NE 255: HW34 April Novak

(1) Due to the added complexity associated with higher dimensions, the P_N equations are derived here for the time-independent form of the 1-D, monoenergetic transport equation in Cartesian geometries given by the following equation, where the external and fission source are bundled together into S, which is assumed isotropic:

$$\mu\frac{\partial\psi(z,\mu)}{\partial z}+\Sigma_t(z)\psi(z,\mu)=\int_{4\pi}d\hat{\Omega}'\Sigma_s(z,\hat{\Omega}\cdot\hat{\Omega}')\psi(z,\hat{\Omega}')+\frac{S(z)}{4\pi}$$

After the P_N equations are derived for this equation, then the extension to the SP_N equations will be performed using heuristic arguments, although with much more effort, you could show that the SP_N equations are an asymptotic solution to the transport equation. The P_N approximation is made by expanding both the flux and scattering cross section in Legendre polynomials. We use Legendre polynomials here because we're in 1-D Cartesian geometry; in higher dimensions or in non-Cartesian frames, the spherical harmonics would be needed.

$$\psi(z,\mu)=\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu) \tag{1}$$

$$\Sigma_s(z,\hat{\Omega}\cdot\hat{\Omega}')=\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\hat{\Omega}\cdot\hat{\Omega}')\rightarrow\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\mu)P_l(\mu') \tag{2}$$

where n is used in the expansion of the flux to differentiate it from l used in expanding the scattering cross section. From the 1-D form of the addition theorem for spherical harmonics, the scattering cross section expansion has been simplified. Now, inserting Eqs. (1) and (1) into the 1-D transport equation listed in the beginning of this section:

$$\begin{aligned}\mu\frac{\partial}{\partial z}\left(\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)\right)+\Sigma_t(z)\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)=\\ \int_{4\pi}d\hat{\Omega}'\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\mu)P_l(\mu')\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)+\frac{S(z)}{4\pi}\end{aligned} \tag{3}$$

Because no quantities depend on $\hat{\Omega}$, the scattering integral can be integrated over $0\leq\phi\leq2\pi$ so that the integral becomes a function of only μ and space.

$$\begin{aligned}\mu\frac{\partial}{\partial z}\left(\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)\right)+\Sigma_t(z)\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)=\\ 2\pi\int_{-1}^{1}d\mu'\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\mu)P_l(\mu')\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)+\frac{S(z)}{4\pi}\end{aligned} \tag{4}$$

The orthogonality property of Legendre polynomials does not permit any extra terms (that depend on μ) to be present in the integrand. Hence, the first term in Eq. (4) must be rewritten using the recursive property of Legendre polynomials:

$$\begin{aligned}\frac{\partial}{\partial z}\left(\sum_{n=0}^{\infty}\frac{\cancel{2n+1}}{4\pi}\phi_n(z)\frac{1}{\cancel{2n+1}}\left[(n+1)P_{n+1}(\mu)+nP_{n-1}(\mu)\right]\right)+\Sigma_t(z)\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)=\\ 2\pi\int_{-1}^{1}d\mu'\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\mu)P_l(\mu')\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\phi_n(z)P_n(\mu)+\frac{S(z)}{4\pi}\end{aligned} \tag{5}$$

Multiplying Eq. (5) by $P_n(\mu)$ and then integrating over $-1 \leq \mu \leq 1$ gives, after canceling the $1/4\pi$ from each term:

$$\frac{\partial}{\partial z}\left(\sum_{n=0}^{\infty}\left[\int_{-1}^{1} d\mu\phi_n(z)(n+1)P_{n+1}(\mu)P_n(\mu) + \int_{-1}^{1} d\mu\phi_n(z)nP_{n-1}(\mu)P_n(\mu)\right]\right) + \Sigma_t(z)\sum_{n=0}^{\infty}(2n+1)\int_{-1}^{1} d\mu\phi_n(z)P_n(\mu)P_n(\mu) = \tag{6}$$
$$\int_{-1}^{1} d\mu P_n(\mu)2\pi\int_{-1}^{1} d\mu'\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\Sigma_{sl}(z)P_l(\mu)P_l(\mu')\sum_{n=0}^{\infty}(2n+1)\phi_n(z)P_n(\mu)P_n(\mu) + \int_{-1}^{1} d\mu S(z)P_n(\mu)$$

