# Auto focus for electrons

Majd Salman

June 28, 2011

LUND UNIVERSITY

Degree project (Bachelor of science), 15 credits

Division of Synchrotron Radiation Research

Department of Physics

Supervisor: Erik P. Månsson

# Contents

# 1 Introduction

This thesis is the result of the diploma work, for a bachelors degree in physics, done with the gas phase group at the division of synchrotron radiation research at Lund university. The purpose of this work is to develop a set of tools that can be used when working with a 3D momentum imaging spectrometer.

The spectrometer is, among other things, used in experiments with attosecond pulses of light, to resolve the energies of electrons released when a target gas is ionized by the light pulses. Because the source of the electrons is extended in space, there is a need to have an electrostatic lens-system that can focus the source (electrons) and create an image on a detector. This electrostatic lens-system requires choosing proper parameters for a set of voltage differences between electrodes in the spectrometer, to create a lensing effect to resolve different energies. The number of parameters can in theory be quite large.

The thesis is partly creating tools to run simulations of the spectrometer and collecting and visualizing the data. This in itself can be used to find proper parameter values to use when running real experiments. But the tools can also be used to find relationships between the parameters.

Another part of the work is to investigate and create tools for automatically finding parameter values that optimally resolve a set of electron energies. This brings about many challenges because of the complexity involved in dealing with many parameter optimization. Genetic algorithms, an optimization technique that draws inspiration from biological evolution is presented as a possible solution to the issue of complexity. In the end two tools, in the form of computer programs with a graphical user interface, are presented and their performance is demonstrated.

## 1.1 High Harmonic Generation and attosecond pulses

High harmonic generation (HHG) is a process where short pulses of light are produced when focusing a bright, ultra-short wavelength, laser on a noble gas target [2]. This produces a comb of odd harmonics in the extreme ultraviolet range (XUV). The high harmonics form a plateau, known as the harmonic plateau, where the intensity is comparable over a wide range of harmonics. After this the intensity drops off exponentially, something known as the harmonic cutoff. Figure 1 shows such a comb of odd harmonics.

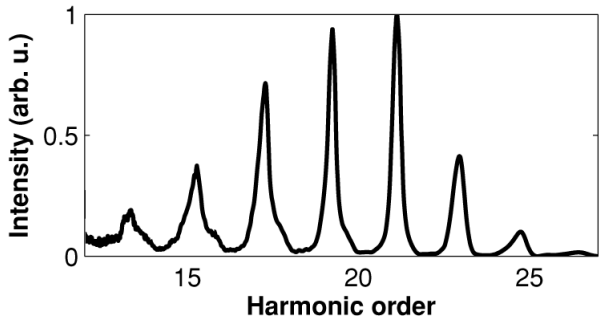


Figure 1: Spectrum of odd high-order harmonics [6]

The process of generating harmonics can be explained as a three step process, which is illustrated schematically in figure 2. Because of the intensive laser field, the electric component of the field can result in an electron tunneling through the Coulomb potential of the atom. The electron then becomes free and can be accelerated by the laser field. When the field changes sign, the electron can be accelerated back and recombine with the atom, emitting a XUV photon.

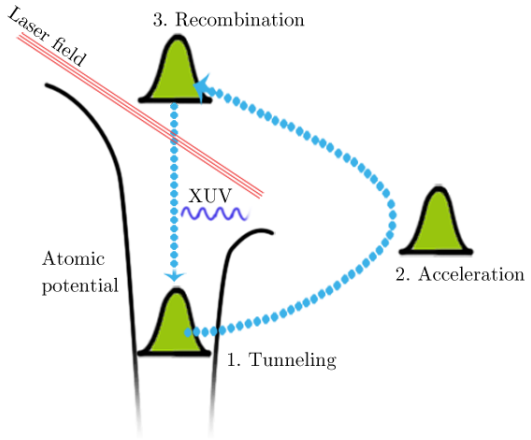


Figure 2: High harmonic generation explained in three steps [8]

One XUV photon can be emitted for every turn of the oscillating electric field of the driving laser and two photons for every optical cycle. The driving laser has a pulse duration in the femtosecond range. This leads to a train of many attosecond pulses being created during a single pulse of the driving laser. The attosecond pulses are called thus because the duration of a XUV pulse can be lower than 100 as. Attosecond pulses are interesting because they are the

shortest pulses ever created. They are within the time scale of all chemical reactions, even that of an electron orbiting its atom.

By combining the driving IR-laser and with the XUV pulses one can perform two-photon experiments. In these experiments, where odd harmonics are combined with the driving laser, one can reach even harmonics, also called side bands. Figure 3 shows a schematic energy level diagram of two possible paths that, in a two-photon process, lead to such a side band.

