Final Exam for PHYS 512. Due before end of exam period. There are two bonus points for following instructions, plus a bonus question, which can make up for mistakes elsewhere. All codes/work should be put into a zip/tar file and emailed to me. Do NOT put your answers on github. You may refer to notes/old codes and use google, but you may not consult anyone (including online) besides the TAs and the instructor. You may not consult LLMs such as chatgpt. There are three questions, plus some bonus points available. The final score is capped at 100. You have 24 hours to complete from when you start working.

Problem 0: (2 points)
Please put everything into a directory. Include a file called "myinfo.txt" that has your name and ID in it, so I can match grades to students at the end. Please do NOT put your name elsewhere in your codes/answers.

Problem 1: Numerical integration and the Nyquist theorem. (25 points)
Consider a function that goes to zero for large $|x|$, such as a Gaussian. We will investigate integrating such a function from $-\infty$ to ∞.
Part A: We saw we can integrate functions by approximating pieces of them with polynomials, and then our final integral can be expressed as

$$dx \sum_{i=0}^{n-1} c_i f(x_i) \tag{1}$$

where dx is the spacing between our points x_i, $f(x_i)$ is our function evaluated at our x-points, and c_i is a set of coefficients that specifies our integration scheme. For example, Simpson's rule would have

$$c_i = [1/6, 4/3, 2/3, 4/3, 2/3, 4/3, ..., 2/3, 4/3, 1/6]$$

. What is the *average* value of the coefficients c_i, ignoring the coefficients at the beginnings/ends? How does it depend on the order of the polynomial you would have used to generate the c_i? You may wish to consider $f(x_i) = 1$ since any sensible integration scheme should get the correct answer for that.
Part B: If we are trying to integrate a function that has gone to zero at the edges, we are free to pick any starting point we wish to carry out our integration. Use this fact and your results from Part A to explain why

$$\int_{-\infty}^{\infty} f(x)dx \sim dx \sum f(x_i) \tag{2}$$

is accurate to *arbitrary* polynomial order, again as long as $f(x)$ goes to zero sufficiently quickly for large x.
Part C: This may seem quasi-miraculous (Equation 2 is normally a *terrible* way to integrate a function with error scaling like dx), but deep down it's a reflection of the Nyquist theorem. If we have evenly sampled our function, we have complete information on all frequencies up to the Nyquist frequency. There isn't extra information to be gleaned by increasing our polynomial order as long as the Fourier transform has also gone to zero (which it does for a Gaussian). Carry out the numerical integration of a Gaussian as a function of dx where $x = 0$ is one of your points with dx ranging from some small number up to $dx = 2$. Plot the error in your integration as a function of dx. Please do this on a log scale for the y-axis, and plot the absolute value of the error. What is the error for $dx = 1$? Is the error consistent with the naive expectation of dx?

Problem 2: Stability of the Diffusion and Schrodinger Equations (35 points)
The diffusion equation is

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} \tag{3}$$

In this problem, we'll examine the stability of this equation, and of the closely-related Schrodinger equation.
Part A: show that the second derivative with respect to x can be written

$$\frac{\partial^2 f(x,t)}{\partial x^2} \sim \frac{f(x+dx,t) - 2f(x,t) + f(x-dx,t)}{dx^2} \tag{4}$$

Part B: What is the condition for stability for the diffusion equation when using the $x-$derivative as expressed in Equation 4? Use $\frac{f(x,t+dt)-f(x,t)}{dt}$ for the time derivative. Please express your answer in terms of $\delta t < stuff$ where you work out the expression for $stuff$.
Part C: The time-dependent Schrodinger equation is

$$i\hbar\frac{\partial \Psi}{\partial t} = H\Psi \tag{5}$$

For a particle in empty space where the potential is zero, and merging all constants into D, we're left with

$$i\frac{\partial \Psi}{\partial t} = D\frac{\partial^2 \Psi}{\partial x^2} \tag{6}$$

This is the same as the diffusion equation, but with an extra factor of i. Redo your stability analysis from Part B, but now including the i.

Part D: Write a code to evolve a Gaussian using both the Schrodigner and diffusion equations. Include a plot that shows the stability behaves as expected. You may assume that all functions go to zero at the edges for your boundary conditions. I suggest using 500 points from -10 to 10 for your $x-$values.
Bonus (up to 5 points missed elsewhere): Implicit schemes like the one we say for ODEs also play an important role in PDEs. Redo the stability analyses from Parts B and C, but with the derivative at time t evaluated using function values at time $t+dt$. What are the conditions for stability for both equations now?

Problem 3: Searching for an Exoplanet (40 points)
In this problem, we'll look for an "exoplanet" hiding in some simulated data. As the exoplanet goes around its sun, if the orbit is right, it will periodically pass in front of the star for a short period of time. When it's in front of the star, it will block a small fraction of the star's light. Our job is to look at a long stretch of data, and try to find and characterize these small dips. Once we do that, we can say how far the planet is from its star (via the spacing of eclipses), how big the planet is (via the depth of the eclipses), and something about the tilt of the planet's orbit (via the duration of each eclipse).

