Wednesday, January 29, 2025 2:30 PM

1. 1D, Mono-energetic, Purely-Absorbing Transport Derivations: The 1D Cartesian transport equation with no scattering and a constant source is

$$\mu\frac{d\psi}{dx} + \Sigma_t\psi = Q \tag{1}$$

and has the general solution of

$$\psi(x) = \psi_i e^{-\frac{\Sigma_t}{\mu}(x-x_i)} + \frac{Q}{\Sigma_t}\left(1 - e^{-\frac{\Sigma_t}{\mu}(x-x_i)}\right) \tag{2}$$

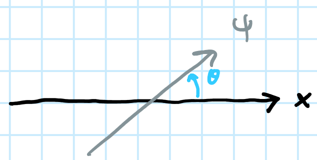
(a) In your own words, define each term in this equation and solution:
- μ:
- x:
- ψ:
- Σ_t:
- Q:
- ψ_i:
- x_i:

(b) Derive the solution for the average angular flux over a given distance in the x-direction, $\langle\psi\rangle = \frac{1}{x_e - x_i}\int_{x_i}^{x_e}\psi(x)dx$, and put it in the form of: $\langle\psi\rangle = A + B\left[\psi(x_e) - \psi(x_i)\right]$, where A and B are constants. Hint: using a coefficient, $\tau = \frac{\Sigma_t(x_e - x_i)}{\mu}$, and the definition of the exiting angular flux, $\psi(x_e)$, will help.

(c) Is your derivation valid for neutrons traveling in both the positive x direction, where $x_e > x_i$ and $\mu > 0$, and the negative x direction, where $x_e < x_i$ and $\mu < 0$?

a) $\mu = \cos\theta$

where θ is the angle between a flux and the 1D direction we are solving.



b) x is the direction for which we are solving the flux. The material is homogenous in all directions perpendicular to $\vec{x}$.

c) ψ is the angular flux in units of something like $\left[\frac{\text{particles}}{cm^2 \cdot s \cdot eV \cdot sr}\right]$

So it is the amount of particles traveling

So it is the amount of particles traveling through a unit area per unit time in a specific direction and energy.

d) Σ_t is the total macroscopic cross section, which is the avg. # of reactions per particle per unit track length, where a reaction removes the particle from $\psi(r, \Omega, E, t)$.

e) Q is the source of neutrons per unit volume in the energy and direction of ψ

f) ψ_i is the flux at x_i, which is the inlet, $\psi_i = \psi(x_i)$ in the direction and energy of the flux. (Boundary condition)

g) x_i is the location of the inlet

//

5)
$$\langle\psi\rangle \cdot (x_e - x_i) = \int_{x_i}^{x_e} \psi(x)\,dx$$

$$= \int_{x_i}^{x_e} \psi_i e^{-\frac{\Sigma_t}{\mu}(x-x_i)}\,dx + \frac{Q}{\Sigma_t}\int_{x_i}^{x_e} \left(1 - e^{-\frac{\Sigma_t}{\mu}(x-x_i)}\right)dx$$

$$= \frac{Q}{\Sigma_t}\int_{x_i}^{x_e} dx + \left(\psi_i - \frac{Q}{\Sigma_t}\right)\underbrace{\int_{x_i}^{x_e} e^{-\frac{\Sigma_t}{\mu}(x-x_i)}\,dx}$$

$u = x - x_i$ $\quad du = dx$

$u_i = 0$ $\quad u_e = x_e - x_i$

$$\int_0^{u_e} e^{-\frac{\Sigma_t}{\mu}u}\,du = \left[-\frac{\mu}{\Sigma_t} e^{-\frac{\Sigma_t}{\mu}u}\right]_{u=0}^{u=x_e-x_i}$$

$$= -\frac{\mu}{\Sigma_t}\left(e^{-\frac{\Sigma_t}{\mu}(x_e-x_i)} - 1\right)$$

$$= \frac{Q}{\Sigma_t}(x_e - x_i) + \left(\psi_i - \frac{Q}{\Sigma_t}\right)\frac{\mu}{\Sigma_t}\left(1 - e^{-\tau}\right)$$

$$\langle\psi\rangle = \frac{Q}{\varepsilon_t} + \frac{\mu}{\varepsilon_t}\left(\psi_i - \frac{Q}{\varepsilon_t}\right)\cdot\left(\frac{1}{x_e - x_i}\right)\left(1 - e^{-\tau}\right)$$

Let $\psi_e = \psi(x_e) = \psi_i e^{-\frac{\varepsilon_t}{\mu}(x_e - x_i)} + \frac{Q}{\varepsilon_t}\left(1 - e^{-\frac{\varepsilon_t}{\mu}(x_e - x_i)}\right)$

$$\psi_e = \frac{Q}{\varepsilon_t} + \left[\psi_i - \frac{Q}{\varepsilon_t}\right)e^{-\tau}$$

$$\left(\psi_i - \frac{Q}{\varepsilon_t}\right)\left(1 - e^{-\tau}\right)$$

$$= \psi_i - \psi_i e^{-\tau} - \frac{Q}{\varepsilon_t} + \frac{Q}{\varepsilon_t}e^{-\tau}$$

$$= \psi_i - \left[\frac{Q}{\varepsilon_t} + \left(\psi_i - \frac{Q}{\varepsilon_t}\right)e^{-\tau}\right]$$

$$= \psi_i - \psi_e$$

$$\langle\psi\rangle = \frac{Q}{\varepsilon_t} + \frac{\mu}{\varepsilon_t(x_e - x_i)}(\psi_i - \psi_e)$$

$$A = \frac{Q}{\varepsilon_t} \qquad B = \frac{\mu}{\varepsilon_t(x_i - x_e)}$$

$$\langle\psi\rangle = A + B(\psi_e - \psi_i)$$

//

c) Yes, the derivation is agnostic of sign.
You just have to be careful with the sign.

2abd

Wednesday, January 29, 2025 4:03 PM

2. 1D, Mono-energetic, Purely-Absorbing, Source-Free Transport Coding:

- The source, Q, is zero everywhere.
- The total cross section, Σ_t in $\frac{reactions}{neutron-cm}$, is uniform in space.
- The problem domain is finite in one dimension, $x \in [0, X]$, with N_x equal-sized mesh cells ($\Delta x = \frac{X}{N_x}$)
- The radiation can travel in any number of polar directions $\theta \in [0, \pi]$, or $\mu = cos(\theta) \in [-1, 1]$, except $\mu = 0$.
- The unknown values include the angular flux in each direction at each node, $\psi(x_i, \mu)$, where $x_i = i\Delta x$ for $i \in [0, N_x]$, and the average angular flux in a mesh cell, $\langle\psi\rangle_{i,\mu}$ for $x \in [x_{i-1}, x_i]$ and $i \in [1, N_x]$.