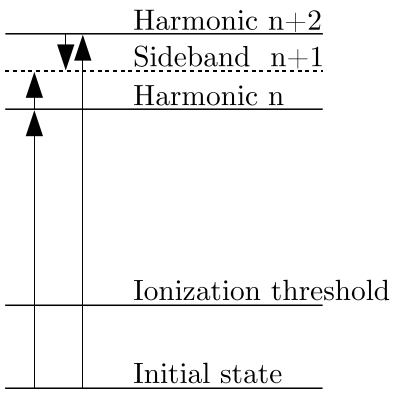


Figure 3: Two paths that lead to same side band. Small arrow corresponds to IR photon.

The Heisenberg uncertainty principle, $\Delta E \Delta t \gtrsim h$, and the duration of the attosecond pulse train, 10 fs, gives an uncertainty in energy of around 0.4 eV. This means that we can't get better energy resolution than about 0.4 eV with such light. But it is enough to be able to tell the harmonics apart. This can be achieved by varying the delay between the driving laser and the harmonics with technique is known as RABITT (reconstruction of attosecond harmonic beating by inference of two-photon transitions).

## 1.2 Energies

In the setup described later, the harmonics used to perform experiments are generated using an 800 nm infrared laser. There is a constant energy difference of 3.10 eV between each odd harmonic order. In experiments where the odd harmonics are combined with light from the laser, even harmonics show up, with 3.1 eV/2 = 1.55 eV, which is the energy of an 800 nm photon. This explains why we, in the simulations later, see lists of energies that are separated by 1.55 eV. Depending on what gas is being investigated, the lists of electron energies will have different start values, but the increment is always 1.55 eV.

We will mostly look at helium which has an ionization threshold of 24.59 eV. To free an electron from helium, the energy corresponding to sideband 16 is required, because it is the first harmonic that has an energy higher than the ionization threshold: $1.55\text{ eV} \cdot 16 - 24.59\text{ eV} = 0.21\text{ eV}$.

## 1.3 The spectrometer

The 3D momentum imaging spectrometer is used to obtain full 3D information about a disassociation or or ionization event. A model of the spectrometer is shown in figure 4. The spectrometer consists of multiple hollow electrodes. Gas can be injected into an interaction region, and is excited and ionized by attosecond pulses. The interaction region is 30 mm wide and the detector has a diameter of 80 mm. There are two sides on spectrometer; one 2D and one 3D imaging part. The 2D side can measure the position of the impact of charged particles with the help of a microchannel plate (MCP), a phosphor screen and a CCD camera. When a charged particle impacts the MCP the incident charge is amplified, resulting in a shower of particles hitting the phosphor screen. The screen emits photons that are recorded with a CCD camera.

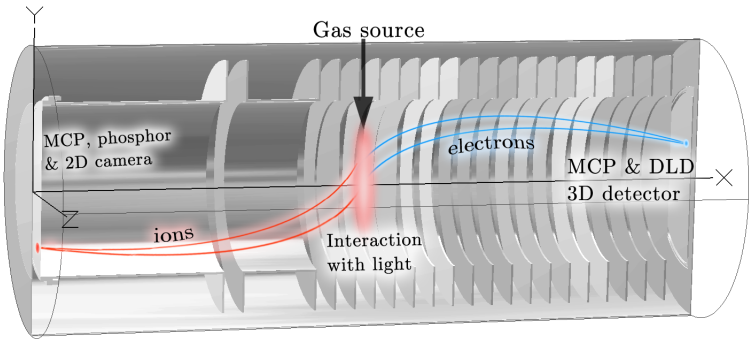


Figure 4: Cross section of a 3D model of the spectrometer

The 3D part contains a similar setup but is capable of recording 3D information from an event. This is done with an MCP in front of a delay line detector (DLD). The MCP amplifies an incident charged particle and the DLD is used to measure the position, where the particle hit the MCP. The DLD consists of two perpendicular layers of wires, as shown in figure 5. When the amplified signal reaches the wires a current can be measured. Timers connected to both ends of a wire can measure the time it takes the current to travel along the wire. The difference between these times is then a measure of the position of the event along one of the measurement axes. There are a total of four timers, two for each axis, as shown in figure 5.

