Tensor Multigrid

Steffen Börm
Joint work with Ralf Hiptmair

Christian-Albrechts-Universität, Kiel

University of Zurich, 13th of May, 2015

Overview

Overview

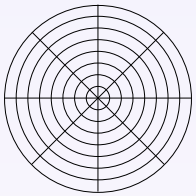
Example: Polar coordinates

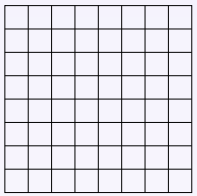
Poisson's equation on the unit disc

$$-\Delta u(x) = f(x) \qquad \text{for all } x \in D := \{x \in \mathbb{R}^2 \ : \ \|x\| < 1\}.$$

Approach: Use angular coordinates

$$\Phi : [0,1] \times [0,2\pi] \to D, \quad \widehat{x} = (r,\varphi) \mapsto \begin{pmatrix} r\sin(\varphi) \\ r\cos(\varphi) \end{pmatrix},$$





with $u \circ \Phi = \widehat{u}$ and $f \circ \Phi = \widehat{f}$ to obtain the equivalent equation

$$-\operatorname{div}\begin{pmatrix} r & \\ & 1/r \end{pmatrix} \operatorname{grad} \widehat{u}(\widehat{x}) = r\widehat{f}(\widehat{x}) \qquad \text{for all } \widehat{x} \in \widehat{D} := (0,1) \times (0,2\pi).$$

Challenge: Coefficients unbounded for $r \to 0$.

General model problem

Domain: $\Omega = \omega_x \times \omega_y$.

Coefficients: $\alpha, \beta : \omega_x \to \mathbb{R}$ with $\alpha, \beta > 0$ almost everywhere.

Differential equation given by

$$-\nabla \cdot \begin{pmatrix} \alpha(x) & \\ & \beta(x) \end{pmatrix} \nabla u(x,y) = f(x,y) \qquad \text{for all } x \in \omega_x,\ y \in \omega_y.$$

Properties

- α and β are allowed to be unbounded,
- α and β depend only on x, not on y,
- Ω is a Cartesian product.

Discretization

Variational formulation: Find $u \in V$ such that

$$a(v,u) = \int_\Omega v(x,y) f(x,y)\, d(x,y) \qquad \text{for all } v \in V,$$

where the bilinear form is given by

$$a(v,u) := \int_\Omega \left\langle \begin{pmatrix} \alpha(x) & \\ & \beta(x) \end{pmatrix} \nabla v(x,y), \nabla u(x,y) \right\rangle d(x,y).$$

Galerkin discretization with a hierarchy $V_0 \subseteq V_1 \subseteq V_2 \subseteq \ldots$ of nodal finite element spaces yields hierarchy of linear systems

$$A_\ell u_\ell = b_\ell.$$

Multigrid solver

Discretization leads to very large systems of linear equations.
$\rightarrow$ Apply multigrid solver.

```
procedure MG(ℓ, b_ℓ, var u_ℓ);
begin
  if ℓ > 0 begin
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ);
    b_{ℓ−1} ← R_ℓ(b_ℓ − A_ℓ u_ℓ);   u_{ℓ−1} ← 0;
    MG(ℓ − 1, b_{ℓ−1}, u_{ℓ−1});
    u_ℓ ← u_ℓ + P_ℓ u_{ℓ−1};
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ)
  end else
    u_0 ← A_0^{-1} b_0
end
```

Multigrid solver

Discretization leads to very large systems of linear equations.
$\rightarrow$ Apply multigrid solver.

```
procedure MG(ℓ, b_ℓ, var u_ℓ);
begin
  if ℓ > 0 begin
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ);
    b_{ℓ−1} ← R_ℓ(b_ℓ − A_ℓ u_ℓ);   u_{ℓ−1} ← 0;
    MG(ℓ − 1, b_{ℓ−1}, u_{ℓ−1});
    u_ℓ ← u_ℓ + P_ℓ u_{ℓ−1};
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ)
  end else
    u_0 ← A_0^{-1} b_0
end
```

Multigrid solver

Discretization leads to very large systems of linear equations.
$\rightarrow$ Apply multigrid solver.

```
procedure MG(ℓ, b_ℓ, var u_ℓ);
begin
  if ℓ > 0 begin
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ);
    b_{ℓ−1} ← R_ℓ(b_ℓ − A_ℓ u_ℓ);   u_{ℓ−1} ← 0;
    MG(ℓ − 1, b_{ℓ−1}, u_{ℓ−1});
    u_ℓ ← u_ℓ + P_ℓ u_{ℓ−1};
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ)
  end else
    u_0 ← A_0^{-1} b_0
end
```

