Double sweep preconditioner for Schwarz methods applied to the Helmholtz and Maxwell equations

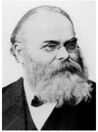
C. Geuzaine and A. Vion

University of Liège, Montefiore Institute, Liège, Belgium

DD22 Lugano – September 2013

Who's who

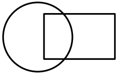
H. von Helmholtz
1821-1894



K. H. Schwarz
1843-1921

J. C. Maxwell
1831-1879

$$(\Delta + k^2)u = 0$$

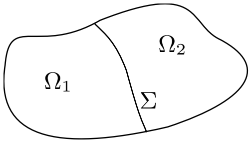


$$(\boldsymbol{\Delta} + k^2)\mathbf{E} = 0$$

For large values of k, FE systems become too large to be solved with usual direct solvers

pictures from Wikipedia

Basic DDM principle (without overlap)



$$\begin{aligned} \mathcal{H}u &= f && \text{in } \Omega = \cup\Omega_i \\ u &= u_D && \text{on } \partial\Omega \end{aligned}$$

(or any other set of BC's)

Equivalent to:

$$\begin{array}{ll|ll} \mathcal{H}(u_1) = f & \text{in } \Omega_1 & \mathcal{H}(u_2) = f & \text{in } \Omega_2 \\ u_1 = u_D & \text{on } \partial\Omega_1 \setminus \Sigma & u_2 = u_D & \text{on } \partial\Omega_2 \setminus \Sigma \end{array}$$

on Σ :

$$\begin{aligned} u_1 &= u_2 \\ \partial_n u_1 &= -\partial_n u_2 \end{aligned}$$

Need to design iterations as u is not known on Σ

Non-overlapping optimized Schwarz Lions (1990), Desprès (1991)

Iteration: solve in all domains (in parallel, direct solver)

$$\left\{\begin{array}{rcll} \mathcal{H}u_i^{(k+1)} & = & f & \text{in } \Omega_i \\ u_i^{(k+1)} & = & u_D & \text{on } \partial\Omega_i \setminus \Sigma \\ \partial_n u_i^{(k+1)} + \mathcal{S}u_i^{(k+1)} & = & -\partial_n u_j^{(k)} + \mathcal{S}u_j^{(k)} & \text{on } \Sigma_{ij} = \Sigma_{ji}. \\ & = & g_{ij}^{(k)} & \end{array}\right.$$

with the update:

$$\begin{array}{rcl} g_{ij}^{(k+1)} & = & -\partial_n u_j^{(k+1)} + \mathcal{S}u_j^{(k+1)} \\ & = & -g_{ji}^{(k)} + 2\mathcal{S}u_j^{(k)} \end{array}$$

New unknowns: the g_{ij} functions defined on Σ_{ij} (2 per interface)

Optimum: $\mathcal{S}$ should be the Dirichlet-to-Neumann (DtN) map:

$$\mathcal{D} : H^{1/2}(\Sigma) \to H^{-1/2}(\Sigma) : \mathcal{D}u_{|\Sigma} = \partial_n u_{|\Sigma}$$

GMRES acceleration and matrix-free implementation

Schwarz iteration operator $\mathcal{A}$:= one step of the algorithm

$$\mathcal{A}: \quad \prod \times H^{-1/2}(\Sigma_i) \to \prod \times H^{-1/2}(\Sigma_i)$$

$$: \quad g^{(k+1)} = \mathcal{A}g^{(k)} + b \qquad (\Rightarrow \text{solve } N \text{ subproblems})$$

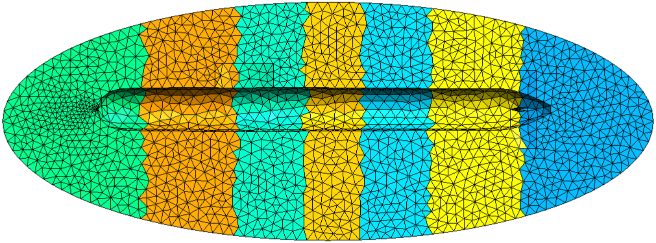
Solve the linear system $\mathcal{F}g = (\mathcal{I} - \mathcal{A})g = b$ with GMRES

"Matrix free": give the application (matrix-vector product) of $\mathcal{F}$ as a routine that solves the subproblems and updates g

Focus on “layered” or “sliced” decompositions

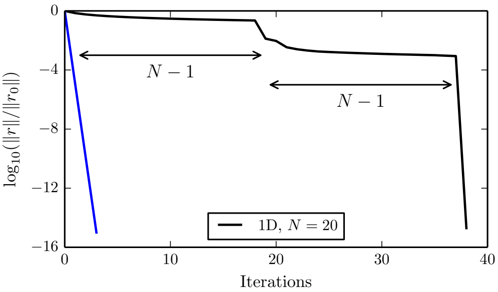
The decomposition should not contain any loop

- easy to generate
- naturally avoid crosspoints



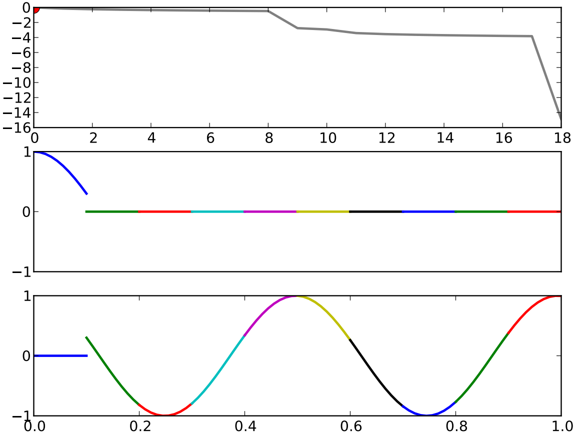
Problem: large plateaus in the convergence curve urge the need for a preconditioner

Even with the best possible transmission condition $\mathcal{S} = \mathcal{D}$, optimized Schwarz requires $\mathcal{O}(N)$ iterations