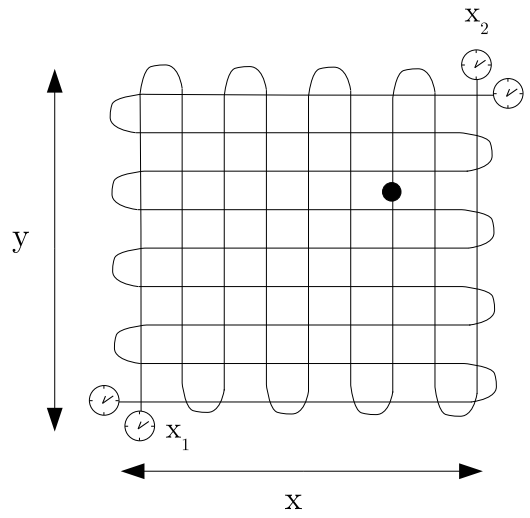


Figure 5: Schematic image of a delay line detector. In a real DLD the wires are wound tightly around insulating rods in two layers per orientation

The radial distance from the center of the detector can be stated as a time

$$t_x = t_{x_1} - t_{x_2}$$

$$t_r = \sqrt{t_x^2 + t_y^2}$$

where $t_r$ is the radial distance from the center of the detector. One can convert this to a distance by a conversion factor of around 2 ns/mm

$$t_r = 2 \text{ ns/mm} \cdot r$$

The mean value of the time that two timers, belonging to a set of wires, measure is constant for any position on the DLD. This time correspond to the time of flight of the particle

$$TOF = \frac{t_{x_1} + t_{x_2}}{2}$$

There are 1000 pulses/s from the laser. A photo diode can be used to measure when a pulse occurs and start a timer. This can be used in conjunction with the DLD to measure the time of flight of the charged particles that are created when the laser ionizes a gas target. The delay between the time at which the pulses ionize the target gas and the light hitting the photo diode is not negligible, but it is at least constant, meaning that the ionization happens at a well defined moment in time.

### 1.3.1 Voltage definitions

Although the two sides of the spectrometer allow for measurement of electrons and ions simultaneously, we will concentrate on the electrons and the 3D side of the spectrometer. To simplify the work we will define and work with three main voltage differences that are used to extract the electrons from the interaction region and shape their paths towards the detector. These voltages along with some named electrodes are shown in figure 6.

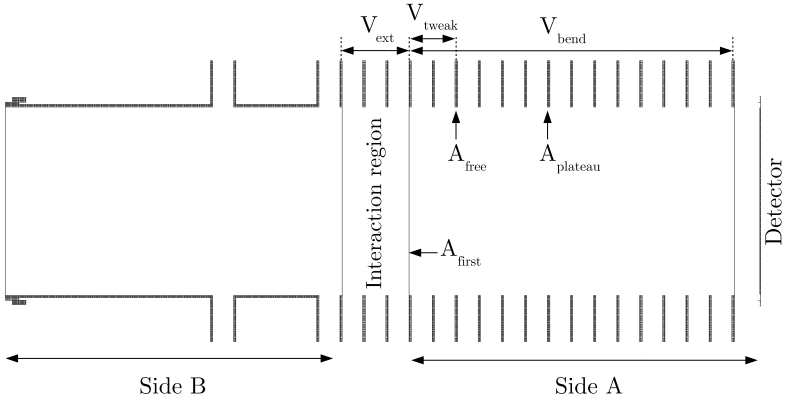


Figure 6: 2D model of the spectrometer with some voltage differences defined.

On either side of the interaction region there are metallic meshes, drawn as solid lines in figure 6. A voltage difference between the meshes gives rise to a homogeneous electric field that accelerates the electrons in the direction of the detector. This voltage difference will be referred to as the extraction voltage or $V_{ext}$.

There is also mesh in front of the MCP. The voltage difference between the mesh on the A-side of the interaction region (named $A_{first}$) and the end mesh is referred to as $V_{bend}$. A voltage can be applied between the end mesh and the front of the MCP to give the electrons enough kinetic energy to be detected.

A number of electrodes after $A_{first}$ can be shorted to create a potential plateau. The last electrode in such a shorted series is referred to as the plateau electrode or $A_{plateau}$. One of the electrodes between $A_{first}$ and $A_{plateau}$ can be controlled independently of the others. This electrode will be referred to as the free electrode or $A_{free}$. The voltage difference between $A_{first}$ and $A_{free}$ is referred to as $V_{tweak}$. In figure 6, the case where $A_{free} = 2$, and $A_{plateau} = 6$, has been marked. The values are a numbering of the electrodes starting from the mesh, $A_{first} = 0$ .

## 1.4 Focusing with electrostatic lenses

The gas to investigate is injected into the interaction region. This injection along with the intensity profile of the light source determines the shape of the source of electrons in the interaction region. Because the source is not point like there is a need for focusing. The aim of this is to focus all electrons that start of with some initial kinetic energy and direction on the same spot on the detector, independent of their initial starting position. This can be achieved with electrostatic lenses.