Then, applying the orthogonality property of Legendre polynomials gives:

$$\frac{\partial}{\partial z}\left(\sum_{n=0}^{\infty}\left[\frac{2(n+1)}{2n+1}\phi_{n+1}(z) + \frac{2n}{2n+1}\phi_{n-1}(z)\right]\right) + \Sigma_t(z)\sum_{n=0}^{\infty}2\phi_n(z) = 2\sum_{l=0}^{\infty}\Sigma_{sl}(z)\sum_{n=0}^{\infty}\phi_n(z) + \int_{-1}^{1} d\mu S(z)P_n(\mu) \tag{7}$$

where orthogonality was applied twice to the scattering term. Dividing through by 2 then gives the P_N equations for 1-D Cartesian geometries:

$$\frac{\partial}{\partial z}\left(\sum_{n=0}^{\infty}\left[\frac{n+1}{2n+1}\phi_{n+1}(z) + \frac{n}{2n+1}\phi_{n-1}(z)\right]\right) + \Sigma_t(z)\sum_{n=0}^{\infty}\phi_n(z) = \sum_{l=0}^{\infty}\Sigma_{sl}(z)\sum_{n=0}^{\infty}\phi_n(z) + \delta_{n,even}\frac{1}{2}\int_{-1}^{1} d\mu S(z)P_n(\mu) \tag{8}$$

Because $S(z)$ is not a function of μ, then the integral of a Legendre polynomial over its basis will give zero if that Legendre polynomial is an odd function, and will be nonzero otherwise. Hence, the source term is only present for even n, based on the form of the Legendre polynomials. For odd-n, the Legendre polynomials contain a sum of odd functions, and the integral of any odd function over a region symmetric about the origin is zero. Now, in order to make this solution method tractable, the infinite sums have to be truncated at some point. It is also customary to set $l = n$ such that the above reduce to:

$$\frac{\partial}{\partial z}\left(\sum_{n=0}^{N}\left[\frac{n+1}{2n+1}\phi_{n+1}(z) + \frac{n}{2n+1}\phi_{n-1}(z)\right]\right) + \Sigma_t(z)\sum_{n=0}^{N}\phi_n(z) = \sum_{n=0}^{N}\Sigma_{sn}(z)\phi_n(z) + \delta_{n,even}\frac{1}{2}\int_{-1}^{1} d\mu S(z)P_n(\mu) \tag{9}$$

We can require Eq. (9) to hold for all N at once, but we can also require it to hold for each N. This stricter requirement returns the requirement stated in Eq. (9), and hence the P_N equations in practice produce N coupled ODEs:

$$\frac{\partial}{\partial z}\left[\frac{n+1}{2n+1}\phi_{n+1}(z) + \frac{n}{2n+1}\phi_{n-1}(z)\right] + \Sigma_t(z)\phi_n(z) = \Sigma_{sn}(z)\phi_n(z) + \delta_{n,even}\frac{1}{2}\int_{-1}^{1} d\mu S(z)P_n(\mu) \tag{10}$$

This gives $N+1$ coupled equations. For example, some of the first P_N equations are:

$$\begin{aligned}
\frac{\partial \phi_1(z)}{\partial z} + \Sigma_t(z)\phi_0(z) &= \Sigma_{s0}\phi_0(z) + S_0(z) \qquad n=0 \\
\frac{2}{3}\frac{\partial \phi_2(z)}{\partial z} + \frac{1}{3}\frac{\partial \phi_0(z)}{\partial z} + \Sigma_t(z)\phi_1(z) &= \Sigma_{s1}\phi_1(z) \qquad n=1 \\
\frac{3}{5}\frac{\partial \phi_3(z)}{\partial z} + \frac{2}{5}\frac{\partial \phi_1(z)}{\partial z} + \Sigma_t(z)\phi_2(z) &= \Sigma_{s2}\phi_2(z) + S_2(z) \qquad n=2
\end{aligned} \tag{11}$$

In order to truncate the infinite series to N unknowns, $\phi_{-1} = 0$ is often set, and either $\phi_{N+1} = 0$ or $\partial\phi_{N+1}/\partial z = 0$ is set as the other condition, where both give equivalent results. N is usually selected to be odd. If N were even, then artificial symmetry would be introduced into the problem through application of boundary conditions. In addition, for even N, you do not obtain any additional information (non-linearly independent) from the P_N equations.

