Joseph Murphy, February 2020

ASTR 202 Final Project

An interative approach to solving the two-stream problem

In [1]:
```
# Two Stream object
from TwoStream import TwoStream
from TwoStream_plot_utils import plot

# Plotting
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams["font.family"] = "serif"
rcParams["font.serif"] = "Times New Roman"
%matplotlib inline
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = [r'\usepackage{amsmath} \usepackage{b
m} \usepackage{physics}']
%config InlineBackend.figure_format = 'retina' # For high quality figure
s

# Autoreload for loading changes to class definition document without re
starting Kernel
%load_ext autoreload
%autoreload 2
```

The two-stream problem

Consider a plane-parallel distribution of material from $z = 0$ to $z = 1$. The material emits thermally with $B = 1$ everywhere, and has uniform photon destruction probability ϵ. The extinction coefficient in the material varies with height as $\alpha(z) = 10^{5-6z}$, such that $\alpha(0) = 10^5$ and $\alpha(1) = 10^{-1}$.

Assume that I_- and I_+ are oriented at $\cos(\theta) = \mu = \pm 1/\sqrt{3}$ such that $d\tau = \alpha \frac{dz}{\mu}$ along each stream, where θ is the angle between the z-axis and the ray.

Goal: Iteratively solve for a discretized two-stream solution to the specific intensities I_- and I_+, the mean intensity $J = \frac{1}{2}(I_- + I_+)$, and the source function $S = \epsilon B + (1 - \epsilon)J$.

Boundary conditions:

At $z = 0$, $I_+ = B$.

At $z = 1$, $I_- = 0$.

Iterative scheme for approximate radiative transfer

The general approach to solving the two stream problem will follow:

```
while NOT converged:
```

$$\{$$

1. Given an approximation to the source function S^n, compute I_+^n and I_-^n by interpolating S^n from a grid.
2. Compute the mean intensity J^n from I_+^n and I_-^n.
3. Compute an updated approximation to the source function, $S^{n+1} = \epsilon B + (1-\epsilon)J^n$.

$$\}$$

To calculate I_+^n and I_-^n in step 1, we will interpolate the source function to third-order with terms weighted by the local change in the optical depth, $\Delta\tau$:

$$\begin{aligned} I_{+,i} &= I_{+,i-1}\exp(-\Delta\tau_{i-1}) + \gamma_+ S_{i-1} + \beta_+ S_i + \alpha_+ S_{i+1} \\ &\text{and} \\ I_{-,i} &= I_{-,i+1}\exp(-\Delta\tau_{i+1}) + \gamma_- S_{i-1} + \beta_- S_i + \alpha_- S_{i+1}. \end{aligned}$$

The coefficients $\alpha_\pm$, $\beta_\pm$, and $\gamma_\pm$ are computed according to:

$$\begin{aligned} e^0_{+,i} &= 1 - \exp(-\Delta\tau_i) \\ e^1_{+,i} &= \Delta\tau_i - e^0_{+,i} \\ e^2_{+,i} &= \Delta\tau_i^2 - 2e^1_{+,i} \\ \alpha_+ &= \frac{e^2_{+,i} - \Delta\tau_i e^1_{+,i}}{\Delta\tau_{i+1}(\Delta\tau_i + \Delta\tau_{i+1})} \\ \beta_+ &= \frac{(\Delta\tau_i + \Delta\tau_{i+1})e^1_{+,i} - e^2_{+,i}}{\Delta\tau_i\Delta\tau_{i+1}} \\ \gamma_+ &= e^0_{+,i} + \frac{e^2_{+,i} - (\Delta\tau_{i+1} + 2\Delta\tau_i)e^1_{+,i}}{\Delta\tau_i(\Delta\tau_i + \Delta\tau_{i+1})} \end{aligned}$$

For I_- the coefficients are

$$\begin{aligned} e^0_{-,i} &= 1 - \exp(-\Delta\tau_{i+1}) \\ e^1_{-,i} &= \Delta\tau_{i+1} - e^0_{-,i} \\ e^2_{-,i} &= \Delta\tau_{i+1}^2 - 2e^1_{-,i} \\ \alpha_- &= e^0_{-,i} + \frac{e^2_{-,i} - (\Delta\tau_i + 2\Delta\tau_{i+1})e^1_{-,i}}{\Delta\tau_{i+1}(\Delta\tau_i + \Delta\tau_{i+1})} \\ \beta_- &= \frac{(\Delta\tau_i + \Delta\tau_{i+1})e^1_{-,i} - e^2_{-,i}}{\Delta\tau_i\Delta\tau_{i+1}} \\ \gamma_- &= \frac{e^2_{-,i} - \Delta\tau_{i+1} e^1_{-,i}}{\Delta\tau_i(\Delta\tau_i + \Delta\tau_{i+1})}. \end{aligned}$$

Solution

We'll use the TwoStream object in TwoStream.py to solve for the iterative solution.

