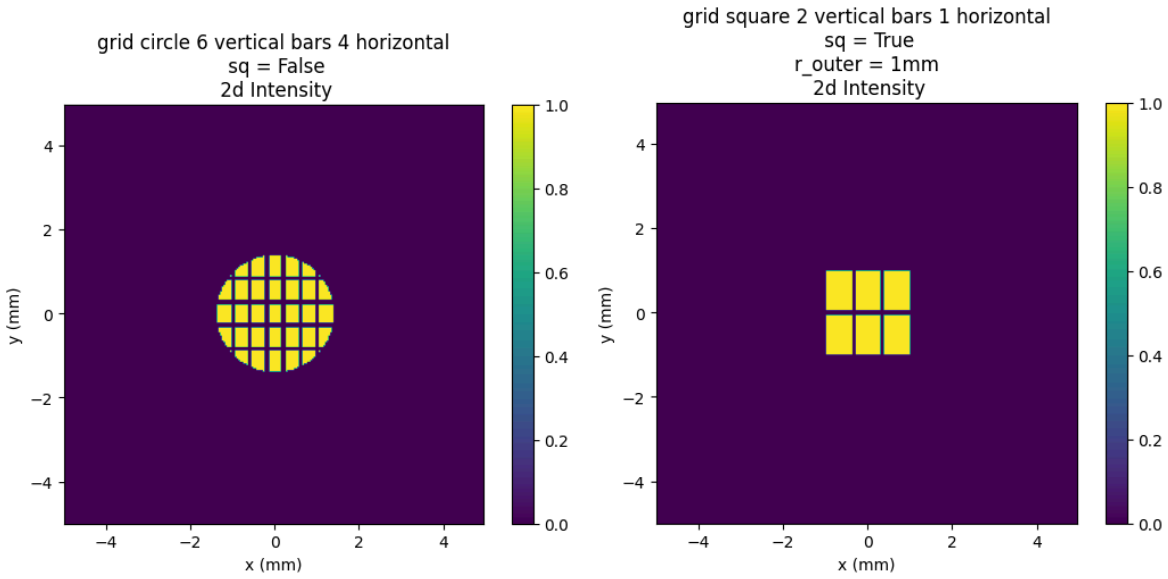
Due December 16th 2024
NEED TO FIX NAMING OF GRIDCOORDS AND COORDINATES (FOR EXAMPLE GRIDCOORDS -> EXAMPLE COORDINATES)
In this project, the goal is to construct a simulation of a laser cavity, featuring simple models of key components, the gain medium, the cavity mirrors (which are only partially reflective and thus represent a loss), and an aperture to determine which modes are allowed. Importantly, the simulation will account for the power in the beam, not exclusively working with unit normalized amplitudes. This project is partially a reimplementation and adaptation of the LightPipes library example simulation of a laser cavity. If you are really stuck trying to get your simulation to produce reasonable results, I recommend running their interactive simulation to get an intuition for low order modes. I also recommend LightPipes for serious use. In lecture and course work I have shown code that implements only a small amount of axial simulation functionality as is available in LightPipes.

You need to pip install tqdm, pip install scipy if you are working with python. Image processing toolbox if MATLAB.

To block certain beam modes from forming in the cavity, different aperture geometries will be used, implementation is provided for a spider aperture, which has blocking lines that fall in a spoke pattern. The aperture is represented by a field with 1 entries where light may pass through, and 0 where some absorptive material is present. The aperture is equipped with a set of grid coordinates, just like an amplitude field. If the aperture represented as above has the same grid coordinates as a simulation amplitude field, the aperture may be applied to the amplitude field by a placewise multiplication.

Task: Implement **grid_aperture(coordinates, r_outer, arm_thickness, N_x, N_y, sq = False)** Where **coordinates** is the grid of points to sample the aperture at, **r_outer** is the radius of a circular hole, or half the side length of a square hole if **sq** = True. Argument **arm_thickness** defines how thick to make the blocking bars, and **N_x, N_y,** give the number of gridlines or bars which run in the *x* and *y* transverse directions.