A heuristic derivation of the SP_N equations can be performed using simple arguments regarding the form of the 1-D P_N equations in Eq. (10). The key to deriving the SP_N equations is to transform Eq. (10) such that the derivative in the even-n equations is replaced by a divergence, while the derivative in the odd-n equations is replaced by a gradient. The SP_N equations therefore are:

$$\begin{aligned}
\frac{n+1}{2n+1}\nabla\cdot\vec{\phi}_{n+1}(z) + \frac{n}{2n+1}\nabla\cdot\vec{\phi}_{n-1}(z) + \Sigma_t(z)\phi_n(z) &= \Sigma_{sn}\phi_n(z) + S_n(z) \quad \text{even - } n \\
\frac{n+1}{2n+1}\nabla\phi_{n+1}(z) + \frac{n}{2n+1}\nabla\phi_{n-1}(z) + \Sigma_t(z)\vec{\phi}_n(z) &= \Sigma_{sn}\vec{\phi}_n(z) \quad \text{odd - } n
\end{aligned} \tag{12}$$

This first-order form gives $N+1$ equations. The SP_N equations can also be written in second-order form by solving for the odd moments in the odd-n equations, and then substituting this into the even moment equations. The SP_5 equations are therefore:

$$\begin{aligned}
\nabla\cdot\vec{\phi}_1(z) + \Sigma_t(z)\phi_0(z) &= \Sigma_{s0}\phi_0(z) + S_0(z) \quad n=0 \\
\frac{2}{3}\nabla\phi_2(z) + \frac{1}{3}\nabla\phi_0(z) + \Sigma_t(z)\vec{\phi}_1(z) &= \Sigma_{s1}\vec{\phi}_1(z) \quad n=1 \\
\frac{3}{5}\nabla\cdot\vec{\phi}_3(z) + \frac{2}{5}\nabla\cdot\vec{\phi}_1(z) + \Sigma_t(z)\phi_2(z) &= \Sigma_{s2}\phi_2(z) + S_2(z) \quad n=2 \\
\frac{4}{7}\nabla\phi_4(z) + \frac{3}{7}\nabla\phi_2(z) + \Sigma_t(z)\vec{\phi}_3(z) &= \Sigma_{s3}\vec{\phi}_3(z) \quad n=3 \\
\frac{5}{9}\nabla\cdot\vec{\phi}_5(z) + \frac{4}{9}\nabla\cdot\vec{\phi}_3(z) + \Sigma_t(z)\phi_4(z) &= \Sigma_{s4}\phi_4(z) + S_4(z) \quad n=4 \\
\frac{6}{11}\nabla\phi_6(z) + \frac{5}{11}\nabla\phi_4(z) + \Sigma_t(z)\vec{\phi}_5(z) &= \Sigma_{s5}\vec{\phi}_5(z) \quad n=5
\end{aligned} \tag{13}$$

If we assume an isotropic source, then $S_n = 0$ for $n \neq 0$. This simplifies the above to:

$$\begin{aligned}
\nabla\cdot\vec{\phi}_1(z) + \Sigma_t(z)\phi_0(z) &= \Sigma_{s0}\phi_0(z) + S_0(z) \quad n=0 \\
\frac{2}{3}\nabla\phi_2(z) + \frac{1}{3}\nabla\phi_0(z) + \Sigma_t(z)\vec{\phi}_1(z) &= \Sigma_{s1}\vec{\phi}_1(z) \quad n=1 \\
\frac{3}{5}\nabla\cdot\vec{\phi}_3(z) + \frac{2}{5}\nabla\cdot\vec{\phi}_1(z) + \Sigma_t(z)\phi_2(z) &= \Sigma_{s2}\phi_2(z) \quad n=2 \\
\frac{4}{7}\nabla\phi_4(z) + \frac{3}{7}\nabla\phi_2(z) + \Sigma_t(z)\vec{\phi}_3(z) &= \Sigma_{s3}\vec{\phi}_3(z) \quad n=3 \\
\frac{5}{9}\nabla\cdot\vec{\phi}_5(z) + \frac{4}{9}\nabla\cdot\vec{\phi}_3(z) + \Sigma_t(z)\phi_4(z) &= \Sigma_{s4}\phi_4(z) \quad n=4 \\
\frac{6}{11}\nabla\phi_6(z) + \frac{5}{11}\nabla\phi_4(z) + \Sigma_t(z)\vec{\phi}_5(z) &= \Sigma_{s5}\vec{\phi}_5(z) \quad n=5
\end{aligned} \tag{14}$$

For vacuum boundary conditions, we set Marshak boundary conditions on the odd flux moments.