Smoother W_ℓ,

Multigrid solver

Discretization leads to very large systems of linear equations.
$\rightarrow$ Apply multigrid solver.

```
procedure MG(ℓ, b_ℓ, var u_ℓ);
begin
  if ℓ > 0 begin
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ);
    b_{ℓ−1} ← R_ℓ(b_ℓ − A_ℓ u_ℓ);   u_{ℓ−1} ← 0;
    MG(ℓ − 1, b_{ℓ−1}, u_{ℓ−1});
    u_ℓ ← u_ℓ + P_ℓ u_{ℓ−1};
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ)
  end else
    u_0 ← A_0^{-1} b_0
end
```

Smoother W_ℓ, prolongation P_ℓ, restriction R_ℓ.

Multigrid solver

Discretization leads to very large systems of linear equations.
$\rightarrow$ Apply multigrid solver.

```
procedure MG(ℓ, b_ℓ, var u_ℓ);
begin
  if ℓ > 0 begin
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ);
    b_{ℓ−1} ← R_ℓ(b_ℓ − A_ℓ u_ℓ);   u_{ℓ−1} ← 0;
    MG(ℓ − 1, b_{ℓ−1}, u_{ℓ−1});
    u_ℓ ← u_ℓ + P_ℓ u_{ℓ−1};
    u_ℓ ← u_ℓ + W_ℓ^{-1}(b_ℓ − A_ℓ u_ℓ)
  end else
    u_0 ← A_0^{-1} b_0
end
```

Smoother W_ℓ, prolongation P_ℓ, restriction R_ℓ.

Convergence

First experiment: Poisson problem on the unit square.

Level	1	2	3	4	5	6	7
Rate	0.100	0.153	0.167	0.168	0.170	0.170	0.170

Second experiment: Poisson's equation in polar coordinates.

Level	1	2	3	4	5	6	7
Rate	0.301	0.685	0.903	0.974	0.994	0.998	0.999

Convergence

First experiment: Poisson problem on the unit square.

Level	1	2	3	4	5	6	7
Rate	0.100	0.153	0.167	0.168	0.170	0.170	0.170

Second experiment: Poisson's equation in polar coordinates.

Level	1	2	3	4	5	6	7
Rate	0.301	0.685	0.903	0.974	0.994	0.998	0.999

Third experiment: Model problem with tensor multigrid solver.

Level	1	2	3	4	5	6	7
Rate	0.175	0.197	0.198	0.199	0.199	0.200	0.200

Overview

Factorization

Bilinear form for tensor functions:

$$\begin{aligned} a(v_x \otimes v_y, u_x \otimes u_y) &= \int_{\omega_x} \int_{\omega_y} \alpha(x) v_x'(x) v_y(y) u_x'(x) u_y(y) \, dy \, dx \\ &\quad + \int_{\omega_x} \int_{\omega_y} \beta(x) v_x(x) v_y'(y) u_x(x) u_y'(y) \, dy \, dx \\ &= a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y) \end{aligned}$$

with the one-dimensional bilinear forms

$$\begin{aligned} a_x(v_x, u_x) &= \langle \alpha v_x', u_x' \rangle_{L^2}, & a_y(v_y, u_y) &= \langle v_y', u_y' \rangle_{L^2}, \\ m_x(v_x, u_x) &= \langle \beta v_x, u_x \rangle_{L^2}, & m_y(v_y, u_y) &= \langle v_y, u_y \rangle_{L^2}. \end{aligned}$$

Factorization

Bilinear form for tensor functions:

$$\begin{aligned} a(v_x \otimes v_y, u_x \otimes u_y) &= \int_{\omega_x} \int_{\omega_y} \alpha(x) v_x'(x) v_y(y) u_x'(x) u_y(y)\, dy\, dx \\ &\quad + \int_{\omega_x} \int_{\omega_y} \beta(x) v_x(x) v_y'(y) u_x(x) u_y'(y)\, dy\, dx \\ &= a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y) \end{aligned}$$

with the one-dimensional bilinear forms

$$a_x(v_x, u_x) = \langle \alpha v_x', u_x' \rangle_{L^2}, \qquad a_y(v_y, u_y) = \langle v_y', u_y' \rangle_{L^2},$$
$$m_x(v_x, u_x) = \langle \beta v_x, u_x \rangle_{L^2}, \qquad m_y(v_y, u_y) = \langle v_y, u_y \rangle_{L^2}.$$