Some examples of the aperture function are called on a 10mm sidelength sample grid. Note all the bars should be the same thickness, but because of inadequate sampling, they may not be perfectly resolved and so have greater or lesser pixel widths.



1.0 indicates light is allowed through

Hint: Notice that the spacing for **two** vertical bars for example is determined by dividing the side length or diameter (in the case of a circular outer aperture) into **three** sections

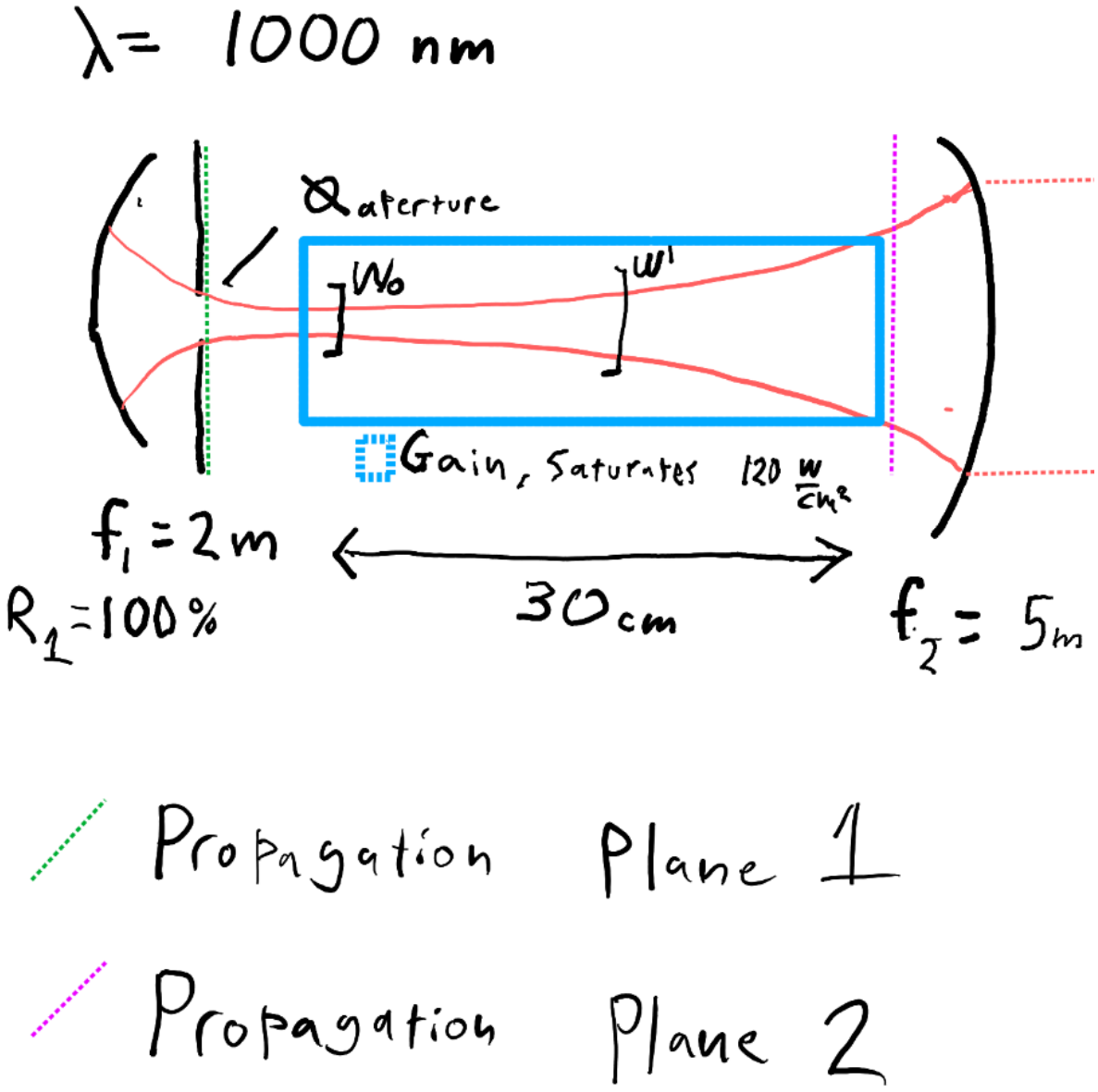
Task: Implement **random_amplitude(coordinates)**
This function should return an amplitude pattern where each amplitude sample is randomly and independently identically distributed about zero. The returned pattern should have amplitude normalized in the sense that the sum of square amplitude samples should be 1.0. Careful use of normalization will allow the power and thus intensity of the beam to be kept track of.

