Neutron Slowing Down Spectrum

SH2774 Numerical Methods in Nuclear Engineering

Wolf Timm

Abstract

Numerical analysis is a powerful tool with incredible potential for addressing all manner of complex issues, especially those not feasibly solvable through conventional methods. However, great care must be taken that its power is harnessed correctly and with caution; there are many pitfalls that require diligence and patience to avoid when solving problems with this class of methods. Often, numerical methods are used in conjunction with analytical techniques to achieve the best possible results, as each approach tends to have its own strengths. The results achieved in this report were concluded through both numerical and analytical means, and provide a good example of the capabilities of applying a numerical approach where appropriate—as well as an illustration of the limitations and shortcomings.

In this project, a multitude of concepts in mathematics, computer science and nuclear reactor theory are explored and must be applied in order to reach the objective. These culminate in deriving a solution that ultimately stems from a single integro-differential equation known as the neutron continuity equation. This equation on its own is not analytically solvable, but through a series of assumptions and conditional limitations, a fully integrable equation with an analytic solution may be derived. The integrand takes the form of a convolution, however, and therefore must be converted to a complex frequency domain analogue using the Laplace transform to separate the two underlying functions. The main challenge arises when attempting the inversion of the transform back to the energy domain, which is not analytically possible to solve fully in addition to being notoriously ill-conditioned and inefficient when approached from a numerical perspective. The solution is found as the result of a Python algorithm written with the specific intent of solving this problem as efficiently as possible without sacrificing precision, while at the same time attempting to address the inherent and compounding numerical instability of both the Laplace inversion and the solution equation itself.

As applied to in-depth studies of physical phenomena—and indeed all complex problems requiring a numerical approach—the aim is to strike a balance between precision and simplicity. Though analytical exactness has undeniable elegance, a well-balanced and thoroughly thought-out numerical process simply cannot be beat in terms of versatility and power. This report explores careful selection of methodology and leveraging the strengths of each type of approach to achieve the best possible outcome.

1. Introduction

There are a few equations that are usually the starting point for research and problem-solving in the field of nuclear reactor physics, and one of them is known as the neutron continuity equation, of a general form

$$\frac{\partial n}{\partial t} = S(r,E,t) + \int_0^\infty \Sigma(r,E' \to E)\varphi(r,E',t)dE' - \Sigma_t(r,E)\varphi(r,E,t) - \nabla \cdot J(r,E,t), \quad (1)$$

where $\partial n/\partial t$ is the rate of change of the neutron density, S is the emission rate of neutrons (per cm^3) from an external source, Σ_t is the total macroscopic cross-section, J is the neutron current density and φ is the neutron flux. The basic operating principle is that of balance; according to the concept of neutron diffusion, free neutrons can be approximated to a behavior of tending towards equilibrium density in many scenarios, much like a solute dissolving in a solvent.

The reason for starting with this equation is that, if some assumptions are made, it can be converted so that it only depends on one variable: energy. To that end, it is helpful to recognize that all the parameters involved depend either on spatial dimensions, time, or both. The goal of this project is to model the behavior of fast neutrons slowing down in a medium, not due to time-related phenomena, so the assumption can be safely made that the sought-after results are time-independent, making the neutron density with respect to time $\partial n/\partial t$ irrelevant in this case. In order to neglect the spatial dimensions as well, it must be assumed that the medium is both infinite and uniform. Under these assumptions, the neutron current density J—since it is a gradient vector and can be thought of as the flux of the neutron flux—becomes zero, leaving only the total and differential reaction rates $\Sigma\varphi$, as well as the source emission S, affecting the balance equation with respect to energy:

$$\Sigma_t(E)\varphi(E) = \int_0^\infty \Sigma(E' \to E)\varphi(E)dE' + S(E). \quad (2)$$

Now that the continuity equation has been simplified to a steady-state formulation, the process of manipulating the equation into a form that can be numerically approximated can begin.

2. Analysis

2.1. Scattering and the scattering coefficient

Since the continuity equation is now expressed solely in terms of energy, each type of reaction cross-section for free neutron-nucleus interactions should now be considered, because, as is evident from Eq. (2) above, the slowing down of a neutron depends entirely on the reaction rate and the source itself in this scenario. There are three possibilities of occurrence: fission, absorption or scattering.

In a uniform infinite medium that is not comprised of a fissile material, there will be no fission, and the fission cross-section does not exist for that isotope or element. Instead, there is only an absorption cross-section, which encompasses fission in addition to general radiative capture and subsequent decays—essentially indicating the likelihood that an incident neutron will be absorbed by the nucleus. Finally, there is the scattering cross-section, which describes the probability of incident neutrons being influenced by the strong nuclear force to change direction and, therefore, lose energy. This is the most prominent cross-section responsible for the characteristics of the neutron energy spectrum, and is therefore a necessary factor to consider.

As pertains to nuclear reactors, absorption reactions' contribution to the neutron energy spectrum is secondary to scattering. In fact, it often serves only to interfere with and obfuscate the well-defined effects of scattering in the spectrum. For this reason, and for the sake of simplicity, the influence of these interactions will be excluded from the following calculations. Neglecting all absorption reactions—fission included—simplifies

Eq. (2) even further by allowing for the identification of the differential reaction rate as being entirely due to scattering, since it is the only reaction type left to consider:

$$\Sigma_s(E)\varphi(E) = \int_0^\infty \Sigma_s(E' \to E)\varphi(E')dE' + S(E).$$

There are two kinds of scattering: elastic and inelastic. Both play a large role in the distribution of the energy spectrum. However, inelastic collisions only play a large role with neutrons of energy in the keV order of magnitude. Although fast neutrons are indeed the type of interest, all collisions are assumed to be elastic—again, in favor of simplicity. The attention is not on the medium through which the neutrons travel either, so it is assumed to be a single homogenous material, the atomic mass of which determines the scattering parameter:

$$\alpha = \left(\frac{A-1}{A+1}\right)^2.$$

Focusing now on the differential scattering rate integral, it can be seen that there is a probability function governing the differential scattering cross-section. According to kinematics, the probability density for elastic scattering is equal to:

$$f_s(E' \to E)dE = \begin{cases} \frac{dE}{(1-\alpha)E'} & E \le E' \le \frac{E}{\alpha} \\ 0 & otherwise \end{cases} \quad (3)$$

The normalization condition of the probability function should be equal to 1, since this statement covers the total domain of possibility. This can be verified by integrating the non-zero component as

$$\int_0^\infty f_s(E' \to E)dE = 1 \to \int_E^{E/\alpha} \frac{dE}{(1-\alpha)E'} = \frac{E}{\alpha E' - \alpha^2 E'} - \frac{\alpha E}{\alpha E' - \alpha^2 E'},$$

$$= \frac{E}{\alpha E'}, \quad where\ E' \le \frac{E}{\alpha} \to \alpha E' \le E$$

$$The\ upper\ limit\ of\ \alpha E' = E \to = \frac{E}{E} = 1.$$

2.2. Collision rate density

Another simplification that can be made to Eq. (2) is specifying each pair of fluxes and scattering cross-sections as collision rate densities:

$$F(E) = \Sigma_s(E)\varphi(E).$$

Also, the neutron source can be specified simply as a monoenergetic impulse of strength S_0 and neutrons of energy E_0. This finally gives the slowing down equation in terms of collision rate density:

$$F(E) = \int_0^\infty f_s(E' \to E)F(E')dE' + S_0\delta(E - E_0). \quad (4)$$

However, since this is still an extension of the neutron continuity equation, there must be a term to balance out the source, which takes form

$$F(E) = F_c(E) + C\delta(E - E_0),$$

where C is a constant and $F_c(E)$ represents the collision rate density of neutrons that have experienced at least one collision. This term is then plugged back into Eq. (4)

$$if \quad F(E) = F_c(E) + C\delta(E - E_0), \qquad then \qquad F(E') = F_c(E') + C\delta(E' - E_0),$$

$$F(E) = F_c(E) + C\delta(E - E_0) = \int_0^\infty f_s(E' \to E)(F_c(E') + C\delta(E' - E_0))dE' + S_0\delta(E - E_0),$$

From here, the integral can be split and the Dirac delta function $\delta(t - a)$ integrates with a function of form $f(t)$ under these conditions to become $f(a)$:

$$F_c(E) + C\delta(E - E_0) = \int_0^\infty f_s(E' \to E)F_c(E')dE' + Cf_s(E_0 \to E) + S_0\delta(E - E_0),$$

and it is now taken into account that energy contributions of sources must be equal to $C = S_0$, cancelling out the probability densities on either side. This results in a formula for the collision rate density of neutrons that have collided at least once:

$$F_c(E) = \int_0^\infty f_s(E' \to E)F_c(E')dE' + S_0 f_s(E_0 \to E). \qquad (5)$$

2.3. Defining the first collision interval

Now that the general form of the slowing down spectrum equation is known in terms of energy and collision rate density, a more specific approach can be taken. Recalling that in elastic scattering collisions energy reduces by a factor of α, it can therefore be said the energy interval for the initial collision of a neutron would be

$$\alpha E_0 \le E \le E_0.$$

Since the elastic scattering probability density was defined earlier, the limits of integration for the first collision are now known. The probability density can then be substituted into Eq. (4) to arrive at the form

$$F_c(E) = \int_E^{E_0} \frac{F_C(E')dE'}{(1 - \alpha)E'} + \frac{S_0}{(1 - \alpha)E_0},$$

which can be simplified and expressed as an ordinary differential equation by first taking the derivative of both sides

$$F_c(E) = \frac{1}{(1 - \alpha)} \int_E^{E_0} \frac{F_C(E')}{E'} dE' + \frac{S_0}{(1 - \alpha)E_0},$$

$$\frac{d}{dE}(F_c(E)) = \frac{d}{dE}\left(\frac{1}{(1 - \alpha)} \int_E^{E_0} \frac{F_C(E')}{E'} dE' + \frac{S_0}{(1 - \alpha)E_0}\right),$$

$$F_c'(E) = \frac{1}{(1 - \alpha)} \frac{d}{dE}\left(\int_E^{E_0} \frac{F_C(E')}{E'} dE'\right) + \frac{d}{dE}\left(\frac{S_0}{(1 - \alpha)E_0}\right),$$

and implicitly differentiating the definite integral:

$$F_c'(E) = \frac{1}{(1-\alpha)}\frac{d}{dE}(G(E_0) - G(E)) \quad where\ G(x) = \int_{x_1}^{x_2} \frac{F_c(x)}{x}dx,$$

$$F_c'(E) = \frac{1}{(1-\alpha)}\left(-\left(G'(E)\cdot\frac{d}{dE}(E)\right)\right),$$

then substituting back in for function G; use of the product rule of differentiation is required:

$$F_c'(E) = \frac{1}{(1-\alpha)}\left(-\left(F_c(E)\cdot\frac{d}{dE}(E^{-1}) + E^{-1}\cdot\frac{d}{dE}(F_c(E))\right)\right),$$

$$F_c'(E) = \frac{1}{(1-\alpha)}\left(-\left(F_c(E)\cdot\frac{d}{dE}(E^{-1}) + E^{-1}\cdot\frac{d}{dE}(F_c(E))\right)\right),$$

$$F_c'(E) = \frac{1}{(1-\alpha)}\left(-\left(-\frac{F_c(E)}{E^2} + \frac{F_c'(E)}{E}\right)\right),$$

$$F_c'(E) = \frac{1}{(1-\alpha)}\left(\frac{F_c(E) - EF'(E)}{E^2}\right),$$

$$F_c'(E)\left(1 + \frac{1}{E(1-\alpha)}\right) = \frac{F_c(E)}{E^2(1-\alpha)},$$

$$F_c'(E) = \frac{F_c(E)}{E^2(1-\alpha) + E}.$$

2.4. Neutron lethargy

A broad definition of "fast" is being applied to the study of neutrons here; there are actually several range definitions in between thermal and fast. However, where most nuclear reactors are concerned (which is the frame of reference for this project), any neutron above the thermal energy range is considered a target for moderation. Therefore, neutrons from 1 eV up to 10^7 eV are to be a part of this slowing down spectrum.

Since this energy range spans eight orders of magnitude, a new variable that logarithmically represents the ratio of initial energy to post-collision(s) energy is defined:

$$u = \ln\frac{E_0}{E},$$

known as the neutron lethargy.

In subsequent calculations, it will be advantageous to have a representation of the maximum change in lethargy per scattering collision

$$q = \ln\frac{1}{\alpha} = (\Delta u)_{\max},$$

wherein

$$e^q = \frac{1}{\alpha}, \quad e^{-q} = \alpha.$$

Lethargy is to be the new independent variable in the previously derived equations, meaning a conversion must take place between energy E and lethargy u. This begins with the differential relationship

$$du = -\frac{dE}{E} \quad \leftrightarrow \quad dE = -E_0 e^{-u} du.$$

Considering that this change of variables is between physical quantities, one of which relates as per unit range to its counterpart, the collision rate density for example exhibits the identity

$$F(u)du = -F(E)dE.$$

Thus, the transformation can be made between energy and lethargy in Eq. (3) by first finding the identities of every function in terms of energy and lethargy with respect to the other. This can be performed by substituting the relationship between differentials into the previous identity for each variable:

$$-F(u)\frac{dE}{E} = -F(E)dE, \qquad F(u)du = -F(E)(-E_0 e^{-u} du),$$

$$F(u) = EF(E), \qquad F(E) = \frac{1}{E_0} e^{u} F(u).$$

Now that these identities are known, the probability density function can be solved in terms of the independent variable using Eq. (3):

$$f_s(u' \to u)du = -f_s(E' \to E)dE,$$

$for\ \alpha E \le E' \le E_0$:

$$f_s(u' \to u)du = -\left(\frac{1}{(1-\alpha)E'}\right)dE,$$

$$f_s(u' \to u)\left(-\frac{dE}{E}\right) = -\left(\frac{1}{(1-\alpha)E'}\right)dE,$$

$$f_s(u' \to u) = \frac{E}{(1-\alpha)E'},$$

$$f_s(u' \to u) = \frac{E_0 e^{-u}}{(1-\alpha)(E_0 e^{u'})},$$

$$f_s(u' \to u) = \frac{e^{-(u-u')}}{(1-\alpha)}, \qquad (6)$$

and because q is the maximum change in lethargy for one collision, the impulse from the source does not apply past this limit. This will be taken into account later.

Next, the slowing down spectrum equation will be arranged in terms of u using the relations above and applying them to all variables of Eq. (3):

$$\left(-F(u)\frac{du}{dE}\right) = \int_0^\infty (-F(u')du')(-f_s(u' \to u)\frac{du}{dE}) + S_0\delta(E - E_0),$$

$$\frac{dE}{du}\left(F(u)\frac{du}{dE}\right) = \frac{dE}{du}\left(\int_0^\infty f_s(u' \to u)F(u')du'\frac{du}{dE} - S_0\delta(E - E_0))\right),$$

$$F(u) = \int_{-\infty}^{\infty} f_s(u' \to u)F(u')du' - S_0 \frac{dE}{du}\big(\delta(E - E_0)\big),$$

$$F(u) = \int_{-\infty}^{\infty} f_s(u' \to u)F(u')du' + S_0 E\big(\delta(E - Ee^{u})\big),$$

then, from the property of the Dirac delta function that $\delta(f(t)) = \delta(t) / |f'(0)|$ when $f(t)$ only has a single root at $t = 0$:

$$F(u) = \int_{-\infty}^{\infty} f_s(u' \to u)F(u')du' + S_0 E \frac{\delta(u)}{\left|\frac{d}{du}(E - Ee^{(0)})\right|},$$

$$F(u) = \int_{-\infty}^{\infty} f_s(u' \to u)F(u')du' + S_0 E \frac{\delta(u)}{|-Ee^{0}|},$$

$$F(u) = \int_{-\infty}^{\infty} f_s(u' \to u)F(u')du' + S_0\delta(u).$$

Finally, the probability density term in the equation can be substituted with Eq. (5), and after taking into account the previously discussed condition at $u \geq q$, the complete expression of the neutron slowing down energy spectrum has been derived as

$$F(u) = \int_{max\{0,u-q\}}^{u} \frac{e^{-(u-u')}}{(1-\alpha)} F(u')du' + S_0\delta(u). \quad (7)$$

2.5. Laplace transform

There is no possibility to solve Eq. (7) through conventional integration. However, if the equation is converted to a new domain, it may be possible to circumvent this fact. A practical way to do this is through a Laplace transform.

Laplace transforms are very useful in evaluating otherwise difficult problems in linear dynamical systems, as well as ordinary differential equations. This is primarily due to the fact that differentiation and integration become multiplication and division respectively with the transformation of the independent variable, along with an added bonus that these transforms are well suited to representing discontinuous functions.

The Laplace transform, for a function $F(u)$ where $u \geq 0$ and is real, becomes a new function in terms of p as

$$\hat{F}(p) = \mathcal{L}[F(u)] = \int_{-0}^{\infty} e^{-pu}F(u)du,$$

where -0 represents the inclusion of the limit to +0, ensuring the Dirac delta function in the equation to be transformed is represented.

In order to return the p-space Laplace function to the domain of u, the inverse of this function must then be taken. One of the ways to do so is by using a complex integral known as the Bromwich integral

$$F(u) = \mathcal{L}^{-1}[\hat{F}(p)] = \frac{1}{2\pi i}\int_{\gamma+i\cdot\infty}^{\gamma+i\cdot\infty} e^{pu}\hat{F}(p)dp, \tag{8}$$

where γ is a real number representing a vertical contour in the complex plane that the path of integration follows, and must be set such that $\gamma \geq \mathrm{Re}(p_n)$ for all poles p_n in order to avoid singularities and consequent instability.

Now, the slowing down equation can be transformed using Eq. (7) and known properties of the Laplace transform:

$$\hat{F}(p) = \mathcal{L}[F(u)] = \mathcal{L}\left[\int_0^u \frac{e^{-(u-u')}}{(1-\alpha)}F(u')du' + S_0\delta(u)\right],$$

$$= \mathcal{L}\left[\int_0^u \frac{e^{-(u-u')}}{(1-\alpha)}F(u')du'\right] + \mathcal{L}[S_0\delta(u)] = \frac{1}{(1-\alpha)}\mathcal{L}\left[\int_0^u e^{-(u-u')}F(u')du'\right] + S_0\mathcal{L}[\delta(u)],$$

$$= \frac{1}{(1-\alpha)}\mathcal{L}\left[\int_0^u e^{-(u-u')}F(u')du'\right] + S_0,$$

where at this point, the integral term is recognized to be a convolution of two functions. Using the properties of convolutions and exponential functions in regards to Laplace transforms, the result is found as

$$= \frac{1}{(1-\alpha)}\mathcal{L}[G(u) * F(u)] + S_0 = \frac{1}{(1-\alpha)}\mathcal{L}[G(u)]\hat{F}(p) + S_0,$$

$$= \frac{1}{(1-\alpha)}\mathcal{L}\left[\int_0^u e^{-(u-u')}du'\right]\hat{F}(p) + S_0 = \frac{1}{(1-\alpha)}\mathcal{L}[e^{-u}(e^u - 1)]\hat{F}(p) + S_0,$$

$$= \frac{1}{(1-\alpha)}\mathcal{L}[1 - e^{-u}]\hat{F}(p) + S_0 = \frac{1}{(1-\alpha)}\left[-\left(\frac{-1}{p}\right) - \frac{1}{p+1}\right]\hat{F}(p) + S_0,$$

$$= \frac{1}{(1-\alpha)}\left(\frac{1}{p(p+1)}\right)\hat{F}(p) + S_0. \tag{9}$$

This is analogous to the differential equation derivation of the first collision interval made previously:

$$\hat{F}(p) = \frac{1}{(1-\alpha)}\left(\frac{1}{p(p+1)}\right)\hat{F}(p) + S_0 \quad \sim \quad F_c'(E) = \frac{1}{(1-\alpha)}\left(\frac{1}{E(E+1)}\right)F_c(E).$$

Recall the case from Eq. (7) where $u \geq q$ in which no non-collided neutrons exist; this must be expressed as well. It will be kept in mind in the operations and algorithm design to come, and should be represented in the results (i.e. there should be a discontinuity at $u = q$, where the source impulse no longer exerts its effects).

3. Numerical calculations

3.1. Inverse of the Laplace transform

It is at this point that analytical methods reach their limits. It is not possible to derive the symbolic representation of the inverse of Eq. (9), therefore numerical methods must be applied.

There already exist many algorithms designed to invert Laplace transforms, yet none work in every case—especially if *F^(p)* is quite complicated, as is the case here. From this perspective, in the interest of precision and control, a custom algorithm with carefully set parameters is the best option moving forward.

To this end, the Bromwich integral (Eq. (8)) is a common choice and is viable in the case of the slowing down equation. For this purpose, however, it is simpler and easier to process if this equation is restructured according to the Fourier cosine integral form

$$F(u) = \frac{2e^{\gamma u}}{\pi u} \int_0^{\infty} \mathrm{Re}\left[\overline{F}\left(\gamma + i\omega/u\right)\right] \cos\omega \, d\omega,$$

where $\overline{F}(p)$ is known as the image function, which is extended to complex space and restricted to avoid integrating along the contour over any singularities by using γ to shift the vertical edge along the positive real axis.

The problem that arises here is that the equation in this form, as previously with Eq. (8), contains an improper complex integral. A method to avoid this issue is to convert this representation to an infinite series. The embodiment of this takes the generalized form of

$$F(u) = \frac{2e^{\gamma u}}{\pi u} \int_0^{\infty} (\cdot)\, d\omega = \frac{2e^{\gamma u}}{\pi u} \left[\int_0^{\pi/2} (\cdot)\, d\omega + \int_{-\pi/2}^{3\pi/2} (\cdot)\, d\omega + K \right],$$

for which we can define an iterable *k* to be applied to the integration limits $\left[k\frac{\pi}{2}, (k+1)\frac{\pi}{2}\right]$ and exponentiate it to account for the oscillatory nature of the equation. This adjustment gives the infinite series formulation that will be used in the algorithm to approximate the inverse Laplace transform of the slowing down equation:

$$F(u) = \frac{2e^{\gamma u}}{\pi u} \left[\int_0^{\pi/2} \mathrm{Re}\left[\overline{F}\left(\gamma + i\omega/u\right)\right] \cos d\omega + \sum_{k=1}^{\infty} (-1)^k \int_{-\pi/2}^{\pi/2} \mathrm{Re}\left[\overline{F}\left(\gamma + i(\omega + k\pi)/u\right)\right] \cos d\omega \right]. \quad (10)$$

This series, while relatively simple and convenient to program, is known to have a low convergence rate. If precision is the objective, a large number of iterations are required to revert the function in question from *p*-space while not having a significant amount of error.

Fortunately, there are many processes that can linearly or even quadratically increase the rate of convergence. One such method is the Aitken's delta-squared iteration method. This method is highly effective at accelerating convergence, especially for a series that converges linearly. The mechanism behind this process will be explained in detail in the following section on the topic of optimizations.

With the principle of Aitken's process in mind and applied to the Bromwich integral shown in Eq. (10), the core of the algorithm (written in the Python language) for the inverse Laplace numerical approximation was created. The initial source code for this vital component of the numerical analysis, along with commented annotations for readability, is written within and represented by the following functions `f_0()` and `f_u()`:

```
def f_0(i):    # Initial integral in the series. Added at end at closing()
    x_0 = 2 * np.e ** (gamma * u[i]) / (np.pi * u[i]) * quad(
        f_p, 0, np.pi / 2, args=(u[i], 0))[0]
    return x_0

def f_u():    # Main algorithm. Here, the summation integrals iterate
    k = 1    # Number of iterations

    for i in range(iterations):    # First overall series iteration n - 1 (initial 1st)
        y[0][i] = 2 * np.e ** (gamma * u[i]) / (np.pi * u[i]) * (-1) ** k * quad(
            f_p, -np.pi / 2, np.pi / 2, args=(u[i], k))[0]

    for j in range(iterations):    # Iteration loop. Goes until closing() or k = iterations
        if j > 0:
            for i in range(max_count):    # Sets the first iteration n - 1 (out of 3 for Aitken's) equal -
                y[0][i] = x_iter[i]       # to the previous Aitken's or the first iteration if j = 0 (1st)

        k += 1

        for i in range(max_count):    # Second iteration n loop for Aitken's (2nd)
            y[1][i] = 2 * np.e ** (gamma * u[i]) / (np.pi * u[i]) * (-1) ** k * quad(
                f_p, -np.pi / 2, np.pi / 2, args=(u[i], k))[0]

        k += 1

        for i in range(max_count):    # Third iteration (n + 1) loop for Aitken's (3rd)
            y[2][i] = 2 * np.e ** (gamma * u[i]) / (np.pi * u[i]) * (-1) ** k * quad(
                f_p, -np.pi / 2, np.pi / 2, args=(u[i], k))[0]

        k -= 1

        for i in range(max_count):    # Aitken's delta-squared method iteration (n') loop using the previous 3 (4th)
            y_denom[i] = (y[2][i] - y[1][i]) - (y[1][i] - y[0][i])

            if abs(y_denom[i]) < epsilon:
                print("Denominator too small to calculate. Exiting...\n")
                return

            y[3][i] = y[2][i] - ((y[2][i] - y[1][i]) ** 2) / y_denom[i]

        printer(k)

        for i in range(max_count):    # Sets current Aitken's iteration equal to first iteration
            x_iter[i] = y[3][i]
    closing(k)   # Multiplies by constants and adds non-iterated initial integral
    return
```

Figure 1. Generalized Laplace inversion code block, functions f_u() and f_0()

Essentially, the two functions `f_0()` and `f_u()` represent the initial and series-iterated integrals, respectively, of Eq. (10)—the latter of which is looped with variable `k` until a set number of iterations is reached. The domain, vector `u`, is initialized at set interval from a number as close to zero as possible (without invoking a domain error via overflow or rounding) to an arbitrary endpoint, with a set number of equally-spaced points in between. This vector is then fed into `f_u()` when the function is called, which begins the process of iterating the summation series portion of Eq. (10) using the `quad()` method of Gaussian adaptive integration on the image function defined in `f_p`. Upon calling `f_u()`, the first term of the series is integrated, then used as the starting point for the iterated Aitken's process until the iteration limit is reached. In contrast with how the symbolic equation might be approached, the initial integral is calculated at the end when function `f_0()` is called and the outputs subsequently added to the total value at each point. This is handled by the `closing()` function, which also multiplies the shared term of the equation with respect to `u` at each point immediately thereafter. The output is the Laplace inversion of the original input function.

A version of the algorithm in Figure 1 was created to invert six different Laplace representations of various known equations in sequence. The algorithm was put to the test over counts spanning orders of magnitude up to even 100,000 total iterations to ensure convergence to an extremely high degree. Each time, the overall function outputs were stored in an array `y` and plotted separately in a 3 x 2 subplot:

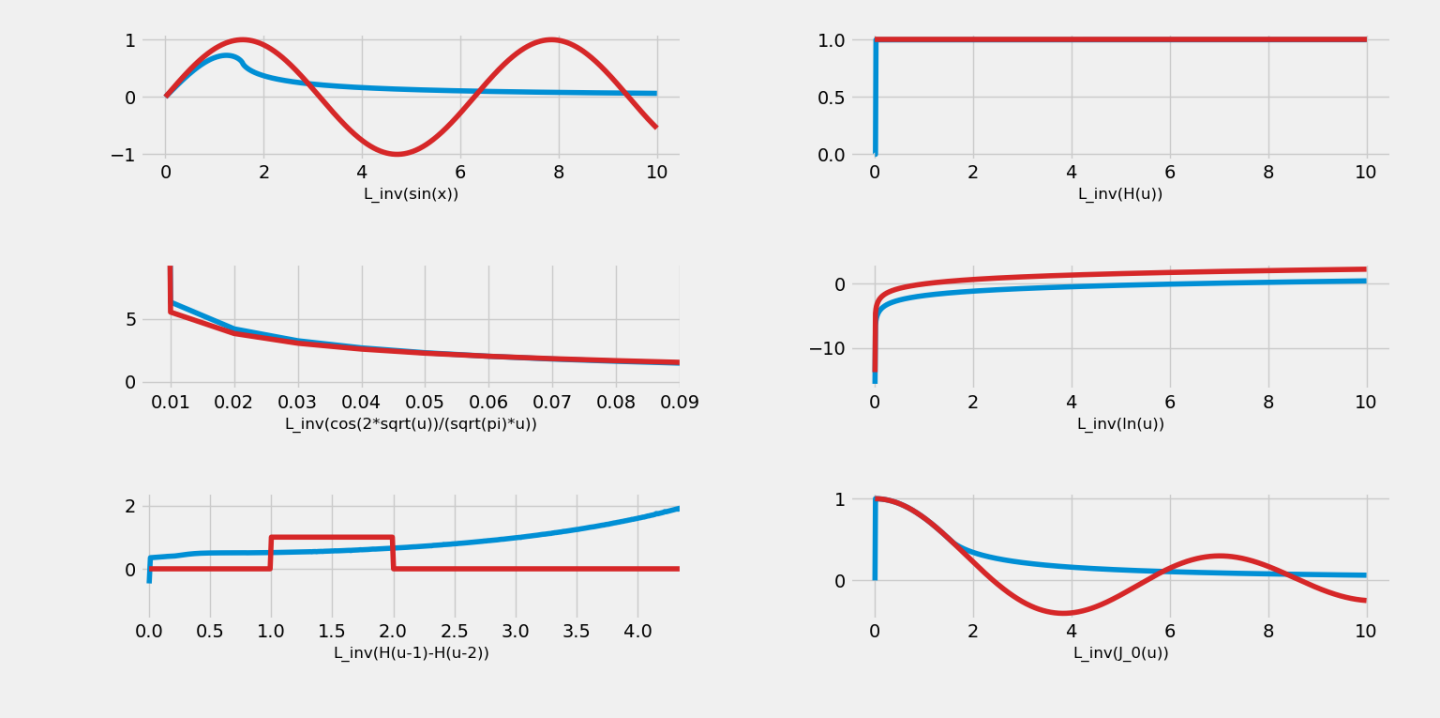

Figure 2. Demonstration of the numerical Laplace inversion method using Gaussian adaptive quadrature integration – 1,000 iterations

Red = Exact solution, Blue = Calculated numerical inversion

Upon observation of the resulting function plots, it is immediately evident that, while some of the approximations appear to be quite accurate, others drift apart from the functions they are meant to represent as x increases or even misrepresent their counterpart entirely. This was assumed to be due to numerical instability. However, any relevant possible sources of error were also considered: incorrect or inefficient Bromwich contour value, mischaracterization of the image function, approximation error introduced by discretization of the domain, inaccuracies due to an ill-suited numerical integration method or simply deceleration of convergence. Of these possibilities, only the viability of the integration method was not disproven to be the cause.

During the troubleshooting process, a pattern was observed: all of the functions that had the most pronounced deviation from the ideal were oscillatory in nature. This stands to reason, since the Gaussian method relies on a system of weights and nodes of orthogonal polynomials n which, in the specific case of Gauss-Kronrod, can exactly integrate polynomials to a degree of $3n + 1$ or less. This does not prove as useful when it comes to transforms whose integrals result in oscillating function however; approximating the infinitely repeating local minima and maxima when beginning from a function with a well-defined max, min and limit mean that polynomials of an ever-increasing degree are necessary to approximate even domains of a small magnitude, explaining the deviant behavior shown in the sine, Bessel, and even narrow step functions from Figure 2. Trapezoidal based methods are often even less suited to dealing with this issue; in fact, there exist few efficient methods of numerically integrating a transform of a high-frequency oscillating function with precision that does not require considerable computational power and/or does not work in a general sense, requiring significant adjustments on a case-by-case basis.

The slowing down spectrum solution is not expected to have oscillatory behavior however; logically, as lethargy increases there are fewer and fewer collisions, meaning the result would not oscillate, but rather behave somewhat similar to an exponential or polynomial function. Therefore, the adaptive Gauss-Kronrod quadrature

method applied by this algorithm is maintained as the candidate for numerical integration when moving on to the neutron slowing down spectrum's Laplace inverse approximation, albeit with significant optimizations, which will be discussed in detail in the following sections.

3.2. Laplace transform and inversion of the neutron spectrum equation

Almost everything has been prepared to perform the inverse Laplace transform on the neutron slowing down spectrum to finally reveal its result in a graphical representation.

First, returning to the definition of the neutron slowing down equation found in Eq. (7), there is still the case $u \geq q$ where the maximum change in lethargy per collision $(\Delta u)_{max}$ is exceeded, and the neutron source no longer has a direct influence on the collision rate density $F(u)$. This condition is vital to acquiring an accurate dataset of the change in energy of neutrons in a medium due to scattering, and so must also be transformed to Laplace space. Considering this equation is nearly identical to the one for the first collision interval that was previously transformed as Eq. (9), the transformation is trivial, and results in

$$F\hat{}(p) = \frac{1}{(1-\alpha)} \frac{1-e^{-q(p+1)}}{p+1} F\hat{}(p), \tag{11}$$

where the image function resolves to

$$F\hat{}(p) = S_0 \frac{1}{1 - \frac{1}{1-\alpha}\frac{1-e^{-q(p+1)}}{p+1}} = S_0 \frac{1}{1 - Q\hat{}(p+1)},$$

in which $Q\hat{}(p)$ is an auxiliary function defined for simplicity:

$$Q\hat{}(p) = \frac{1}{1-\alpha}\frac{1-e^{-qp}}{p},$$

and can be further simplified using the transform delay property's inverse, shown to be

$$\mathcal{L}[e^{au}G(u)] = G\hat{}(p-a) \rightarrow \mathcal{L}^{-1}[G\hat{}(p+1)] = e^{-u}G(u),$$

which factors out a term that dictates the decay of collision rate density for decreasing energy independent of collision and gives function $Q\hat{}(p)$ in a reasonably streamlined form

$$F(u) = S_0 e^{-u} \mathcal{L}^{-1}\left[\frac{1}{1-Q\hat{}(p)}\right]. \tag{12}$$

In Eq. (12), it is evident that a singularity is possible if $Q\hat{}(p) = 1$ for some value of p. It is simple to deduce that when $p = 1$

$$Q\hat{}(1) = \frac{1}{1-\alpha}\frac{1-e^{-q(1)}}{(1)} = \frac{1}{1-\alpha}\frac{1-e^{-q}}{1} = \frac{1}{1-\alpha}\frac{1-\alpha}{1},$$

$$Q\hat{}(1) = 1,$$

and with the knowledge that if any singularity in the real domain is greater than or equal to the Bromwich contour parameter γ the inverse operation will exhibit instability, it will be necessary to select a number for γ that is greater than, but as close as possible to 1.

Finally, substituting a dummy variable *x* in place of the auxiliary function *Q^(p)*, Eq. (12) can be converted to a series form that proves to be extremely advantageous for numerical analysis:

$$F(u) = S_0 e^{-u} \mathcal{L}^{-1}\left[\frac{1}{1-Q\hat{}(p)}\right] = S_0 e^{-u} \mathcal{L}^{-1}[1 + x + x^2 + x^3 + \cdots], \quad (13)$$

because, due to the linearity property of Laplace transforms, these terms can be separated and operated upon individually. In addition, each term represents a collision density for increasing collision count of neutrons *n*, therefore

$$F(u) = F_0(u) + F_1(u) + \cdots + F_n(u) = S_0 e^{-u}(\mathcal{L}^{-1}[1] + \mathcal{L}^{-1}[x] + \cdots + \mathcal{L}^{-1}[x^n]),$$

where the first three terms can be solved analytically. Therefore, the series will be cast in four terms: uncollided, exactly one or two collisions, and three or more collisions in a lumped term:

$$\mathcal{L}^{-1}\left[\frac{1}{1-x}\right] = \mathcal{L}^{-1}[1] + \mathcal{L}^{-1}[x] + \mathcal{L}^{-1}[x^2] + \mathcal{L}^{-1}\left[\frac{1}{1-x} - 1 - x - x^2\right]. \quad (14)$$

The first three terms in Eq. (14) can be evaluated analytically. The first is the Dirac delta function:

$$S_0 e^{-u} \mathcal{L}^{-1}[1] = S_0 e^{-u} \delta(u) = S_0 \delta(u),$$

then, the one-collision group can be evaluated using common Laplace function rules:

$$S_0 e^{-u} \mathcal{L}^{-1}[x] = \frac{S_0 e^{-u}}{1-\alpha} \mathcal{L}^{-1}\left[\frac{1 - e^{-qp}}{p}\right],$$

$$= \frac{S_0 e^{-u}}{1-\alpha} \mathcal{L}^{-1}\left[\frac{1}{p} - \frac{e^{-qp}}{p}\right] \quad \rightarrow \quad F_1(u) = \frac{S_0 e^{-u}}{1-\alpha}\big(H(u) - H(u-q)\big),$$

and finally, the two-collision group using the integration properties of Laplace transforms:

$$S_0 e^{-u} \mathcal{L}^{-1}[x^2] = \frac{S_0 e^{-u}}{(1-\alpha)^2} \mathcal{L}^{-1}\left[\left(\frac{1 - e^{-qp}}{p}\right)^2\right],$$

$$= \frac{S_0 e^{-u}}{(1-\alpha)^2} \mathcal{L}^{-1}\left[\frac{(1 - e^{-qp})^2}{p^2}\right] = \frac{S_0 e^{-u}}{(1-\alpha)^2} \mathcal{L}^{-1}\left[\frac{(1 - e^{-qp})(1 - e^{-qp})}{p^2}\right],$$

$$= \frac{S_0 e^{-u}}{(1-\alpha)^2} \mathcal{L}^{-1}\left[\frac{1 - 2e^{-qp} + e^{-2qp}}{p^2}\right] = \frac{S_0 e^{-u}}{(1-\alpha)^2} \mathcal{L}^{-1}\left[\frac{1}{p^2} - \frac{2e^{-qp}}{p^2} + \frac{e^{-2qp}}{p^2}\right],$$

$$= \frac{S_0 e^{-u}}{(1-\alpha)^2} \mathcal{L}^{-1}\left[\frac{1}{p}\left(\frac{1}{p} - \frac{2e^{-qp}}{p} + \frac{e^{-2qp}}{p}\right)\right],$$

$$= \frac{S_0 e^{-u}}{(1-\alpha)^2} \int_0^u H(u) - 2\int_0^u H(u-q) + \int_0^u H(u-2q),$$

$$F_2(u) = \frac{S_0 e^{-u}}{(1-\alpha)^2}\big(u - 2(u-q)H(u-q) + (u-2q)H(u-2q)\big).$$

The final lumped collision interval representing three or more collisions must now be numerically approximated. This is done using the algorithm from Figure 1 as the basis for the calculation, and, along with specifying all associated constants and parameters, writing the code for a new image function `f_p`. This starts with simplifying the last term in Eq. (14):

$$\mathcal{L}^{-1}\left[\frac{1}{1-x}-1-x-x^2\right]=\mathcal{L}^{-1}\left[\frac{x^3}{1-x}\right]=\mathcal{L}^{-1}\left[\frac{\left(\frac{1}{1-\alpha}\frac{1-e^{-qp}}{p}\right)^3}{1-\left(\frac{1}{1-\alpha}\frac{1-e^{-qp}}{p}\right)}\right],$$

$$=\frac{1}{(1-\alpha)^3}\mathcal{L}^{-1}\left[\frac{(1-e^{-qp})^3}{p^3\left(1-\left(\frac{1}{1-\alpha}\frac{1-e^{-qp}}{p}\right)\right)}\right]=\frac{1}{(1-\alpha)^3}\mathcal{L}^{-1}\left[\frac{(1-e^{-qp})^3}{-\frac{p^2(\alpha p-e^{-qp}-p+1)}{1-\alpha}}\right],$$

at which point the independent variable from the image function in Eq. (10) is substituted for p:

$$=\frac{1}{(1-\alpha)^2}\mathcal{L}^{-1}\left[\frac{\left(1-e^{-q\left(\gamma+i(\omega+k\pi)/u\right)}\right)^3}{-\left(\gamma+i(\omega+k\pi)/u\right)^2\left(\alpha\left(\gamma+i(\omega+k\pi)/u\right)-e^{-q\left(\gamma+i(\omega+k\pi)/u\right)}-\left(\gamma+i(\omega+k\pi)/u\right)+1\right)}\right]. \quad (15)$$

Since this is a complex function in terms of u and the integrand of Eq. (10) requires the real portion of this expression, Eq. (15) must be solved for its real value. This can be done using the `.real` method, a built-in Python operation for use with complex numbers.

At this point, the value of the Bromwich contour is chosen to be $\gamma = 1 + 1\cdot10^{-9}$, which was the lowest stable number possible to ensure maximum convergence and accuracy based on the recognition of a singularity when $p = 1$ in Eq. (12), and iron ($A = 56$) is chosen as the medium. Then, all of the analytical terms of Eq. (14) are plotted alongside the numerical solution to determine whether the solution makes sense:

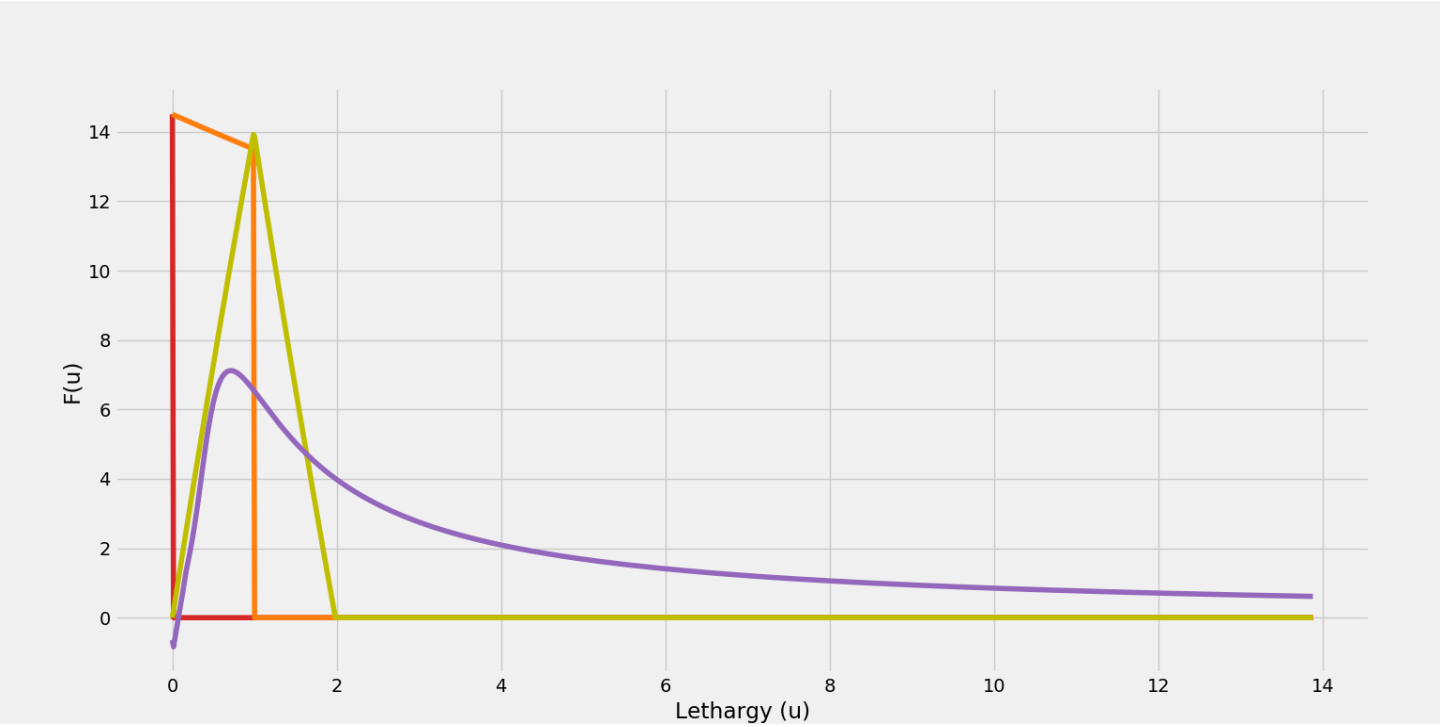

Figure 3. Collision group-based plot of neutron slowing down spectrum with respect to lethargy

Red = 0 collision group, Orange = 1 collision group, Yellow = 2 collision group, Purple = 3+ collision group

Overall, the logic of neutron behavior and the energy representation in Figure 3 is consistent. Of note is the discontinuity at the end of the first collision interval; it is evident that this induces some numerical instability in

the approximated term of Eq. (14) when the domain approaches this point. This is the maximum change in lethargy for a single collision, q, after which point no neutrons with only a single collision exist. A good way to illustrate its significance is by plotting in terms of the total collided neutrons only, $F_c(u)$:

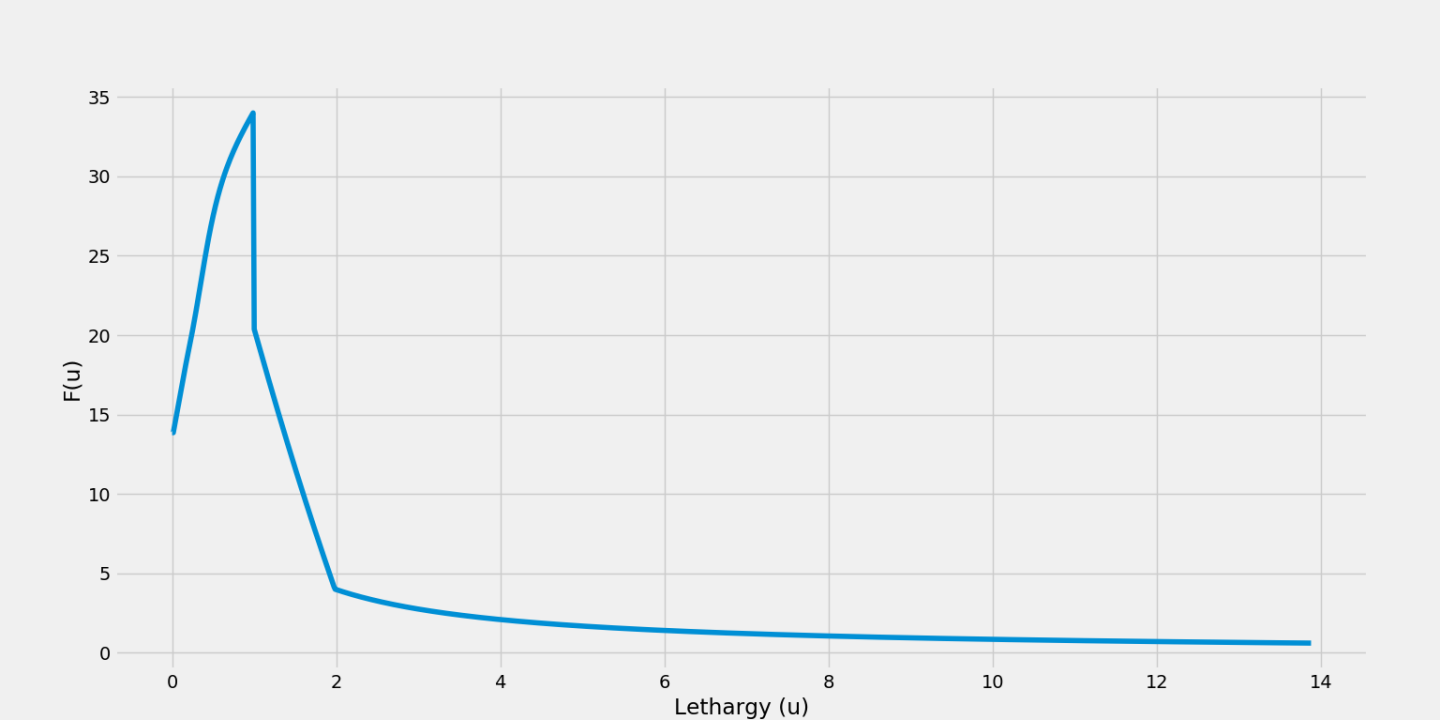

Figure 4. Plot of collided neutrons only for neutron slowing down spectrum with respect to lethargy.

4. Optimizations

4.1. Aitken's process

The first optimization made to the numerical solution was applying Aitken's delta-squared process to the slowly converging series in Eq. (10). The process behind this is the formula

$$(x_n)_{Aitken's} = x_{n+2} - \frac{(x_{n+2} - x_{n+1})^2}{(x_{n+2} - x_{n+1}) - (x_{n+1} - x_n)},$$

Where x_n is the iteration to be accelerated. This process was chosen because it provides a combination of efficiency and ease of implementation. The result is not quite quadratic convergence in most cases, however still highly accelerated.

4.2. Image function real component derivation

The image function of the Bromwich integral is a complex expression, but for the algorithmic calculation of Eq. (10) only the real part is required. This is easily accomplished using existing functionality built into Python. However, when used in the process of numerically deriving the neutron slowing down spectrum, it returned unsatisfactory results. The reason is unknown, but it is suspected that the Python operation was not able to interpret the complex image function expression correctly, or performed too many calculations, resulting in error propagation.

Therefore, it was decided that the best course of action was to derive the real component of the expression analytically, and use it directly as the image function. The results were greatly improved.

The derivation is very lengthy, therefore it is not presented here. Rather, the process is explained conceptually. Beginning with the assumption that the result will involve the division of complex expressions, the real solution is expected to be of the form

$$\frac{Re \cdot re + Im \cdot im}{Re^2 + Im^2},$$

where the capital and lowercase terms represent the real or imaginary components of the denominator and numerator of the divisors, respectively. Reorganizing Eq. (15) to standard complex form in the numerator and denominator gives an expression of the following form, rearranged to fit the previous expression:

$$\frac{a + ib}{c + id} \quad \rightarrow \quad \frac{c \cdot a + d \cdot b}{c^2 + d^2},$$

resulting in the real component of the image function, expressed in a Python function as `f_p` for the neutron slowing down algorithm:

```
def f_p(omega, u, k):    # Neutron lethargy image equation (f) for the fourth+ collision intervals, Laplace form
    b = (omega + k * np.pi) / u    # Imaginary component of I.V. Re(p)
    a = gamma    # Real component of I.V. (Bromwich contour) Im(p)

    f = (((a ** 2 - b ** 2) * (a * (alpha - 1) - np.e ** (-q * a) * np.cos(q * b) + 1) - (2 * a * b) * (np.e ** (
        -q * a) * np.sin(q * b) + b * (a - 1))) * (1 - 3 * np.e ** (-q * a) * np.cos(q * b) + 3 * np.e ** (
            -2 * q * a) * np.cos(2 * q * b) - np.e ** (-3 * q * a) * np.cos(3 * q * b)) + ((a ** 2 - b ** 2) * (
                np.e ** (-q * a) * np.sin(q * b) + b * (a - 1)) + (2 * a * b) * (a * (alpha - 1) - np.e ** (
                    -q * a) * np.cos(q * b) + 1)) * (3 * np.e ** (-q * a) * np.sin(q * b) - 3 * np.e ** (
                        -2 * q * a) * np.sin(2 * q * b) + np.e ** (-3 * q * a) * np.sin(3 * q * b))) / (((
                            a ** 2 - b ** 2) * (a * (alpha - 1) - np.e ** (-q * a) * np.cos(q * b) + 1) - (
                                2 * a * b) * (np.e ** (-q * a) * np.sin(q * b) + b * (a - 1))) ** 2 + ((
                                    a ** 2 - b ** 2) * (np.e ** (-q * a) * np.sin(q * b) + b * (a - 1)) + (
                                        2 * a * b) * (a * (alpha - 1) - np.e ** (-q * a) * np.cos(
                                            q * b) + 1)) ** 2) * np.cos(omega)

    return f
```

4.3. Numba's just-in-time function compiler

Python is an interpreted programming language. This means that, upon execution, a software processor known as an interpreter reads the code, translates it into a sequence of subroutines, and then sends the resulting machine code to the CPU. In contrast, in a compiled language such as C the code is first run through a compiler, which takes the language code, translates all of it to machine code and creates a machine code executable file that can then be run directly. The latter process has its own disadvantages, but is much more efficient, resulting in a drastically lower computation time.

One way to take advantage of compiled code efficiency in Python is to make use of a just-in-time compiler. This is a compiler that is initiated by the interpreter on one or more sections of the code and outputs a set of machine language instructions so that whenever the section of code is run, it is executed directly by the CPU. This mechanism is extremely useful when executing a numerically intensive piece of code or a section of code that is iterated many times in succession. Since the previously discussed image function of Eq. (10) is both a complex numerical operation and may be run many hundreds of times throughout the course of the algorithm, it is an ideal candidate for just-in-time compilation.

The just-in-time compiler chosen in this instance comes from the `numba` package, a collection of functions that optimize Python code efficiency, and uses a decoration before functions to denote what to compile. Since

the image function is to be run numerous times, there is a cache option that stores the compiled code in memory so that it only needs to be compiled once. The code for executing the compiler in such a way is:

```
@jit(nopython=True, cache=True)
```

This technique has a massive effect on a program's executional efficiency. In the case of the neutron slowing down spectrum algorithm running 1000 iterations, the runtime with no compilation was:

```
Runtime: 0:29:09.848013
```

whereas just-in-time compiling and caching the image function `f_p` completed with remarkable speed at:

```
Runtime: 0:00:27.877441
```

5. *Conclusions*

Although numerically approximating an unknown equation that is inherently unstable is a very difficult procedure to get right—especially with consideration for the fact that it is not possible to verify the solution directly by an exact analytical process—using a combination of logic, careful selection of parameters, testing and completing as much ancillary analysis as possible is an excellent combination to make up for that fact. Even if there is error in the result, of which, at and around $u = q$, there likely is a not-insignificant amount, the information obtained through the numerical portion of this project is indispensable and is not obtainable through analytical means. As computational power increases, ever more problems previously thought impossible or impractical to the degree of pointlessness will continue to be solved though numerical methods—but generally not without significant analytical undertaking to provide the foundation.

6. *References*

[1] https://people.math.ethz.ch/~hiptmair/Seminars/CONVQUAD/Articles/DAM79.pdf

[2] https://www.nada.kth.se/~olofr/Approx/IzzoFrachon_ApprTh_project.pdf

[3] http://www.mi.sanu.ac.rs/~gvm/radovi/MilovanovicStanic.pdf

[4] *Hazewinkel, Michiel, ed. (2001) [1994], "Gauss quadrature formula", Encyclopedia of Mathematics, Springer Science+Business Media B.V. / Kluwer Academic Publishers, ISBN 978-1-55608-010-4*

[5] Kendall E. Atkinson, *An Introduction to Numerical Analysis*, (1989) John Wiley & Sons, Inc, ISBN 0-471-62489-6