Figure 7 shows a simulation where 0.8 eV electrons are emitted from two source points in four different directions. The electrons are extracted by the potential difference $V_{ext}$, shown as a downhill slope in 7. An electrostatic lens, a so called Einzel lens is formed when when the free electrode has a potential ($V_{tweak}$) higher than the surrounding plateau. Electrons that move in the region of the lens will be bent towards the middle. After the plateau electrode $V_{bend}$ provides a constant potential decrease and additional bending along the rest of the flight path.

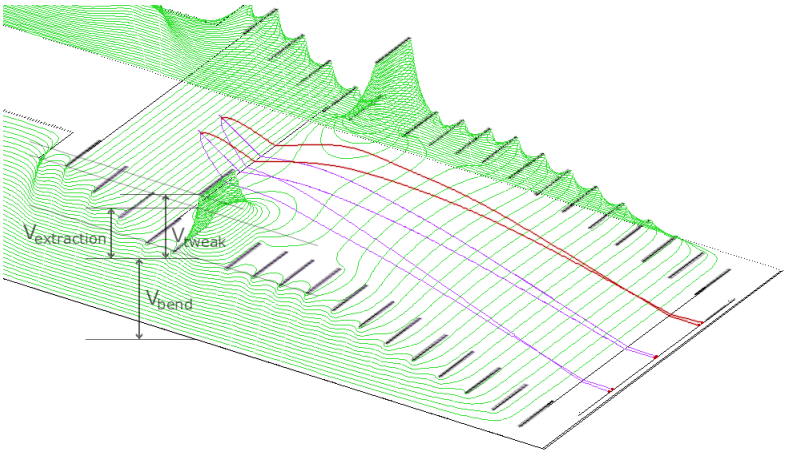


Figure 7: Example of lensing effect. The potential differences are shown as height. The potentials are $V_{ext} = 20$, $V_{tweak} = 25$, $V_{bend} = 25$ and the electrodes $A_{free} = 2$, $A_{plateau} = 6$. [image courtesy of Erik P. Månsson]

## 1.5 SIMION

SIMION is a commercially available charged particle optics simulation software. Given a model or geometry of electrodes to simulate and input data for potentials and particles, SIMION will solve the Laplace equation using a finite difference method. The electric field strength in the space between electrodes is first calculated and then the motion and trajectories of predefined particles are simulated in these fields.

SIMION is fully programmable using a scripted language called Lua (since version 8). The user can define events in a simulation which can be captured by SIMION and sent to a user based program. This allows for interaction between the user and the simulation. Particle definition, initiation of a simulation and calculations can all be made using Lua.

The geometry of a problem, along with precalculated potential array files and a user defined program is bundled into a SIMION workbench. SIMION can be run in either a graphical mode, with a graphical user interface (GUI), or via a command line interface. This allows for access to SIMION from either a command line or from other programming environments. A simulation can for example be launched from Matlab by calling a SIMION executable with some input flags and parameters such as what workbench to use and what Lua-file is to be run.

## 1.6 Simulations

A simulation with SIMION yields full information about initial and final positions and time of flight information. The number of source points, the shape of the source in the interaction region and the spacing of source point can be varied. Each source point acts as a source of electrons in the simulation. The number of directions in which the electrons can emanate from the source point can also be varied. The initial direction of an electron is measured as the elevation from the $x$-axis (shown in figure 4), where the an initial angle of $0^\circ$ means that the particle starts towards the detector. Because the spectrometer is cylindrically symmetrical, the motion of the electrons is only simulated in two spatial dimensions, $x$ and $y$.

One can also specify the shape of the source points, that is their positions within the interaction region. The default value is a set of points on a line on the $y$-axis. Other shapes include a set of source points in a diamond formation and randomly placed source points. The most realistic simulation case would probably be a random or Gaussian distribution of the source points. For the sake of simplicity, during all simulations, if nothing else is stated, the default linear shape will be used as a source point pattern. The number of source points is set to 11, the spacing between source points is 0.8 mm along the $y$-axis, and the number of initial directions per source point is 16. With this setup, 176 electrons are simulated for every energy.

Figure 8 shows the result a simulation, where the distance from the center of the detector (in ns), $t_r$, is plotted against the time of flight (TOF). Electrons with five different energies were simulated. The electrons were sent out from 11 different source points along the $y$-axis in the interaction region. From every source point electrons are emitted in 16 directions. Particles with higher initial kinetic energies will be bent off less than particles with lower energies. Electrons with higher energies will therefore hit the detector at a larger radius from the center. Figure 8 shows all energies in the same diagram. Every energy forms

an ellipse with groups corresponding to the 16 initial directions and 11 source points in each direction.

