors $\pi_A = P\mathcal{B}^{-1}P^T A = P(P^T A P)^{-1} P^T$, and let $\widetilde{\mathcal{M}} = \mathcal{M}^T(\mathcal{M} + \mathcal{M}^T - A)^{-1}\mathcal{M}$. The best constant K is given by

$$K = \sup_{\mathbf{v} \in \mathrm{Range}(I-\pi_A)} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)J\mathbf{w}} \frac{\mathbf{w}^T \widetilde{\mathcal{M}} \mathbf{w}}{\mathbf{v}^T A \mathbf{v}} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)J\mathbf{w}} \frac{\mathbf{w}^T \widetilde{\mathcal{M}} \mathbf{w}}{\mathbf{v}^T A (I-\pi_A) \mathbf{v}}.$$

Two-grid preconditioner: convergence

What can we say about the **convergence properties** of such method?

Convergence constant

We would like to estimate the best constant

$$\mathbf{v}^T A \mathbf{v} \leq \mathbf{v}^T B_{TG} \mathbf{v} \leq K_{TG} \mathbf{v}^T A \mathbf{v}.$$

Theorem

Assume that J and P are such that any vector $\mathbf{v}$ can be decomposed as $\mathbf{v} = J\mathbf{w} + P\mathbf{x}$ with $[J, P]$ invertible. We introduce the projectors $\pi_A = P\mathcal{B}^{-1}P^T A = P(P^T AP)^{-1}P^T$, and let $\widetilde{\mathcal{M}} = \mathcal{M}^T(\mathcal{M} + \mathcal{M}^T - A)^{-1}\mathcal{M}$. The best constant K is given by

$$K = \sup_{\mathbf{w}} \frac{\mathbf{w}^T \widetilde{\mathcal{M}} \mathbf{w}}{\mathbf{w}^T J^T A(I - \pi_A) J\mathbf{w}}.$$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$.

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$. We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}}$$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$.
We apply the previous theorem and find:

$$
\begin{aligned}
K_{TG} &= \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}} \\
&= \sup_{\mathbf{v}} \frac{\inf_{\mathbf{w}} \left[(\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v})^T \widetilde{M}(\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v})\right]}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})}
\end{aligned}
$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$. We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}}$$

$$= \sup_{\mathbf{v}} \frac{\inf_{\mathbf{w}} \left[(\pi_A \mathbf{w} + (I-\pi_A)\mathbf{v})^T \widetilde{M} (\pi_A \mathbf{w} + (I-\pi_A)\mathbf{v}) \right]}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,

$\Rightarrow$ the $\inf_{\mathbf{w}}[\cdot]$ is attained a $\mathbf{w}$ such that $\pi_A(\mathbf{v}-\mathbf{w}) = \pi_{\widetilde{M}}\mathbf{v}$, i.e.,

$$A_c^{-1}P^TA(\mathbf{v}-\mathbf{w}) = \widetilde{M}^{-1}P^T\widetilde{M}\mathbf{v}$$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$. We apply the previous theorem and find:

$$\begin{aligned} K_{TG} &= \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}} \\ &= \sup_{\mathbf{v}} \frac{\inf_{\mathbf{w}} \left[(\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v})^T \widetilde{M}(\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v})\right]}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})} \end{aligned}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,

$\Rightarrow$ the $\inf_{\mathbf{w}}[\cdot]$ is attained a $\mathbf{w}$ such that $\pi_A(\mathbf{v}-\mathbf{w}) = \pi_{\widetilde{M}}\mathbf{v}$, i.e.,

$$A_c^{-1}P^TA(\mathbf{v}-\mathbf{w}) = \widetilde{M}^{-1}P^T\widetilde{M}\mathbf{v} \;\Rightarrow\; \mathbf{w}_c = A_c^{-1}P^TA\mathbf{v} - \widetilde{M}_c^{-1}P^T\widetilde{M}\mathbf{v}.$$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$. We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}}$$

$$= \sup_{\mathbf{v}} \frac{\inf_{\mathbf{w}} \left[(\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v})^T \widetilde{M}(\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v})\right]}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,

$\Rightarrow$ the $\inf_{\mathbf{w}}[\cdot]$ is attained a $\mathbf{w}$ such that $\pi_A(\mathbf{v}-\mathbf{w}) = \pi_{\widetilde{M}}\mathbf{v}$, i.e.,

$$A_c^{-1}P^TA(\mathbf{v}-\mathbf{w}) = \widetilde{M}^{-1}P^T\widetilde{M}\mathbf{v} \;\Rightarrow\; \mathbf{w}_c = A_c^{-1}P^TA\mathbf{v} - \widetilde{M}_c^{-1}P^T\widetilde{M}\mathbf{v}.$$

$\Rightarrow$ $P\mathbf{w}_c = \pi_A\mathbf{w} = \mathbf{w} = (\pi_A - \pi_{\widetilde{M}})\mathbf{v}$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$. We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}}$$

$$= \sup_{\mathbf{v}} \frac{\inf_{\mathbf{w}} \left[(\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v})^T \widetilde{M} (\pi_A\mathbf{w} + (I-\pi_A)\mathbf{v}) \right]}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,

$\Rightarrow$ the $\inf_{\mathbf{w}}[\cdot]$ is attained a $\mathbf{w}$ such that $\pi_A(\mathbf{v}-\mathbf{w}) = \pi_{\widetilde{M}}\mathbf{v}$, i.e.,

$$A_c^{-1}P^TA(\mathbf{v}-\mathbf{w}) = \widetilde{M}^{-1}P^T\widetilde{M}\mathbf{v} \;\Rightarrow\; \mathbf{w}_c = A_c^{-1}P^TA\mathbf{v} - \widetilde{M}_c^{-1}P^T\widetilde{M}\mathbf{v}.$$

$\Rightarrow$ $P\mathbf{w}_c = \pi_A\mathbf{w} = \mathbf{w} = (\pi_A - \pi_{\widetilde{M}})\mathbf{v} \Rightarrow \pi_A\mathbf{w} + (I-\pi_A)\mathbf{v} = (I-\pi_{\widetilde{M}})\mathbf{v}$.

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T A P$. We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}}$$
$$= \sup_{\mathbf{v}} \frac{\left((I-\pi_{\widetilde{M}})\mathbf{v}\right)^T \widetilde{M}\left((I-\pi_{\widetilde{M}})\mathbf{v}\right)}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,

$\Rightarrow$ the $\inf_{\mathbf{w}}[\cdot]$ is attained a $\mathbf{w}$ such that $\pi_A(\mathbf{v}-\mathbf{w}) = \pi_{\widetilde{M}}\mathbf{v}$, i.e.,

$$A_c^{-1}P^T A(\mathbf{v}-\mathbf{w}) = \widetilde{M}^{-1}P^T\widetilde{M}\mathbf{v} \;\Rightarrow\; \mathbf{w}_c = A_c^{-1}P^T A\mathbf{v} - \widetilde{M}_c^{-1}P^T\widetilde{M}\mathbf{v}.$$

$\Rightarrow$ $P\mathbf{w}_c = \pi_A\mathbf{w} = \mathbf{w} = (\pi_A - \pi_{\widetilde{M}})\mathbf{v} \Rightarrow \pi_A\mathbf{w} + (I-\pi_A)\mathbf{v} = (I-\pi_{\widetilde{M}})\mathbf{v}.$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$.
We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w} : \mathbf{v} = (I - \pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M} \mathbf{w}}{\mathbf{v}^T A \mathbf{v}}$$

$$= \sup_{\mathbf{v}} \frac{\big((I - \pi_{\widetilde{M}})\mathbf{v}\big)^T \widetilde{M} \big((I - \pi_{\widetilde{M}})\mathbf{v}\big)}{((I - \pi_A)\mathbf{v})^T A((I - \pi_A)\mathbf{v})} = \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I - \pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A \mathbf{v}}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1} P^T \widetilde{M} = P(P^T \widetilde{M} P)^{-1} P^T \widetilde{M}$,
- $(I - \pi_{\widetilde{M}})P = P - P(P^T \widetilde{M} P)^{-1} P^T \widetilde{M} P = 0 \Rightarrow (I - \pi_{\widetilde{M}})(I - \pi_A) = I - \pi_{\widetilde{M}}$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$.
We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}}$$

$$= \sup_{\mathbf{v}} \frac{\left((I-\pi_{\widetilde{M}})\mathbf{v}\right)^T \widetilde{M}\left((I-\pi_{\widetilde{M}})\mathbf{v}\right)}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})} = \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I-\pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A\mathbf{v}}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,
- $(I-\pi_{\widetilde{M}})P = P - P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}P = 0 \Rightarrow (I-\pi_{\widetilde{M}})(I-\pi_A) = I-\pi_{\widetilde{M}}$

$$K_{TG} = \sup_{\mathbf{v}=(I-\pi_A)\mathbf{v}} \frac{\left((I-\pi_{\widetilde{M}})\mathbf{v}\right)^T \widetilde{M}\left((I-\pi_{\widetilde{M}})\mathbf{v}\right)}{\mathbf{v}^T A\mathbf{v}} \leq \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I-\pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A\mathbf{v}},$$

Two-grid preconditioner: convergence

For the **two-grid case** we have $J = I$, $\mathcal{M} = M$ a *smoother* for A and $\mathcal{D} = \mathcal{B} = P^T AP$. We apply the previous theorem and find:

$$K_{TG} = \sup_{\mathbf{v}} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\mathbf{w}^T \widetilde{M}\mathbf{w}}{\mathbf{v}^T A\mathbf{v}}$$

$$= \sup_{\mathbf{v}} \frac{\left((I-\pi_{\widetilde{M}})\mathbf{v}\right)^T \widetilde{M} \left((I-\pi_{\widetilde{M}})\mathbf{v}\right)}{((I-\pi_A)\mathbf{v})^T A((I-\pi_A)\mathbf{v})} = \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I-\pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A\mathbf{v}}$$

- We introduce $\pi_{\widetilde{M}} = P\widetilde{M}_c^{-1}P^T\widetilde{M} = P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}$,
- $(I-\pi_{\widetilde{M}})P = P - P(P^T\widetilde{M}P)^{-1}P^T\widetilde{M}P = 0 \Rightarrow (I-\pi_{\widetilde{M}})(I-\pi_A) = I - \pi_{\widetilde{M}}$
- $\mathbf{v}^T A\mathbf{v} \geq \mathbf{v}^T A(I-\pi_A)\mathbf{v}$ and thus the opposite inequality holds:

$$\sup_{\mathbf{v}} \frac{\mathbf{v}^T\widetilde{M}(I-\pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A\mathbf{v}} \leq \sup_{\mathbf{v}} \frac{\mathbf{v}^T\widetilde{M}(I-\pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A(I-\pi_A)\mathbf{v}} = K_{TG}.$$

Two-grid preconditioner: convergence

The first take-home message

The Theorem we have just seen shows that

$$\rho(E_{TG}) = 1 - \frac{1}{K_{TG}}, \quad K_{TG} = \begin{cases} \sup\limits_{\mathbf{v}} \dfrac{\mathbf{v}^T \widetilde{M}(I - \pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A \mathbf{v}}, \\ \sup\limits_{\mathbf{v}} \dfrac{\|(I - PR)\mathbf{v}\|_{\widetilde{M}}^2}{\|\mathbf{v}\|_A^2}, \end{cases} \qquad \pi_{\widetilde{M}} = P(P^T \widetilde{M} P)^{-1} P^T \widetilde{M},$$

for $\widetilde{M} = M^T(M + M^T - A)^{-1}M$, and $R = (P^T \widetilde{M} P)^{-1} P^T \widetilde{M}$.

Observe that RP is the identity on the coarse space.

Two-grid preconditioner: convergence

Working with the symmetrized smoother $\widetilde{M}$ is useful for proving estimates, but not so much for estimating constants.

Corollary

Let $\widetilde{M}$ be spectrally equivalent to an SPD matrix D, i.e., such that

$$\exists\, c_1, c_2 > 0 \,:\, c_1 \mathbf{v}^T D\mathbf{v} \le \mathbf{v}^T \widetilde{\mathcal{M}}\mathbf{v} \le c_2 \mathbf{v}^T D\mathbf{v} \quad \forall v.$$

Then, with $\pi_D = P(P^T DP)^{-1}P^T D$ the following estimate for K_{TG} holds

$$c_1 \sup_{\mathbf{v}} \frac{\mathbf{v}^T D(I - \pi_D)\mathbf{v}}{\mathbf{v}^T A\mathbf{v}} \le K_{TG} \le c_2 \sup_{\mathbf{v}} \frac{\mathbf{v}^T D(I - \pi_D)\mathbf{v}}{\mathbf{v}^T A\mathbf{v}}.$$

Two-grid preconditioner: convergence

Example

If M is SPD and such that $M - A$ is positive semidefinite, $\widetilde{M} = M(2M - A)^{-1}M$ is spectrally equivalent to M such that

$$\frac{1}{2}\mathbf{v}^T M\mathbf{v} \leq \mathbf{v}^T \widetilde{M}\mathbf{v} \leq \mathbf{v}^T M\mathbf{v},$$

thus $c_1 = 1/2$, and $c_2 = 1$.

Two-grid preconditioner: convergence

Example

If M is the Gauss-Seidel iteration matrix, i.e., $M = D - L$, then $\widetilde{M} = (D - U)D^{-1}(D - L)$ is spectrally equivalent to D with

$$\frac{1}{4}\mathbf{v}^T M\mathbf{v} \leq \mathbf{v}^T \widetilde{M}\mathbf{v} \leq \kappa^2 \mathbf{v}^T M\mathbf{v},$$

thus $c_1 = {}^1\!/_4$, and $c_2 = \kappa$ the maximum number of nonzero entries of A per row.

