PARALLEL PRECONDITIONERS

Gérard MEURANT

CEA, DIF

Evolution of supercomputers

- Years 60–mid 70 : scalar computers
- mid 70–... 2000 vector computers
- 80–... 2000 ? multiprocessor vector computers with shared memory
- mid 80–... 2000 distributed memory parallel computers
- end 90–?? SMP clusters (distributed clusters of nodes with shared memory)

◦ PC clusters

Trends in computer architecture

- Tflops class computers need a "large" (1000) processors
- Use of "off the shelf" microprocessors
- Need for very efficient networking (latency, bandwith)
- Actual trend:

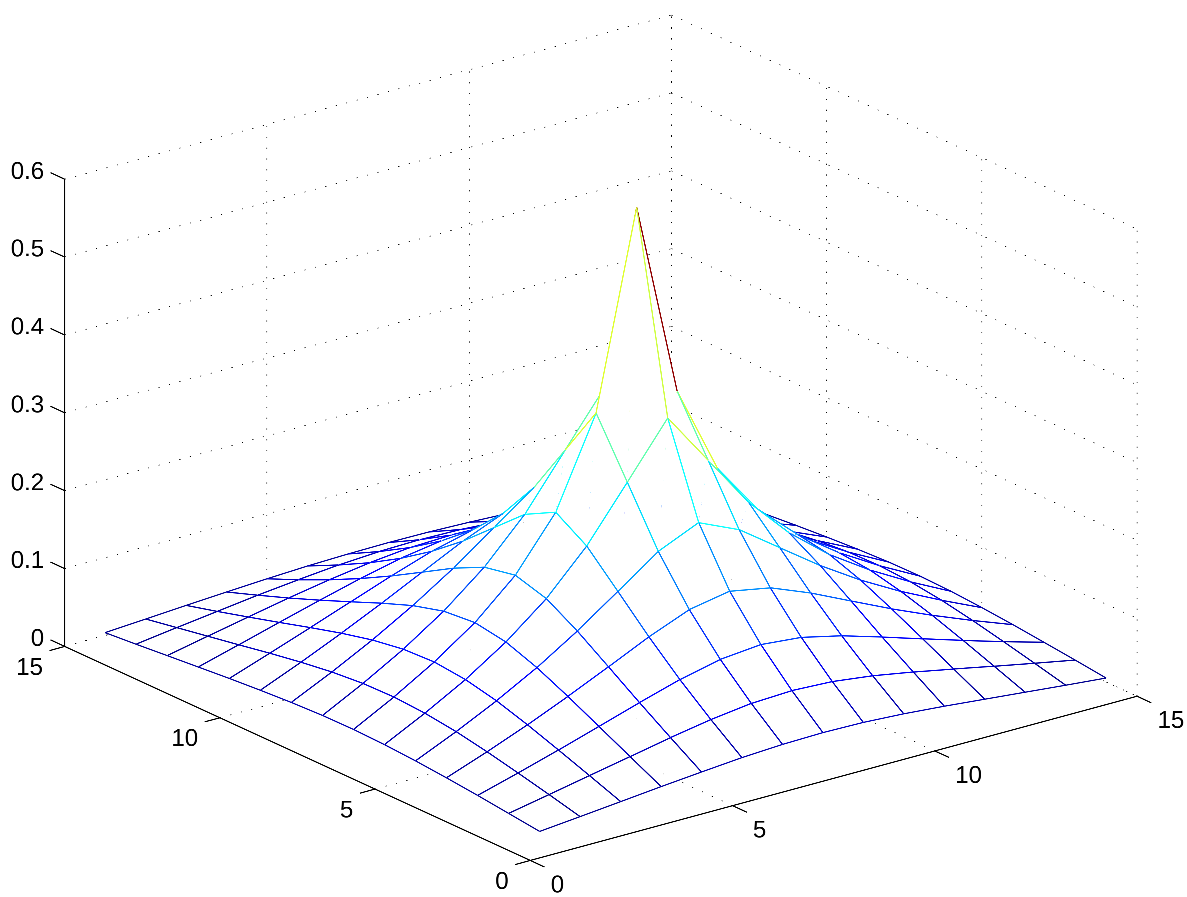
clusters of nodes with (4,. . .) microprocessors, shared memory within the node, distributed across nodes

Problems

- How to use efficiently these SMPs
- Programming is difficult: MPI, OpenMP, mixed model ?
- Lack of good development software (compilers, debuggers, etc. . .)

Lack of parallelism is often linked with the mathematical problem, ex: elliptic problems

Leads to $Ax = b$ with A sparse but A^{-1} is full! But there is a decrease of the discrete Green function which can help introducing parallelism (points far away have not much influence)



Main problem

- Find scalable numerical methods

We want to solve with the same efficiency:

  - "small" problems on a small number of processors (10),
  - "large" problems on a very large number of processors (1000)

Elapsed time must be constant when we rise proportionally the problem size and the number of processors

This is a tough problem: most known algorithms are not scalable

Solving linear systems

If A is symmetric positive definite (SPD) we use the preconditioned conjugate gradient (PCG)

x^0 given, $r^0 = b - Ax^0$. For $k = 0, 1, \ldots$

$$Mz^k = r^k,$$

$$\beta_k = \frac{(z^k, Mz^k)}{(z^{k-1}, Mz^{k-1})}, \quad \beta_0 = 0,$$

$$p_k = z^k + \beta_k p^{k-1},$$

$$\gamma_k = \frac{(z^k, Mz^k)}{(p^k, Ap^k)},$$

$$x^{k+1} = x^k + \gamma_k p^k,$$

$$r^{k+1} = r^k - \gamma_k Ap^k.$$

M is the SPD preconditioner

∘ For PDEs the number of flops for one iteration is proportional to n (depends on the sparsity structure of A and M)

∘ the number of iterations depends on the condition number $\kappa(M^{-1}A)$

∘ for PCG to be scalable we need $\kappa = O(1)$

∘ On parallel computers problem with the scalar products ($n \log n$)

• This algorithm is well suited to vector computing but not to parallel computing (many synchronization points)

• Same problems arise with Krylov methods for non symmetric systems (BiCGstab, GMRES, etc. . .)

• Most known preconditioners give $\kappa = O(n^\delta), \delta > 1$

• Many efficient preconditioners are not naturally parallel, ex: incomplete Cholesky decomposition (recurrences)

To decrease the number of synchronization points, we may use the other form of PCG

x^0 given, for $k = 0, 1, \ldots$

$$Mz^k = r^k (= b - Ax^k),$$

$$\alpha_k = \frac{(z^k, Mz^k)}{(z^k, Az^k)},$$

$$\omega_{k+1} = \frac{1}{1 - \frac{\alpha_k}{\omega_k \alpha_{k-1}} \frac{(z^k, Mz^k)}{(z^{k-1}, Mz^{k-1})}}, \quad \omega_1 = 1,$$

$$x^{k+1} = x^{k-1} + \omega_{k+1}(\alpha_k z^k + x^k - x^{k-1}),$$

$$r^{k+1} = r^{k-1} - \omega_{k+1}(\alpha_k A z^k - r^k + r^{k-1}).$$

However, the number of flops is larger (2 s.p., 1 matvec + $10n$ vs $6n$). Also less stable?

Preconditioners

Suppose A SPD large and sparse

- ◦ M SPD
- ◦ M sparse
- ◦ M easy and cheap to compute
- ◦ $Mz = r$ easy to solve
- ◦ "good" eigenvalue distribution for $M^{-1}A$

- Constructing good preconditioners is more art than science

- ◦ Computing M must be parallel
- ◦ Solving $Mz = r$ must be parallel

Many well known preconditioners are based on direct or "classical" iterative methods:

- Diagonal, based on Jacobi iteration

$$M = D = diag(A)$$

- SSOR, based on successive over relaxation

$$A = D + L + L^T$$

$$M = \frac{1}{\omega(2-\omega)}(D + \omega L)D^{-1}(D + \omega L^T)$$

- Incomplete Cholesky, based on Gaussian elimination

If you do a Cholesky decomposition of A, $A = \tilde{L}\tilde{L}^T$, you get **fill–in**

To obtain an incomplete Cholesky decomposition $M = LD^{-1}L^T$, before computing a column of L, you throw away some fill-in (based on position or value)

There are dependencies in the computation of L and in the solves

PCG for Poisson problem $m \times m$ mesh

m	IC(0)	IC($\epsilon = 0.005$)	IC($\epsilon = 0.001$)
10	16 op=66009 str=100	9 op=44173 str=525	7 op=42573 str=758
20	27 op=446957 str=400	15 op=296841 str=2245	10 op=257393 str=3488
30	38 op=1410785 str=900	21 op=934509 str=5165	13 op=762053 str=8218
40	49 op=3230793 str=1600	26 op=2056749 str=9285	16 op=1673553 str=14948
50	60 op=6176681 str=2500	31 op=3829389 str=14605	19 op=3108893 str=23678
60	71 op=10519049 str=3600	38 op=6747485 str=21125	22 op=5185073 str=34408

