PHY408 Lab 2

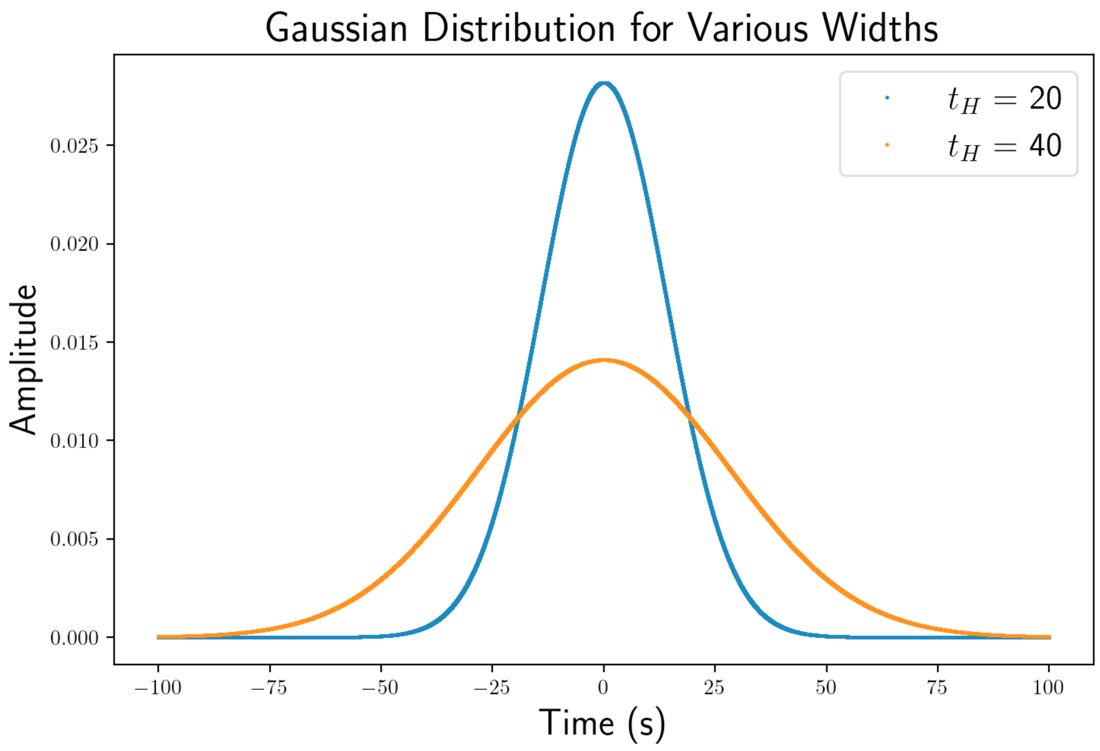
Wednesday, March 6, 2024

Jace Alloway - 1006940802 - alloway1

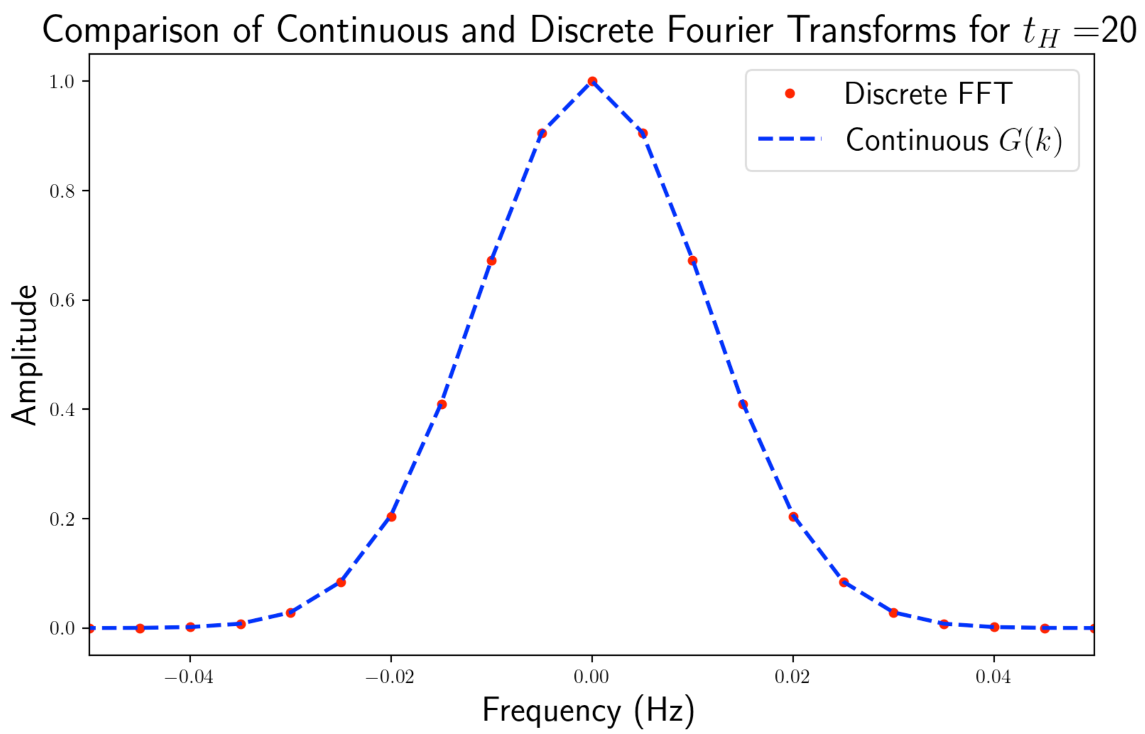
This assignment was compiled in VS Code using the Python and LaTeX extensions. Matplotlib does not produce inline plots in terminal, so the `%matplotlib inline` command was commented out. **Collaborators for all questions: none.**

