MA 580; Iterative Methods for Linear Equations

C. T. Kelley
NC State University
tim_kelley@ncsu.edu

Version of October 23, 2016

Read Chapter 1 of the Red book.

NCSU, Fall 2016

Part VIa: Stationary Iterative Methods for Linear Equations

Iterative Methods for $\mathbf{Ax} = \mathbf{b}$

$\mathbf{A}$ is $N \times N$, nonsingular.

- Iterative methods produce a sequence $\{\mathbf{x}_n\}$ converging (you hope) to $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$.
- Typically one terminates the iteration on small relative residuals:
$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} < \tau \text{ where } \mathbf{r} = \mathbf{b} - \mathbf{Ax}.$$
- So we care about the check-your-answer theorem

$$\kappa(\mathbf{A})^{-1}\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} \le \frac{\|\mathbf{e}\|}{\|\mathbf{x}^*\|} \le \kappa(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

Banach Lemma Again

Recall the Banach Lemma. Let $\mathbf{M} \in \mathbf{R}^{N \times N}$. Assume that

$$\|\mathbf{M}\| < 1$$

for some induced matrix norm. Then

- $(\mathbf{I} - \mathbf{M})$ is nonsingular
- $(\mathbf{I} - \mathbf{M})^{-1} = \sum_{l=0}^{\infty} \mathbf{M}^l$
- $\|(\mathbf{I} - \mathbf{M})^{-1}\| \leq (1 - \|\mathbf{M}\|)^{-1}$

Consequence

If the **iteration matrix M** has spectral radius < 1 then the stationary iterative method

$$\mathbf{x}_{n+1} = \mathbf{M}\mathbf{x}_n + \mathbf{b}$$

converges to $\mathbf{x}^* = (\mathbf{I} - \mathbf{M})^{-1}\mathbf{b}$.
Moreover

$$\|\mathbf{x}_n - \mathbf{x}^*\| = O(\rho(\mathbf{M})^n)$$

where

$$\rho(\mathbf{M}) = \max\{|\lambda| \,|\lambda \in \sigma(\mathbf{M})\}$$

is the spectral radius.

Sketch of Linear Richardson (Picard, Fixed-Point) Iteration

```
$\mathbf{r} = \mathbf{b} - \mathbf{x} + \mathbf{M}\mathbf{x}$
while $\|\mathbf{r}\| > \tau\|\mathbf{b}\|$ do
    $\mathbf{r} = \mathbf{b} - \mathbf{x} + \mathbf{M}\mathbf{x}$
    $\mathbf{x} \leftarrow \mathbf{b} + \mathbf{M}\mathbf{x}$
end while
```

Of course, you'd only compute $\mathbf{M}\mathbf{x}$ once in the loop.

Residuals and steps

Since

$$\mathbf{x}^{new} = \mathbf{b} + \mathbf{M}\mathbf{x}^{old}$$

the residual at the old step

$$\mathbf{r}^{old} = \mathbf{b} + \mathbf{M}\mathbf{x}^{old} - \mathbf{x}^{old} = \mathbf{x}^{new} - \mathbf{x}^{old}$$

is the step. So you when you terminate on small residuals, you can return $\mathbf{x}^{new}$, which you've already computed.

Better Version

```
$\mathbf{x}^{new} = \mathbf{b} + \mathbf{M}\mathbf{x}$
$\mathbf{r} = \mathbf{x}^{new} - \mathbf{x}$
while $\|\mathbf{r}\| > \tau\|\mathbf{b}\|$ do
    $\mathbf{x} = \mathbf{x}^{new}$
    $\mathbf{x}^{new} = \mathbf{b} + \mathbf{M}\mathbf{x}$
    $\mathbf{r} = \mathbf{x}^{new} - \mathbf{x}$
end while
$\mathbf{x} = \mathbf{x}^{new}$
```

Preconditioned Richardson Iteration

If $\|\mathbf{I} - \mathbf{A}\| < 1$ then one can apply Richardson iteration directly to $\mathbf{A}\mathbf{x} = \mathbf{b}$

$$\mathbf{x}_{n+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}_n + \mathbf{b}$$

Sometimes one can find a approximate inverse $\mathbf{B}$ for which

$$\|\mathbf{I} - \mathbf{B}\mathbf{A}\| < 1$$

and precondition with $\mathbf{B}$ to obtain

$$\mathbf{B}\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{b} \text{ and the iteration is } \mathbf{x}_{n+1} = (\mathbf{I} - \mathbf{B}\mathbf{A})\mathbf{x}_n + \mathbf{B}\mathbf{b}$$