Factorization

Bilinear form for tensor functions:

$$\begin{aligned} a(v_x \otimes v_y, u_x \otimes u_y) &= \int_{\omega_x} \int_{\omega_y} \alpha(x) v_x'(x) v_y(y) u_x'(x) u_y(y) \, dy \, dx \\ &\quad + \int_{\omega_x} \int_{\omega_y} \beta(x) v_x(x) v_y'(y) u_x(x) u_y'(y) \, dy \, dx \\ &= a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y) \end{aligned}$$

with the one-dimensional bilinear forms

$$\begin{aligned} a_x(v_x, u_x) &= \langle \alpha v_x', u_x' \rangle_{L^2}, & a_y(v_y, u_y) &= \langle v_y', u_y' \rangle_{L^2}, \\ m_x(v_x, u_x) &= \langle \beta v_x, u_x \rangle_{L^2}, & m_y(v_y, u_y) &= \langle v_y, u_y \rangle_{L^2}. \end{aligned}$$

Factorization

Bilinear form for tensor functions:

$$\begin{aligned} a(v_x \otimes v_y, u_x \otimes u_y) &= \int_{\omega_x} \int_{\omega_y} \alpha(x) v_x'(x) v_y(y) u_x'(x) u_y(y) \, dy \, dx \\ &\quad + \int_{\omega_x} \int_{\omega_y} \beta(x) v_x(x) v_y'(y) u_x(x) u_y'(y) \, dy \, dx \\ &= a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y) \end{aligned}$$

with the one-dimensional bilinear forms

$$a_x(v_x, u_x) = \langle \alpha v_x', u_x' \rangle_{L^2}, \qquad a_y(v_y, u_y) = \langle v_y', u_y' \rangle_{L^2},$$
$$m_x(v_x, u_x) = \langle \beta v_x, u_x \rangle_{L^2}, \qquad m_y(v_y, u_y) = \langle v_y, u_y \rangle_{L^2}.$$

Factorization

Bilinear form for tensor functions:

$$\begin{aligned} a(v_x \otimes v_y, u_x \otimes u_y) &= \int_{\omega_x} \int_{\omega_y} \alpha(x) v_x'(x) v_y(y) u_x'(x) u_y(y) \, dy \, dx \\ &\quad + \int_{\omega_x} \int_{\omega_y} \beta(x) v_x(x) v_y'(y) u_x(x) u_y'(y) \, dy \, dx \\ &= a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y) \end{aligned}$$

with the one-dimensional bilinear forms

$$\begin{aligned} a_x(v_x, u_x) &= \langle \alpha v_x', u_x' \rangle_{L^2}, & a_y(v_y, u_y) &= \langle v_y', u_y' \rangle_{L^2}, \\ m_x(v_x, u_x) &= \langle \beta v_x, u_x \rangle_{L^2}, & m_y(v_y, u_y) &= \langle v_y, u_y \rangle_{L^2}. \end{aligned}$$

Factorization

Bilinear form for tensor functions:

$$\begin{aligned} a(v_x \otimes v_y, u_x \otimes u_y) &= \int_{\omega_x} \int_{\omega_y} \alpha(x) v_x'(x) v_y(y) u_x'(x) u_y(y) \, dy \, dx \\ &\quad + \int_{\omega_x} \int_{\omega_y} \beta(x) v_x(x) v_y'(y) u_x(x) u_y'(y) \, dy \, dx \\ &= a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y) \end{aligned}$$

with the one-dimensional bilinear forms

$$\begin{aligned} a_x(v_x, u_x) &= \langle \alpha v_x', u_x' \rangle_{L^2}, & a_y(v_y, u_y) &= \langle v_y', u_y' \rangle_{L^2}, \\ m_x(v_x, u_x) &= \langle \beta v_x, u_x \rangle_{L^2}, & m_y(v_y, u_y) &= \langle v_y, u_y \rangle_{L^2}. \end{aligned}$$

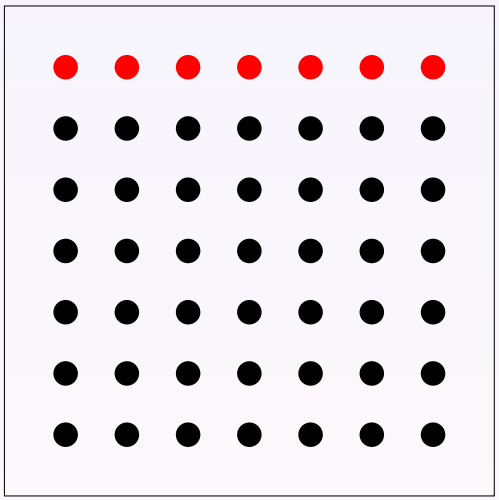
Observation: a_y and m_y well-understood, only a_x and m_x tricky.

