Iterative Methods and Multigrid

Part 3: Preconditioning: Based on Domain Decomposition

Domain Decomposition Book

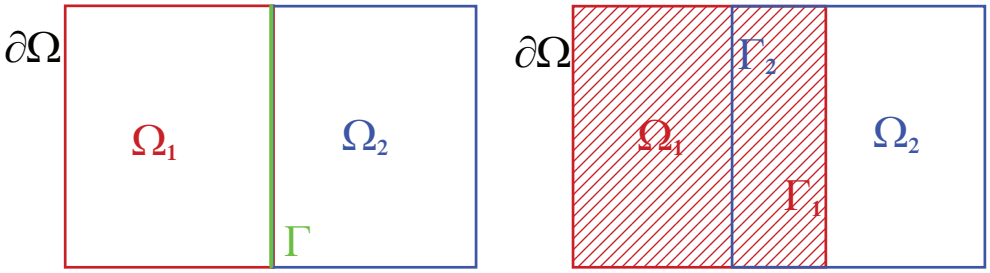
Significant parts of this discussion come from the (excellent) book

Barry Smith, Petter Bjørstad, and William Gropp,
Domain Decomposition, Parallel Multilevel Methods for Ellipitic Partial Differential Equations, Cambridge, 1996.

See chapter 10 in Meurant.
Section 10.2 discusses multiplicative and additive Schwarz methods for domain decompositions with overlap.
Section 10.4 discusses substructuring or Schur complement methods for domain decompositions without overlap. It also surveys the many proposed choices for preconditioning the Schur complement.

Domain decomposition methods are based on the idea that we can approximate the effect of A by solving over subdomains for the restrictions of A to those subdomains and combining the solutions in a clever way. Then iterate till we have a solution that is sufficiently accurate.

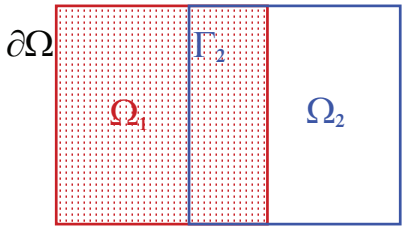
We discuss two basic ways of doing this:
1) with domains that have some overlap
2) with domains that do not have an overlap



Overlapping domain methods

The basic idea (Schwarz):

Consider PDE $Lu = f$ on domain Ω with $u = g$ on boundary $\partial\Omega$.



Choose approximate solution g_1 on Γ_1 and solve
$Lu_1 = f$ in Ω_1 and $u_1 = g$ on $\partial\Omega \cap \partial\Omega_1$ and $u_1 = g_1$ on Γ_1.
This gives approximate solution g_2 on Γ_2 and we solve
$Lu_2 = f$ in Ω_2 and $u_2 = g$ on $\partial\Omega \cap \partial\Omega_2$ and $u_2 = g_2$ on Γ_2.
Solution found if $Lu_1 = f$, $Lu_2 = g$, and $u_1 \equiv u_2$.

Originally introduced by H. A. Schwarz (1870) to prove existence of solutions to elliptic problems on complicated domains.

Matrix notation

Assume overlap large enough that equations for nodes in $\Omega_1 \backslash \Omega_2$ are not coupled with equations for nodes in $\Omega_2 \backslash \Omega_1$.

Order equations: nodes in $\Omega_1 \backslash \Omega_2$ first, then nodes in $\Omega_1 \cap \Omega_2$, and then nodes in $\Omega_2 \backslash \Omega_1$.

Equations: $\begin{pmatrix} A_{11} & A_{21} & \\ A_{21} & A_{22} & A_{23} \\ & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} u_{\Omega_1 \backslash \Omega_2} \\ u_{\Omega_1 \cap \Omega_2} \\ u_{\Omega_2 \backslash \Omega_1} \end{pmatrix} = \begin{pmatrix} f_{\Omega_1 \backslash \Omega_2} \\ f_{\Omega_1 \cap \Omega_2} \\ f_{\Omega_2 \backslash \Omega_1} \end{pmatrix}$.