But now you have two residuals $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ and

$$\mathbf{r}^{pc} = \mathbf{B}\mathbf{b} + (\mathbf{I} - \mathbf{B}\mathbf{A})\mathbf{x} - \mathbf{x} = \mathbf{B}\mathbf{b} - \mathbf{B}\mathbf{A}\mathbf{x}.$$

Matrix Splittings and Classical Methods

One way to convert $\mathbf{Ax} = \mathbf{b}$ to $\mathbf{Mx} = \mathbf{c}$ is to split $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$$

where

- $\mathbf{A}_1$ is nonsingular
- $\mathbf{A}_1\mathbf{y} = \mathbf{q}$ is easy to solve for all $\mathbf{q}$

Two residuals again: $\mathbf{r} = \mathbf{b} - \mathbf{Ax}$ and

$$\mathbf{r}^{split} = \mathbf{A}^{-1}\mathbf{b} - \mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{x} - \mathbf{x}.$$

The iteration measures $\mathbf{r}^{split}$.

Splittings II

Given the splitting $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$

- Solve
$$\mathbf{x} = \mathbf{A}_1^{-1}(\mathbf{b} - \mathbf{A}_2 x) \equiv \mathbf{M}\mathbf{x} + \mathbf{c}.$$
- Where
  - $\mathbf{M} = -\mathbf{A}_1^{-1}\mathbf{A}_2$ and
  - $\mathbf{c} = \mathbf{A}_1^{-1}\mathbf{b}$.
- $\mathbf{A}^{-1}\mathbf{z}$ means solve $\mathbf{A}_1\mathbf{y} = \mathbf{z}$, not compute $\mathbf{A}_1^{-1}$.

Jacobi Iteration: I

Write $\mathbf{Ax} = \mathbf{b}$ explicitly

$$\begin{aligned} a_{11}x_1 + \dots a_{1N}x_N &= b_1 \\ &\vdots \\ a_{N1}x_1 + \dots a_{NN}x_N &= b_N \end{aligned}$$

and solve the ith equation for x_i, pretending the other components are known. You get

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j \right)$$

which is a linear fixed point problem equivalent to $\mathbf{Ax} = \mathbf{b}$.

Jacobi Iteration: II

The iteration is

$$x_i^{New} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{Old} \right)$$

So what are $\mathbf{M}$ and $\mathbf{c}$?

- Split $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$, where $\mathbf{A}_1 = \mathbf{D}, \mathbf{A}_2 = \mathbf{L} + \mathbf{U}$,
- $\mathbf{D}$ is the diagonal of $\mathbf{A}$, and
- $\mathbf{L}$ and $\mathbf{U}$ are the (strict) lower and upper triangular parts.

then $\mathbf{x}^{New} = \mathbf{D}^{-1}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}^{Old})$.

Jacobi Iteration: III

So the iteration is

$$\mathbf{x}_{n+1} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}_n + \mathbf{D}^{-1}\mathbf{b}$$

and the iteration matrix is $\mathbf{M}_{JAC} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$.
Is there any reason for $\rho(\mathbf{M}_{JAC}) < 1$?

Convergence for Strictly Diagonally Dominant A

Theorem: Let $\mathbf{A}$ be an $N \times N$ matrix and assume that $\mathbf{A}$ is strictly diagonally dominant. That is for all $1 \leq i \leq N$

$$0 < \sum_{j \neq i} |a_{ij}| < |a_{ii}|.$$

Then $\mathbf{A}$ is nonsingular and the Jacobi iteration converges to $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ for all $\mathbf{b}$.

Proof: Convergence for Strictly Diagonally Dominant A

Our assumptions imply that $a_{ii} \neq 0$, so the iteration is defined. We can prove everything else showing that

$$\|\mathbf{M}_{JAC}\|_\infty < 1.$$

Remember that $\|\mathbf{M}_{JAC}\|_\infty < 1$ is the maximum absolute row sum. By assumptions, the ith row sum of $\mathbf{M} = \mathbf{M}_{JAC}$ satisfies

$$\sum_{j=1}^{N} |m_{ij}| = \frac{\sum_{j \neq i} |a_{ij}|}{|a_{ii}|} < 1.$$

That's it.

Observations

- Convergence of Jacobi implies $\mathbf{A}$ is nonsingular.
- Showing $\|\mathbf{M}_{JAC}\| < 1$ for any norm would do. The l^∞ norm fit the assumptions the best.
- We have said nothing about the speed of convergence.
- Jacobi iteration does not depend on the ordering of the variables.
- Each x_i^{New} can be processed independently of all the others. So Jacobi is easy to parallelize.

Gauss-Seidel Iteration

Gauss-Seidel changes Jacobi by updating each entry as soon as the computation is done. So

$$x_i^{New} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{New} - \sum_{j>i} a_{ij} x_j^{Old} \right)$$

You might think this is better, because the most up-to-date information is in the formula.