Task: Implement **gain_sheet_saturable(coordinates, field,sat,alpha,power,size)** is similar to LightPipes **Gain** function. This function assumes the propagation through the gain medium both ways can be modeled as applying the gain to the vacuum propagated field at the end mirrors of the cavity. The gain is said to be saturable as the gain factor depends on the intensity, falling off as the beam gets more intense. If this was not the case, the cavity would become infinitely energetic which is not physically possible. The gain under such approximations can be represented by samples in a similar way to the aperture, where the sample point is a value in $[1, \infty)$ that represents the gain factor (rather than a 0, or 1 for a binary aperture). Just like the aperture, if the gain is sampled over the same grid points as some amplitude field, the gain may be applied simply by placewise multiplication. The following equations define the gain factor of the saturable gain medium with cross section that covers the entire transverse plane:

$$A_{out}(x,y) = A_{in}(x,y)e^{\alpha[2L]},\ \alpha = \frac{\alpha_0}{1 + 2I_{in}(x,y)/I_{sat}}$$

Essentially, use samples of $I_{in}(x,y) \propto |A_{in}(x,y)|^2$ to generate samples of the roundtrip gain factor $e^{\alpha[2L]}$ which has *x,y* dependence through α. The input amplitude, **field,** should be normalized so that the sum of its sample magnitude squares are 1.0. Then to get samples that represent the intensity at the sample grid points, scale the magnitude squared of the normalized

amplitudes samples by the **power**, and divide by the grid spacing squared, since the intensity is power per unit area. Getting the conversion of our power variable, and normalized amplitude samples to intensity samples correct is key in determining a physically reasonable output beam power.



Task: Implement **propagate_roundtrip(coordinates, amplitude, power, aperture, R)**. This function will take some amplitude at the propagation plane 1:

- add by direct sum a random field with power **power_random**
- apply lensing due to mirror 1 with focal length $f_1 = \frac{r_1}{2}$, r_1 is the radius of curvature of the mirror
- an **aperture** (this will reduce the power), which allows us to select modes

- Fresnel propagate *L = 30cm to* plane 2
- apply roundtrip saturable gain
- apply lensing due to the curvature of mirror 2, and a loss *R* (non-unity reflectivity of mirror 2) in the power, since *(1- R)* is emitted from the cavity as laser radiation.
- Finally propagate back *L* to plane 1.

The function then returns the new amplitude, and power resulting from the roundtrip in the cavity. To deal with the aperture, gain/loss, and random injected radiation affecting power, one should carefully move between using unit normalized amplitudes, and unit normalized amplitudes multiplied by their sqrt(power) (for example to add the random radiation the addition should be done between two normalized fields with amplitudes scaled by the square root of their power). There only need to be two propagation planes because we treat the aperture, curved mirrors as thin components, and approximate the gain medium propagation as a vacuum propagation and a scaling. The mirrors are assumed to be well aligned, so that their only effect is to reverse the beam propagation direction, which does not change anything for our simulation. Note that our simulation is for initially right traveling radiation, but we could equally well formulate one for initially left traveling radiation.

Since this whole process is a round trip, iteration of this function models the bouncing of some photons within the cavity. Note that there are many more than five parameters to this roundtrip simulation, but the choice is made for convenience to store any parameters we do not plan to change within a simulation as global variables, only making dynamic variables arguments to the function.