Need starting guess on $\Omega_2 \backslash \Omega_1$ (nodes coupling to $\Omega_1 \cap \Omega_2$):

Solve for subdomain 1, then for subdomain 2, and so on.

Multiplicative Schwartz

Solve problem on subdomain 1 using data from subdomain 2

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} u_{\Omega_1 \backslash \Omega_2} \\ u_{\Omega_1 \cap \Omega_2} \end{pmatrix}^{(k)} = \begin{pmatrix} f_{\Omega_1 \backslash \Omega_2} \\ f_{\Omega_1 \cap \Omega_2} - A_{23} u_{\Omega_2 \backslash \Omega_1}^{(k-1)} \end{pmatrix}$$

Solve problem on subdomain 2 using data from subdomain 1

$$\begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} u_{\Omega_1 \cap \Omega_2} \\ u_{\Omega_2 \backslash \Omega_1} \end{pmatrix}^{(k)} = \begin{pmatrix} f_{\Omega_1 \cap \Omega_2} - A_{21} u_{\Omega_1 \backslash \Omega_2}^{k} \\ f_{\Omega_2 \backslash \Omega_1} \end{pmatrix}$$

Note that we can consider this iteration as a special case of Block Gauss-Seidel (with overlap).

Algebraic framework

Introduce $R_1 : R_1^T u = u_{\Omega_1}$ (including overlaps) and $R_i u_{\Omega_i} = \begin{pmatrix} u_{\Omega_1}^T \\ 0 \end{pmatrix}$.

Define R_2 analogously.

Note $A_{\Omega_i} = R_i^T A R_i$.

Then multiplicative Schwartz becomes:

$$u^{(k+1/2)} = u^{(k)} + R_1 (R_1^T A R_1)^{-1} R_1 (f - A u^{(k)})$$
$$u^{(k+1)} = u^{(k+1/2)} + R_2 (R_2^T A R_2)^{-1} R_2 (f - A u^{(k+1/2)})$$

Now define $B_i = R_i (R_i^T A R_i)^{-1} R_i$

$$u^{(k+1)} = u^{(k)} + (B_1 + B_2 - B_2 A B_1)(f - A u^{(k)}) = u^{(k)} + B(f - A u^{(k)})$$

solves $BAu = Bf$ (preconditioned equation) ➔ Krylov method.

Algebraic framework

Using Krylov method to solve $BAu = Bf$, compute efficiently $v = Br$:

1) $v = B_1 r$

2) $v = v + B_2 (r - Av)$

For CG or MINRES (A HPD) we want to symmetrize preconditioner:

$$u^{(k+1/3)} = u^{(k)} + B_1 (f - A u^{(k)})$$
$$u^{(k+2/3)} = u^{(k+1/3)} + B_2 (f - A u^{(k+1/3)})$$
$$u^{(k+1)} = u^{(k+2/3)} + B_1 (f - A u^{(k+2/3)})$$

This is equivalent to

$$u^{(k+1)} = u^{(k)} + (B_1 + B_2 - B_2 A B_1 - B_1 A B_2 + B_1 A B_2 A B_1)(f - A u^{(k)})$$

with corresponding preconditioner

$$B = (B_1 + B_2 - B_2 A B_1 - B_1 A B_2 + B_1 A B_2 A B_1) =$$
$$B_1 + (I - B_1 A) B_2 (I - A B_1)$$

Algebraic framework

These corrections are in fact projections of the error to respective subspaces (orthogal if A SPD):

Let $P_i = B_i A$, then $R_i (R_i^T A R_i)^{-1} R_i^T r = B_i r = B_i A A^{-1} r = P_i e$

$P_i^2 = R_i (R_i^T A R_i)^{-1} R_i^T A R_i (R_i^T A R_i)^{-1} R_i^T A = R_i (R_i^T A R_i)^{-1} R_i^T A = P_i$
Hence P_i is a projector onto $\mathrm{range}(R_i)$

Furthermore,
$\langle P_i x, y \rangle_A = y^T A P_i x = y^T A B_i A x = (B_i A y)^T A x = \langle x, P_i y \rangle_A$.