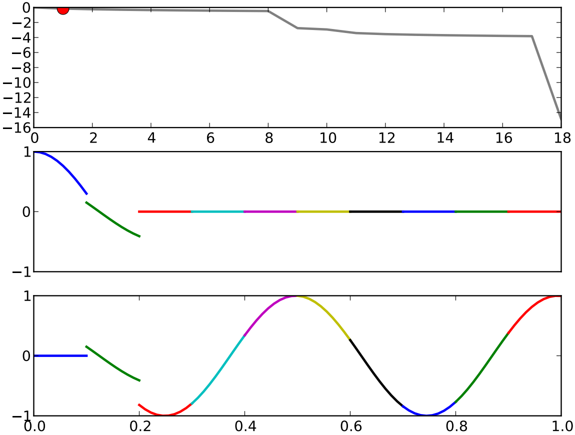


Problem:	This talk:
$N_{it} = \mathcal{O}(N)$	$N_{it} = \mathcal{O}(1)$
$= 2(N-1)$	Fast and smooth

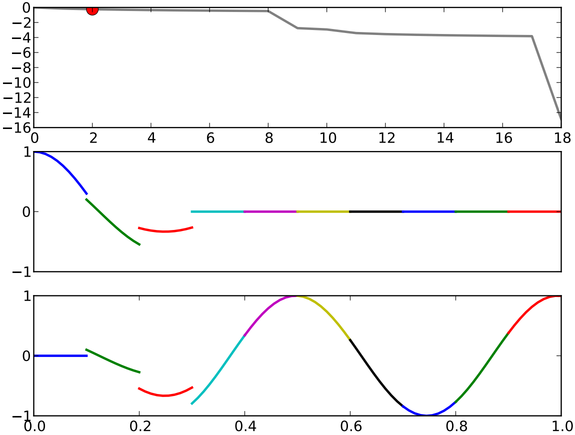
Unpreconditioned optimized Schwarz



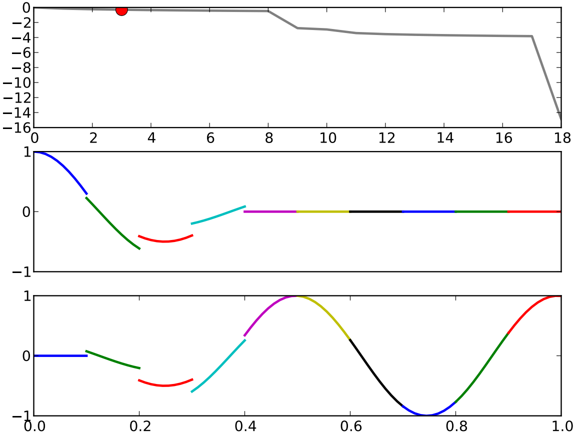
Unpreconditioned optimized Schwarz



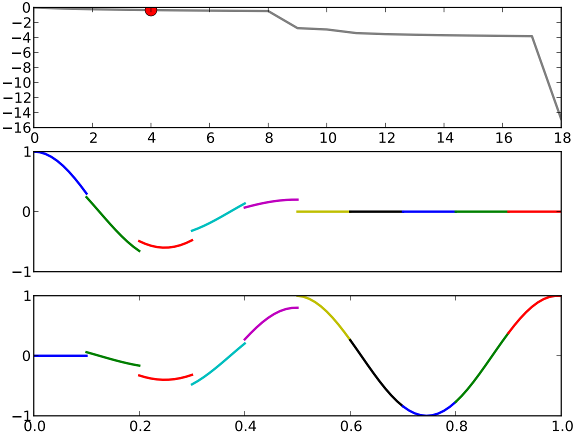
Unpreconditioned optimized Schwarz



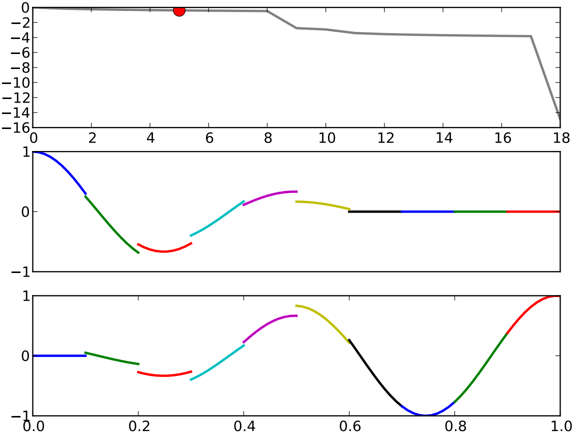
Unpreconditioned optimized Schwarz



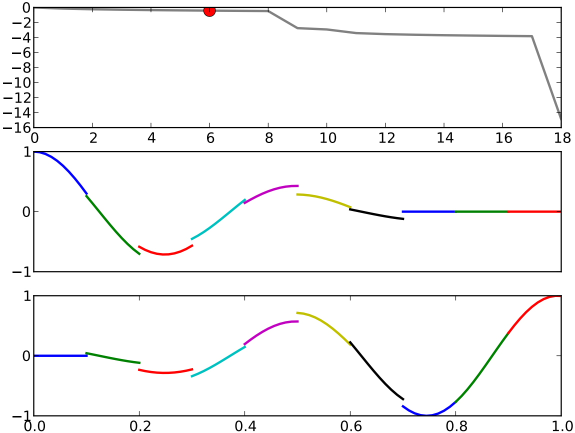
Unpreconditioned optimized Schwarz



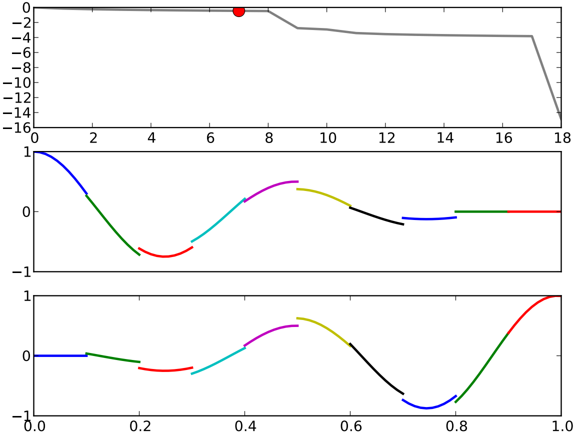
Unpreconditioned optimized Schwarz



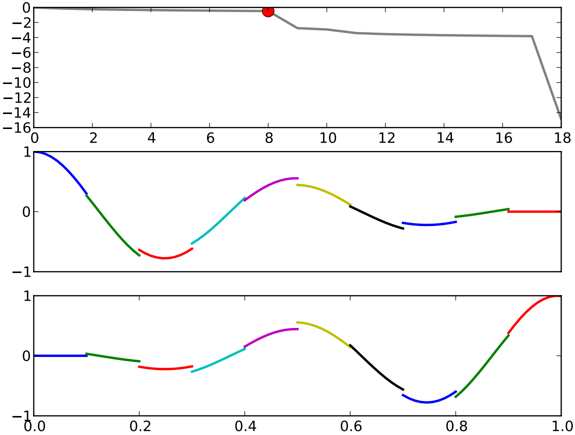
Unpreconditioned optimized Schwarz



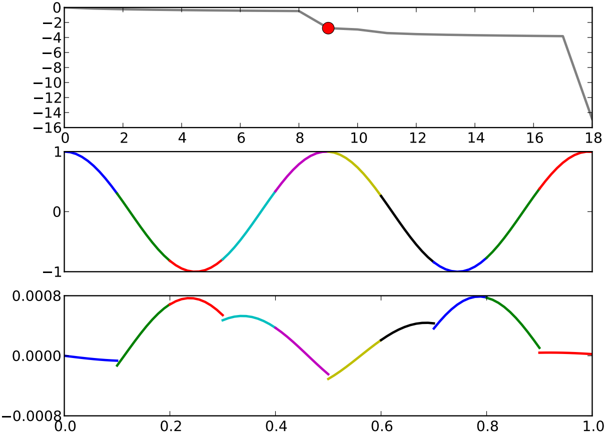
Unpreconditioned optimized Schwarz



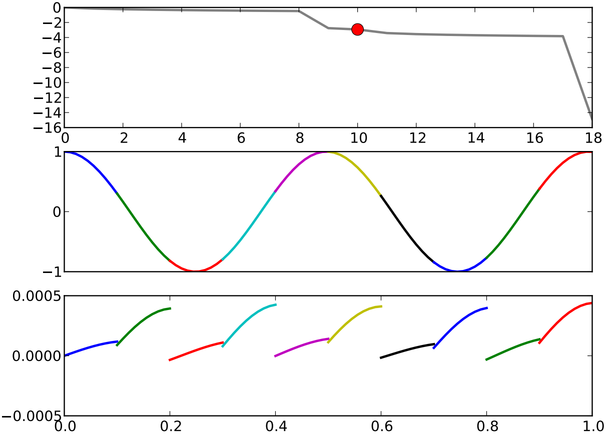