We know how to estimate these quantities for both Jacobi and Gauss–Seidel type methods.

Increasing the number of levels

We have described the two-grid method as

$$\hat{\mathcal{B}} = \begin{bmatrix} M & O \\ P^T A & I \end{bmatrix} \begin{bmatrix} (M^T + M - A)^{-1} & O \\ O & A_c \end{bmatrix} \begin{bmatrix} M^T & AP \\ O & I \end{bmatrix}, \quad \mathcal{D} = A_c = P^T A P,$$

Increasing the number of levels

We have described the two-grid method as

$$\hat{\mathcal{B}} = \begin{bmatrix} I & O \\ P^T A M^{-1} & I \end{bmatrix} \begin{bmatrix} M(M^T + M - A)^{-1} M^T & O \\ O & A_c \end{bmatrix} \begin{bmatrix} I & M^{-T} A P \\ O & I \end{bmatrix}, \quad \mathcal{D} = A_c = P^T A P,$$

- For the following discussion is better to represent it by having the block triangular matrix with unit diagonal,

Increasing the number of levels

We have described the two-grid method as

$$\hat{\mathcal{B}} = \begin{bmatrix} I & O \\ P^T A & I \end{bmatrix} \begin{bmatrix} (M^T + M - A)^{-1} M^T & O \\ O & A_c \end{bmatrix} \begin{bmatrix} I & AP \\ O & I \end{bmatrix}, \quad \mathcal{D} = A_c = P^T A P,$$

- For the following discussion is better to represent it by having the block triangular matrix with unit diagonal,
- Assume that we have $\ell \geq 1$ levels and define
  - $A_0 = A$,
  - $P_k : V_{k+1} \equiv \mathbb{R}^{n_{k+1}} \mapsto V_k \equiv \mathbb{R}^{n_k}$ interpolation matrix $PV_{k+1} \subset V_k$,
  - $A_{k+1} = P_k^T A_k P_k$ coarse-grid $k+1$ matrix,
  - M_k a convergent smoother for A_k, i.e., $\|I - A_k^{1/2} M_k^{-1} A_k^{1/2}\| < 1$.

Increasing the number of levels

We have described the two-grid method as

$$\overline{B}_k = \begin{bmatrix} I & O \\ P_k^T A_k M_k^{-1} & I \end{bmatrix} \begin{bmatrix} M_k(M_k^T + M_k - A_k)^{-1} M_k^T & O \\ O & B_{k+1} \end{bmatrix} \begin{bmatrix} I & M_k^{-T} A_k P_k \\ O & I \end{bmatrix}$$

- For the following discussion is better to represent it by having the block triangular matrix with unit diagonal,
- Assume that we have $\ell \geq 1$ levels and define
  - $A_0 = A$,
  - $P_k \,:\, V_{k+1} \equiv \mathbb{R}^{n_{k+1}} \mapsto V_k \equiv \mathbb{R}^{n_k}$ interpolation matrix $PV_{k+1} \subset V_k$,
  - $A_{k+1} = P_k^T A_k P_k$ coarse-grid $k+1$ matrix,
  - M_k a convergent smoother for A_k, i.e., $\|I - A_k^{1/2} M_k^{-1} A_k^{1/2}\| < 1$.
- With this ingredient we define a MG as a recursive 2×2 block-factorization preconditioner $B_k^{-1} = [I, P_k]\overline{B}_k^{-1}[I, P_k]^T$.

Increasing the number of levels

```
At the coarsest level set $B_\ell = A_\ell$, the action of $B_k^{-1}\mathbf{r}$ is given by;
Solve for $M_k\mathbf{x}_k = \mathbf{r}$;                                        /* Presmooth */
Compute the residual $\mathbf{d} = \mathbf{r} - A_k\mathbf{x}_k = (I - A_kM_k^{-1})\mathbf{r}$;
Compute $\mathbf{x}_{k+1} = B_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;          /* Coarse grid correction */
Update $\mathbf{x}_k = \mathbf{x}_k + P\mathbf{x}_{k+1} = M_k^{-1}\mathbf{r} + P_kB_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;
Solve for $M_k^T\mathbf{y} = \mathbf{r} - A_k\mathbf{x}_k$;                              /* Postsmooth */
Set $B_k^{-1}\mathbf{r} = \mathbf{x}_k + \mathbf{y}$.
```

Algorithm 4: Generic MG Algorithm

Increasing the number of levels

```
At the coarsest level set $B_\ell = A_\ell$, the action of $B_k^{-1}\mathbf{r}$ is given by;
Solve for $M_k\mathbf{x}_k = \mathbf{r}$;                                   /* Presmooth */
Compute the residual $\mathbf{d} = \mathbf{r} - A_k\mathbf{x}_k = (I - A_kM_k^{-1})\mathbf{r}$;
Compute $\mathbf{x}_{k+1} = B_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;          /* Coarse grid correction */
Update $\mathbf{x}_k = \mathbf{x}_k + P\mathbf{x}_{k+1} = M_k^{-1}\mathbf{r} + P_kB_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;
Solve for $M_k^T\mathbf{y} = \mathbf{r} - A_k\mathbf{x}_k$;                            /* Postsmooth */
Set $B_k^{-1}\mathbf{r} = \mathbf{x}_k + \mathbf{y}$.
```

Algorithm 5: Generic MG Algorithm

That is:

$$B_k^{-1}\mathbf{r} = (M_k^{-1} + M_k^{-T} - M_k^{-1}A_kM_k^{-T} + (I - M_k^{-T}A_k)P_kB_{k+1}^{-1}P_k^T(I - A_kM_k^{-1}))\mathbf{r}$$

Increasing the number of levels

```
At the coarsest level set $B_\ell = A_\ell$, the action of $B_k^{-1}\mathbf{r}$ is given by;
Solve for $M_k\mathbf{x}_k = \mathbf{r}$;                                  /* Presmooth */
Compute the residual $\mathbf{d} = \mathbf{r} - A_k\mathbf{x}_k = (I - A_kM_k^{-1})\mathbf{r}$;
Compute $\mathbf{x}_{k+1} = B_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;     /* Coarse grid correction */
Update $\mathbf{x}_k = \mathbf{x}_k + P\mathbf{x}_{k+1} = M_k^{-1}\mathbf{r} + P_kB_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;
Solve for $M_k^T\mathbf{y} = \mathbf{r} - A_k\mathbf{x}_k$;                       /* Postsmooth */
Set $B_k^{-1}\mathbf{r} = \mathbf{x}_k + \mathbf{y}$.
```

Algorithm 6: Generic MG Algorithm

That is:

$$B_k^{-1} = \overline{M}_k^{-1} + (I - M_k^{-T}A_k)P_kB_{k+1}^{-1}P_k^T(I - A_kM_k^{-1}),$$

for $\overline{M}_k$ the symmetrized smoother and $B_\ell = A_\ell$.

Increasing the number of levels

```
At the coarsest level set $B_\ell = A_\ell$, the action of $B_k^{-1}\mathbf{r}$ is given by;
Solve for $M_k\mathbf{x}_k = \mathbf{r}$;                                        /* Presmooth */
Compute the residual $\mathbf{d} = \mathbf{r} - A_k\mathbf{x}_k = (I - A_kM_k^{-1})\mathbf{r}$;
Compute $\mathbf{x}_{k+1} = B_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;          /* Coarse grid correction */
Update $\mathbf{x}_k = \mathbf{x}_k + P\mathbf{x}_{k+1} = M_k^{-1}\mathbf{r} + P_kB_{k+1}^{-1}P_k^T(I - A_kM_k^{-1})\mathbf{r}$;
Solve for $M_k^T\mathbf{y} = \mathbf{r} - A_k\mathbf{x}_k$;                              /* Postsmooth */
Set $B_k^{-1}\mathbf{r} = \mathbf{x}_k + \mathbf{y}$.
```

Algorithm 7: Generic MG Algorithm

That is:

$$B_k^{-1} = \overline{M}_k^{-1} + (I - M_k^{-T}A_k)P_kB_{k+1}^{-1}P_k^T(I - A_kM_k^{-1}),$$

for $\overline{M}_k$ the symmetrized smoother and $B_\ell = A_\ell$.

Definition

We call this method the symmetric $V(1,1)$-cycle Multigrid.

Sufficient condition for convergence

Proposition

Under the assumption that the smoothers M_k are convergent in the A_k-norm, the symmetric $V(1,1)$-cycle Multigrid preconditioner B_k is such that $B_k - A_k$ is symmetric positive semidefinite.

Sufficient condition for convergence

Proposition

Under the assumption that the smoothers M_k are convergent in the A_k-norm, the symmetric $V(1,1)$-cycle Multigrid preconditioner B_k is such that $B_k - A_k$ is symmetric positive semidefinite.

😀 This means that B_k is a *convergent* stationary solver already under very non-restrictive hypothesis.

Sufficient condition for convergence

Proposition

Under the assumption that the smoothers M_k are convergent in the A_k-norm, the symmetric $V(1,1)$-cycle Multigrid preconditioner B_k is such that $B_k - A_k$ is symmetric positive semidefinite.

- 😀 This means that B_k is a *convergent* stationary solver already under very non-restrictive hypothesis.
- 😒 We still don't know anything about the convergence velocity of the method.

Sufficient condition for convergence

Proposition

Under the assumption that the smoothers M_k are convergent in the A_k-norm, the symmetric $V(1,1)$-cycle Multigrid preconditioner B_k is such that $B_k - A_k$ is symmetric positive semidefinite.

- 😀 This means that B_k is a *convergent* stationary solver already under very non-restrictive hypothesis.
- 😒 We still don't know anything about the convergence velocity of the method.
- 😒 We still don't know how it behaves when employed as preconditioner for the Conjugate Gradient algorithm. Can we discover under which hypothesis we get a *strong cluster* for the *eigenvalues*?

Sufficient condition for convergence

Proposition

Under the assumption that the smoothers M_k are convergent in the A_k-norm, the symmetric $V(1,1)$-cycle Multigrid preconditioner B_k is such that $B_k - A_k$ is symmetric positive semidefinite.

- 😀 This means that B_k is a *convergent* stationary solver already under very non-restrictive hypothesis.
- 😒 We still don't know anything about the convergence velocity of the method.
- 😒 We still don't know how it behaves when employed as preconditioner for the Conjugate Gradient algorithm. Can we discover under which hypothesis we get a *strong cluster* for the *eigenvalues*?

Idea

We have just rebuilt the construction without investigating the "high" and "low frequency" ideas. This will be our next target.

Stable decompositions

Let $\overline{V}_k = \mathrm{Range}(P_0, \ldots, P_{k-1})$ be the kth-level coarse space.

Stability

We say that a decomposition

$$\mathbf{v} = \sum_k \overline{\mathbf{v}}_k^f \qquad \overline{\mathbf{v}}_k^f \in \overline{V}_k,$$

is *stable* if there exists a level independent constant $\sigma > 0$ such that

$$\sum_k (\overline{\mathbf{v}}_k^f)^T A \overline{\mathbf{v}}_k \equiv \sum_k (\overline{\mathbf{v}}_k^f)^T A_k \overline{\mathbf{v}}_k^f \le \sigma \mathbf{v}^T A \mathbf{v}.$$

Stable decompositions

Let $\overline{V}_k = \mathrm{Range}(P_0, \dots, P_{k-1})$ be the kth-level coarse space.

Stability

We say that a decomposition

$$\mathbf{v} = \sum_k \overline{\mathbf{v}}_k^f \qquad \overline{\mathbf{v}}_k^f \in \overline{V}_k,$$

is *stable* if there exists a level independent constant $\sigma > 0$ such that

$$\sum_k (\overline{\mathbf{v}}_k^f)^T A \overline{\mathbf{v}}_k \equiv \sum_k (\overline{\mathbf{v}}_k^f)^T A_k \overline{\mathbf{v}}_k^f \le \sigma \mathbf{v}^T A \mathbf{v}.$$

Complementary space

For a space V_j we define the subspace $V_j^f \subset V_j$ complementary to the coarse space $P_j V_{j+1}$.