PCG for an anisotropic problem

m	IC(0)	IC($\epsilon = 0.005$)	IC($\epsilon = 0.001$)
10	9 op=38373 str=100	6 op=28041 str=434	4 op=20253 str=461
20	14 op=236913 str=400	7 op=134025 str=1864	6 op=119829 str=1975
30	20 op=755081 str=900	9 op=384533 str=4294	8 op=370325 str=4995
40	26 op=1736529 str=1600	10 op=758817 str=7724	8 op=679245 str=9435
50	31 op=3227149 str=2500	12 op=1411905 str=12154	9 op=1204573 str=15275
60	37 op=5533017 str=3600	14 op=2358353 str=17584	10 op=1935861 str=22515

PCG for a discontinuous problem

m	IC(0)	IC($\epsilon = 0.005$)	IC($\epsilon = 0.001$)
10	16 op=66009 str=100	7 op=41229 str=716	5 op=35013 str=898
20	29 op=478233 str=400	10 op=247713 str=3268	7 op=228521 str=4817
30	39 op=1447213 str=900	13 op=747101 str=7951	9 op=710413 str=20541
39	49 op=3070512 str=1521	16 op=1565299 str=13835	10 op=1382077 str=22361
50	61 op=6278389 str=2500	19 op=3072973 str=23229	12 op=2777061 str=38407
59	73 op=10453792 str=3481	22 op=4962385 str=32715	13 op=4244652 str=54841

- Are there any way to introduce more parallelism in IC?

- change of ordering

- modifications of algorithm

Change of ordering

Do an incomplete Cholesky decomposition of

$$A_P = PAP^T$$

Numerical experiments showed that the ordering has some impact on the rate of convergence

- Lichnewsky 1984 (nested dissection)
- Simon 1985
- Duff–Meurant 1989 (many numerical experiments)

This effect has been rediscovered over and over by other people since 1989

Theoretical explanation:

- ○ V. Eijkhout 1990
- ○ S. Doi 1990

$$M = LDL^T = A + R$$

Examples of orderings

- ◦ ROW (row)
- ◦ CM (Cuthill–Mc Kee)
- ◦ MIND (Minimum degree)
- ◦ RB (Red–Black)
- ◦ ND (Nested dissection)
- ◦ VDV2 (Van der Vorst)

RB, ND and VDV2 have more parallelism

Poisson problem 30×30 mesh

ordering	nit	nb of fill	nb R	$\|R\|_F^2$
ROW	23	24389	841	142.5
CM	23	16675	841	142.5
MIND	39	7971	1582	467.3
RB	38	12853	1681	525.5
ND	25	15228	1012	157.1
VDV2	20	17413	841	140.7

nb of elements in L : 2639

Anisotropic problem $a = 100, b = 1$, 30×30 mesh

ordering	nit	nb of fill	nb R	$\|R\|_F^2$
ROW	9	24389	841	$0.12\ 10^4$
CM	9	16675	841	$0.12\ 10^4$
MIND	48	7971	1582	$0.18\ 10^7$
RB	47	12853	1681	$0.21\ 10^7$
ND	26	15228	1012	$0.43\ 10^6$
VDV2	9	17413	841	$0.11\ 10^4$

These results are explained by the Doi and Eijkhout theory

Results are different if we keep some fill

Exemple : Poisson with one level of fill

ordering	nit	nb of fill	nb R	nb L	$\|R\|_F^2$
ROW	17	24389	1653	3481	24.7
CM	17	16675	1653	3481	24.7
MIND	23	7971	2467	4222	38.81
RB	16	12853	2016	4321	16.47
ND	19	15228	2187	3652	35.34
VDV2	17	17413	1651	3481	25.20

Anisotropic problem with one level of fill

ordering	nit	nb of fill	nb R	nb L	$\|R\|_F^2$
ROW	8	24389	1653	3481	823
CM	8	16675	1653	3481	844
MIND	27	7971	2467	4222	$0.22\ 10^6$
RB	8	12853	2016	4321	806
ND	23	15228	2187	3652	$0.18\ 10^6$
VDV2	8	17413	1651	3481	795

Why are the results different (and better) with RB when we keep some fill?

Let us look at the number and the absolute values of the fills

If

$$A = LDL^T,$$

$$\|A\|_F = \left(\sum_{i,j} a_{i,j}^2 \right)^{1/2}.$$

$$\|A\|_F^2 = trace(A^T A) = trace(AA^T).$$

Then

$$\|L\sqrt{D}\|_F^2 = trace(LDL^T) = trace(A)$$

If $A_P = PAP^T$ and

$$A_P = L_P D_P^{-1} L_P^T.$$

then

$$\|PAP^T\|_F = \|A\|_F,$$

and

$$\|L_P \sqrt{D_P^{-1}}\|_F = \|L\sqrt{D^{-1}}\|_F = \sqrt{trace(A)}, \quad \forall P$$

If there are a few fills, they are large

With RB there are only a few fills, their absolute values are larger than with ROW

Changes of algorithm

- ◦ Pothen and Hysom
- ◦ Magolu Monga Made and Van der Vorst

Pothen's ILU(k) algorithm:

- partition the graph of A (subdomains) 1 subgraph=1 processor

- for each subgraph, order interior nodes first, then boundary nodes

- form the subdomain graph, color the vertices

- factor the interior rows in parallel

- receive information from lower-numbered adjacent subdomains

- factor boundary rows enforcing the subdomain graph constraint

$$G_S(L + U - I) = G_S(A)$$

- send information to higher-numbered subdomains

We would like to directly compute M^{-1} and then

$$z^k = M^{-1} r^k,$$

parallel matrix vector products

- Approximate inverses:
  - Huckle et Grote (1994)
  - Gould et Scott (1995)
  - Chow et Saad (1994–1995)
  - Benzi (1995–1996)

We want $M^{-1}A$ "to look like" I

We compute $C = M^{-1}$ to minimize

$$\|AC - I\| \text{ or } \|CA - I\|$$

Generally one takes the Frobenius norm:

$$\|AC - I\|_F^2 = \sum_{k=1}^{n} \|(AC - I)e_k\|^2,$$

e_k k–th column of I, we minimize the l_2 norms

$$\|Ac_k - e_k\|, \; k = 1, \ldots, n$$

n independent least squares problems (parallel)

Generally A^{-1} is dense, how to choose the sparsity structure of c_k?

Let $\hat{c}_k$ be the vector of the non zero elements of c_k

Let $\hat{A}_k$ be the matrix whose columns are those of A with indices $G_k = \{j | (c_k)_j \neq 0\}$ and whose rows i are such that $\exists a_{i,j} \neq 0, j \in G_k$

$$\min_{\hat{c}_k} \|\hat{A}_k \hat{c}_k - \hat{e}_k\|, \quad k = 1, \ldots, n$$

These small least squares problems are solved with QR

Structure of c_k

Huckle & Grote start from G_k^0 (diagonal or same structure as A)

One solves the problem and augment G_k iteratively

At iteration p, consider the residual $r = Ac_k^p - e_k$. We want to decrease $\|r\|$

Let $\mathcal{L} = \{j | (r)_j \neq 0\}$ and $\forall l \in \mathcal{L}$ $\mathcal{N}_l = \{j | a_{l,j} \neq 0, j \notin G_k^p\}$. The candidates are chosen in

$$\cup_{l \in \mathcal{L}} \mathcal{N}_l$$

For j in this set, solve

$$\min_{\mu_j \in \Re} \|r + \mu_j A e_j\| \quad \Longrightarrow \quad \mu_j = -\frac{(r, Ae_j)}{\|Ae_j\|^2}$$

The new residual is

$$\|r\|^2 - \frac{(r, Ae_j)^2}{\|Ae_j\|^2}$$

One chooses indices that give the smallest residuals and iterate the process

This method is denoted as SPAI. Parallel implementation was considered by Deshpande, Grote, Messmer and Sawyer

Gould and Scott improved the choice of the new indices

Chow and Saad iteratively solve $Ac_j = e_j$ (which is as hard as the original problem) with a small number of iterations. They can precondition with the already computed columns

For these methods, there are conditions for C being non singular; remark that C is not symmetric. We can keep symmetry by computing only the lower triangular part, but this is not parallel.

