Iteratively solving the neutron transport equation: a new convergence result and a look at domain decomposition approaches.

November, 2014

The problem

The (mono-energetic, steady-state) neutron transport equation in 5D with an isotropic source, $q(\mathbf{r})$, is given by

$$\Omega \cdot \nabla \psi(\mathbf{r}, \Omega) + \sigma_T(\mathbf{r})\psi(\mathbf{r}, \Omega) = \frac{\sigma_S(\mathbf{r})}{4\pi}\int_{\mathbb{S}^2} \psi(\mathbf{r}, \Omega')\, \mathrm{d}\Omega' + q(\mathbf{r})$$

with $\mathbf{r} \in V \subset \mathbb{R}^3$ and $\Omega \in \mathbb{S}^2$. The argument $\psi(\mathbf{r}, \Omega)$ is called the neutron flux.

The problem

The (mono-energetic, steady-state) neutron transport equation in 5D with an isotropic source, $q(\mathbf{r})$, is given by

$$\Omega \cdot \nabla\psi(\mathbf{r}, \Omega) + \sigma_T(\mathbf{r})\psi(\mathbf{r}, \Omega) = \frac{\sigma_S(\mathbf{r})}{4\pi}\int_{\mathbb{S}^2} \psi(\mathbf{r}, \Omega')\,\mathrm{d}\Omega' + q(\mathbf{r})$$

with $\mathbf{r} \in V \subset \mathbb{R}^3$ and $\Omega \in \mathbb{S}^2$. The argument $\psi(\mathbf{r}, \Omega)$ is called the neutron flux. Boundary conditions on the incoming flux

$$\psi(\mathbf{r}, \Omega) = 0, \quad \text{when} \quad \hat{n}(\mathbf{r}) \cdot \Omega < 0, \quad \mathbf{r} \in \delta V.$$

Note: σ_T, σ_S and σ_A are called cross-sections. They are all strictly positive and satisfy $\sigma_T = \sigma_S + \sigma_A$

Operator notation

Introduce

$$\mathcal{T}(\cdot) \equiv \Omega \cdot \nabla(\cdot) + \sigma_T(\mathbf{r})(\cdot) \quad \text{and define} \quad \phi(\mathbf{r}) = \frac{1}{4\pi}\int_{\mathbb{S}^2} \psi(\mathbf{r}, \Omega)\, \mathrm{d}\Omega$$

ϕ is called the scalar flux. So the transport equation can be written as

$$\mathcal{T}\psi = \sigma_S \phi + q$$

Operator notation

Introduce

$$\mathcal{T}(\cdot) \equiv \Omega \cdot \nabla(\cdot) + \sigma_T(\mathbf{r})(\cdot) \quad \text{and define} \quad \phi(\mathbf{r}) = \frac{1}{4\pi}\int_{\mathbb{S}^2} \psi(\mathbf{r}, \Omega)\, \mathrm{d}\Omega$$

ϕ is called the scalar flux. So the transport equation can be written as

$$\mathcal{T}\psi = \sigma_S\phi + q$$

Also we define $\mathcal{K} : L^2(V) \to L^2(V)$ such that if $\mathcal{T}\psi = g$ for $g \in L^2(V)$, then

$$\phi = \mathcal{K}g$$

Source Iteration

Source iteration is an iterative method defined as

$$\mathcal{T}\psi^{(k+1)} = \sigma_S \phi^{(k)} + q,$$
$$\phi^{(k+1)} = \frac{1}{4\pi}\int_{\mathbb{S}^2} \psi^{(k+1)} \,\mathrm{d}\Omega$$

Source Iteration

Source iteration is an iterative method defined as

$$\mathcal{T}\psi^{(k+1)} = \sigma_S \phi^{(k)} + q,$$
$$\phi^{(k+1)} = \frac{1}{4\pi}\int_{\mathbb{S}^2} \psi^{(k+1)} \, \mathrm{d}\Omega$$

Using the operator $\mathcal{K}$, source iteration can also be defined as follows

$$\phi^{(k+1)} = \mathcal{K}\left(\sigma_S \phi^{(k)} + q\right)$$

If the cross-sections (σ_T, σ_S and σ_A) are constant, this basic iterative method is known to converge since $\sigma_S/\sigma_T < 1$.
What if they are not constant?