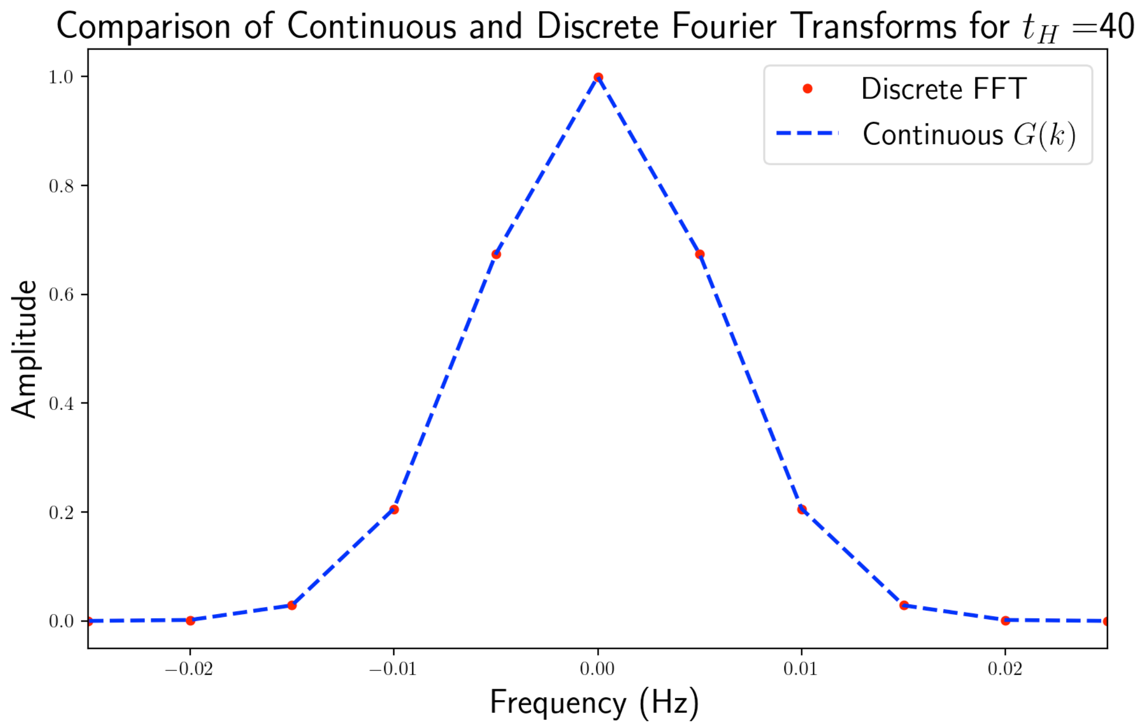
Problem 1

(**a**) We begin by plotting the Gaussian function $g(t) = \dfrac{1}{\sqrt{\pi}\, t_H} e^{-(t/t_H)^2}$ for widths $t_H = 20$s, 40s. Setting $dt = 10^{-3}$ and plotting over the domain $[-100, 100]$, we have



(**b**) One may analytically compute the Fourier Transform (FT) of $g(t)$ to obtain $G(k) = \exp\left[-\dfrac{k^2 t_H^2}{4}\right]$. However, using the `np.fft.fft` operator, the Discrete Fourier Transform (DFT) can be calculated in python and compared with the continuous form. This was done by first defining the frequency spectrum using `np.fft.fftfreq()`, then taking the FT of the sample data from part (a). Applying the `np.abs()` function over the computed DFT to show the amplitude (instead of plotting complex numbers), it was found that the DFT and the continuous FT data obtained from the transform of the Gaussian were practically identical:



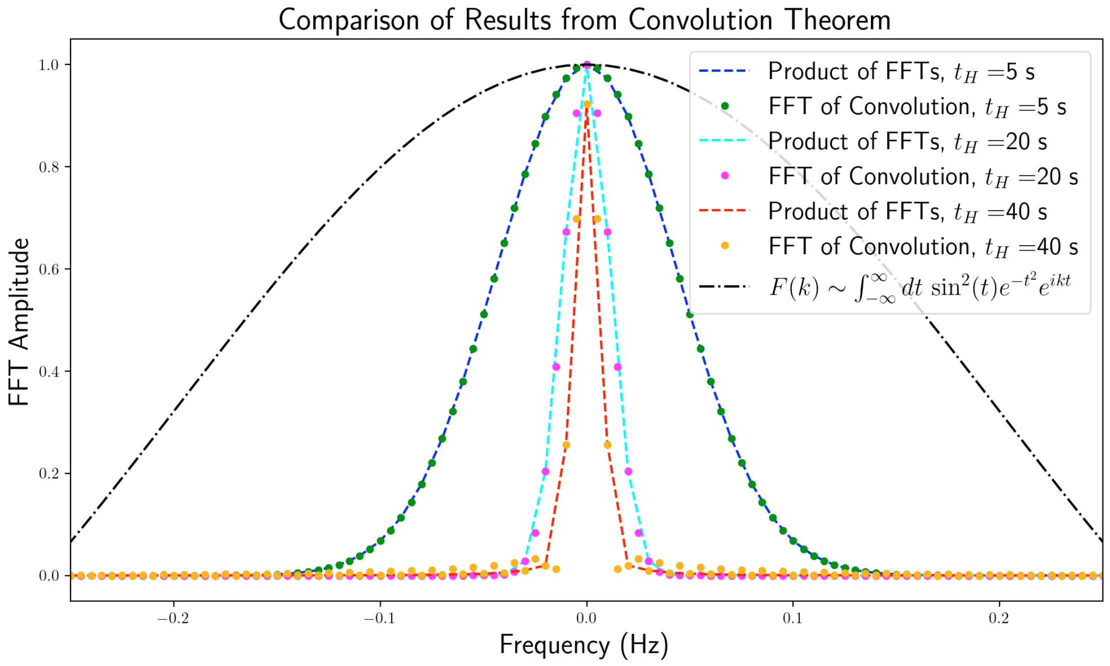


(c) Suppose we were to convolve (filter) some input function $f(t)$ by some Gaussian $g(t, t_H)$. The Convolution Theorem states that, for two functions f and g with respective Fourier Transforms F and G, the FT of the convolution between f and g is equivalent to that of the product of F and G:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g) = F \cdot G \tag{1.1}$$

where '$\mathcal{F}()$' denotes the FT operation on the input. In the case of filtering, the output is then just the point-wise multiplication of the transformed output arrays. Hence, whatever function $f(t)$ is inputted, it's filtered FT is just that of the multiplication of $\mathcal{F}(f)$ with another Gaussian.