Is C positive definite? One can look for C as KK^T

Benzi, Meyer et Tuma approximate inverse

A SPD

If $Z = [z_1, z_2, \ldots, z_n]$ is a set of conjugate directions for A,

$$Z^T A Z = D$$

D diagonal and $A^{-1} = Z D^{-1} Z^T$

The direction are computed by Gram–Schmidt applied to

$v_1, v_2, \ldots, v_n$

If $V = [v_1, v_2, \ldots, v_n] = I$, Z is upper triangular

1) $z_i^{(0)} = e_i, \quad i = 1, \ldots, n$

2) for $i = 1, \ldots, n \quad d_j^{(i-1)} = (a_i, z_j^{(i-1)}), \quad j = i, \ldots, n$ where

a_i is the ith column of A

if $j \neq n$, $z_j^{(i)} = z_j^{(i-1)} - \left(\frac{d_j^{(i-1)}}{d_i^{(i-1)}} \right) z_i^{(i-1)}, \quad j = i+1, \ldots, n$

3) $z_i = z_i^{(i-1)}, d_i = d_i^{(i-1)}, \quad i = 1, \ldots, n$

To preserve the sparsity structure, fills are thrown away based on position or value (or both)

This method is known as AINV

Benzi, Meyer and Tuma shown that this method is feasible for H–matrices

There exists a “robust” variant SAINV (Benzi, Cullum and Tuma)

This is generalized to non symmetric matrices by considering two sets $Z = [z_1, \ldots, z_n]$ and $W = [w_1, \ldots, w_n]$ such that

$$W^T A Z = D$$

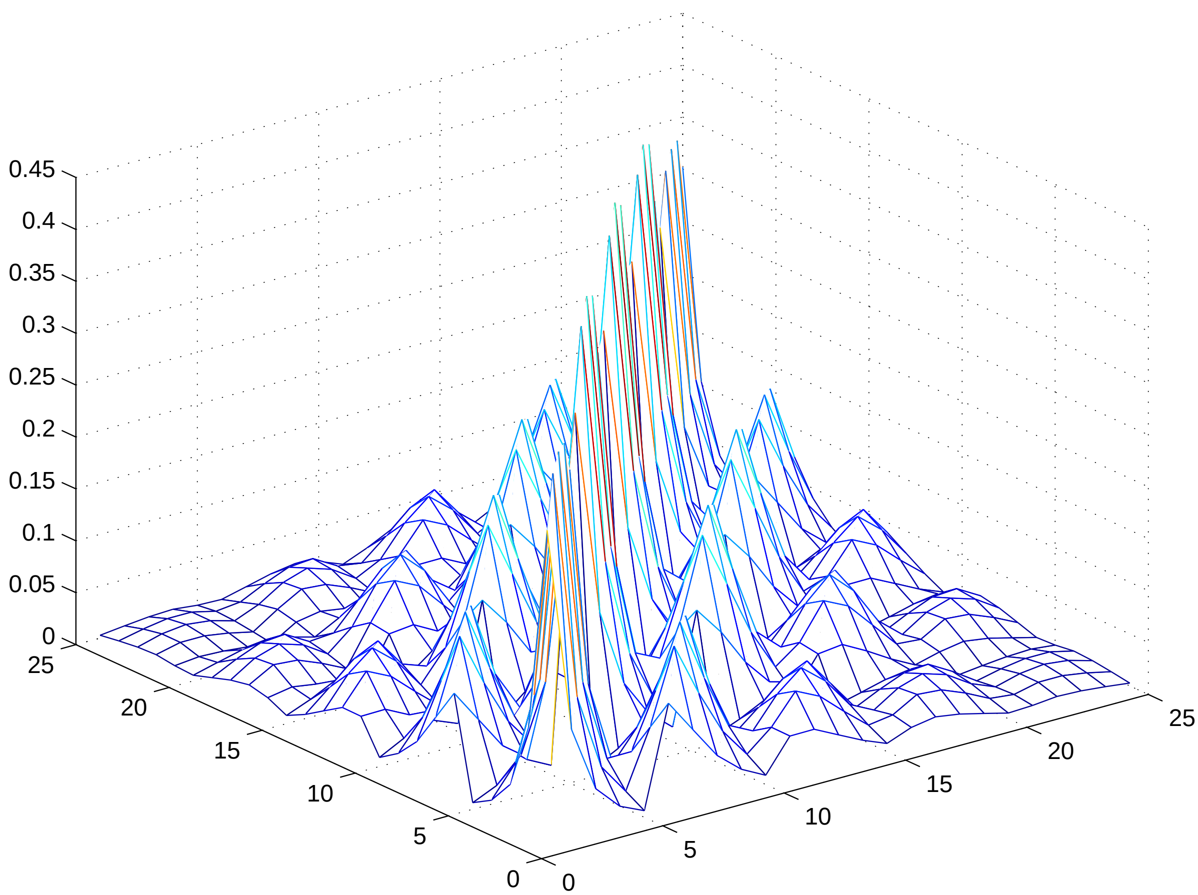
Pb : Poisson, L-shaped region, mixed b.c.

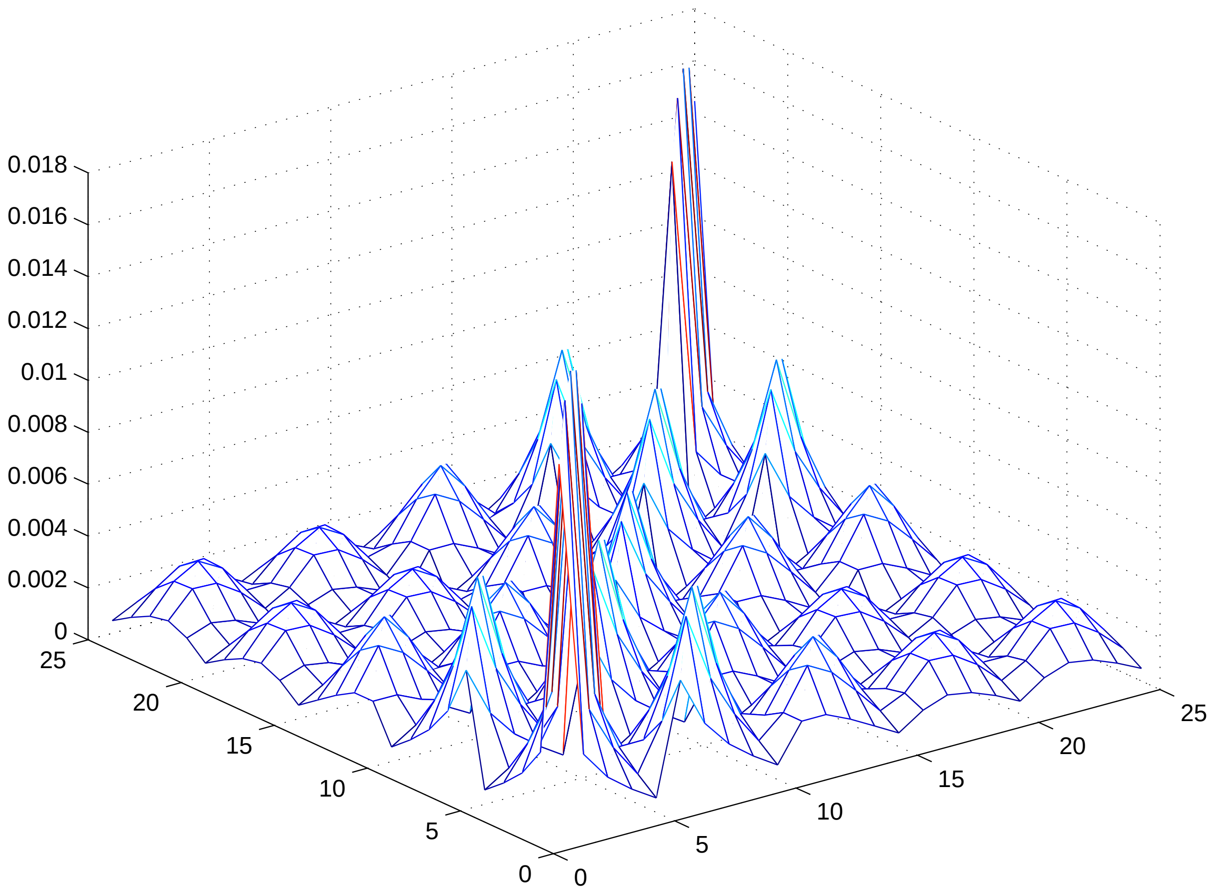
Comparison between IC and AINV (Benzi)

IC			AINV		
fill	nb. iter	time	fill	nb. iter	time
675	87	0.33	743	76	0.32
897	53	0.18	780	74	0.32
912	51	0.18	1135	54	0.26
1204	38	0.14	1208	47	0.18
1439	32	0.14	1300	40	0.21
1565	24	0.10	3654	22	0.14