- Because the scalar flux is the integral of all angular fluxes, $\phi(x) = \int_{-1}^{1} \psi(x, \mu) d\mu$, and the angular flux is composed of only two directions: one positive in x ($\mu > 0$) and one negative in x ($\mu < 1$), so that $\phi(x) = \sum_{\mu} \psi(x, \mu)$, we can compute the average scalar flux in the cell with a simple addition of the two average angular fluxes.
- This leads to two independent equations that must be solved, followed by three computations for the average flux:

$$\mu\frac{d\psi_+}{dx} + \Sigma_t\psi_+ = 0 \quad \psi(0, \mu > 0) = \psi_+^L \tag{3}$$
$$\mu\frac{d\psi_-}{dx} + \Sigma_t\psi_- = 0 \quad \psi(X, \mu < 0) = \psi_-^R \tag{4}$$
$$\langle\psi\rangle_{i,+} = A + Bfunc(\psi_+) \qquad i \in [1, N_x] \tag{5}$$
$$\langle\psi\rangle_{i,-} = A + Bfunc(\psi_-) \qquad i \in [1, N_x] \tag{6}$$
$$\langle\phi\rangle_i = \langle\psi\rangle_{i,+} + \langle\psi\rangle_{i,-} \qquad i \in [1, N_x] \tag{7}$$

- or for all $i \in [1, N_x]$:

$$\psi_+(x_i) = \psi_+(x_{i-1})e^{-\tau} \tag{8}$$
$$\psi_-(x_{i-1}) = \psi_-(x_i)e^{-\tau} \tag{9}$$
$$\langle\phi\rangle_i = \langle\psi\rangle_{i,+} + \langle\psi\rangle_{i,-} \tag{10}$$

(a) Why, in the problem specification, do we not solve for the angular flux with $\mu = 0$?

(b) Single, positive direction: For a single direction (μ) going in the positive x direction, $\mu \in (0, 1]$, with a known boundary condition, $\psi(x_b, \mu)$, on the left side of the problem ($x_b = 0$). Describe the $Ax = b$ problem that could be used to solve for the angular flux at each node, $\psi(x_i, \mu)$ for $i \in [1, N_x]$. What does the matrix, A, look like? What does the vector b look like? Where do you put the left boundary condition, $\psi(0, \mu)$?

(c) Write a code to solve for the right-going angular flux at each node, $\psi(x_i, \mu)$ for $i \in [1, N_x]$, and the average flux, $\langle\psi\rangle_{i,\mu}$, in each mesh cell for a given direction, $\mu \in (0, 1]$.

(d) Single, negative direction: For a single direction (μ) going in the negative x direction, $\mu \in [-1, 0)$, with a known boundary condition, $\psi(x_b, \mu)$, on the right side of the problem ($x_b = X$). Describe the $Ax = b$ problem that could be used to solve for the angular flux at each node, $\psi(x_i, \mu)$ for $i \in [0, N_x - 1]$. What does the matrix, A, look like? What does the vector b look like? Where do you put the right boundary condition, $\psi(X, \mu)$?

(e) Write a code to solve for the left-going angular flux at each node, $\psi(x_i, \mu)$ and the average flux, $\langle\psi\rangle_{i,\mu}$, in each mesh cell for a given direction, $\mu \in [-1, 0)$.

(f) Modify your code to be able to solve for the average scalar flux in each mesh cell and plot the angular fluxes and scalar flux on the mesh.

a) if $\mu = 0$:

$$0 \cdot \frac{d\psi}{dx} + \Sigma_t \psi = 0$$

$$\Sigma_t \psi = 0$$

$$\psi = 0$$

Only the trivial solution exists.

We are solving in 1D so a flux perpendicular to the direction of interest is irrelevant.

b) $\psi_+(x_0 = 0) = \psi_+^L$ $\quad \tau = \frac{\Sigma_t \Delta x}{\mu}$

$\psi_+(x_i) = \psi_+(x_{i-1}) e^{-\tau}$

$\hookrightarrow \psi_+(x_i) - \psi_+(x_{i-1}) e^{-\tau} = 0$

for $i = 0, 1, 2$

$$\begin{bmatrix} 1 & 0 & 0 \\ -e^{-\tau} & 1 & 0 \\ 0 & -e^{-\tau} & 1 \end{bmatrix} \begin{bmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} \psi_+^L \\ 0 \\ 0 \end{bmatrix}$$

d) $\psi_-(x_{i-1}) - \psi_-(x_i) e^{-\tau} = 0$

$\psi_-(x_n) = \psi_R$

$$\begin{bmatrix} 1 & -e^{-\tau} & 0 \\ 0 & 1 & -e^{-\tau} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \psi_R \end{bmatrix}$$

2c

Thursday, January 30, 2025 11:30 AM

2c

2c

January 30, 2025

[1]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
```

[2]:
```
sigma_t = 1
x_left_boundary = 0
x_right_boundary = 1
mu = 1

psi_left_initial = 1
```

[3]:
```
number_of_nodes = 10
x = np.linspace(x_left_boundary, x_right_boundary, number_of_nodes)
delta_x = x[1] - x[0]

tau_coeff = sigma_t * (delta_x) / mu
exp_term = -np.exp(-tau_coeff)

A_mat = sp.sparse.diags([1, exp_term], [0, -1], shape=(number_of_nodes,␣
↪number_of_nodes), format='csc')
b_vec = [psi_left_initial] + [0] * (number_of_nodes - 1)

flux_sol = sp.sparse.linalg.spsolve(A_mat, b_vec)
```

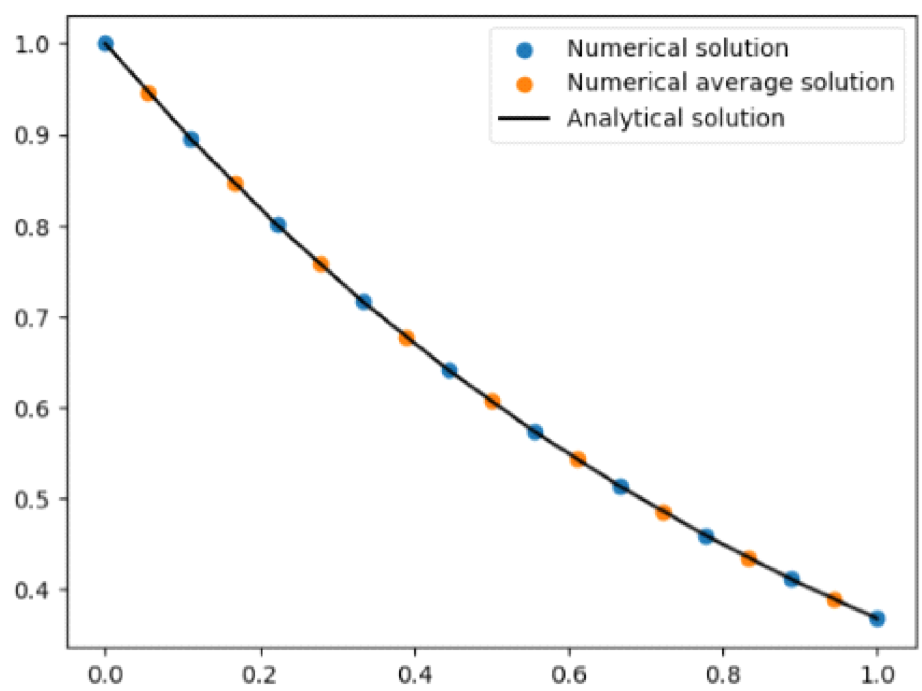
[4]:
```
A_coeff = 0
B_coeff = lambda xi, xe : mu / (sigma_t * (xi - xe))

x_average = np.zeros(number_of_nodes-1)
flux_average = np.zeros(number_of_nodes-1)
for i in range(1, number_of_nodes):
    x_left = x[i-1]
    x_right = x[i]

