# One-Dimensional Neutron Transport Equation ($S_N$) Solver using the Discrete Ordinates Method

## Machine Learning Methods are used to reduce the data requirements of Scattering and Fission Cross-Sections

### Current Work

- ☐ Add Testing Functions with pytest
- ☐ Clean up `source.py` and test functions
- ☐ Make time-dependent source problems for `source.py`
- ☐ Make `class` to run through benchmark problems

### Current Data Saving Techniques

1. `dj_prob.py` incorporates Deep Jointly-Informed Neural Networks[1] (DJINN) into the $S_N$ code for $\Sigma_\mathrm{s}\phi$ and $\Sigma_\mathrm{f}\phi$ calculations.
2. `ae_prob.py` incorporates an autoencoder into the $S_N$ code for $\phi$, $\Sigma_\mathrm{s}\phi$, and $\Sigma_\mathrm{f}\phi$ to compress the energy groups and use in conjuction with DJINN.
3. `svd_prob.py` incorporates an SVD into the $S_N$ code for the $\Sigma_\mathrm{s}$ and $\Sigma_\mathrm{f}$ matrices.
4. `hybrid.py` separates the collided and uncollided terms to be used with different numbers of ordinates ($N$) and energy groups ($G$).

### Hybrid Method for Time Dependent Multigroup Problems

0. Initialize $\psi^n$ to zero
1. Calculate the uncollided $\psi_u^{n+1}$ and $\phi_u^{n+1}$ through the sweep

$$\Omega \cdot \nabla \psi_u^{n+1} + \left( \Sigma_\mathrm{t} + \frac{1}{v\Delta t} \right) \psi_u^{n+1} = Q_u + \frac{1}{v\Delta t} \psi_u^n \tag{1}$$

$$\begin{aligned} \psi_u^{n+1} \left( \frac{\mu_n}{\Delta x} + \frac{1}{2}\Sigma_\mathrm{t} + \frac{1}{2v\Delta t} \right) = \\ Q_u + \frac{1}{v\Delta t}\psi_u^n + \psi_u^{n+1} \left( \frac{\mu_n}{\Delta x} - \frac{1}{2}\Sigma_\mathrm{t} - \frac{1}{2v\Delta t} \right) \end{aligned} \tag{2}$$

2. Use $\phi_u^{n+1}$ to create source term ($Q_c$) for the collided equation:

$$Q_c = \Sigma_\mathrm{s}\phi_u^{n+1} + \Sigma_\mathrm{f}\phi_u^{n+1} \tag{3}$$

3. Solve the collided equation with the new source term ($Q_c$)

$$\Omega \cdot \nabla \psi_c^{n+1} + \left( \Sigma_\mathrm{t} + \frac{1}{v\Delta t} \right) \psi_c^{n+1} = \Sigma_\mathrm{s}\phi_c^{n+1} + \Sigma_\mathrm{f}\phi_c^{n+1} + Q_c \tag{4}$$

[1] K. D. Humbird, J. L. Peterson, and R. G. McClarren. "Deep neural network initialization with decision trees." *IEEE transactions on neural networks and learning systems*,volume 30(5), pp. 1286–1295 (2018)

4. Solve the angular flux for the next time step ($\psi^{n+2}$)

$$\begin{aligned}\Omega \cdot \nabla \psi_u^{n+2} + \left( \Sigma_{\mathrm{t}} + \frac{1}{v\Delta t} \right) \psi_u^{n+2} = \\ \Sigma_{\mathrm{s}}(\phi_c^{n+1} + \phi_u^{n+1}) + \Sigma_{\mathrm{f}}(\phi_c^{n+1} + \phi_u^{n+1}) + Q_u + \frac{1}{v\Delta t}\psi_c^{n+1}\end{aligned} \tag{5}$$

5. Repeat Steps 1-4 with the new angular flux