Gauss-Seidel Iteration

One advantage of Gauss-Seidel is that you need only store one copy of x. This loop does the job with only one vector.

```
for i=1:N do
   sum=0;
   for $j \neq i$ do
      $sum = sum + a_{ij} * x_j$
   end for
   $x_i = (b_i + sum)/a_{ii}$
end for
```

Gauss-Seidel Iteration Matrix

From the formula, running for $i = 1, \ldots N$.

$$x_i^{New} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{New} - \sum_{j>i} a_{ij} x_j^{Old} \right)$$

you can see that

$$(\mathbf{D} + \mathbf{L}) x_{n+1} = \mathbf{b} - \mathbf{U} x_n$$

so

$$\mathbf{M}_{GS} = -(\mathbf{D} + \mathbf{L})^{-1} \mathbf{U} \text{ and } \mathbf{c} = (\mathbf{D} + \mathbf{L})^{-1} \mathbf{b}.$$

Backwards Gauss-Seidel

Gauss-Seidel depends on the ordering. Backwards Gauss-Seidel is

$$x_i^{New} = \frac{1}{a_{ii}} \left(b_i - \sum_{j>i} a_{ij} x_j^{New} - \sum_{j<i} a_{ij} x_j^{Old} \right)$$

running from $i = N, \dots 1$. So $\mathbf{M}_{BGS} = -(\mathbf{D} + \mathbf{U})^{-1}\mathbf{L}$.

Symmetric Gauss-Seidel

A symmetric Gauss-Seidel iteration is a forward Gauss-Seidel iteration followed by a backward Gauss-Seidel iteration. This leads to the iteration matrix

$$\mathbf{M}_{SGS} = \mathbf{M}_{BGS}\mathbf{M}_{GS} = (\mathbf{D}+\mathbf{U})^{-1}\mathbf{L}(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}.$$

If **A is symmetric then $\mathbf{U} = \mathbf{L}^T$**. In that event

$$\mathbf{M}_{SGS} = (\mathbf{D}+\mathbf{U})^{-1}\mathbf{L}(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U} = (\mathbf{D}+\mathbf{L}^T)^{-1}\mathbf{L}(\mathbf{D}+\mathbf{L})^{-1}\mathbf{L}^T.$$

SOR iteration

Add a relaxation parameter ω to Gauss-Seidel.

$$\mathbf{M}_{SOR} = (\mathbf{D} + \omega \mathbf{L})^{-1}((1-\omega)\mathbf{D} - \omega \mathbf{U}).$$

Much better performance with good choice of ω.

Example: 2×2

$$\mathbf{A} = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{x}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

One iteration of Jacobi:

$$x_{11} = (1/2)x_{10} = 1/2, x_{21} = (1/2)x_{10} = 1/2$$

Gauss-Seidel:

$$x_{11} = (1/2)x_{20} = 1/2, x_{21} = (1/2)x_{11} = 1/4$$

What about the 3×3 version of this problem?

Observations

- Gauss-Seidel and SOR depend on order of variables.
- So they are harder to parallelize.
- While they may perform better than simple Jacobi, it's not a lot better.
- These methods are not competitive with Krylov methods.
- They require the least amount of storage, and are still used for that reason.

Splitting Methods to Preconditioners

Splitting methods can be seen as preconditioned Richardson iteration.
You want to find the preconditioner **B** so that the iteration matrix from the splitting

$$\mathbf{M} = -\mathbf{A}_1^{-1}\mathbf{A}_2 = \mathbf{I} - \mathbf{B}\mathbf{A}.$$

So $\mathbf{I} - \mathbf{M} = \mathbf{B}\mathbf{A}$.

Jacobi preconditioning

For the Jacobi splitting $\mathbf{A}_1 = \mathbf{D}$, $\mathbf{A}_2 = \mathbf{L} + \mathbf{U}$, we get

- $-\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) = \mathbf{I} - \mathbf{BA}$ so
- $\mathbf{BA} = \mathbf{I} + \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) = \mathbf{D}^{-1}\mathbf{A}$
- Jacobi preconditioning is multiplication by $\mathbf{D}^{-1}$.

This can be a surprisingly good preconditioner for the Krylov methods we get to later.

Discrete Laplacian 1D

We're solving $\mathbf{Au} = \mathbf{b}$ where

$$\mathbf{A} = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \dots & 0, & 0 \\ -1 & 2 & -1 & ,0 & \dots & 0 \\ 0 & -1 & 2 & -1, & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots, & ,0, & -1 & 2 & -1 \\ 0 & \dots, & \dots,, & 0 & -1 & 2 \end{pmatrix}$$

and $h = 1/(N+1)$.