$$2\pi\int_{\hat{\Omega}\cdot\hat{n}<0} d\mu P_l(\mu)\psi(\mu) = 2\mu\int_{\hat{\Omega}\cdot\hat{n}<0} d\mu P_l(\mu)\psi_b(\mu) \qquad l = 1,3,5\cdots N \tag{15}$$

where $\psi_b = 0$ because for vacuum boundaries we assume no incoming flux. Then, to be explicit, the boundary conditions listed above become:

$$\begin{aligned} 2\pi \int_{\hat{\Omega}\cdot\hat{n}<0} d\mu P_1(\mu)\psi(\mu) = 0 \quad &\rightarrow \quad \phi_1(\mu) = 0, \mu < 0 (\text{incoming}) \\ 2\pi \int_{\hat{\Omega}\cdot\hat{n}<0} d\mu P_3(\mu)\psi(\mu) = 0 \quad &\rightarrow \quad \phi_3(\mu) = 0, \mu < 0 (\text{incoming}) \\ 2\pi \int_{\hat{\Omega}\cdot\hat{n}<0} d\mu P_5(\mu)\psi(\mu) = 0 \quad &\rightarrow \quad \phi_5(\mu) = 0, \mu < 0 (\text{incoming}) \end{aligned} \tag{16}$$

(2) (a) The following integral will be evaluated using the LQ_N quadrature rules:

$$\int_{4\pi} d\hat{\Omega}|\hat{\Omega}| = \int_{4\pi} d\hat{\Omega}\sqrt{(\xi^2+\eta^2+\mu^2)} = \sum_{i=1}^{N(N+2)} w_i \sqrt{(\xi_i^2+\eta_i^2+\mu_i^2)} \tag{17}$$

where

$$\hat{\Omega} = \xi\hat{x}_1 + \eta\hat{x}_2 + \mu\hat{x}_3 \tag{18}$$

However, the integration in Eq. (17) assumes that the weights are normalized such that integration over the entire unit sphere gives unity. This quadrature rule is designed such that it is normalized over a single octant. Over an octant, μ ranges from 0 to 1 and $\phi$from 0 to $\pi/2$. Hence, the end result of the integration with the quadrature rules from Lewis and Miller must be multiplied by $2/\pi$ to attain the correct normalization. The Appendix contains the code used for this section. Using the S_4 LQ_N quadrature, integration over the unit sphere gives a value of 4π, which is expected analytically. Because the quadrature points are all on the unit sphere, the integrand in Eq. (17) can be exactly integrated with the level-symmetric quadrature.

(b) With S_6 quadrature, a value of 4π is again obtained. Hence, for this simple integration, again because all quadrature points already lie on the unit sphere, this integral can be exactly integrated.

(c) The code developed was extended to compute the quadrature integrations of several functions for several quadrature orders. The results of these integrations is given in the table below. As can be seen, and by investigation of other functions, integrations of any odd function give an integral of zero, as expected since the integration is performed over a domain symmetric about the origin. The order of the quadrature makes negligible difference for these integrations (to four significant digits), but would possibly give significant differences for integrands that are not or are not well-approximated as polynomials (since quadrature rules or order p can usually integrate a polynomial of order proportional to p exactly (the exact relationship depends on the quadrature rule)).

Table 1. Numerical integration results for various level-symmetric quadratures for several functions. The values shown are truncated at four decimal points.

Function	S_4 integration	S_6 integration	S_8 integration
μ	0.0000	0.0000	0.0000
μ^2	4.1887	4.1887	4.1887
$\eta^2 + 2$	29.321	29.321	29.321
$\cos(\xi)$	10.574	10.574	10.574

(3) (a) Complexity: In terms of complexity, deterministic methods are the most difficult to derive, understand, and encode. Diffusion theory is a deterministic method, and because it consists of one equation (assuming there is no anisotropic source, and that the time rate of change of the current with respect to its magnitude is small), it is not very complex to understand or encode. Diffusive equations also have very nice numerical properties, and do not require the messy stabilization methods required for nonlinear physics such as the Navier-Stokes equations. Deterministic methods require procedures for discretizing in space,

angle, energy, and time, while this is not required of Monte Carlo methods. The Monte Carlo code Serpent, for instance, is continuous in all of these variables except time, but finite differencing time stepping is only performed in the depletion subroutines (I think?). Monte Carlo methods are the least complex because they require simply tracking particle motion around repeatedly, though they can become complex once weight windows and particle tracking are introduced to improve the statistics.

Accuracy: Without the restrictions of computational cost, Monte Carlo methods are the most accurate, since there is no "approximation" of the physics or numerical errors that are always present in deterministic methods. For instance, in deterministic methods, there is always some type of truncation error that is present due to approximating a continuous process over a discrete mesh. In general, transport solutions are more accurate than diffusive solutions provided that the transport solutions were not performed with such a poor mesh that they are "useless." However, the benefit of deterministic methods is that they can potentially be more accurate than Monte Carlo solutions when constrained to a fixed run time, especially for shielding problems where a very large number of particles would need to be run in order to obtain sufficient statistics.

Run time: It is difficult to rank these methods in terms of run time, since increasing the number of particles or refining the mesh can always lead to a higher run time. To obtain the same level of error, I cannot say beforehand which method, deterministic or Monte Carlo, could obtain the result faster. If someone knew then we might not be studying one of the methods entirely! Diffusion problems will most likely run much faster than Monte Carlo and other deterministic methods. Deterministic methods on average likely run faster than Monte Carlo problems, since, especially for large geometries, Monte Carlo simulations can require very many particles to obtain satisfactory statistics.

Applicability: When using the diffusion equations, you are inherently assuming that 1) the angular flux is at most linearly anisotropic, 2) the time rate of change of the current is small with respect to its magnitude, and 3) there is no anisotropic source. Diffusion theory breaks down when the medium becomes poorly-scattering, i.e. near strong absorbers, near void regions or vacuum boundaries, and near material interfaces with strong changes in cross section. Due to the high level of generality of Monte Carlo methods, they are applicable under all situations that are governed by stochastic processes. Assuming that nuclear interactions are always stochastic, then Monte Carlo methods are applicable so long as the underlying nuclear data is accurate. Deterministic methods often place limitations of the degree of angular discretization, so for highly-dependent angular solutions, it may be better to use a Monte Carlo solution instead of a deterministic method with an unknown mesh requirement.

(b) Some strengths of deterministic methods include:

1. Generally have lower run times than Monte Carlo methods
2. Accuracy is always the same (i.e. you don't need to re-run the simulation many times to obtain an averaged result)
3. More closely related to transport theory, so interesting academic and mathematical results can be derived to inspire new methods

Some weaknesses of deterministic methods include:

1. More difficult to program and understand as a general user than Monte Carlo codes
2. Mesh refinement studies have to be performed before being able to trust the results
3. You always need to check that the range of validity of the equations fits your application before using a deterministic code

(4) (a) Figures **??** and **??** plot the real and imaginary components of Y_{lm} for $l = 0, 1, 2$ and for $-l \leq m \leq l$.

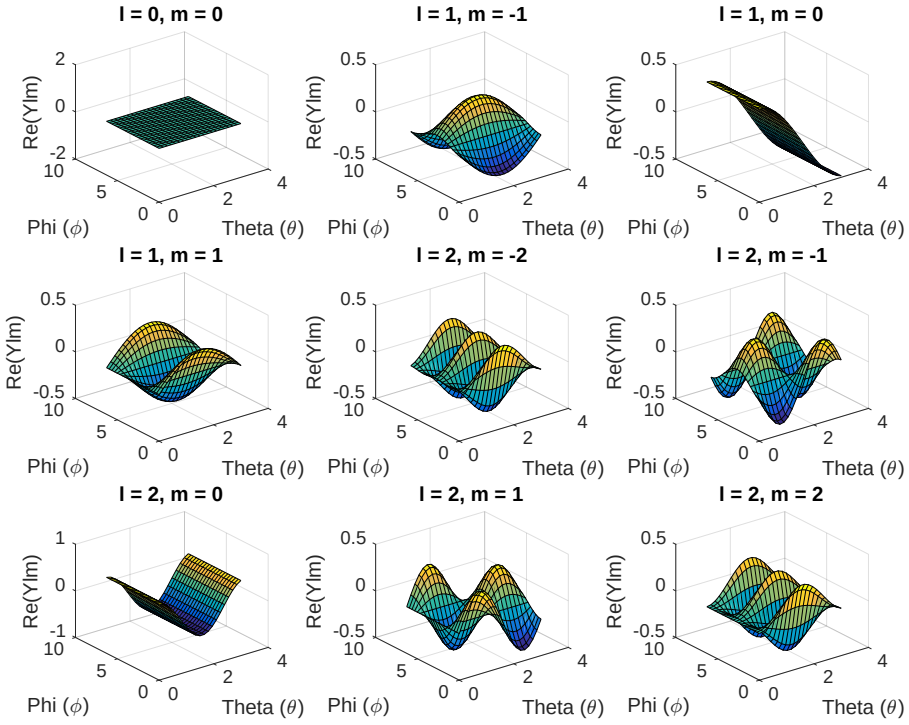

Figure 1. Real component of Y_{lm}.

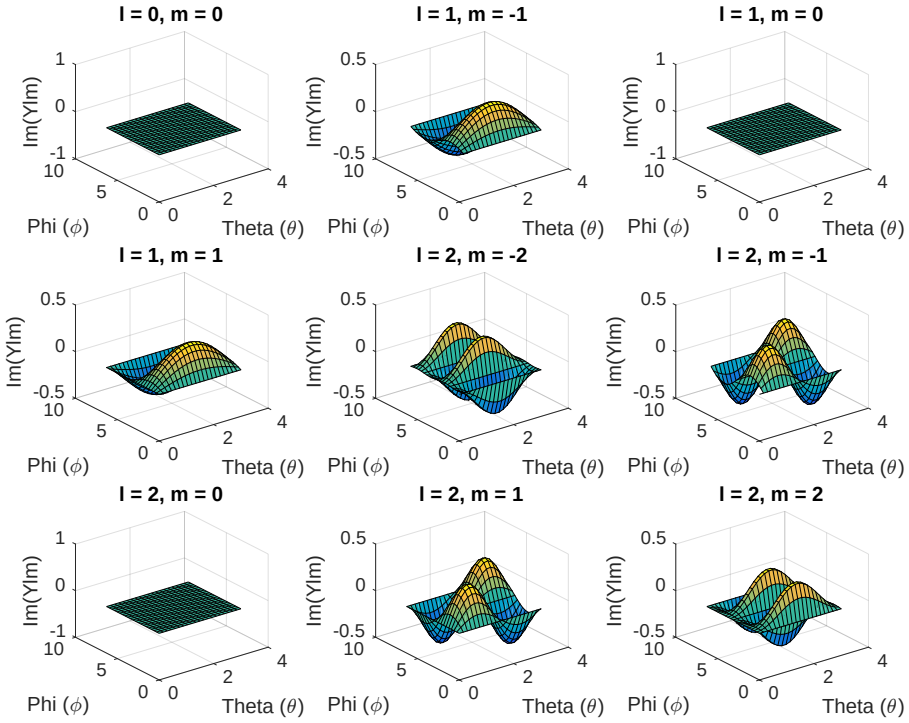

Figure 2. Imaginary component of Y_{lm}.

(b) When expanded in the spherical harmonics, the external (assumed monoenergetic) source is:

$$S(\vec{r}, E, \hat{\Omega}) = \sum_{l=0}^{N} \sum_{m=-l}^{l} Y_{lm}(\hat{\Omega}) S_{lm}(\vec{r}, E) \tag{19}$$

where from orthogonality:

$$S_{lm} = \int_{4\pi} S(\vec{r}, E, \hat{\Omega}) Y_{lm}^{*}(\hat{\Omega}) d\hat{\Omega} \tag{20}$$

By combining Eq. (19) with Eq. (20), the expansion becomes:

$$S(\vec{r}, E, \hat{\Omega}) = \int_{4\pi} \sum_{l=0}^{N} \sum_{m=-l}^{l} Y_{lm}(\hat{\Omega}) S(\vec{r}, E, \hat{\Omega}) Y_{lm}^{*}(\hat{\Omega}) d\hat{\Omega} \tag{21}$$

The addition theorem of spherical harmonics is:

$$P_l(\hat{\Omega}' \cdot \hat{\Omega}) = \frac{4\pi}{2l+1} \sum_{m=-l}^{l} Y_{lm}^{*}(\hat{\Omega}') Y_{lm}(\hat{\Omega}) \tag{22}$$

Using this addition theorem in Eq. (21):

$$S(\vec{r}, E, \hat{\Omega}) = \int_{4\pi} \sum_{l=0}^{N} S(\vec{r}, E, \hat{\Omega}) P_l(\hat{\Omega}' \cdot \hat{\Omega}) \frac{2l+1}{4\pi} d\hat{\Omega} \tag{23}$$

By requiring the expansion to only contain real components, it can be shown that:

$$P_l(\hat{\Omega}' \cdot \hat{\Omega}) = \frac{4\pi}{2l+1} \left[Y_{l0}^{e}(\hat{\Omega}) Y_{l0}^{e}(\hat{\Omega}') + \sum_{m=1}^{l} \left(Y_{lm}^{e}(\hat{\Omega}) Y_{lm}^{e}(\hat{\Omega}') + Y_{lm}^{o}(\hat{\Omega}) Y_{lm}^{o}(\hat{\Omega}') \right) \right] \tag{24}$$

Inserting Eq. (24) into Eq. (26):

$$S(\vec{r}, \hat{\Omega}) = \int_{4\pi} \sum_{l=0}^{N} S(\vec{r}, \hat{\Omega}) \left[Y_{l0}^{e}(\hat{\Omega}) Y_{l0}^{e}(\hat{\Omega}') + \sum_{m=1}^{l} \left(Y_{lm}^{e}(\hat{\Omega}) Y_{lm}^{e}(\hat{\Omega}') + Y_{lm}^{o}(\hat{\Omega}) Y_{lm}^{o}(\hat{\Omega}') \right) \right] d\hat{\Omega} \tag{25}$$

This can be shown in shorthand notation as:

$$S(\vec{r}, \hat{\Omega}) = \sum_{l=0}^{N} \left[q_{l0}(\vec{r}) Y_{l0}^{e}(\hat{\Omega}') + \sum_{m=1}^{l} \left(q_{lm}(\vec{r}) Y_{lm}^{e}(\hat{\Omega}') + s_{lm}(\vec{r}) Y_{lm}^{o}(\hat{\Omega}') \right) \right] \tag{26}$$

$$\begin{aligned} q_{lm}(\vec{r}) &= \int_{4\pi} S(\vec{r}, \hat{\Omega}) Y_{lm}^{e}(\hat{\Omega}) \\ s_{lm}(\vec{r}) &= \int_{4\pi} S(\vec{r}, \hat{\Omega}) Y_{lm}^{o}(\hat{\Omega}) \end{aligned} \tag{27}$$

Hence, in order to compute the external source using quadrature rules, q_{lm} and s_{lm} must be computed using the quadrature rules according to their definitions in Eq. (27), and then the result summed according to Eq. (26). For this problem, $S(\vec{r}, \hat{\Omega}) = 1$. Hence, for this specific problem, we are computing the following with S_4 quadrature:

$$S(\vec{r}, \hat{\Omega}) = \sum_{l=0}^{N} \left[q_{l0}(\vec{r}) Y_{l0}^{e}(\hat{\Omega}') + \sum_{m=1}^{l} \left(q_{lm}(\vec{r}) Y_{lm}^{e}(\hat{\Omega}') + s_{lm}(\vec{r}) Y_{lm}^{o}(\hat{\Omega}') \right) \right] \tag{28}$$

$$
\begin{aligned}
q_{lm}(\vec{r}) &= \int_{4\pi} Y^e_{lm}(\hat{\Omega}) \\
s_{lm}(\vec{r}) &= \int_{4\pi} Y^o_{lm}(\hat{\Omega})
\end{aligned}
\tag{29}
$$

Because level-symmetric quadrature is to be used in this problem, the code developed for question 2 is used for this question. Instead of integrating arbitrary functions, we are integrating the even and odd components of the spherical harmonics. From parts (a) and (b), the code for this question already computes Y_{lm}. The spherical harmonics are:

$$
Y_{lm}(\theta,\phi) = C_{lm} P_{lm} \exp(im\phi) \tag{30}
$$

where

$$
C_{lm} = (-1)^m \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} \tag{31}
$$

The even and odd components of the spherical harmonics are defined as:

$$
\begin{aligned}
Y^e_{lm} &= \sqrt{2-\delta_{m0}} C_{lm} P_{lm} \cos(m\phi) \\
Y^o_{lm} &= \sqrt{2-\delta_{m0}} C_{lm} P_{lm} \sin(m\phi)
\end{aligned}
\tag{32}
$$

Hence, the vast majority of the code developed for parts (a) and (b) can be re-used. The code simply needs to be modified to compute Y^e_{lm} and Y^o_{lm}. Then, the quadrature code developed for question 2 is used to perform the integration. The quadrature is set up to perform integration with respect to μ, η, and ξ, which are defined here to be (the definition varies upon which source they come from):

$$
\begin{aligned}
\xi \equiv \sin(\theta)\cos(\phi) &= \Omega_x \\
\eta \equiv \sin(\theta)\sin(\phi) &= \Omega_y \\
\mu \equiv \cos(\theta) &= \Omega_z
\end{aligned}
\tag{33}
$$

I finished all the code for this, I just didn't finish the plotting. The code is all in `Q4b.m`, hopefully I can get partial credit (I thought it'd be ok to turn this in by 5 pm).

(5) Some of the major nuclear data libraries include:

1. ENDF (Evaluated Nuclear Data File): United States
2. JENDL (Japanese Evaluated Nuclear Data Library): Japan
3. TENDL (TALYS-based Evaluated Nuclear Data Library): (TALYS is managed by the Netherlands). TALYS is a code that contains nuclear physics to predict cross sections. Up until 2015, this library was managed in the Netherlands, but since then management has switched to France and Switzerland.
4. JEFF (Joint Evaluated Fission and Fusion File): collaboration between NEA Data Bank members
5. CENDL (Chinese Evaluated Neutron Data Library): China
6. RUSFOND: Russia
7. BROND: Russia

1 Appendix

This section contains all the code used for each question.

1.1 Question 2

```
% QUESTION 2 - NE 255 hw 2
clear all

N = 8;                                          % quadrature order
syms xe eta mu                                  % components of \Omega

% Parts A and B
%f(xe, eta, mu) = sqrt(xe^2 + eta^2 + mu^2);    % function to integrate

% Part C (uncomment to see the result)
%f(xe, eta, mu) = mu;
%f(xe, eta, mu) = mu^2;
f(xe, eta, mu) = cos(mu);

% perform the integration using LQn quadrature
[integral] = LQnQuadrature(N, f);
```

1.2 Question 4

```
% Question 4, HW 3
clear all

j = 1;              % subplot number
real_plot = 0;      % 1 = plot real components, 0 = plot imaginary

for l = [0, 1, 2];
    for m = -l:l
        spacing = 0.15;
        theta = meshgrid(0:spacing:pi);
        mu = cos(theta);
        phi = meshgrid(linspace(0, 2*pi, length(theta)));
        [Theta, Phi] = meshgrid(0:spacing:pi, linspace(0, 2*pi, length(theta))
            ↪ );

        % compute Legendre polynomials derivative term
        [derivative] = LegendrePolynomialDerivative(l, m);

        % plotting domain for mu
        x = mu;

        % for Legendre polynomials
        prefactor = ((-1).^m) / ((2^l) * factorial(l));
        prefactor2 = (1 - x.^2) .^ (m./2);
        P_lm = prefactor .* prefactor2 .* eval(derivative);

        % for spherical harmonics
        prefactor = sqrt((2*l + 1) * factorial(l - m));
        prefactor2 = sqrt(4 * pi * factorial(l + m));
        Y_lm = ((-1).^ m) .* (prefactor ./ prefactor2) .* P_lm .* exp(1i .* m
            ↪ .* Phi);
```

```
        if (real_plot == 1)
            % plot the real component
            subplot(3, 3, j)
            surf(Theta, Phi, real(Y_lm));
            title(sprintf('l = %i, m = %i', l, m))
            zlabel('Re(Ylm)')
            xlabel('Theta (\theta)')
            ylabel('Phi (\phi)')
            plot_title = 'Real_Ylm';
        else
            % plot the imaginary component
            subplot(3, 3, j)
            surf(Theta, Phi, imag(Y_lm));
            title(sprintf('l = %i, m = %i', l, m))
            zlabel('Im(Ylm)')
            xlabel('Theta (\theta)')
            ylabel('Phi (\phi)')
            plot_title = 'Imag_Ylm';
        end

        j = j + 1;
    end
end

saveas(gcf, plot_title, 'pdf')
```

1.3 Question 4b

```
% Question 4, part b

% This code calculates the external source.

clear all

N = 6;                                          % quadrature order
l = 8;                                          % l in spherical harmonics

syms xe eta mu                                  % components of \Omega

for m = [0, 1:l]
    C_lm = (-1)^m * sqrt((2*l+1) * factorial(l-m) / (4 * pi * factorial(l+m)))
        ↪ ;

    % compute Associated Legendre polynomials derivative term
    if ((l + m) == 0)
        derivative = (mu.^2 -1) .^ l;
    elseif ((l + m) < 0)
        disp('Incorrect combination of l and m');
    else
        derivative = diff((mu.^2 - 1).^l, mu);
        if (l + m) > 1
            for i = 1:(l + m - 1)
                derivative = diff(derivative, mu);
```

```
            end
        end
    end

    P_lm = ((-1).^m) / ((2^l) * factorial(l)) .* (1 - mu.^2) .^ (m./2) .*
        ↪ derivative;

    if m == 0
        term1 = sqrt(2-1);
    else
        term1 = sqrt(2);
    end

    theta = acos(mu);
    phi = asin(eta/sin(theta));

    f(xe, eta, mu) = term1 * C_lm * P_lm * cos(m * phi);     % Y_{lm}^e
    f2(xe, eta, mu) = term1 * C_lm * P_lm * sin(m * phi);    % Y_{lm}^o

    % perform the integration using LQn quadrature
    [integral_even] = LQnQuadrature(N, f);              % integrate Y_{lm}^e
    %[integral_odd] = LQnQuadrature(N, f2);              % integrate Y_{lm}^o

end

% perform the plotting
spacing = 0.15;
theta = meshgrid(0:spacing:pi);
mu = cos(theta);
phi = meshgrid(linspace(0, 2*pi, length(theta)));
[Theta, Phi] = meshgrid(0:spacing:pi, linspace(0, 2*pi, length(theta)));
```