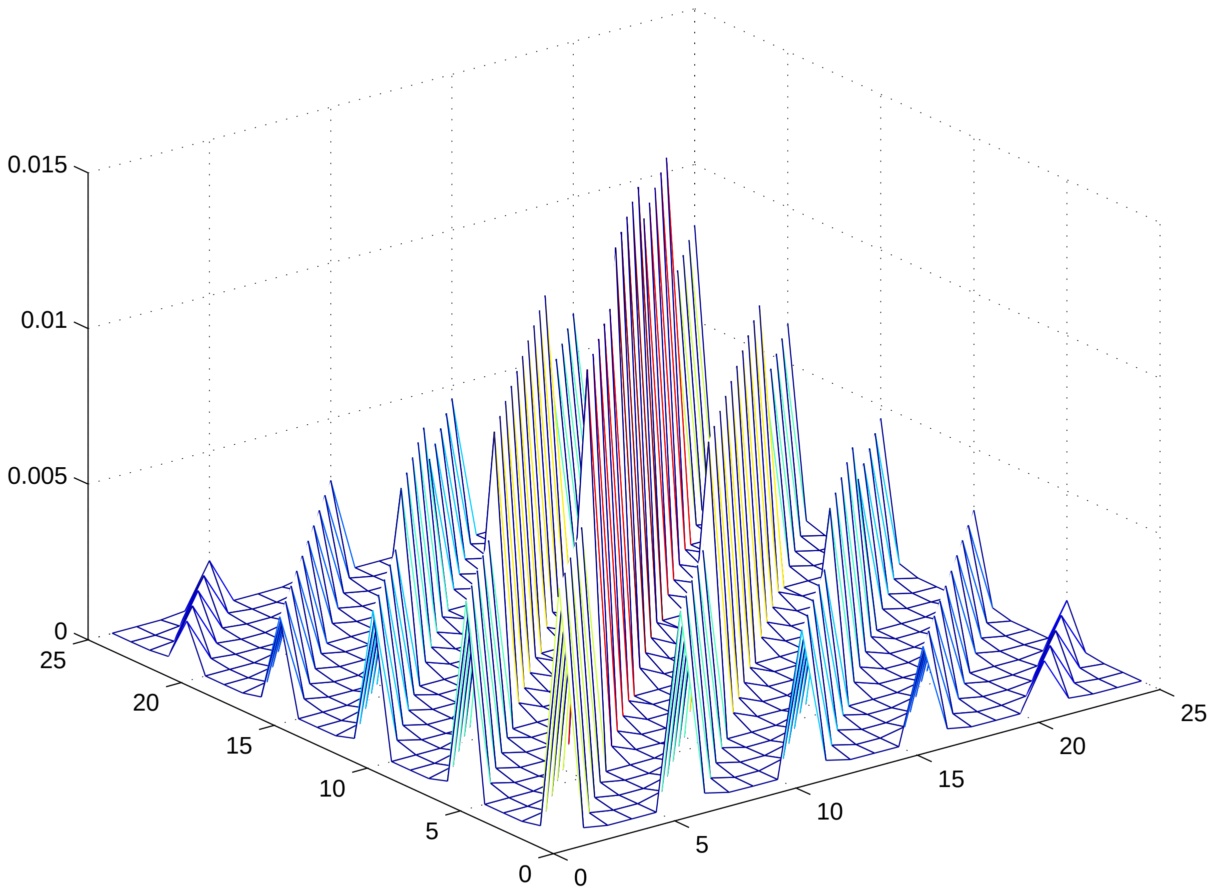
Comparison between SPAI and AINV (Benzi)

Matrix	SPAI			AINV		
	Its	init	t its	Its	init	t its
3DCD	40	10.63	0.111	25	1.885	0.068
ALE3D	45	30.79	0.088	43	1.446	0.094
ORSREG1	40	3.309	0.033	33	0.550	0.031
SHERMAN1	62	0.878	0.029	43	0.201	0.021
PORES3	111	0.941	0.044	75	0.127	0.038
WATT2	377	2.590	0.384	111	0.505	0.116


0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
25
20
15
10
5
0
0
5
10
15
20
25



0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
25
20
15
10
5
0
0
5
10
15
20
25


0.015
0.01
0.005
0
25
20
15
10
5
0
0
5
10
15
20
25

Polynomial preconditioners

$$M^{-1} = P_k(A) = \sum_{i=0}^{k} \alpha_i A^i$$

P_k polynomial of degree k

Eigenvalues of $M^{-1}A$ are $P_k(\lambda_i)\lambda_i$

We ask for $P_k(\lambda)\lambda$ being close to 1 on $[\lambda_{min}, \lambda_{max}] \subset [a, b]$

Neumann series

$$A = D - L - L^T$$

$$A = D^{1/2}(I - D^{-1/2}(L + L^T)D^{-1/2})D^{1/2}$$

$$A^{-1} = D^{-1/2}(I - D^{-1/2}(L + L^T)D^{-1/2})^{-1}D^{-1/2}$$

$$\rho(I - D^{-1}A) = \rho(D^{-1}(L + L^T)) < 1$$

We take

$$M^{-1} = D^{-1/2}[I + D^{-1/2}(L + L^T)D^{-1/2}]D^{-1/2}$$

$$= D^{-1} + D^{-1}(L + L^T)D^{-1}$$

or more terms (odd nb to add to D^{-1})

MINMAX preconditioner

(Johnson, Michelli & Paul)

$$q_{k+1}(\lambda) = p_k(\lambda)\lambda$$

$\mathcal{Q}_k = \{$polynomials of degree $k,$ positive, being 0 in 0 $\}$

Eigenvalues of A in $[a, b]$, we want to minimize over $\mathcal{Q}_k$

$$cond(q) = \frac{sup_{\lambda \in [a,b]} q_{k+1}(\lambda)}{inf_{\lambda \in [a,b]} q_{k+1}(\lambda)}$$

Solution:

$$q_{k+1}(\lambda) = 1 - \frac{T_{k+1}(\mu(\lambda))}{T_{k+1}(\mu(0))}$$

where $\mu(\lambda) = \frac{2\lambda - b - a}{b - a}$ and T_k Chebychev pol

Least squares

(Saad)

Find a polynomial s, to minimize

$$\int_a^b (1 - \lambda s(\lambda))^2 w(\lambda) \; d\lambda$$

w is a weight. Usual choice:

$$w(\lambda) = (b - \lambda)^\alpha (\lambda - a)^\beta$$

$\alpha \geq \beta \geq -1/2$: Jacobi polynomials

In practice $\alpha = \beta = \frac{-1}{2}$ (Chebychev) or $\alpha = \beta = 0$ (Legendre)

Drawbacks:

○ we need to know a and b

○ Evaluating polynomials of high degree ($\geq$ 10 or 20) is numerically difficult (32 bits) – instability of Horner's scheme

• Solution: use 3 term recurrences (orthogonal polynomials)

Example for Minmax, polynomial p_k is such that

$$p_k(\lambda) = \frac{4}{a-b}\frac{c_k}{c_{k+1}} + 2\mu(\lambda)\frac{c_k}{c_{k+1}}p_{k-1}(\lambda) - \frac{c_{k-1}}{c_{k+1}}p_{k-2}(\lambda),$$

$$p_0(\lambda) = \frac{2}{a+b}, \quad p_1(\lambda) = \frac{8(a+b-\lambda)}{a^2+b^2+6ab}$$

$$c_k = T_k(\mu(0))$$

Problem: the cost is higher

• Polynomial preconditioners are not very efficient on difficult problems

1138-bus

$x^0 = 0$, b random, $\varepsilon = 10^{-10}$

prec	nb it	flops	str
diag	1120	$2.57\ 10^7$	1138
ic	163	$5.80\ 10^6$	2596+
ssor	553	$2.18\ 10^7$	0
lev 1	163	$5.80\ 10^6$	2596+
lev 2	77	$3.13\ 10^6$	3877+
lev 3	54	$2.45\ 10^6$	5025+
lev 4	40	$1.99\ 10^6$	6168+
ch 0.1	98	$3.57\ 10^6$	2807+
ch 0.05	79	$3.05\ 10^6$	3347+
ch 0.01	43	$1.96\ 10^6$	5104+
ch 0.005	36	$1.78\ 10^6$	6085+
ai 0.1	111	$5.09\ 10^6$	5725+
ai 0.05	85	$5.19\ 10^6$	9525+
ai 0.01	46	$6.92\ 10^6$	31874+
pol ls 1	818	$3.95\ 10^7$	0
pol ls 2	580	$4.27\ 10^7$	0