For instance, consider the (normalized) function $f(t) = \dfrac{2e}{\sqrt{\pi}(e-1)} \sin^2(t) e^{-t^2}$. Upon convolving f with a Gaussian g, we just have FT-multiplication of both functions. Below is plotted the effects of filtering for both Fourier Transformed convolution and FT-multiplication for various Gaussian widths:



Thus the effect of filtering is to isolate frequencies (which is useful for removing noise in signals) by using various widths.

(d) In terms of the time-frequency uncertainty principle, one may notice that as the Gaussian width appears wider, the peak in the frequency spectrum is more narrow. The similar is true for lower values of t_H (narrow peaks along the temoral axis), implying wider distributions along the in frequency spectrum. For noise filtering noisey signals, this means that noisey signals can be filtered by Fourier Transforming the signal into frequency space, then filtering the noise out with a Gaussian, then transforming the filtered signal back to the temporal axis.

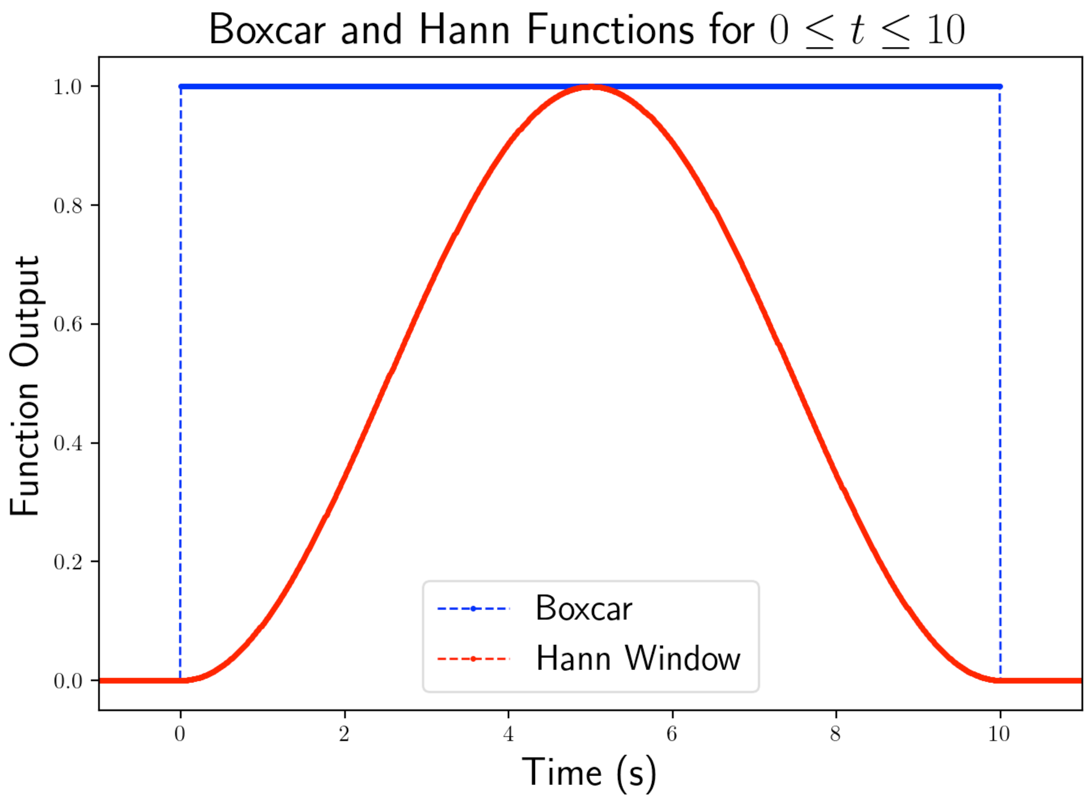
Problem 2

(**a**) This problem focuses on the effects of truncation with the Boxcar and Hinn window functions,

$$b(t) = \begin{cases} 1 & 0 \leq t \leq T \\ 0 & \text{else} \end{cases} \tag{2.1}$$

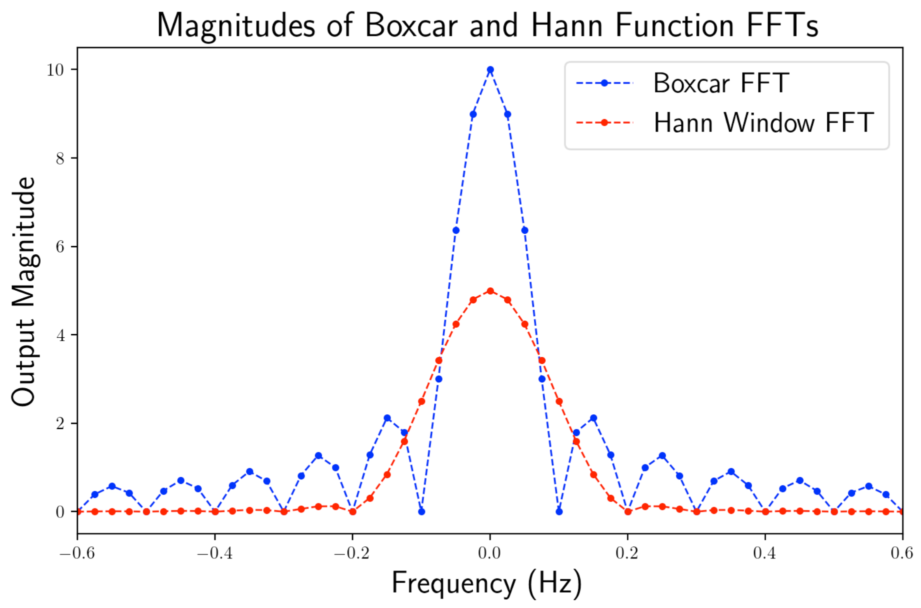
$$w(t) = \begin{cases} \frac{1}{2}\left[1 - \cos\left(\frac{2\pi t}{T}\right)\right] & 0 \leq t \leq T \\ 0 & \text{else} \end{cases} \tag{2.2}$$