For simplicity, we'll model the eclipse profile as a chain of evenly-spaced Gaussians. One simple way to generate this is via a cosine. Let each eclipse looks like $\exp(-(t-t_0-np)^2/2\sigma^2)$ where t_0 is the time of the first eclipse, p is the period between eclipses, n is an integer, and σ is the width of each eclipse.
Part A: Show that

$$y = \exp(\alpha^2(\cos(2\pi(t-t0)/p) - 1)) \tag{7}$$

closely approximates the chain of Gaussians by analytically expanding to second order around a peak. Derive the relation between α and σ with this approximation (your expression will also include the period and some constants).
Part B: One way to get started searching for the planet is by looking at the Fourier transform of the data. In class, we saw that the Fourier transform of a set of spikes separated by n samples was a set of n evenly-spaced spikes. By replacing each spike with a Gaussian, we're effectively convolving the set of spikes with a single Gaussian. Consider the case where t goes from 0 to 5000, the spikes are spaced every 50 samples, and have a sigma of 3 samples. You should get that your signal looks like Figure 1, which shows

the full timestream, and a zoom-in on a stretch so you can see the individual peaks. Now plot the Fourier transform of your peak chain on top of the Fourier transform of a single peak (please take absolute values of the Fourier transforms before plotting). To get the two curves on the same scale, divide each time series by its sum before Fourier transforming. Explain what you see based on the Fourier transform of the picket fence. Show that the first spike in the FT of the chain happens at a k equal to the total number of spikes in the chain.
Part C: The number of spikes we see in the FT also tells us about the width of each individual Gaussian. Show that the envelope is a Gaussian with $\sigma_k = \frac{n}{2\pi\sigma_t}$ where n is the number of data samples, σ_t is the σ for an individual spike, and σ_k is the envelope of the Fourier-space spikes (you may show this via a plot if you don't want to work it out analytically).

Part D: Look at the data in planet_data.npy (which can be loaded with "dat=np.load('planet_data.npy')"). If you look at the raw data, any individual transit/Gaussian spike is too faint to see. However, the signal is very bright in Fourier space. Plot the absolute value of the Fourier transform, and use that to quantitatively estimate the number of transits (spikes) and the sigma of each transit. Please give your plot using an sensible range on the $x - axis$ and give your estimates for the number of spikes and the σ_t of each spike. Convert the number of spikes to the period of each transit and report that. As a hint, the *period* of the spikes is a random number between 300 and 600, and σ_t is between 5 and 10. You can use np.argmax to find the index corresponding to the maximum value in an array, which may come in handy.
Part E: We still need to guess the time when the first spike happens since that sets the phase of the cosine. Use the first 10% of the data and use a matched filter to estimate the position of the first transit (t_0 in Equation 7). What is your estimate for the position of the first transit? The data are close to 1, with dips slightly below one. You may find this part easier to do if you work with (1-data) so the background is zero and the peaks are positive. Note - if you were unable to get part D to work, I have saved an alternate data set "alternate_data.npy" with period 495.792 and σ=6.5. You can report the position of the first peak using this dataset instead.
Part F: Do a least-squares fit to the planet data to get your best estimate for the period, σ_t, the depth, and the time of the first transit. Estimate the noise in the data, and use that to estimate error bars on your fit parameters and report them. You are free to use whatever technique you wish (including MCMC, LM), but I have included a Newton's method solver with numerical

derivatives in newton.py. If you import newton, then newton.newton will attempt to solve for best-fit parameters with inputs (starting parameters, function, data, white noise, step_size) with optional parameters n_iter for number of iterations of Newton's method, and fac (default value=1) which is what fraction of a Newton's method step to take. The function newton.manygauss will return the chain of Gaussian timestreams, but you will need to put in your conversion from Gaussian σ to α from part A.

As an aside, we've looked at this in the context of expolanet searches, but the same techniques apply in many other areas of physics. For instance, X-ray crystallography is essentially identical to this - the X-ray diffraction *is* the Fourier transform, and there's a nearly direct analogy between regular crystal structure and eclipse patterns.

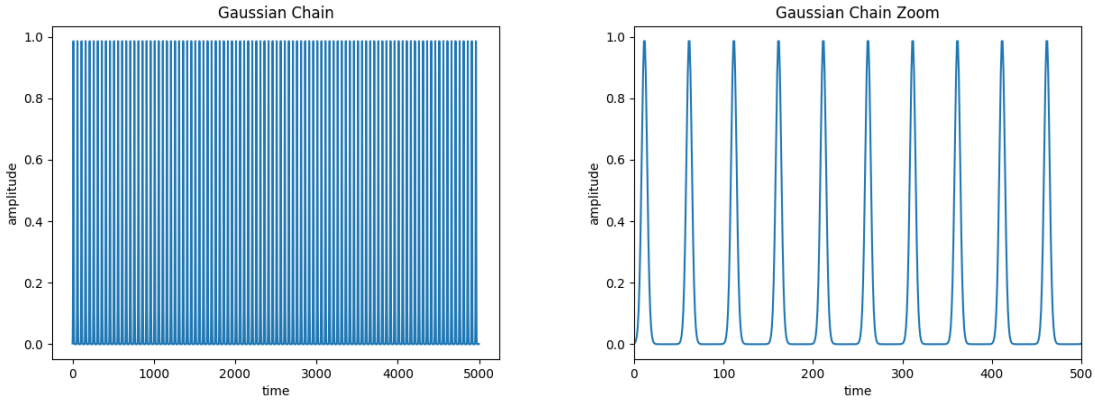

Figure 1: Examples of a chain of Gaussians. The left panel shows a full timestream, while the right shows a zoom on the first part so the individual Gaussians are visible. This sort of data is typical for the amount of light an eclipsing exoplanet might block. A realistic set of data would be one minus a small number times this function.