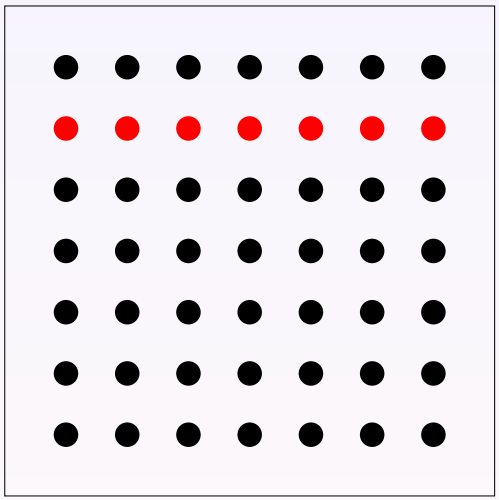
Block smoother

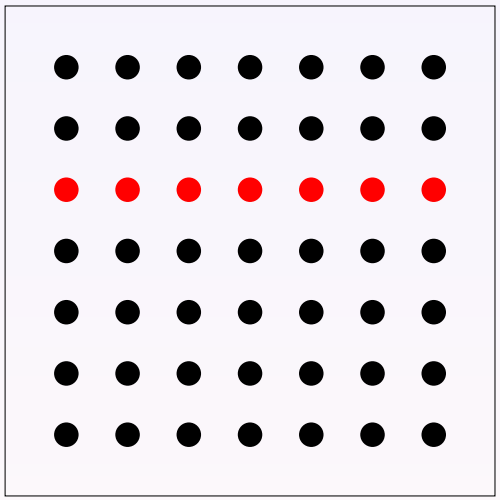
Idea: Use smoothers $w_{a,y}$ and $w_{m,y}$ for a_y and m_y.

$$w(v_x \otimes v_y, u_x \otimes u_y) = a_x(v_x, u_x) w_{m,y}(v_y, u_y) + m_x(v_x, u_x) w_{a,y}(v_y, u_y).$$

Algorithm: Where standard smoothers work with individual grid points, block smoothers work with entire rows of grid points.





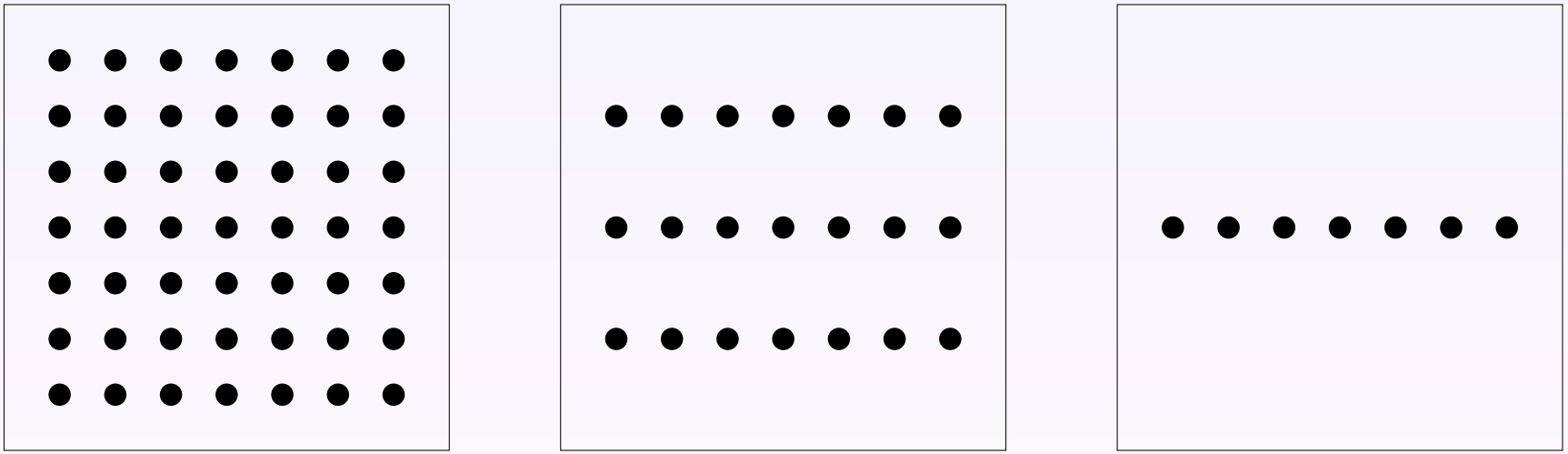


Examples: Schwarz iterations like block Jacobi and block Gauß-Seidel.