Now that we have available functions to form the cavity simulation, we should make estimations of what aperture is needed for the cavity. This is where the Gaussian beam is useful, it will provide a simple calculation of roughly how big to make the aperture to allow only the lowest order Gaussian mode (higher order modes are larger spatially and will be extinguished by a small enough aperture radius). The following analytic form predicts the minimum width of a Gaussian cavity mode w_0:

$$w_0^4 = \left(\frac{\lambda}{\pi}\right)^2 \frac{L(r_1-L)(r_2-L)(r_1+r_2-L)}{(r_1+r_2-2L)^2}$$

(from *Alignment of resonant optical cavities* Dana Z. Anderson, Eqn. 2)
Further we will increase the size of the cavity in simulation to calculate more complicated modes.

Question: For our cavity with $\lambda = 1000nm$ what is the minimum beam width w_0? We can roughly extrapolate this width to be the width at the mirrors within some small factor since the length of the cavity is short compared to the focal length of the lensing walls.

Question: Given the small signal linear density of gain $\alpha_0 = 0.5m^{-1}$, the small signal gain factor $e^{\alpha_0[2L]}$, predict at what reflectivity *R* of the mirror (the smaller the reflectivity the greater the cavity loss) the gain will be sufficient for gain to overcome losses and produce a stable beam.

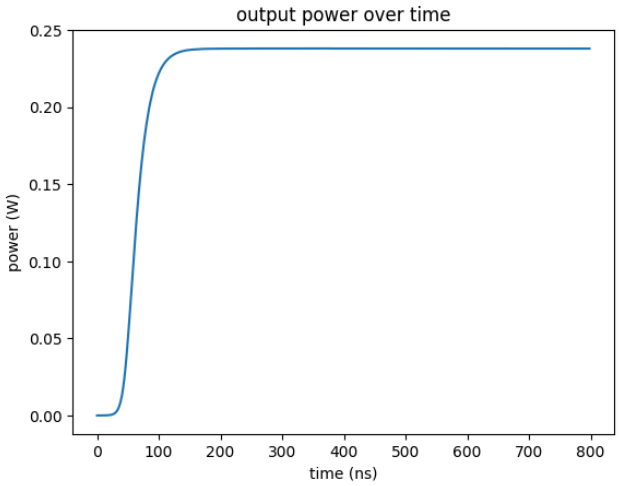
Question: Estimate the order of magnitude of the cavity power, by multiplying the square of w_0 with the saturation intensity $120\frac{W}{cm^2}$. The output laser power will be roughly then *(1 - R)* of this power.

Task: Simulate 100 bounces with the aperture radius (use a circular aperture) approximately set to the width calculated above. Set **N_x, N_y =** 0, so that the aperture is a simple circular hole. Give the width **a factor of 2 or 3**, since w_0 will be smaller than the width of the beam away from the minimum, at the aperture. The simulation grid should have 256 samples in each axis, and should be 10x larger than the aperture width so that significant error does not build up *over the course of many iterations. Plot the resulting intensity pattern in 2d. I also recommend plotting a 1d cross section of the intensity. You should get a Gaussian form.

*we could do more clever things like absorptive boundary conditions and get a smaller grid dimension, however for brevity we will brute force it with a large simulation region relative to the beam size.

If you are in MATLAB, you need to set your above determined simulation region size *D* parameter in multiple files:
beam_widths_with_40cm_lens.m
propagate_roundtrip.m
sim_aperture.m
Aswell as main.mlx

Task: As you iterate the bounces, save the output power*, (1 - R)* of the beam power, in an array. Plot the output power versus time. If your simulation is successful, it should grow until being approximately the power you estimated by hand above, then becoming stable in time. **You are probably wasting your time if this step does not check out for your simulation, and you move on. Make sure everything above works.**