Choosing the complementary space

The idea from the simple Poisson case

We select V_j^f so that the symmetrized smoother $\overline{M}_j$ is efficient when restricted to V_j^f, i.e.,

$$\sum_k (\mathbf{v}_k^f)^T \overline{M}_k \mathbf{v}_k^f \leq \sigma \mathbf{v}^T A \mathbf{v}$$

Choosing the complementary space

The idea from the simple Poisson case

We select V_j^f so that the symmetrized smoother $\overline{M}_j$ is efficient when restricted to V_j^f, i.e.,

$$\sum_k (\mathbf{v}_k^f)^T \overline{M}_k \mathbf{v}_k^f \leq \sigma \mathbf{v}^T A \mathbf{v}$$

Vector decomposition

$$\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1} = [I, P_k] \begin{bmatrix} \mathbf{v}_k^f \\ \mathbf{v}_{k+1} \end{bmatrix} \text{ with } \mathbf{v}_j^f \in V_j^f \subset V_j,\ \mathbf{v}_{j+1} \in V_{j+1},\ j = k, k+1, \ldots, \ell - 1.$$

Choosing the complementary space

The idea from the simple Poisson case

We select V_j^f so that the symmetrized smoother $\overline{M}_j$ is efficient when restricted to V_j^f, i.e.,

$$\sum_k (\mathbf{v}_k^f)^T \overline{M}_k \mathbf{v}_k^f \leq \sigma \mathbf{v}^T A \mathbf{v}$$

Vector decomposition

$\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1} = [I, P_k] \begin{bmatrix} \mathbf{v}_k^f \\ \mathbf{v}_{k+1} \end{bmatrix}$ with $\mathbf{v}_j^f \in V_j^f \subset V_j$, $\mathbf{v}_{j+1} \in V_{j+1}$, $j = k, k+1, \ldots, \ell - 1$.

$$B_k^{-1} = [I, P_k] \overline{B}_k^{-1} [I, P_k]^T, \quad I = GG^T = (B_k^{1/2} [I, P_k] \overline{B}_k^{-1/2})(B_k^{1/2} [I, P_k] \overline{B}_k^{-1/2})^T,$$

Choosing the complementary space

The idea from the simple Poisson case

We select V_j^f so that the symmetrized smoother $\overline{M}_j$ is efficient when restricted to V_j^f, i.e.,

$$\sum_k (\mathbf{v}_k^f)^T \overline{M}_k \mathbf{v}_k^f \leq \sigma \mathbf{v}^T A \mathbf{v}$$

Vector decomposition

$$\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1} = [I, P_k] \begin{bmatrix} \mathbf{v}_k^f \\ \mathbf{v}_{k+1} \end{bmatrix} \text{ with } \mathbf{v}_j^f \in V_j^f \subset V_j,\ \mathbf{v}_{j+1} \in V_{j+1},\ j = k, k+1, \dots, \ell - 1.$$

$$B_k^{-1} = [I, P_k]\overline{B}_k^{-1}[I, P_k]^T, \quad I = GG^T = (B_k^{1/2}[I, P_k]\overline{B}_k^{-1/2})(B_k^{1/2}[I, P_k]\overline{B}_k^{-1/2})^T,$$

$$\Rightarrow \|G\|_2 < 1 \text{ and } \begin{bmatrix} \mathbf{v}_k^f \\ \mathbf{v}_{k+1} \end{bmatrix}^T [I, P_k] B_k [I, P_k]^T \begin{bmatrix} \mathbf{v}_k^f \\ \mathbf{v}_{k+1} \end{bmatrix} \leq \begin{bmatrix} \mathbf{v}_k^f \\ \mathbf{v}_{k+1} \end{bmatrix}^T \overline{B}_k \begin{bmatrix} \mathbf{v}_k^f \\ \mathbf{v}_{k+1} \end{bmatrix}.$$

Choosing the complementary space

By using the definition of $\overline{B}_k$ we can estimate

$$\begin{aligned}
0 \leq & \mathbf{v}_k^T (B_k - A_k)\mathbf{v}_k \\
\leq & \sum_{j=k}^{\ell-1} \left[\left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right)^T (M_j + M_j^T - A_j)^{-1} \left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right) \right] \\
& + \mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell - \mathbf{v}_k^T A_k \mathbf{v}_k .
\end{aligned}$$

If we select the decomposition for which $\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1}$

Choosing the complementary space

By using the definition of $\overline{B}_k$ we can estimate

$$\begin{aligned}
0 \leq & \mathbf{v}_k^T (B_k - A_k)\mathbf{v}_k \\
\leq & \sum_{j=k}^{\ell-1} \left[\left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right)^T (M_j + M_j^T - A_j)^{-1} \left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right) \right] \\
& + \mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell - \mathbf{v}_k^T A_k \mathbf{v}_k .
\end{aligned}$$

If we select the decomposition for which $\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1}$

1. $\sum_{j \geq k} (\mathbf{v}_j^f)^T \overline{M} \mathbf{v}_j^f \leq \sigma \mathbf{v}_k^T A_k \mathbf{v}_k,$

Choosing the complementary space

By using the definition of $\overline{B}_k$ we can estimate

$$\begin{aligned}
0 \leq & \mathbf{v}_k^T (B_k - A_k)\mathbf{v}_k \\
\leq & \sum_{j=k}^{\ell-1} \left[\left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right)^T (M_j + M_j^T - A_j)^{-1} \left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right) \right] \\
& + \mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell - \mathbf{v}_k^T A_k \mathbf{v}_k .
\end{aligned}$$

If we select the decomposition for which $\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1}$

1. $\sum_{j \geq k} (\mathbf{v}_j^f)^T \overline{M} \mathbf{v}_j^f \leq \sigma \mathbf{v}_k^T A_k \mathbf{v}_k,$
2. $\sum_{j \geq k} \mathbf{v}_{j+1}^T P_j^T A_j (M_j + M_j^T - A_j)^{-1} A_j P_j \mathbf{v}_{j+1} \leq \mu \mathbf{v}_k^T A_k \mathbf{v}_k^T,$

Choosing the complementary space

By using the definition of $\overline{B}_k$ we can estimate

$$\begin{aligned}0 \leq & \mathbf{v}_k^T (B_k - A_k)\mathbf{v}_k \\ \leq & \sum_{j=k}^{\ell-1} \left[\left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right)^T (M_j + M_j^T - A_j)^{-1} \left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right) \right] \\ & + \mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell - \mathbf{v}_k^T A_k \mathbf{v}_k .\end{aligned}$$

If we select the decomposition for which $\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1}$

1. $\sum_{j \geq k} (\mathbf{v}_j^f)^T \overline{M} \mathbf{v}_j^f \leq \sigma \mathbf{v}_k^T A_k \mathbf{v}_k$,
2. $\sum_{j \geq k} \mathbf{v}_{j+1}^T P_j^T A_j (M_j + M_j^T - A_j)^{-1} A_j P_j \mathbf{v}_{j+1} \leq \mu \mathbf{v}_k^T A_k \mathbf{v}_k^T$,
3. $\mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell \leq \sigma_c \mathbf{v}_k^T A_k \mathbf{v}_k$.

Choosing the complementary space

By using the definition of $\overline{B}_k$ we can estimate

$$
\begin{aligned}
0 \leq & \mathbf{v}_k^T (B_k - A_k)\mathbf{v}_k \\
\leq & \sum_{j=k}^{\ell-1} \left[\left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right)^T (M_j + M_j^T - A_j)^{-1} \left(M_j^T \mathbf{v}_j^f + A_j P_j \mathbf{v}_{j+1} \right) \right] \\
& + \mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell - \mathbf{v}_k^T A_k \mathbf{v}_k \leq (\sigma_c + 2(\sigma + \mu) - 1)\mathbf{v}_k^T A_k \mathbf{v}_k .
\end{aligned}
$$

If we select the decomposition for which $\mathbf{v}_j = \mathbf{v}_j^f + P_j \mathbf{v}_{j+1}$

1. $\sum_{j \geq k} (\mathbf{v}_j^f)^T \overline{M} \mathbf{v}_j^f \leq \sigma \mathbf{v}_k^T A_k \mathbf{v}_k,$
2. $\sum_{j \geq k} \mathbf{v}_{j+1}^T P_j^T A_j (M_j + M_j^T - A_j)^{-1} A_j P_j \mathbf{v}_{j+1} \leq \mu \mathbf{v}_k^T A_k \mathbf{v}_k^T,$
3. $\mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell \leq \sigma_c \mathbf{v}_k^T A_k \mathbf{v}_k.$

Theorem for the optimal choice

Theorem (Vassilevski 2008, Theorem 5.7)

Given A_j-convergent smoother M_j, $j = 0, \dots, \ell - 1$ for the $V(1,1)$-cycle MG preconditioner (with $B = B_0$ and $A = A_0$). If any fine-grid vector $\mathbf{v} = \mathbf{v}_0$ allows for a decomposition of the form $\mathbf{v}_j^f = \mathbf{v}_j - P_j\mathbf{v}_{j+1}$, $j = 0, 1, \dots, \ell - 1$, such that

- A1 **Stable decomposition:** $\sum_j (\mathbf{v}_j^f)^T \overline{M}_k \mathbf{v}_j^f \leq \sigma \mathbf{v}^T A\mathbf{v}$,
- A2 **Smoother scaling:** $(1+\delta)\mathbf{v}_j^T A\mathbf{v}_j \leq \mathbf{v}_j^T(M_j^T + M_j)\mathbf{v}_j = 2\mathbf{v}_j^T M_j \mathbf{v}_j$,
- A3 **Stable coarse component:** $\mathbf{v}_\ell^T A_\ell \mathbf{v}_\ell \leq \sigma_c \mathbf{v}^T A\mathbf{v}$,
- A4 **Efficiency of the smoothers** on the components of $A_j P_j \mathbf{v}_{j+1}$ so that

$$\sum_j \mathbf{v}_{j+1}^T P_j^T A_j (M_j + M_j^T - A_j)^{-1} A_j P_j \mathbf{v}_{j+1} \leq \mu \mathbf{v}^T A \mathbf{v}^T.$$

Then, the MG preconditioner B is uniformly spectrally equivalent to A:

$$\mathbf{v}^T A\mathbf{v} \leq \mathbf{v}^T B\mathbf{v} \leq (\sigma_c + 2(\sigma + \mu) - 1)\mathbf{v}^T A\mathbf{v}.$$

Where are we now?

1. We have now obtained an analogous ot first Convergence theorem we have seen that uses only the matrix properties.

Where are we now?

1. We have now obtained an analogous ot first Convergence theorem we have seen that uses only the matrix properties.
2. We now need to **compute** a **stable decomposition** for given a matrix A,

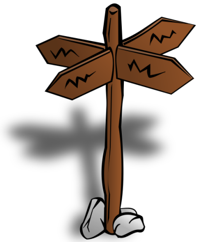
Where are we now?

1. We have now obtained an analogous ot first Convergence theorem we have seen that uses only the matrix properties.
2. We now need to **compute** a **stable decomposition** for given a matrix A,
3. Equivalently, finding matrices M_0 and a way of building P for which the assumptions A1-A4 hold.

Where are we now?

1. We have now obtained an analogous ot first Convergence theorem we have seen that uses only the matrix properties.
2. We now need to **compute** a **stable decomposition** for given a matrix A,
3. Equivalently, finding matrices M_0 and a way of building P for which the assumptions A1-A4 hold.

A comes from a FEM discretization of a PDE and we can use Sobolev space and grid properties to attain **stable decompositions**.



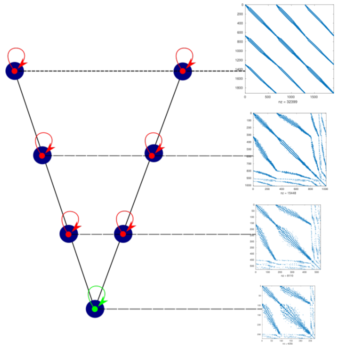
We forget about the source of A and try to build a *black-box* approach that **enforces the needed condition**.

We are at a crossroad

The algebraic idea

Given	Matrix $A \in \mathbb{R}^{n \times n}$ SPD
Wanted	Iterative method B to precondition the CG method:
Missing	Structural/geometric infos

- Hierarchy of systems
$$A_l \mathbf{x}_= \mathbf{b}_l, l = 0, \dots, \ell$$
- Transfer operators:
$$P^l_{l+1} : \mathbb{R}^{n_{l+1}} \rightarrow \mathbb{R}^{n_l}$$



Smoother: “High frequencies”

$$M_l : \mathbb{R}^{n_l} \rightarrow \mathbb{R}^{n_l}$$

Prolongator: “Low frequencies”

$$P_l : \mathbb{R}^{n_l} \rightarrow \mathbb{R}^{n_l+1}$$

The algebraic idea

Given	Matrix $A \in \mathbb{R}^{n\times n}$ SPD
Wanted	Iterative method B to precondition the CG method:

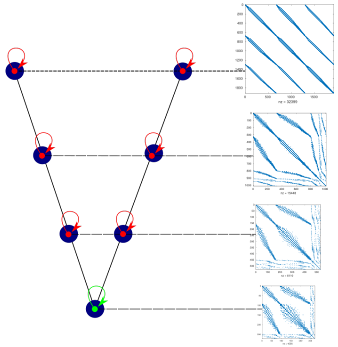
- Hierarchy of systems

$$A_l \mathbf{x}_= \mathbf{b}_l, l = 0, \dots, \ell$$

- Transfer operators:

$$P_{l+1}^{l} : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_l}$$

Missing Structural/geometric infos



Smoother: "High frequencies"

$$M_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$$

Prolongator: "Low frequencies"

$$P_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_{l+1}}$$

Complementarity of Smoother and Prolongator

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,

$\Rightarrow$ $\boldsymbol{\psi}_{i_c} = P\boldsymbol{\delta}_{i_c}$, $i_c = 1, \ldots, n_c$ is a basis for the range of $\pi_{\widetilde{M}}$: $\pi_{\widetilde{M}}\boldsymbol{\psi}_{i_c} = \delta_{i_c}$.