Source iteration with piecewise continuous cross-sections

A new result for piecewise continuous cross-sections:

Theorem

$$\left\| {\sigma_T}^{\frac{1}{2}} e^{(k+1)} \right\|_{L^2(V)} \leq \left\| \frac{\sigma_S}{\sigma_T} \right\|_{L^\infty(V)} \left\| {\sigma_T}^{\frac{1}{2}} e^{(k)} \right\|_{L^2(V)}$$

with $e^{(k)} \equiv \phi - \phi^{(k)}$, *and so*

$$\left\| \phi - \phi^{(k)} \right\|_{L^2(V)} \to 0, \quad as \quad k \to \infty,$$

Source iteration with piecewise continuous cross-sections

A new result for piecewise continuous cross-sections:

Theorem

$$\left\| {\sigma_T}^{\frac{1}{2}} e^{(k+1)} \right\|_{L^2(V)} \leq \left\| \frac{\sigma_S}{\sigma_T} \right\|_{L^\infty(V)} \left\| {\sigma_T}^{\frac{1}{2}} e^{(k)} \right\|_{L^2(V)}$$

with $e^{(k)} \equiv \phi - \phi^{(k)}$, *and so*

$$\left\| \phi - \phi^{(k)} \right\|_{L^2(V)} \rightarrow 0, \quad as \quad k \rightarrow \infty,$$

So source iteration converges with piecewise continuous cross-sections. Discrete 1D versions of this result are given in both Ashby et. al, 1995, and Brown, 2009, for different discretisations.

Reminder Slide...

- $\psi(\mathbf{r}, \Omega)$ is the neutron flux,
- $\phi(\mathbf{r}) = \frac{1}{4\pi} \int_{\mathbb{S}^2} \psi$ is the scalar flux,
- Transport equation: $\mathcal{T}\psi = \sigma_S \phi + q$
- $\mathcal{T}(\cdot) \equiv \Omega \cdot \nabla(\cdot) + \sigma_T(\mathbf{r})(\cdot)$
- if $\mathcal{T}\psi = g$ then $\phi = \mathcal{K}g$
- Source iteration: $\phi^{(k+1)} = \mathcal{K}\left(\sigma_S \phi^{(k)} + q\right)$
- $\sigma_T(\mathbf{r}) = \sigma_S(\mathbf{r}) + \sigma_A(\mathbf{r})$
- We are proving:

$$\left\| \phi - \phi^{(k)} \right\|_{L^2(V)} \to 0, \quad \text{as} \quad k \to \infty,$$

for piecewise continuous cross-sections

What next for this talk?

Theorem says that source iteration has the potential to converge slowly if $\|\sigma_S/\sigma_T\|_\infty$ is close to 1.

For the rest of this talk we will:

- Consider methods which converge quickly when $\|\sigma_S/\sigma_T\|_\infty$ is close to 1

What next for this talk?

Theorem says that source iteration has the potential to converge slowly if $\|\sigma_S/\sigma_T\|_\infty$ is close to 1.