    x_average[i-1] = (x_left + x_right) / 2

    flux_left = flux_sol[i-1]
    flux_right = flux_sol[i]
```

```
    flux_average[i-1] = A_coeff + B_coeff(x_left, x_right) * (flux_right -␣
↪flux_left)

flux_average
```

[4]:
```
array([0.94644615, 0.84691723, 0.75785483, 0.6781583 , 0.60684271,
       0.54302672, 0.48592165, 0.4348218 , 0.38909564])
```

[5]:
```
fig, ax = plt.subplots()
ax.scatter(x, flux_sol, label='Numerical solution')
ax.scatter(x_average, flux_average, label='Numerical average solution')

analytical_sol = lambda x: psi_left_initial * np.exp(-sigma_t * (x -␣
↪x_left_boundary) / mu)
ax.plot(x, analytical_sol(x), label='Analytical solution', color='black')

ax.legend()
```

[5]:
```
<matplotlib.legend.Legend at 0x169dc8a0b30>
```



2e

Thursday, January 30, 2025 11:37 AM

2e

2e

January 30, 2025

[6]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
```

[7]:
```
sigma_t = 1
x_left_boundary = 0
x_right_boundary = 1
mu = -1

psi_right_initial = 1
```

[8]:
```
number_of_nodes = 10
x = np.linspace(x_left_boundary, x_right_boundary, number_of_nodes)
delta_x = x[0] - x[1]

tau_coeff = sigma_t * (delta_x) / mu
exp_term = -np.exp(-tau_coeff)

A_mat = sp.sparse.diags([1, exp_term], [0, 1], shape=(number_of_nodes,␣
 ↪number_of_nodes), format='csc')
b_vec = [0] * (number_of_nodes - 1) + [psi_right_initial]

flux_sol = sp.sparse.linalg.spsolve(A_mat, b_vec)
```

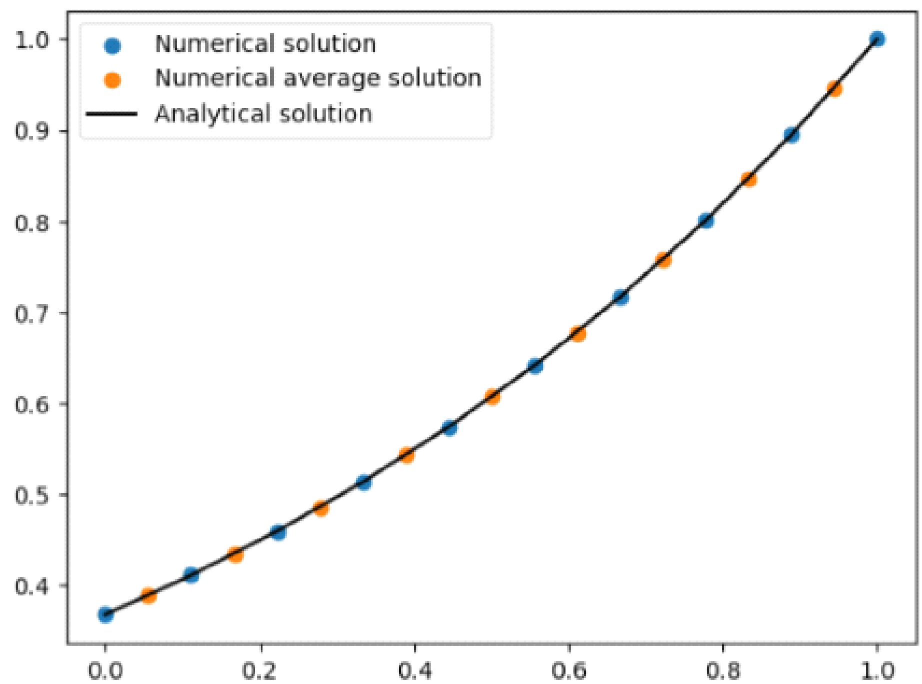
[9]:
```
A_coeff = 0
B_coeff = lambda xi, xe : mu / (sigma_t * (xi - xe))

x_average = np.zeros(number_of_nodes-1)
flux_average = np.zeros(number_of_nodes-1)
for i in range(1, number_of_nodes):
    x_left = x[i-1]
    x_right = x[i]

    x_average[i-1] = (x_left + x_right) / 2

    flux_left = flux_sol[i-1]
    flux_right = flux_sol[i]
```

```
    flux_average[i-1] = A_coeff + B_coeff(x_left, x_right) * (flux_right -
 ↪flux_left)

flux_average
```

[9]: array([0.38909564, 0.4348218 , 0.48592165, 0.54302672, 0.60684271,
       0.6781583 , 0.75785483, 0.84691723, 0.94644615])

[10]:
```
fig, ax = plt.subplots()
ax.scatter(x, flux_sol, label='Numerical solution')
ax.scatter(x_average, flux_average, label='Numerical average solution')

analytical_sol = lambda x: psi_right_initial * np.exp(-sigma_t * (x -
 ↪x_right_boundary) / mu)
ax.plot(x, analytical_sol(x), label='Analytical solution', color='black')

ax.legend()
```

[10]: <matplotlib.legend.Legend at 0x1c3ed661d00>



2f

Thursday, January 30, 2025 11:38 AM

2f

2f

January 30, 2025

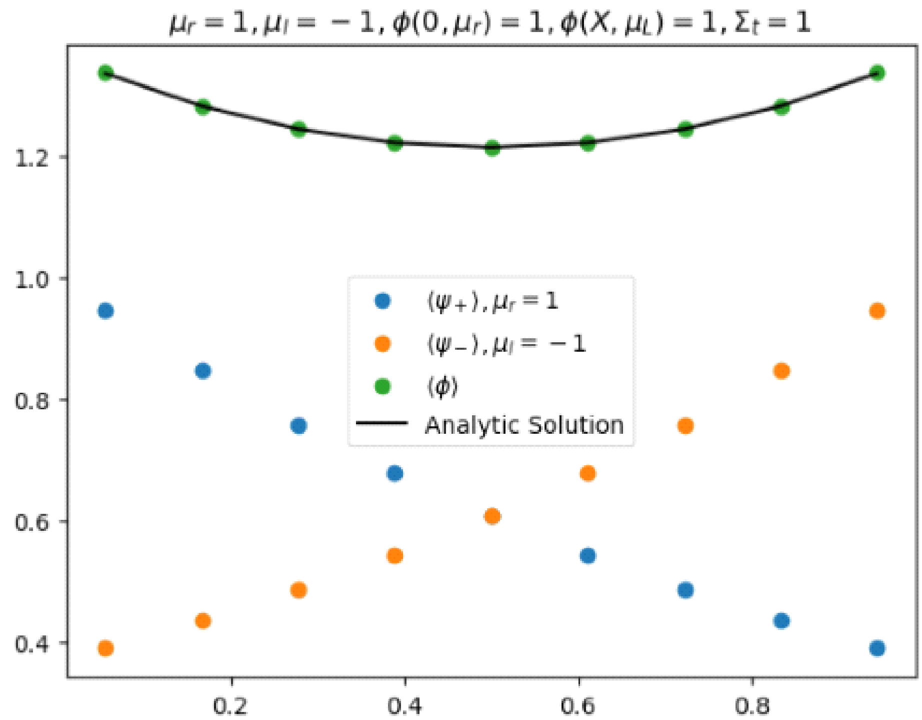
[90]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
```

[91]:
```
def angular_flux_one_direction(sigma_t=1, x_start=0, x_end=1, mu=1,
 ↪psi_initial=1, number_of_nodes=10):
    assert x_start < x_end, "x_start must be less than x_end"
    assert mu != 0, "mu cannot be zero"