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,
- Decompose $\mathbf{v} = \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + P\mathbf{v}_c$, then

$$K_{TG} = \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I - \pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A\mathbf{v}} = \sup_{\mathbf{v}} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M}(I - \pi_{\widetilde{M}}) \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\mathbf{v}^T A\mathbf{v}}$$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,
- Decompose $\mathbf{v} = \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + P\mathbf{v}_c$, then

$$\begin{aligned} K_{TG} &= \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I - \pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A \mathbf{v}} = \sup_{\mathbf{v}} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M}(I - \pi_{\widetilde{M}}) \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\mathbf{v}^T A \mathbf{v}} \\ &= \sup_{\overline{\mathbf{v}}_f} \sup_{\mathbf{v}_c} \sup_{t \in \mathbb{R}} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M}(I - \pi_{\widetilde{M}}) \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\left(\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + tP\mathbf{v}_c \right)^T A \left(\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + tP\mathbf{v}_c \right)} \end{aligned}$$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,
- Decompose $\mathbf{v} = \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + P\mathbf{v}_c$, then

$$K_{TG} = \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I - \pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A \mathbf{v}} = \sup_{\mathbf{v}} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M}(I - \pi_{\widetilde{M}}) \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\mathbf{v}^T A \mathbf{v}}$$

$$= \sup_{\overline{\mathbf{v}}_f} \sup_{\mathbf{v}_c} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M}(I - \pi_{\widetilde{M}}) \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T A \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} - \left(\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T A P \mathbf{v}_c \right)^2 / \mathbf{v}_c^T P^T A P \mathbf{v}_c}$$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,
- Decompose $\mathbf{v} = \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + P\mathbf{v}_c$, then

$$K_{TG} = \sup_{\mathbf{v}} \frac{\mathbf{v}^T \widetilde{M}(I - \pi_{\widetilde{M}})\mathbf{v}}{\mathbf{v}^T A \mathbf{v}} = \sup_{\mathbf{v}} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M}(I - \pi_{\widetilde{M}}) \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\mathbf{v}^T A \mathbf{v}}$$

$$= \sup_{\overline{\mathbf{v}}_f} \sup_{\mathbf{v}_c} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M}(I - \pi_{\widetilde{M}}) \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T A \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} - \left(\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T A P \mathbf{v}_c \right)^2 / \mathbf{v}_c^T P^T A P \mathbf{v}_c}$$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,
- Decompose $\mathbf{v} = \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + P\mathbf{v}_c$, then

$$K_{TG} \leq \sup_{\overline{\mathbf{v}}_f} \sup_{\mathbf{v}_c} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M} \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T A \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} - \left(\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T AP\mathbf{v}_c \right)^2 \Big/ \mathbf{v}_c^T P^T AP\mathbf{v}_c}$$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and denote by $\mathbb{R}^{n_c}$ the *coarse space* we are targeting.

- Let $\{\boldsymbol{\delta}_{i_c}\}$ be the basis of unit coordinate vectors in $\mathbb{R}^{n_c}$,
- Decompose $\mathbf{v} = \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} + P\mathbf{v}_c$, then

$$K_{TG} \leq \sup_{\overline{\mathbf{v}}_f} \sup_{\mathbf{v}_c} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M} \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T A \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix} - \left(\underbrace{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T AP\mathbf{v}_c}_{=0 \quad \forall\ \mathbf{v}_c \text{ and } \overline{\mathbf{v}}_f} \right)^2 \Big/ \mathbf{v}_c^T P^T AP\mathbf{v}_c}$$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and to optimize the bound on K_{TG} we want

$$\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T AP\mathbf{v}_c = 0 \quad \forall\ \mathbf{v}_c \text{ and } \overline{\mathbf{v}}_f$$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix}$$

and to optimize the bound on K_{TG} we want

$$\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} W \\ I \end{bmatrix} \mathbf{v}_c = 0 \quad \forall\, \mathbf{v}_c \text{ and } \overline{\mathbf{v}}_f$$

$\Rightarrow$ Select W such that $A_{ff} W + A_{fc} = 0$

Optimal prolongation

Let us assume that P has the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix} = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} \qquad A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}$$

and to optimize the bound on K_{TG} we want

$$\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} W \\ I \end{bmatrix} \mathbf{v}_c = 0 \quad \forall \, \mathbf{v}_c \text{ and } \overline{\mathbf{v}}_f$$

Since it gives us

$$K_{TG} \leq \sup_{\overline{\mathbf{v}}_f} \frac{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T \widetilde{M} \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}}{\begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}^T A \begin{bmatrix} \overline{\mathbf{v}}_f \\ \mathbf{0} \end{bmatrix}} = \sup_{\overline{\mathbf{v}}_f} \frac{\overline{\mathbf{v}}_f^T \widetilde{M}_{ff} \overline{\mathbf{v}}_f}{\overline{\mathbf{v}}_f^T A_{ff}} = \frac{1}{\lambda_{\min}(\widetilde{M}_{ff}^{-1} A_{ff})}.$$

Selecting c and f nodes

The second take-home message

A *reasonable* guideline to construct P is to find for any coarse unit vector $\boldsymbol{\delta}_{i_c}$ in $\mathbb{R}^{n_c}$, an **approximate solution** to

$$A_{ff}\mathbf{w}_{i_c} = -A_{fc}\boldsymbol{\delta}_{i_c}$$

and build the i_cth column of P as $\psi_{i_c} = [\mathbf{w}_{i_c}^T, \boldsymbol{\delta}_{i_c}^T]^T$.

- We cannot solve exactly the systems for the $\mathbf{w}$ both for sparsity and cost reasons,
- We are now left with the problem of finding the *coarse* nodes.

Selecting c and f nodes

The second take-home message

A *reasonable* guideline to construct P is to find for any coarse unit vector $\boldsymbol{\delta}_{i_c}$ in $\mathbb{R}^{n_c}$, an **approximate solution** to

$$A_{ff}\mathbf{w}_{i_c} = -A_{fc}\boldsymbol{\delta}_{i_c}$$

and build the i_cth column of P as $\psi_{i_c} = [\mathbf{w}_{i_c}^T, \boldsymbol{\delta}_{i_c}^T]^T$.

- We cannot solve exactly the systems for the $\mathbf{w}$ both for sparsity and cost reasons,
- We are now left with the problem of finding the *coarse* nodes.

Here begins the fun

What differentiates the available AMG algorithms is the procedure for identifying the coarse space through a combination of *heuristics*, *brute force* and *clever guesses*.

Ruge-Stuben Splitting Algorithm

Assumption:

Geometrically smooth functions are in the near null space of A.

Wlog assume A s.t. $\lambda_{\max}(A) = 1$, and let $\mathbf{e}$ be a small normalized eigenmode of A, i.e.,

$$A\mathbf{e} = \lambda \mathbf{e},\ \|\mathbf{e}\| = 1,\ \lambda \ll 1$$

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

Ruge-Stuben Splitting Algorithm

Assumption:

Geometrically smooth functions are in the near null space of A.

Wlog assume A s.t. $\lambda_{\text{max}}(A) = 1$, and let $\mathbf{e}$ be a small normalized eigenmode of A, i.e.,

$$A\mathbf{e} = \lambda \mathbf{e}, \ \|\mathbf{e}\| = 1, \ \lambda \ll 1$$

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

Heuristic

Smooth error varies slowly in the direction of relatively large (negative) coefficients of the matrix.

Ruge-Stuben Splitting Algorithm

Strong dependence (Ruge and Stüben 1987)

For a chosen tolerance $\theta \in (0,1]$, we say that a dof i is strongly influenced by $j \neq i$ if

$$-a_{i,j} \geq \max_{k\neq i}(-a_{k,i}).$$

Define:

- $W_i = \{j \in \Omega_i \ : \ j \text{ is weakly connected to } j\}$,
- $S_i = \{j \in \Omega_i \ : \ j \text{ is strongly connected to } j\}$,
- C_i set of coarse points that are allowed to interpolate i.

The (i, i_c) entry of P is given by

$$-\frac{\left(a_{i,i_c} + \sum_{i_\chi \in S_i} a_{i,i_\chi} \frac{a_{i_\chi,i_c}}{\sum_{j_c \in C_i} a_{i_\chi,j_c}} \right)}{a_{ii} + \sum_{i_\chi \in W_i} a_{i,i_\chi}}$$

Ruge-Stuben Splitting Algorithm

Strong dependence (Ruge and Stüben 1987)

For a chosen tolerance $\theta \in (0,1]$, we say that a dof i is strongly influenced by $j \neq i$ if

$$-a_{i,j} \geq \max_{k \neq i}(-a_{k,i}).$$

Define:

- $W_i = \{j \in \Omega_i \ : \ j \text{ is weakly connected to } j\}$,
- $S_i = \{j \in \Omega_i \ : \ j \text{ is strongly connected to } j\}$,
- C_i set of coarse points that are allowed to interpolate i.

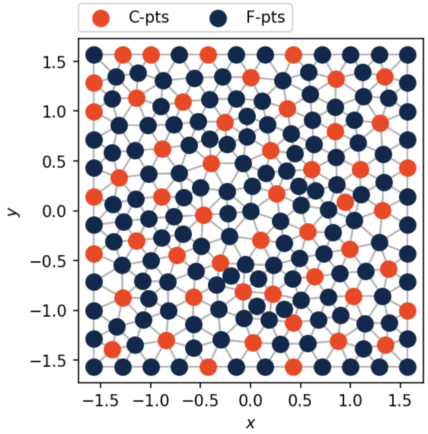
Without delving into the details, the expression for the interpolation can be obtained by

1. Defining a *strength matrix*, A_s, obtained deleting the weak connections in A,
2. **First pass** choosing an *independent set* of fine grid points based on the graph of A_s,
3. **Second pass** choosing additional points (if needed) to satisfy interpolation requirements.

Ruge-Stuben Splitting Algorithm

To see the algorithm at work, we test it by means of the PyAMG library (Bell, Olson, and Schroder 2022) on a small problem, specifically we use it to highlight the division in Coarse and Fine dofs of a given grid.

You can run the example in Google Colab by using the GitHub Gist https://bit.ly/3MToLtN.



Coarsening via aggregation

Aggregation idea

The aggregation idea is using an algorithm that splits the *set of vertices* of the graph of A or of a re-weighted version of A (sometimes called filtered matrix) as a *union of non-overlapping subsets* – **aggregates** – each of which forms a connected sub-graph.

Coarsening via aggregation

Aggregation idea

The aggregation idea is using an algorithm that splits the *set of vertices* of the graph of A or of a re-weighted version of A (sometimes called filtered matrix) as a *union of non-overlapping subsets* – **aggregates** – each of which forms a connected sub-graph.

$$\{1, \ldots, n\} = \bigcup_{j=1}^{J} \mathcal{A}_j, \quad \mathcal{A}_i \bigcap \mathcal{A}_j = \emptyset, i \neq j,$$

Coarsening via aggregation

Aggregation idea

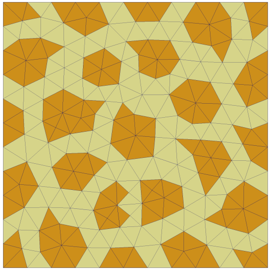
The aggregation idea is using an algorithm that splits the *set of vertices* of the graph of A or of a re-weighted version of A (sometimes called filtered matrix) as a *union of non-overlapping subsets* – **aggregates** – each of which forms a connected sub-graph.

$$\{1,\ldots,n\} = \bigcup_{j=1}^{J} \mathcal{A}_j, \quad \mathcal{A}_i \bigcap \mathcal{A}_j = \emptyset, i \neq j,$$

The FEM case

For a FEM discretization of PDE on a set Ω this corresponds to a partition of the domain:

$$\Omega = \bigcup_{j=1}^{J} \Omega_j, \quad \Omega_i \bigcap \Omega_j = \emptyset, i \neq j$$



-G https://bit.ly/3vnAV82

Coarsening via aggregation

Having selected **aggregates**

$$\{1,\ldots,n\} = \bigcup_{j=1}^{J} \mathcal{A}_j, \quad \mathcal{A}_i \bigcap \mathcal{A}_j = \emptyset, i \neq j,$$

The prolongator operator is then given simply by posing

$$P : \mathbb{R}^{n_c} \to \mathbb{R}^n, \quad (P\mathbf{x})_i \mapsto x_j, \quad i \in \mathcal{A}_j.$$

- Since the aggregates are mutually disjoint $\forall\, i \in \{1,\ldots,n\}$ exist only one index $j \in \{1,\ldots,n_c\}$ such that $i \in \mathcal{C}_j$.
- “the jth component of the vector $\mathbf{x} \in \mathbb{R}^m$, $m = |\mathcal{A}_j|$ will be mapped onto all components of the vector $\mathbf{y} \in \mathbb{R}^n$ indices of which are in $\mathcal{A}_j$”
- P represents a *piece-wise constant* interpolation operator.