These functions can be defined in python by using input parameters t and T with $dt = 0.01$. They are plotted below for $0 \leq t \leq T = 10$s:



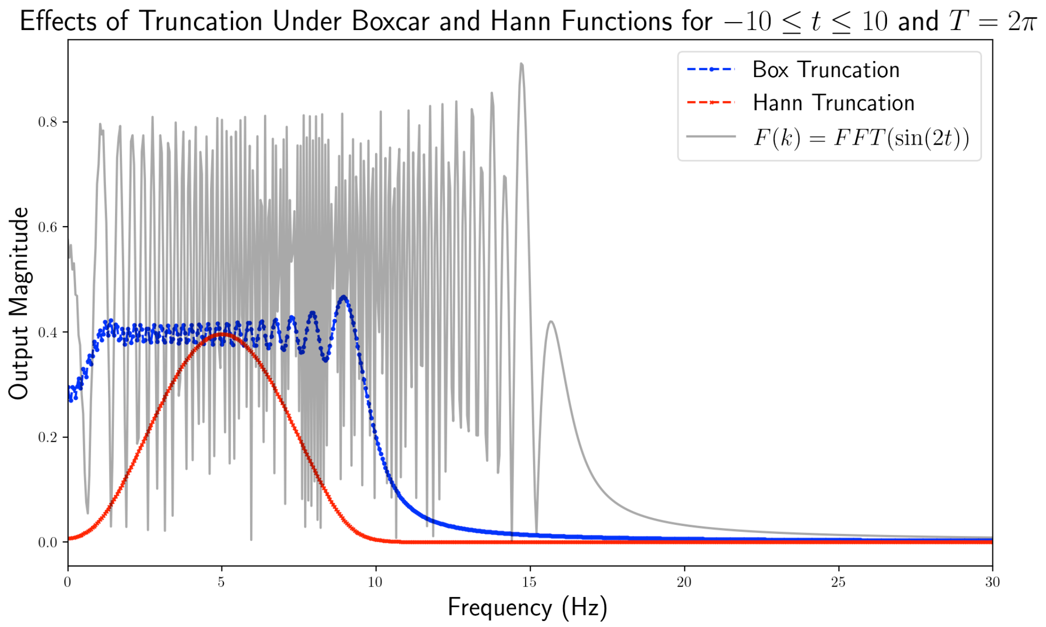
(**b**) The FFT of both window functions can be calulated by noting the input is restricted from 0 to T, is it zero elsewhere. Using `np.fft.fft()`, the FFT of both functions was computed then shifted with `np.fft.fftshift()` so that the zero frequency component of both the frequency array and the output array was centered at 0. The magnitude was then calculated with `np.abs()`.

(c)



I note that I have reduced the domain of the plot so that peaks can be clearly seen, since otherwise they would be unclear. This is interesting because it shows that the boxcar FFT is comprised of multiple frequencies, just as one would expect from approximating a square wave with a Fourier Series. There is one prominent frequency, then the other frequencies are higher and are added to approximate the square. Similarly, the FFT of the Hann window has one distinct peak of small width, since the input frequency $\frac{2\pi}{T}$ is constant and is determined by T.

(d) To understand the effect of truncating an input signal by one of the window functions $b(t)$ or $w(t)$, let's consider an example where we calculate the FFT of the function $f(t) = \sin(2t)$ between -10 and 10 with $T = 2\pi$s and $dt = 0.01$s as before. `np.fft.fft()` assumes that the input signal extends for all time, which isn't necessarily the case.



Periodic input signals can be discontinuous at boundaries, leading to inaccurate FFT outputs. T was chosen so that there would be no discontinuity on the input, since $\sin(2t)$ is discontinuous as $t = -10, 10$s. I have also plotted only the positive frequencies to remove the redundancy of alias frequencies. This is referred to as spectral leakage from discontinuous signals. Since multiplication in the temporal domain is convolution in the frequency domain by the Convolution theorem, we see that the FFT result is a smoothened output absent of the effects of discontinuities. For perfectly continuous FFT's, we would then expect to see spectral leakage from the initial delta-peak in the frequency domain.

(e) From part (d), we can observe that truncation with a boxcar function also introduces other frequencies into the spectrum and still does not quite accurately depict the true, continuous signal. Since the square wave is comprised of multiple frequencies by the Fourier Series, we thus expect to see broadening in the frequency domain. However, with the Hann function, since the window is comprised of one frequency $\frac{2\pi}{T}$, the FFT of the truncation depicts a clearer picture as to what the true input frequency is, since the discontinuity is 'faded out' and not suddenly chopped off. This is the advantage of using the Hann window over the boxcar window, since the Hann window only consists of one frequency, while the boxcar is more precise is isolating periodicity of the input signal, but is comprised of multiple frequencies. Regardless of the case, both functions reduce spectral leakage, but I would prefer to use a Hann window for truncation instead of introducing more frequencies, which may not always assist in reducing leakage.