    x = np.linspace(x_start, x_end, number_of_nodes)
    delta_x = x[1] - x[0] if mu > 0 else x[0] - x[1]

    tau_coeff = sigma_t * delta_x / mu
    exp_term = np.exp(-tau_coeff)

    diag_index = -1 if mu > 0 else 1

    A = sp.sparse.diags([1, -exp_term], [0, diag_index],
 ↪shape=(number_of_nodes, number_of_nodes), format='csc')

    if mu > 0:
        b_vec = [psi_initial] + [0] * (number_of_nodes - 1)
    elif mu < 0:
        b_vec = [0] * (number_of_nodes - 1) + [psi_initial]

    angular_flux_sol = sp.sparse.linalg.spsolve(A, b_vec)

    # calculate average
    A_coeff = 0
    B_coeff = lambda xi, xe: mu / (sigma_t * (xi - xe))

    x_average = np.zeros(number_of_nodes-1)
    flux_average = np.zeros(number_of_nodes-1)
    for i in range(1, number_of_nodes):
        x_left = x[i-1]
        x_right = x[i]
        x_average[i-1] = (x_left + x_right) / 2
```

```
        flux_left = angular_flux_sol[i-1]
        flux_right = angular_flux_sol[i]
        flux_average[i-1] = A_coeff + B_coeff(x_left, x_right) * (flux_right -
 ↪flux_left)

    return x, angular_flux_sol, x_average, flux_average
```

[92]:
```
sigma_t = 1
mu_r = 1
mu_l = -1
phi_r = 1
phi_l = 1

# def angular_flux(mu_r, mu_l, phi_r, phi_l, sigma_t=1):
# r and l mean going the flux is going in the right or left direction
# so r corresponds to the left boundary

_, _, x_pos, flux_pos = angular_flux_one_direction(
    sigma_t=sigma_t, x_start=0, x_end=1, mu=mu_r, psi_initial=phi_r
)

_, _, x_neg, flux_neg = angular_flux_one_direction(
    sigma_t=sigma_t, x_start=0, x_end=1, mu=mu_l, psi_initial=phi_l
)

# assert x_pos == x_neg

pos_analytic = lambda x: phi_r * np.exp(-sigma_t * (x - 0) / mu_r)
neg_analytic = lambda x: phi_l * np.exp(-sigma_t * (x - 1) / mu_l)

flux_analytic = lambda x: pos_analytic(x) + neg_analytic(x)

fig, ax = plt.subplots()
ax.scatter(x_pos, flux_pos, label=rf"$\langle \psi_+ \rangle, \mu_r = {mu_r}$")
ax.scatter(x_neg, flux_neg, label=rf"$\langle \psi_- \rangle, \mu_l = {mu_l}$")
ax.scatter(x_pos, flux_pos + flux_neg, label=r"$\langle \phi \rangle$")
ax.plot(x_pos, flux_analytic(x_pos), label="Analytic Solution", color="black")
ax.legend()