Jacobi and Gauss-Seidel

Jacobi:

```
for i=1:n do
    $u_i^{New} \leftarrow (1/2)(h^2 b_i + u_{i-1}^{Old} + u_{i+1}^{Old})$
end for
```

Gauss-Seidel:

```
for i=1:n do
    $u_i \leftarrow (1/2)(h^2 b_i + u_{i-1} + u_{i+1})$
end for
```

Jacobi Iteration in MATLAB

```
for ijac=1:N_{jac}
   xnew(1) = .5*(h^2 * b(1) + xold(2));
   for i=2:N-1
       xnew(i) = .5*(h^2 * b(i) + xold(i-1) + xold(i
           +1));
   end
   xnew(N) = .5*(h^2 * b(N) + xold(N-1));
   xold=xnew;
end
```

How would you turn this into Gauss-Seidel with a text editor?

Jacobi Example

Let's solve

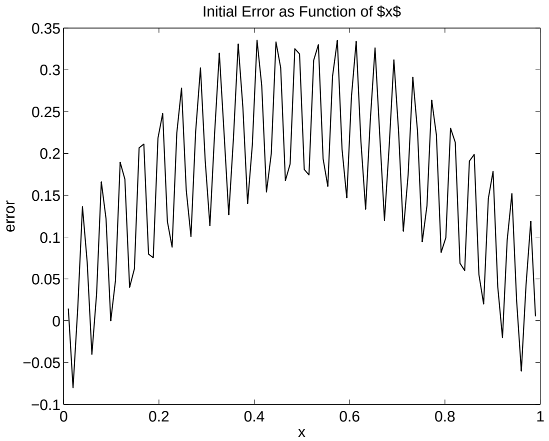
$$-u'' = 0, \; u(0) = u(1) = 0.$$

with $h = 1/101$ and $N = 100$. The solution is $u = 0$. We will use as an intial iterate

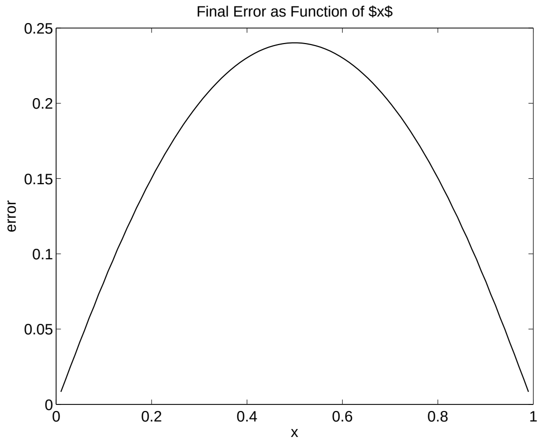
$$u_0 = x(1-x) + \frac{1}{10}\sin(49\pi x)$$

We will take 100 Jacobi iterations.

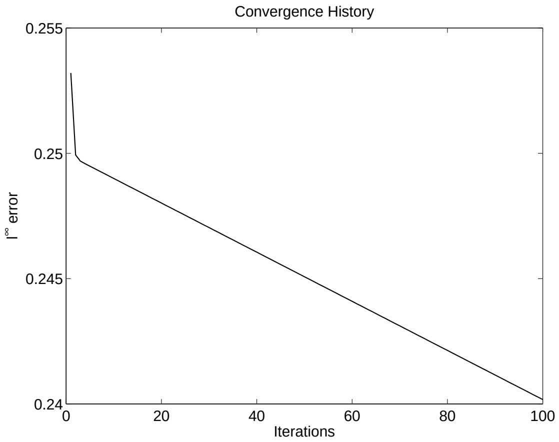
Initial Error as Function of x



Final Error as Function of x



Final Error Norm as Function of Iteration.



What happened?

- Jacobi did a great job on the high-frequency part of the error,
- and a very poor job on the rest.

The eigen-decomposition of $\mathbf{A}$ explains this mess ...

Eigenvalues/vectors of $\mathbf{A}$

Theorem: $\mathbf{A}$ is symmetric positive definite. The eigenvalues are

$$\lambda_n = h^{-2} 2\left(1 - \cos(\pi n h)\right) = \pi^2 n^2 + O(h^2).$$

The eigenvectors $\mathbf{u}_n = (u_1^n, \dots, u_N^n)^T$ are given by

$$u_i^n = \sqrt{2/h} \sin(n i \pi h)$$

So what?