For the rest of this talk we will:

- Consider methods which converge quickly when $\|\sigma_S/\sigma_T\|_\infty$ is close to 1
- Work in 2 spatial dimensions and 1 angular dimension:

Transport equation in 3D:

$$\mathcal{T}\psi(\mathbf{r},\Omega) = \sigma_S(\mathbf{r})\phi(\mathbf{r}) + q(\mathbf{r})$$

with $\mathbf{r} \in V \subset \mathbb{R}^2$, $\Omega \in \mathbb{S}^1$, and where

$$\phi(\mathbf{r}) = \frac{1}{2\pi}\int_{\mathbb{S}^1} \psi(\mathbf{r},\Omega)\, \mathrm{d}\Omega.$$

Diffusion Synthetic Acceleration (DSA)

Diffusion equation: $$-\nabla\cdot\left(\frac{1}{3\sigma_T(\mathbf{r})}\nabla\Theta(\mathbf{r})\right)+\sigma_A(\mathbf{r})\Theta(\mathbf{r}) = q(\mathbf{r}),$$

subject to: $$\hat{n}\cdot\nabla\Theta(\mathbf{r})+\frac{\sigma_T(\mathbf{r})}{\lambda}\Theta(\mathbf{r}) = 0, \quad \forall\mathbf{r}\in\delta V$$

λ a known constant.

Diffusion Synthetic Acceleration (DSA)

Diffusion equation: $$-\nabla\cdot\left(\frac{1}{3\sigma_T(\mathbf{r})}\nabla\Theta(\mathbf{r})\right)+\sigma_A(\mathbf{r})\Theta(\mathbf{r}) = q(\mathbf{r}),$$

subject to: $$\hat{n}\cdot\nabla\Theta(\mathbf{r})+\frac{\sigma_T(\mathbf{r})}{\lambda}\Theta(\mathbf{r}) = 0, \quad \forall\mathbf{r}\in\delta V$$

λ a known constant.

As $\|\sigma_S/\sigma_T\|_\infty \to 1$:

- Source iteration converges more slowly,
- Θ better aproximates ϕ.

Diffusion Synthetic Acceleration (DSA)

Diffusion equation: $-\nabla\cdot\left(\frac{1}{3\sigma_T(\mathbf{r})}\nabla\Theta(\mathbf{r})\right)+\sigma_A(\mathbf{r})\Theta(\mathbf{r})=q(\mathbf{r}),$

subject to: $\hat{n}\cdot\nabla\Theta(\mathbf{r})+\frac{\sigma_T(\mathbf{r})}{\lambda}\Theta(\mathbf{r})=0, \quad \forall\mathbf{r}\in\delta V$

λ a known constant.

As $\|\sigma_S/\sigma_T\|_\infty \to 1$:

- Source iteration converges more slowly,
- Θ better aproximates ϕ.

Idea: use the good diffusion approximation to "counterbalance" the poor convergence of source iteration as $\|\sigma_S/\sigma_T\|_\infty \to 1$.

Diffusion Synthetic Acceleration (DSA)

Diffusion equation: $$-\nabla\cdot\left(\frac{1}{3\sigma_T(\mathbf{r})}\nabla\Theta(\mathbf{r})\right)+\sigma_A(\mathbf{r})\Theta(\mathbf{r}) = q(\mathbf{r}),$$

subject to: $$\hat{n}\cdot\nabla\Theta(\mathbf{r})+\frac{\sigma_T(\mathbf{r})}{\lambda}\Theta(\mathbf{r}) = 0, \quad \forall \mathbf{r}\in\delta V$$

λ a known constant.

As $\|\sigma_S/\sigma_T\|_\infty \to 1$:

- Source iteration converges more slowly,
- Θ better aproximates ϕ.

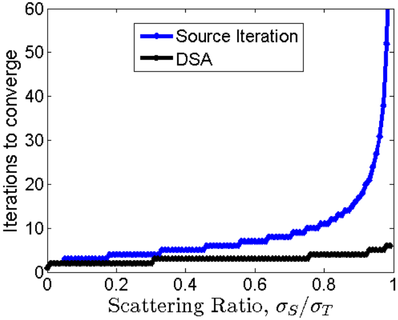
DSA is a 2-step process:

1. Do one step of source iteration,
2. use the diffusion approximation to estimate the error in step 1.

DSA: The Good, the Bad, and the Ugly

GOOD

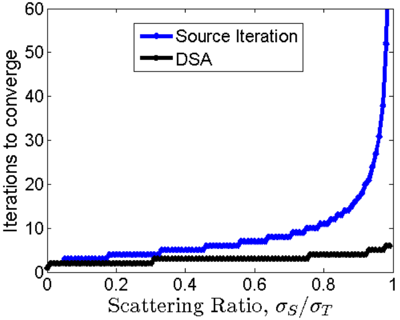
DSA improves upon or maintains the convergence of plain source iteration for all values of σ_S/σ_T



DSA: The Good, the Bad, and the Ugly

GOOD

DSA improves upon or maintains the convergence of plain source iteration for all values of σ_S/σ_T



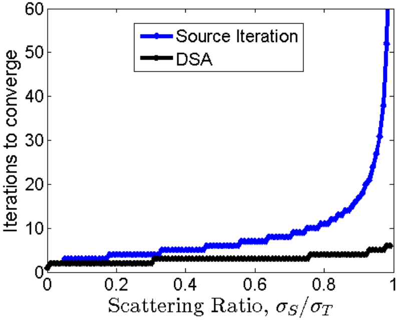
BAD

DSA is more expensive per iteration than source iteration

DSA: The Good, the Bad, and the Ugly

GOOD

DSA improves upon or maintains the convergence of plain source iteration for all values of σ_S/σ_T



BAD

DSA is more expensive per iteration than source iteration

UGLY

In the presence of discontinuities in the cross-sections, multidimensional DSA suffers degraded effectiveness (Azmy, 1998, Warsa et. al., 2004).

DSA with discontinuous cross-sections

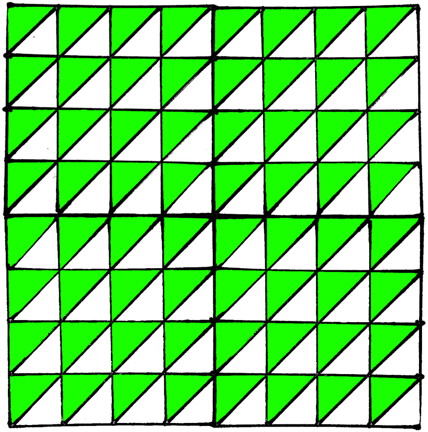
Warsa et. al., 2004, explore one solution to the issue of degrading effectiveness: they form DSA as a preconditioner and apply it to a Krylov iterative method.

This solves the ugly, but not necessarily the bad.

DSA with discontinuous cross-sections

Warsa et. al., 2004, explore one solution to the issue of degrading effectiveness: they form DSA as a preconditioner and apply it to a Krylov iterative method.

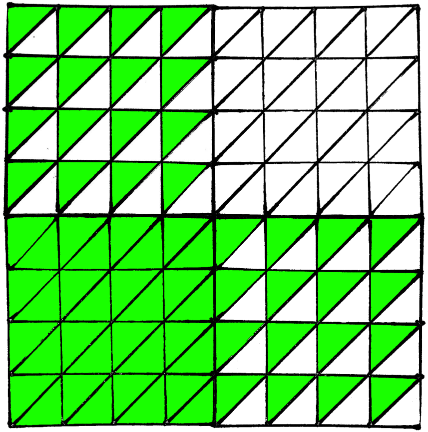
This solves the ugly, but not necessarily the bad.



DSA with discontinuous cross-sections

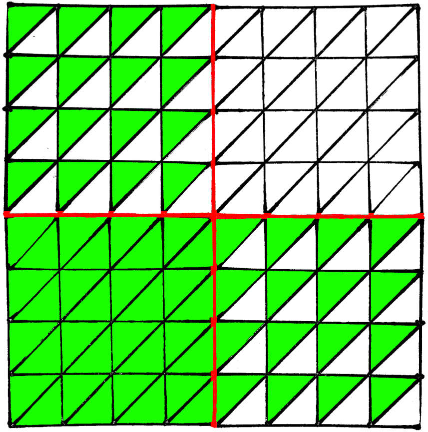
Warsa et. al., 2004, explore one solution to the issue of degrading effectiveness: they form DSA as a preconditioner and apply it to a Krylov iterative method.