ax.set_title(
    rf"$\mu_r = {mu_r}, \mu_l = {mu_l}, \phi(0,\mu_r) = {phi_r}, \phi(X,\mu_L)
 ↪= {phi_l}, \Sigma_t = {sigma_t}$"
)
```

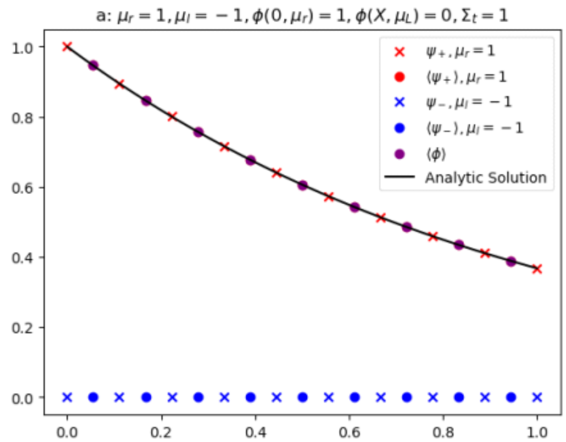
[92]:
```
Text(0.5, 1.0, '$\\mu_r = 1, \\mu_l = -1, \\phi(0,\\mu_r) = 1, \\phi(X,\\mu_L) =
1, \\Sigma_t = 1$')
```

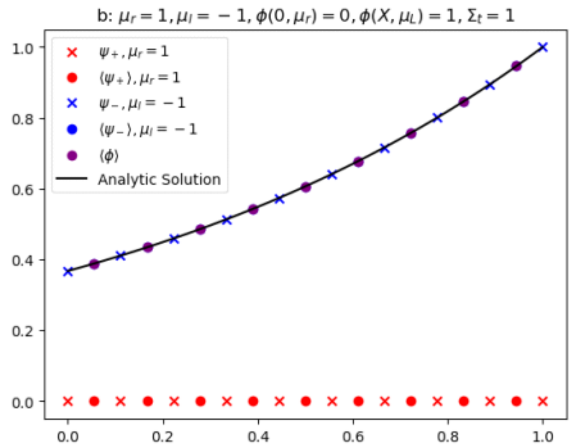

$\mu_r = 1, \mu_l = -1, \phi(0, \mu_r) = 1, \phi(X, \mu_L) = 1, \Sigma_t = 1$
$\langle\psi_+\rangle, \mu_r = 1$
$\langle\psi_-\rangle, \mu_l = -1$
$\langle\phi\rangle$
Analytic Solution
1.2
1.0
0.8
0.6
0.4
0.2
0.4
0.6
0.8

3

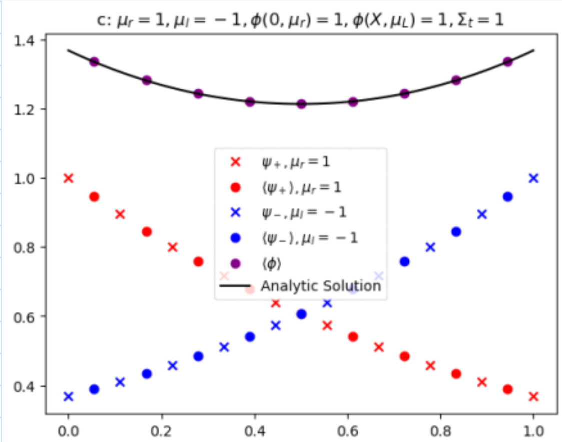
Thursday, January 30, 2025 1:50 PM

3. Code Application: Use your code to solve for and plot the angular flux at each node, and average scalar flux, $\langle \phi \rangle$, in each mesh cell for the problems in Table 3. Note, you can use the analytic solution to check your work.
   (a) Right-going angular flux.
   (b) Left-going angular flux; is the solution the opposite of the previous problem?
   (c) Two angular fluxes; is the solution the sum of the previous problems?
   (d) Two angular fluxes, but going less perpendicular with the direction of travel; what impact did changing the angle if the radiation have on the solution?
   (e) Larger cross section; what change did that make on the solution?
   (f) Reduced cross section, how does that compare with problem 1?
   (g) Zero cross section; what is happening in this problem?

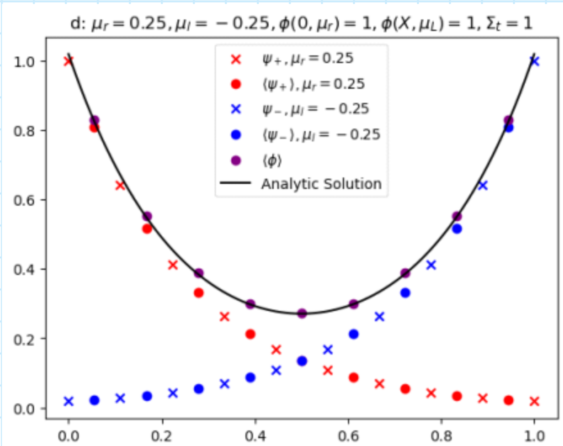




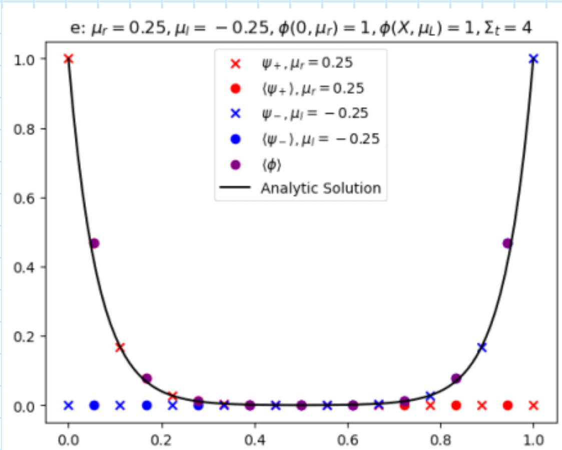
b) The solution is opposite of the previous problem



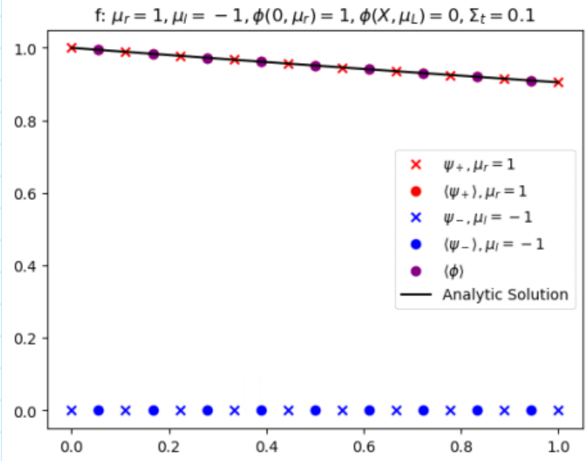
c) The solution is the sum of the previous problem.



d) As the angle gets steeper, the flux travels for more total distance per distance in the x direction, so a particle is more likely to undergo and interaction per unit path length in x. That is, the flux has a steeper rate of decay when the angle is less parallel with the direction being solved for.

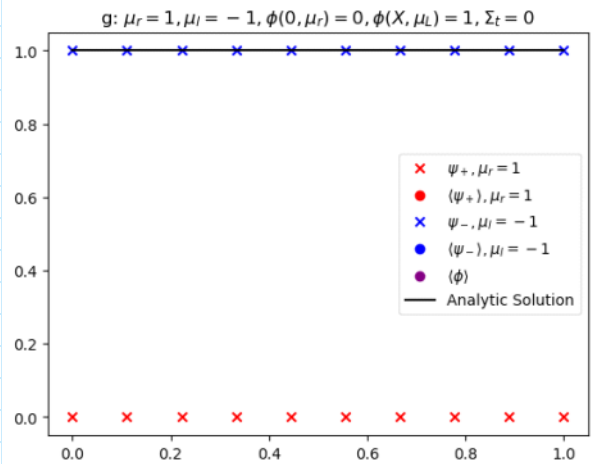


e) As the cross section increases, more particles are removed per unit traveled, so the rate of decay is greater with a larger cross section.



f) Vice versa above: As the cross section decreases, it is less likely that a particle will be removed from the flux per unit track length, so the rate of decay of flux is less.

The rate of decay is less than problem 1 since the cross section is less.



g) When the cross section is zero, no particles will be removed from the flux per unit track length, so the flux stays the same as the initial. This is equivalent to a vacuum.

Note: Averaging breaks down with Sigma_t = 0.

4. 1D, Mono-energetic, with Scattering Transport Coding: In this example, we still have only two directions of travel, but all reactions are scattering, which leads to coupled equations:

$$\psi_+(x_i) = \psi_+(x_{i-1})e^{-\tau} + \frac{Q_i}{\Sigma_t}\left(1-e^{-\tau}\right) \quad (11)$$

$$\psi_-(x_{i-1}) = \psi_-(x_i)e^{-\tau} + \frac{Q_i}{\Sigma_t}\left(1-e^{-\tau}\right) \quad (12)$$

$$Q_i = \frac{\Sigma_s}{2}\langle\phi\rangle_i = \frac{\Sigma_s}{2}\left[\langle\psi\rangle_{i,\mu>0} + \langle\psi\rangle_{i,\mu<0}\right] \quad (13)$$

(a) Modify your code to be able to solve the problem with two directions and scattering. You may need to create an iteration loop.

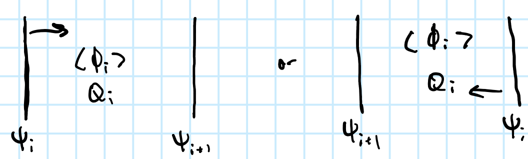
(b) For $\Sigma_s = 0.1$, solve the previous (a-g) problems and discuss the differences.

Going right

$$\psi_+(x_i) - \psi_+(x_{i-1})e^{-\tau} = \frac{Q}{\varepsilon_\tau}(1-e^{-\tau})$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -e^{-\tau} & 1 & 0 \\ 0 & -e^{-\tau} & 1 \end{bmatrix}\begin{bmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} \psi_L \\ \frac{Q_1}{\varepsilon_t}(1-e^{-\tau}) \\ \frac{Q_2}{\varepsilon_t}(1-e^{-\tau}) \end{bmatrix}$$

where $Q_i = \frac{\varepsilon_s}{2}\langle\phi_i\rangle$



3

The solutions are mostly the same, but there are some noticeable small additions to a flux with a low magnitude when the other flux has a high magnitudes.

3

January 30, 2025

[39]:
```python
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
```

[40]:
```python
class AngularFlux:
    def __init__(
        self,
        mu_r,
        mu_l,
        phi_r,
        phi_l,
        sigma_t=1,
        sigma_s=0,
        title_start="",
        x_start=0,
        x_end=1,
        n_surfaces=10,
    ):
        self.mu_r = mu_r
        self.mu_l = mu_l
        self.phi_r = phi_r
        self.phi_l = phi_l
        self.title_start = title_start
        self.sigma_t = sigma_t
        self.sigma_s = sigma_s
        self.x_start = x_start
        self.x_end = x_end
        self.n_surfaces = n_surfaces
        self.n_cells = n_surfaces - 1

        assert x_start < x_end, "x_start must be less than x_end"
        assert mu_r != 0, "mu_r cannot be zero"
        assert mu_l != 0, "mu_l cannot be zero"

        self.surface_x = np.linspace(x_start, x_end, n_surfaces)
        delta_x = self.surface_x[1] - self.surface_x[0]
        self.cell_x = np.linspace(
            x_start + delta_x / 2, x_end - delta_x / 2, self.n_cells
```