If you apply Jacobi to Poisson's equation, iteration matrix is

$$\mathbf{M} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) = \mathbf{I} - \mathbf{D}^{-1}(\mathbf{D} + \mathbf{L} + \mathbf{U}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$$

as we have seen. For Poisson, $\mathbf{D} = (2/h^2)\mathbf{I}$ so

$$\mathbf{M} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = \mathbf{I} - (h^2/2)\mathbf{A}.$$

The eigenvalues of $\mathbf{M}$ are

$$0 < \mu_n = 1 - (h^2/2)\lambda_n < 1, \text{ So } \rho(\mathbf{M}) = 1 - O(h^2)$$

which is very bad.
The performance gets worse as the mesh is refined!

Observations

- Jacobi (and GS, SOR, . . .) are not scalable.
  - The number of iterations needed to reduce the error by a given amount depends on the grid.
- Fixing this for PDE problems requires a different approach.
- You can solve the 1D problem in $O(N)$ time with a tridiagonal solver, but . . .
- direct methods become harder to use for 2D and 3D problems on complex geometries with unstructured grids.

Poisson's Equation in Two Dimensions

Equation: $-u_{xx} - u_{yy} = f(x, y)$ for $0 < x, y < 1$

Boundary conditions: $u(0, y) = u(x, 0) = u(1, y) = u(x, 1) = 0$

- Similar properties to 1-D
- Physical Grid: (x_i, x_j), $x_i = i * h$.
- Begin with two-dimensional matrix of unknowns $u_{ij} \approx u(x_i, x_j)$.
- Must order the unknowns (ie the grid points) to prepare for a packaged linear solver.

$$u_{xx} \approx \frac{1}{h^2}\left(u(x-h,y) - 2u(x,y) + u(x+h,y)\right)$$

$$u_{yy} \approx \frac{1}{h^2}\left(u(x,y-h) - 2u(x,y) + u(x,y_h)\right)$$

which leads to . . .

Discrete 2D Poisson, Version 1

$$\frac{1}{h^2}\left(-U_{i-1,j} - U_{i,j-1} + 4U_{ij} - U_{i+1,j} - U_{i,j+1}\right) = f_{ij} \equiv f(x_i, x_j)$$

Jacobi, Gauss-Seidel, . . . are still easy. Here's GS

```
for i=1:N do
    for j=1:N do
        $U_{ij} \leftarrow \frac{1}{4}\left(h^2 f_{ij} + U_{i-1,j} + U_{i,j-1} + U_{i+1,j} + U_{i,j+1}\right)$
    end for
end for
```

It's rarely this simple.

- Not all problems have simple matrix representations.
  - Sometimes you only have a black box that returns $\mathbf{Ax} + \mathbf{b}$.
  - You may not have access to the entries of $\mathbf{A}$ or even know what $\mathbf{D}$ is.
- Not all problems fit on a single viewgraph.
- Some problems inspire panic in the novice, but ...

It's rarely this simple.

- Not all problems have simple matrix representations.
  - Sometimes you only have a black box that returns $\mathbf{Ax} + \mathbf{b}$.
  - You may not have access to the entries of $\mathbf{A}$ or even know what $\mathbf{D}$ is.
- Not all problems fit on a single viewgraph.
- Some problems inspire panic in the novice, but ...
  You are no longer a novice.

Neutron Transport Equation

The monoenergetic transport equation in slab geometry with isotropic scattering is

$$\mu \frac{\partial I}{\partial x}(x, \mu) + I(x, \mu) = \frac{c(x)}{2} \int_{-1}^{1} I(x, \mu')\, d\mu' + q(x),$$

for $0 < x < \tau$ and $\mu \in [-1, 0) \cup (0, 1]$.
Boundary Conditions:

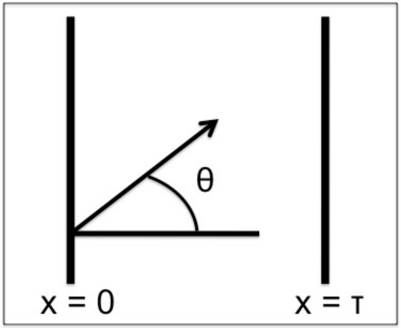
$$I(0, \mu) = I_l(\mu), \mu > 0; I(\tau, \mu) = I_r(\mu), \mu < 0.$$

Terms in the Equation

- I is intensity (aka angular flux) of radiation at point x at angle $\cos^{-1}(\mu)$
- $\tau < \infty$
- $c \in C([0, \tau])$ is mean number of secondaries per collision at x
- I_l and I_r are incoming intensities at the bounds
- $q \in C([0, \tau])$ is the source

Objective: Solve for I

Orientation: $\mu = \cos(\theta)$



Integral Equation Formulation: I

Define the scalar flux

$$f(x) = \int_{-1}^{1} I(x, \mu') \, d\mu'.$$

If f is known we can write the transport equation as

$$\mu \frac{\partial I}{\partial x}(x, \mu) + I(x, \mu) = c(x) f(x)/2 + q(x).$$

We can solve this for I if we are given f.