Unpreconditioned optimized Schwarz



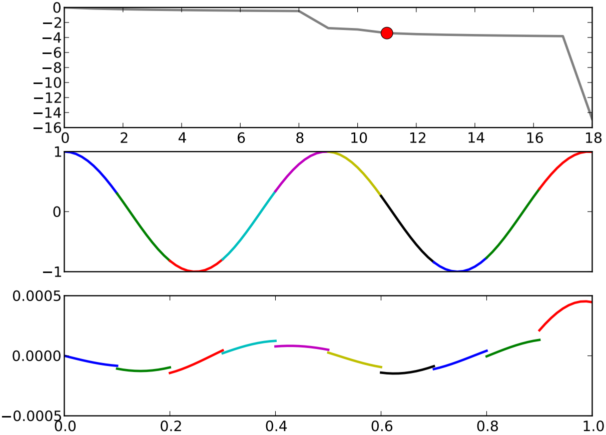
Unpreconditioned optimized Schwarz



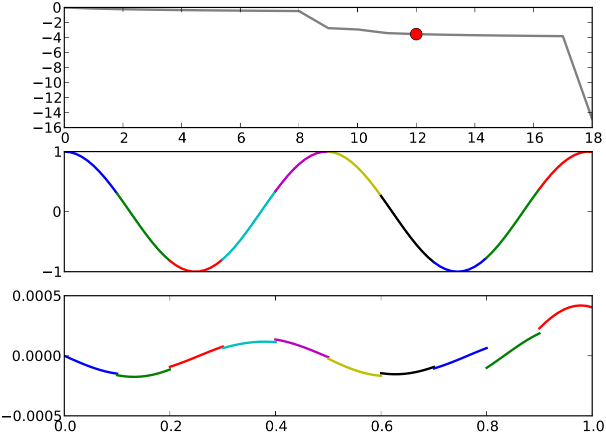
Unpreconditioned optimized Schwarz



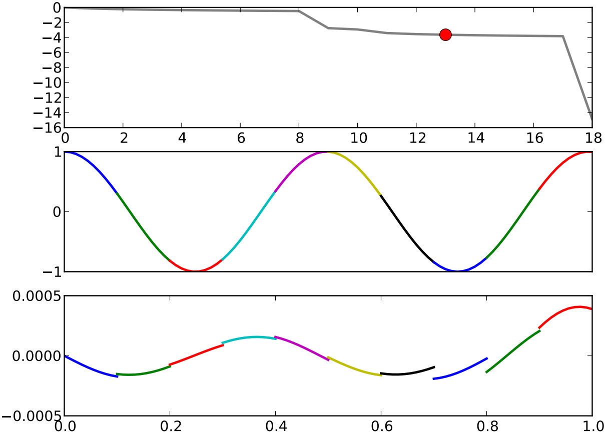
Unpreconditioned optimized Schwarz



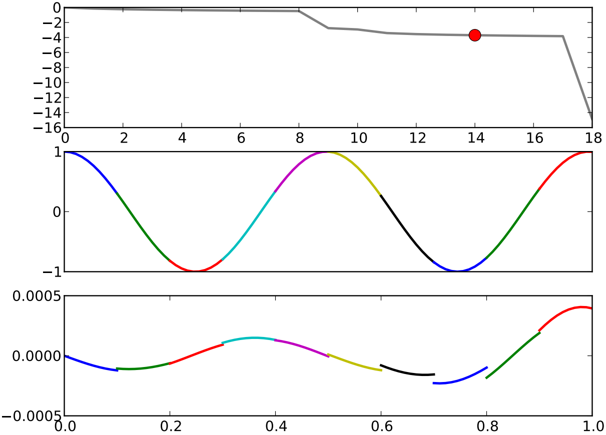
Unpreconditioned optimized Schwarz



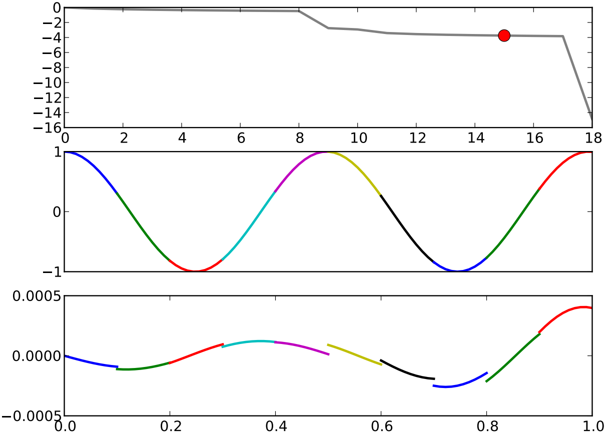
Unpreconditioned optimized Schwarz



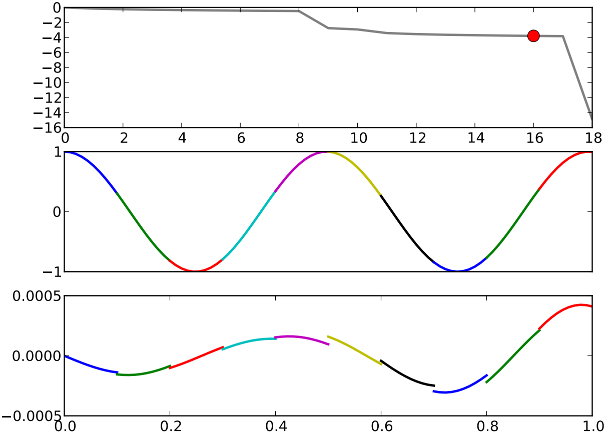
Unpreconditioned optimized Schwarz



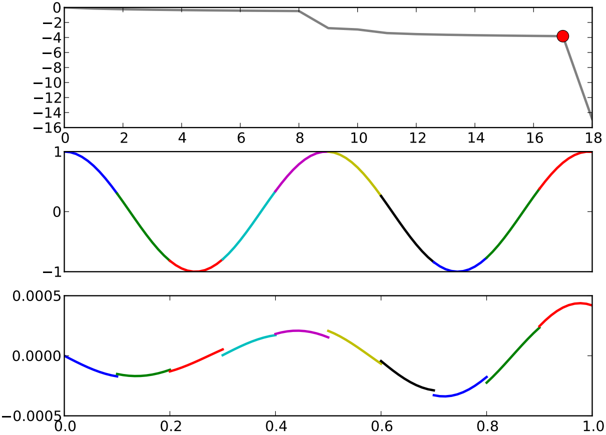
Unpreconditioned optimized Schwarz



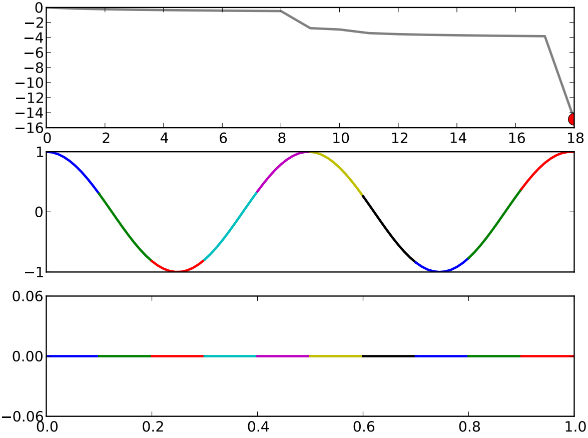
Unpreconditioned optimized Schwarz



Unpreconditioned optimized Schwarz

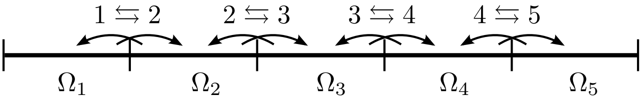


Unpreconditioned optimized Schwarz



Explanation: information is exchanged between neighboring subdomains only

Local interactions: $N-1$ steps to travel from Ω_1 to Ω_N



Remedy: propagate information globally ("coarse grid")

- Well known for Laplace-like problems (Dryja–Widlund, 1989)
- Not obvious how to do that for Helmholtz (plane waves, eigenmodes, ...)

Form the matrix of the iteration operator and use standard algebra to analyze it

Notations:

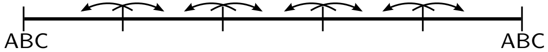
- Operator $\mathcal{F} = \mathcal{I} - \mathcal{A}$ (linear, implies solving N subproblems)
- Matrix F, with equivalence $\mathcal{F}v = Fv, \quad \forall v$

$$\Rightarrow F = \mathcal{F}I$$

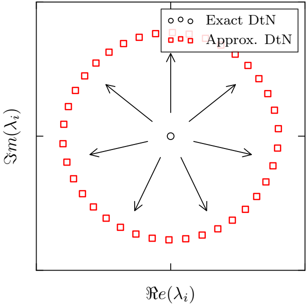
Linear system $Fg = b$, equivalent to the Schwarz problem
N.B.: if one can compute F^{-1}, the problem is solved...