```python
        )

        self.rightward_angular_flux = np.zeros(n_surfaces)
        self.leftward_angular_flux = np.zeros(n_surfaces)

        self.rightward_average_angular_flux = np.zeros(self.n_cells)
        self.leftward_average_angular_flux = np.zeros(self.n_cells)
        self.average_scalar_flux = np.ones(self.n_cells)

    def angular_flux_one_direction(
        self,
        mu=1,
        psi_initial=1,
        # sigma_t=1, x_start=0, x_end=1, mu=1, psi_initial=1, n_surfaces=10
    ):

        x = self.surface_x
        delta_x = x[1] - x[0] if mu > 0 else x[0] - x[1]

        tau_coeff = self.sigma_t * delta_x / mu
        exp_term = np.exp(-tau_coeff)

        diag_index = -1 if mu > 0 else 1

        A = sp.sparse.diags(
            [1, -exp_term],
            [0, diag_index],
            shape=(self.n_surfaces, self.n_surfaces),
            format="csc",
        )

        cell_sources = [
            self.sigma_s / 2 * scalar_flux for scalar_flux in self.
↪average_scalar_flux
        ]
        # TODO tau isn't constant for variable material properties
        scatter_source = [
            cell_source / self.sigma_t * (1 - exp_term) for cell_source in␣
↪cell_sources
        ]

        if mu > 0:
            b_vec = [psi_initial] + scatter_source
        elif mu < 0:
            b_vec = scatter_source + [psi_initial]

        angular_flux_sol = sp.sparse.linalg.spsolve(A, b_vec)
```

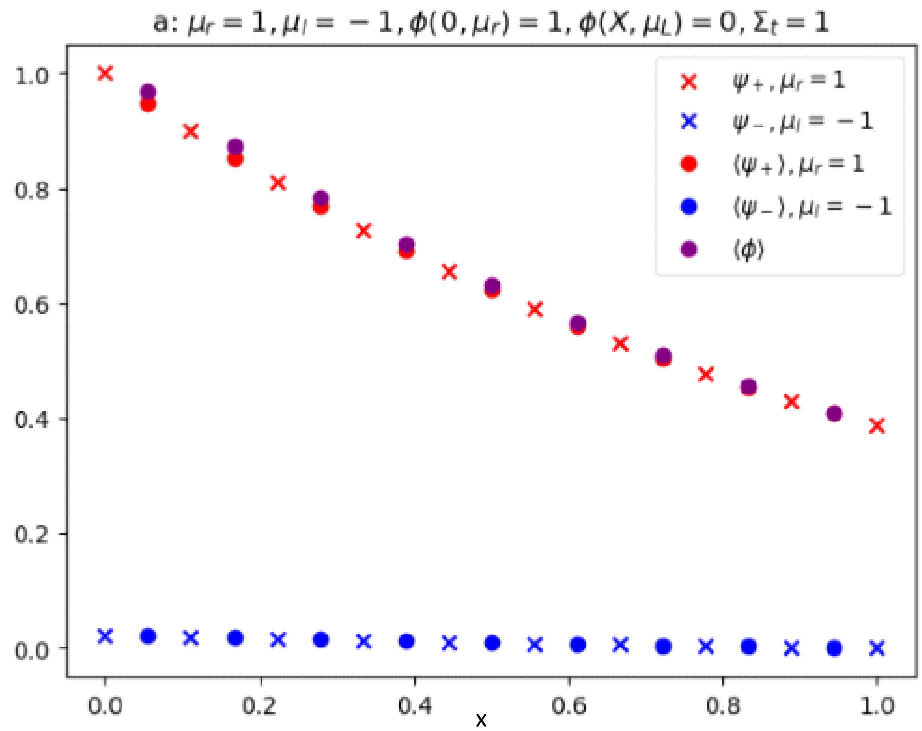
```
        # calculate average
        A_coeff = lambda i: cell_sources[i] / self.sigma_t
        B_coeff = lambda xi, xe: mu / (self.sigma_t * (xi - xe))

        x_average = np.zeros(self.n_cells)
        flux_average = np.zeros(self.n_cells)
        for i in range(1, self.n_surfaces):
            x_left = x[i - 1]
            x_right = x[i]
            x_average[i - 1] = (x_left + x_right) / 2

            flux_left = angular_flux_sol[i - 1]
            flux_right = angular_flux_sol[i]
            flux_average[i - 1] = A_coeff(i - 1) + B_coeff(x_left, x_right) * (
                flux_right - flux_left
            )

        return angular_flux_sol, flux_average

    def angular_flux(self, max_iter=1000, tol=1e-6):
        # r and l mean going the flux is going in the right or left direction
        # so r corresponds to the left boundary

        for iter in range(max_iter):
            old_scalar_flux = self.average_scalar_flux.copy()

            (
                self.rightward_angular_flux,
                self.rightward_average_angular_flux,
            ) = self.angular_flux_one_direction(mu=self.mu_r, psi_initial=self.
↪phi_r)

            self.leftward_angular_flux, self.leftward_average_angular_flux = (
                self.angular_flux_one_direction(mu=self.mu_l, psi_initial=self.
↪phi_l)
            )

            self.average_scalar_flux = (
                self.leftward_average_angular_flux + self.
↪rightward_average_angular_flux
            )

            if np.allclose(old_scalar_flux, self.average_scalar_flux, atol=tol):
                print(f"{self.title_start}: Converged after {iter} iterations")
                break
```

```python
        fig, ax = plt.subplots()

        ax.scatter(
            self.surface_x,
            self.rightward_angular_flux,
            label=rf"$\psi_+, \mu_r = {self.mu_r}$",
            color="red",
            marker="x",
        )
        ax.scatter(
            self.surface_x,
            self.leftward_angular_flux,
            label=rf"$\psi_-, \mu_l = {self.mu_l}$",
            color="blue",
            marker="x",
        )

        ax.scatter(
            self.cell_x,
            self.rightward_average_angular_flux,
            label=rf"$\langle \psi_+ \rangle, \mu_r = {self.mu_r}$",
            color="red",
        )
        ax.scatter(
            self.cell_x,
            self.leftward_average_angular_flux,
            label=rf"$\langle \psi_- \rangle, \mu_l = {self.mu_l}$",
            color="blue",
        )
        ax.scatter(
            self.cell_x,
            self.average_scalar_flux,
            label=r"$\langle \phi \rangle$",
            color="purple",
        )

        ax.legend()

        ax.set_title(
            rf"{self.title_start}: $\mu_r = {self.mu_r}, \mu_l = {self.mu_l}, 
↪\phi(0,\mu_r) = {self.phi_r}, \phi(X,\mu_L) = {self.phi_l}, \Sigma_t = {self.
↪sigma_t}$"
        )