So, P_i is symmetric (self-adjoint) w.r.t. A-inner product;
therefore, P_i is an orthogonal projector (A-ip) onto $\mathrm{range}(R_i)$.

This means that $P_i e$ is the best approximation in $\mathrm{range}(R_i)$ to the error, in the A-norm.

Algebraic framework

The multiplicative Schwartz method can now be written as

$u^{(k+1/2)} = u^{(k)} + P_1 e^{(k)}$
$u^{(k+1)} = u^{(k+1/2)} + P_2 e^{(k+1/2)}$

So the algorithm at each step projects the error onto a subdomain (subspace) and adds the correction to the approximation on the subdomain. Note that this can be done without knowing the error.

The procedure of restriction to a subspace, solving a local (small) system and mapping the result back to the original space underlies most domain decomposition/multigrid/multilevel methods.

The selection of the appropriate subdomains plays an important role in the domain decomposition method.

Additive Schwartz

An alternative to the so-called multiplicative Schwarz method is the additive Schwarz method.

This method typically converges more slowly, but has better parallel properties and is symmetric if A is. Just as the multiplicative Schwarz method is Gauss-Seidel-like, the additive method is Jacobi-like.

$$u^{(k+1)} = u^{(k)} + (B_1 + B_2)(f - Au^{(k)})$$

Note that this iteration is derived from simultaneously solving

$$u_{\Omega_1}^{(k+1)} = u_{\Omega_1}^{(k)} + (R_1^T A R_1)^{-1} R_1^T (f - Au^{(k)})$$

$$u_{\Omega_2}^{(k+1)} = u_{\Omega_2}^{(k)} + (R_2^T A R_2)^{-1} R_2^T (f - Au^{(k)})$$

where the results in the overlap region are added.

Additive Schwartz

The method is Jacobi-like because we (can) solve for the blocks of equations simultaneously.

It is also easy to see that each projection is symmetric if A is.

Since, we solve for each block independently the method is easy to parallelize and has good parallel performance.
In contrast, the multiplicative Schwarz method requires each subdomain solve (except the first) to wait for the results of the previous one.

This problem can be reduced by relaxing the strict ordering of domains that do not share any unknowns. Assign colors to the subdomains in a such a way that no two subdomains that share unknowns have the same color. Then updates all subdomains with the same color simultaneously.

Additive Schwartz

The additive Schwartz method in this basic form does not converge, in general. Therefore, the method is always applied with either acceleration (Krylov method) or damping.

Generally we would like to work on multiple domains rather than just two. What would be an obvious extension?

More subdomains

For multiple domains both the multiplicative Schwartz method and the additive Schwartz method have obvious extensions.

Multiplicative (strict version):

$$u^{(k+1/p)} = u^{(k)} + B_1(f - Au^{(k)})$$
$$u^{(k+2/p)} = u^{(k+1/p)} + B_2(f - Au^{(k+1/p)})$$
...
$$u^{(k+1)} = u^{(k+(p-1)/p)} + B_p(f - Au^{(k+(p-1)/p)})$$

This gives:

$$e^{(n+1)} = (I - B_pA)(I - B_{p-1}A)\cdots(I - B_1A)e^{(n)} \quad \text{and}$$

$$B = [I - (I - B_pA)\cdots(I - B_1A)]A^{-1} = A^{-1}[I - (I - B_pA)\cdots(I - B_1A)].$$

Obviously A^{-1} is not needed for implementation.