Anisotropic problem, $m = 40$

$x^0 = 0$, b random, $\varepsilon = 10^{-10}$

prec	nb it	flops	str
diag	288	$1.05\ 10^7$	1600
ic	26	$1.52\ 10^6$	4720+
ssor	97	$6.15\ 10^6$	0
lev 1	26	$1.52\ 10^6$	4720+
lev 2	10	$0.65\ 10^6$	6241+
lev 3	10	$0.71\ 10^6$	7723+
lev 4	10	$0.74\ 10^6$	8501+
ch 0.1	30	$1.57\ 10^6$	3160+
ch 0.05	30	$1.57\ 10^6$	3160+
ch 0.01	30	$1.57\ 10^6$	3160+
ch 0.005	10	$0.64\ 10^6$	6124+
ai 0.1	31	$3.68\ 10^6$	20520+
ai 0.05	30	$4.05\ 10^6$	24460+
ai 0.01	30	$4.76\ 10^6$	30560+
tw	161	$8.15\ 10^6$	7840+
pol ls 1	162	$1.31\ 10^7$	0
pol ls 2	113	$1.41\ 10^7$	0

Discontinuous problem, $m = 40$

$x^0 = 0$, b random, $\varepsilon = 10^{-10}$

prec	nb it	flops	str
diag	168	$6.13\ 10^6$	1600
ic	54	$3.16\ 10^6$	4720+
ssor	62	$3.93\ 10^6$	0
lev 1	54	$3.16\ 10^6$	4720+
lev 2	33	$2.13\ 10^6$	6241+
lev 3	27	$1.91\ 10^6$	7723+
lev 4	29	$2.14\ 10^6$	8501+
ch 0.1	41	$2.49\ 10^6$	5227+
ch 0.05	31	$2.00\ 10^6$	6196+
ch 0.01	19	$1.52\ 10^6$	10097+
ch 0.005	16	$1.46\ 10^6$	12878+
ai 0.1	51	$4.39\ 10^6$	12430+
ai 0.05	36	$5.21\ 10^6$	27026+
ai 0.01	18	$9.50\ 10^6$	122907+
tw	90	$4.55\ 10^6$	7840+
pol ls 1	96	$7.75\ 10^6$	0
pol ls 2	67	$8.37\ 10^6$	0

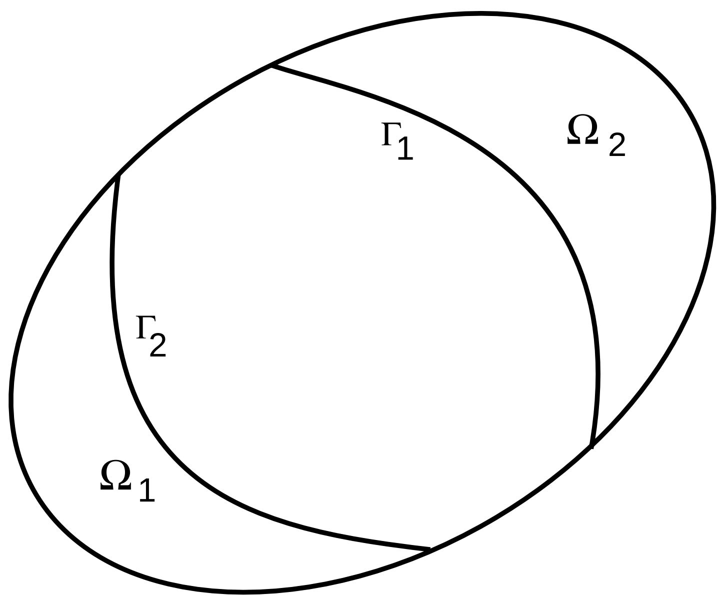
Introduction to DD

- Domain decomposition is a “divide and conquer” technique
- Natural framework to introduce parallelism in the solution of PDE’s
- General scheme:
    - Decompose the problems into subproblems
    - Solve the subproblems in parallel
    - Glue the (sub)solutions together to get the global solution
- The modern view on DD is to construct preconditioners for Krylov iterative methods for solving linear systems

- There are hundreds of variants of DD preconditioners
- Two main classes
  - methods with overlapping (Schwarz)
  - methods without overlapping (interface problems)
- Methods differ also on other issues:
  - exact or inexact solvers for subproblems
  - solve a reduced system or the global system
  - etc...
- Most DD methods for PDEs rely on mesh partitioning

The classical Schwarz alternating method

- Solve a 2^{nd} order elliptic PDE in a bounded 2D domain Ω
- The domain Ω is split into two overlapping subdomains Ω_1 and Ω_2
- Γ_i, $i = 1, 2$, is the part of the boundary of Ω_i enclosed in Ω



◦ Guess a value for the unknowns on the inner boundary Γ_1

◦ Solve the problem exactly in Ω_1

◦ Use the computed values on the inner boundary Γ_2 to solve exactly in Ω_2

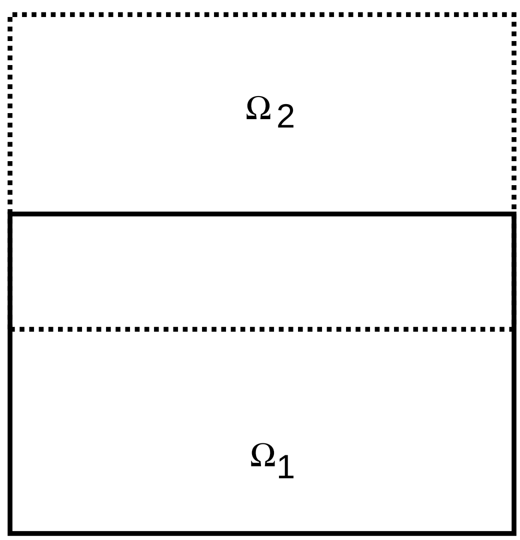
◦ Repeat the process until convergence

Convergence was studied on the continuous pb by P.L. Lions

• Solve a 2^{nd} order elliptic equation in a rectangle using a 5 point FD scheme with the natural (rowwise) ordering

$$A = \begin{pmatrix} D_1 & -B_2^T & & & \\ -B_2 & D_2 & -B_3^T & & \\ & \ddots & \ddots & \ddots & \\ & & -B_{m-1} & D_{m-1} & -B_m^T \\ & & & -B_m & D_m \end{pmatrix}.$$

Suppose the mesh is partitioned as



• The matrix $A^{(1)}$ corresponding to Ω_1 is

$$A^{(1)} = \begin{pmatrix} D_1 & -B_2^T & & & \\ -B_2 & D_2 & -B_3^T & & \\ & \ddots & \ddots & \ddots & \\ & & -B_{p-2} & D_{p-2} & -B_{p-1}^T \\ & & & -B_{p-1} & D_{p-1} \end{pmatrix},$$

• The matrix $A^{(2)}$ corresponding to Ω_2 is

$$A^{(2)} = \begin{pmatrix} D_{l+1} & -B_{l+2}^T & & & \\ -B_{l+2} & D_{l+2} & -B_{l+3}^T & & \\ & \ddots & \ddots & \ddots & \\ & & -B_{m-1} & D_{m-1} & -B_m^T \\ & & & -B_m & D_m \end{pmatrix}.$$

• Let us denote the matrix A in block form as

$$A = \begin{pmatrix} A^{(1)} & A^{(1,2)} \\ X & X \end{pmatrix} \text{ and } A = \begin{pmatrix} Y & Y \\ A^{(2,1)} & A^{(2)} \end{pmatrix},$$

and let b_1 and b_2 be the restrictions of the right hand side b to Ω_1 and Ω_2

• Note that $A^{(1,2)}$ has only one non–zero block in the left lower corner and $A^{(2,1)}$ is zero except for the upper right block

- We denote by x_1 and x_2 the unknowns in Ω_1 and Ω_2

- We extend the vectors x_1 and x_2 to Ω by completing with the components of the previous iterate

- The Schwarz alternating method is

$$A^{(1)} x_1^{2k} = b_1 + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ B_p^T (x_2^{2k-1})_p \end{pmatrix}, \quad A^{(2)} x_2^{2k+1} = b_2 + \begin{pmatrix} B_{l+1} (x_1^{2k})_l \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

$\Longrightarrow$

$$\begin{aligned} x_1^{2k} &= x_1^{2k-1} + (A^{(1)})^{-1} (b_1 - A^{(1)} x_1^{2k-1} - A^{(1,2)} x_{1,2}^{2k-1}), \\ x_2^{2k+1} &= x_2^{2k} + (A^{(2)})^{-1} (b_2 - A^{(2)} x_2^{2k} - A^{(2,1)} x_{2,1}^{2k}). \end{aligned}$$

$$\begin{aligned} x^{2k} &= x^{2k-1} + \begin{pmatrix} (A^{(1)})^{-1} & 0 \\ 0 & 0 \end{pmatrix} (b - A x^{2k-1}), \\ x^{2k+1} &= x^{2k} + \begin{pmatrix} 0 & 0 \\ 0 & (A^{(2)})^{-1} \end{pmatrix} (b - A x^{2k}). \end{aligned}$$

