

## APPENDICES

## APPENDIX A

### Evolutionary Algorithm

#### A.1 Evolutionary Algorithm

This appendix details the Evolutionary Algorithm (EA) used in this thesis work. A broad description of Evolutionary Algorithms in general as well as this algorithm was given in Chapter 2. EAs are a generic problem solving tool, and are not optimal for all problems [62, 63, 103]. In fact, many simple problems are easily solved either analytically or with methods such as steepest descent. EAs are better suited for problems with a complex, diverse parameter space. Nevertheless, the algorithm should be tweaked and tailored to each problem being tackled. The use of adaptive operators and operator constants has made algorithms a bit more general. However, it is difficult to say which algorithm is optimal a given problem.

The specific algorithm used here was arrived at more by convenience rather than by design, and is based on the code written by Erik Zeek. It is quite general, using only one operator to perturb the solutions as the parameter space is explored, and it adapts the amount of perturbation as the optimization progresses. This allowed us to use the same algorithm easily in both atomic and molecular systems with

very different algorithm requirements. The bottom line is the algorithm works well and rapidly finds an optimal solution. Primarily for that reason, the simple version of an EA used in this work was never "upgraded" with other operators. I have observed that the algorithm can act as a "local hill climber" towards the end of an optimization. In my work, I did not employ cross-over, but I suspect that the cross-over operator will help to add diversity to the population and might explore parameter space a bit more than the current algorithm. Other more complicated problems may require the modification of this algorithm, but for the purposes of this thesis, the algorithm worked well, and as they say, "If it ain't broke, don't fix it."

Below I have the details of the EA. I start with pseudocode, then describe the pseudocode, then briefly discuss the Gaussian noise function.

## A.2 Pseudocode

### Definitions:

$j$  is a test solution, integer ( $j = \{0, 1, \dots, M\}$ )

$M$ =number of individuals in a population

$k$  is a control knob number, integer ( $k = \{1, 2, \dots, N\}$ )

$N$ =the number of control knobs

$V_k$  =solution variable (pad voltage on pulse shaper  $\rightarrow$  spectral phase control)

$\mathbf{V} = \{V_1, V_2, \dots, V_N\}$

$p_j$  = individual solution =  $\{V_{j,1}, V_{j,2}, \dots, V_{j,N}\}$

$F_j$  is a function that evaluates solutions (spanning  $\mathbf{V}$ ) and is maximized by an "optimal" solution

$\mu$  = parents

$\lambda$  = offspring

$M = \mu + \lambda$

$\mathbf{P}$  = population =  $\{p_1, p_2, \dots, p_M\}$

$R$  = uniform random number generator

$G$  = Gaussian white noise generator

$r$  = standard deviation ratio

**Initialize:**

Set an initial standard deviation  $\sigma_{init}$  (we typically use  $\sim 500$  and the  $\max(V) = 2^{16} - 1$ ) and choose the number of members of parents ( $\mu$ ) and offspring ( $\lambda$ ). We typically use  $\mu = 20$ ,  $\lambda = 100$ . Note a standard deviation is attached to each member of the population set. Chose a standard deviation ratio  $r = 20$ .

Start with a set of random voltages for pulse shaping:  $p_j = \{R * \max(V), \dots\}$ , for  $j = 1 \dots M$

**Loop:**

Test the population of solutions and generate harmonic spectra from each solution  $H_j$

Evaluate each harmonic spectrum  $H_j$  with the fitness function  $F_j$  Here there are a couple of options. The most successful have been:

- (a) pick a harmonic and maximize its brightness
- (b) maximize the energy of a given harmonic
- (c) maximize the brightness difference between a harmonic and the average of

neighboring peaks

Sort the solutions  $p_j$  according to the fitness value  $F_j$ . Keep the  $\mu$  top solutions (largest fitness values) and discard the others—these are the new parents.

Make  $\lambda/\mu$  copies of the new parents (I set  $\lambda/\mu = \text{integer}$ ).

You now have a population  $\mathbf{P}$  with  $M = \mu + \lambda$  members

Take each member of the offspring population  $\lambda$  and generate a new standard deviation then use that to perturb (mutate) the solutions:

*for*  $j = 1 : \lambda$

$$\sigma_{new} = \sigma + G\left(\frac{\sigma}{r}\right)$$

Then for that population member, calculate

*for*  $k = 1 : N$

$$V_k = V_k + G(\sigma_{new})$$

*end*

*end*

The mutated solution now forms a new population set (the unmutated parents and the mutated children); the loop is repeated until the increase in the fitness value saturates.

### A.3 Description of the pseudocode

In my program, I track a set of information about a given solution:  $\{p_j, \sigma_j, F_j, H_j\}$ . For the initial step, the  $\sigma_{init}$  is unused, however, it is used in the loop to adaptively modify the standard deviation. The idea is that as the solution approaches an optimal one, we want to perturb the solutions less so that the changes in pulse shape

are smaller and to allow the algorithm to converge better for narrow fitness peaks.