        return fig, ax
```

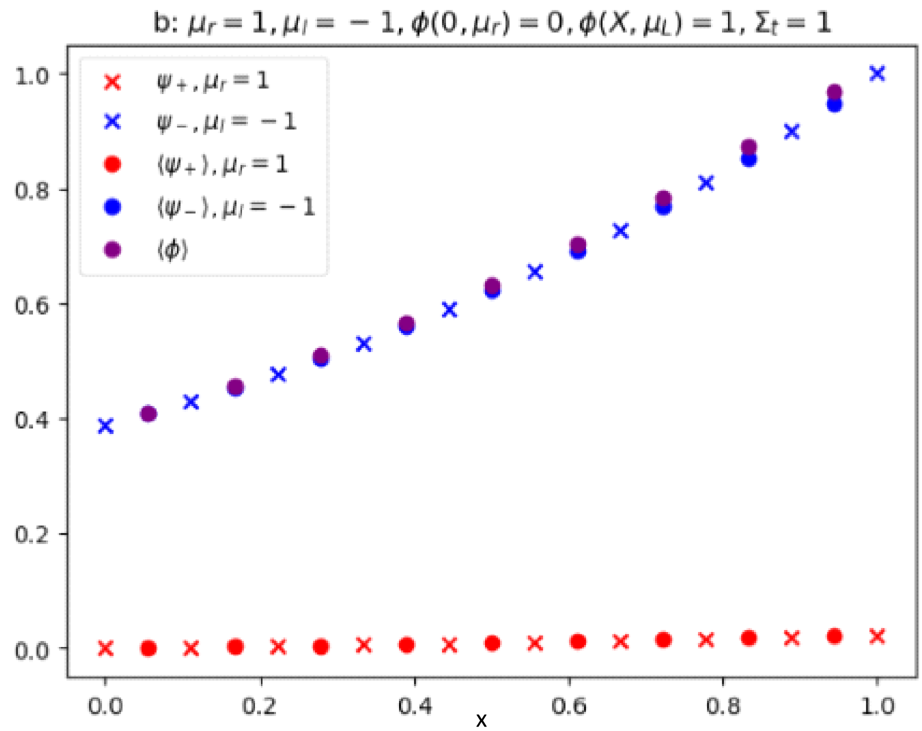
4

```python
sigma_s = 0.1
a = [1, -1, 1, 0, 1, sigma_s, "a"]
b = [1, -1, 0, 1, 1, sigma_s, "b"]
c = [1, -1, 1, 1, 1, sigma_s, "c"]
d = [0.25, -0.25, 1, 1, 1, sigma_s, "d"]
e = [0.25, -0.25, 1, 1, 4, sigma_s, "e"]
f = [1, -1, 1, 0, 0.1, sigma_s, "f"]
g = [
    1,
    -1,
    0,
    1,
    1e-7,
    sigma_s,
    "g",
]  # can't actually have Sigma_t = 0 because of division by zero