By eliminating x^{2k} we obtain

$$x^{2k+1} = x^{2k-1} + [\begin{pmatrix} (A^{(1)})^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & (A^{(2)})^{-1} \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & (A^{(2)})^{-1} \end{pmatrix} A \begin{pmatrix} (A^{(1)})^{-1} & 0 \\ 0 & 0 \end{pmatrix}] r^{2k-1},$$

$$r^{2k-1} = b - Ax^{2k-1}.$$

• The Schwarz alternating method is nothing else than a preconditioned Richardson iteration

• This method can also be written with another notation

∘ We introduce restriction operators R_1 and R_2

$$x_1^k = R_1 x^k, \quad x_2^k = R_2 x^k.$$

R_1 is simply $\begin{pmatrix} I_{p-1} & 0 \end{pmatrix}$ and $R_2 = \begin{pmatrix} 0 & I_{m-l+1} \end{pmatrix}$

$$A^{(1)} = R_1 A R_1^T, \quad A^{(2)} = R_2 A R_2^T.$$

- The first step of the iteration is:
  - ◦ restriction by R_1
  - ◦ apply the inverse of $R_1 A R_1^T$
  - ◦ extension of the result by R_1^T

$$x^{2k} = x^{2k-1} + R_1^T (R_1 A R_1^T)^{-1} R_1 (b - A x^{2k-1}).$$

- The second step is

$$x^{2k+1} = x^{2k} + R_2^T (R_2 A R_2^T)^{-1} R_2 (b - A x^{2k}).$$

Proposition

The matrix $P_i = R_i^T (R_i A R_i^T)^{-1} R_i A, i = 1, 2$ is an orthogonal projection in the scalar product defined by A

If ε^k is the error, we have

$$\varepsilon^{2k} = (I - P_1)\varepsilon^{2k-1}, \quad \varepsilon^{2k+1} = (I - P_2)\varepsilon^{2k}.$$

Other boundary conditions

• A way to reduce the overlap while maintaining a good convergence rate is to use other inner boundary conditions than Dirichlet for the subproblems (W.P. Tang)

• WPT proposed using inner mixed boundary conditions like continuity of

$$\omega u + (1 - \omega) \frac{\partial u}{\partial n}.$$

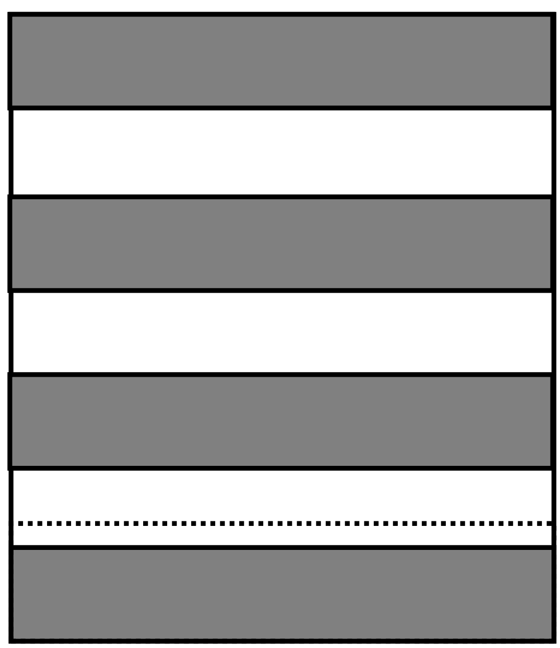
• Numerical results show that this can substantially improve the rate of convergence for small overlaps

Parallelizing Schwarz methods

• There is no parallelism in the Schwarz alternating method

• To get a parallel algorithm we use a coloring of the subdomains such that a subdomain of one color is only connected to subdomains of other colors

• For strips a red–black ordering is used, every other strip is black, and red strips alternate with black strips



The additive Schwarz method

• The alternating Schwarz method can be considered as a kind of `Gauss-Seidel` algorithm

• A way to get a parallel algorithm is to use instead a block `Jacobi`–like method
This is known as the Additive Schwarz method, (Dryja and Widlund)

$$M^{-1} = \sum_i R_i^T (R_i A R_i^T)^{-1} R_i,$$

where the summation is over the number of overlapping subdomains

• More generally, one can replace the exact solves for each subdomain by approximations and define

$$M^{-1} = \sum_i R_i^T M_i^{-1} R_i.$$

Adding a coarse mesh correction

• The rate of convergence of the multiplicative or additive Schwarz methods depends on the number of subdomains

• To improve on this we add a coarse grid correction

• The coarse grid corresponds to the interfaces in the partitioning

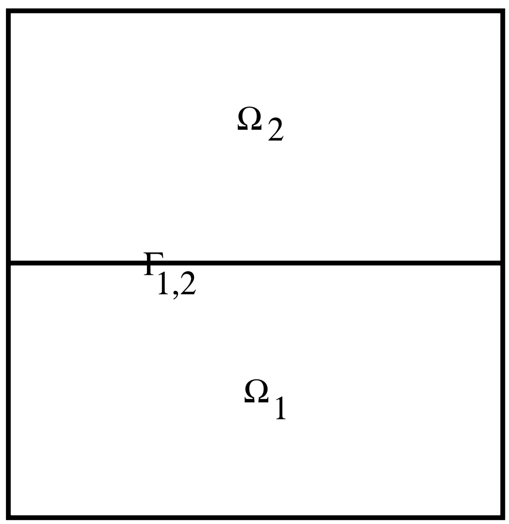
$$M^{-1} = \sum_i R_i^T (R_i A R_i^T)^{-1} R_i + R_0^T A_C^{-1} R_0,$$

• The coarse grid operator may be chosen as a Galerkin approximation $A_C = R_0 A R_0^T$

• If the extent of overlap is kept proportional to the "sizes" of the subdomains the number of iterations is independent of n and of the number of subdomains

Algebraic domain decomposition methods without overlapping

- We consider a square domain Ω decomposed into two subdomains
- An elliptic second order PDE in a rectangle discretized by FD
- Let Ω_1 and Ω_2 be the two subdomains and $\Gamma_{1,2}$ the interface which is a mesh line



- We denote by m_1 (resp. m_2) the number of mesh lines in Ω_1 (resp. Ω_2), each mesh line having m mesh points ($m = m_1 + m_2 + 1$)

• We renumber the unknowns in Ω

Let x_1 (resp. x_2) be the vector of unknowns in Ω_1 (resp. in Ω_2) and $x_{1,2}$ be the vector of the unknowns on the interface

$$\begin{pmatrix} A_1 & 0 & E_1 \\ 0 & A_2 & E_2 \\ E_1^T & E_2^T & A_{12} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_{1,2} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_{1,2} \end{pmatrix}.$$

$$E_1 = (0\ 0\ \ldots\ 0\ E_1^{m_1})^T, \qquad E_2 = (E_2^1\ 0\ \ldots\ 0)^T,$$

where $E_1^{m_1}$ and E_2^1 are diagonal matrices

• Most algebraic DD methods are based on block Gaussian elimination (or approximate block Gaussian factorization) of the matrix

• Basically, we have two possibilities depending on the fact that we can or cannot (or do not want to) solve linear systems corresponding to subproblems like

$$\begin{cases} A_1 y_1 = c_1 \\ \\ A_2 y_2 = c_2 \end{cases}$$

"exactly" with a direct method (or with a fast solver)

Exact solvers for the subdomains

• We eliminate the unknowns x_1 and x_2 in the subdomains

This gives a reduced system for the interface unknowns

$$Sx_{1,2} = \overline{b_{1,2}},$$

with

$$\overline{b_{1,2}} = b_{1,2} - E_1^T A_1^{-1} b_1 - E_2^T A_2^{-1} b_2$$

and

$$S = A_{12} - E_1^T A_1^{-1} E_1 - E_2^T A_2^{-1} E_2.$$

The matrix S is the Schur complement of A_{12} in A

• Constructing and factoring S is costly

• A more economical solution is to solve the reduced system with matrix S on the interface with an iterative method

Theorem

For the Poisson model problem the condition number of the Schur complement is

$$\kappa(S) = O(\frac{1}{h}).$$

- The product, Sp can be computed easily as

$$Sp = A_{1,2}p - E_1^T A_1^{-1} E_1 p - E_2^T A_2^{-1} E_2 p,$$

p being a vector defined on the interface

$$E_1 p = (0 \ \ldots 0 \ E_1^{m_1})^T p = (0 \ \ldots 0 \ E_1^{m_1} p)^T,$$

$$E_2 p = (E_2^1 \ 0 \ \ldots 0 \ )^T p = (E_2^1 p \ 0 \ \ldots 0)^T.$$

Then $w^1 = A_1^{-1} E_1 p$ is computed by solving

$$A_1 w^1 = E_1 p,$$

This is solving a linear system corresponding to a problem in Ω_1

• Note that only the last block of the right hand side is different from 0 and because we only need $E_1^T w^1$, the last block $w^1_{m_1}$ of the solution w^1 is what we must compute