Problem 3

(**a**) The radial distribution function is given as

$$g(r) = 1 + \frac{1}{2\pi^2 \rho r} \int_0^\infty dk\, k[S(k) - 1] \sin(kr). \tag{3.1}$$

Considering the integral term $\int_0^\infty dk\, k[S(k) - 1] \sin(kr)$, we observe that we can re-write it using complex exponentials, since

$$\sin(kr) = \frac{1}{2i}\left[e^{ikr} - e^{-ikr}\right]. \tag{3.2}$$

Assuming $S(k)$ is an even function $(S(-k) = S(k))$, we find that

$$\begin{aligned}
\int_0^\infty dk\, k[S(k) - 1] \sin(kr) &= \frac{1}{2i} \int_0^\infty dk\, k[S(k) - 1][e^{ikr} - e^{-ikr}] &(3.3)\\
&= \frac{1}{2i} \int_0^\infty dk\, k[S(k) - 1]e^{ikr} - \frac{1}{2i} \int_0^\infty dk\, k[S(k) - 1]e^{-ikr} &(3.4)\\
&= \frac{1}{2i} \int_0^\infty dk\, k[S(k) - 1]e^{ikr} + \frac{1}{2i} \int_{-\infty}^0 (-dk)\,(-k)[S(-k) - 1]e^{-i(-k)r} &(3.5)\\
&= \frac{1}{2i} \int_0^\infty dk\, k[S(k) - 1]e^{ikr} + \frac{1}{2i} \int_{-\infty}^0 dk\, k[S(k) - 1]e^{ikr} &(3.6)\\
&= \frac{\pi}{i} \frac{1}{2\pi} \int_{-\infty}^\infty dk\, k[S(k) - 1]e^{ikr} &(3.7)
\end{aligned}$$

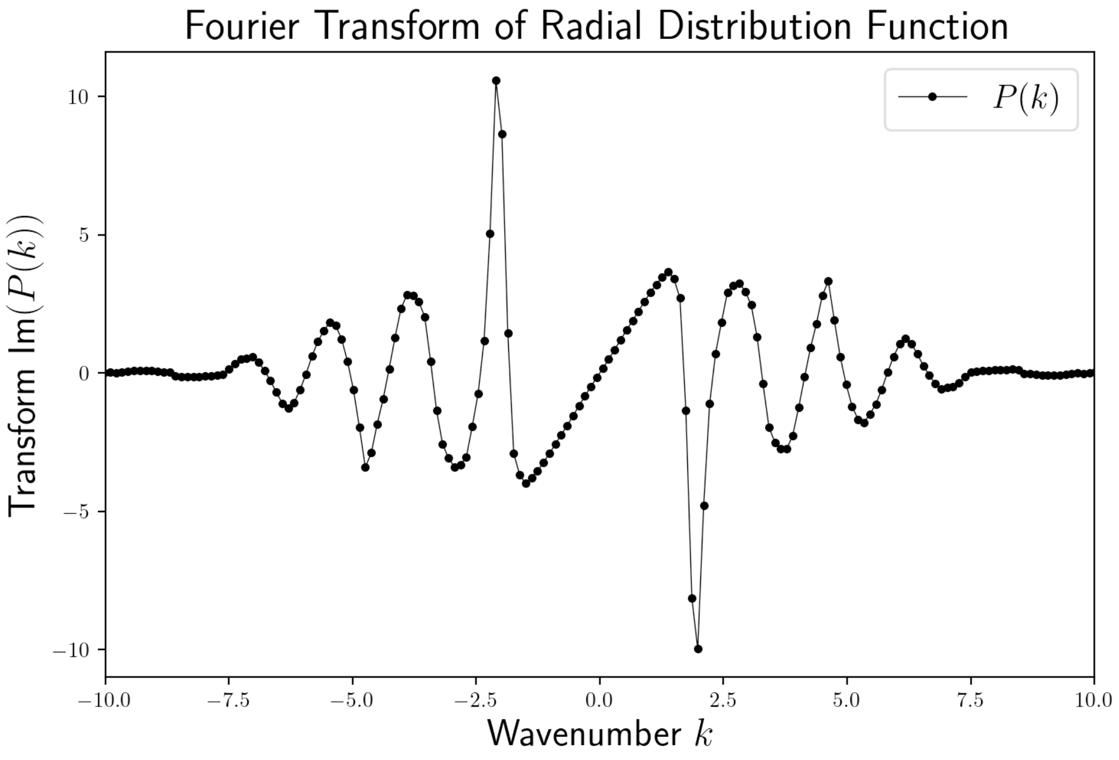
and thus we can write (2.1) in terms of a Fourier Transform,

$$g(r) = 1 + \frac{1}{2\pi^2 \rho r} \cdot \frac{\pi}{i} \frac{1}{2\pi} \int_{-\infty}^\infty dk\, k[S(k) - 1]e^{ikr}. \tag{3.8}$$

(**b**) Considering (3.7), let us now find the Fourier Transform of $p(r) = \int_0^\infty dk\, k[S(k) - 1] \sin(kr)$, from which we have

$$\begin{aligned}
P(k) &= \int_{-\infty}^\infty dr\, p(r) e^{-ikr} &(3.9)\\
&= \frac{\pi}{i} \frac{1}{2\pi} \int_{-\infty}^\infty dr \int_{-\infty}^\infty dk'\, k'[S(k') - 1]e^{ik'r}e^{-ikr} &(3.10)\\
&= \frac{\pi}{i} \frac{1}{2\pi} \int_{-\infty}^\infty \int_{-\infty}^\infty dr dk'\, k'[S(k') - 1]e^{i(k'-k)r} &(3.11)\\
&= \frac{\pi}{i} \int_{-\infty}^\infty dk'\, k'[S(k') - 1] \left[\frac{1}{2\pi} \int_{-\infty}^\infty dr\, e^{i(k'-k)r}\right] &(3.12)\\
&= \frac{\pi}{i} \int_{-\infty}^\infty dk'\, k'[S(k') - 1]\, \delta(k' - k) &(3.13)\\
&= \frac{\pi}{i} k[S(k) - 1] &(3.14)\\
&= \pi i\, k[1 - S(k)]. &(3.15)
\end{aligned}$$

We now see that $P(k)$ is in general complex, and since $S(k)$ is even, $kS(k)$ is odd, hence $P(k)$ is also odd. Since the Fourier transform of $p(r)$ is imaginary and odd, this implies that $p(r)$ must be a real, odd-valued function.
To plot $P(k)$, an even function was created out of `YanData` by using `np.concatenate((YanData[::-1], YanData[1:]))` (ie, flipping the first array and adding to it the original minus the first element).