This solves the ugly, but not necessarily the bad.



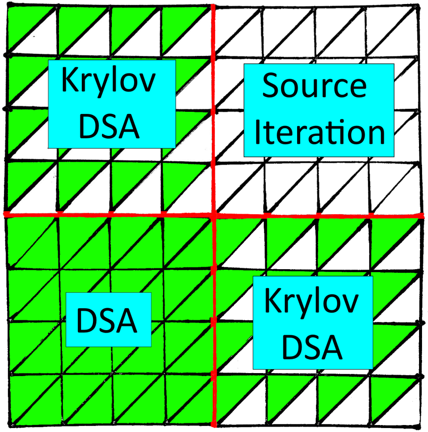
Domain Decomposition: A fistful of methods

We have taken a domain-decomposition approach, splitting the domain into subdomains where different methods can be applied. We believe this approach shows tremendous potential.



Domain Decomposition: A fistful of methods

We have taken a domain-decomposition approach, splitting the domain into subdomains where different methods can be applied. We believe this approach shows tremendous potential.



Domain Decomposition: A few steps back...

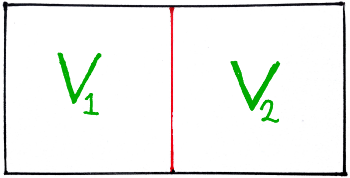
Before we get too excited: how do we organise a domain decomposed source iteration algorithm?

Domain Decomposition: A few steps back...

Before we get too excited: how do we organise a domain decomposed source iteration algorithm?

Model problem:

- – Internal boundary, $V_1 \cap V_2$
- – External boundary, δV where $V = V_1 \cup V_2$

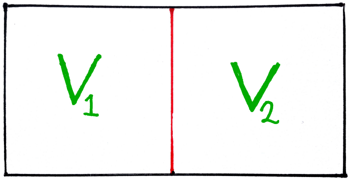


Domain Decomposition: A few steps back...

Before we get too excited: how do we organise a domain decomposed source iteration algorithm?

Model problem:

- – Internal boundary, $V_1 \cap V_2$
- – External boundary, δV where $V = V_1 \cup V_2$



Boundary conditions:

- Zero incoming flux on the external boundary,
- **Internal boundary?**

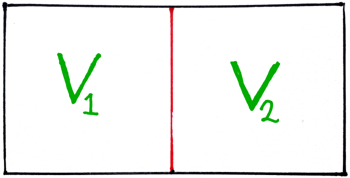
In general: use the flux on V_1 to impose internal conditions for V_2,
use the flux on V_2 to impose internal conditions for V_1.

Domain Decomposition: Algorithm 1

Let $\psi_j \equiv \psi|_{V_j \times \mathbb{S}^1}$,

$\phi_j \equiv \frac{1}{2\pi} \int \psi_j \mathrm{d}\Omega$,

and otherwise let the subscript j denote restriction to subdomain V_j.

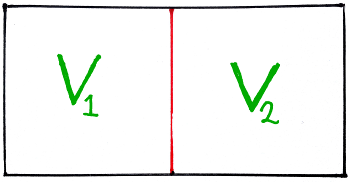


Domain Decomposition: Algorithm 1

Let $\psi_j \equiv \psi|_{V_j \times \mathbb{S}^1}$,

$\phi_j \equiv \frac{1}{2\pi} \int \psi_j \mathrm{d}\Omega$,

and otherwise let the subscript j denote restriction to subdomain V_j.