Semicoarsened mesh hierarchy

Idea: Keep the mesh resolution fixed in x direction, apply coarsening only in y direction.

$$V_\ell = V_x \otimes V_{y,\ell}, \qquad V_{y,0} \subseteq V_{y,1} \subseteq V_{y,2} \subseteq \dots$$



Increases storage requirements and computing time, but still optimal complexity for V-cycle multigrid.

Overview

General multigrid theory

If we can prove the smoothing property

$$a(v_\ell, v_\ell) \leq w_\ell(v_\ell, v_\ell) \qquad \text{for all } v_\ell \in V_\ell$$

and the approximation property

$$w_\ell(v_\ell, v_\ell) \leq C_A a(v_\ell, v_\ell) \qquad \text{for all } v_\ell \in V_\ell \text{ with } a(v_\ell, V_{\ell-1}) = 0,$$

the convergence rate of the multigrid solver is bounded by

$$\varrho \leq \frac{C_A}{C_A + 2}.$$

☞ D. N. Arnold, R. S. Falk, R. Winther: Preconditioning in $H(\mathrm{div})$ and applications, Math. Comp. 66(219):957–984 (1997)

Eigenvalue problem

Approach: For tensor functions, we have

$$a(v_x \otimes v_y, u_x \otimes u_y) = a_x(v_x, u_x) m_y(v_y, u_y) \quad + m_x(v_x, u_x) a_y(v_y, u_y)$$
$$w_\ell(v_x \otimes v_y, u_x \otimes u_y) = a_x(v_x, u_x) w_{m,\ell}(v_y, u_y) + m_x(v_x, u_x) w_{a,\ell}(v_y, u_y)$$

Eigenvalue problem

Approach: For tensor functions, we have

$$a(v_x \otimes v_y, u_x \otimes u_y) = a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y)$$
$$w_\ell(v_x \otimes v_y, u_x \otimes u_y) = a_x(v_x, u_x) w_{m,\ell}(v_y, u_y) + m_x(v_x, u_x) w_{a,\ell}(v_y, u_y)$$

Eigenvalue problem

Approach: For tensor functions, we have

$$a(v_x \otimes v_y, u_x \otimes u_y) = a_x(v_x, u_x) m_y(v_y, u_y) + m_x(v_x, u_x) a_y(v_y, u_y)$$
$$w_\ell(v_x \otimes v_y, u_x \otimes u_y) = a_x(v_x, u_x) w_{m,\ell}(v_y, u_y) + m_x(v_x, u_x) w_{a,\ell}(v_y, u_y)$$

Idea: Use eigenvector basis $(e_\nu)_{\nu \in \mathcal{I}}$ satisfying

$$a_x(e_\nu, e_\mu) = \lambda_\nu \delta_{\nu\mu}, \qquad m_x(e_\nu, e_\mu) = \delta_{\nu\mu} \qquad \text{for all } \nu, \mu \in \mathcal{I}.$$

Result: We have

$$a(e_\nu \otimes v_y, e_\mu \otimes u_y) = \delta_{\nu\mu} \left(\lambda_\nu m_y(v_y, u_y) + a_y(v_y, u_y)\right),$$
$$w_\ell(e_\nu \otimes v_y, e_\mu \otimes u_y) = \delta_{\nu\mu} \left(\lambda_\nu w_{m,\ell}(v_y, u_y) + w_{a,\ell}(v_y, u_y)\right),$$

i.e., a and w_ℓ reduce to simple one-dimensional bilinear forms.

Orthogonal decomposition

Direct sum: Since $(e_\nu)_{\nu\in\mathcal{I}}$ is a basis, each $v \in V_\ell$ can be written as

$$v = \sum_{\nu\in\mathcal{I}} e_\nu \otimes v_\nu$$

for suitable $(v_\nu)_{\nu\in\mathcal{I}}$.