Computing I if $\mu < 0$

If $\mu > 0$ we use the left boundary condition $x = 0$ and get

$$\begin{aligned} I(x,\mu) &= \frac{1}{\mu}\int_0^x \exp(-(x-y)/\mu)\left(\frac{c(y)}{2}f(y)+q(y)\right)\,dy \\ &+ \exp(-x/\mu)I_l(\mu),\ \mu > 0. \end{aligned}$$

Computing I if $\mu > 0$

If $\mu < 0$, we use the right boundary condition

$$\begin{aligned} I(x,\mu) &= -\frac{1}{\mu}\int_x^\tau \exp(-(x-y)/\mu)\left(\frac{c(y)}{2}f(y)+q(y)\right)\,dy \\ &+\exp((\tau-x)/\mu)I_r(\mu) \\ &= \frac{1}{|\mu|}\int_x^\tau \exp(-|x-y|/|\mu|)\left(\frac{c(y)}{2}f(y)+q(y)\right)\,dy \\ &+\exp(-|\tau-x|/|\mu|)I_r(\mu),\ \mu < 0. \end{aligned}$$

Equation for the Scalar Flux: I

Integrate over $\mu \in (0,1]$ to obtain

$$\int_0^1 I(x,\mu)\,d\mu = \int_0^x k(x,y)f(y)\,dy + g_l(x)$$

where

$$k(x,y) = \frac{1}{2}\int_0^1 \exp(-|x-y|/\mu)\frac{d\mu}{\mu}\,c(y)$$

and

$$g_l(x) = \int_0^x \int_0^1 \frac{1}{\mu}\exp(-(x-y)/\mu)\,d\mu q(y)\,dy + \int_0^1 \exp(-x/\mu)I_l(\mu).$$

Equation for the Scalar Flux: II

Integrate over $\mu \in [-1, 0)$ to obtain

$$\int_{-1}^{0} I(x,\mu)\,d\mu = \int_{x}^{\tau} k(x,y)f(y)\,dy + g_r(y)$$

where

$$\begin{aligned} g_r(y) &= \int_{x}^{\tau}\int_{-1}^{0} \frac{1}{\mu}\exp(-(x-y)/\mu\,d\mu q(y)\,dy \\ &+ \int_{-1}^{0} \exp(-|\tau - x|/|\mu|)I_r(\mu)\,d\mu. \end{aligned}$$

Equation for the Scalar Flux: III

Let I be the solution of the transport equation and f the scalar flux.
We just proved

$$f - \mathcal{K}f = g$$

where the integral operator $\mathcal{K}$ is defined by

$$(\mathcal{K}f)(x) = \int_0^\tau k(x,y)f(y),$$

and

$$g(x) = g_l(x) + g_r(x).$$

Why is this good?

- f is a function of x alone.
- Solving the equation for f allows us to recover I
- Analyzing the integral equation for f is easier than analyzing the integro-differential equation for I

Theorem (Busbridge): If $\|c\|_\infty \le 1$, then the transport equation has a unique solution and the source iteration

$$f_{n+1} = g + \mathcal{K} f_n$$

converges to the scalar flux f from any $f_0 \ge 0$.

Problems?

- Approximating k is hard, so you can't discretize the equation for f directly.
- If c is close to 1 and τ is large, source iteration will converge very slowly.

We can solve the first of these prolbems with a better formulation. Solving the second will have to wait for Krylov methods.

S_N or Discrete Ordinates Discretization: I

Angular Mesh:

- Composite Gauss rule with N_A points
- Subintervals: $(-1, 0)$ and $(0, 1)$
- Nodes: $\{\mu_k\}_{i=1}^{N_A}$; Weights: $\{w_k\}_{i=1}^{N_A}$
- We use 20 point Gauss on each interval, so $N_A = 40$.

Spatial mesh: $\{x_i\}_{i=1}^{N}$

$$x_i = \tau(i-1)/(N-1), \text{ for } i = 1, \dots, N; \; h = \tau/(N-1);$$

Discrete Transport Equation: I

Key idea: Discretize the derivation of the integral equation.
Let $\Phi \in R^N$ be the approximation to the flux

$$\phi_i \approx f(x_i).$$

and let $\Psi \in R^{N \times N_A}$ approximate I

$$\psi_i^j \approx I(x_i, \mu_j).$$

We solve

$$\mu_j \frac{\psi_{i+1}^j - \psi_i^j}{h} + \frac{\psi_{i+1}^j + \psi_i^j}{2} = \frac{S_{i+1} + S_i}{2},$$

where ...