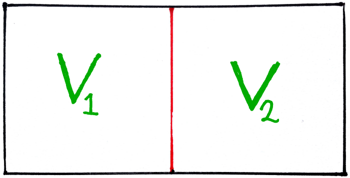
- Solve: $\mathcal{T}_1 \psi_1^{(k+1)} = \sigma_{S1} \phi_1^{(k)} + q_1$

  subj. to:

$$\psi_1^{(k+1)}(\mathbf{r}, \Omega) = \begin{cases} 0 & \text{if } \mathbf{r} \in \delta V_1 \backslash V_2, \quad \text{with } \hat{n}_1(\mathbf{r}) \cdot \Omega < 0, \\ \psi_2^{(k)}(\mathbf{r}, \Omega) & \text{if } \mathbf{r} \in \delta V_1 \cap V_2, \text{ with } \hat{n}_1(\mathbf{r}) \cdot \Omega < 0. \end{cases}$$

Domain Decomposition: Algorithm 1

Let $\psi_j \equiv \psi|_{V_j \times \mathbb{S}^1}$,
$\phi_j \equiv \frac{1}{2\pi} \int \psi_j \mathrm{d}\Omega$,
and otherwise let the subscript j denote restriction to subdomain V_j.

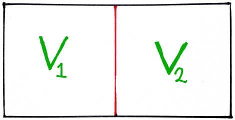


- Solve: $\mathcal{T}_1 \psi_1^{(k+1)} = \sigma_{S1} \phi_1^{(k)} + q_1$
  subj. to:
  $$\psi_1^{(k+1)}(\mathbf{r}, \Omega) = \begin{cases} 0 & \text{if } \mathbf{r} \in \delta V_1 \backslash V_2, \quad \text{with } \hat{n}_1(\mathbf{r}) \cdot \Omega < 0, \\ \psi_2^{(k)}(\mathbf{r}, \Omega) & \text{if } \mathbf{r} \in \delta V_1 \cap V_2, \text{ with } \hat{n}_1(\mathbf{r}) \cdot \Omega < 0. \end{cases}$$

- Solve: $\mathcal{T}_2 \psi_2^{(k+1)} = \sigma_{S2} \phi_2^{(k)} + q_2$
  subj. to:
  $$\psi_2^{(k+1)}(\mathbf{r}, \Omega) = \begin{cases} 0 & \text{if } \mathbf{r} \in \delta V_2 \backslash V_1, \quad \text{with } \hat{n}_2(\mathbf{r}) \cdot \Omega < 0, \\ \psi_1^{(k)}(\mathbf{r}, \Omega) & \text{if } \mathbf{r} \in \delta V_2 \cap V_1, \text{ with } \hat{n}_2(\mathbf{r}) \cdot \Omega < 0. \end{cases}$$

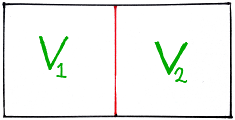
Domain Decomposition: Algorithm 2

We can take much better advantage of angle, Ω, and in particular the flow of information through the subdomains...



Domain Decomposition: Algorithm 2

We can take much better advantage of angle, Ω, and in particular the flow of information through the subdomains...



So we let $U_2 \equiv \left\{\Omega \in \mathbb{S}^1 | \hat{n}_2(\mathbf{r}) \cdot \Omega < 0, \forall \mathbf{r} \in V_1 \cap V_2\right\}$, then for example:

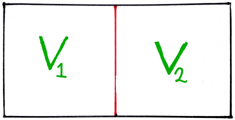
For all $\Omega \in U_2$, we solve on V_1 and then V_2:

- Solve: $\mathcal{T}_1 \psi_1^{(k+1)} = \sigma_{S1} \phi_1^{(k)} + q_1$

  subj. to: $\psi_1^{(k+1)} = 0, \forall \mathbf{r} \in \delta V_1, \text{with } \hat{n}_1(\mathbf{r}) \cdot \Omega < 0$

Domain Decomposition: Algorithm 2

We can take much better advantage of angle, Ω, and in particular the flow of information through the subdomains...