In the initialization stage, parents and children (as defined in Chapter 2) are irrelevant. They simply define the number of solutions (pad voltages for the mirror) that will be generated by the uniform white noise generator (to allow complete random coverage of the parameter space). That is, we take a population with  $M$  solutions and generate pad voltages with a random number scaled from 0 to 1, then multiply by the maximum voltage to distribute the random solutions all over the pad voltage parameter space. This set of ( $M$ ) randomly distributed voltages controls the spectral phase of the input pulse. However, this is uniform random noise, not a Gaussian probability density function.

The next step involves the evaluation of the fitness function. This is highly system dependent, as well as dependent on the final goal. For the HHG control experiments, this involved measuring the HHG spectrum. The spectrum was then divided into sub-arrays ( $s_j$ ) that each contains one harmonic. Then operations can be performed on the harmonic arrays individually to affect each harmonic separately and the results of those operations can be combined to produce relative changes in the harmonic spectrum. In the case of the molecules, a similar approach was taken, but in that case the spectrum of interest was the probe spectrum and the sub-arrays corresponded to locations of vibrational scatter.

The next step is the all-important solution perturbation (or mutation). The children for the next generation are formed from mutated copies of the new parents (as selected above). For example, if we have 5 times the number of children as parents,

we will now make 5 copies of each of the parents selected as described above. This set of children will then be evaluated as follows:

(a) we will loop through  $\lambda$  children (note the error on the previous psuedocode document)

(b) for each child, the standard deviation of the parent from which it was copied is contained in it's set of information  $\{p_j, \sigma_j, F_j, H_j\}$  and a new standard deviation is computed from:

$$\sigma_{new} = \sigma + G\left(\frac{\sigma}{r}\right)$$

so the set for that member of the population of children is now:

$$\{p_j, \sigma_{new}, F_j, H_j\}$$

the new standard deviation is used to mutate the pad voltages of this particular child:

*for*  $k = 1 : N$

$$V_k = V_k + G(\sigma_{new})$$

*end*

The mutated voltages give a new trial solutions  $p_{new} \longrightarrow \{p_{new}, \sigma_{new}, X, X\}$ , where  $X$  to indicates the harmonic spectrum and fitness value no longer applies to the new trial solution. This information will be gathered in the step after the mutations of all of the children are completed.

(c) step b is repeated for all of the children created for the next generation

(d) The next generation population has then been completed and consists of the parents (the top  $\mu$  members of the previous generation, unmutated) and children for

the next generation (mutated [with a new standard deviation] copies of the top  $\mu$  members of the previous generation)

(e) Use the new population set (the new generation) the pad voltages on the deformable mirror one-by-one and then measure the generated harmonic spectrum by downloading the information from the ccd camera. Then the fitness values are computed and the process is repeated.

The role of standard deviation ratio,  $r$ , is to help convergence later in the running of the algorithm. Assuming you have evaluated the spectra to get a fitness value, the solutions are then sorted by the fitness value from largest to smallest (larger fitness values mean better solutions). This is when the concepts of children and parents come into play. For the current generation, we just have a population set. This population set is evaluated by the fitness value to choose parents and then consequently create children for the next generation. The number of parents is  $\mu$ , so the solutions (pad voltages) with the  $\mu$  highest fitness values are selected from the current generation. These are now referred to as the parents for the next generation ("new parents"). The remainder of the current generation solutions are discarded.

Note: The new population for the next generation is constructed from the solutions whose harmonic spectra evaluated with the fitness function have the  $\mu$  largest fitness values in the current generation.  $\lambda$  identical copies are made and modified versions of those constitute the children for the next generation. When I referred to the "new parents", I mean the parents for the next generation, which are the top  $\mu$  fitness values from the current generation.

## A.4 Mutation operator

The mutation operator uses a Gaussian noise generator. This perturbs the solutions, but limits the size of most perturbations to a value of roughly  $\sigma$ . It utilizes the probability density function:

where  $\sigma$  is the root of the second moment (standard deviation).

and  $E\{\}$  is the expected value defined as

$$E(x) = \int x f(x) dx$$

The first moment (expected mean value)  $\mu = E\{x\} = 0$

$$\text{and } \sigma = \sqrt{E\{(x - \mu)^2\}}$$

Note that I utilize two different random number generators. In the initialization step, I use the uniform white noise generator. This provides numbers scaled from 0 to 1.0 uniformly. That is, if we ran the generator for a long time, the histogram would have an equal distribution of "hits" for number in the range of 0 to 1. Once the main loop is entered, I switch to a colored noise. That is the histogram will not have equally probability for generating all numbers. The histogram will have a Gaussian profile with a certain standard deviation  $\sigma$ .