As noted above, $P(k)$ is complex and odd.

(c) To program the radial distribution function, we must realize that the $g(r)$ function is written as a inverse Fourier Transform, and the `np.fft.ifft()` function requires the $a[0]$ component of the input array to be the 0-frequency component. Since

$$g(r) = 1 + \frac{1}{2\pi^2 \rho r} \cdot \frac{\pi}{i} \frac{1}{2\pi} \int_{-\infty}^{\infty} dk\, k[S(k) - 1]e^{ikr} \tag{3.16}$$

is the continuous FT, we note that the inverse DFT is

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} x_k e^{2\pi i \frac{k}{N} n} \Delta k \tag{3.17}$$

where the zero-point frequency component is $x_k[0]$ with $k[0]$. To get (3.17) in the form of $p(r)$ in (3.7), we note that the integrand is $\pi i k[1 - S_k]$, so

$$p_n = \frac{1}{N} \sum_{k=0}^{N-1} \pi i k[1 - S_k] e^{2\pi i \frac{k}{N} n} \Delta k. \tag{3.18}$$

Secondly observe that by scaling the k axis by $k' = k \times \frac{N}{2\pi}$, we find that

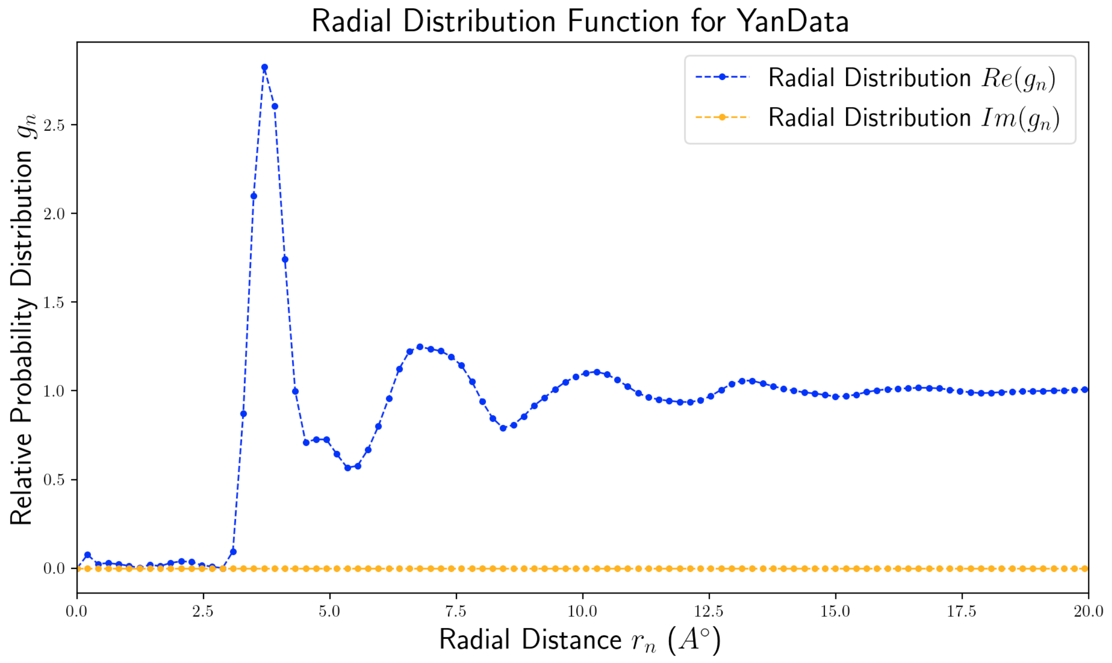
$$p_n = \frac{1}{2\pi} \sum_{k'=0}^{N-1} \pi i k' [1 - S_{k'}] e^{ik'n} \Delta k \tag{3.19}$$

where the k' axis is then defined as `np.arange(-(N+1)/2, (N+1)/2) * (N*dk) / (2*np.pi)`. We note that we only are scaling k into k', and that this does not change the number of samples N in either k or S_k, so we still sum from $k' = 0$ to $N-1$. Once k has been scaled, we use `np.fft.ifftshift()` to shift both k' and $S_{k'}$ arrays to a form such that the zero-frequency components are located in the first array element. Under this condition, we can then invoke the inverse DFT, `np.fft.ifft()` of our p_n in (3.19), since this is in the form of $p(r)$ in (3.16).
We lastly obtain p_n along an axis r_n of length N (since array lengths are preserved under DFT's) spaced at dk^{-1}, the inverse of the input spacing. However, these r_n values are scaled by $\frac{2\pi}{N}$ to return back from the continuous form into the discrete form, and hence we have the output radial axis as `rn = np.array(N) * (2*np.pi)/(N*dk)` (this is `rn[i] = 1/k[i]`, the inverse array, as expected). We lastly write out our g_n as defined in (3.16), returning the array of r_n and g_n values `[rn, gn]`.