Coarsening via aggregation

Usually *piece-wise constant* is not enough...

```
A = pyamg.gallery.poisson((500, 500), format='csr')
b = np.ones((A.shape[0],1))
standalone_residuals = []
mls = pyamg.smoothed_aggregation_solver(A,
↪   symmetry='hermitian', smooth=None)
standalone_residuals = []
x = mls.solve(b, tol=1e-10, accel=None,
↪   residuals=standalone_residuals)
```

Coarsening via aggregation

Usually *piece-wise constant* is not enough...

```
A = pyamg.gallery.poisson((500, 500), format='csr')
b = np.ones((A.shape[0],1))
standalone_residuals = []
mls = pyamg.smoothed_aggregation_solver(A,
↪  symmetry='hermitian', smooth=None)
standalone_residuals = []
x = mls.solve(b, tol=1e-10, accel=None,
↪  residuals=standalone_residuals)
```

- We get this Multigrid Hierarchy (that seems plausible)

```
MultilevelSolver
Number of Levels:     7
Operator Complexity:  1.262
Grid Complexity:      1.188
Coarse Solver:        'pinv'
level   unknowns     nonzeros
0         250000      1248000 [79.24%]
1          41750       290584 [18.45%]
2           4704        32370 [2.06%]
3            532         3538 [0.22%]
4             70          424 [0.03%]
5             12           58 [0.00%]
6              3            9 [0.00%]
```

Coarsening via aggregation

Usually *piece-wise constant* is not enough...

```
A = pyamg.gallery.poisson((500, 500), format='csr')
b = np.ones((A.shape[0],1))
standalone_residuals = []
mls = pyamg.smoothed_aggregation_solver(A,
↪  symmetry='hermitian', smooth=None)
standalone_residuals = []
x = mls.solve(b, tol=1e-10, accel=None,
↪  residuals=standalone_residuals)
```

- We get this Multigrid Hierarchy (that seems plausible)
- But after **101 iterates** we have a residual of 1.8e+02, so we are not converging.

```
MultilevelSolver
Number of Levels:     7
Operator Complexity:  1.262
Grid Complexity:      1.188
Coarse Solver:        'pinv'
level   unknowns     nonzeros
0         250000      1248000 [79.24%]
1          41750       290584 [18.45%]
2           4704        32370 [2.06%]
3            532         3538 [0.22%]
4             70          424 [0.03%]
5             12           58 [0.00%]
6              3            9 [0.00%]
```

Coarsening via aggregation

Usually *piece-wise constant* is not enough...

```
A = pyamg.gallery.poisson((500, 500), format='csr')
b = np.ones((A.shape[0],1))
standalone_residuals = []
mls = pyamg.smoothed_aggregation_solver(A,
↪   symmetry='hermitian', smooth=None)
standalone_residuals = []
x = mls.solve(b, tol=1e-10, accel=None,
↪   residuals=standalone_residuals)
```

- We get this Multigrid Hierarchy (that seems plausible)
- But after **101 iterates** we have a residual of `1.8e+02`, so we are not converging.

```
MultilevelSolver
Number of Levels:     7
Operator Complexity:  1.262
Grid Complexity:      1.188
Coarse Solver:        'pinv'
level   unknowns     nonzeros
0         250000      1248000 [79.24%]
1          41750       290584 [18.45%]
2           4704        32370 [2.06%]
3            532         3538 [0.22%]
4             70          424 [0.03%]
5             12           58 [0.00%]
6              3            9 [0.00%]
```

How can we make things better?

Smoothed aggregation (Vaněk, Mandel, and Brezina 1996)

- We could play around to get *better aggregates* for getting a convergence constant,

Smoothed aggregation (Vaněk, Mandel, and Brezina 1996)

- We could play around to get *better aggregates* for getting a convergence constant,
- Select the weighting in P so that one or more vectors of the *near* Kernel are preserved (usually very useful for *elasticity problems*),

Smoothed aggregation (Vaněk, Mandel, and Brezina 1996)

- We could play around to get *better aggregates* for getting a convergence constant,
- Select the weighting in P so that one or more vectors of the *near* Kernel are preserved (usually very useful for *elasticity problems*),
- Use a procedure to **smooth out** the *basis function* induced by the aggregation procedure by using the smoother, which is, using few applications of smoothing on the prolongation matrix:

$$P_S = (I - M^{-1}A)^{\nu} P, \quad \text{for some } \nu \geq 1.$$

Smoothed aggregation (Vaněk, Mandel, and Brezina 1996)

- We could play around to get *better aggregates* for getting a convergence constant,
- Select the weighting in P so that one or more vectors of the *near* Kernel are preserved (usually very useful for *elasticity problems*),
- Use a procedure to **smooth out** the *basis function* induced by the aggregation procedure by using the smoother, which is, using few applications of smoothing on the prolongation matrix:

$$P_S = (I - M^{-1}A)^{\nu} P, \quad \text{for some } \nu \geq 1.$$

```
mls = pyamg.smoothed_aggregation_solver(A,symmetry='hermitian',
↪  smooth=('jacobi',{'omega':4/3}))
print(mls)
standalone_residuals_jacobi = []
x = mls.solve(b, tol=1e-10, accel=None, residuals=standalone_residuals_jacobi)
```

That run for 12 iteration with last residual 8.620525e-09

Smoothed aggregation (Vaněk, Mandel, and Brezina 1996)

Hierarchy for unsmoothed aggregation

```
MultilevelSolver
Number of Levels:     7
Operator Complexity:  1.262
Grid Complexity:      1.188
Coarse Solver:        'pinv'
level   unknowns     nonzeros
0        250000      1248000 [79.24%]
1         41750       290584 [18.45%]
2          4704        32370 [2.06%]
3           532         3538 [0.22%]
4            70          424 [0.03%]
5            12           58 [0.00%]
6             3            9 [0.00%]
```

Hierarchy for smoothed aggregation

```
MultilevelSolver
Number of Levels:     6
Operator Complexity:  1.337
Grid Complexity:      1.188
Coarse Solver:        'pinv'
level   unknowns     nonzeros
0        250000      1248000 [74.82%]
1         41750       373416 [22.39%]
2          4704        41554 [2.49%]
3           532         4526 [0.27%]
4            65          509 [0.03%]
5             9           65 [0.00%]
```

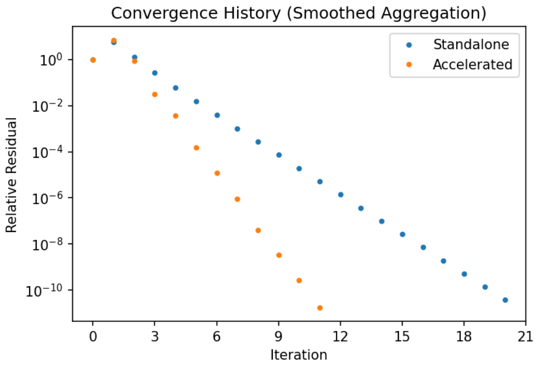
Smoothed aggregation (Vaněk, Mandel, and Brezina 1996)

- *Smoothed aggregation* produces hierarchies with more nonzero entries,
- To reduce the *fill-in* filtering (dropping) strategies are usually implemented,

Operator Complexity

$$\mathrm{opc} = \frac{\sum_{l=0}^{\ell-1} \mathrm{nnz}(A_l)}{\mathrm{nnz}(A_0)}$$

- AMG is more often used as preconditioner for the CG algorithm that as solver.



You can run this example on Google Colab from https://bit.ly/3OHYKPJ.

Compatible relaxation (Brandt 2000)

First comes the smoother

The approach called **compatible relaxation** consists in selecting a set of coarse degrees of freedom **based solely** on the **smoother**, the *interpolation matrix* is constructed later.

Compatible relaxation (Brandt 2000)

First comes the smoother

The approach called **compatible relaxation** consists in selecting a set of coarse degrees of freedom **based solely** on the **smoother**, the *interpolation matrix* is constructed later.

From the definition of K_{TG} we have that if we find a matrix J_* such that

J1 $\mathrm{Range}(J_*) = \mathrm{Range}(I - PR_*)$, $R_* = (P^T \widetilde{M} P)^{-1} P^T \widetilde{M}$

we have the inequality: $\mathbf{v} J_*^T \widetilde{M} J_* \mathbf{v} \leq K_{TG} \mathbf{v}^T J^T A J \mathbf{v}$

Compatible relaxation (Brandt 2000)

First comes the smoother

The approach called **compatible relaxation** consists in selecting a set of coarse degrees of freedom **based solely** on the **smoother**, the *interpolation matrix* is constructed later.

From the definition of K_{TG} we have that if we find a matrix J_* such that

J1 $\mathrm{Range}(J_*) = \mathrm{Range}(I - PR_*)$, $R_* = (P^T \widetilde{M} P)^{-1} P^T \widetilde{M}$

we have the inequality: $\mathbf{v} J_*^T \widetilde{M} J_* \mathbf{v} \leq K_{TG} \mathbf{v}^T J^T A J \mathbf{v}$

Idea

J picks a principal submatrix from A and $\widetilde{M}$, the inequality thus means that A has a principal submatrix that is spectrally equivalent to the same principal submatrix of $\widetilde{M}$.

Compatible relaxation (Brandt 2000)

First comes the smoother

The approach called **compatible relaxation** consists in selecting a set of coarse degrees of freedom **based solely** on the **smoother**, the *interpolation matrix* is constructed later.

From the definition of K_{TG} we have that if we find a matrix J_* such that

J1 $\mathrm{Range}(J_*) = \mathrm{Range}(I - PR_*)$, $R_* = (P^T \widetilde{M} P)^{-1} P^T \widetilde{M}$

we have the inequality: $\mathbf{v} J_*^T \widetilde{M} J_* \mathbf{v} \leq K_{TG} \mathbf{v}^T J^T A J \mathbf{v}$

Heuristic

Fix R, then select J such that the constant K_{CR} in

$$\mathbf{v}^T J A J \mathbf{v} \leq \mathbf{v}^T J^T \widetilde{M} J \mathbf{v} \leq K_{CR} \mathbf{v}^T J^T A J \mathbf{v},$$

is close to 1.

Compatible relaxation the idea

Let A be ans SPD matrix, M an A-convergent smoother, one can prove that

$$\|(I - \widetilde{M}^{-1}A)^m \mathbf{e}\|_A \leq \frac{1}{\sqrt{m+1}} \|\mathbf{e}\|_{\widetilde{M}} \qquad \text{(Smoothing property)}$$

Take a **projection on the coarse space** Q being $\widetilde{M}$-orthogonal satisfying

$$\|(I - Q)\mathbf{e}\|_{\widetilde{M}} \leq \delta \|\mathbf{e}\|_A \qquad \text{(Approximation property)}$$

Then for any $\mathbf{e} = (I - Q)\mathbf{e}$ and any integer $m \geq 1$ the following estimate holds

$$\|(I - \widetilde{M}^{-1}A)^m \mathbf{e}\|_{\widetilde{M}} \leq \frac{\delta}{\sqrt{1+m}} \|\mathbf{e}\|_{\widetilde{M}}.$$

Compatible relaxation the algorithm

We apply our inequality for a solution of the homogeneous system $A\mathbf{x} = \mathbf{0}$

```
Input: $\mathbf{e}$ random initial iterate
$m = 1$;
Compute $\mathbf{e}_0 = (I - Q)\mathbf{e}$;
Smooth $\mathbf{e}_m = (I - \widetilde{M}^{-1}A)\mathbf{e} = (I - M^{-1}A)(I - M^{-T}A)\mathbf{e}_{m-1}$;
if $\|\mathbf{e}_m\|_{\widetilde{M}} / \|\mathbf{e}_0\|_{\widetilde{M}}$ is small then
    The process has converged, convergence is now fast.;
else
    Use $\mathbf{e}_m$ to augment the coarse space, build a new $Q$ and try again.
end
```

Since

$$\|(I - \widetilde{M}^{-1}A)^m \mathbf{e}\|_{\widetilde{M}} \leq \frac{\delta}{\sqrt{1+m}} \|\mathbf{e}\|_{\widetilde{M}},$$

an m large enough for which this procedure converge exists.

Compatible relaxation the algorithm

We apply our inequality for a solution of the homogeneous system $A\mathbf{x} = \mathbf{0}$

```
Input: $\mathbf{e}$ random initial iterate
$m = 1$;
Compute $\mathbf{e}_0 = (I - Q)\mathbf{e}$;
Smooth $\mathbf{e}_m = (I - \widetilde{M}^{-1}A)\mathbf{e} = (I - M^{-1}A)(I - M^{-T}A)\mathbf{e}_{m-1}$;
if $\|\mathbf{e}_m\|_{\widetilde{M}}/\|\mathbf{e}_0\|_{\widetilde{M}}$ is small then
    The process has converged, convergence is now fast.;
else
    Use $\mathbf{e}_m$ to augment the coarse space, build a new $Q$ and try again.
end
```

Since

$$\|(I - \widetilde{M}^{-1}A)^m\mathbf{e}\|_{\widetilde{M}} \leq \frac{\delta}{\sqrt{1+m}}\|\mathbf{e}\|_{\widetilde{M}},$$

an m large enough for which this procedure converge exists.

A list of available libraries

As we have discussed **implementing these methods efficiently** requires **some thought**.

A list of available libraries

The good news is that there are several libraries that one can resort to.

hypre
: is a library of high performance preconditioners and solvers featuring multigrid methods for the solution of large, sparse linear systems of equations on massively parallel computers. https://github.com/hypre-space/hypre

A list of available libraries

The good news is that there are several libraries that one can resort to.

hypre
: is a library of high performance preconditioners and solvers featuring multigrid methods for the solution of large, sparse linear systems of equations on massively parallel computers. https://github.com/hypre-space/hypre

ML - Trilinos
: The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. https://github.com/trilinos/Trilinos

A list of available libraries

The good news is that there are several libraries that one can resort to.

PETSc the Portable, Extensible Toolkit for Scientific Computation is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.
https://petsc.org/release/

A list of available libraries

The good news is that there are several libraries that one can resort to.

PETSc
: the Portable, Extensible Toolkit for Scientific Computation is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.
https://petsc.org/release/

AGMG
: solves systems of linear equations with an aggregation-based algebraic multigrid method. For this library, several level of parallelism are provided: multi-threading (multi-core acceleration of sequential programs), MPI-based, or hybrid mode (MPI+multi-threading). http://agmg.eu/

A list of available libraries

The good news is that there are several libraries that one can resort to.