Discrete Transport Equation: II

the source is

$$S_i = \frac{c(x_i)\phi_i}{2} + q(x_i).$$

The boundary conditions are

$$\psi_1^j = I_L(\mu_j) \text{ for } \mu_j > 0$$

and

$$\psi_N^j = I_R(\mu_j) \text{ for } \mu_j < 0.$$

We discreteize the flux equation by discretizing the derivation, not trying to approximate k.

Forward Sweep

For $\mu_j > 0$ (i.e. $\frac{NA}{2} + 1 \leq j \leq NA$) we sweep forward from $i = 1$ to $i = N$,

$$(\mu_j + h/2)\, \psi^j_{i+1} = h\frac{S_{i+1} + S_i}{2} + (\mu_j - h/2)\psi^j_i,$$

so

$$\psi^j_{i+1} = (\mu_j + h/2)^{-1}\left(h\frac{S_{i+1} + S_i}{2} + (\mu_j - h/2)\psi^j_i\right),$$

for $i = 1, \dots, N-1$.

Forward Sweep Algorithm

This algorithm computes Ψ for $\mu_j > 0$
$\Psi(:, N_A/2+1 : N_A) = \textbf{Forward\_Sweep}(\Phi, I_R, I_L, q)$

```
for $j = N_A/2+1 : N_A$ do
    $\psi_1^j = I_L(\mu_j)$
    for $i = 1 : N-1$ do
        $\psi_{i+1}^j = (\mu_j + h/2)^{-1} \left( h\frac{S_{i+1}+S_i}{2} + (\mu_j - h/2)\psi_i^j \right)$
    end for
end for
```

Backward Sweep

For $\mu_j < 0$ (i.e. $1 \le j \le \frac{NA}{2}$) we sweep backward from $i = N$ to $i = 1$

$$(-\mu_j + h/2)\,\psi_i^j = h\frac{S_{i+1} + S_i}{2} + (-\mu_j - h/2)\psi_{i+1}^j$$

so

$$\psi_i^j = (-\mu_j + h/2)^{-1}\left(h\frac{S_{i+1} + S_i}{2} + (-\mu_j - h/2)\psi_{i+1}^j\right)$$

for $i = N - 1, \dots, 1$.

Backward Sweep Algorithm

This algorithm computes Ψ for $\mu_j < 0$
$\Psi(:, 1 : N_A/2) =$ **Backward_Sweep**(Φ, I_R, I_L, q)

```
for $j = 1 : N_A/2$ do
    $\psi_N^j = I_R(\mu_j)$
    for $i = N - 1 : -1 : 1$ do
        $\psi_i^j = (-\mu_j + h/2)^{-1} \left( h \frac{S_{i+1} + S_i}{2} + (-\mu_j - h/2) \psi_{i+1}^j \right)$
    end for
end for
```

Source Iteration Map

Given Φ, compute Ψ with a forward and backward sweep. The source iteration map $\mathcal{S} : R^N \to R^N$ is

$$\mathcal{S}(\Phi, I_R, I_L, q)_i \equiv \sum_{j=1}^{N_A} \psi_i^j w_j$$

and we have solve the transport equation when

$$\Phi = \mathcal{S}(\Phi, I_R, I_L, q).$$

Algorithmic Description

```
$\mathcal{S} = \textbf{Source}(\Phi, I_R, I_L, q)$
    for $i = 1 : N$ do
        $S_i = \frac{c(x_i)\phi_i}{2} + q(x_i).$
    end for
    $\Psi(:, N_A/2 + 1 : N_A) = \textbf{Forward\_Sweep}(\Phi, I_R, I_L, q)$
    $\Psi(:, 1 : N_A/2) = \textbf{Backward\_Sweep}(\Phi, I_R, I_L, q)$
    for $i = 1 : N$ do
        $\mathcal{S}_i = \sum_{j=1}^{N_A} \psi_i^j w_j$
    end for
```

Expression as a Linear System

$$\Phi = M\Phi + b$$

where

$$M\phi = \textbf{Source}(\Phi, 0, 0, 0) \text{ and } b = \textbf{Source}(0, I_R, I_L, q).$$

No matrix representation! You can only get the matrix-vector product via the source iteration map.

Recovering Intensities from Fluxes: I

Suppose you have computed Φ and want to approximate

$$I(x, \nu_j) \text{ for } j = 1, \dots, N_{out}$$

where $\{\nu_j\}$ are some output angles. A typical scenario is computing exit distributions

$$I(0, -\nu_j) \text{ and } I(\tau, \nu_j)$$

for a $\nu_j > 0$, $1 \leq j \leq N_{out}$.
One forward and one backward sweep will do this.