More subdomains

Additive Schwartz for multiple subdomains can be written as

$$u^{(k+1)} = u^{(k)} + B_1(f - Au^{(k)}) + B_2(f - Au^{(k)}) + \cdots + B_p(f - Au^{(k)})$$

$$B = B_1 + B_2 + \cdots + B_p$$

Convergence for many subdomains

Imagine dividing a large rectangular domain in many strips, each couple with some overlap.
We solve the Laplace equation $\Delta u = 0$ with zero boundary conditions except at the east boundary where we set $u = 1$. Now we use multiplicative Schwartz starting with the left(west)-most strip and moving to the right or additive Schwartz.

What will happen?
What is the minimum number of iterations to converge?

Multilevel methods

At each step only one extra strip will get a nontrivial update. Consequently, it takes at least #number of subdomains iterations to converge.

The reason is the mismatch between the physics underlying elliptic problems (diffusion, pressure, ...) and the solution procedure. The essence of elliptic problems is that a boundary condition (or local source term) eventually influences the solution over the entire domain. So a solution procedure that only step-wise updates neighboring regions is inherently bounded in its rate of convergence by the 'distance' between regions.

We would like to do updates that have a more global effect:

A coarse grid (subdomain decomposition) correction →
Multi-level methods

Multilevel methods

As we saw previously, in the many domain case we typically have very slow convergence, because there is no global transfer of information.

This leads to slow convergence for low frequency functions, because (apart from constant term) they almost satisfy the homogenous differential equation.

Hence, we must look for a way to force quicker convergence of those terms. To this end we introduce coarser levels (multilevel method). This means we have fewer domains (typically) and so better global coupling, and the low frequency modes look more oscillatory.

Multilevel methods

Consider a coarse 'grid' embedded in the fine grid.

R represents linear interpolation from coarse grid to fine grid
R^T represents restriction from fine to coarse grid

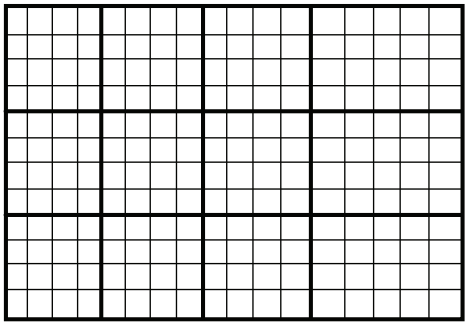
A simple coarse grid correction can be defined as

$$u_F = u_F + RA_C^{-1}R^T(f - A_F u_F)$$

Compute fine grid residual, restrict to coarse grid, solve coarse grid equations, interpolate back to fine grid, and add correction.

However, $RA_C^{-1}R^T$ alone as preconditioner won't work, because it has a large null-space. The null space mainly contains the high frequency modes. Hence we need to augment this step with a fine grid preconditioner (correction).

Multilevel methods



Multilevel methods

Call $B_C = R A_C^{-1} R^T$ the coarse grid preconditioner. For the fine grid preconditioner we typically take a local operator, subdomain solves or diagonal preconditioning (you can't get more local than a point). However, in principle any nonsingular preconditioner is possible.

Define some B_F and iteration becomes:

$$u_F^{n+1/2} = u_F^n + B_C(f - A_F u_F^n)$$
$$u_F^{n+1} = u_F^{n+1/2} + B_F(f - A_F u_F^{n+1/2})$$

Same form as before (2 subdomains), but very different construction. B_F may be the entire multiplicative or additive Schwartz iteration.

Written as one step method iteration looks like

$$u_F^{n+1} = u_F^n + (B_C + B_F - B_F A_F B_C)(f - A_F u_F^n)$$

Multilevel methods

Using the same operators or preconditioners we can also define an additive two level method:

$$u_F^{n+1} = u_F^n + (B_C + B_F)(f - A_F u_F^n)$$

Again the additive variant is always used with a Krylov method.