for i in [a, b, c, d, e, f, g]:
    AngularFlux(*i).angular_flux()
```

```
a: Converged after 4 iterations
b: Converged after 4 iterations
c: Converged after 4 iterations
d: Converged after 5 iterations
e: Converged after 4 iterations
f: Converged after 2 iterations
g: Converged after 3 iterations
```

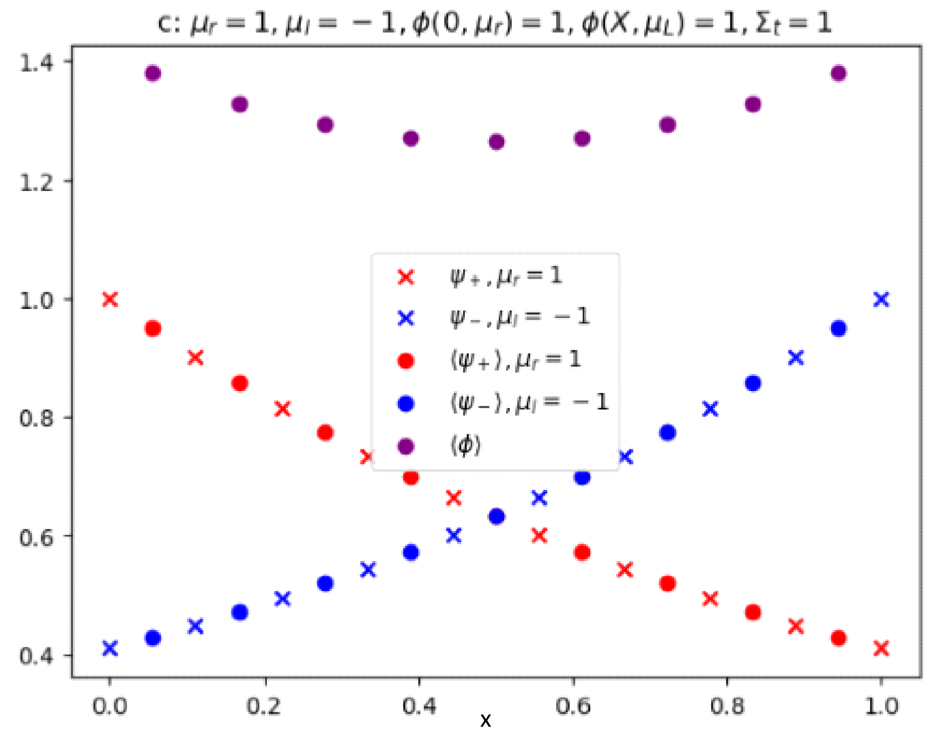


The leftward flux gets some "in scatter" from the right flux, causing the leftward flux to increase in magnitude.
As a results, the scalar flux magnitude is slightly higher, especially when the rightward flux is large (causing the leftward flux to increase).

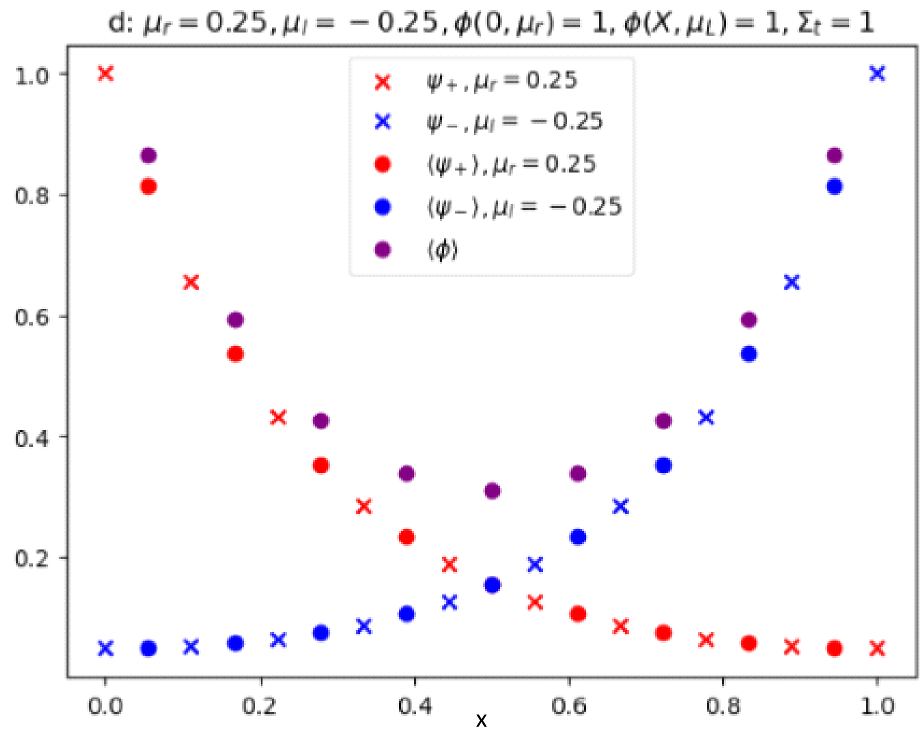


Same as above but vice versa

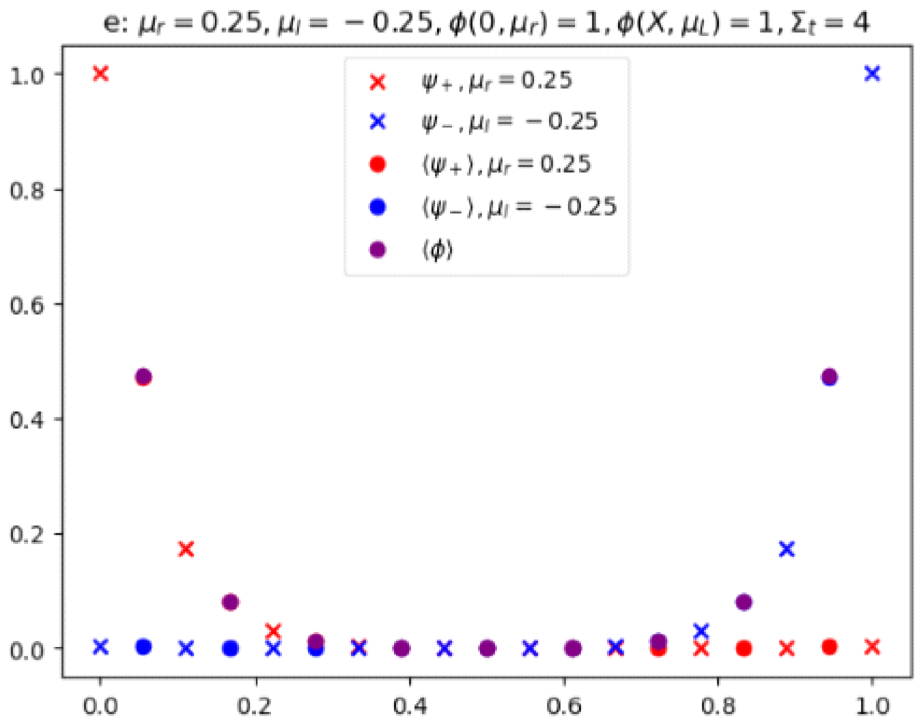
7



There isn't much of a noticeable difference in this one.

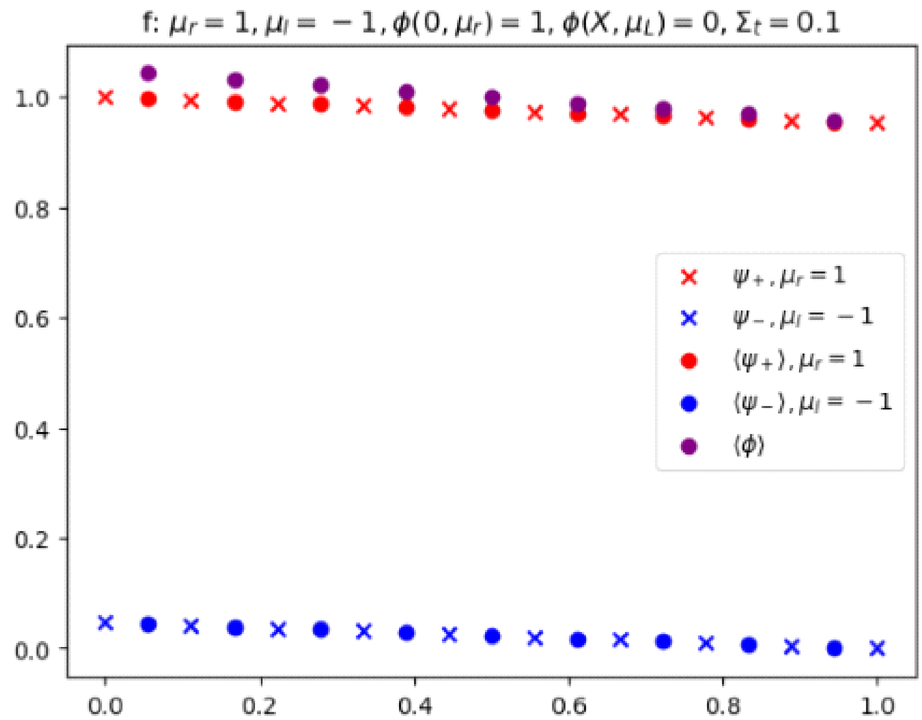


Difference is not very noticeable.

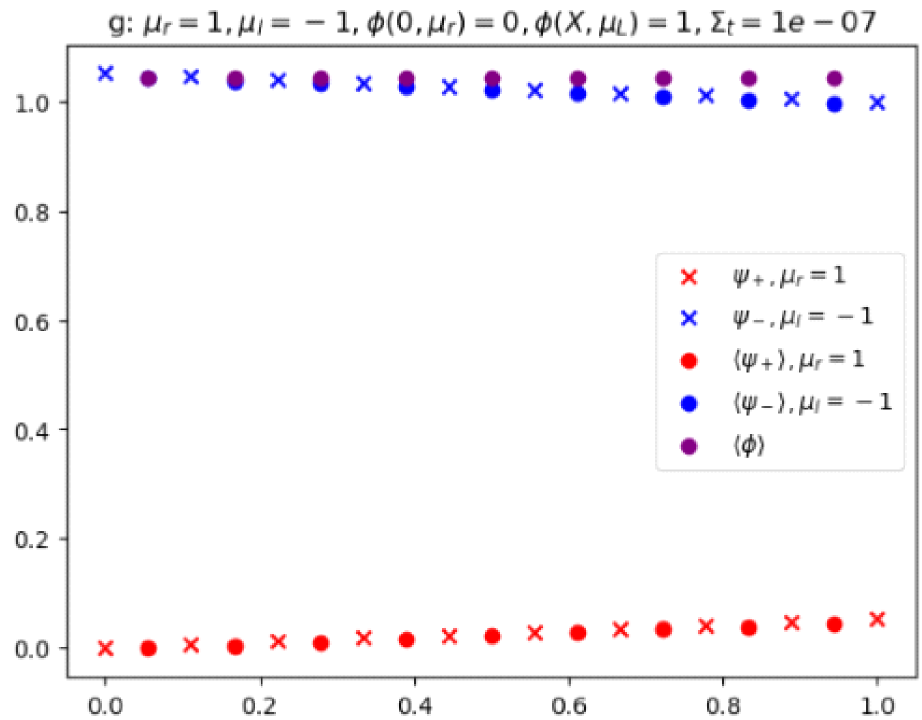


Difference is not very noticeable.

10



Same as a and b,
The high rightward flux causes a non zero leftward flux towards the left boundary, which Increases the scalar flux.



You can't calculate this with a true Sigma_t = 0 (also doesn't make much since to have Sigma_t=0
While Sigma_s = nonzero)..., but hypothetically you would see a flat rightward flux and a builtup
Of leftward flux with the highest value being on the right.