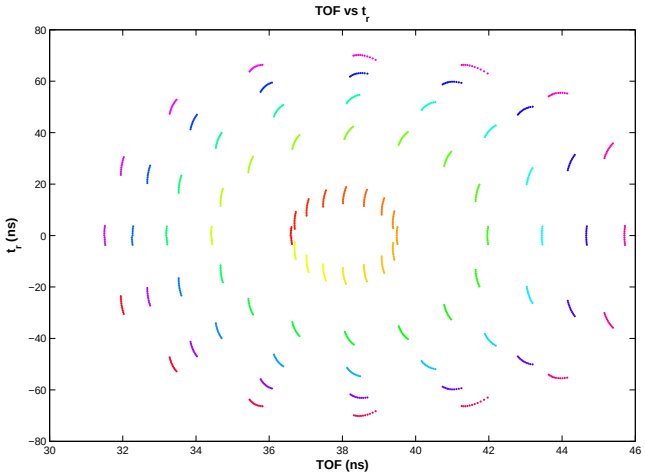


Figure 8: Simulation of electrons with energies 0.27 eV, 1.82 eV, 3.37 eV, 4.92 eV and 6.47 eV (Xenon). The size of the ellipses correspond to the energy of the particles, with smaller energies resulting in smaller ellipses.

# 2 Resolution

## 2.1 New resolution measure

To determine if a set of voltage parameters are better than some others, a measure of how "good" the results of a simulation are has to be developed. Depending on what one wants to study there are different ways to define useful measures of resolution. Increasing $V_{Bend}$, $V_{Tweak}$ or $V_{Ext}$ leads to a smaller image, and may allow higher energies to reach the detector. The time resolution of the detector is however limited. Therefore two energy-ellipses have to be separated by more than the intrinsic resolution of the detector to be considered separable.

The measure of energy resolution is based on reshaping the ellipses resulting from a simulation (see figure 8) into circles and doing a coordinate transform from Cartesian to polar coordinates. The energies belong to a sequence of energies based on the ionization threshold of Xenon.

For every energy in the simulation the center of the ellipse is found by identifying particles that started at 0° and 180° and extracting the TOF-information for the same. Particles starting with an initial velocity vector pointing away from

the detector (180° initial angle) will have the longest TOF, because they can travel some distance before being accelerated back towards the detector. Particles starting towards the detector (at 0°) have the shortest TOF. The mean between these two determines the center of the ellipse along the TOF-axis. The center is calculated for the ellipse of each energy. This gives better results when the ellipses are asymmetrical (when the TOF-distance between the ellipses are smaller on one side than the other). Each ellipse is then shifted so that the horizontal center is situated at TOF=0, see figure 9.

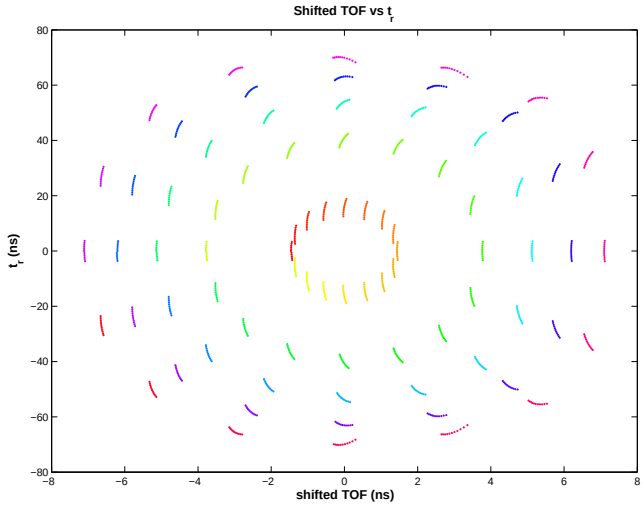


Figure 9: Shifted version of the plot in figure 8.

The width of the ellipse is taken to be the mean difference in TOF between particles starting at 0° and 180°. Similarly the height of the ellipse along the $t_r$-axis is found by identifying particles starting at ±90°. A measure of the eccentricity, $e$, of the ellipse is then the width divided by the height.

The coordinate transformation $(t, t_r) \to (\rho, \alpha)$ is then made by

$$\begin{aligned} \rho &= \sqrt{(e \cdot t_r)^2 + t^2} \\ \alpha &= 2 \arctan\left(\frac{e \cdot t_r}{t + \rho}\right) \end{aligned} \tag{1}$$

where $t$ is the shifted TOF. Figure 10 illustrates the transformation. For clarity figure 10 shows the result of a smaller simulation with only two energies. Figure 11 shows the result of the transformation for the same two energies.