So we let $U_2 \equiv \left\{\Omega \in \mathbb{S}^1 | \hat{n}_2(\mathbf{r}) \cdot \Omega < 0, \forall \mathbf{r} \in V_1 \cap V_2\right\}$, then for example:

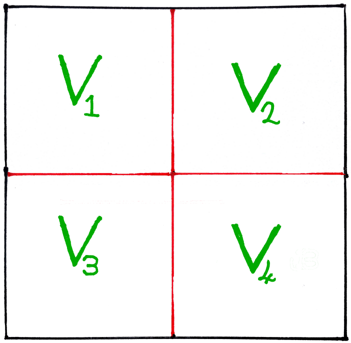
For all $\Omega \in U_2$, we solve on V_1 and then V_2:

- Solve: $\mathcal{T}_1 \psi_1^{(k+1)} = \sigma_{S1} \phi_1^{(k)} + q_1$
  subj. to: $\psi_1^{(k+1)} = 0, \forall \mathbf{r} \in \delta V_1, \text{with } \hat{n}_1(\mathbf{r}) \cdot \Omega < 0$

- Solve: $\mathcal{T}_2 \psi_2^{(k+1)} = \sigma_{S2} \phi_2^{(k)} + q_2$
  subj. to:
  $$\psi_2^{(k+1)} = \begin{cases} 0 & \text{if } \mathbf{r} \in \delta V_2 \backslash V_1, \ \text{with } \hat{n}_2(\mathbf{r}) \cdot \Omega < 0, \\ \psi_1^{(k+1)} & \text{if } \mathbf{r} \in \delta V_2 \cap V_1, \text{with } \hat{n}_2(\mathbf{r}) \cdot \Omega < 0. \end{cases}$$

Results

Model Problem: A 2×2 set of subdomains.



Results

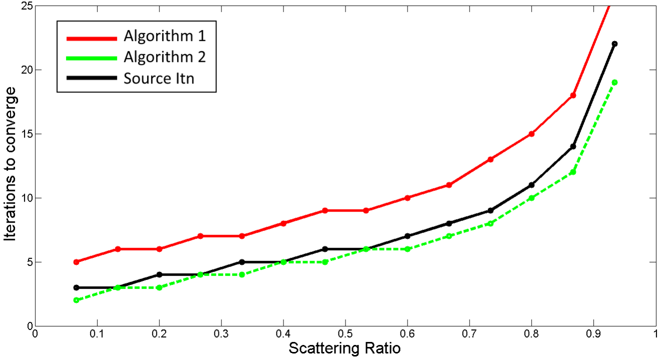


Figure: Iterations to converge to an error of 10^{-4} on a 2×2 subdomain grid. Results for Algorithms 1 & 2 versus source iteration applied over the whole domain.

Results

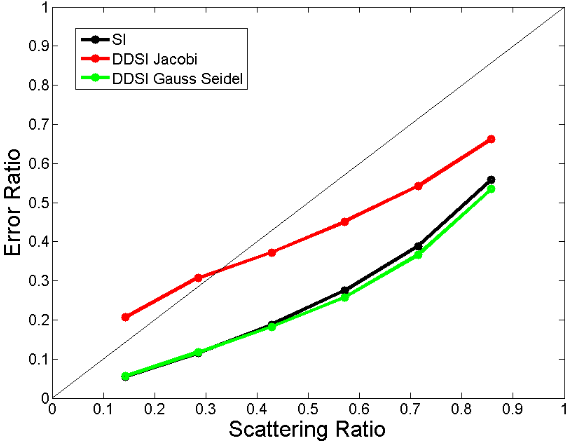


Figure: Ratio between successive errors when solving on a 2×2 subdomain grid. Results for Algorithms 1 & 2 versus source iteration applied over the whole domain.