• Similarly, $w^2 = A_2^{-1} E_2 p$ is computed by solving

$$A_2 w^2 = E_2 p,$$

a problem in Ω_2

Finally, we have

$$Sp = A_{1,2} p - w^1_{m_1} - w^2_1.$$

• To improve the convergence rate of CG on the reduced system, a preconditioner M is needed

• The main problem is:

Find an approximation of the Schur complement S

Approximate solvers for the subdomains

• Let us choose M in the form

$$M = L \begin{pmatrix} M_1^{-1} & & \\ & M_2^{1} & \\ & & M_{1,2}^{-1} \end{pmatrix} L^T,$$

where M_1 (resp. M_2) is of the same order as A_1 (resp. A_2) and $M_{1,2}$ is of the same order as $A_{1,2}$. L is block lower triangular

$$L = \begin{pmatrix} M_1 & & \\ 0 & M_2 & \\ E_1^T & E_2^T & M_{1,2} \end{pmatrix}$$

• At each PCG iteration, we must solve a linear system like

$$Mz = M \begin{pmatrix} z_1 \\ z_2 \\ z_{1,2} \end{pmatrix} = r = \begin{pmatrix} r_1 \\ r_2 \\ r_{1,2} \end{pmatrix}.$$

• This is done by first solving $Ly = r$, where the first parallel two steps are

$$M_1 y_1 = r_1, \quad M_2 y_2 = r_2.$$

• Finally, we solve for the interface

$$M_{1,2}y_{1,2} = r_{1,2} - E_1^T y_1 - E_2^T y_2.$$

• To obtain the solution, we have a backward solve step as

$$\begin{pmatrix} I & 0 & M_1^{-1}E_1 \\ & I & M_2^{-1}E_2 \\ & & I \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_{1,2} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_{1,2} \end{pmatrix}.$$

This implies that $z_{1,2} = y_{1,2}$ and

$$M_1 w_1 = E_1 z_{1,2}, \quad z_1 = y_1 - w_1,$$

$$M_2 w_2 = E_2 z_{1,2}, \quad z_2 = y_2 - w_2.$$

• How to choose the approximations M_1, M_2 and $M_{1,2}$?

$$M = \begin{pmatrix} M_1 & 0 & E_1 \\ 0 & M_2 & E_2 \\ E_1^T & E_1^T & M_{1,2}^* \end{pmatrix},$$

where

$$M_{1,2}^* = M_{1,2} + E_1^T M_1^{-1} E_1 + E_2^T M_2^{-1} E_2.$$

• We would like M to be an approximation of A, it makes sense to choose

$$M_1 \approx A_1, \quad M_2 \approx A_2,$$

and

$$M_{1,2}^* \approx A_{1,2} \Longrightarrow M_{1,2} \approx A_{12} - E_1^T M_1^{-1} E_1 - E_2^T M_2^{-1} E_2.$$

• We are back to the same problem as before; that is to say, $M_{1,2}$ must be an approximation to the Schur complement S

$$
A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix},
$$

$$
T = Q \Lambda Q^T,
$$

Q being such that $QQ^T = I$ and Λ being a diagonal matrix whose diagonal elements are the eigenvalues of T

In the simple square 2 subdomain case we can compute the eigenvalues of S

Theorem

The spectral decomposition of the Schur complement is

$$S = Q\Theta Q^T,$$

where Θ is a diagonal matrix whose diagonal elements θ_l are given by

$$\theta_l = \lambda_l - \frac{(r_l)_+^{m_1} - (r_l)_-^{m_1}}{(r_l)_+^{m_1+1} - (r_l)_-^{m_1+1}} - \frac{(r_l)_+^{m_2} - (r_l)_-^{m_2}}{(r_l)_+^{m_2+1} - (r_l)_-^{m_2+1}},$$

where $(r_l)_\pm = \frac{\lambda_l \pm \sqrt{\lambda_l^2 - 4}}{2}$

• We do not need to explicitly know the eigenvectors Q to compute the eigenvalues

Proposition

Let $\lambda_l = 2 + \sigma_l$ and $\gamma_l = \left(1 + \frac{\sigma_l}{2} - \sqrt{\sigma_l + \frac{\sigma_l^2}{4}}\right)^2$, then

$$\theta_l = \left(\frac{1 + \gamma_l^{m_1+1}}{1 - \gamma_l^{m_1+1}} + \frac{1 + \gamma_l^{m_2+1}}{1 - \gamma_l^{m_2+1}}\right)\sqrt{\sigma_l + \frac{\sigma_l^2}{4}}, \quad \forall l = 1, \ldots, m$$

• Let us now look at the eigenvalues of S when, for a fixed h, the domains Ω_1 and Ω_2 extend to infinity

Theorem

If $\lambda_l > 2$,

$$\theta_l \rightarrow 2\sqrt{\sigma_l + \frac{\sigma_l^2}{4}} \text{ when } \; m_i \rightarrow \infty, \; i = 1, 2.$$

Dryja's preconditioner

Let T_2 be the matrix corresponding to finite difference discretization of the one–dimensional Laplacian

$$T_2 = Q_2 \Sigma_2 Q_2^T,$$

where Σ_2 is the diagonal matrix of the eigenvalues

$$\sigma_i = 2 - 2\cos(i\pi h), \quad i = 1, \ldots, m$$

$$q_{i,j} = \sqrt{\frac{2}{m+1}} \sin(ij\pi h), \quad i, j = 1, \ldots, m.$$

• We define the Dryja's preconditioner M_D as

$$M_D = Q_2 \sqrt{\Sigma_2} Q_2^T.$$

• In a practical way, the action of M_D^{-1} on a vector can be implemented as two one dimensional FFTs and a division by the eigenvalues

Golub and Mayers' preconditioner

- The Golub and Mayers' preconditioner is an improvement upon Dryja's preconditioner

$$M_{GM} = Q_2 \sqrt{\Sigma_2 + \frac{\Sigma_2^2}{4}} Q_2^T.$$

The Neumann–Dirichlet preconditioner

- This preconditioner was introduced by Bjørstad and Widlund

$$\begin{pmatrix} A_1 & 0 & E_1 \\ 0 & A_2 & E_2 \\ E_1^T & E_2^T & A_{1,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_{1,2} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_{1,2} \end{pmatrix},$$

- We can distinguish what in $A_{1,2}$ comes from subdomain Ω_1 and what comes from Ω_2

$$A_{1,2} = A_{1,2}^{(1)} + A_{1,2}^{(2)}.$$

Since we know that

$$S = A_{1,2} - E_1^T A_1^{-1} E_1 - E_2^T A_2^{-1} E_2,$$

we can define

$$S^{(1)} = A_{1,2}^{(1)} - E_1^T A_1^{-1} E_1, \quad S^{(2)} = A_{1,2}^{(2)} - E_2^T A_2^{-1} E_2,$$

and $S = S^{(1)} + S^{(2)}$

- The Neumann–Dirichlet preconditioner is defined as

$$M_{ND} = S^{(1)}.$$

Note, that we could also have chosen $S^{(2)}$ instead of $S^{(1)}$

The Neumann–Neumann preconditioner

- This preconditioner was introduced by Le Tallec

$$M_{NN}^{-1} = \frac{1}{2}\left[(S^{(1)})^{-1} + (S^{(2)})^{-1}\right]$$

Note that we directly define the inverse of the preconditioner as an average of inverses of “local” (to each subdomain) inverses of Schur complements.