The coarse grid correction takes care of the low frequencies (smooth modes) of the error.
So B_F must be effective in damping the high frequency components of the error: Schwarz smoothers (MG-like).
Schwarz smoothers are typically one level additive or multiplicative overlapping domain decomposition methods.

We may apply the smoothing step more than once.

Multilevel methods

In two level Schwarz methods the fine grid preconditioner may be either a additive or multiplicative (overlapping) Schwarz method/preconditioner.

We then may combine the fine and coarse grid preconditioners in either a multiplicative or additive way.

This gives rise to various combinations.

Let the B_i denote the fine grid subdomain solves: $R_i(R_i^T A R_i)^{-1} R_i^T$
The coarse grid correction is given by $RA_C^{-1}R^T$

Multilevel methods

1. $v = (RA_C^{-1}R^T + \Sigma_i B_i)r$
   two level additive Schwarz preconditioner

2. $v = RA_C^{-1}R^T r$
   $v = v + B_1(f - A_F v)$ ... $v = v + B_p(f - A_F v)$
   two level multiplicative Schwarz preconditioner

3. $v = B_1 r$
   $v = v + B_2(f - A_F v)$ ... $v = v + B_p(f - A_F v)$
   $v = v + RA_C^{-1}R^T r$
   two level hybrid Schwarz preconditioner

4. $v = \Sigma_i B_i r$
   $v = v + RA_C^{-1}R^T(r - A_F v)$
   two level hybrid Schwarz preconditioner

Other variants, multiple smoothing steps, symmetric versions

Multilevel methods

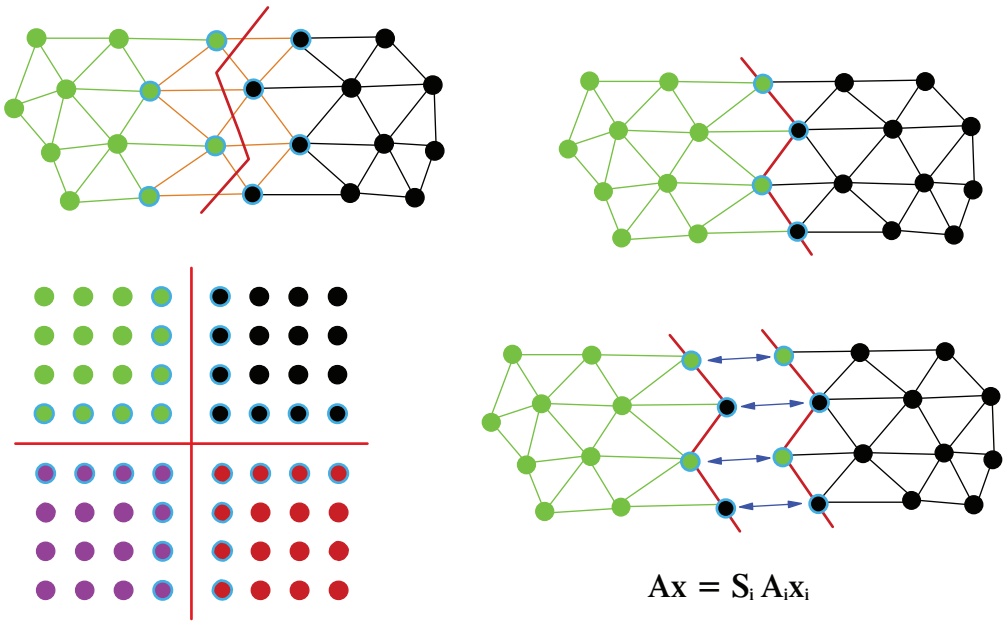
For nonsymmetric problems not necessary to have the restriction and interpolation operators be each other's transpose.

Possible, $B_C = JA_C^{-1}R$, where J is interpolation operator.
However, generally best (theory) to have $J^T = R$.

Many possibilities for coarse grid correction.
Coarse grid problem and the transfer operators should be chosen together.