Orthogonality: By construction, we have

$$a(e_\nu \otimes v_\nu, e_\mu \otimes u_\mu) = \delta_{\nu\mu}\, a_\nu(v_\nu, u_\nu),$$
$$w_\ell(e_\nu \otimes v_\nu, e_\mu \otimes u_\mu) = \delta_{\nu\mu}\, w_\nu(v_\nu, u_\nu),$$

with the bilinear forms

$$a_\nu(v_\nu, u_\nu) = \lambda_\nu m_y(v_\nu, u_\nu) + a_y(v_\nu, u_\nu),$$
$$w_\nu(v_\nu, u_\nu) = \lambda_\nu w_{m,\ell}(v_\nu, u_\nu) + w_{a,\ell}(v_\nu, u_\nu).$$

Smoothing property

Let $v_\ell \in V_\ell$. Let $(v_\nu)_{\nu\in\mathcal{I}}$ be such that

$$v_\ell = \sum_{\nu\in\mathcal{I}} e_\nu \otimes v_\nu.$$

Since a_ν corresponds to a well-behaved problem, we can prove

$$a_\nu(v_\nu, v_\nu) \leq w_\nu(v_\nu, v_\nu).$$

Result: Using the eigenvector expansion, this implies

$$a(v_\ell, v_\ell) = \sum_{\nu\in\mathcal{I}} a_\nu(v_\nu, v_\nu) \leq \sum_{\nu\in\mathcal{I}} w_\nu(v_\nu, v_\nu) = w_\ell(v_\ell, v_\ell).$$

Approximation property

Let $v_\ell \in V_\ell$ with $a(v_\ell, V_{\ell-1}) = 0$, let $(v_\nu)_{\nu\in\mathcal{I}}$ be as before.

Semicoarsening: $e_\nu \otimes u_{\ell-1} \in V_{\ell-1}$ for all $u_{\ell-1} \in V_{y,\ell-1}$, so we get

$$0 = a(v_\ell, e_\nu \otimes u_{\ell-1}) = a_\nu(v_\nu, u_{\ell-1}) \qquad \text{for all } \nu \in \mathcal{I},\ u_{\ell-1} \in V_{y,\ell-1}.$$

Standard multigrid theory can be applied to prove

$$w_\nu(v_\nu, v_\nu) \leq C_A a_\nu(v_\nu, v_\nu)$$

with an independent constant C_A.
(e.g., $C_A = 4$ by Fourier analysis for the model problem).

Using the eigenvalue decomposition again, we find

$$w_\ell(v_\ell, v_\ell) = \sum_{\nu\in\mathcal{I}} w_\nu(v_\nu, v_\nu) \leq C_A \sum_{\nu\in\mathcal{I}} a_\nu(v_\nu, v_\nu) = C_A a(v_\ell, v_\ell).$$

Robustness

Goal: Investigate performance for problems of the form

$$-\operatorname{div}\begin{pmatrix}\alpha(x) & \\ & \beta(x)\end{pmatrix}\operatorname{grad} u(x,y) = f(x,y).$$

α	β	5	6	7	8	9	10	
1	1	0.35	0.35	0.35	0.35	0.35	0.35	Jacobi
x	$1/x$	0.40	0.40	0.40	0.39	0.36	0.35	
10^8	1	0.56	0.56	0.56	0.56	0.56	0.56	
10^{-8}	1	0.40	0.40	0.40	0.40	0.40	0.40	
1	1	0.09	0.09	0.09	0.09	0.09	0.09	Gauß-Seidel
x	$1/x$	0.09	0.09	0.09	0.09	0.09	0.09	
10^8	1	0.09	0.09	0.09	0.09	0.09	0.09	
10^{-8}	1	0.00	0.00	0.00	0.00	0.00	0.01	

Result: Very robust, particularly with red-black Gauß-Seidel.

Overview

Model problem

Classical formulation: We let

$$\operatorname{curl} \mathbf{u} = \partial_1 u_2 - \partial_2 u_1, \qquad \operatorname{curl}^* u = \begin{pmatrix} -\partial_1 u \\ \partial_2 u \end{pmatrix},$$

and consider the equation

$$\operatorname{curl}^* \alpha(x) \operatorname{curl} \mathbf{u}(x,y) + \beta(x)\mathbf{u}(x,y) = \mathbf{f}(x,y) \quad \text{for all } x \in \omega_x,\ y \in \omega_y,$$

where again $\alpha, \beta > 0$ almost everywhere.

Example: Axisymmetric coordinates for three-dimensional electromagnetic simulations.