All these preconditioners give $\kappa(M^{-1}S) = O(1)$ for the Poisson problem with 2 subdomains

Inexact subdomain solvers

• If we cannot solve exactly for the subproblems, we are not able to use an iterative method with S as we cannot compute the matrix×vector product Sv

• We need a global parallel preconditioner

$$M = L \begin{pmatrix} M_1^{-1} & & & & & & \\ & M_2^{-1} & & & & & \\ & & \ddots & & & & \\ & & & M_k^{-1} & & & \\ & & & & M_{1,2}^{-1} & & \\ & & & & & \ddots & \\ & & & & & & M_{k-1,k}^{-1} \end{pmatrix} L^T$$

$$
L = \begin{pmatrix}
M_1 & & & & & & & & \\
 & M_2 & & & & & & & \\
 & & \ddots & & & & & & \\
 & & & \ddots & & & & & \\
 & & & & M_k & & & & \\
 & & & & & & & & \\
C_1^T & E_2^T & & & & M_{1,2} & & & \\
 & C_2^T & E_3^T & & & H_2 & M_{2,3} & & \\
 & & \ddots & & & & \ddots & \ddots & \\
 & & & C_{k-1}^T & E_k^T & & & H_{k-1} & M_{k-1,k}
\end{pmatrix}
$$

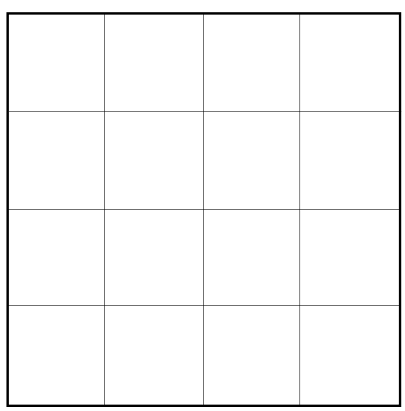
• The matrices M_i can be chosen as for the two subdomains case

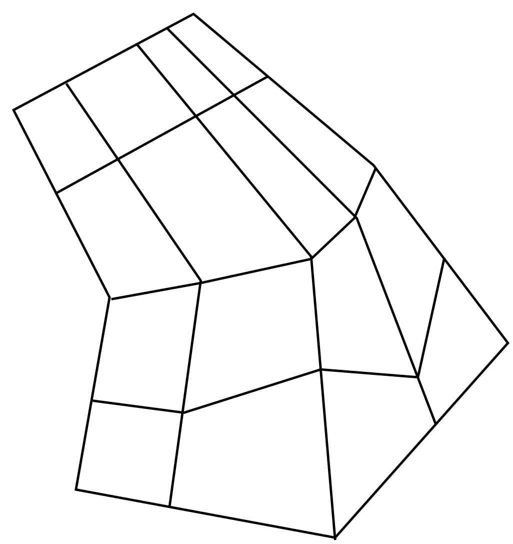
• For matrices $M_{i,i+1}$ and H_i, we have many possible choices

Domain decomposition with boxes

• A domain decomposition with strips can be done for more general domains by finding pseudo–peripheral nodes and constructing the level structure corresponding to one of these nodes

• However, except for very large problems, when partitioning in this way, we cannot use many subdomains. A way to partition with many subdomains is to use so–called boxes





t

With exact solves for the subdomains, variants of the Bramble, Pasciak and Schatz BPS preconditioner can be denoted as

$$M^{-1}v = \sum_{edges} R_{E_i}^T (\alpha_i M_i)^{-1} R_{E_i} v + R_H^T A_H^{-1} R_H v,$$

where R_{E_i} denotes the restriction to the edge E_i and R_H is a weighted restriction onto the coarse mesh, M_i being one of the preconditioners for two subdomain case: either Dryja or Golub–Mayers

Vertex space preconditioners

• A way to improve on BPS is to allow for some coupling between the vertices and the edge nodes

• Some points are considered around each vertex on each of the edges

Let V_k be this set of points. Then the preconditioner is defined as

$$M^{-1}v =$$

$$R_H^T A_H^{-1} R_H v + \sum_{edges} R_{E_i}^T (M_{E_i})^{-1} R_{E_i} v + \sum_{vertices} R_{V_k}^T (M_{V_k})^{-1} R_{V_k} v$$

This includes some coupling between neighboring edges

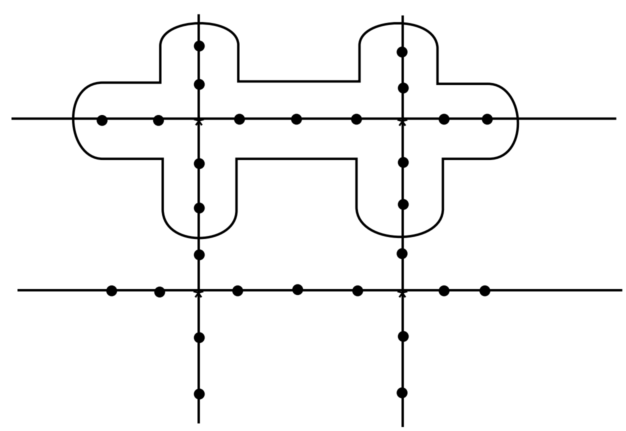
• The edge preconditioner can be chosen as a weighting of Dryja's or Golub–Mayers' preconditioners

• L. Carvalho considered some preconditioners whose spirit is quite close to the vertex space preconditioners

• Because they involve some kind of overlapping between the edge and vertex parts, they are denoted as algebraic additive Schwarz (`AAS`)

• He studied several local block preconditioners for the subdomains and several coarse space preconditioners

• For one of the local preconditioners, the main difference with the vertex space preconditioner is that the edge and the adjacent vertices are considered together



• Another proposal was to consider the complete boundary of one subdomain, to be able to retrieve all the couplings between the edge nodes and the vertices when the interior nodes are eliminated

• It is necessary to add a coarse space component in the algorithm

• A restriction operator R_0 is defined (depending on the choice of the coarse part of the preconditioner)

• The coarse component of the preconditioner is defined as $R_0^T A_0^{-1} R_0$ where A_0 is the Galerkin coarse space operator $A_0 = R_0 S R_0^T$

• Several possibilities were considered:

◦ i) a subdomain–based coarse space where all the boundary points of a subdomain are considered. The coarse space is spanned by vectors which have non–zero components for the points around a subdomain, for all subdomains.

◦ ii) a vertex–based coarse space where the vertices and some few adjacent edge points are considered.

◦ iii) an edge–based coarse space where the points of an edge and the adjacent vertices are considered.

• When combining these coarse space preconditioners with the local parts, a preconditioner for which the condition number is insensitive to the mesh size or the number of subdomains is obtained except for very highly anisotropic problems

Numerical experiments

- 16×16 mesh for each subdomain
- Pb 1: Poisson equation

nb of subd	4×4	8×8	16×16
M_E	13	28	51
M_{VE}	12	22	40
M_S	11	19	32
M_{C-E}	9	11	11
M_{C-VE}	10	12	12
M_{C-S}	10	10	11

• Pb 2: Isotropic discontinuous pb on the Scottish flag, coefficients $1, 10^3, 10^{-3}$

nb of subd	4×4	8×8	16×16
M_{C-E}	11	11	15
M_{C-VE}	12	12	16
M_{C-S}	10	11	14

• Pb 3: Anisotropic and discontinuous pb on the Scottish flag, coefficient 1 in x, same as before in y

nb of subd	4×4	8×8	16×16
M_{C-E}	25	65	103
M_{C-VE}	23	80	141
M_{C-S}	20	43	79

Multilevel preconditioners

• We have seen that it is useful to add a coarse space component to the additive Schwarz preconditioners

• It is relatively easy to generalize these two level methods to a multilevel algorithm

• This is very close to multigrid algorithms, specially to the algebraic multigrid methods

• We consider additive multilevel Schwarz preconditioners

◦ Suppose we have L different levels, each level being decomposed into $N^{(l)}$ subdomains denoted as Ω_i^l

Then, the fully additive Schwarz preconditioner is defined as

$$M^{-1} = \sum_{l=0}^{L} \sum_{i=1}^{N^{(l)}} (R_i^l)^T (A_i^l)^{-1} R_i^l.$$

The index $l = 0$ corresponds to the coarsest grid (eventually one node)

Note that the subdomains Ω_i^l overlap each other as in the one level case

• A particularly simple case is the multilevel diagonal scaling preconditioner

Then, if the coarsest grid has only a single subdomain

$$M^{-1} = (R^0)^T(A^0)^{-1}R^0 + \sum_{l=1}^{L-1}(R^l)^T(D^l)^{-1}R^l + (D^L)^{-1},$$

where D^l is the diagonal of A^l

• A closely related preconditioner was developed by Bramble, Pasciak and Xu (BPX)

∘ In finite element methods with linear approximations, the diagonal elements of the matrix at level l must be of order $(h^l)^{d-2}$ where h is the mesh size and d is the dimension (1, 2 or 3)

The BPX preconditioner is defined as

$$M^{-1} = (R^0)^T(A^0)^{-1}R^0 + \sum_{l=1}^{L-1}(h^l)^{2-d}(R^l)^TR^l + (h^L)^{2-d}I$$

• It has been proved that the BPX is theoretically optimal, the condition number being $O(1)$.

- These additive Schwarz methods can be mixed with multiplicative methods in different ways
- One can define as before fully additive methods which are additive among subdomains and between levels
- Another possibility is to be multiplicative between subdomains on one level and additive between levels
- A third kind of algorithm is being multiplicative between both subdomains and levels
- This is very close to a V–cycle multigrid (without smoothing)