Iteration operator matrix for layered partitioning and constant velocity

1D:



$$F = \begin{bmatrix} 1 & \epsilon & b_2^b & & & & \\ \epsilon & 1 & & & & & \\ & & 1 & \epsilon & \ddots & & \\ & b_2^f & \epsilon & 1 & & & \\ & & \ddots & & \ddots & b_{N-1}^b & \\ & & & & & 1 & \epsilon \\ & & & & b_{N-1}^f & \epsilon & 1 \end{bmatrix}$$



F_A: $\mathcal{S} = \mathcal{D} \Rightarrow \epsilon = 0$; F_N: $\mathcal{S} \approx \mathcal{D} \Rightarrow \epsilon \neq 0$

N.B.: variable velocity has a similar effect (reflection)

Properties of the iteration operator F_A

It is defective: $\lambda_{1,\ldots,M} = 1$

- Algebraic multiplicity $= M = 2(N-1)$
- Geometric multiplicity $= 2$

$\Rightarrow$ slow Krylov convergence, despite perfect conditioning $(\kappa = 1)$

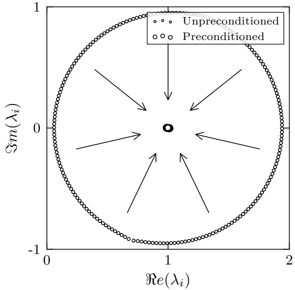
Chen (1977), Strang (1988), Zhongxiao (1998)

Its inverse exists and is easy to find $\forall N$, via recursion formula

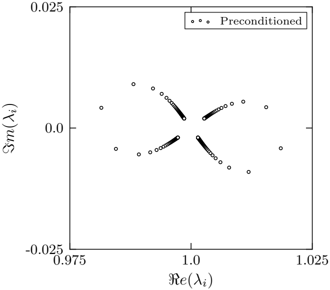
$$\underbrace{\begin{bmatrix} 1 & & b_2^b & & & \\ & 1 & & & & \\ & & 1 & & b_3^b & \\ & b_2^f & & 1 & & \\ & & & & 1 & \\ & & & b_3^f & & 1 \end{bmatrix}}_{F_A} \rightarrow \underbrace{\begin{bmatrix} 1 & & -b_2^b & & b_2^b b_3^b & \\ & 1 & & & & \\ & & 1 & & -b_3^b & \\ & -b_2^f & & 1 & & \\ & & & & 1 & \\ & b_3^f b_2^f & & -b_3^f & & 1 \end{bmatrix}}_{F_A^{-1}}$$

Strategy: use the easy to form F_A^{-1} to precondition the slow convergent F_N

Modify the system: $F_N F_A^{-1} g' = b \quad ; \quad g = F_A^{-1} g'$



Eigenvalues of $F_A^{-1} F_N$

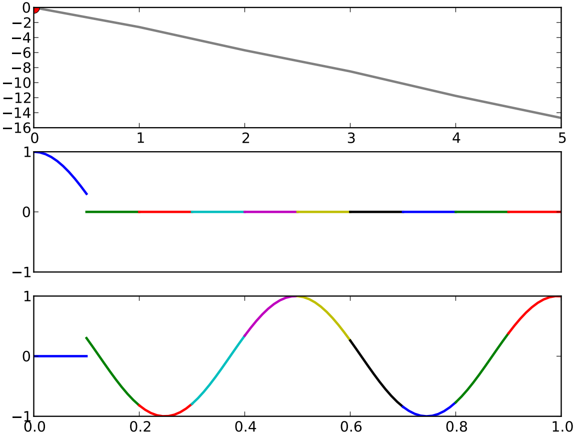


Zoom on $(1, 0)$

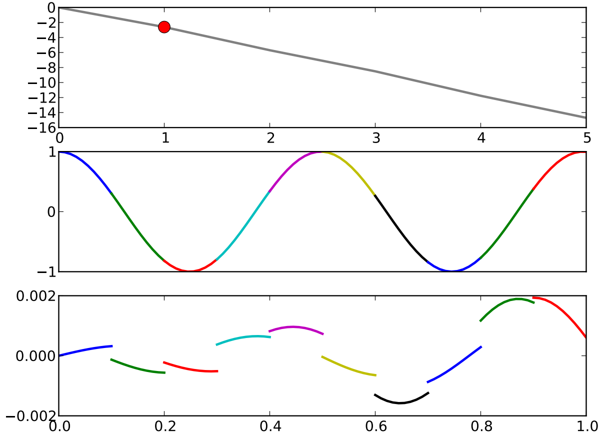
- Excellent clustering of the eigenvalues
- Preconditioned operator is no longer defective

This opens the way to fast convergence !