First, we'll start with the case where there is appreciable absorption, with $\epsilon = 0.1$.

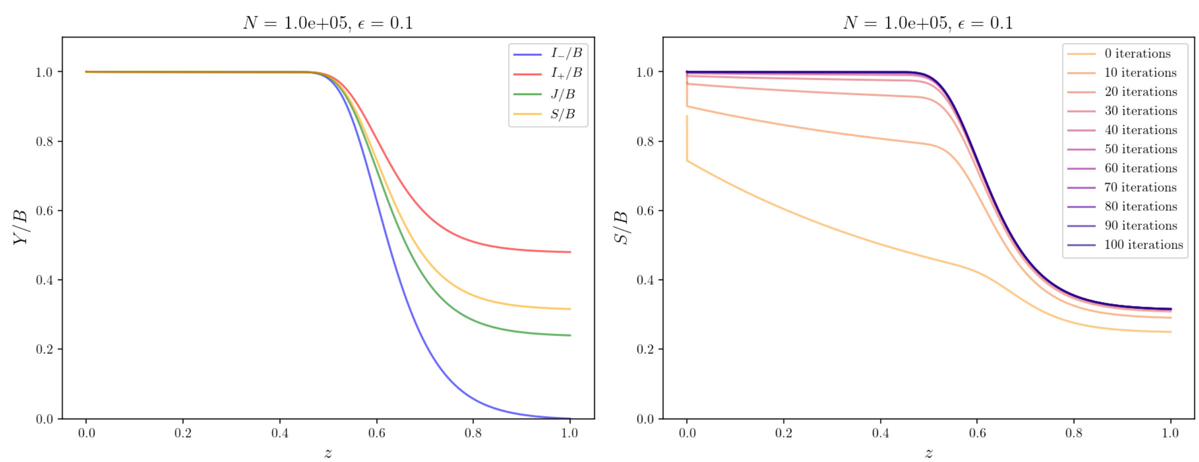
In [2]:
```
# Input params
max_iters=100
grid_num=int(1e5)
epsilon=0.1

# Run the simulation
ts = TwoStream(max_iters=max_iters, grid_num=grid_num, epsilon=epsilon)
results = ts.iterate()
```

```
100%|██████████| 100/100 [01:19<00:00,  1.25it/s]

Convergence criterion met within tolerance level of 1.00e-06.
Convergence first reached after 79 iterations.
```

In [3]:
```
# Plot the results
fig, axl, axr = plot(ts)
```



Note: Above N is the number of points in the z grid.

Now decrease the absorption by setting $\epsilon = 0.01$.

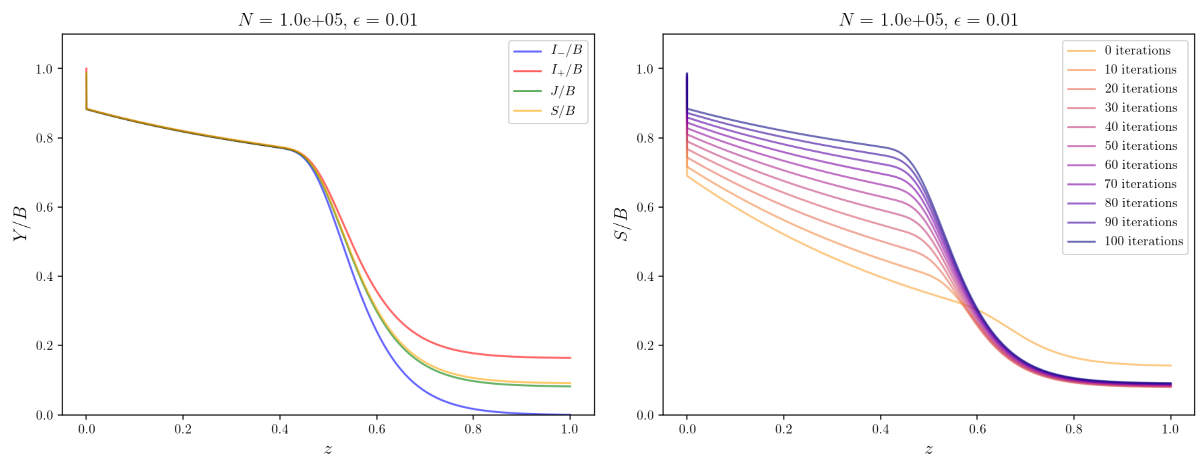
In [4]:
```
# Input params
max_iters=100
grid_num=int(1e5)
epsilon=0.01

# Run the simulation
ts_small_eps = TwoStream(max_iters=max_iters, grid_num=grid_num, epsilon
=epsilon)
results = ts_small_eps.iterate()
```

```
100%|██████████| 100/100 [01:27<00:00,  1.19it/s]

Warning, source function approximation not converged within tolerance l
evel.
```