Rule of thumb: If one is chosen, take the other such that coarse grid correction is as close as possible to (orthogonal) projection.

Mesh Partitioning



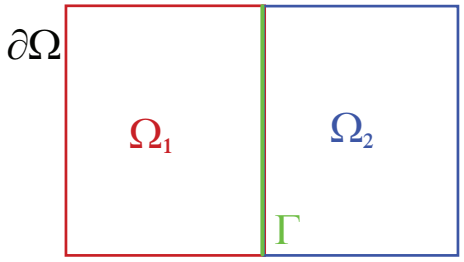
$Ax = S_i A_i x_i$

Substructuring methods

Now we will discuss methods based on nonoverlapping subdomains
These methods are also called substructuring methods

These methods actually have their roots in out-of-core direct methods, and we look at those first

Consider the equations for two subdomains (uncoupled) and an interface (appropriately ordered).



Substructuring methods

Order the equations: first Ω_1, then Ω_2, and finally Γ.

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

Suppose matrix too large to store at once. How to solve in pieces?
First eliminate block in last row in first column:

$$\begin{pmatrix} I & & \\ & I & \\ -A_{31}A_{11}^{-1} & 0 & I \end{pmatrix} \begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} I & & \\ & I & \\ -A_{31}A_{11}^{-1} & 0 & I \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} - A_{31}A_{11}^{-1}A_{13} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 - A_{31}A_{11}^{-1}f_1 \end{pmatrix}$$

Substructuring methods

This last step only involved the first block of rows and the last one

In next step we eliminate block in last row second column:

$$\begin{pmatrix} I & & \\ & I & \\ 0 & -A_{32}A_{22}^{-1} & I \end{pmatrix}\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & A_{32} & A_{33}-A_{31}A_{11}^{-1}A_{13} \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} =$$
$$\begin{pmatrix} I & & \\ & I & \\ 0 & -A_{32}A_{22}^{-1} & I \end{pmatrix}\begin{pmatrix} f_1 \\ f_2 \\ f_3-A_{31}A_{11}^{-1}f_1 \end{pmatrix}$$

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & S \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3-A_{31}A_{11}^{-1}f_1-A_{32}A_{22}^{-1}f_2 \end{pmatrix}$$ where

$$S = A_{33}-A_{31}A_{11}^{-1}A_{13}-A_{32}A_{22}^{-1}A_{23}$$

Substructuring methods

This last step only involved the 2nd block of rows and the last one

We can now solve independently for u_3 using the equation

$$(A_{33}-A_{31}A_{11}^{-1}A_{13}-A_{32}A_{22}^{-1}A_{23})u_3 = f_3-A_{31}A_{11}^{-1}f_1-A_{32}A_{22}^{-1}f_2$$

The small (dense) matrix $(A_{33}-A_{31}A_{11}^{-1}A_{13}-A_{32}A_{22}^{-1}A_{23})$ is called the Schur complement.

After we solve for u_3 we can again solve independently the systems

$A_{22}u_2 = f_2-A_{23}u_3$ and
$A_{11}u_1 = f_1-A_{13}u_3$

Note that computing the Schur complement (and rhs) involve subdomain solves. This allows the direct solution of very large systems without ever having the entire matrix in core.

Substructuring methods

We may also solve the Schur complement equation iteratively. Krylov methods only need matrix-vector product. This can be carried out without forming the Schur complement explicitly!

$$S \equiv (A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23})$$

Major advantage because S dense and not necessarily small.

For positive definite A, the condition number of S at least as good as that of A. In many cases, even much better.

For many second order elliptic PDEs the condition number of A grows as $O(1/h^2)$, whereas the condition number of S grows only as $O(1/h)$.

Substructuring methods

We can solve iteratively for $Su_3 = f_S$ and afterwards solve independently for other subdomains (obviously the approach can be extended to many subdomains).