(d) Taking $dk = 0.12$ ($\mathring{A}^{-1}$) and

$$\rho = \left[6.022136 \times 10^{23} \frac{\#\text{ atoms}}{\text{mol}}\right] \times \left[\frac{1}{39.948} \frac{\text{mol}}{\text{grams}}\right] \times \left[1.4273 \frac{\text{grams}}{\text{cm}^3}\right] \times \left[\frac{\text{cm}^3}{10^{24}\mathring{A}^3}\right] \approx 0.021516 \frac{\#\text{ atoms}}{\mathring{A}^3} \tag{3.22}$$

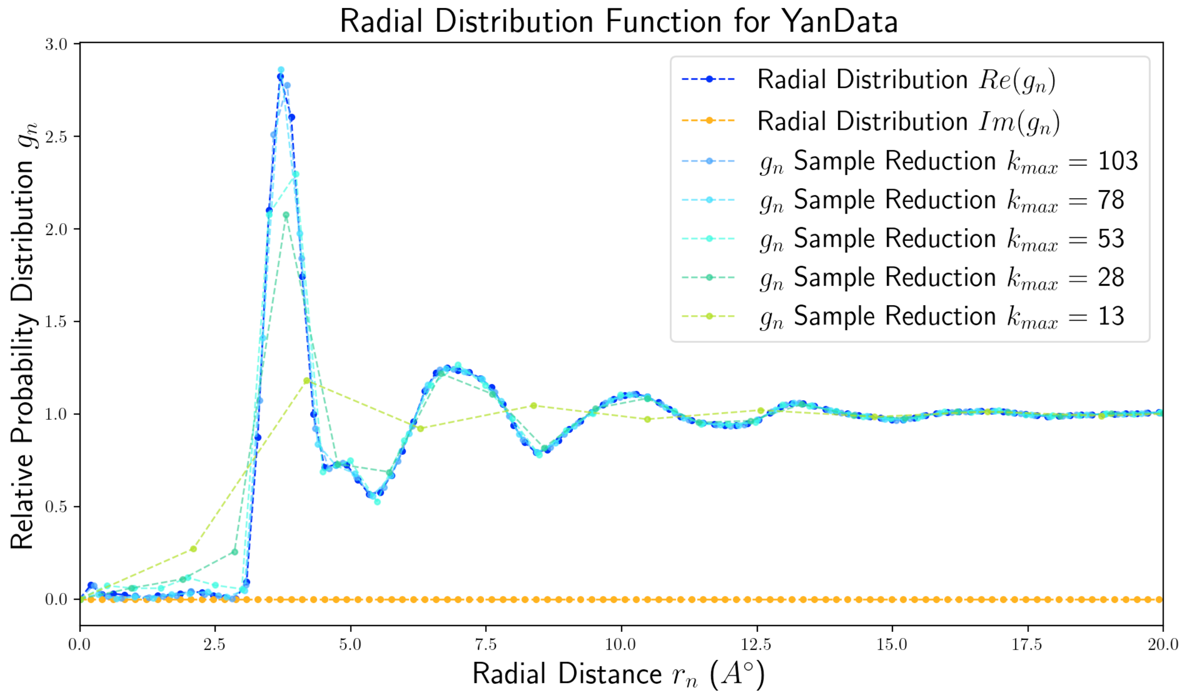
we can plot g_n as computed before:



Observe that, as expected, g_n approaches 0 as r_n goes to 0, and approaches 1 as $r_n \to \infty$, with a distinct peak around the $r = 3.5\mathring{A}$ mark. Although only the $(0, 20)\mathring{A}$ domain was plotted, $g(r)$ for radius greater than 20Å appears to become oscillatory again, so any radius value less than 20Å can be trusted.

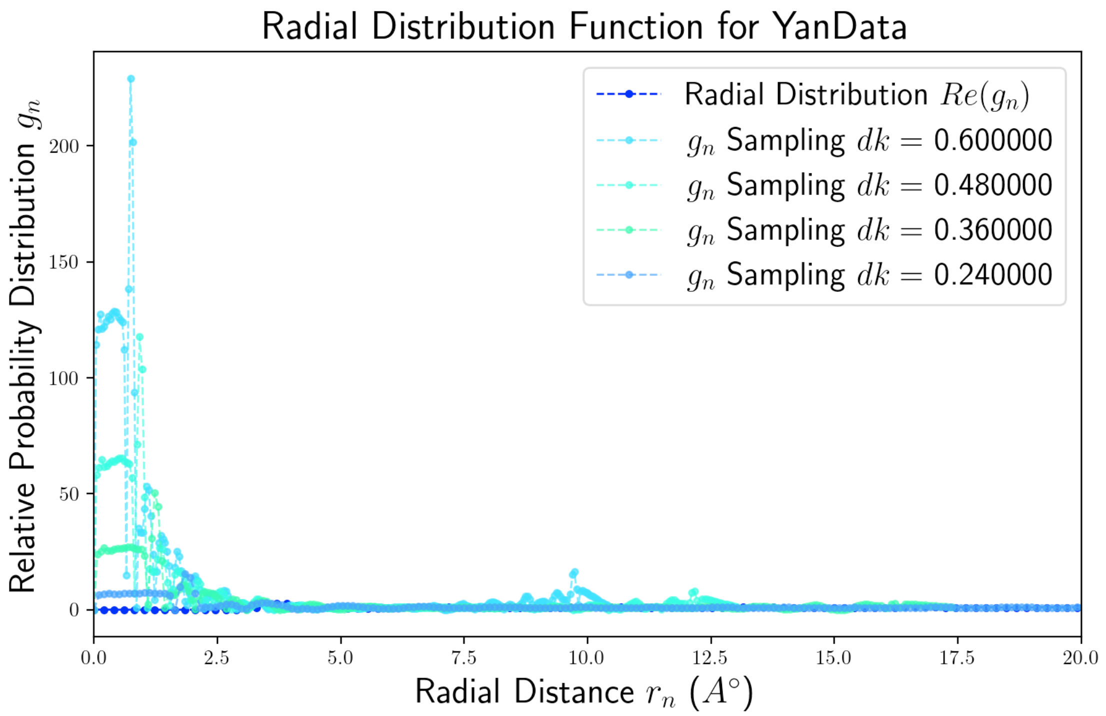
(e) The average molecular radius can be calculated via the expectation value $\langle r \rangle$, defined (continuously) as $\langle r \rangle = \int_{-\infty}^{\infty} dr\, r g(r)$ since $g(r)$ is a probability distribution function. This can be discretized to a sum $\sum_{n=0}^{N-1} r_n g_n$ over the plotted $(0, 20)$Å interval, and this was found to be $r_{\text{exp}} = 3.7$Å which also appears to be the maximum of the PDF.
The uncertainty attributed to this value can be derived from the variance about r_{exp}, calculated (discretely) from the sum as $\sum_{n=0}^{N-1} (r_n - r_{\text{exp}})^2 g_n$, from which an uncertainty of $\delta r = \pm 0.4$Å. Hence we compute $\langle r \rangle = 3.7 \pm 0.4$Å.