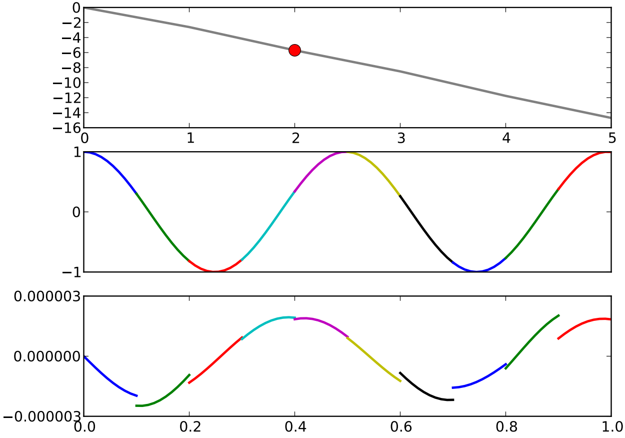
Preconditioned optimized Schwarz



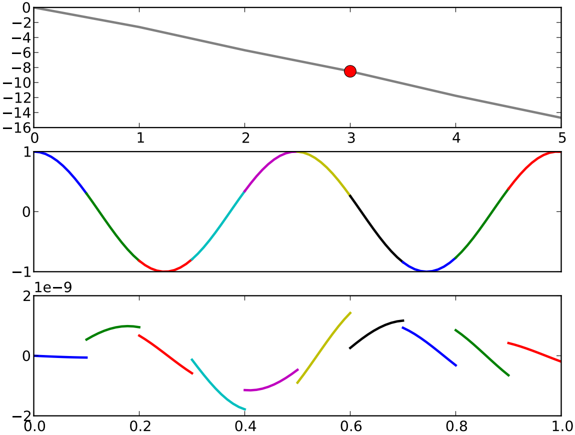
Preconditioned optimized Schwarz



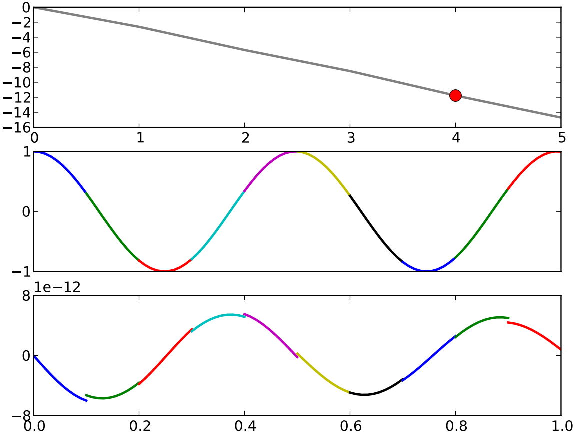
Preconditioned optimized Schwarz



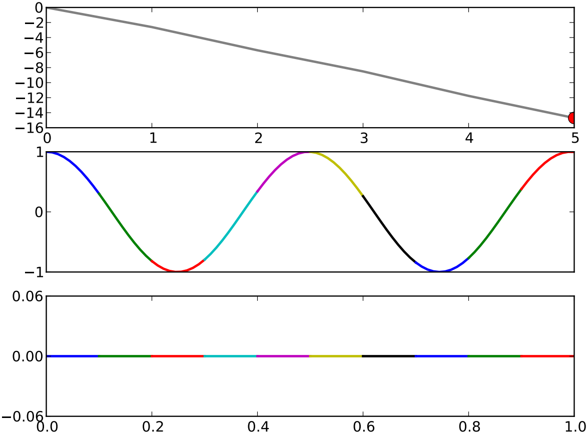
Preconditioned optimized Schwarz



Preconditioned optimized Schwarz



Preconditioned optimized Schwarz



Generalization 2D/3D: matrix coefficients become operators

$$
\mathcal{F}_A^{-1} = \begin{bmatrix}
\mathcal{I} & & -\mathcal{B}_2^b & & \mathcal{B}_2^b\mathcal{B}_3^b & & -\mathcal{B}_2^b\mathcal{B}_3^b\mathcal{B}_4^b & \\
 & \mathcal{I} & & & & & & \\
 & & \mathcal{I} & & -\mathcal{B}_3^b & & \mathcal{B}_3^b\mathcal{B}_4^b & \\
 & -\mathcal{B}_2^f & & \mathcal{I} & & & & \\
 & & & & \mathcal{I} & & -\mathcal{B}_4^b & \\
 & \mathcal{B}_3^f\mathcal{B}_2^f & & -\mathcal{B}_3^f & & \mathcal{I} & & \\
 & & & & & & \mathcal{I} & \\
 & -\mathcal{B}_4^f\mathcal{B}_3^f\mathcal{B}_2^f & & \mathcal{B}_4^f\mathcal{B}_3^f & & -\mathcal{B}_4^f & & \mathcal{I}
\end{bmatrix}
$$

$$
\begin{aligned}
\mathcal{B}_i^{\{f,b\}} : H^{-1/2}(\Sigma_{\{l,r\}}) &\to H^{-1/2}(\Sigma_{\{r,l\}}) \\
g_{\{l,r\}} &\longmapsto 2\mathcal{S}u_i(\{g_l, 0\}, \{0, g_r\})_{|\Sigma_{\{r,l\}}} = \mathcal{B}_i^{\{f,b\}} g_{\{l,r\}};
\end{aligned}
$$

Problem: the cost of applying the preconditioner grows quickly with N

Matrix-free version of the preconditioner: the double sweep

Rearranging the terms of the matrix-vector product $g' = \mathcal{F}_A^{-1} r$ yields the double recurrence relation:

$$g = \begin{bmatrix} g_{12} & g_{21} & \dots & g_{N-1,N} & g_{N,N-1} \end{bmatrix}^T$$

Forward sweep
$$\begin{cases} g'_{21} = r_{21}; \\ g'_{i+1,i} = r_{i+1,i} - \mathcal{B}_i^f g'_{i,i-1}, \quad i = 2, \dots, N-1; \end{cases}$$

Backward sweep
$$\begin{cases} g'_{N-1,N} = r_{N-1,N}; \\ g'_{i-1,i} = r_{i-1,i} - \mathcal{B}_i^b g'_{i,i+1}, \quad i = N-1, \dots, 2. \end{cases}$$

Cost of the preconditioner: $2(N-2)$ sequential subproblem solves

Application of the preconditioner as a double sweep of subproblem solves

Algorithm 1: Application of the double sweep preconditioner $g' \leftarrow F_A^{-1} r$

```
// Forward sweep
```
$g'_{21} \leftarrow r_{21}$
for $i = 2 : N-1$
- $g_l \leftarrow r_{i,i-1}$
- $g_r \leftarrow 0$
- Solve $\mathcal{H}_i u_i = f_i$
- $g'_{i+1,i} \leftarrow r_{i+1,i} + 2\mathcal{S} u_i|_{\Sigma_{i,i+1}}$

end

```
// Backward sweep
```
$g'_{N-1,N} \leftarrow r_{N-1,N}$
for $i = N-1 : 2$
- $g_l \leftarrow 0$
- $g_r \leftarrow r_{i,i+1}$
- Solve $\mathcal{H}_i u_i = f_i$
- $g'_{i-1,i} \leftarrow r_{i-1,i} + 2\mathcal{S} u_i|_{\Sigma_{i,i-1}}$

end

The sweeps are independent and can be performed in parallel
No precomputation required !

