The Linear Boltzmann Transport Equation

C. T. Kelley
NC State University
tim_kelley@ncsu.edu
Research Supported by NSF, DOE, ARO, USACE

DTU ITMAN, 2011
Outline

Linear Boltzmann Transport Equation
Integral Equation Formulation
$S_N$ or Discrete Ordinates Discretization
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 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 $I$
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 \( f \).
If \( \mu > 0 \) we use the left boundary condition \( x = 0 \) and get

\[
I(x, \mu) = \frac{1}{\mu} \int_0^x \exp\left(-\frac{(x - y)}{\mu}\right) \left( \frac{c(y)}{2} f(y) + q(y) \right) \, dy
\]

\[
+ \exp\left(-\frac{x}{\mu}\right) I_l(\mu), \quad \mu > 0.
\]
If $\mu < 0$, we use the right boundary condition

\[
I(x, \mu) = -\frac{1}{\mu} \int_{x}^{\tau} \exp\left(-\frac{(x - y)}{\mu}\right) \left(\frac{c(y)}{2} f(y) + q(y)\right) \, dy
\]

\[
+ \exp\left(-\frac{(\tau - x)}{\mu}\right) I_r(\mu)
\]

\[
= \frac{1}{|\mu|} \int_{x}^{\tau} \exp\left(-\frac{|x - y|}{|\mu|}\right) \left(\frac{c(y)}{2} f(y) + q(y)\right) \, dy
\]

\[
+ \exp\left(-\frac{|	au - x|}{|\mu|}\right) I_r(\mu), \quad \mu < 0.
\]
Equation for the Scalar Flux: \( I \)

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

\[
\int_{0}^{1} I(x, \mu) \, d\mu = \int_{0}^{x} k(x, y) f(y) \, dy + g_{l}(y)
\]

where

\[
k(x, y) = \frac{1}{2} \int_{0}^{1} \exp\left(-|x - y|/\mu\right) \frac{d\mu}{\mu} c(y)
\]

and

\[
g_{l}(y) = \int_{0}^{x} \int_{0}^{1} \frac{1}{\mu} \exp\left(-(x - y)/\mu\right) d\mu q(y) \, dy + \int_{0}^{1} \exp\left(-x/\mu\right) I_{l}(\mu).
\]
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

$$g_r(y) = \int_{x}^{\tau} \int_{-1}^{0} \frac{1}{\mu} \exp\left(-\frac{(x - y)}{\mu}\right) d\mu q(y) \, dy$$

$$+ \int_{-1}^{0} \exp\left(-\frac{\tau - x}{|\mu|}\right) I_r(\mu) \, d\mu.$$
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 \( \|c\|_\infty \leq 1 \), then the transport equation has a unique solution and the source iteration

\[
f_{n+1} = g + Kf_n
\]

converges to the scalar flux \( f \) from any \( f_0 \geq 0 \).
Problems?

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

We can solve the first of these problems 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: $\{x_i\}_{i=1}^N$

$$x_i = \tau(i - 1)/(N - 1), \text{ for } i = 1, \ldots, N; \quad h = \tau/(N - 1);$$
Let $\Phi \in \mathbb{R}^N$ be the approximation to the flux

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

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

$$\psi^j_i \approx I(x_i, \mu_j).$$

We solve

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

where ...
the source is

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

The boundary conditions are

\[ \psi_j^1 = I_L(\mu_j) \text{ for } \mu_j > 0 \]

and

\[ \psi_j^N = I_R(\mu_j) \text{ for } \mu_j < 0. \]

We discreteize the flux equation by discretizing the derivation, not trying to approximate \( k \).
For $\mu_j > 0$ (i.e. $\frac{NA}{2} + 1 \leq j \leq NA$) we sweep forward from $i = 1$ to $i = N$,

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

so

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

for $i = 1, \ldots, N-1$. 
Forward Sweep Algorithm

This algorithm computes $\Psi$ for $\mu_j > 0$

$$
\Psi(\cdot, N_A/2 + 1 : N_A) = \text{Forward\_Sweep}(\Phi, l_R, l_L, q)
$$

for $j = N_A/2 + 1 : N_A$ do
    $\psi^j_1 = l_L(\mu_j)$
    for $i = 1 : N - 1$ do
        $\psi^j_{i+1} = (\mu_j + h/2)^{-1}\left(h\frac{S_{i+1} + S_i}{2} + (\mu_j - h/2)\psi^j_i\right)$
    end for
end for
For $\mu_j < 0$ (i.e., $1 \leq j \leq \frac{NA}{2}$) 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 $i = N - 1, \ldots, 1$. 
This algorithm computes $\Psi$ for $\mu_j < 0$