However, since we do subdomain solves anyway, we can solve for other subdomains at the same time.

Krylov method for

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & S \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_S \end{pmatrix}$$

Substructuring methods

We can further improve convergence by preconditioning the equation $Su_3 = f_S$ in

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & S \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_S \end{pmatrix}$$

For example Neumann-Dirichlet or Neumann-Neumann preconditioners (see book Smith, Bjørstad, and Gropp).
This leads to number of iterations for convergence independent of h (the mesh width).

Many other approaches/preconditioners based on the substructuring idea.

Substructuring methods

Condition number of $S \equiv (A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23})$
Note if A HPD then S HPD (proof?)

$\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$, where $\lambda_{\max} = \max_x \frac{x^T A x}{x^T x}$ and $\lambda_{\min} = \min_x \frac{x^T A x}{x^T x}$

Consider x such that

$$A_{11}x_1 + A_{13}x_3 = 0 \Rightarrow x_1 = -A_{11}^{-1}A_{13}x_3$$
$$A_{22}x_2 + A_{23}x_3 = 0 \Rightarrow x_2 = -A_{22}^{-1}A_{23}x_3$$

Then

$$x^T A x = x_3^T (A_{31}x_1 + A_{32}x_2 + A_{33}x_3) =$$

$$-x_3^T A_{31}A_{11}^{-1}A_{13}x_3 - x_3^T A_{32}A_{22}^{-1}A_{23}x_3 + x_3^T A_{33}x_3 = x_3^T S x_3$$

Substructuring methods

Condition number of $S \equiv (A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23})$

Note if A HPD then S HPD (proof?)

$\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$, where $\lambda_{\max} = \max_x \frac{x^T Ax}{x^T x}$ and $\lambda_{\min} = \min_x \frac{x^T Ax}{x^T x}$

Now we have

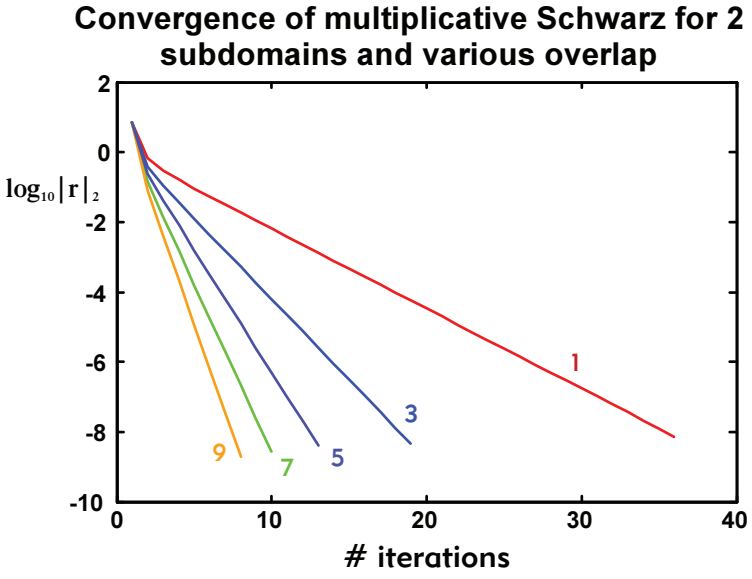
$$\lambda_{\min}(S) = \min_{x_3 \neq 0} \frac{x_3^T S x_3}{x_3^T x_3} = \min_{\substack{x : x_3 \neq 0 \\ A_{11}x_1 + A_{13}x_3 = 0 \\ A_{22}x_2 + A_{23}x_3 = 0}} \frac{x^T Ax}{x^T x} \geq \min_{x \neq 0} \frac{x^T Ax}{x^T x} = \lambda_{\min}(A)$$

Likewise, $\lambda_{\max}(S) \leq \lambda_{\max}(A)$.

Hence $\kappa(S) \leq \kappa(A)$.