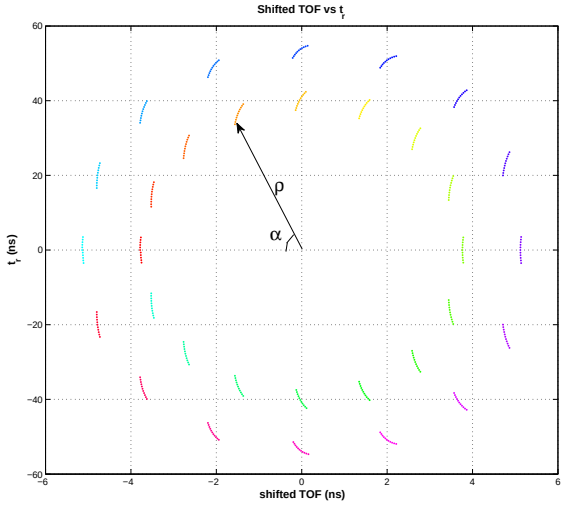


Figure 10: Simulation of electrons with energies 1.82 eV and 3.37 eV. The TOF axis is shifted and the transform $(t, t_r) \to (\rho, \alpha)$ is indicated.

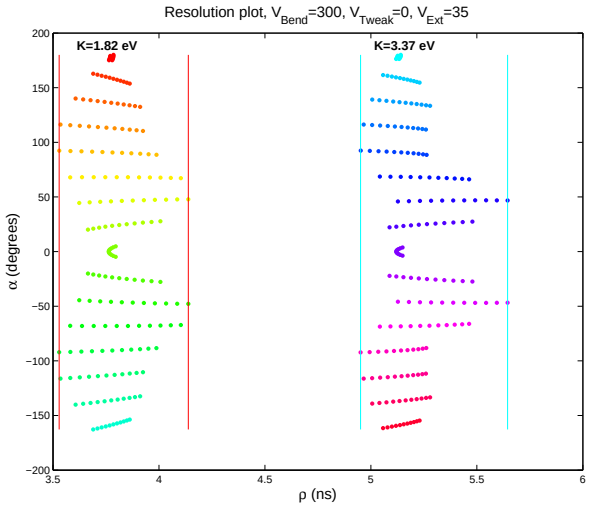


Figure 11: Result of the transformation $(t, t_r) \to (\rho, \alpha)$. Vertical lines indicate minimum and maximum values of $\rho$ within one energy. Colors are used to group particles with the same original angle and can be used to identify the same groups of particles in figure 10.

As can be seen in figure 11, groups of particles with the same energy are aligned in vertical columns in this coordinate system. A measure of the energy resolution is then the separation between the energies along the $\rho$-axis. If one could transform the data in figure 8 into perfect circles, the mean values within each element would form straight, vertical lines in $(\rho, \alpha)$-space. As can be seen in figure 11 this is not the case. The centers of the ellipses do not align, which leads to the curved shape of the particle groups in figure 11. One workaround would be to find a more complicated transformation that does not assume that the original data is perfectly circular. Another way is to find a measure that is less sensitive to this issue.

Figure 12 illustrates the definitions that are used in the derivation of the resolution.

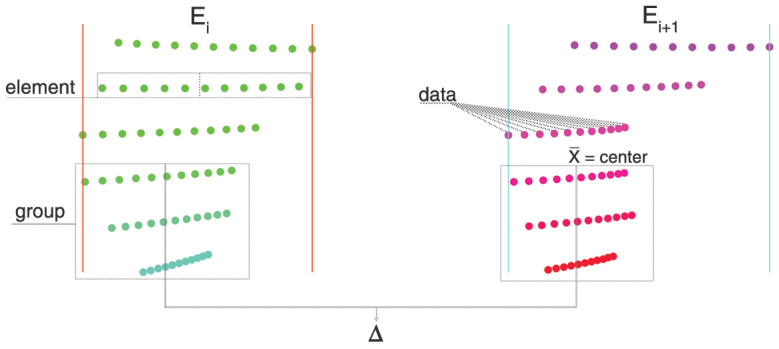


Figure 12: Definitions used in the resolution. Only 6 of 16 elements or directions are shown here for clarity