Discretization

Variational formulation: Find $u \in V$ such that

$$a(v,u) = \int_\Omega \langle \mathbf{v}(x,y), \mathbf{f}(x,y) \rangle \, d(x,y) \qquad \text{for all } v \in V,$$

where the bilinear form is given by

$$\begin{aligned} a(v,u) = &\int_\Omega \alpha(x) \operatorname{curl} \mathbf{v}(x,y) \operatorname{curl} \mathbf{u}(x,y) \, d(x,y) \\ &+ \int_\Omega \beta(x) \langle \mathbf{v}(x,y), \mathbf{u}(x,y) \rangle \, d(x,y). \end{aligned}$$

Galerkin discretization with a hierarchy $V_0 \subseteq V_1 \subseteq V_2 \subseteq \ldots$ of Nédélec's edge-element spaces yields linear systems

$$A_\ell u_\ell = b_\ell.$$

Multigrid solver

Observation: Even for bounded coefficients, standard multigrid does not show robust convergence.

Solution: Handle errors in the null space of curl by hybrid smoother.

☞ R. Hiptmair: Multigrid method for Maxwell's equations, SIAM Num. Anal. 36(1):204–225 (1998)

Multigrid solver

Observation: Even for bounded coefficients, standard multigrid does not show robust convergence.

Solution: Handle errors in the null space of curl by hybrid smoother.

☞ R. Hiptmair: Multigrid method for Maxwell's equations, SIAM Num. Anal. 36(1):204–225 (1998)

☞ S. Börm, R. Hiptmair: Tensor product multigrid for Maxwell's equation with aligned anisotropy, Computing 66:321–342 (2001)

Multigrid solver

Observation: Even for bounded coefficients, standard multigrid does not show robust convergence.

Solution: Handle errors in the null space of curl by hybrid smoother.

☞ R. Hiptmair: Multigrid method for Maxwell's equations, SIAM Num. Anal. 36(1):204–225 (1998)

☞ S. Börm, R. Hiptmair: Tensor product multigrid for Maxwell's equation with aligned anisotropy, Computing 66:321–342 (2001)

Goal: Extend proof obtained for scalar case.

Representation operator

Reminder: In the scalar case, we have used

$$v_\ell = \sum_{\nu \in \mathcal{I}} e_\nu \otimes v_\nu.$$

Generalization: Replace tensor product by representation operator

$$\mathbf{v}_\ell = \sum_{\nu \in \mathcal{I}} R_\nu[v_\nu].$$

Representation operator

Reminder: In the scalar case, we have used

$$v_\ell = \sum_{\nu \in \mathcal{I}} e_\nu \otimes v_\nu.$$

Generalization: Replace tensor product by representation operator

$$\mathbf{v}_\ell = \sum_{\nu \in \mathcal{I}} R_\nu[v_\nu].$$

Our approach: Take rows and gradients explicitly into account.

$$R_\nu[v] := \begin{pmatrix} \epsilon_\nu \otimes v \\ 0 \end{pmatrix} + \nabla(e_\nu \otimes v)$$

with piecewise constant ϵ_ν and piecewise linear e_ν.

Eigenvalue problem

Goal: Find $(\epsilon_\nu)_{\nu\in\mathcal{I}}$ and $(e_\nu)_{\nu\in\mathcal{I}}$ such that

$$\nu \neq \mu \quad \Rightarrow \quad a(R_\nu[v], R_\mu[u]) = 0.$$

Observation of a and R_ν show that we have to ensure

$$\begin{aligned} \nu \neq \mu \quad &\Rightarrow \quad \langle \alpha\epsilon_\nu, \epsilon_\mu\rangle_{L^2} + \langle \beta e_\nu, e_\mu\rangle_{L^2} = 0, \\ \nu \neq \mu \quad &\Rightarrow \quad \langle \beta(\epsilon_\nu + e'_\nu), (\epsilon_\mu + e'_\mu)\rangle_{L^2} = 0. \end{aligned}$$

Idea: Use eigenvector basis $((\epsilon_\nu, e_\nu))_{\nu\in\mathcal{I}}$ given by

$$\begin{aligned} &\langle \alpha\epsilon_\nu, \epsilon_\mu\rangle_{L^2} + \langle \beta e_\nu, e_\mu\rangle_{L^2} = \delta_{\nu\mu}, \\ &\langle \beta(\epsilon_\nu + e'_\nu), (\epsilon_\mu + e'_\mu)\rangle_{L^2} = \lambda_\nu \delta_{\nu\mu} \qquad \text{for all } \nu, \mu \in \mathcal{I}. \end{aligned}$$

Orthogonal decomposition

Direct sum: Each $\mathbf{v} \in V$ can be represented as

$$\mathbf{v} = \sum_{\nu \in \mathcal{I}} R_\nu [v_\nu]$$

Orthogonal decomposition

Direct sum: Each $\mathbf{v} \in V$ can be represented as

$$\mathbf{v} = \sum_{\nu \in \mathcal{I}} R_\nu [v_\nu] + \begin{pmatrix} 0 \\ v_0 \otimes 1 \end{pmatrix}.$$