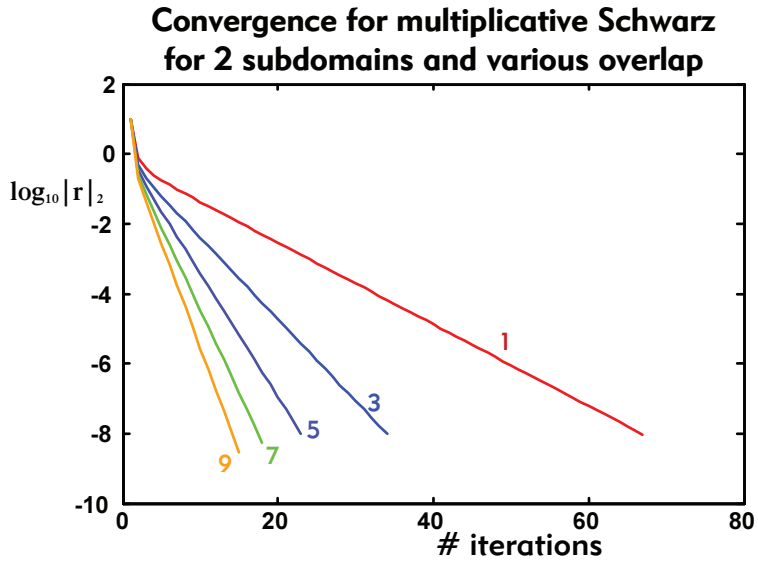
Convergence of Mult. Schwarz

$p=q=1, r=s=0, t=0, f=0, u_s=u_w=1, u_n=u_e=0, h=1/26$



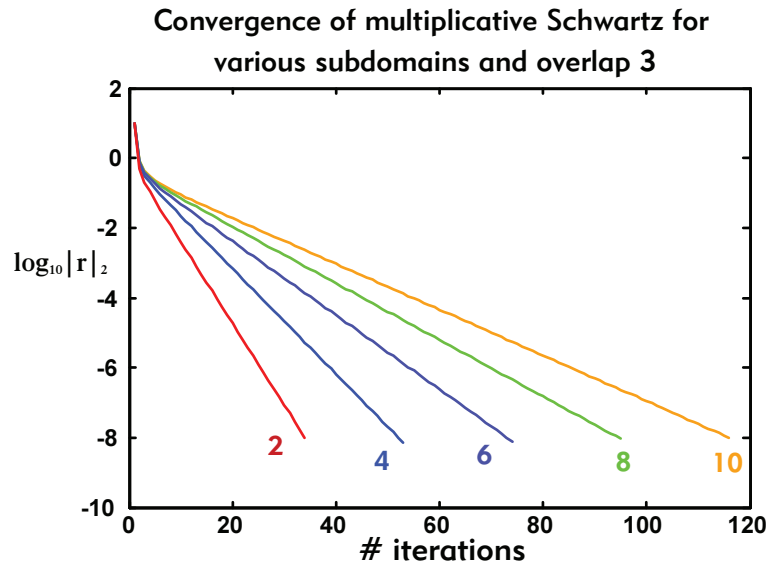
Convergence of Mult. Schwarz

$p=q=1, r=s=0, t=0, f=0, u_s=u_w=1, u_n=u_e=0, h=1/51$



Convergence of Mult. Schwarz

$p=q=1, r=s=0, t=0, f=0, u_s=u_w=1, u_n=u_e=0, h=1/51$



Multilevel methods

As we saw previously, in the many domain case we typically have very slow convergence, because there is no global transfer of information.

This leads to slow convergence for low frequency functions, because (apart from constant term) they almost satisfy the homogenous differential equation.

Hence, we must look for a way to force quicker convergence of those terms. To this end we introduce coarser levels (multilevel method). This means we have fewer domains (typically) and so better global coupling, and the low frequency modes look more oscillatory.