PETSc the Portable, Extensible Toolkit for Scientific Computation is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.
https://petsc.org/release/

AGMG solves systems of linear equations with an aggregation-based algebraic multigrid method. For this library, several level of parallelism are provided: multi-threading (multi-core acceleration of sequential programs), MPI-based, or hybrid mode (MPI+multi-threading). http://agmg.eu/

PSCToolkit parallel BLAS feature for sparse matrices that are capable of running on machines with thousands of high-performance cores, and construction of higher-level iterative solvers and preconditioners.
https://psctoolkit.github.io/

A list of available libraries

The good news is that there are several libraries that one can resort to.

PETSc	the Portable, Extensible Toolkit for Scientific Computation is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. https://petsc.org/release/
AGMG	solves systems of linear equations with an aggregation-based algebraic multigrid method. For this library, several level of parallelism are provided: multi-threading (multi-core acceleration of sequential programs), MPI-based, or hybrid mode (MPI+multi-threading). http://agmg.eu/
PSCToolkit	parallel BLAS feature for sparse matrices that are capable of running on machines with thousands of high-performance cores, and construction of higher-level iterative solvers and preconditioners. https://psctoolkit.github.io/

Why all this interest in *large* and *parallel*?

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$,

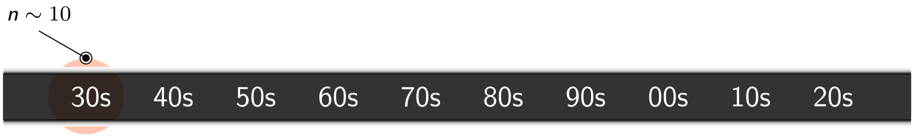
But what does large mean?

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



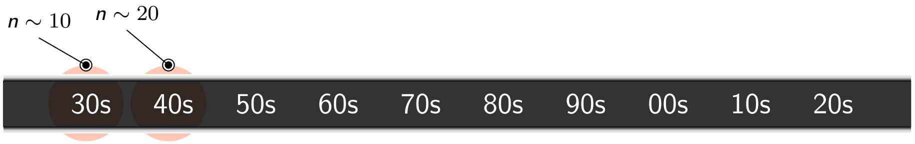
"In a ground wire problem involving a large number of ground conductors, 13 simultaneous equations were solved..." – Dwight (1930)"
"The second machine, now in operation, was designed for the direct solution of nine or fewer simultaneous equations." – Wilbur, J. B. (1936)

How large is large?

$$\text{Solve}: \ A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



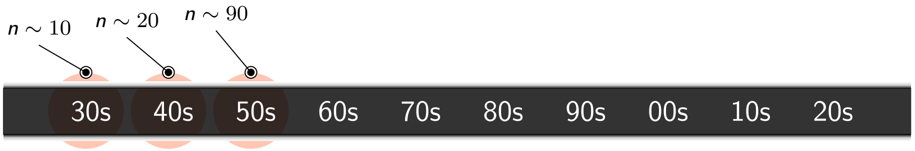
“Finally, though the labour of relaxation in three dimensions is prohibitively great, the future use of the new electronic calculating machines in this connexion is a distinct possibility” – Fox, L. (1947)

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n \times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



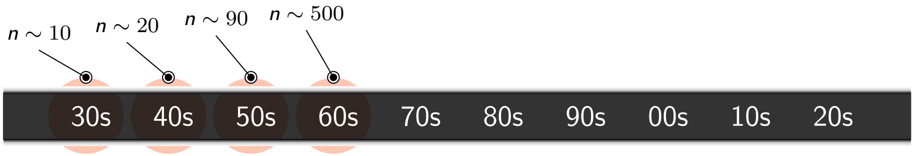
“The Ferranti PEGASUS computer, with a main store of 4096 words, can solve a maximum of 86 simultaneous equations by its standard subroutine and takes about 45 minutes to complete this calculation.”
– Wilson, L. B. (1959)

How large is large?

$$\text{Solve : } A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



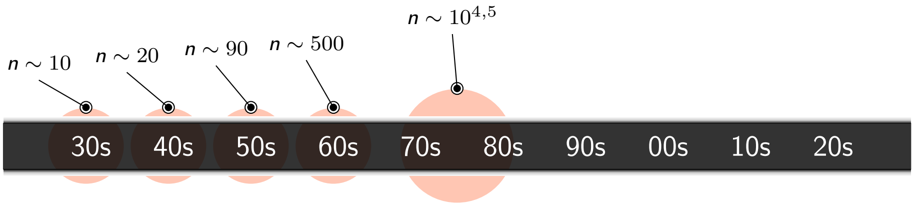
"...the bound imposed by this is $m + n \leq 474$. In addition, this number of equations would fill one standard (1.800ft) reel of magnetic tape, and the fifty-odd hours taken in the calculation might be thought excessive."
– Barron, Swinnerton-Dyer (1960)

How large is large?

$$\text{Solve}: A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



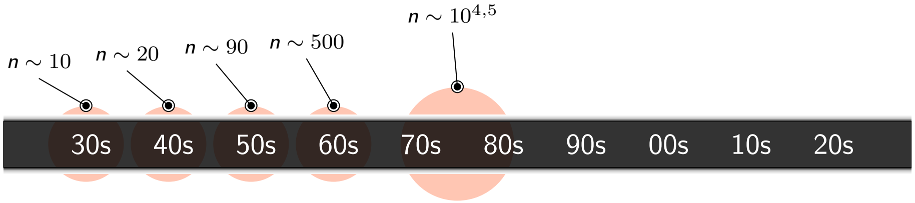
"...handling problems involving sets of simultaneous equations of two-thousandth order, and SAMIS available through "Cosmic" at the University of Georgia, which can treat up to 10,000 simultaneous equations." – Melosh, Schmele (1969)

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



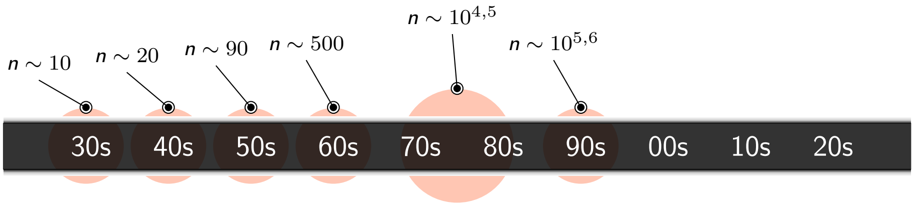
"The mini-computer cost algorithm is applied to the same complex shell problem used previously, with 9120 degrees of freedom [...]. The running times, however, are 40 and 70 hr, respectively! It would appear that improvement of mini-computer speeds is required..." – Kamel, McCabe (1978)

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



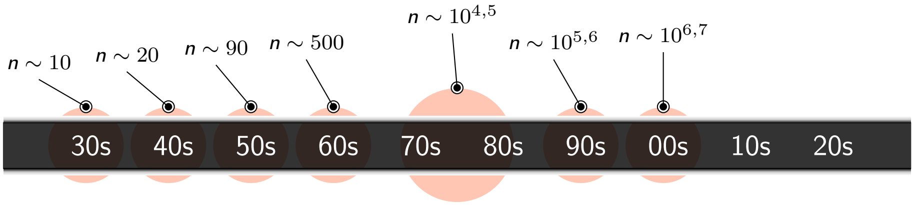
"For instance, Pomerell in 1994 reports on successful application of preconditioned Krylov methods for very ill-conditioned unstructured finite element systems of order up to 210,000 that arise in semiconductor device modeling." – Saad Y., van der Vorst, H.A. (2000)

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



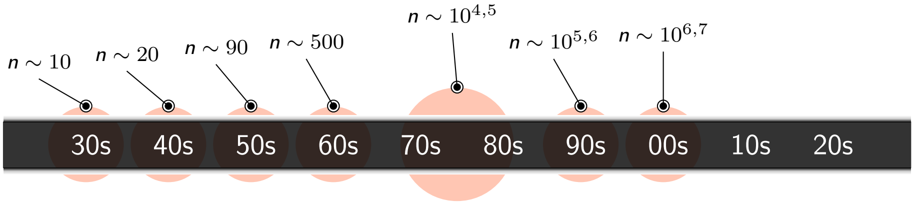
"As a second example, we show results (Table VIII) for a problem arising in ocean modeling (barotropic equation) with $n = 370,000$ unknowns and approximately 3.3 million nonzero entries..." – Benzi, M. (2002)

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



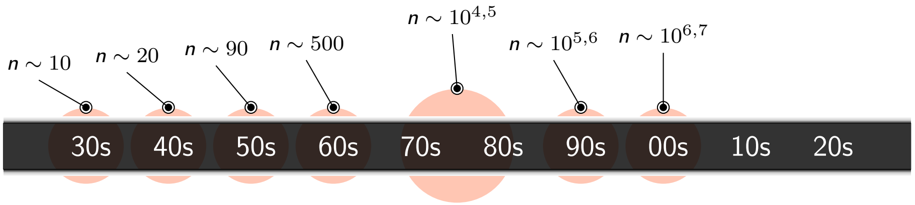
"Problem: Large, mesh size: $180 \times 60 \times 30$, $\sharp$ unknowns (in simulation): 1,010,160, Solution time 45.7 h" – Wang, de Sturler, Paulino (2006)

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



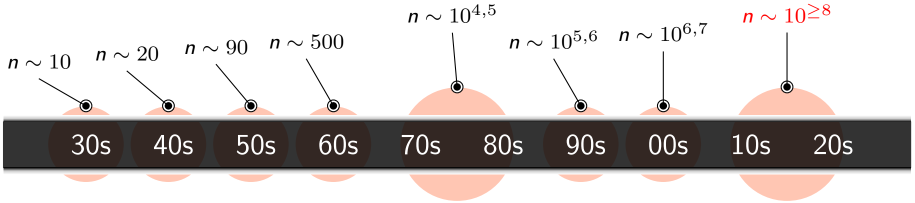
"The parallel GMRES was tested on the Tesla T10P GPU using a set of matrix data from the oil field simulation data of Conoco Phillips. The order of the system ranges from ~ 2000 to ~ 1.1 million." – M. Wang, H. Klie, M. Parashar, H. Sudan (2009)

How large is large?

$$\text{Solve}: \; A\mathbf{x} = \mathbf{b},$$

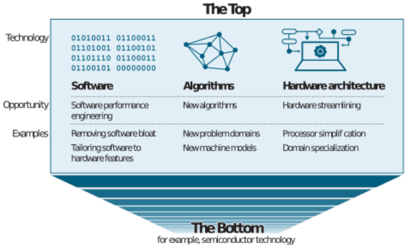
where

- $A \in \mathbb{R}^{n\times n}$ is a very large and sparse matrix $\mathrm{nnz}(A) = O(n)$,
- $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.



The exascale challenge, using computer that do 10^{15} Flops, targeting next-gen systems doing 10^{18} Flops to solve problems with tens of billions of unknowns.

The philosophy behind the effort



(Leiserson et al. 2020)

"As miniaturization wanes, the silicon-fabrication improvements at the Bottom will no longer provide the predictable, broad-based gains in computer performance that society has enjoyed for more than 50 years. Software performance engineering, development of algorithms, and hardware streamlining at the Top can continue to make computer applications faster in the post-Moore era."

Where we want to solve it[1]

	System	Cores	Rmax (TFlops/s)
1	Fugaku	7,630,848	442,010.0
2	Summit	2,414,592	148,600.0
3	Sierra	1,572,480	94,640.0
⋮	⋮	⋮	⋮
18	Marconi-100	347,776	21,640.0
⋮	⋮	⋮	⋮
20	Piz Daint	387,872	21,230.0
⋮	⋮	⋮	⋮
74	MareNostrum	153,216	6,470.8

Marconi-100 - CINECA

Piz Daint - CSCS

- Machines with hundreds of MPI cores,
- Hybrid form of parallelism: MPI, OpenMP, CUDA/OpenCL, ...

[1]TOP500 list, November 2021 – https://www.top500.org

What do we ask to it?

Solve the preconditioned system:

$$B^{-1}Ax = B^{-1}b,$$

with matrix $B^{-1} \approx A^{-1}$ (left preconditioner) such that:

Algorithmic scalability $\max_i \lambda_i(B^{-1}A) \approx 1$ being independent of n (all the work we did on the K constant!),

Linear complexity the action of B^{-1} costs as little as possible, the best being $\mathcal{O}(n)$ flops,

Implementation scalability in a massively parallel computer, B^{-1} should be composed of local actions, performance should depend linearly on the number of processors employed.

PSCToolkit – psctoolkit.github.io

Two central libraries PSBLAS and AMG4PSBLAS:

- Existing software standards:
    - MPI, OpenMP, CUDA
    - Serial sparse BLAS,
    - (Par)Metis,
    - AMD
- Attention to performance using modern Fortran;
- Research on new preconditioners;
- No need to delve in the data structures for the user;
- Tools for error and mesh handling beyond simple algebraic operations;
- Standard Krylov solvers

PSCToolkit – psctoolkit.github.io

Two central libraries PSBLAS and AMG4PSBLAS:

- Domain decomposition preconditioners
- Algebraic multigrid with aggregation schemes
    - Parallel coupled weighted matching based aggregation
    - Parallel decoupled smoothed aggregation (Vaněk, Mandel, Brezina)
- Parallel Smoothers (Block-Jacobi, DD-Schwartz, Hybrid-GS/SGS/FBGS, ℓ_1 variants) that can be coupled with specialized block (approximate) solvers MUMPS, SuperLU, incomplete factorizations ((H)AINV, (H)INVK/L, (H)ILU-type)
- V-Cycle, W-Cycle, K-Cycle

PSCToolkit – `psctoolkit.github.io`

Two central libraries PSBLAS and AMG4PSBLAS.