The idea of sweeping has emerged some time ago

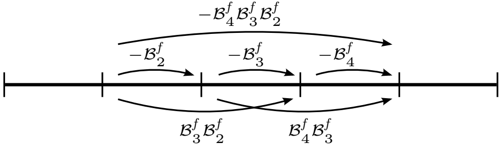
- Nataf, Nier (1997)
- Engquist, Ying (2011)
- Stolk (DD21)

cf. DD22 talk by Hui Zhang this morning

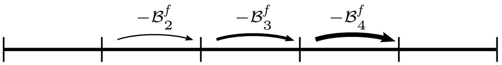
N.B. : we use it as a preconditioner for the Schwarz iteration operator (not the Helmholtz operator), which acts as a coarse grid

Interpretation of the double sweep as a "coarse grid"

The preconditioner F_A^{-1} distributes information all over the domains, instead of only between adjacent domains:



Equivalently, the sweep collects and transports information:



Combined application of operator and preconditioner

Compute the product of the matrices and rearrange terms to avoid redundant solves ; cost is $\mathcal{O}(2N)$ vs. $\mathcal{O}(3N)$

Algorithm 2: Combined application $r \leftarrow FF_A^{-1} r$

```
// g^c contains the correction to the input data
// g^t saves data for use at next step.
```

```
// Forward sweep
```

$g^t_{2,1} \leftarrow 0$

for $i = 2 : N$

- $g_l \leftarrow r_{i,i-1} + g^t_{i,i-1}$
- $g_r \leftarrow 0$
- Solve $\mathcal{H}_i u_i = f_i$
- $g^c_{i-1,i} \leftarrow g_l - 2\mathcal{S} u_i|_{\Sigma_{i,i-1}}$
- $g^t_{i+1,i} \leftarrow 2\mathcal{S} u_i|_{\Sigma_{i,i+1}}$

end

```
// Backward sweep
```

$g^t_{N-1,N} \leftarrow 0$

for $i = N-1 : 1$

- $g_l \leftarrow 0$
- $g_r \leftarrow r_{i,i+1} + g^t_{i,i+1}$
- Solve $\mathcal{H}_i u_i = f_i$
- $g^c_{i+1,i} \leftarrow g_r - 2\mathcal{S} u_i|_{\Sigma_{i,i+1}}$
- $g^t_{i-1,i} \leftarrow 2\mathcal{S} u_i|_{\Sigma_{i,i-1}}$

end

```
// Add correction
```

$r \leftarrow r + g^c$

Some approximations of the DtN map

Local approximations:

IBC — Després (1991) ; Boubendir (2007)

$\mathcal{S}^{IBC(\chi)} = -ik + \chi$

OO_2 — Gander, Magoulès, Nataf (2002)

$\mathcal{S}^{\mathsf{OO}_2} = (a - b\Delta_\Sigma)$

a and b obtained from an optimization procedure

GIBC (Padé – square root) — Boubendir, Antoine & G. (2012)

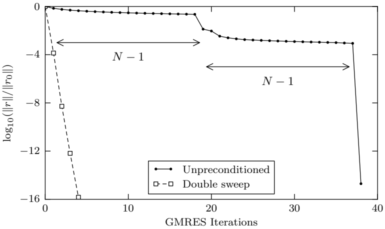
$\mathcal{S}^{\mathsf{GIBC}(N_p)} = C_0 + \sum_{\ell=1}^{N_p} A_\ell \mathrm{div}_\Sigma(k_\varepsilon^{-2}\nabla_\Sigma)(1 + B_\ell \mathrm{div}_\Sigma(k_\varepsilon^{-2}\nabla_\Sigma))^{-1}$

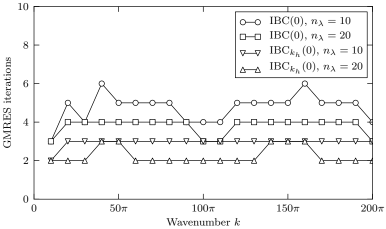
Padé rational expansion of the square-root operator

Non-local approximations:

PML : $\mathcal{S}^{PML(n_{\mathsf{PML}})}$

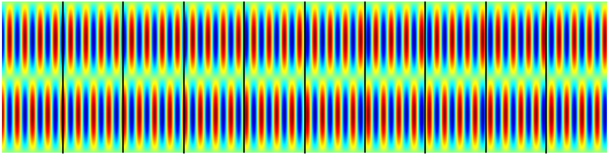
Numerical results — 1D, homogeneous medium





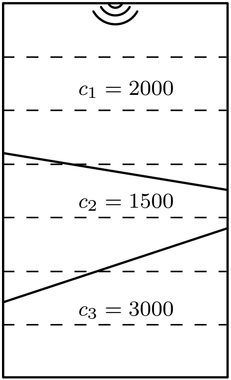
	$N = 5$	25	50	100	150	200
IBC(0), $n_\lambda = 10$	4 (8)	4 (48)	5 (98)	5 (198)	6 (298)	6 (398)
IBC(0), $n_\lambda = 20$	3 (8)	3 (48)	4 (98)	4 (198)	4 (298)	4 (398)
IBC$_{k_h}$(0), $n_\lambda = 10$	3 (8)	3 (48)	3 (98)	3 (198)	3 (298)	3 (398)
IBC$_{k_h}$(0), $n_\lambda = 20$	2 (8)	2 (48)	2 (98)	2 (198)	2 (298)	3 (398)

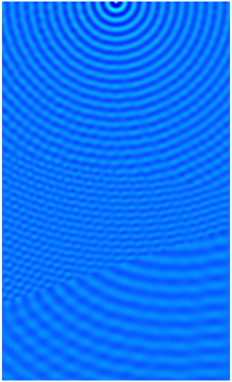
Numerical results — 2D, homogeneous waveguide



	$\omega = 20\pi$					$\omega = 40\pi$				
	$N = 5$	10	25	50	100	5	10	25	50	100
IBC(0)	3 (8)	3 (18)	4 (48)	4 (98)	4 (198)	3	3	4	4	4
OO_2	3 (8)	3 (18)	4 (46)	4 (98)	4 (201)	3	3	3	3	4
GIBC(2)	3 (8)	3 (18)	3 (48)	4 (119)	4 (239)	3	3	4	4	8
PML(5)	4 (8)	4 (18)	5 (48)	6 (96)	6 (196)	4	4	6	8	12
PML(15)	3 (8)	3 (18)	3 (48)	4 (98)	4 (198)	3	3	3	3	4
PML(75)	2 (8)	2 (18)	2 (48)	3 (98)	3 (198)	2	2	2	2	2

Numerical results — 2D, 'Underground'



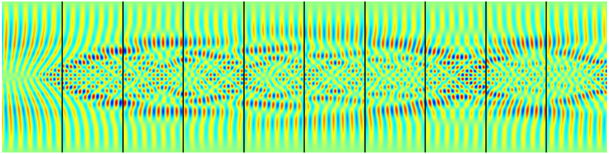


Numerical results — 2D, 'Underground'