Orthogonal decomposition

Direct sum: Each $\mathbf{v} \in V$ can be represented as

$$\mathbf{v} = \sum_{\nu \in \mathcal{I}} R_\nu[v_\nu] + \begin{pmatrix} 0 \\ v_0 \otimes 1 \end{pmatrix}.$$

Orthogonality: By construction, we have

$$a(R_\nu[v_\nu], R_\mu[u_\mu]) = \delta_{\nu\mu}\, a_\nu(v_\nu, u_\nu)$$

with the bilinear forms

$$a_\nu(v_\nu, u_\nu) = \lambda_\nu m_y(v_\nu, u_\nu) + a_y(v_\nu, u_\nu)$$

corresponding to standard one-dimensional problems.

Smoother

Reminder: In the scalar case, block smoothers are subspace iterations for the subspaces

$$U_\iota = \operatorname{span}\{e_\nu \otimes \varphi_\iota \ : \ \nu \in \mathcal{I}\},$$

where φ_ι is a nodal basis function.

Generalization: Also use subspace iterations, but with subspaces defined by the representation operator

$$U_\iota = \operatorname{span}\{R_\nu[\varphi_\iota] \ : \ \nu \in \mathcal{I}\}.$$

Implementation: Using suitable bases for U_ι, subspace iterations can be performed efficiently.

Smoothing property

For subspace iterations with subspaces $(U_\iota)_{\iota\in\mathcal{J}}$, the smoothing property is equivalent to

$$a(\mathbf{v},\mathbf{v}) \lesssim \sum_{\iota\in\mathcal{J}} a(\mathbf{v}_\iota,\mathbf{v}_\iota) \qquad \text{for } \mathbf{v} = \sum_{\iota\in\mathcal{J}} \mathbf{v}_\iota,\ \mathbf{v}_\iota \in U_\iota.$$

Idea: Split $\mathbf{v} = \sum_{\nu\in\mathcal{I}} R_\nu[v_\nu]$ and $v_\nu = \sum_{j\in\mathcal{J}} v_{\nu,\iota}$ with $v_{\nu,\iota} \in U_{y,\iota}$.

$$\begin{aligned}
a(\mathbf{v},\mathbf{v}) &= \sum_{\nu\in\mathcal{I}} a(R_\nu[v_\nu],R_\nu[v_\nu]) = \sum_{\nu\in\mathcal{I}} a_\nu(v_\nu,v_\nu)\\
&\lesssim \sum_{\nu\in\mathcal{I}}\sum_{\iota\in\mathcal{J}} a_\nu(v_{\nu,\iota},v_{\nu,\iota}) = \sum_{\nu\in\mathcal{I}}\sum_{\iota\in\mathcal{J}} a(R_\nu[v_{\nu,\iota}],R_\nu[v_{\nu,\iota}])\\
&= \sum_{\iota\in\mathcal{J}} a\Bigg(\underbrace{\sum_{\nu\in\mathcal{I}} R_\nu[v_{\nu,\iota}]}_{=\mathbf{v}_\iota}, \underbrace{\sum_{\nu\in\mathcal{I}} R_\nu[v_{\nu,\iota}]}_{=\mathbf{v}_\iota}\Bigg) = \sum_{\iota\in\mathcal{J}} a(\mathbf{v}_\iota,\mathbf{v}_\iota).
\end{aligned}$$

Convergence

First experiment: Overlapping Schwarz smoothers.

Level	3	4	5	6	7	8	9
Additive	0.55	0.54	0.54	0.54	0.54	0.54	0.54
Multiplicative	0.04	0.05	0.05	0.05	0.05	0.05	0.05

Second experiment: Direct subspace decomposition.

Level	3	4	5	6	7	8	9
Additive	0.37	0.37	0.37	0.37	0.36	0.36	0.36
Multiplicative	0.04	0.05	0.05	0.05	0.05	0.05	0.05

Overview

Summary

Tensor multigrid: Use block smoothing and semicoarsening to hide the effects of unbounded coefficients.

Convergence proof: Use eigenvector basis to reduce to family of one-dimensional problems that can be treated by standard theory.
$\rightarrow$ Convergence rates carry over from one-dimensional case.

Maxwell's equations: Introduce representation operators as a replacement of tensor products, define matching subspace iterations.
$\rightarrow$ Same convergence estimate as in scalar case.