For every energy, $E_i$, we compare one group of three elements or "three-group" to the corresponding three-group in the closest energy above it. The standard deviation along the $\rho$-axis is calculated for the data within each element. To take into account the intrinsic resolution of the detector we define the standard deviation of one element to be

$$\sigma_{\text{element}} = \sqrt{\sigma_{\text{data}}^2 + \sigma_{\text{det}}^2} \tag{2}$$

where $\sigma_{\text{det}} = 0.5$ ns is the resolution of the detection system. The standard deviation of the whole three-group is then

$$\sigma_{\text{group}} = \sqrt{\sum \sigma_{\text{element}}^2} \tag{3}$$

The center of each three-group, $\bar{X}$, is the average position along the $\rho$-axis within the group. The resolution for one pair of three-groups is defined as

$$R_{\text{group}} = \frac{\sqrt{\sigma^2_{\text{group i}} + \sigma^2_{\text{group i+1}}}}{\left|\bar{X}_{\text{group i+1}} - \bar{X}_{\text{group i}}\right|} = \frac{\sqrt{\sigma^2_{\text{group i}} + \sigma^2_{\text{group i+1}}}}{\Delta} \tag{4}$$

Three-groups are formed by cyclically traversing all elements. If there are 16 different initial angles we will get 16 different values for the resolution. These are added together to form a total measure of resolution between two energies

$$R_{\text{tot}} = \frac{\sqrt{\sum R^2_{\text{group}}}}{\sqrt{N}} \tag{5}$$

where N is the number of initial directions that are simulated. Applying this resolution measure on the simulation results illustrated by figure 10 and 11 we get figure 13. In figure 13 the resolution is calculated and overlayed between the energies so the variation in the resolution between different three-groups becomes apparent.

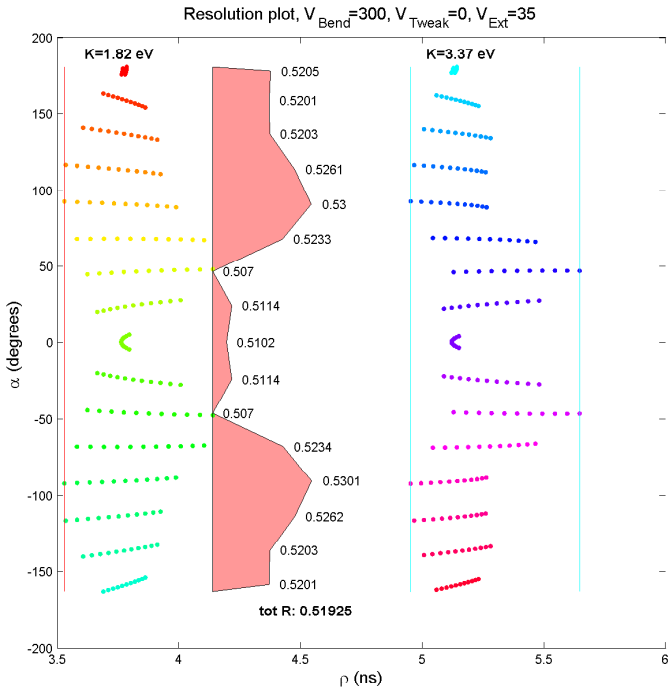


Figure 13: Plot in $(\rho, \alpha)$-space of a run with two energies, 1.82 eV and 3.37 eV. Resolution plot is overlayed.

A large value of $\Delta$ is desirable in equation 4 since it indicates more separated three-groups. This means that a lower value for the total resolution, as defined by equation 5, is better.

### 2.2 Old resolution measure

Previously another measure for resolution was used. It was based on calculating the relative TOF-resolution and the relative radial resolution and combining them to form a measure of total resolution. Due to division by (almost) zero for radial resolution at $\alpha = 0°$ and $180°$, and for temporal resolution at $\alpha = \pm 90°$, ad hoc tricks and thresholds had to be used to combine them into some value for total resolution. It was felt that this value did not accurately agree with a human judgment of total resolution. Although the separation into temporal and radial parts give important insights into the optimization problem, there was a wish for a total energy resolution quantity that the program could evaluate, to be used for automatic optimization.

## 3 Simulation and visualization

Infrastructure for running SIMION from Matlab, via the command line interface, was already in place when this work began. A typical simulation begins by the user specifying what type of particles, which energies, how many source points and directions to use and then initiating a SIMION process with these parameters. A simple simulation with typical input values consists of five energies, 11 source points and 16 directions for every source point. It takes a couple of seconds to run and read the output from such a simulation.

It was realized early on in this work that the simulation time would be a bottleneck when doing larger scale simulations by varying parameter values. The first part of the thesis is therefore focused on dealing with this problem and later on building tools to run large scale simulations and visualizing them.

### 3.1 Improved simulation speed

The existing Matlab script could be used to define and run a single simulation. The script launched SIMION with a dos command from Matlab. The workbench file containing the geometry of the spectrometer was passed as an input parameter along with the energies, potentials and other command line parameters. The workbench includes information about which Lua-script to run to control the simulation and record data that is later stored as a file on disk. This output file can then be read and parsed by another Matlab script and the result is a so called “Flys” structure that contains all relevant information about the settings and the results of a simulation.