	$\omega = 80\pi$					$\omega = 160\pi$				
	$N = 5$	10	25	50	100	5	10	25	50	100
IBC($k/4$)	62 (70)	63 (110)	68 (231)	92 (404)	178 (dnc)	66	67	73	90	168
OO_2	22 (38)	24 (77)	28 (207)	46 (384)	70 (dnc)	25	27	42	74	186
GIBC(2)	25 (40)	27 (74)	29 (186)	35 (369)	41 (dnc)	25	26	29	36	56
PML(5)	15 (38)	16 (75)	17 (195)	23 (368)	29 (dnc)	22	27	43	143	dnc
PML(15)	14 (36)	15 (74)	16 (183)	16 (359)	15 (dnc)	14	15	15	16	79
PML(75)	14 (35)	14 (72)	14 (182)	14 (357)	14 (dnc)	14	14	14	15	15

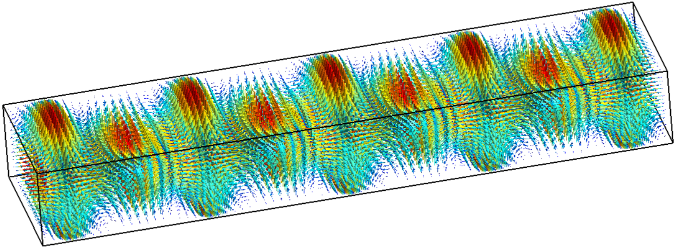
Numerical results — 2D, 'Gaussian' waveguide

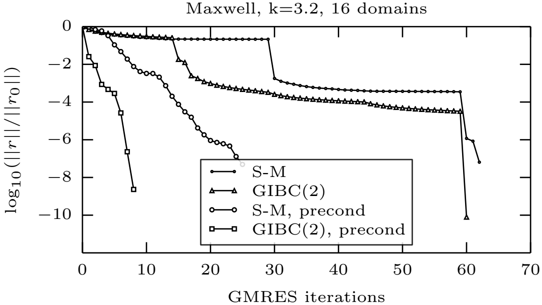
$$c(y) = 1.25(1 - .4\exp(-32(y - .5)^2))$$



	$\omega = 20\pi$					$\omega = 40\pi$				
	$N = 5$	10	25	50	100	5	10	25	50	100
IBC($k/2$)	35 (71)	45 (157)	134 (412)	314 (dnc)	dnc (dnc)	56	82	241	495	dnc
OO_2	30 (62)	33 (128)	69 (356)	175 (dnc)	303 (dnc)	41	53	123	202	dnc
GIBC(2)	19 (53)	20 (114)	42 (314)	98 (dnc)	149 (dnc)	27	31	67	103	288
PML(5)	13 (47)	12 (103)	13 (271)	15 (dnc)	16 (dnc)	16	20	30	52	115
PML(15)	12 (44)	12 (101)	12 (266)	12 (dnc)	12 (dnc)	13	13	13	14	15
PML(75)	11 (44)	11 (99)	11 (264)	11 (dnc)	11 (dnc)	13	13	13	13	13

Numerical results — 3D, waveguide (Maxwell)





Conclusion

The double sweep preconditioner is a coarse grid
for the optimized Schwarz algorithm

- Very simple implementation, no additional preprocessing
- Time to solution:
  - is reduced in sequential (1 proc)
  - *can* be reduced in parallel,
    depending on $\frac{N}{\#\text{proc}}$ and convergence tolerance
- Energy to solution drops drastically

Conclusion

Provided that an accurate enough DtN map approximation is used as transmission operator, the number of GMRES iterations is small and independent of number of domains N and wavenumber k

In homogeneous media, the local approximations perform well;
In variable media, we still need improvements

Perspectives:

Fast application of the preconditioner (approximate solutions)

More general decompositions ?

Thank you for your attention

Preprint available on request

✉ a.vion@ulg.ac.be

Thank you for your attention

Preprint available on request

✉ a.vion@ulg.ac.be

Randomized matrix probing: approximate a matrix when only the matrix-vector product is available

Chiu, Demanet (2012)

$D \in \mathbb{C}^{n \times n}$ is an unknown matrix,
but we have access to $v = Du$

Model: $\exists \; B = \{B_i\}_{1<i<p}$ s.t. $D \approx \widetilde{D} = \sum_{i=1}^{p} x_i B_i$

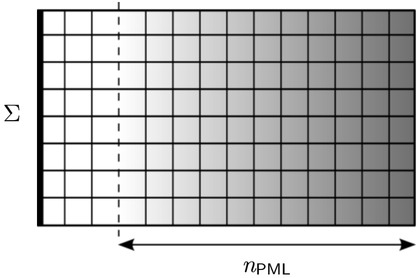
For some random vector(s) u: $v = Du \approx \sum_{i=1}^{p} x_i B_i u = \psi_u x$

Solve for x by taking the pseudo-inverse of ψ_u
$=>$ $\widetilde{D}$ is a least-square approximation of D in span$\{B_i\}_{1<i<p}$

Drawback: (small) probability of failure

The matrix-vector product with the DtN map is obtained by means of a "black-box" that involves a PML

Bélanger-Rioux, Demanet (2012)



$v = \mathcal{D}u$: impose u on Σ (Dirichlet) and solve $\mathcal{H}u = 0$ in Ω_{bb};
"measure" $v = \partial_n u_{|\Sigma}$ (Neumann).

The main challenge is to choose an appropriate set of basis matrices

Use *a priori* knowledge to ensure a good quality and small basis B:

- free-space: geometrical optics
- relaxed terms of the Padé expansion of the square-root operator
- [your input here...]

Low-rank basis matrices ($B_i = b_i b_i^*$) yield fast matrix-vector product, hence fast implementation of the probing procedure and application of the DtN map !

Example: the singular vectors (low-rank by nature) of the DtN map in a waveguide are the modes on the artificial interfaces Σ_{ij}.

Thank you for your attention

Preprint available on request

✉ a.vion@ulg.ac.be