In [5]:
```
# Plot the results
fig_small_eps, axl_small_eps, axr_small_eps = plot(ts_small_eps)
```



We can see from the discontinuity in the output near $z = 0$ (and from the convergence warning) that the solution is not yet converged. The convergence test we use for the source function calculates if the L2 norm of the difference between consecutive (× the sampling frequency) source function solutions is within the tolerance level. The default tolerance level is 1e-6.

Now we'll increase the maximum number of iterations allowed to see how many iterations it takes to reach convergence.

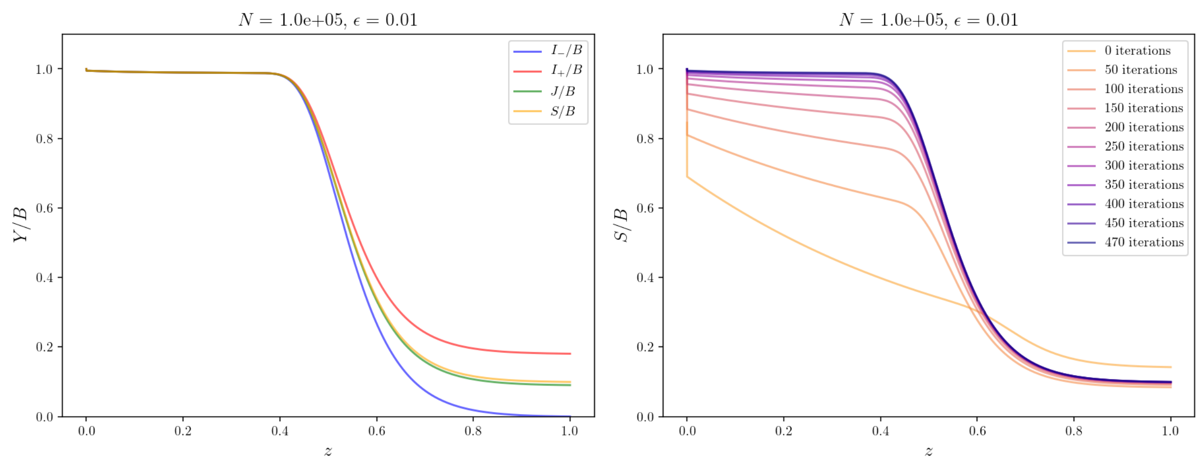
In [3]:
```python
# Input params
max_iters_long=1000
grid_num=int(1e5)
epsilon=0.01

# Run the simulation
ts_small_eps_long = TwoStream(max_iters=max_iters_long, grid_num=grid_nu
m, epsilon=epsilon)
results = ts_small_eps_long.iterate(break_if_converged=True) # Exit the
 loop if we reach convergence.
```

```
 47%|████▌     | 469/1000 [05:58<07:50,  1.13it/s]

Convergence criterion met within tolerance level of 1.00e-06.
Exited early after 469 iterations...
```

In [16]:
```python
# Plot the results
fig_small_eps_long, axl_small_eps_long, axr_small_eps_long = plot(ts_sma
ll_eps_long)
```



So we see from above that with $\epsilon = 0.01$, we need about 470 iterations to reach the convergence criterion.

In [ ]: