Resonance Scattering Treatment Leveraging the Windowed Multipole Formalism

Gavin Ridley, Benoit Forget, Timothy Burke

March 14, 2023

1 Abstract

A new method for directly sampling the resonance upscattering effect is presented. Alternatives have relied on inefficient rejection sampling techniques or large tabular storage of relative velocities. None of these approaches, which require pointwise energy data, are particularly well suited to the windowed multipole cross section representation. The new method called multipole analytic resonance scattering (MARS) overcomes these limitations by inverse transform sampling from the target relative velocity distribution where the cross section is expressed in the multipole formalism. The closed form relative speed distribution contains a novel special function we deem the incomplete Faddeeva function, and we present the first results on its efficient numerical evaluation.

2 Introduction

Early continuous energy Monte Carlo neutron transport programs sampled scattering from nuclei in thermal motion assuming that the scattering cross section is effectively constant within the scattering kernel [17]. However, as [32, 30] detail , the resulting scattering kernel implied by the constant cross section approximation may be far from the actual double-differential cross section near a scattering resonance. As shown in [24, 29], the resulting error tends to cause a worst-case 11% underestimation of the Doppler feedback coefficient in a PWR, with even larger discrepancies in HTGR problems.

As exhibited by the PRAGMA project [8, 9], the conventional methods for treating this effect leave something to be desired on graphics processing unit (GPU) architectures, which constitute the majority of computational power on today's leading supercomputers. Due to the unique architecture of the GPU, algorithmic modifications to standard Monte Carlo algorithms for neutron tracking can tangibly accelerate computation [9]. In the same direction, we herein present a GPU-friendly method for handling resonance upscatter when the windowed multipole (WMP) [23] formalism is employed to represent cross sections. In particular, the heuristic for fast GPU code is to avoid rejection sampling and accesses to distantly spaced places in memory, which the new method achieves. To give context to

the new method, we first recall some of the conventional methods for modeling resonance upscattering.

The Doppler Broadening Rejection Correction (DBRC) [4] was one of the early proposed techniques to treat the effect of strong variations of the interaction cross section within the energetic vicinity of a scattering neutron, whereas $S(\alpha, \beta)$ tables had been used prior [11]. The method has since been implemented in numerous continuous energy Monte Carlo neutron transport programs [4, 39, 35, 18], and has shown to successfully model the resonance upscatter effect. However, DBRC suffers from rejection probabilities as high as 99.995% [31] for neutron energies in the vicinity of a resonance.

The weight correction method (WCM) [24] also successfully models the effect of resonances on the double-differential cross section of nuclei in thermal motion. This method adjusts the weight of particles to provide numerically correct results even when the constant cross section double-differential free gas distribution is employed. WCM carries the same benefit of our newly proposed method of not requiring an additional rejection loop or tables; however, adjustments to the particle weights introduce substantial variance to the overall Monte Carlo simulation, thus degrading estimates on quantities of interest [8].

Another technique known as target motion sampling (TMS) [36] can be used to model the resonance upscatter effect. However, this method is only applicable to Monte Carlo neutron transport programs employing the delta tracking technique. Unfortunately, performance of delta tracking appears to be lackluster on GPUs [33].

The relative velocity sampling (RVS) method [38, 31] was created to ameliorate the high rejection rates characteristic to the rejection algorithms used to model resonance upscatter. These schemes, in essence, sample probability distributions proportional to $f(x)g(x)$, where $f(x)$ is a distribution and $g(x) \in [0, 1]$. One then samples from $f(x)$ and accepts the sample with probability $g(x)$. The RVS method moves the direct sampling from the thermal motion term to the cross section term, thus worsening the average case rejection rate but massively improving the worst-case rejection rate.

The relative speed tabulation (RST) method [8] was developed to address the rejection sampling performance impact on GPUs. RST is the first resonance upscatter modeling technique not requiring a rejection loop or bivariate scattering distribution tables. The key observation underlying RST is that the target velocity distribution can be factorized into a marginal relative speed distribution encapsulating the information about the resonances and a simple distribution of the target polar angle conditioned on the target relative speed. Consequently, the univariate relative speed cumulative distribution can be tabulated in select areas of the pointwise cross section, and the conditional polar angle distribution is then directly sampled without requiring any additional data or rejection step.

Despite its simplicity and efficacy on GPUs, the RST method comes with some clear disadvantages. Gigabytes of additional memory are used in storing the relative speed cumulative distributions (CDFs) at each energy point and temperature on a pointwise cross section representation, if all nuclides have the resonance upscatter effect treated. The resonance influence on the double differential cross section is thus restricted for practical reasons to a select few nuclides in the problem to avoid extreme memory usage. Moreover, the method introduces some discretization error in temperature, although this was shown to be a reasonable approximation.

If one could avoid pre-tabulated relative speed distributions, this could substantially

reduce the memory needs and avoid costly divergent memory accesses. Reducing serialized memory accesses on GPUs typically leads to significant speedups.

We propose a new method which does just that, and achieves this using the windowed multipole (WMP) cross section representation [23]. Our new method introduces a novel special function we have deemed the incomplete Faddeeva function which encodes the behavior of the temperature-dependent influence of resonances on the double differential scattering distribution. It relies on the numerical inversion of the analytic representation of the relative speed distribution under the WMP formalism, and polar angle sampling in the same manner as the RST method, but without any precomputed tables.

This contrasts the WMP-based target velocity sampling technique presented in [27], which is similar in nature to the newly presented method in this work. However, [27]'s method relies on the separation of the target velocity distribution into a zero kelvin cross section component and a Maxwell-Boltzmann component. This method thus requires a numerical inversion step of the integrated scattering cross section function wrapped in a rejection loop, similar to [31] but instead using a functional representation of the integrated cross section rather than tabular.

The algorithm presented in [6] shows how the relative speed can be sampled in the windowed multipole framework similarly to our work. However, [6] relies on fitting a sum of Gaussians to replace the poles in Eq. 4, thus representing the zero kelvin scattering cross section in the form:

$$\sigma(E) = \frac{1}{E} \sum_k \sum_j \left[h_{s,k,j} e^{\left(\sqrt{E}-u_k\right)^2 / w_{s,k,j}^2} + h_{a,k,j} e^{\left(\sqrt{E}-u_k\right)^2 / w_{a,k,j}^2} \right] \tag{1}$$

While it remains to be seen that Gaussians can be used to approximate all poles appearing in a windowed multipole library, this proposed approximation introduces a considerable number of degrees of freedom to the problem, with eighteen unknowns for each pole. The optimization problem thus encountered is highly nonlinear and nonconvex leading to fitting difficulties. Additionally, scattering kernels in [6]'s formalism cannot be straightforwardly differentiated with respect to windowed multipole parameters which would be needed to perform sensitivity analysis.

Our new method, multipole analytic resonance scattering (MARS), only requires the same windowed multipole data as would be used in a calculation without any treatment of the resonance upscatter effect, avoiding the need for additional tables or fitting steps. We demonstrate the new method's negligible performance overhead compared to other methods for sampling the resonance upscatter effect on CPU architectures, with future work exploring its optimized implementation on GPUs.

3 Theory

Rothenstein et. al. [32] expounds the rigorous Doppler broadened double-differential cross section. Since the exact expression for the lab frame scattering distribution is quite complicated, authors presenting algorithms to model it typically choose to forgo the lab frame expression and instead reason in terms of joint distributions of the target velocity and direction cosine relative to the direction of projectile motion. After sampling the target speed and

direction cosine, standard two-body collision kinematics for elastic scattering are employed, where the center of mass angular distribution comes from the nuclear data file. The resulting physics matches the complicated expressions of [32].

Similarly considering the distribution of collision target velocities, this joint distribution is:

$$f(V, \mu) = C v_r \sigma(v_r) M(T, V) f'(\mu) \tag{2}$$

where M is the Maxwell-Boltzmann distribution of target speeds:

$$M(T, V) = \frac{4}{\sqrt{\pi}} \beta^3 V^2 e^{-\beta^2 V^3} \tag{3}$$

and the variable with units of inverse velocity $\beta = \sqrt{\frac{A}{2kT}}$ parameterizes the target velocities, and the distribution $f'(\mu)$ is a uniform distribution between -1 and 1. The other variables are C, the distribution's normalizing constant; v_r, the relative speed of the neutron with respect to the target; σ, the zero kelvin scattering cross section; A, the target mass; k, the Boltzmann constant; T, the absolute temperature; and V, the target speed.

Classically, the approximation that $\sigma(v_r)$ is constant has been employed. However, it was shown in [30, 24] that this approximation is incorrect in the vicinity of resonances, where $\sigma(v_r)$ varies over a few orders of magnitude, preferentially causing scattering with targets of relative velocity more closely matching the scattering resonance peaks. The various aforementioned techniques are all just methods for sampling from the distribution of Eq. 2 with arbitrary forms of the function $\sigma(v_r)$.

More specific knowledge about the form of $\sigma(v_r)$ can be employed. It has been shown extensively [28, 13] at this point that the cross section is accurately represented as a sum of poles in addition to a low order Laurent expansion ($N \approx 7$), vis:

$$\sigma(E) = \frac{1}{E} \left(\Re \left[\sum_j \frac{r_j}{p_j - \sqrt{E}} \right] + \sum_{n=0}^{N} a_n E^{n/2} \right) \tag{4}$$

In fact, for the purposes of sampling the resonance upscattering effect, we claim and later numerically demonstrate that the narrow range of attainable v_r leads the zero-kelvin cross section to be accurately represented as a single pole and a linear term, over a sufficiently narrow range of energies:

$$\sigma(E) = \frac{1}{E} \Re \left[\frac{r_j}{p_j - \sqrt{E}} \right] + \sigma_0 + \sigma_1 \sqrt{E} \tag{5}$$

The σ_0 and σ_1 terms are calculated through a linearization process to avoid some complexities with higher order polynomial fitting. An algorithm for finding the best values of σ_0 and σ_1 is presented in section 3.4.

With this approximation, we use the technique developed in [8]: rather than attempting to sample the target speed (V) and direction cosine (μ), one instead samples first the relative

velocity v_r, and then samples μ from the distribution of μ conditioned on v_r. The distribution in this form for projectile speed v is thus[8]:

$$P(v_r|v) = \left(e^{-\beta^2(v-v_r)^2} - e^{-\beta^2(v+v_r)^2}\right) v_r^2 \sigma_0(v_r) \tag{6}$$

Next, as previously shown for Doppler-broadening of the Windowed Multipole format [23] and the original full multipole format [21], we note the term $e^{-\beta^2(v+v_r)}$ to be negligible and use the following:

$$P(v_r|v) \approx e^{-\beta^2(v-v_r)^2} v_r^2 \sigma_0(v_r) \tag{7}$$

At this point, we write the multipole cross section in terms of the relative velocity rather than in terms of energy. The formula to use is:

$$\sigma(v_r) = \frac{2}{m_n v_r^2} \Re \left[\frac{r_j}{p_j - \sqrt{2m_n} v_r}\right] + \sigma_0 + \sigma_1 \sqrt{2m_n} v_r \tag{8}$$

If we define the auxiliary variables $x = \beta(v_r - v)$ and $y = \beta v$, and insert Eq. 8 in the marginal target collision rate distribution in terms of relative speed, Eq. 7, some algebra reveals that:

$$p(x|y) = e^{-x^2} \left(\Re \left[\frac{\beta r_j}{z - x}\right] + \beta^{-2}(x+y)^2 \left(\sigma_0 + \sigma_1 x\right)\right) \tag{9}$$

where $z = \beta p_j - y$, which represents a dimensionless measure of the energy gap between the center of the Maxwell-Boltzmann distribution and the location of the resonance.

At this point, we can consider the CDF for the random variable x (dimensionless target velocity) conditioned on y (dimensionless projectile velocity). Integrating Eq. 9 yields:

$$CP(x|y) = \int_{-\infty}^{x} e^{-x'^2} \left(\Re \left[\frac{\beta r_j}{z - x'}\right] + \beta^{-2}(x'+y)^2 \left(\sigma_0 + \sigma_1 x'\right)\right) \mathrm{d}x' \tag{10}$$

where C is the normalizing constant. After distributing the integral and interchanging the $\Re$ operator with integration, we obtain:

$$CP(x|y) = \Re \left[\frac{r_j \pi}{i\beta^{-1}} w(z,x)\right] + \frac{\beta^{-2}\sigma_0}{4}(-2e^{-x^2}(x+2y) + \sqrt{\pi}(1+2y^2)(1+\mathrm{erf}(x))) + \frac{1}{2\beta^2}\sigma_1 e^{-x^2}\left(1 + (x+y)^2 + \sqrt{\pi} y (1+\mathrm{erf}(x))\right) \tag{11}$$

where the normalizing constant for the distribution is:

$$C = \Re \left[\frac{r_j \pi}{i\beta^{-1}} w(z)\right] + \frac{\sqrt{\pi}}{2\beta^2}\left(\sigma_0(1+2y^2) + \sigma_1 y\right) \quad , \tag{12}$$

which we point out is nothing more than the Doppler-broadened scattering cross section at temperature T under the single pole approximation.

The new special function we deem the "incomplete Faddeeva function" is defined as:

$$w(z,x) = \frac{i}{\pi}\int_{-\infty}^{x} \frac{e^{-t^2}}{z-t}\,\mathrm{d}t \tag{13}$$

And it can easily be seen that $w(z,\infty) = w(z)$ as per the definition of the Faddeeva function for $\Im[z] > 0$:

$$w(z) = \frac{i}{\pi}\int_{-\infty}^{\infty} \frac{e^{-t^2}}{z-t}\,\mathrm{d}t \tag{14}$$

Indeed, for this application, $\Im[z] > 0$ and we maintain this assumption going forward. A specialized root finder has been developed to quickly invert this CDF and consequently sample the relative velocity. This thus constitutes a method to sample target velocities without rejection sampling or extensive tables.

The novelty in our approach lies entirely in the treatment of the zero kelvin cross section and analytical representation of the relative speed cumulative distribution. After sampling from the relative speed distribution, we must sample the target polar angle distribution conditioned on the relative speed as done in [8]. For completeness, we conclude with the CDF of the target speed:

$$C(V|v_r) \propto \begin{cases} 0 & V \leq |v_r - v| \\ 1 - e^{-\beta^2 V^2} & |v_r - v| < V < v_r + v \\ 1 & v_r + v \leq V \end{cases} \tag{15}$$

for which [8] provides a straightforward sampling technique. The remaining work is purely numerical, particularly in requiring an efficient, reasonably accurate algorithm for the incomplete Faddeeva function $w(z,x)$.

3.1 The Incomplete Faddeeva Function

The forthcoming discussion explores the properties of the incomplete Faddeeva function, with a particular focus on properties which can be leveraged to obtain efficient numerical approximations to it. We advise the reader that absorbing this section in depth is not necessary to grasp the claims and algorithms made herein.

The incomplete Faddeeva function, as defined by Eq. 13 is $w : \mathbb{C} \times \mathbb{R} \to \mathbb{C}$. This section attempts to build some intuition as to how this function behaves as z and x individually vary. In resonance upscatter treatment, z parametrizes the location, height, and width of the resonance with respect to the Maxwell-Boltzmann distributed velocities. Values of $\Re[z] = 0$ correspond to scattering resonances exactly situated at the mean component of relative speed along the neutron's line of flight predicted by the Maxwell-Boltzmann distribution. Values of $\Re[z] > 0$ correspond to resonances at higher energies than the particle's energy, and therefore induce preferential scattering with relative velocities higher than the incident velocity, and vice-versa for $\Re[z] < 0$. x parametrizes the dimensionless target relative speed. Small values of $\Im[z]$ imply tall, narrow resonances, with a limit of $\Im[z] = 0$ being a singularity representing a resonance of infinite cross section. Large values of $\Im[z]$ model wider, weaker resonances.

Fig. 1 plots the incomplete Faddeeva function as a function of z for a few values of x. This illustrates the rapidly varying behavior of this function for small values of $\Im[z]$ when $\Re[z] \approx x$, where a sharp peak follows the value of x near the real line. On top of that, it shows the approach to the familiarly shaped $w(z)$ as $x \to \infty$. The figure shows that $w(z, x)$ is an increasing (in the sense of increasing in both real and imaginary part) function in x for many values of z, but not all. This behavior is explained by the following asymptotic analysis.

Humliĉek [19] provides the following asymptotic formula for the Faddeeva function without derivation:

$$w(z) \approx \frac{i}{\sqrt{\pi} z} \quad . \tag{16}$$

Considering the definition of the Faddeeva function once more in Eq. 14, This approximation results from supposing that if z is large, and that only values of t near zero contribute to the integral, we can approximate $w(z)$ as:

$$w(z) \approx \frac{i}{\pi} \int_{-\infty}^{\infty} \frac{e^{-t^2}}{z} \, \mathrm{d}t \tag{17}$$

which immediately yields the asymptotic estimate of Eq. 16. Proceeding with the same approximation in the context of the incomplete Faddeeva function, we thus obtain:

$$w(z, x) \approx \frac{i}{2\sqrt{\pi} z} \left(\mathrm{erf}(x) + 1 \right) \qquad \text{if} \quad |z| \gg 1 \tag{18}$$

Suggesting a close connection between the incomplete Faddeeva function's behavior in x and the error function. In fact, this hunch is confirmed by the following identity which connects the incomplete Faddeeva to the standard Faddeeva function:

$$w(z, x) = \frac{1}{2}(1 + \mathrm{erf}(x))w(z) + \frac{ie^{-x^2}}{\pi} \int_0^{\infty} \frac{e^{-t^2} e^{2itz} \, \mathrm{d}t}{i(x - z) + t} \tag{19}$$

Proof of this relation is provided in Appendix A.

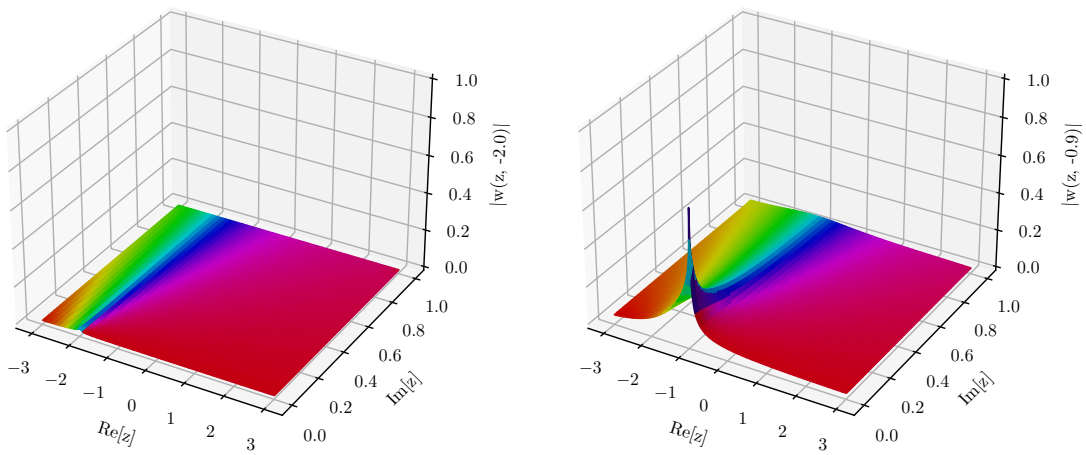
An even more accurate asymptotic estimate can be obtained for large $\Re[z]$, approximating the pole term as

$$\frac{1}{z - t} \approx \frac{1}{2z} \left(1 + e^{2t/z} \right) \quad . \tag{20}$$

This matches the value, slope, and curvature with respect to t of the pole about $t = 0$. Substituting this back to Eq. 13 and adjusting the expression such that $\Re[w(z, x)]$ is strictly increasing (as suggested by Eq. 18), and matching the asymptotic value of $w(z)$ as suggested by Eq. 19, results in

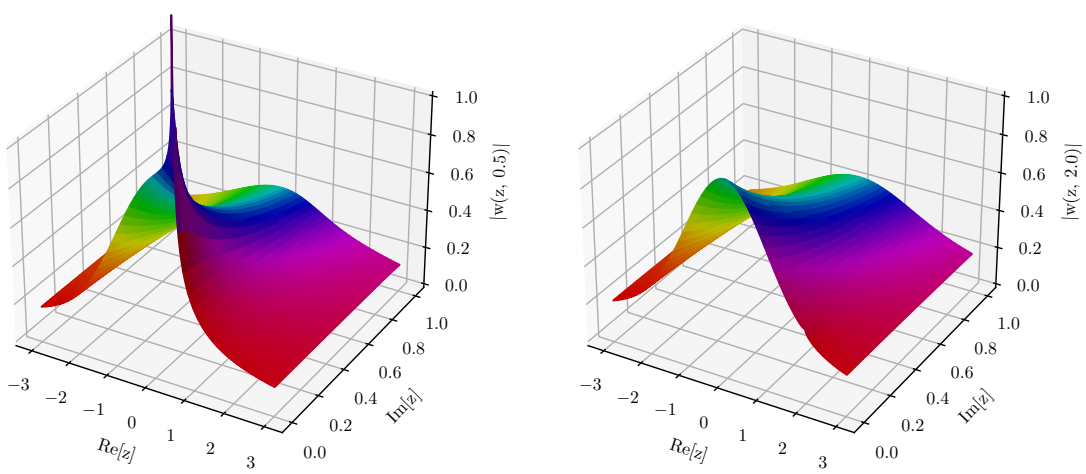
$$w(z, x) \approx \frac{1}{2} \left(1 + \mathrm{erf}\left(x - \Re[z]^{-1} \right) \right) w(z) \quad . \tag{21}$$

Eq. 21 is sufficiently accurate to be used in practical computations, as shown by Fig. 3. In the context of resonance scattering, Eq. 21 shows that the relative speed distribution gets shifted forward by a nondimensionalized factor of $1/\Re[z]$ with its influence scaling by $\Re[r_j w(z)]$ times the resonances residue as suggested by Eq. 11.



(a) $w(z, -2.0)$

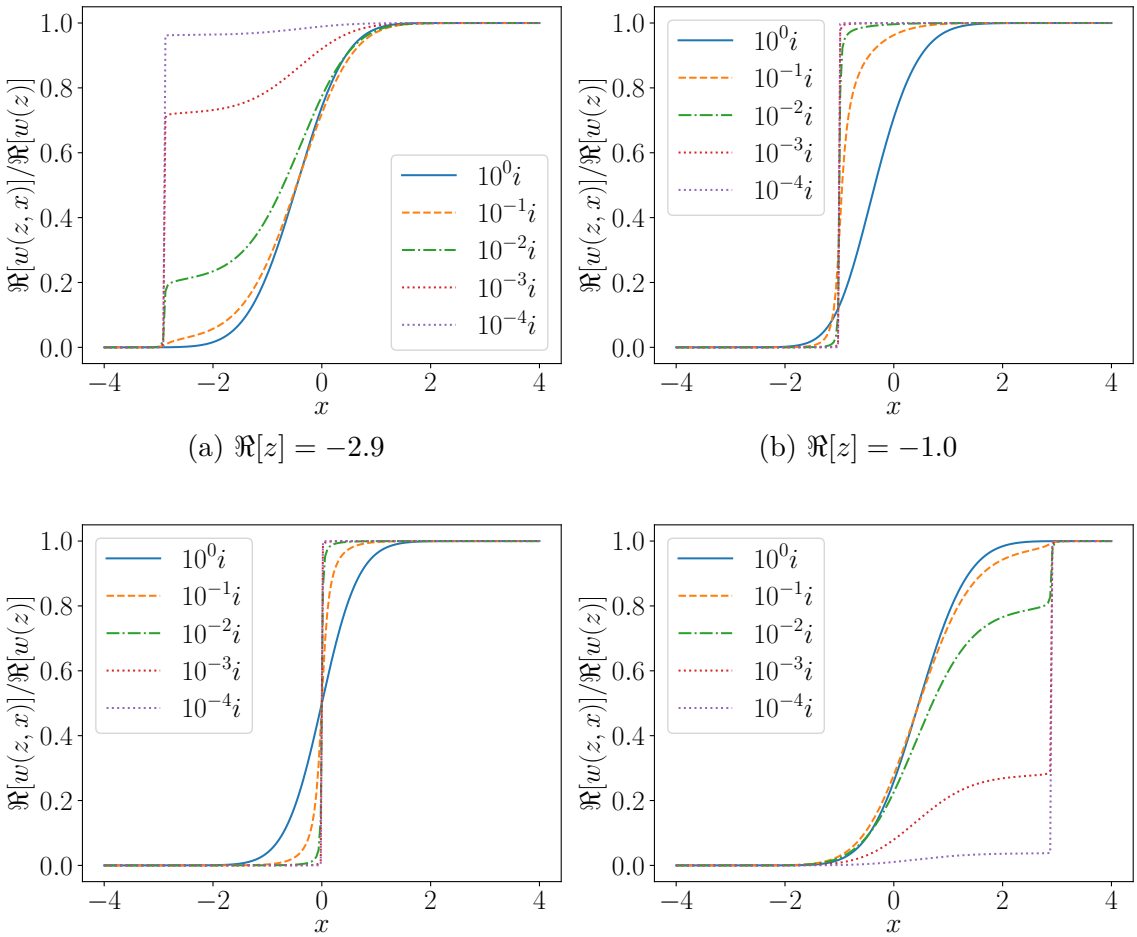
(b) $w(z, -0.9)$



(c) $w(z, 0.5)$

(d) $w(z, 2.0)$

Figure 1: $w(z, x)$ for a few values of x. The height represents the magnitude, and coloring is done by phase.



(a) $\Re[z] = -2.9$ (b) $\Re[z] = -1.0$

(c) $\Re[z] = 0.0$ (d) $\Re[z] = 2.9$

Figure 2: $\Re[w(z,x)]$ for a few values of z. The legend is the imaginary number added to the real part specified in each figure's caption. The plotted quantity is normalized by $\Re[w(z)]$ so all lines tend to unity as x grows.

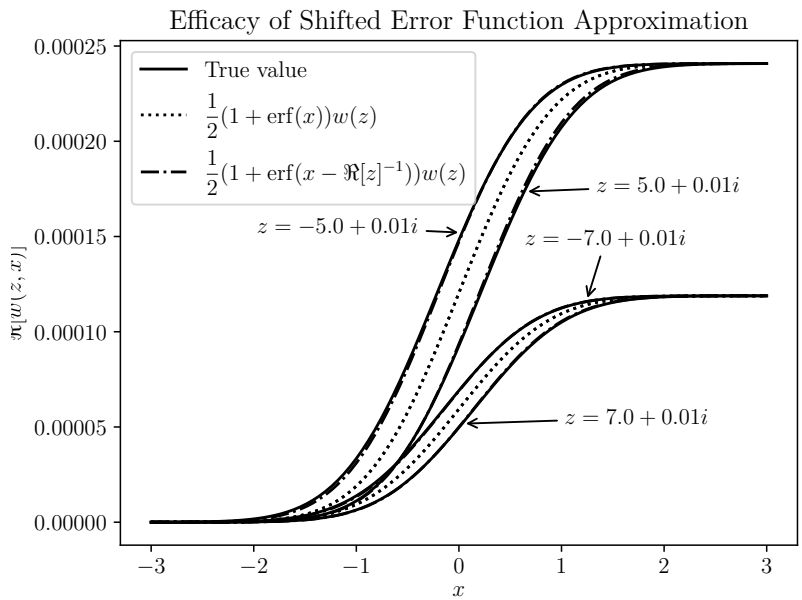


Figure 3: Accuracy of the $w(z, x)$ approximation for $|\Re[z]| > 5$. An unshifted error function is shown for comparison, representing a more naive asymptotic approximation to $w(z, x)$.

Another useful property of the incomplete Faddeeva function is a simple connection between its derivative in the complex plane and its value. This is similar in nature to the derivative of the Faddeeva function [1]:

$$\frac{\mathrm{d}w(z)}{\mathrm{d}z} = \frac{2i}{\sqrt{\pi}} - 2zw(z) \tag{22}$$

The relation we have obtained generalizes this as:

$$\frac{\mathrm{d}w(z,x)}{\mathrm{d}z} = \frac{i}{\sqrt{\pi}}\left(1 + \mathrm{erf}(x) + \frac{e^{-x^2}}{\sqrt{\pi}(x-z)}\right) - 2zw(z,x) \tag{23}$$

which clearly maintains consistency with Eq. 22 as $x \to \infty$. This fairly simple connection between the derivative of $w(z, x)$ and its value can be utilized for efficient sensitivity analysis of the scattering kernel with respect to windowed multipole parameters.

The forthcoming discussion presents some further concepts in the direction of efficient numerical evaluation of the incomplete Faddeeva function. Going forward, we denote the second integral appearing in Eq. 19 as:

$$I(z,m) = \int_0^\infty \frac{e^{-t^2}e^{2itz}\,\mathrm{d}t}{im + t} \tag{24}$$

where $\mathbb{R} \ni m = x - z$. While it may seem that a half-range Gauss-Hermite quadrature may work well to efficiently approximate Eq. 24, this is not the case. Firstly, complex exponentials would have to be calculated at each quadrature point. Secondly, as the real

part of z grows, the integrand oscillates more. In practice, values of $\Re[z] > 10$ are frequently encountered, and low degree quadratures would not capture the oscillation. Additionally, given that the value of $\Im[z]$ is small, the denominator becomes nearly singular when $x \approx \Re[z]$. In fact, when $\Im[z] = 0$, $I(z, m)$ becomes a discontinuous function in x when interpreted as a principal value integral.

Eq. 24 is equivalent to the *incomplete Goodwin-Staton integral*, referred to in [12]. However, to our knowledge, only asymptotic analysis has been performed on this type of integral before, without any development of numerical routines. Recent work in the field of finance [3] presents results for computing what the authors define as the *extended incomplete Goodwin-Staton integral*, for which the $\nu = 1$ case is of interest in the present discussion. Unfortunately, the authors' numerical method works for all cases except $\nu = 1$, suggesting Eq. 24 to be of a fundamentally different nature.

In our experience, the difficulty with large $\Re[z]$ cannot be ameliorated by a stationary phase technique [5], as these tend to accentuate the pole behavior and remain of similar difficulty for half range Gauss-Hermite quadrature.

In the windowed multipole method, $\Im[z]$ is near zero, as shown in Fig. 4. Therefore, we can expect to frequently encounter nearly singular integrands in Eq. 24. An efficient numerical technique which explicitly treats this behavior can be devised by first noticing that $I(z, m)$ satisfies this differential equation in the complex plane:

$$\frac{\mathrm{d}I}{\mathrm{d}z} + 2zI = \frac{1}{x - z} \tag{25}$$

This can be used to connect the value $I(z_0, x)$ at a point z_0 to another point z_1. The integrating factor technique shows that:

$$I(z_1, x) = \int_{z_0}^{z_1} \frac{e^{t^2 - z_1^2}\,\mathrm{d}t}{x - t} + e^{z_0^2 - z_1^2} I(z_0, x) \tag{26}$$

If the distance between z_0 and z_1 is small, the exponential in the numerator of Eq. 26 can be Taylor expanded about z_0 to yield an efficient numerical scheme.

We have found that the imaginary part of $I(\Re[z], x)$ can be calculated with a closed-form, discontinuous-in x formula. Because the nearly discontinuous behavior of $I(z, x)$ in x is the main source of difficulty here, Eq. 26 can be used to resolve this behavior accurately after calculating the value of $I(\Re[z], x)$. The real part of $I(\Re[z], x)$ is continuous in x but not available in formula in terms of elementary functions; however, it is readily amenable to numerical approximation. In conclusion, the real and imaginary part of $I(\Re[z], x)$ contrast each other: the former is readily numerically approximated by series or similar methods, whereas the latter is discontinuous and therefore not amenable to series or rational function approximation, and fortunately has an exact formula.

The first goal at hand is to calculate $\Im[I(z, m)]$ for real values of z. This can be written as:

$$\Im[I(z, m)] = \int_0^\infty \frac{e^{-t^2}\left(t\sin(2zt) - m\cos(2zt)\right)\mathrm{d}t}{t^2 + m^2} \quad . \tag{27}$$

The linearity of integration can be distributed over both trigonometric terms. Each of the resulting integrals can be found as respective sine and cosine transform integrals, which are

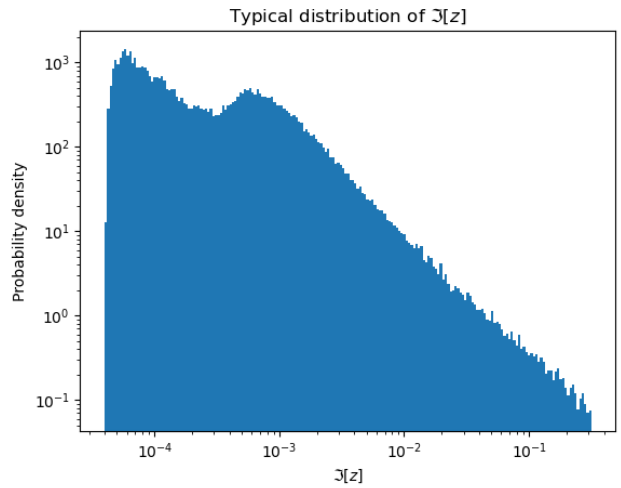


Figure 4: Distribution of imaginary part of the poles in the windowed multipole method. These were collected from every nuclide in OpenMC's regression testing dataset, based on ENDFVII.1.

listed in [14]. Combining results from the sine and cosine transform tables yields:

$$\Im[I(z,m)] = \pi e^{-z^2+(m+z)^2}\left(\frac{1}{2}\mathrm{erf}(m+z) - \mathrm{sign}(m)\right) \quad z \in \mathbb{R} \tag{28}$$

Next, we can find an approximation for the real part of $I(z,m)$:

$$\Re[I(z,m)] = \int_0^\infty \frac{e^{-t^2}\left(t\cos(2zt) + m\sin(2zt)\right)\mathrm{d}t}{t^2+m^2} \quad . \tag{29}$$

This expression can neither be written terms of elementary nor special functions, to our knowledge. In order to manipulate it to obtain a numerical expression, consider the auxiliary function:

$$R(z,m) = \int_0^\infty \frac{e^{-t^2}\sin(2zt)\,\mathrm{d}t}{t^2+m^2} \tag{30}$$

which again, of course, cannot be represented in terms of elementary or special functions. This pinpoints the difficulty in calculating the real part of $I(z,m)$ because:

$$\Re[I(z,m)] = mR(z,m) + \frac{1}{2}\frac{\partial R}{\partial z} \quad . \tag{31}$$

We thus seek a straightforwardly differentiable approximation to $R(z,m)$. The intuition behind our forthcoming numerical approximation to $R(m,z)$ comes from the fact that for $m \gg 1$, t is negligible in the denominator compared to m over the range where the e^{-t^2} weighting is large, hence

$$R(m,z) \approx \frac{1}{m^2}\int_0^\infty e^{-t^2}\sin(2zt)\,\mathrm{d}t = \frac{1}{m^2}F(z) \tag{32}$$

where $F(z)$ is the Dawson F function [1]. By standard asymptotic analysis, matching the z derivative at $z = 0$ and the $z \gg 1$ asymptote for the Dawson F function results in the improved estimate:

$$R(m, z) \approx \frac{1}{m}\sqrt{e^{m^2}E_1(m^2)}F\left(mz\sqrt{e^{m^2}E_1(m^2)}\right) \tag{33}$$

which is accurate to within about 5% across the full range of m values. However, this level of accuracy is not appropriate for engineering calculations.

To build more intuition for $R(m, z)$, its close relation to the Dawson F function is evinced by considering the Maclaurin series in z for both:

$$F(z) = z - \frac{2}{3}z^3 + \frac{4}{15}z^5 - \frac{8}{105}z^7 + \cdots \quad , \tag{34}$$

$$R(m, z) = e^{m^2}\left(E_1(m^2)z - \frac{2}{3}E_2(m^2)z^3 + \frac{4}{15}E_3(m^2)z^5 - \frac{8}{105}E_4(m^2)z^7 + \cdots\right) \tag{35}$$

which further highlights the utility of maintaining consistency of $R(m, z)$ with its asymptotic sister $F(z)$. The series are the same, but with the addition of exponential integral multiplying factors in $R(m, z)$.

In order to find such a numerical relation which maintains this consistency in the asymptotic case, Eq. 30 can be transformed via interchange of differentiation and integration. Consider the generalized function:

$$R(m, z, \alpha) = e^{m^2}\int_0^\infty \frac{e^{-\alpha(t^2+m^2)}\sin(2zt)\,\mathrm{d}t}{t^2 + m^2} \tag{36}$$

Differentiating reveals that:

$$\frac{\partial R(m, z, \alpha)}{\partial \alpha} = -e^{m^2}\int_0^\infty e^{-\alpha(t^2+m^2)}\sin(2zt)\,\mathrm{d}t = -\frac{e^{-m^2(\alpha-1)}}{\sqrt{\alpha}}F\left(\frac{z}{\sqrt{\alpha}}\right) \tag{37}$$

The fundamental theorem of calculus then applies:

$$R(m, z, \infty) - R(m, z, 1) = e^{m^2}\int_1^\infty \frac{1}{\sqrt{\alpha}}e^{-m^2\alpha}F\left(\frac{z}{\sqrt{\alpha}}\right)\mathrm{d}\alpha \tag{38}$$

Using the fact that $R(m, z, \infty) = 0$ and doing a change of variables, Eq. 30 becomes:

$$R(m, z) = 2e^{m^2}\int_1^\infty e^{-m^2t^2}F(z/t)\,\mathrm{d}t \tag{39}$$

which confirms our hunch about the close relation of $F(z)$ to $R(m, z)$; it is an infinite superposition of stretched and scaled Dawson F functions.

We have found success in inserting an approximation for $F(z)$ to Eq. 39. Well-known approximations to $F(z)$ based on rational expressions and other elementary functions [26, 25] do not result in numerically useful expressions. However, noting that:

$$\int_1^\infty e^{-m^2t^2}(z/t)^{(2k+1)}\,\mathrm{d}t = \frac{1}{2}z^{1+2k}E_{1+k}(m^2) \tag{40}$$

it can be seen that approximations to $F(z)$ in the form a power series can be computationally efficient in light of the recursion relation for exponential integrals:

$$ne^{m^2}E_{n+1}(m^2) = (1 - m^2e^{m^2}E_n(m^2)) \tag{41}$$

Careful attention must be paid to the floating point properties of this relation [16]. The magnification of computational errors grows arbitrarily large, and we later present a specialized numerical algorithm guaranteeing floating point stability.

In order to thus obtain a simple, efficient approximation to $R(m, z)$, we employ the Chebyshev expansion valid for $z \in [-5, 5]$ presented in [20]. The results of our method could be improved by using a finer piecewise division for the Chebyshev expansion of $F(z)$ as in [10], but we have used the present approach for simplicity of implementation. Thus, if $F(z)$'s truncated Chebyshev expansion is converted to the power series basis:

$$F(z) \approx \sum_{i=0}^{n} c_n(z/5)^{2*i+1} \tag{42}$$

we obtain approximations of the form

$$R(m, z) \approx \sum_{i=0}^{n} \frac{c_n}{2} E_{1+i}(m^2)(z/5)^{2*i+1} \quad . \tag{43}$$

3.2 Numerical Implementation

3.2.1 Stable, Efficient Calculation of an E_n Sequence

The magnification of error in the forward recurrence relation for exponential integrals from [16] is:

$$|\rho_n| = \frac{x^n E_1(x)}{n! E_{n+1}(x)} \tag{44}$$

Notably, the error magnification of the reverse recurrence relation is the reciprocal of this quantity. Moreover, $|\rho_n|$ is a function increasing from 1, reaching a maximum, and monotonically descending below 1 [16]. As a consequence, a critical index n^* exists such that iterating outward from it results in a numerically stable recursion algorithm. In terms of evaluating Eq. 43, this means splitting the polynomial in z into parts above the index n^* and those below. After computing $e^{m^2}E_{n^*}(m^2)$, Horner's method is used in the reverse recurring relation down to the term of order z, and forward recursion is employed to evaluate the polynomial of degree leading from $2n^* + 1$ up to $2n + 1$.

A simple result we have obtained is that the smallest value of n^* such that:

$$\frac{x^{n^*} E_1(x)}{n^*! E_{n^*+1}(x)} < 1 \tag{45}$$

is well approximated by:

$$n^* \approx ex - \frac{1}{2}\log\pi \quad . \tag{46}$$

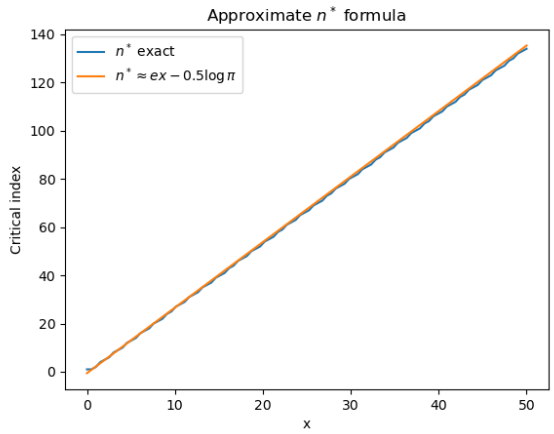


Figure 5: Approximate solution to finding the first value of n such that $\frac{x^n E_1(x)}{n! E_{n+1}(x)} < 1$.

Appendix B shows how this can be obtained. Fig. 5 illustrates the accuracy of Eq. 46. Using this information, algorithm 1 explicitly states the procedure to calculate $R(m, z)$ and its derivative with respect to z. While the use of a power basis polynomial is sub-optimal, numerically speaking, the main source of numerical error in this scenario originates from the recursive exponential integral formula. The specification of the algorithm assumes that an accurate method for computing $E_n(x)e^x$ has been provided, which is well documented in many other works. We have employed a C++ adaptation of the continued fraction approximation employed by the Cephes library [7], which is documented in [1].

3.2.2 The Jump Integral

After computing the value of $I(\Re[z], m)$, the differential equation of Eq. 25 which I follows can be used to calculate $I(z, m)$. We calculate $I(z, m)$ in this manner due to the nearly discontinuous behavior of $I(z, m)$; it has a jump discontinuity about $m = 0$ if $z \in \mathbb{R}$. Because the imaginary part of z is small in windowed multipole libraries, the resulting behavior is nearly discontinuous and hence is not captured efficiently by general approximation techniques; finely resolved tables or high polynomial orders would be required. Our approach thus resolves the discontinuous component exactly with the piecewise function Eq. 28. The nontrivial part of Eq. 26 is the transcendental integral:

$$J(z, x) = e^{-\Re[z]^2} \int_{\Re[z]}^{z} \frac{e^{t^2}\,\mathrm{d}t}{x - t} \tag{47}$$

We have deemed this term the *jump integral* because it allows jumping from values of $I(z, m)$ on the real line to values above the real line in the complex plane. While it seems that our issue of approximating the transcendental integral $w(z, x)$ has seemingly not been heretofore ameliorated due to the appearance of yet another transcendental integral Eq. 47, a change of variables puts it into a form suitable for numerical approximation:

$$J(z, x) = \int_{0}^{i\Im[z]} \frac{e^{u^2 + 2u\Re[z]}\,\mathrm{d}u}{m - u} \tag{48}$$

where again, $m = \Re[z] - x$. Because $\Im[z]$ is small as shown by Fig. 4, the argument to the exponential term is similarly small. Where this integral is well-defined ($m \neq 0$), the exponential term can be expanded in its Maclaurin series and integrated term by term:

$$e^{u^2+2u\Re[z]} = \sum_{n=0}^{\infty} \frac{a_n}{n!} u^n \quad . \tag{49}$$

It is verified that the coefficients a_n satisfy the two-term recurrence:

$$a_{n+1} = 2\Re[z]a_n + 2(n-1)a_{n-1}; \quad a_0 = 1; \quad a_1 = 2\Re[z] \tag{50}$$

Next, the term-by-term integrals appear in the form:

$$\int_0^{i\Im[z]} \frac{u^n \, \mathrm{d}u}{m-u} = m^n B_{i\Im[z]/m}(1+n, 0) \tag{51}$$

Where $B_x(\cdot, \cdot)$ is the incomplete beta function, defined as:

$$B_x(a, b) = \int_0^z t^{a-1}(1-t)^{b-1} \, \mathrm{d}t \quad . \tag{52}$$

A recursion formula derived as a special case of formulas in [1] efficiently calculates these incomplete beta function values of higher n in sequence:

$$B_x(n+1, 0) = B_x(n, 0) - \frac{x^n}{n} \tag{53}$$

In combination with the fact that:

$$B_x(1, 0) = -\log(1-x) \quad , \tag{54}$$

this yields an efficient numerical scheme for evaluating an integral of the truncated Maclaurin series of the exponential of Eq. 48. Algorithm 2 details the combination of all of these facts for an efficient approximation to $J(m, z)$. This approximation works very well for problems with $|\Re[z]| \leq 5$, which easily covers the range of scattering events where resonances appreciably affect the double differential at temperature. Outside of that range, the integral becomes increasingly oscillatory, so an asymptotic approximation is employed for $|z| > 5$. This approximation is documented in Appendix D.

Lastly, Algorithm 3 gives the overall algorithm to compute $w(z, x)$ efficiently. It relies on access to some implementation of calculating $w(z)$, e.g. the permissively licensed [22] which implements a variety of approximations to achieve high accuracy, or one of the various rational approximations [34, 2, 19] when higher error is permitted. Regardless of the chosen $w(z)$ implementation, our algorithm maintains asymptotic consistency such that $\lim_{x\to\infty} w(z, x) = w(z)$. This work leverages a recent approximation tailored for WMP [15].

Input : $m \in \mathbb{R}$, $z \in \mathbb{R}$
Output: $R(m, z)$ from Eq. 30 and $\frac{\partial R}{\partial z}$
Data: c_n coefficients of Eq. 42, $1 \leq n \leq n_{max}$
$n^* \leftarrow \min(\max(em^2 - 0.57, 1), n_{max})$;
expnexp $\leftarrow E_{n^*}(m^2)e^{m^2}$;
expnexp_orig $\leftarrow$ expnexp;
result $\leftarrow 0$;
derivative $\leftarrow 0$;
`// Backward recurse with Horner scheme`
for $n \leftarrow n^* - 1$ **to** 1 **do**
  expnexp $\leftarrow (1 - n\text{expnexp})\,/m^2$;
  derivative $\leftarrow$ derivative $\cdot$ pow$(z/5, 2)+$ result;
  result $\leftarrow$ (result)pow$(z/5, 2) +$ expnexp $\cdot\, c_{n-1}$;
end
derivative $\leftarrow$ 2derivative $\cdot$ pow$(z/5, 2)$;
derivative $\leftarrow$ derivative $+$ result ;
result $\leftarrow$ result $\cdot\, z/5$;
`// Forward recursing polynomial evaluation`
x2 $\leftarrow$ pow$(z/5, 2n^* - 1)$;
x1 $\leftarrow$ pow$(z/5, 2(n^* - 1))$;
expnexp $\leftarrow$ expnexp_orig;
for $n \leftarrow n^*$ **to** n_{max} **do**
  result $\leftarrow c_{n-1}\cdot$expnexp $\cdot$ x2;
  derivative $\leftarrow c_{n-1}\cdot$expnexp $\cdot$ x1 $\cdot(2n - 1)$;
  x2 $\leftarrow x2 \cdot$ pow$(z/5, 2)$;
  x1 $\leftarrow x1 \cdot$ pow$(z/5, 2)$;
  expnexp $\leftarrow (1 - m^2 \cdot$ expnexp$)/n$;
end
return (result, derivative);

Algorithm 1: Stable $R(m, z)$ approximation for $|z| \leq 5$. Appendix C is used for $|z| > 5$.

Input : $x \in \mathbb{R}$, $z \in \mathbb{C}$
Output: $J(m, z)$ from Eq. 47
result $\leftarrow 0$;
a0 $\leftarrow 1.0$;
a1 $\leftarrow 2\Re[z]$;
result $\leftarrow$ b1 $\cdot$ a0;
b1 $\leftarrow b1 - i\Im[z]/(x - \Re[x])$;
`// ` n_{max} `adjusts the number of truncated series terms`
for $n \leftarrow 2$ **to** n_{max} **do**
  tmp $\leftarrow 2\Re[z] \cdot$ a1 $+ 2(n-1) \cdot$ a0;
  a0 $\leftarrow$a1;
  a1 $\leftarrow$tmp;
  result $\leftarrow$ result $+$ b1 $\cdot$ a1$\frac{m^n}{n!}$;
  b1 $\leftarrow$ b1 $- \frac{1}{n+1}\left(\frac{i\Im[z]}{(x-\Re[x])}\right)^n$;
end
return result;

Algorithm 2: Efficient $J(m, z)$ Approximation for $\Im[z] < 1$ and $|\Re[z]| < 5$. Appendix D describes the approximation for $|\Re[z]| > 5$.

Input : $z \in \mathbb{C}$, $x \in \mathbb{R}$
Output: $w(z, x)$ from Eq. 13
m $\leftarrow x - \Re[z]$;
`// ` $R(m, z)$ `integral and its derivative`
rmz, drmzdz $\leftarrow$ call(Alg. 1);
`// ` $I(m, z)$ `integral`
imz $\leftarrow m \cdot$ rmz $+ \frac{1}{2}$drmzdz $+ \frac{i\pi}{2} e^{-\Re[z]^2 + x^2}\left(\text{erf}(x) - \text{sign}(m)\right)$;
`// ` $J(m, z)$ `integral`
ji $\leftarrow$ call(Alg. 2);
imz $\leftarrow$ imz$+$ ji;
imz $\leftarrow$ imz$\cdot e^{\Re[z]^2 - z^2}$;
result $\leftarrow \frac{i}{\pi} e^{-x^2} \cdot$ imz;
result $\leftarrow$ result $+ \frac{1}{2}\left(\text{erf}(x) + 1\right) w(z)$;
return result;

Algorithm 3: Efficient $w(z, x)$ approximation for $\Im[z] << 1$.

3.3 The Pole Sampling Approximation

A key approximation of our technique that enables its computational efficiency is viewing the multipole cross section in the relative speed distribution as a mixture distribution. The theoretical justification is that if poles are present and sufficiently close to the incident neutron energy ($|z| < 20$ specifically), the relative speed PDF is well-approximated by ignoring the polynomial contribution:

$$P(x|y) \approx e^{-x^2} \Re \left[\sum_{j \in W(\beta^{-2} y^2)} \frac{\beta r_j}{z_j - x} \right] \tag{55}$$

This expression is not employed to actually sample the scattering distribution. Rather, it is best viewed as a mixture distribution in which each pole contributes a probability proportional to:

$$\mathbb{P}[\sigma_s(x) = \frac{\beta r_j}{z - x} + \sigma_{0,j} + \sigma_{1,j} x] \propto \Re \left[r_j w(z_j) \right] \tag{56}$$

which defines a discrete distribution. In order to avoid the need for auxiliary storage and the calculation of a normalizing constant to this distribution, we recommend finding the maximum of $-|r_j w(z_j)| / \log(\xi_j)$ where ξ_j are uniform random numbers differing for each pole. The j corresponding to the maximum of this expression follows the desired discrete distribution. We also note that the quantity $\Re \left[r_j w(z_j) \right]$ is exactly the Doppler broadened contribution to the integrated scattering cross section, so this sampling procedure incurs no additional Faddeeva function evaluation overhead if this is done in tandem with a WMP cross section lookup operation.

Finally, we emphasize that the pole sampling approximation is not precisely consistent with the original multipole cross section representation. Instead, it uses the fact that polynomial contributions to the cross section negligibly affect the scattering kernel, while poles do so substantially.

3.4 Finding the values of σ_0 and σ_1

While it may seem that the polynomial contribution to Eq. 5 could come as the first terms from the polynomials defined within the windows, as [6] used, we have found in practice that this choice is inconsistent with the approximation of the pole sampling technique and leads to negative cross-section values.

To remedy this issue, we use the heuristic that the relative error of the cross section's local approximation is minimized by matching polynomial values in the vicinity of the dip of the resonance. The location of the scattering resonance trough is calculated as:

$$\sqrt{E}_{\text{trough}} = \frac{-b + \sqrt{b^2 - abc + a^2 d}}{a} \tag{57}$$

where

$$a = -\Im[\left(r_j^* - \bar{r_j^*}\right)] \tag{58}$$

$$b = -\Im[\left(\bar{r_j^*}p_j^* - r_j^*\bar{p_j^*}\right)] \tag{59}$$

$$c = -\Re[\left(\bar{p_j^*} + p_j^*\right)] \tag{60}$$

$$d = |p_j^*|^2 \tag{61}$$

$$\tag{62}$$

At this point, the window index of $\sqrt{E}_{\text{trough}}$ is calculated [1] . This may be a different window from the incident neutron energy's window. The windowed multipole cross section of Eq. 4 is then evaluated at $\sqrt{E}_{\text{trough}}$ but *excluding* the sampled pole p_j, i.e.

$$\sigma_0 = \frac{1}{E_{\text{trough}}}\Re\left[\sum_{j^* \neq j \in W(E_{\text{trough}})} \frac{r_j}{p_j - \sqrt{E_{\text{trough}}}}\right] + \sum_{n=0}^{N} a_n E_{\text{trough}}^{n/2} \tag{63}$$

From there, $\sigma_1 \approx \frac{\partial \sigma_s(E)}{\partial \sqrt{E}}$ is calculated at the same point, this time *only* including contributions from the polynomial expansion but not from any poles. This linearization technique has been found to improve the accuracy of our method when considering nuclides with tightly spaced resonances such as ^{235}U. For complete clarity, the resulting expression is:

$$\sigma_1 = \sum_{n=0}^{N} a_n \frac{n-2}{2} E_{\text{trough}}^{n/2-2} \tag{64}$$

Finally, a linearization of the cross section in $\sqrt{E}$ space has been obtained. For use with the root finder, the nondimensional variable $\beta(\sqrt{E} - \sqrt{E_{\text{incident}}})$ is preferable, so σ_0 is appropriately shifted and σ_1 appropriately scaled.

3.5 Inverting the relative speed CDF

To sample from the relative speed PDF, we employ the CDF inversion technique. A naive attempt at this would be a few bisection root finding steps followed by a handful of Newton-like iterations. In practice, we've found that five bisection iterations followed by three Halley-Newton iterations resolves the root to within acceptable tolerance; however, a far more efficient root finder has been developed which takes a maximum of four iterations total, only requiring more work for unusual edge cases.

The *bootstrapping* step, as we call it, is essential to an efficient implementation of MARS. The bootstrapping step cheaply obtains an initial guess to the solution of the CDF inversion problem, from which a small number of Newton-like iterations improve the solution.

[1] The term under the square root in Eq. 57 can sometimes be negative in the vicinity of nonphysical poles which are artifacts of the fitting process, and dealing with imaginary quantities in this case is undesirable. Therefore, our calculation uses a linearization of the non-pole cross section at the incident energy instead in that case.

The key to doing so lies in finding a cheap approximation to the inverse of the CDF with general pole parameters. In order to do so, we first move from the root finding space of $x \in (-\infty, \infty)$ to the nonlinearly mapped variable $\tilde{x} = \frac{1}{2}(1 + \mathrm{erf}x)$. The intuition behind using this modified space is that as the resonances become weak and the incident neutron energy becomes high, it can be shown that the CDF is simply equal to $\tilde{x}$ which ranges between zero and one. Resonances and low energy free gas effects simply act as perturbations to this linear function, which enables a good starting point for approximating the root location.

The next step in improving the CDF model in the mapped space is to observe that the contribution of collision probability from the resonance largely does not depend on its imaginary part. Increasing the imaginary part of the resonance broadens it and decreases its width. Therefore, the magnitude of the jump in $w(z, x)$ when x is near $\Re[z]$ quantifies the probability that the neutron experiences a collision near the peak of the resonance. The jump in $w(z, x)$ for small $\Im[z]$ is:

$$\lim_{\Im[z]\to 0} \int_{\Re[z]-\epsilon}^{\Re[z]+\epsilon} \frac{e^{-t^2}\,\mathrm{d}t}{z-t} = e^{-\Re[z]^2} \tag{65}$$

and therefore the probability contribution due to the resonance is approximately:

$$p_{\mathrm{jump}} = \mathbb{P}[x \approx Re[z]] \approx \beta r_{j^*} \pi e^{-\Re[z]^2}/C \tag{66}$$

where C is the normalizing constant given by Eq. 12. Because this is an approximation, the probability of Eq. 66 may not be bounded between zero and one, so we threshold it to that range. In practice, the estimate provided here is accurate. We have found that this probability tends to be added into the CDF about $\Re[z]$ over the interval $[\Re[z] - \frac{3}{2}\Im[z], \Re[z] + \frac{3}{2}\Im[z]]$. This estimate could obviously be tuned for greater accuracy.

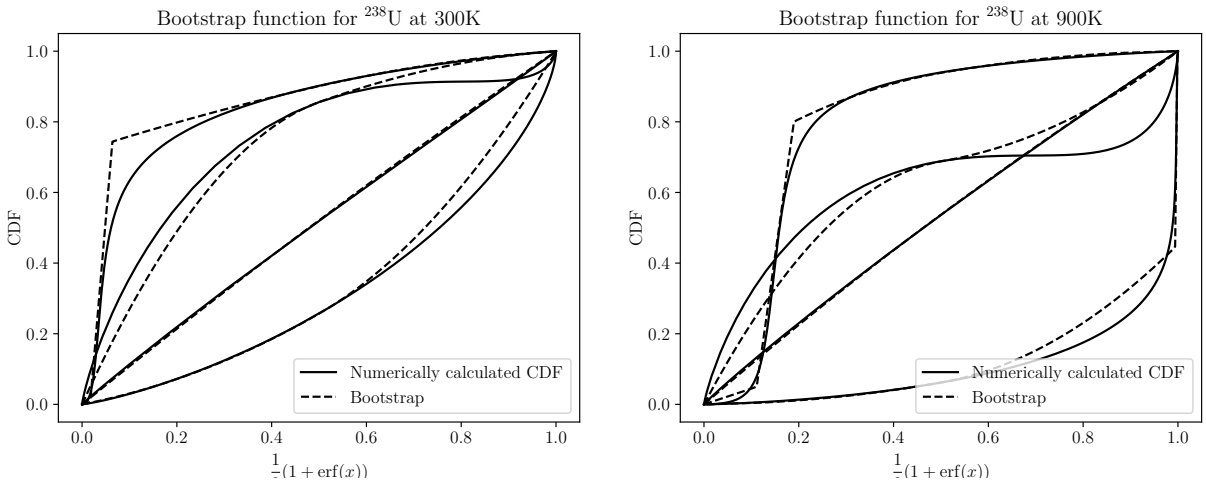
One final tool we employ to bootstrap the root finding process pertains to the values of the CDF about $x = 0$. In this case, numerous instances of functions occuring in its expression such as $\mathrm{erf}(x)$ and e^{-x^2} take on easily calculated values. On top of that, the incomplete Faddeeva function has a closed form expression when $x = 0$:

$$w(z, 0) = \frac{1}{2}w(z) + \frac{i}{2\pi}e^{-z^2}E_1(-z^2) \quad . \tag{67}$$

Because e^{-z^2} is already computed and cached for the CDF inversion, the calculation of a complex exponential integral is the only difficulty. This is much easier and computationally cheaper to do than the more involved $w(z, x)$ evaluation, so any off-the-shelf approximation of $E_1(-z^2)$ can be employed here.

Addtionally, the derivatives of the CDF with respect to x about $x = 0$ are also easily obtainable, which we use to further improve our rootfinding guess. So far, we have only incorporated information from the first derivative which has proven sufficient.

This leaves us with the following pieces of information from which the root estimate is extracted: the probability due to the resonance, its width, the value and slope of the CDF about $x = 0$ i.e. $\tilde{x} = 1/2$, and the known endpoint values of the CDF at 0 and 1. We therefore construct a function which is piecewise quadratic on the left and right of $\tilde{x} = 1/2$. This quadratic interval ranges to either the endpoints $\tilde{x} = 0$ or $\tilde{x} = 1$, or the resonance's



(a) $T = 300$ kelvin (b) $T = 900$ kelvin

Figure 6: The bootstrapping CDF provides a fairly accurate, easily invertible approximation to the true relative speed CDF to kickstart the root finding process. The pairs of lines, moving from top to bottom, represent 35.25, 36.25, 38.25, and 66.25 eV incident neutron energies.

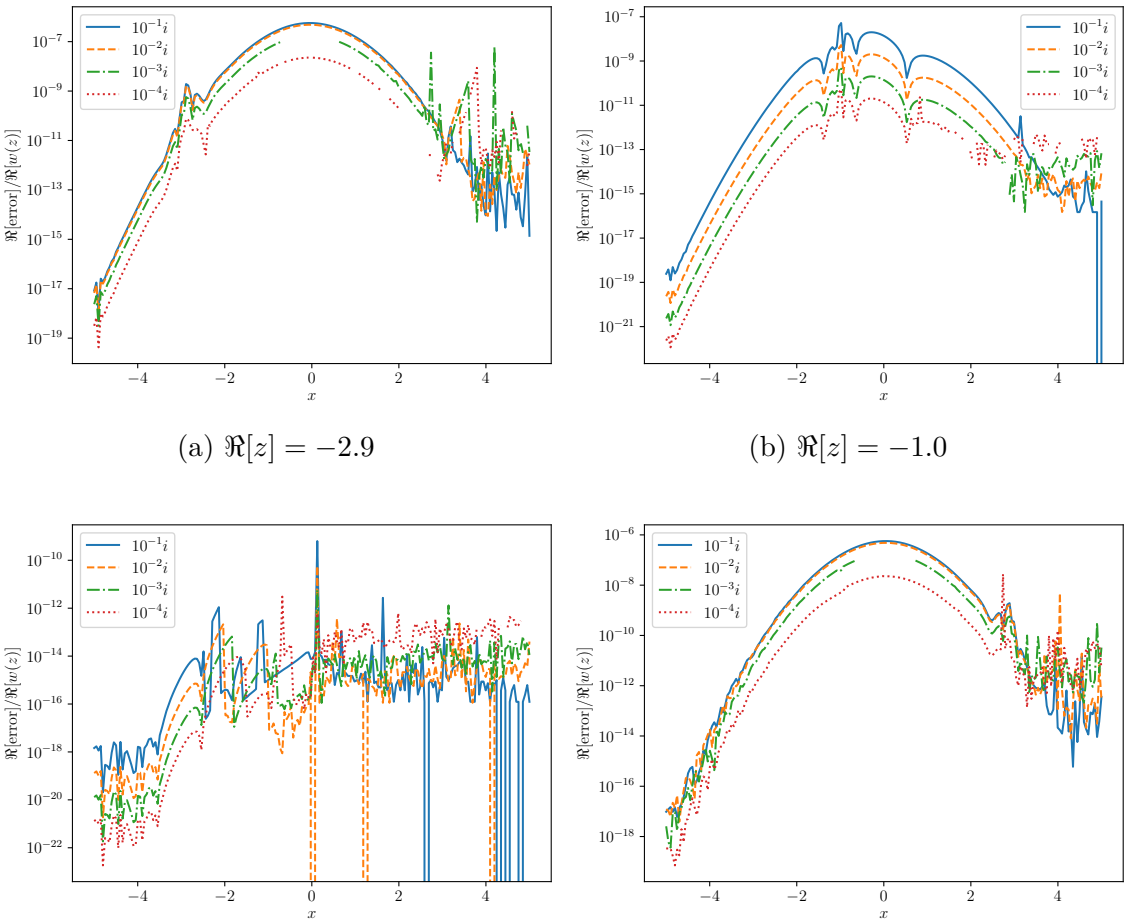
upper or lower range of probability gain, estimated here as x$\in [\Re[z] - \frac{3}{2}\Im[z], \Re[z] + \frac{3}{2}\Im[z]]$. Note that this interval has to be mapped to an interval in $\tilde{x}$ space. Because the interval of the resonance is small, the Jacobian of the transformation $x \mapsto \tilde{x}$ which is proportional to $e^{-\Re[z]^2}$ (a quantity already computed) can be used to calculate the range in $\tilde{x}$ space.

With this knowledge, the CDF can be approximated somewhat accurately in $\tilde{x}$ space. Despite the apparent complexity of what was just described, the inversion of the previous paragraph's function can be done using simple branching logic and, at worst, the solution of a quadratic equation. Because translating the inverse of the above function into code can take nontrivial effort, C++ code to achieve this has been provisioned in Appendix E. Figure 6 shows two examples of how this can be a quite satisfactory approximation of the CDF in $\tilde{x}$ space when resonances are influencing the scattering distribution.

4 Results

4.1 Calculation of $w(z, x)$

In order to test the accuracy of Alg. 3, we haved computed reference values of $w(z, x)$ using `scipy`'s [37] adaptive quadrature routine, `scipy.integrate.quad`, to evaluate the integral formulation Eq. 13. In approximation of the jump integral Eq. 47, only the first five terms in the series are retained. Where functions such as $\log(x)$ or e^x appear, C++ standard library implementations have been employed. The implementation of $w(z)$ from [22] has been employed. This results in the error profiles exhibited by Fig. 7, where we have plotted the real part of $\left(w_{\text{approx}}(z, x) - w(z, x)\right)/w(z)$. Because only the real part is of interest in resonance upscatter calculations, results on the imaginary component's error are omitted.



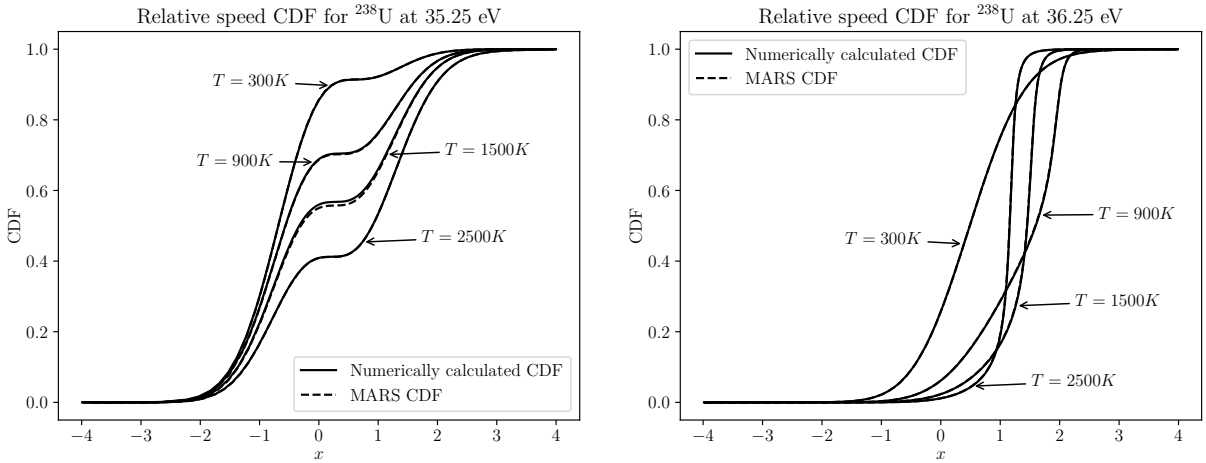
(a) $\Re[z] = -2.9$

(b) $\Re[z] = -1.0$

(c) $\Re[z] = 0.0$

(d) $\Re[z] = 2.9$

Figure 7: Error of $\Re[w(z,x)]$ for a few values of z. The legend is the imaginary number added to the real part specified in each figure's caption. The plotted error, $w_{\text{approx}}(z,x) - w(z,x)$ is normalized by $\Re[w(z)]$ to match the scaling of Fig. 2.



(a) 35.25 eV

(b) 36.25 eV

Figure 8: The MARS analytic CDF matches numerically integrated relative speed cumulative distributions. Some error can be observed for the 1500K case at 35.25 eV; scattering in the resonance dip is fortunately an extremely rare event.

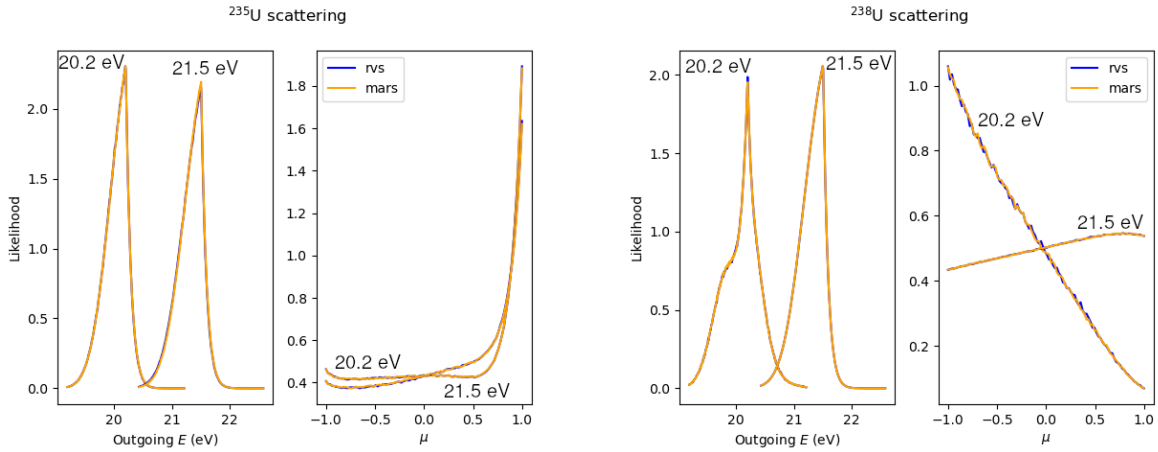
4.2 Single Energy Testing

We first present in Fig. 8 the relative speed distribution of ^{238}U for two different energies and a few temperatures as calculated both by numerical integration and the MARS analytic CDF. The energies correspond to being in the trough and near the peak of a scattering resonance. These plots clearly show the influence of the resonances on the double differential cross section; a nuclide with constant cross section has a relative speed distribution which is very nearly an error function at epithermal energies. The relative speed distribution near resonances has a jumping effect which is governed by $w(z, x)$. They bear a resemblance to $w(z, x)$ behavior depicted by Fig. 2.

If the relative speed distribution is correct, the resultant double-differential scattering distribution is also correct. Fig. 9 shows this is the case for our method when compared to the RVS method of [31]. These results were obtained from our modified version of OpenMC, available at `github.com/gridley/openmc/tree/mars`. It also shows that the pole sampling technique successfully works for ^{235}U and its tightly spaced resonances.

4.3 Pin Cell Reactivity Feedback

The 2.4% enriched PWR pin cell example from OpenMC's suite of example problems was used to calculate Doppler reactivity feedback effects with four different models. The first and second used pointwise cross sections that were interpolated between 300, 600, 900, 1200, and 2500 kelvin. The model was run at temperatures ranging from 300 to 1800 kelvin in increments of 20 kelvin. Of the two using pointwise cross sections, one used the historical constant cross section free gas scattering approximation, and the second used the RVS method. The second two cases both used windowed multipole cross sections, one using RVS and the second MARS. The ENDFB-VII.1 nuclear dataset was employed. Figure 10



(a) ^{235}U (b) ^{238}U

Figure 9: Scattering at 1200K matches results from RVS method well at two different energies. These energies interact with resonances for both nuclides.

shows how these cases compare. Two hundred cycles with ten inactive were employed, using 200,000 particles per cycle. k_{eff} was thus converged to 20 pcm for each case.

It can be seen that the pointwise cross section representation incurs some interpolation error between 1200 and 1800 kelvin. The MARS method matches the RVS results where multipole cross sections were employed. We can thus conclude that the new method works correctly across the range of energies where resonances influence the double-differential cross section at temperature for nuclides of both strong, distantly spaced resonances (^{238}U) and closely spaced weak resonances (^{235}U).

4.4 Influence on Tracking Rate

Finally, in order to determine the computational efficiency of the new method, tracking rate comparisons were carried out on the same PWR pin cell example problem. The computational performance of both inactive and active cycles was assessed. For the active cycles, a 100x100 Cartesian mesh tallied flux, fission rates, and neutron production rates using track-length estimators. In addition, a spatially homogenized energy spectrum tally consisting of 500 equal lethargy bins was applied.

An Intel Xeon W-2133 with six physical cores carried out the calculations, and obtained the results depicted in Table 1. This clearly demonstrates the computational efficiency of MARS compared to the RVS and DBRC methods. While it does not outperform RVS in this scenario, future work will explore its performance on vector computer architectures where we expect it to outperform.

The computational expense incurred by tallying tends to render the performance impact of our new method particularly negligible. Collision estimators could be used on the mesh tally to improve the tracking rate, but we arbitrarily opted for track length estimators. Due to subtle hardware-related effects such as cache utilization or branch prediction, the tracking rates of the three resonance upscatter handling methods have different relative performances

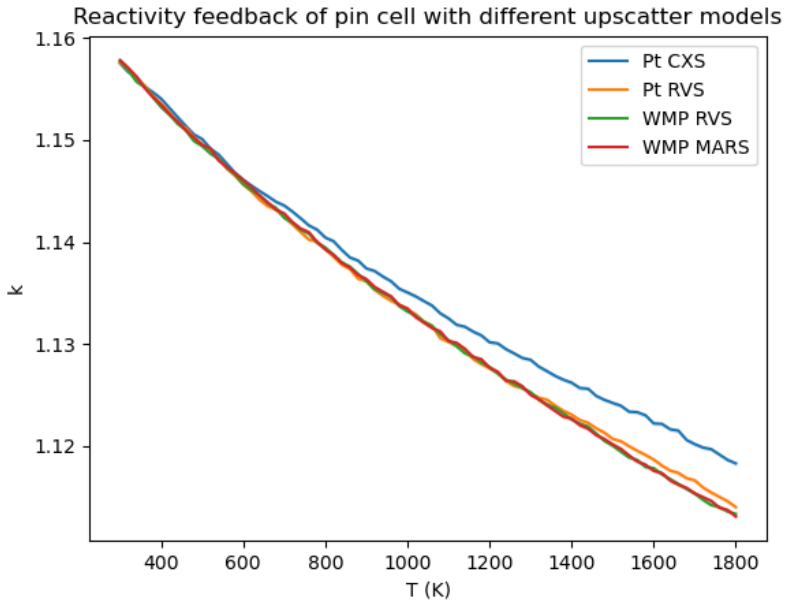


Figure 10: MARS matches the k eigenvalue of the RVS method on a 2.4% enriched fresh PWR pin cell problem. Line width represents estimated standard deviation of the mean.

Method	Inactive	Active
CXS	60.5	11.4
DBRC	57.0	11.1
RVS	58.3	11.3
MARS	60.3	11.1

Table 1: Tracking rate in thousand particles per second obtained by the constant cross section treatment, and three resonance upscatter models. MARS is comparable in speed to widely accepted techniques.

when comparing active and inactive cycles. Future work will explore detailed performance results on a variety of architectures.

5 Conclusion

The multipole formalism carries a variety of advantages compared to pointwise cross sections. Aside from its potential gains in computational efficiency on modern compute architectures, it enables accurate Doppler broadening without a library size tradeoff [23], elegant sensitivity quantification, and narrows the gap between R matrix theory and the cross section representation [13]. This work develops yet another advantage to the windowed multipole formalism: closed-form resonance upscatter treatment.

We have demonstrated that the new method matches the results obtained by other resonance upscatter techniques. To achieve this, we derived an expression for the target relative speed distribution, and identified a novel special function which universally arises in this application. Novel numerical techniques that balance efficiency and accuracy were derived, implemented, and tested. The overall scheme was shown to achieve the same tracking rate as other resonance upscatter modeling methods.

The new method called multipole analytic resonance scattering (MARS) overcomes the storage requirements of relative speed tabulation [9], and avoids rejection sampling as employed by other common approaches. Without a need to access intermediate storage, the accesses to global memory can be reduced on GPU architectures. On top of that, the work discrepancy between threads incurred by rejection sampling on GPUs is similarly overcome. Future work will explore the implementation and optimization of this method on GPUs.

6 Acknowledgements

This work was partially supported by the U.S. Department of Energy through the Los Alamos National Laboratory. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001). This material is also based upon work partially supported under an Integrated University Program Graduate Fellowship. This research was also partially supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Department of Energy Office of Nuclear Energy.

References

[1] Milton Abramowitz and Irene A. Stegun, eds. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. 0009-Revised edition. New York, NY: Dover Publications, June 1, 1965. 1046 pp. ISBN: 978-0-486-61272-0.

[2] S. M. Abrarov and B. M. Quine. "Efficient Algorithmic Implementation of the Voigt/-Complex Error Function Based on Exponential Series Approximation". In: *Applied Mathematics and Computation* 218.5 (Nov. 1, 2011), pp. 1894–1902. ISSN: 0096-3003. DOI: 10.1016/j.amc.2011.06.072. URL: https://www.sciencedirect.com/science/article/pii/S0096300311009179 (visited on 01/10/2023).

[3] René Aïd, Luciano Campi, and Nicolas Langrené. *A Structural Risk-Neutral Model for Pricing and Hedging Power Derivatives*. SSRN Scholarly Paper. Rochester, NY, July 1, 2013. DOI: 10.1111/j.1467-9965.2011.00507.x. URL: https://papers.ssrn.com/abstract=2276600 (visited on 01/01/2023).

[4] B. Becker, R. Dagan, and G. Lohnert. "Proof and Implementation of the Stochastic Formula for Ideal Gas, Energy Dependent Scattering Kernel". In: *Annals of Nuclear Energy* 36.4 (May 1, 2009), pp. 470–474. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2008.12.001. URL: https://www.sciencedirect.com/science/article/pii/S0306454908003186 (visited on 10/29/2022).

[5] Carl M. Bender and Steven A. Orszag. *Advanced Mathematical Methods for Scientists and Engineers I: Asymptotic Methods and Perturbation Theory*. Softcover reprint of hardcover 1st ed. 1999 edition. New York Heidelberg: Springer, Dec. 1, 2010. 607 pp. ISBN: 978-1-4419-3187-0.

[6] Elliott Biondo et al. "Algorithm for Free Gas Elastic Scattering without Rejection Sampling". In: The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering. Raleigh, NC, Oct. 1, 2021. DOI: 10.13182/M&C21-33659.

[7] *Cephes*. URL: https://netlib.org/cephes/ (visited on 01/03/2023).

[8] Namjae Choi and Han Gyu Joo. "Relative Speed Tabulation Method for Efficient Treatment of Resonance Scattering in GPU-Based Monte Carlo Neutron Transport Calculation". In: *Nuclear Science and Engineering* 195.9 (Sept. 2, 2021), pp. 954–964. ISSN: 0029-5639. DOI: 10.1080/00295639.2021.1887701. URL: https://doi.org/10.1080/00295639.2021.1887701 (visited on 09/12/2021).

[9] Namjae Choi, Kyung Min Kim, and Han Gyu Joo. "Optimization of Neutron Tracking Algorithms for GPU-based Continuous Energy Monte Carlo Calculation". In: *Annals of Nuclear Energy* 162 (Nov. 1, 2021), p. 108508. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2021.108508. URL: https://www.sciencedirect.com/science/article/pii/S0306454921003844 (visited on 09/01/2021).

[10] W. J. Cody, Kathleen A. Paciorek, and Henry C. Thacher. "Chebyshev Approximations for Dawson's Integral". In: *Mathematics of Computation* 24.109 (1970), pp. 171–178. ISSN: 0025-5718. DOI: 10.2307/2004886. JSTOR: 2004886.

[11] R. Dagan. "On the Use of S(,) Tables for Nuclides with Well Pronounced Resonances". In: *Annals of Nuclear Energy* 32.4 (Mar. 1, 2005), pp. 367–377. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2004.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0306454904002191 (visited on 12/27/2021).

[12] Alfredo Deaño and Nico M. Temme. "Analytical and Numerical Aspects of a Generalization of the Complementary Error Function". In: *Applied Mathematics and Computation* 216.12 (Aug. 15, 2010), pp. 3680–3693. ISSN: 0096-3003. DOI: 10.1016/j.amc.2010.05.025. URL: https://www.sciencedirect.com/science/article/pii/S0096300310005692 (visited on 01/01/2023).

[13] Pablo Ducru et al. "Windowed Multipole Representation of R-Matrix Cross Sections". In: *Physical Review C* 103.6 (June 14, 2021), p. 064610. DOI: 10.1103/PhysRevC.103.064610. URL: https://link.aps.org/doi/10.1103/PhysRevC.103.064610 (visited on 12/23/2021).

[14] A. ; Magnus F. G.; Harry Bateman Erdelyi. *Tables of Integral Transforms Volume 1*. 1st Edition. McGraw-Hill Book Company, Jan. 1, 1954. 411 pp.

[15] Benoit Forget, Jiankai Yu, and Gavin Ridley. "Performance Improvements of the Windowed Multipole Formalism Using a Rational Fraction Approximation of the Faddeeva Function". In: International Conference on Physics of Reactors 2022. Pittsburg, PA, May 17, 2022.

[16] Walter Gautschi. "Recursive Computation of Certain Integrals". In: *Journal of the ACM* 8.1 (Jan. 1961), pp. 21–40. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/321052.321054. URL: https://dl.acm.org/doi/10.1145/321052.321054 (visited on 12/23/2022).

[17] Ely M. Gelbard. *Epithermal Scattering in VIM*. FRA-TM-123. Argonne, IL, United States: Argonne National Laboratory, Dec. 5, 1979.

[18] S.W.D. Hart et al. "Implementation of the Doppler Broadening Rejection Correction in Keno". In: *Transactions of the American Nuclear Society* 108 (Jan. 1, 2013), pp. 423–425.

[19] J. Humlíček. "An Efficient Method for Evaluation of the Complex Probability Function: The Voigt Function and Its Derivatives". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 21.4 (Apr. 1, 1979), pp. 309–313. ISSN: 0022-4073. DOI: 10.1016/0022-4073(79)90062-1. URL: https://www.sciencedirect.com/science/article/pii/0022407379900621 (visited on 01/02/2023).

[20] David G. Hummer. "Exapansions of Dawson's Function in a Series of Chebyshev Polynomials". In: *Mathematics of Computation* 18.86 (1964), pp. 317–319. ISSN: 0025-5718. DOI: 10.2307/2003311. JSTOR: 2003311.

[21] R. N. Hwang. "A Rigorous Pole Representation of Multilevel Cross Sections and Its Practical Applications". In: *Nuclear Science and Engineering* 96.3 (July 1, 1987), pp. 192–209. ISSN: 0029-5639. DOI: 10.13182/NSE87-A16381. URL: https://doi.org/10.13182/NSE87-A16381 (visited on 02/27/2023).

[22] Steven Johnson. *Faddeeva Package*. URL: http://ab-initio.mit.edu/wiki/index.php/Faddeeva_Package (visited on 01/10/2023).

[23] C. Josey et al. "Windowed Multipole for Cross Section Doppler Broadening". In: *Journal of Computational Physics* 307 (Feb. 15, 2016), pp. 715–727. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2015.08.013. URL: http://www.sciencedirect.com/science/article/pii/S002199911500532X (visited on 01/14/2021).

[24] Deokjung Lee, Kord Smith, and Joel Rhodes. "The Impact of 238U Resonance Elastic Scattering Approximations on Thermal Reactor Doppler Reactivity". In: *Annals of Nuclear Energy*. PHYSOR 2008 36.3 (Apr. 1, 2009), pp. 274–280. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2008.11.026. URL: https://www.sciencedirect.com/science/article/pii/S0306454908003083 (visited on 12/23/2021).

[25] F. G. Lether and P. R. Wenston. "Elementary Approximations for Dawson's Integral". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 46.4 (Oct. 1, 1991), pp. 343–345. ISSN: 0022-4073. DOI: 10.1016/0022-4073(91)90099-C. URL: https://www.sciencedirect.com/science/article/pii/002240739190099C (visited on 11/09/2022).

[26] Frank G. Lether. "Constrained Near-Minimax Rational Approximations to Dawson's Integral". In: *Applied Mathematics and Computation* 88.2 (Dec. 30, 1997), pp. 267–274. ISSN: 0096-3003. DOI: 10.1016/S0096-3003(96)00330-X. URL: https://www.sciencedirect.com/science/article/pii/S009630039600330X (visited on 11/08/2022).

[27] Jingang Liang, Pablo Ducru, and Benoit Forget. "Target Velocity Sampling for Resonance Elastic Scattering Using Windowed Multipole Cross Section Data". In: Nov. 11, 2018.

[28] Shichang Liu et al. "Generation of the Windowed Multipole Resonance Data Using Vector Fitting Technique". In: *Annals of Nuclear Energy* 112 (Feb. 1, 2018), pp. 30–41. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2017.09.042. URL: https://www.sciencedirect.com/science/article/pii/S0306454917303195 (visited on 02/27/2023).

[29] Takamasa MORI and Yasunobu NAGAYA. "Comparison of Resonance Elastic Scattering Models Newly Implemented in MVP Continuous-Energy Monte Carlo Code". In: *Journal of Nuclear Science and Technology* 46.8 (Aug. 1, 2009), pp. 793–798. ISSN: 0022-3131. DOI: 10.1080/18811248.2007.9711587. URL: https://doi.org/10.1080/18811248.2007.9711587 (visited on 10/28/2022).

[30] Mohamed Ouisloumen and Richard Sanchez. "A Model for Neutron Scattering Off Heavy Isotopes That Accounts for Thermal Agitation Effects". In: *Nuclear Science and Engineering* 107.3 (Mar. 1, 1991), pp. 189–200. ISSN: 0029-5639. DOI: 10.13182/NSE89-186. URL: https://doi.org/10.13182/NSE89-186 (visited on 10/28/2022).

[31] Paul Romano and John Walsh. "An Improved Target Velocity Sampling Algorithm for Free Gas Elastic Scattering". In: *Annals of Nuclear Energy* 114 (Apr. 2018), pp. 318–324.

[32] W. Rothenstein and R. Dagan. "Two-Body Kinetics Treatment for Neutron Scattering from a Heavy Maxwellian Gas". In: *Annals of Nuclear Energy* 22.11 (Nov. 1, 1995), pp. 723–730. ISSN: 0306-4549. DOI: 10.1016/0306-4549(95)00002-A. URL: https://www.sciencedirect.com/science/article/pii/030645499500002A (visited on 12/23/2021).

[33] Kelly Rowland et al. "Delta-Tracking in the GPU-accelerated WARP Monte Carlo Neutron Transport Code". In: International Conference on Mathematics&Computational Methods Applied to Nuclear Science&Engineering. Jeju, South Korea, Apr. 2017. URL: https://www.kns.org/files/int_paper/paper/MC2017_2017_2/P100S02-02RowlandK.pdf (visited on 04/27/2021).

[34] Franz Schreier. "The Voigt and Complex Error Function: Humlíček's Rational Approximation Generalized". In: *Monthly Notices of the Royal Astronomical Society* 479.3 (Sept. 21, 2018), pp. 3068–3075. ISSN: 0035-8711. DOI: 10.1093/mnras/sty1680. URL: https://doi.org/10.1093/mnras/sty1680 (visited on 01/10/2023).

[35] Timothy H. Trumbull and Thomas E. Fieno. "Effects of Applying the Doppler Broadened Rejection Correction Method for LEU and MOX Pin Cell Depletion Calculations". In: *Annals of Nuclear Energy* 62 (Dec. 1, 2013), pp. 184–194. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2013.06.013. URL: https://www.sciencedirect.com/science/article/pii/S0306454913003150 (visited on 10/29/2022).

[36] Tuomas Viitanen and Jaakko Leppänen. "Explicit Treatment of Thermal Motion in Continuous-Energy Monte Carlo Tracking Routines". In: *Nuclear Science and Engineering* 171.2 (June 1, 2012), pp. 165–173. ISSN: 0029-5639. DOI: 10.13182/NSE11-36. URL: https://doi.org/10.13182/NSE11-36 (visited on 07/15/2022).

[37] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17.3 (3 Mar. 2020), pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. URL: https://www.nature.com/articles/s41592-019-0686-2 (visited on 01/16/2023).

[38] Jonathan A. Walsh, Benoit Forget, and Kord S. Smith. "Accelerated Sampling of the Free Gas Resonance Elastic Scattering Kernel". In: *Annals of Nuclear Energy* 69 (July 1, 2014), pp. 116–124. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2014.01.017. URL: https://www.sciencedirect.com/science/article/pii/S0306454914000255 (visited on 10/31/2022).

[39] Andrea Zoia et al. "Doppler Broadening of Neutron Elastic Scattering Kernel in Tripoli-4®". In: *Annals of Nuclear Energy* 54 (Apr. 1, 2013), pp. 218–226. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2012.11.023. URL: https://www.sciencedirect.com/science/article/pii/S0306454912004689 (visited on 12/23/2021).

A Derivation of Eq. 19

The forthcoming discussion has not been made mathematically rigorous for sake of brevity and the context of a nuclear engineering journal. We begin by defining the auxiliary complex

function F(z):

$$F(z) = e^{z^2} w(z,x) \tag{68}$$

The complex line integral theorem can then be applied when $\Im[z] > 0$:

$$F(z) - F(0) = \int_0^z \frac{\mathrm{d}F}{\mathrm{d}z}|_{z=z'} \, \mathrm{d}z' \tag{69}$$

Computing $\frac{\mathrm{d}F}{\mathrm{d}z}$ and inserting then reveals:

$$e^{z^2} w(z,x) - w(0,x) = \int_0^z \left(2z' e^{z'^2} w(z',x) - e^{z'^2} \frac{i}{\pi} \int_{-\infty}^{x} \frac{e^{-t^2} \, \mathrm{d}t}{(z'-t)^2} \right) \mathrm{d}z' \tag{70}$$

where the linearity of integration has been employed, and the interchange of differentiation and integration has also been used. The innermost integrals can now be computed exactly, carrying the z' through to the integral defining the incomplete Faddeeva function. This results in:

$$e^{z^2} w(z,x) - w(0,x) = \frac{i}{\pi} \left(\int_0^z \frac{e^{z'^2} e^{-x^2}}{x - z'} \, \mathrm{d}z' - \sqrt{\pi}(1 + \mathrm{erf}(x)) \int_0^z e^{z'^2} \, \mathrm{d}z' \right) \tag{71}$$

Recalling that the Faddeeva function can be defined as

$$w(z) = e^{-z^2} \left(1 + \frac{2i}{\sqrt{\pi}} \int_0^z e^{t^2} \, \mathrm{d}t \right) \quad , \tag{72}$$

we can identify $w(z)$ as the trailing term of Eq. 71. The term $w(0,x)$ must be interpreted in a principal value sense, which results in a contribution in the form of a Heaviside function. The following expression then results:

$$w(z,x) = e^{-z^2} \left(-\frac{1}{2}(\mathrm{Ei}(-x^2) + \frac{i}{\pi} e^{-x^2} \int_0^z \frac{e^{z'^2}}{x - z'} \, \mathrm{d}z' \right) + \frac{1}{2}(1 + \mathrm{erf}(x))(w(z) - e^{-z^2}) + h(x) e^{-z^2} \quad . \tag{73}$$

This result could perhaps be used for numerical calculations of $w(z,x)$. However, it suffers the shortcoming that the exponential integral term goes to infinity for $x = 0$, which is cancelled out by the integral term. However, $w(z,x)$ is well-defined at $x = 0$, and the addition of branching logic to numerical routines to handle this case would be cumbersome. The expression can be made more amenable to numerical approximation with some further simplification.

The integral term goes from 0 to z, and the integrand encloses no poles of the following path whenever $x \neq 0$. As such, a change of integration path is employed: the contour of Fig. 11 results in a convenient cancellation of terms. The top leg of the contour is zero from the e^{t^2} term, resulting in:

$$\int_0^z \frac{e^{t^2}}{x - t} \, \mathrm{d}t = \int_0^{i\infty} \frac{e^{t^2}}{x - t} \, \mathrm{d}t - \int_z^{i\infty} \frac{e^{t^2}}{x - t} \, \mathrm{d}t \tag{74}$$

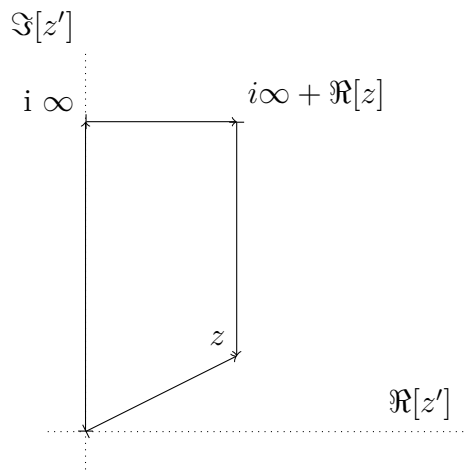


Figure 11: Modified contour used to cancel exponential integral in Eq. 73. The contribution from the top line is zero.

A little bit of algebra shows that:

$$\int_0^{i\infty} \frac{e^{t^2}}{x-t}\,\mathrm{d}t = \frac{1}{2}e^{x^2}\left(-i\pi(\mathrm{erf}(x) - \mathrm{sign}(x)) + \mathrm{Ei}(-x^2)\right) \tag{75}$$

The sign function term ends up cancelling out the Heaviside term upon substitution of Eq. 74 back to Eq. 73. The second term in Eq. 74 easily can be transformed via the change of variables $t' = -it$ to the integral in Eq. 19, thus yielding Eq. 19.

B Derivation of Eq. 46

The goal is to find the smallest n such that

$$\frac{x^n E_1(x)}{n! E_{n+1}(x)} < 1 \tag{76}$$

The variation of the left hand side function of Eq. 76 as n increases, for a constant value of x, has been shown to be first increasing from one, reaching a maximum, and monotonically decreasing from that point, never becoming negative [16]. Firstly, the exponential integrals are replaced with the equivalent upper incomplete gamma function:

$$E_n(x) = x^{n-1}\Gamma(1-n, x) \tag{77}$$

so the equation becomes:

$$\Gamma(0, x) = n!\Gamma(-n, x) \tag{78}$$

Using the asymptotic formula for the upper incomplete gamma function $\Gamma(s, x) \to x^{s-1}e^{-x}$ gives this approximation to solve:

$$x^n \approx n! \tag{79}$$

Which can be solved approximately by first inserting Stirling's formula:

$$x^n \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{80}$$

Taking the nth root results in

$$ex \approx (2\pi n)^{\frac{1}{2n}} n \tag{81}$$

the second term on the right can be approximated by expanding the exponential, for large n:

$$ex \approx (1 + \frac{1}{2n}\log(2\pi n))n \tag{82}$$

This can be solved exactly in terms of the Lambert W function:

$$n \approx \frac{1}{2} W\left(\frac{e^{2ex}}{\pi}\right) \tag{83}$$

The asymptotic property of the Lambert W function that $W(z) \approx \log z - \log\log z$ is then used to obtain Eq. 46.

C Asymptotic Approximation for $R(m, z)$ of Eq. 30

An efficient approximation can be found by grouping the integrand as:

$$R(m,z) = \int_0^\infty f^{(0)}(t,m)\sin(2zt)\,\mathrm{d}t = -\frac{1}{2z}\int_0^\infty f^{(0)}(t,m)\frac{\partial}{\partial t}\left[\cos(2zt)\right]\mathrm{d}t \tag{84}$$

where

$$f^{(0)}(t,m) = \frac{e^{-t^2}}{t^2+m^2} \quad .$$

Integrating the rightmost expression expression in Eq. 84 by parts repeatedly results in a divergent series approximation of the form:

$$R(m,z) = \frac{f^{(0)}(0,m)}{2z} + \frac{f^{(2)}(0,m)}{8z^3} + \frac{f^{(4)}(0,m)}{32z^5} + \frac{f^{(6)}(0,m)}{128z^7} + \cdots \tag{85}$$

where $f^{(n)}(t,m)$ denotes the nth derivative of $f^{(0)}(m,t)$ with respect to t. Some of the subsequent values evaluated about $t = 0$ are:

$$f^{(2)}(0,m) = -\frac{2(1+m^2)}{m^4} \tag{86}$$

$$f^{(4)}(0,m) = \frac{12(2+2m^2+m^4)}{m^6} \tag{87}$$

$$f^{(6)}(0,m) = -\frac{120(6+6m^2+3m^4+m^6)}{m^8} \tag{88}$$

$$f^{(8)}(0,m) = \frac{1680(24+24m^2+12m^4+4m^6+m^8)}{m^{10}} \tag{89}$$

While seemingly progressing without a clear pattern, after a considerable amount of staring at these expressions, a simple recursive formula can be obtained to compute these values: prime for computer implementation. Consider the sequences $a_n, c_n \in \mathbb{R}$ defined by

$$a_0 = m^{-2}; \quad c_0 = 2 \tag{90}$$

and

$$a_{n+1} = \frac{2n(2n-1)a_n + c_n}{m^2}; \quad c_{n+1} = (4n+2)c_n \quad . \tag{91}$$

Using this, one can show that $f^{(2n)}(0, m) = a_n$. This allows for easy evaluation of the asymptotic series of Eq. 85. Numerical experimentation has shown this to be an excellent approximation with a maximum error around 10^{-6} when $|z| = 5$ and $|m| = 1$, retaining only five terms. The error rapidly falls from there as $|z| \to \infty$ and $|m| \to \infty$.

D Asymptotic Approximation for $J(z, x)$ of Eq. 47

Compared to the asymptotic approximation for the $R(m, z)$ integral, a clean expression for simple computer code is not available to our knowledge. Obtaining an asymptotic expression thus relies on access to a computer algebra system. Finding this starts by applying a simple change of variables to Eq. 47 to find:

$$J(z, x) = \int_0^{\Im[z]} \frac{e^{-t^2} e^{2i\Re[z]t}\, \mathrm{d}t}{i(\Re[z] - x) - t} \tag{92}$$

Where it becomes clear that numerical difficulty from expanding the exponential term originates from the $e^{2i\Re[z]t}$ modulation. This is the term to isolate to obtain the correct asymptotic behavior as the integrand becomes increasingly oscillatory. The standard repeated integration by parts procedure can then be applied. This is pure tedium, so we simply report the C++ code which evaluates five terms below.

```
const std::complex<double> pp1 = 6.0 + m2*(6.0 + 3.0*m2) +
  zr*(m*(-3.0 - 3.0*m2) +
  zr*(m2*(2.0 + 2.0*m2) +
  zr*(-2.0*m2*m + 4.0*m2*m2*zr))) +
  zi*(zr*(3.0*ii + m2* (3.0*ii - 6.0*ii*m2) +
  zr*(m*(-4.0*ii - 4.0*ii*m2) +
  zr*(m2*(6.0*ii - 4.0*ii*m2) - 16.0*ii*m2*m*zr))) +
  zi*(6.0 + m2*(6.0
 - 12*m2) + zr*(m*(-3.0 - 18.0*m2) +
  zr*(-2 - 4.0*m2*m2 + zr*(m*(6.0
 - 16.0*m2) - 24.0*m2*zr))) + zi*(40.0*ii*m2*m + zr*(3.0*ii
 + m2*(18.0*ii + 4.0*ii*m2) + zr*(m*(-4.0*ii + 16.0*ii*m2) +
  zr*(-2.0*ii + 24.0*ii*m2 + 16.0*ii*m*zr))) +
  zi*(3.0 + m2*(48.0 +
  4.0*m2) + zi*(m*(-24.0*ii - 16.0*ii*m2) +
  zi*(-4.0 - 24.0*m2 +
```

```
  zi*(16.0*ii*m + 4.0*zi + 4.0*ii*zr) +
  (-16.0*m - 4.0*zr)*zr) +
  zr*(-24.0*ii*m2 + (-16.0*ii*m - 4.0*ii*zr)*zr)) +
  zr*(m*(6.0 +
  16.0*m2) +
  zr*(-2.0 + 24.0*m2 + zr*(16.0*m + 4.0*zr)))))));
const double pp2 = (-6.0 + m2*(-6.0 - 3.0*m2) +
  zr*(m*(3.0 + 3.0*m2) + zr*(m2*(-2.0 -
  2.0*m2) + zr*(2.0*m2*m - 4.0*m2*m2*zr))))/(m2*m2*m);

// Note:std::exp(zi*(zi - 2.0*ii*zr)) = exp(-z^2) / exp(-zr^2)
result =
  (pp2 + pp1 / (cache.emz2 / cache.emrz2 * std::pow(m - ii * zi, 5))) /
  (8. * std::pow(zr, 5));
```

E Root Finding Bootstrap Function

```
double rootfinding_bootstrap_guess(double xi,
        double apprx_0_cdf, // approx CDF at x=0
        double dcdx, // approx PDF at x=0
        double jump, // probability jump at resonance
        std::complex<double> z) {

  // Note: can approximate length as e^{-z^2} * 3 Im[z]
  double yjumplo = 0.5 * (std::erf(z.real() -
    1.5 * z.imag()) + 1.0);
  double yjumphi = 0.5 * (std::erf(z.real() +
    1.5 * z.imag()) + 1.0);

  if (xi <= apprx_0_cdf) {
    if (yjumphi > 0.5 && yjumplo < 0.5) {
      jump *= (0.5 - yjumplo) / (yjumphi - yjumplo);
      yjumphi = 0.5;
    } else if (yjumplo > 0.5) {
      yjumplo = 0.0;
      yjumphi = 0.0;
      jump = 0.0;
    }
    if (jump > apprx_0_cdf) jump = apprx_0_cdf;
    const double d = yjumphi - yjumplo;
    const double sout = (apprx_0_cdf - jump) / (0.5 - d);
    const double sinv = jump > 0.0 ? d / jump : 0.0;
    if (xi >= sout * yjumplo + jump) {
```

```
      const auto r = sout * yjumplo + jump;
      const auto a = (r - dcdx * (yjumphi - 0.5) - apprx_0_cdf)
        / std::pow(yjumphi - 0.5, 2);
      return 0.5 * (-dcdx + std::sqrt(std::pow(dcdx, 2) -
        4.0 * (apprx_0_cdf - xi) * a)) / a + 0.5;
    } else if (xi > sout * yjumplo) {
      return (xi - sout * yjumplo) * sinv + yjumplo;
    } else {
      if (sout > 0.0)
        return xi / sout;
      else return 0.5 * yjumplo;
    }
  } else { // xi > apprx_0_cdf
    if (yjumplo < 0.5 && yjumphi > 0.5) {
      jump *= (yjumphi - 0.5) / (yjumphi - yjumplo);
      yjumplo = 0.5;
    } else if (yjumphi < 0.5) {
      yjumplo = 1.0;
      yjumphi = 1.0;
      jump = 0.0;
    }

    // Clip innapropriately large jumps
    if (jump > 1.0 - apprx_0_cdf) jump = 1.0 - apprx_0_cdf;
    const auto d = yjumphi - yjumplo;
    const auto sout = (1.0 - jump - apprx_0_cdf) / (0.5 - d);
    const auto sinv = jump > 0.0 ? d / jump : 0.0;
    const auto thresh1 = sout * (yjumplo - 0.5) + jump + apprx_0_cdf;
    const auto thresh2 = sout * (yjumplo - 0.5) + apprx_0_cdf;

    if (xi >= thresh1)
      return (xi - thresh1) / sout + yjumphi;
    else if (xi > sout * (yjumplo - 0.5) + apprx_0_cdf)
      return (xi - thresh2) * sinv + yjumplo;
    else {
      const auto a = (thresh2 - dcdx * (yjumplo - 0.5) -
        apprx_0_cdf)/std::pow(yjumplo - 0.5, 2);
      return 0.5 * (-dcdx + std::sqrt(std::pow(dcdx, 2) -
      4.0 * (apprx_0_cdf - xi) * a)) / a + 0.5;
    }
  }

  UNREACHABLE();
}
```