$\Psi(:, 1 : N_A/2) = \text{Backward Sweep}(\Phi, l_R, l_L, q)$

\[
\text{for } j = 1 : N_A/2 \text{ do}
\]
\[
\psi_j^N = l_R(\mu_j)
\]
\[
\text{for } i = N - 1 : 1 \text{ do}
\]
\[
\psi_i^j = (-\mu_j + h/2)^{-1} \left( h \frac{S_{i+1} + S_i}{2} + (-\mu_j - h/2) \psi_{i+1}^j \right)
\]
\[
\text{end for}
\]
\[
\text{end for}
\]
Given $\Phi$, compute $\Psi$ with a forward and backward sweep. The source iteration map $S : R^N \rightarrow R^N$ is

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

and we have solve the transport equation when

$$\Phi = S(\Phi, l_R, l_L, q).$$
Algorithmic Description

\[ S = \text{Source}(\Phi, l_R, l_L, q) \]

\[
\text{for } i = 1 : N \text{ do}
\]
\[
S_i = \frac{c(x_i)\phi_i}{2} + q(x_i).
\]
\[
\text{end for}
\]

\[ \Psi(:, N_A/2 + 1 : N_A) = \text{Forward Sweep}(\Phi, l_R, l_L, q) \]

\[ \Psi(:, 1 : N_A/2) = \text{Backward Sweep}(\Phi, l_R, l_L, q) \]

\[
\text{for } i = 1 : N \text{ do}
\]
\[
S_i = \sum_{j=1}^{N_A} \psi_j^i w_j
\]
\[
\text{end for} \]
Expression as a Linear System

\[ \Phi = M \Phi + b \]

where

\[ M \phi = \text{Source}(\Phi, 0, 0, 0) \] and \[ b = \text{Source}(0, l_R, l_L, q). \]

No matrix representation! You can only get the matrix-vector product via the source iteration map.
Suppose you have computed $\Phi$ and want to approximate $I(x, \nu_j)$ for $j = 1, \ldots, N_{out}$ where $\{\nu_j\}$ are some output angles. A typical scenario is computing exit distributions $I(0, -\nu_j)$ 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.
Right exit distribution: $I(\tau, \nu_j), \nu_j > 0$

\begin{verbatim}
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
    $I(\tau, \nu_j) \approx \psi_N^j$
end for
\end{verbatim}
Left exit distribution: $I(0, -\nu_j), \nu_j > 0$

for $j = 1 : N_{out}$ do

\[
\psi_j^N = I_R(-\nu_j)
\]

for $i = N - 1 : 1$ do

\[
\psi_j^i = (\nu_j + h/2)^{-1} \left( h \frac{S_{i+1} + S_i}{2} + (\nu_j - h/2)\psi_j^{i+1} \right)
\]

end for

end for

for $j = 1 : N_{out}$ do

\[
I(0, -\nu_j) \approx \psi_1^j
\]

end for
Example: Source Iteration

In this example

\[ c(x) = \omega e^{-x/s} \]

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 \( \| \Phi - S(\Phi) \| < 10^{-14} \). 41 iterations for this example with \( \Phi_0 = 0 \).
Results for Easy Problem: $\tau = 5; \omega = 1$, and $s = 1$

$N_A = 80; N = 4001$

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>$I(\tau, \mu)$</th>
<th>$I(0, -\mu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>6.0749e-06</td>
<td>5.8966e-01</td>
</tr>
<tr>
<td>0.10</td>
<td>6.9251e-06</td>
<td>5.3112e-01</td>
</tr>
<tr>
<td>0.20</td>
<td>9.6423e-06</td>
<td>4.4328e-01</td>
</tr>
<tr>
<td>0.30</td>
<td>1.6234e-05</td>
<td>3.8031e-01</td>
</tr>
<tr>
<td>0.40</td>
<td>4.3858e-05</td>
<td>3.3297e-01</td>
</tr>
<tr>
<td>0.50</td>
<td>1.6937e-04</td>
<td>2.9609e-01</td>
</tr>
<tr>
<td>0.60</td>
<td>5.7346e-04</td>
<td>2.6656e-01</td>
</tr>
<tr>
<td>0.70</td>
<td>1.5128e-03</td>
<td>2.4239e-01</td>
</tr>
<tr>
<td>0.80</td>
<td>3.2437e-03</td>
<td>2.2224e-01</td>
</tr>
<tr>
<td>0.90</td>
<td>5.9603e-03</td>
<td>2.0517e-01</td>
</tr>
<tr>
<td>1.00</td>
<td>9.7712e-03</td>
<td>1.9055e-01</td>
</tr>
</tbody>
</table>
These results agree to within one digit in the last place with (Siewert et al)

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.
I will give you a program gauss.m to generate the angular weights and nodes. Use double 40 point Gauss for this exercise as a start.

- Write a source iteration code yourself. Make $c(x)$, $\tau$, $nx$, $\psi_L$, and $\psi_R$ inputs to the program.
- Duplicate the results from the lecture and do the hard problem.
- Perform a grid refinement study on your results for the flux. Increase the angular mesh to double 40 point and let $nx = 8001$. Do you see any significant changes?