- Freely available from: `https://psctoolkit.github.io`,
- Open Source with BSD 3 Clause License.

People involved: S. Filippone, P. D'Ambra, F. Durastante.

Contributors: Soren Rasmussen, Zaak Beekman, Ambra Abdullahi Hassan, Alfredo Buttari, Daniela di Serafino, Michele Martone, Michele Colajanni, Fabio Cerioni, Stefano Maiolatesi, Dario Pascucci

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties, e.g., ℓ_1–Jacobi, Hybrid-FBGS, Hybrid-SGS, CG method, etc.

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties, e.g., ℓ_1–Jacobi, Hybrid-FBGS, Hybrid-SGS, CG method, etc.
- The prolongator P is built by dofs *aggregation based on matching* in the weighted (adjacency) graph of A.

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties, e.g., ℓ_1–Jacobi, Hybrid-FBGS, Hybrid-SGS, CG method, etc.
- The prolongator P is built by dofs *aggregation based on matching* in the weighted (adjacency) graph of A.
- The coarse solver is again a preconditioned CG method.

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties:

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties:
  - GS $A = M - N$, with $M = L + D$ and $N = -L^T$, where $D = \mathrm{diag}(A)$ and $L = \mathrm{tril}(A)$ is intrinsically sequential!

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties:
  - GS $A = M - N$, with $M = L + D$ and $N = -L^T$, where $D = \mathrm{diag}(A)$ and $L = \mathrm{tril}(A)$ is intrinsically sequential!
  - HGS Inexact block-Jacobi version of GS, in the portion of the row-block local to each process the method acts as the GS method.

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties:
  - GS $A = M - N$, with $M = L + D$ and $N = -L^T$, where $D = \mathrm{diag}(A)$ and $L = \mathrm{tril}(A)$ is intrinsically sequential!
  - HGS Inexact block-Jacobi version of GS, in the portion of the row-block local to each process the method acts as the GS method.
  - ℓ_1-HGS On process $p = 1, \ldots, np$ relative to the index set Ω_p we factorize $A_{pp} = L_{pp} + D_{pp} + L_{pp}^T$ for $D_{pp} = \mathrm{diag}(A_{pp})$ and $L_{pp} = \mathrm{trilu}(A_{pp})$ and select:

$$\begin{aligned} M_{\ell_1-HGS} =& \mathrm{diag}((M_{\ell_1-HGS})_p)_{p=1,\ldots np}, \\ (M_{\ell_1-HGS})_p =& L_{pp} + D_{pp} + D_{\ell_1 p}, \\ (d_{\ell_1})_{i=1}^{nb} =& \sum_{j \in \Omega_p^{nb}} |a_{ij}|. \end{aligned}$$

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties:

GS	$A = M - N$, with $M = L + D$ and $N = -L^T$, where $D = \mathrm{diag}(A)$ and $L = \mathrm{tril}(A)$ is intrinsically sequential!
HGS	Inexact block-Jacobi version of GS, in the portion of the row-block local to each process the method acts as the GS method.
ℓ_1-HGS	On process $p = 1, \ldots, np$ relative to the index set Ω_p we factorize $A_{pp} = L_{pp} + D_{pp} + L_{pp}^T$ for $D_{pp} = \mathrm{diag}(A_{pp})$ and $L_{pp} = \mathrm{trilu}(A_{pp})$ and select: $$M_{\ell_1-HGS} = \mathrm{diag}((M_{\ell_1-HGS})_p)_{p=1,\ldots np},$$
AINV	Block-Jacobi with an approximate inverse factorization on the block $\Rightarrow$ suitable for GPU application!

What is our *recipe*?

- The prolongator P is built by dofs *aggregation based on matching* in the weighted (adjacency) graph of A.

Given $\mathbf{w} \in \mathbb{R}^n$, let $P \in \mathbb{R}^{n \times n_c}$ and $P_f \in \mathbb{R}^{n \times n_f}$ be a prolongator and a complementary prolongator, such that:

$$\mathbb{R}^n = \mathrm{Range}(P) \oplus^{\perp} \mathrm{Range}(P_f), \quad n = n_c + n_f$$

$\mathbf{w} \in \mathrm{Range}(P)$: coarse space $\qquad$ $\mathrm{Range}(P_f)$: complementary space

$$[P, P_f]^T A [P, P_f] = \begin{pmatrix} P^T A P & P^T A P_f \\ P_f^T A P & P_f^T A P_f \end{pmatrix} = \begin{pmatrix} A_c & A_{cf} \\ A_{fc} & A_f \end{pmatrix}$$

A_c: coarse matrix $\qquad$ A_f: hierarchical complement

Sufficient condition for efficient coarsening

$A_f = P_f^T A P_f$ as well conditioned as possible, i.e.,
Convergence rate of *compatible relaxation*: $\rho_f = \| I - M_f^{-1} A_f \|_{A_f} \ll 1$

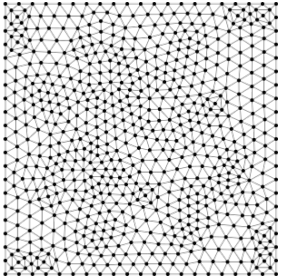
But *how* we achieve it?

Weighted graph matching

Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{A}$:

$$(\hat{A})_{i,j} = \hat{a}_{i,j} = 1 - \frac{2a_{i,j}w_i w_j}{a_{i,i}w_i^2 + a_{j,j}w_j^2},$$

- a *matching* $\mathcal{M}$ is a set of pairwise non-adjacent edges, containing no loops;
- a maximum product matching if it maximizes the product of the weights of the edges $e_{i \mapsto j}$ in it.



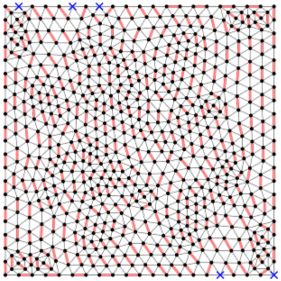
But *how* we achieve it?

Weighted graph matching

Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{A}$:

$$(\hat{A})_{i,j} = \hat{a}_{i,j} = 1 - \frac{2a_{i,j}w_i w_j}{a_{i,i}w_i^2 + a_{j,j}w_j^2},$$

- a *matching* $\mathcal{M}$ is a set of pairwise non-adjacent edges, containing no loops;
- a maximum product matching if it maximizes the product of the weights of the edges $e_{i\mapsto j}$ in it.



We divide the index set into matched vertexes $\mathcal{I} = \bigcup_{i=1}^{n_p} \mathcal{G}_i$, with $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ if $i \neq j$, and unmatched vertexes, i.e., n_s singletons G_i.

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \underbrace{\left[\begin{array}{cc} \underbrace{\begin{bmatrix} \mathbf{w}_{e_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{w}_{e_{n_p}} \end{bmatrix}}_{n_p} 2n_p & \mathbf{0} \\ \mathbf{0} & \underbrace{\begin{bmatrix} w_1/|w_1| & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_{n_s}/|w_{n_s}| \end{bmatrix}}_{n_s} n_s \end{array}\right]}_{n_c = n_p + n_s = J} n = 2n_p + n_s$$

$$= \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \dots, \mathbf{p}_J], \qquad \mathbf{w}_e = \frac{1}{\sqrt{w_i^2 + w_j^2}} \begin{bmatrix} w_i \\ w_j \end{bmatrix}.$$

$\Rightarrow$ The $\mathcal{M}$ on $\hat{A}$ produces A_f with diagonal entries $\hat{a}_{ij}$ for $(i,j) \in \mathcal{M}$ of maximal product.

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \ldots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - B_l A_l = (I - (M_l)^{-T} A_l)(I - P_l B_{l+1} (P_l)^T A_l)(I - M_l^{-1} A_l) \quad \forall l < nl,$$

where $A_{l+1} = (P_l)^T A_l P_l$ for $l = 0, \ldots, nl - 1$.

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \dots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - B_l A_l = (I - (M_l)^{-T} A_l)(I - P_l B_{l+1} (P_l)^T A_l)(I - M_l^{-1} A_l) \quad \forall l < nl,$$

where $A_{l+1} = (P_l)^T A_l P_l$ for $l = 0, \dots, nl - 1$.

- To increase dimension reduction we can perform more than one sweep of matching per step,

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \ldots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - B_l A_l = (I - (M_l)^{-T} A_l)(I - P_l B_{l+1} (P_l)^T A_l)(I - M_l^{-1} A_l) \quad \forall l < nl,$$

where $A_{l+1} = (P_l)^T A_l P_l$ for $l = 0, \ldots, nl - 1$.

- To increase dimension reduction we can perform more than one sweep of matching per step,
- To increase regularity of P_l we can consider a smoothed prolongator by applying a Jacobi smoother,
$$P_l^s = (I - \omega D_l^{-1} A_l) P_l, \text{ for } D_l = \operatorname{diag}(A_l).$$

From the matching to the prolongator

We can formally define a *prolongator*:

$$P = \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \dots, \mathbf{p}_J].$$

Then the preconditioner is the linear operator corresponding to the multiplicative composition of

$$I - B_l A_l = (I - (M_l)^{-T} A_l)(I - P_l B_{l+1} (P_l)^T A_l)(I - M_l^{-1} A_l) \quad \forall l < nl,$$

where $A_{l+1} = (P_l)^T A_l P_l$ for $l = 0, \dots, nl - 1$.

- To increase dimension reduction we can perform more than one sweep of matching per step,
- To increase regularity of P_l we can consider a smoothed prolongator by applying a Jacobi smoother,
- To increase the robustness we can use a non stationary solver as smoother.

Comparison with Hypre - CPU Runs - MareNostrum

Comparison with the preconditioners available in the `Hypre`, a state of the art preconditioning library from LLNL.

- Run on the MareNostrum machine up to 8192 cores
- Test: 3D Constant coefficient Poisson Problem with FCG
- DoF: 256k unknown $\times$ MPI core
- Measures: Operator Complexity opc $= \frac{\sum_{l=0}^{\text{nl}-1} \operatorname{nnz}(A_l)}{\operatorname{nnz}(A_0)}$ and Solve Time (s).

Comparison with Hypre - CPU Runs - MareNostrum

Comparison with the preconditioners available in the `Hypre`, a state of the art preconditioning library from LLNL.

- Run on the MareNostrum machine up to 8192 cores
- Test: 3D Constant coefficient Poisson Problem with FCG
- DoF: 256k unknown $\times$ MPI core
- Measures: Operator Complexity $\text{opc} = \frac{\sum_{l=0}^{\text{nl}-1} \text{nnz}(A_l)}{\text{nnz}(A_0)}$ and Solve Time (s).

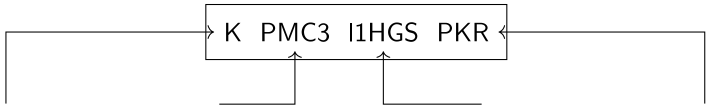
Scaling

There are two common notions of scalability:

- Strong scaling is defined as how the solution time varies with the number of processors for a fixed total problem size.
- Weak scaling is defined as how the solution time varies with the number of processors for a fixed problem size per processor.

Comparison with Hypre - CPU Runs - MareNostrum

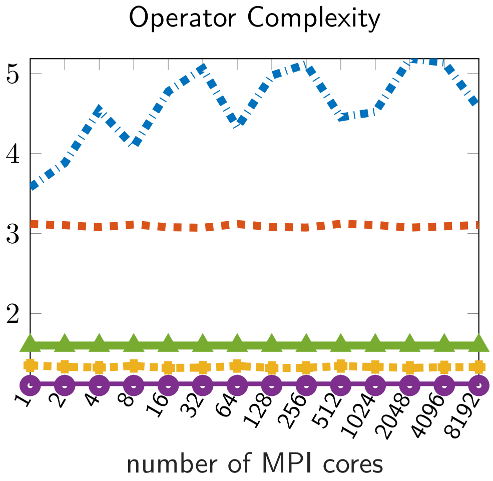
Giving a name to preconditioners with many parameters:

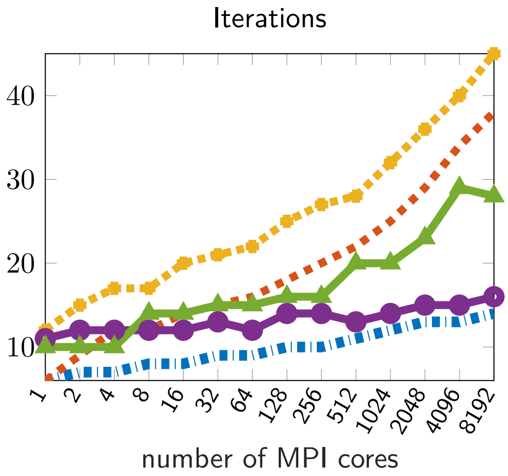


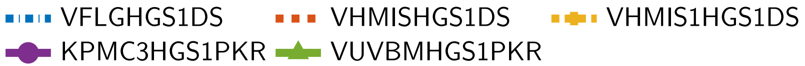
Cycle	Aggregation	Smoother	Coarsest solver
K	Unsmoothed **P**arallel **M**at**c**hing **3/4**	**H**ybrid **G**auss-**S**eidel	**P**reconditioned **Kr**ylov Method
V	**S**moothed **P**arallel **M**atching **3/4**	l_1–**H**ybrid **G**auss-**S**eidel	
	Smoothed **VBM**	**H**ybrid **INVK**	
		l_1–**H**ybrid **INVK**	
		l_1–**Ja**cobi	

For `Hypre` we test HMIS and Falgout coarsening schemes.