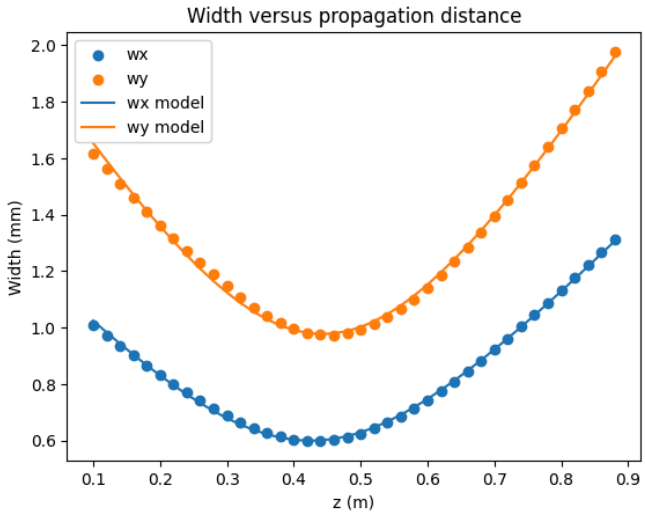


Task: Implement **beam_widths_with_40cm_lens(coordinates, ampl, plot = False)** this function should Fresnel propagate an input amplitude and coordinates after being passed through an *f = 40cm* thin lens over some set of distances out to *90cm*. The returned three arrays are the distances of propagation, and the *x* and *y* widths of the beam at each propagation distance. We will fit these propagation distances, and transverse widths to the M^2 model further. The *40cm* lens is useful as the beam width evolution would not be amenable to M^2 model curve fitting otherwise. To fit M^2 models efficiently a focal waist should be resolved in the width profile measured.

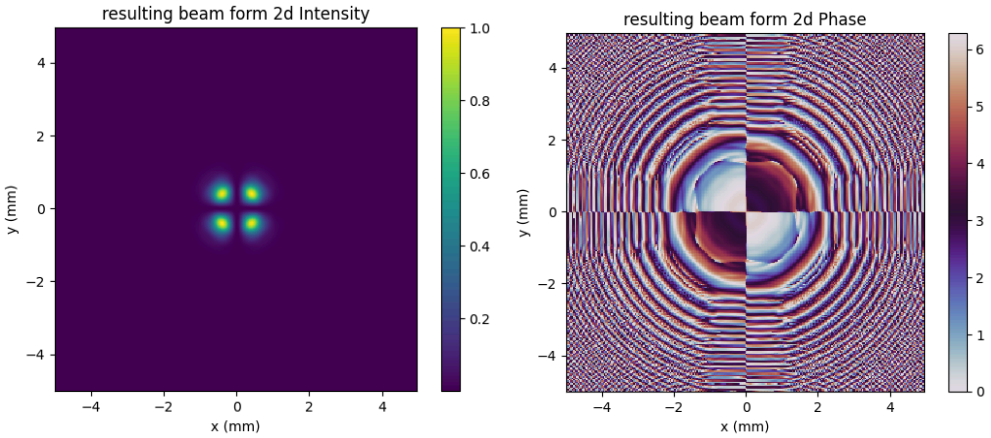
If **plot** = True, plot each propagated intensity plane. Be wary of the failure of the Fresnel approximation, do not propagate too short a distance. This implementation should be extremely similar to that of CHW7, and should use **beam_parameters_2d**.

Task: implement **fit_widths_data(zs, wxs, wys)** which takes the propagation distances and transverse widths, and fits two M^2 model parameter sets (one for each transverse axis), which should be returned. This should be similar to the implementation of CHW7.

Task: Produce plots of the 2d beam intensity and phase for the nearly Gaussian mode of your cavity. Use the above two functions to calculate the M^2 (should be nearly 1.0) of the beam in both transverse axes. Produce fitting plots like those of CHW7:



Task: Introduce a **spider_aperture** or **grid_aperture** (arm thickness of *0.1mm* works for me) to your simulation, ie **N_x, N_y** not 0, or **N_arm** not 0. These arms will block out lower order modes. You should also have to increase the aperture radius significantly to accommodate the higher order modes. Here is an example beam I generated with a crosshair shaped blocking arms aperture (**N_arm** = 4, or **N_x, N_y = 1**).



Produce a beam similar to the above. Also produce **two other different beams**. Plot their intensity and phase in 2d. Calculate their M^2 model parameter sets and display them. You will know you created a stable cavity mode if the power increases at first, then stabilizes, facing only relatively small oscillations. The phase plot also indicates a clean result. Generally it is expected that the phase will alternate between “lobes” (isolated high intensity regions) of a given beam as can be seen in the right plot.

You should end up with three sets of plots. One set should have phase and intensity that is as above. Don't forget the M^2 fitting plots.