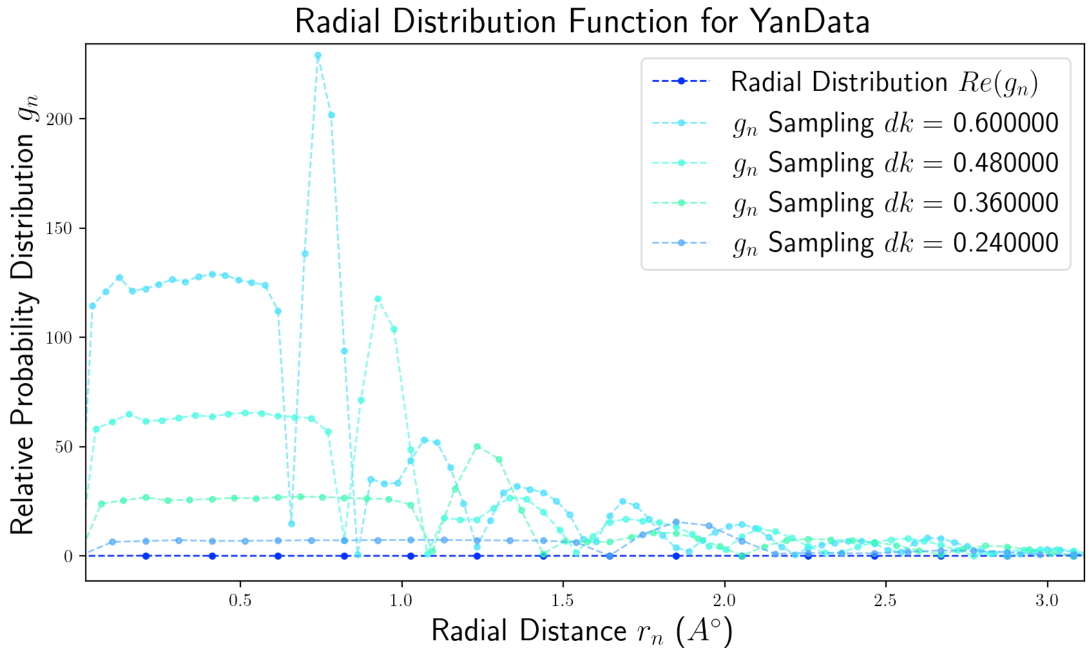
(f) Let us now consider the effect of reducing the length of input structure data `YanData` (hence, reducing S_k). We analytically observe that, when reducing N, we also reduce the number of output samples of g_n. Due to the exponential frequency factors in the inverse DFT $e^{2\pi i \frac{k}{N} n}$, we should therefore observe a reduction in scaling of g_n close to 1 as $N \to 0$ since the frequency terms become infinitely short. Althogether, as we decrease the sampling size, we should observe 'rough' discretizations of the initial input array:



As expected, we see that the output $g_n \to 1$ (constant) as $N \to 0$, so we observe lower peaks with farther sample spacing. Above are shown reductions to the initial 128-sample long YanData array of $-25, -50, -75, -100$, and -115 and although one may still observe peaks for the 13-sample output, they are not exactly distinct and they are also not accurately representative of the true output signal. Thus I would interpolate to conclude that any sample size reduced by $100 - 115$ samples would be sufficient to still be observing peaks.

(g) Lastly let's suppose that instead of reducing k_{max} (the sampling size), we instead increase the sample spacing dk. According to our function g_n as derived in part (c), we should expect two things to happen when we increase dk: the first is that the output sampling will be scaled by the factor inversely proportional to how we scale dk, ie if we scale $dk \to 2dk$, then $r_n \to 0.5r_n$, so whatever what was at 20Å will be scaled to 10Å, etc. Thus every time we increase dk we expect our samples to become closer together, pushing the DFT to the left. We secondly should observe a scaling of p_n, since we multiply every IDFT by a factor of dk (equation (3.18)), so we also expect to see the peaks / valleys of g_n to be increased, while we still have the $g_n \to 1$ and $r_n \to \infty$ behaviour.





Initially, we see that g_n is close to zero when r_n is close to zero, for dk unchanged. However, as soon as dk is increased, the behaviour close to zero no longer vanishes, thus implying that particles could inhabit the region where no particle should be. As dk increases further, we continue to observe this. As expected, we see the domain decreasing and amplitude increasing. While the first two peaks are always still distinct when zoomed in to the appropriate length scale, we can observe that the signal has become heavily distorted otherwise and inaccurately represents the initial signal. Thus I would suggest that any dk scaling above 2 - 3 is the maximum while still being able to see peaks clearly at the initial $20\mathring{A}$ scale, although the initial line (which should be zero) is no longer zero.