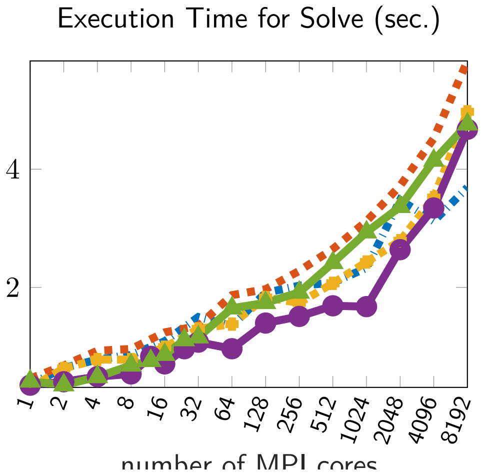
Comparison with Hypre - CPU Runs - MareNostrum

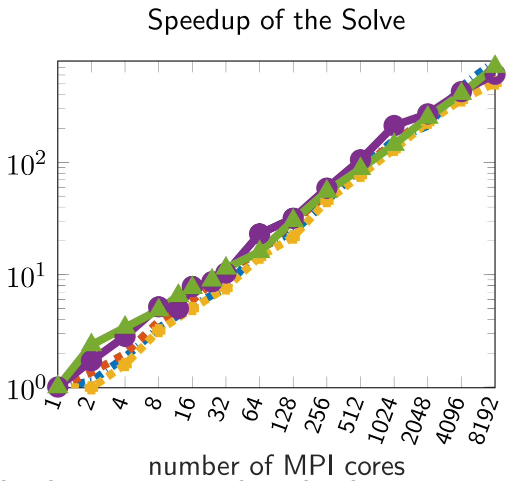






Comparison with Hypre - CPU Runs - MareNostrum





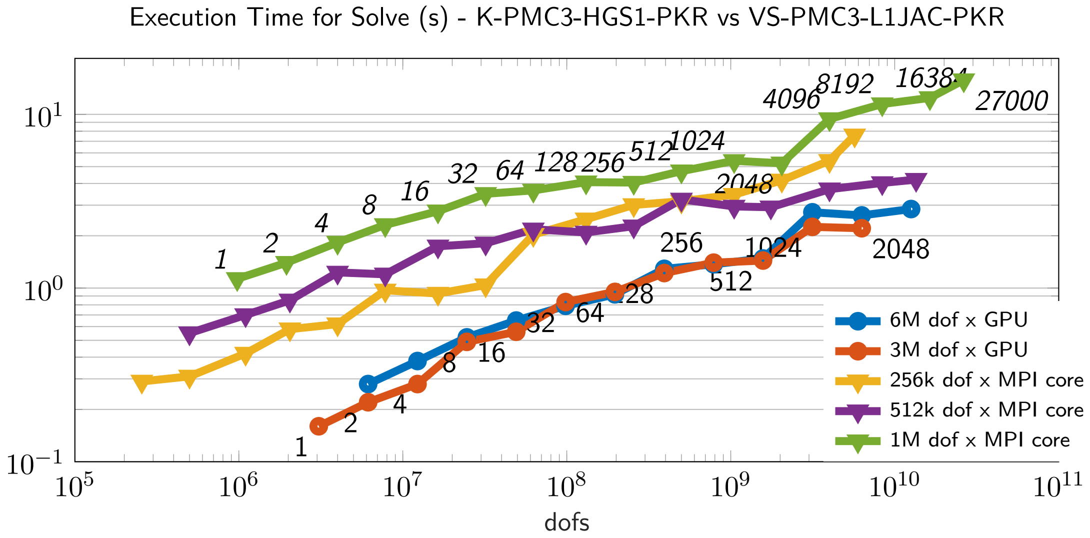
VFLGHGS1DS VHMISHGS1DS VHMISH1GS1DS
KPMC3HGS1PKR VUVBM1HGS1PKR

Weak Scalability - CPU/GPU Runs - Piz Daint

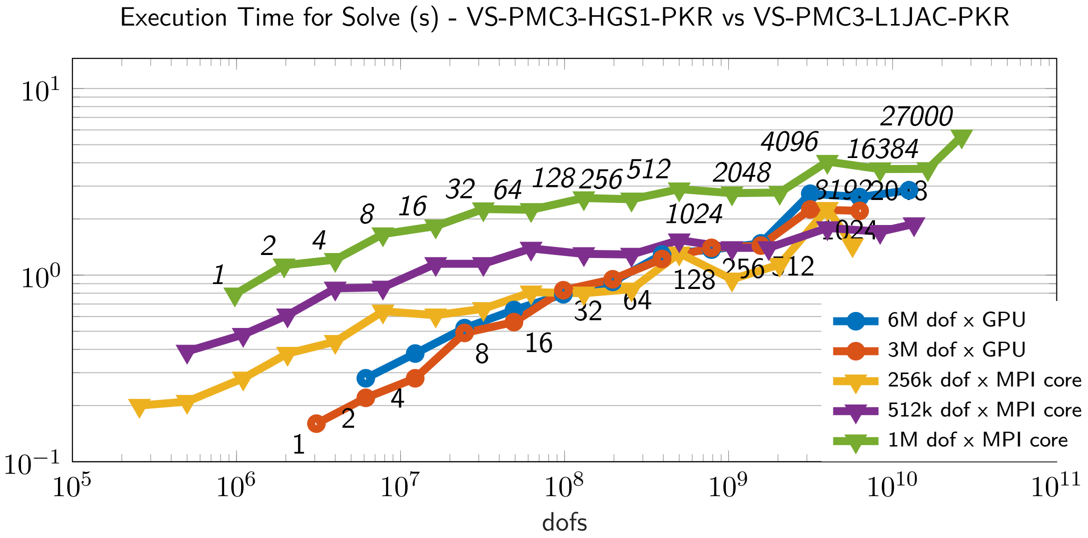
The resulting performance of the multigrid preconditioner in term of implementation scalability depends also on how effective the coarsening procedure is, and on how well balanced is the distribution of the coarsest matrix.

- Run on the Piz Daint machine up to 28800 cores and 2048 GPUs
- Test: 3D Constant coefficient Poisson Problem with FCG
- DoF: 256k/512k/1M unknown $\times$ MPI core and 3M/6M per GPUs
- Measures: execution time for solve

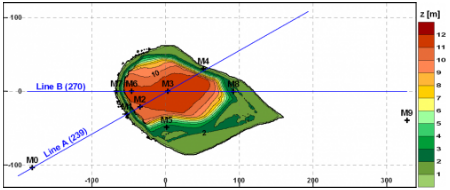
Weak Scalability - CPU/GPU Runs - Piz Daint



Weak Scalability - CPU/GPU Runs - Piz Daint



A Large Eddy Scale simulation inside Alya



Bolund is an isolated hill situated in Roskilde Fjord, Denmark. An almost vertical escarpment in the prevailing W-SW sector ensures flow separation in the windward edge resulting in a complex flow field.

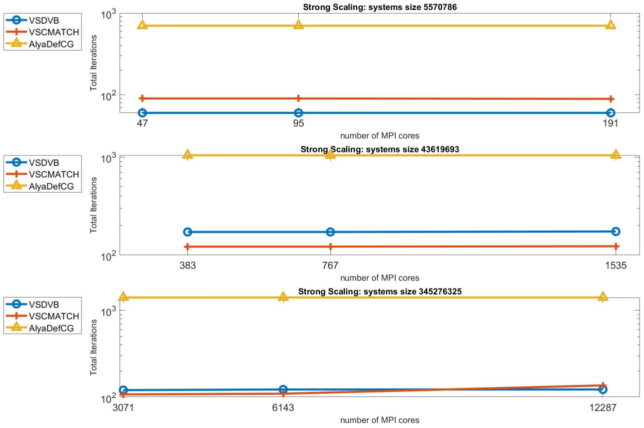
- **Model**: 3D incompressible unsteady Navier-Stokes equations for the Large Eddy Simulations of turbulent flows,
- **Discretization**: low-dissipation mixed FEM (linear FEM both for velocity and pressure),
- **Time-Stepping**: non-incremental fractional-step for pressure, explicit fourth order Runge-Kutta method for velocity.

Alya

Alya is a simulation code for high performance computational mechanics. It solves coupled multiphysics problems using high performance computing techniques for distributed and shared memory supercomputers, together with vectorization and optimization at the node level.

Bolund Test Case - Strong Scaling - Pressure Eq.

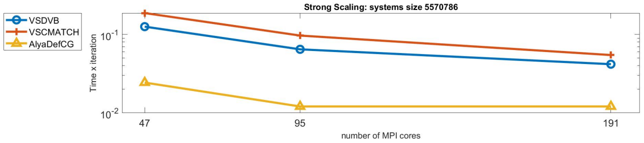
Fixed size problem with $n = 5,570,786; 43,619,693; 345.276.325$ dofs, 100 time steps

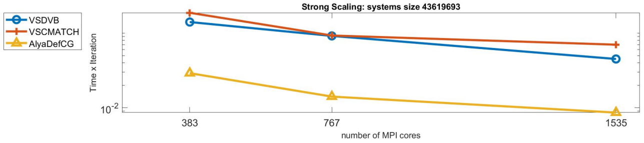


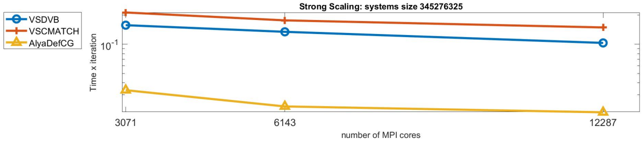
Total number of linear iterations is smaller and stable for increasing number of cores,

Bolund Test Case - Strong Scaling - Pressure Eq.

Fixed size problem with $n = 5,570,786;43,619,693;345.276.325$ dofs, 100 time steps



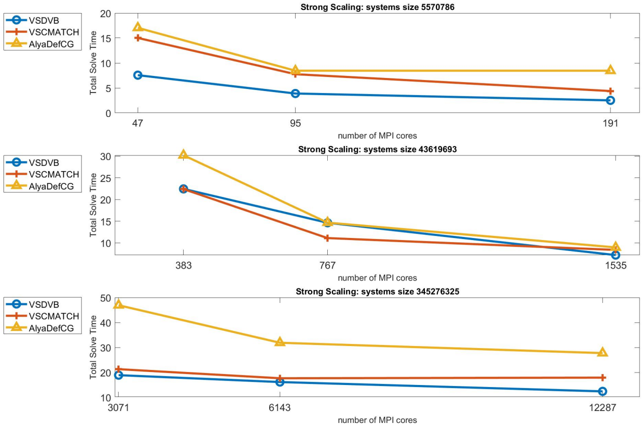




Total number of linear iterations is smaller and stable for increasing number of cores, the time needed per each iteration decreases for increasing number of cores.

Bolund Test Case - Strong Scaling - Pressure Eq.

Fixed size problem with $n = 5,570,786;43,619,693;345.276.325$ dofs, 100 time steps



Total number of linear iterations is smaller and stable for increasing number of cores, the time needed per each iteration decreases for increasing number of cores.

Bibliography I

- Bell, N., L. N. Olson, and J. Schroder (2022). “PyAMG: Algebraic Multigrid Solvers in Python”. In: *Journal of Open Source Software* 7.72, p. 4142. DOI: 10.21105/joss.04142. URL: https://doi.org/10.21105/joss.04142.
- Brandt, A. (2000). “General highly accurate algebraic coarsening”. In: vol. 10. Multilevel methods (Copper Mountain, CO, 1999), pp. 1–20.
- Briggs, W. L., V. E. Henson, and S. F. McCormick (2000). *A multigrid tutorial*. Second. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, pp. xii+193. ISBN: 0-89871-462-1. DOI: 10.1137/1.9780898719505. URL: https://doi.org/10.1137/1.9780898719505.
- Leiserson, C. E. et al. (2020). “There’s plenty of room at the Top: What will drive computer performance after Moore’s law?” In: *Science* 368.6495. ISSN: 0036-8075. DOI: 10.1126/science.aam9744. eprint: https://science.sciencemag.org/content/368/6495/eaam9744.full.pdf. URL: https://science.sciencemag.org/content/368/6495/eaam9744.

Bibliography II

- Ruge, J. W. and K. Stüben (1987). "Algebraic multigrid". In: *Multigrid methods*. Vol. 3. Frontiers Appl. Math. SIAM, Philadelphia, PA, pp. 73–130.
- Trottenberg, U., C. W. Oosterlee, and A. Schüller (2001). *Multigrid*. With contributions by A. Brandt, P. Oswald and K. Stüben. Academic Press, Inc., San Diego, CA, pp. xvi+631. ISBN: 0-12-701070-X.
- Vaněk, P., J. Mandel, and M. Brezina (1996). "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems". In: vol. 56. 3. International GAMM-Workshop on Multi-level Methods (Meisdorf, 1994), pp. 179–196. DOI: 10.1007/BF02238511. URL: https://doi.org/10.1007/BF02238511.
- Vassilevski, P. S. (2008). *Multilevel block factorization preconditioners*. Matrix-based analysis and algorithms for solving finite element equations. Springer, New York, pp. xiv+529. ISBN: 978-0-387-71563-6.
- Xu, J. and L. Zikatanov (2017). "Algebraic multigrid methods". In: *Acta Numer.* 26, pp. 591–721. ISSN: 0962-4929. DOI: 10.1017/S0962492917000083. URL: https://doi.org/10.1017/S0962492917000083.