Recovering Intensities from Fluxes: II

Right exit distribution: $I(\tau, \nu_j), \nu_j > 0$

```
for $j = 1 : N_{out}$ do
    $\psi^j_1 = I_L(\nu_j)$
    for $i = 1 : N - 1$ do
        $\psi^j_{i+1} = (\nu_j + h/2)^{-1} \left( h \frac{S_{i+1} + S_i}{2} + (\nu_j - h/2) \psi^j_{i+1} \right)$
    end for
end for
for $j = 1 : N_{out}$ do
    $I(\tau, \nu_j) \approx \psi^j_N$
end for
```

Recovering Intensities from Fluxes: III

Left exit distribution: $I(0, -\nu_j), \nu_j > 0$

```
for $j = 1 : N_{out}$ do
    $\psi_N^j = I_R(-\nu_j)$
    for $i = N - 1 : -1 : 1$ do
        $\psi_i^j = (\nu_j + h/2)^{-1} \left( h \frac{S_{i+1} + S_i}{2} + (\nu_j - h/2) \psi_{i+1}^j \right)$
    end for
end for
for $j = 1 : N_{out}$ do
    $I(0, -\nu_j) \approx \psi_1^j$
end for
```

Example: Source Iteration

In this example

$$c(x) = \omega e^{-x/s}, q(x) \equiv 0,$$

and

$$I_L \equiv 1, I_R \equiv 0.$$

We consider two cases:

- $\tau = 5$; $\omega = 1$, and $s = 1$ (easy)
- $\tau = 100$, $\omega = 1$, and $s = \infty$ (hard)

Source iteration terminates when

$$\|\Phi - \mathcal{S}(\Phi, I_R, I_L, q)\| < 10^{-14}.$$

37 iterations for this example with $\Phi_0 = 0$.

Results for Easy Problem: $\tau = 5$; $\omega = 1$, and $s = 1$

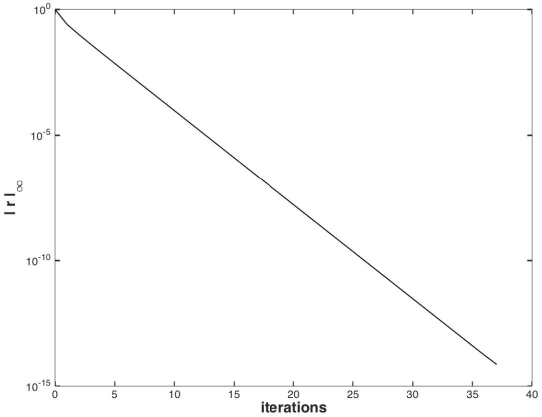
$N_A = 40$; $N = 4001$

μ	$I(\tau, \mu)$	$I(0, -\mu)$
0.05	6.0749e-06	5.8966e-01
0.10	6.9252e-06	5.3112e-01
0.20	9.6423e-06	4.4328e-01
0.30	1.6234e-05	3.8031e-01
0.40	4.3858e-05	3.3296e-01
0.50	1.6937e-04	2.9609e-01
0.60	5.7346e-04	2.6656e-01
0.70	1.5128e-03	2.4239e-01
0.80	3.2437e-03	2.2223e-01
0.90	5.9604e-03	2.0517e-01
1.00	9.7712e-03	1.9055e-01

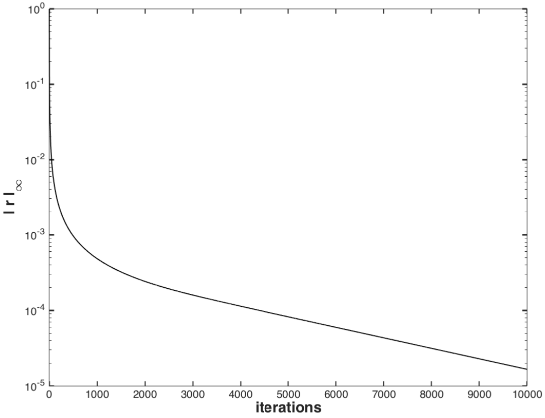
Comments

- These results agree to within one digit in the last place with Tables 1 and 2 of
  R. Garcia and C. Siewert, Radiative transfer in finite inhomogeneous plane-parallel atmospheres, J. Quant. Spectrosc. Radiat. Transfer, 27 (1982), pp. 141–148.
- It will take many more source iterations to get converged results for the hard problem.
- You may need a finer angular/spatial mesh for the harder problem.

Residual History: Easy problem



Residual History: Hard problem


10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
$\| r \|_\infty$
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
iterations