**Generate Population**

- ***What does your population look like?***

A population of solutions is generated by the create\_population() function.

Each column in the population matrix corresponds to a single “solution”, “member”, “entity” or “chromosome” in the algorithm.

Let N represent the number of members, solutions, individuals, chromosomes, or population size.

Let M represent the number of genes that make up a single individual.

It follows that the structure used to represent our population is an M by N matrix.

The code below is used to generate population.

function solutions = create\_population(phase\_range, no\_genes, no\_members)  
 solutions = phase\_range\*(rand(no\_genes, no\_members)-0.5);  
end

**Solutions**

1. ***What are your chromosomes?***
2. ***What are your genes?***
3. ***How many parameters are being optimized?***
4. ***How are your solutions represented in the code?***
5. ***What is in your vectors?***
6. ***What do profiles represent?***

In our LaserPropagation function we pass the phase coefficient as a parameter.

Each coefficient is multiplied by the variable x.

The variable x is a space variable on the SLM (512 pixels for 4mm).

The product of the coefficient and the space variable is a single term in our polynomial phase.

Each term is a **gene**2.

Together, these genes create a single potential solution called a **chromosome1.**

Each column in the population matrix makes up a vector.

Each **vector**4 corresponds to a single “solution”, “member”, “entity” or “chromosome” in the algorithm.

**Each element in the vector5** is the coefficient of the corresponding space variable.

**N parameters are being optimized3** where N represents the degree of our polynomial phase

Note\*\*

As the number of coefficients (degree) of polynomial increases, the phase becomes more complex.

This gives our function’s shape more freedom.

**Profiles6** are the planes of our light beam that we are observing, evaluating, and optimizing.

In our LaserPropagation function, I1,I2, and I3 represent the 3 planes we are optimizing.



**Experimental Setup**

***- How many generations are ran?***

***- What stopping criteria for the GA is being used?***

The stall generation limit is number of consecutive generations without increased performance before stalling (stopping) the evolutionary loop.

It suggests that the population may have reached a plateau.

Either the solution cannot be further optimized, or population is stuck in a local optimum.

(optimal solution within a neighboring set of candidates)

***- What is the minimum fitness allowed?***

*0 is the minimum fitness value and most ideal candidate*

*0 is assigned to a chromosome whose has 3 identical profiles*

***- What is the max fitness?***

*100 is the max fitness value.*

*It is assigned to solutions whose intensity is too small.*

*A “too small” intensity is defined to be less than 0.2*

***- What is the best fitness?***

*Best fitness is minimum fitness value; 0*

***- What measures to report?***

**Default GA Values**

**Stopping criteria** –

*Stall generation limit* - number of consecutive generations without increased performance.

**Mutation –**

function mutants = mutate\_population(solutions, phase\_range, no\_genes)  
 n = size(solutions,2)/2; % number of output mutants  
 mutants = solutions(:,1:n) + phase\_range\*(rand(no\_genes,n)-0.5);  
end

**Selection method** –

**Fitness function** -

## Evaluate fitness of specified members

% returns 1 by n matrix with grades of each member  
 % where n represents the number of members  
function grades = fitnesses(members, no\_genes, phase\_range, pad\_size, technique)  
 no\_members = size(members,2);  
 grades = zeros(1, no\_members);  
  
 for j = 1:no\_members  
 grades(j) = fitness(members(:,j), pad\_size, technique, no\_genes, phase\_range);  
 end  
end

## Define convergence as similarity between profiles

function grade = fitness(solution, pad\_size, phase\_range, no\_genes, technique)  
 [I1,I2,I3] = LaserPropagation(solution, pad\_size, technique);  
 MinOfMax=0.2; % Minimum value allowed for intensity peaks  
  
 F12=sum(abs(I1-I2));  
 F13=sum(abs(I1-I3));  
 F23=sum(abs(I2-I3));  
  
 if (max(I1) < MinOfMax) || (max(I2) < MinOfMax) || (max(I3) < MinOfMax)  
 grade=1e2; % if max intensity too small, assign new value  
 else  
 grade=(F12+F13+F23)/3;  
 end  
end