But why stop at running only one simulation at a time? Running SIMION from the command line is not a very heavy task for most modern computers; especially computers equipped with multiple processing cores. A program was therefore developed in the C# programming language. The program was only responsible for launching multiple SIMION simulations and collecting the output data into a single file that could later be read by the existing Matlab scripts.

Given a set of ranges for the parameters $V_{Bend}$, $V_{Tweak}$, $V_{Ext}$, the program creates input strings for the command line interface of SIMION. It then launches multiple processes (around 50 at a time), on multiple threads on the machine. In some cases one is interested in varying the parameters in small steps, something which could result in thousands of simulations. In order not to stall the computer the program limits the number of threads that can be active at any time. When a thread is available a new simulation is launched. This continues until all combinations of parameters have been simulated.

The C# program was deployed as a dll (dynamically linked library). This is a library of compiled external functions that can be accessed from other programs. This dll can therefore be used from within a Matlab interface or script to do large scale simulations. The usage of the dll in conjunction with a Matlab script along with program and data flow is illustrated in figure 14.

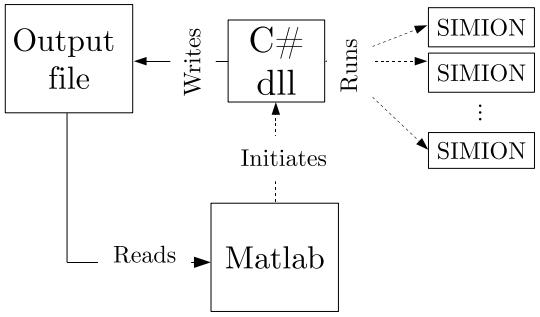


Figure 14: Crossover between two parents to create two offspring.

## 3.2 Development of tools

A tool was created to launch and visualize simulations of the spectrometer. It consists of a graphical user interface built in Matlab. In the simulation part of the user interface one can set what ranges for $V_{Bend}$, $V_{Tweak}$ and $V_{Ext}$ to simulate. The GUI also handles lists of energies to simulate as well as values for $A_{free}$, $A_{plateau}$, source point spacing and pattern. When the simulation parameters have been set they are sent to the external library, described in the previous section, that handles the execution of the simulations and the collection of data.

Within the same GUI the result of a simulation can be analyzed and visualized. The tool supports plotting of 2d line-diagrams, where two voltage parameters are fixed and one is varying. If a simulation has been made where all parameters are varied, the tool lets the user choose fixed values for the parameters for two of the parameters so the dependance of the resolution on the third parameter can be plotted. Alternatively the tool can choose the best values for the voltage parameters that should be fixed. The user can also scale values by the kinetic energy to look for dependencies on these.

The user can define which of the resolution measures to use for determining the best parameters. All information about a simulation is contained within a "simulation" class. This allows the resolution information to be precalculated and stored for fast access at a later time.

The visualization tool also supports plotting surfaces, where two voltage parameters are varied and one is fixed. The third parameter is fixed in a similar fashion as for the 2D plotting.

Finally the simulation data and settings can be saved and loaded for later use.

# 4 Optimization

This section details the second part of the thesis, namely finding optimal parameter values to resolve a given set of energies with our spectrometer. In some ways this is the "Auto focus" part of the project. A comparison will be made between classical and non-classical optimization techniques to solve this problem.

In this part $V_{Bend}$, $V_{Tweak}$, $V_{Ext}$, $A_{free}$ and $A_{plateau}$, that were defined in section 1.3, will be treated as parameters or variables of an unknown function, the output of which we wish to minimize. Given these parameter values, the output of the function is one of the resolution measures presented in section 2. The parameters are often bounded to some physically relevant interval to reduce the complexity of optimization. The set of all bounded parameters will be referred to as the parameter space. Every point in this space is referred to as a possible solution.

## 4.1 Classical optimization techniques

There are a number of classical optimization techniques that can be used to find the extrema of a function. Classical methods are often based on estimating derivatives or gradients of a function and are in some form deterministic and give reproducible results when run multiple times with the same input parameters.

Newtons method and gradient descent utilize the derivatives of a function and an initial guess to find a minimum. These methods are based on the function being differentiable, and the initial guess lying within a basin of attraction of the

function. In our case the function we wish to minimize is the resolution measure. This is for all intents and purposes a black box function that is not continuous and not easily differentiable all over the parameter space. As we will see later, if a set of parameters lead to particles missing the detector, the number of misses will be added to the resolution measure, to punish bad parameter values. This in conjunction with numerical noise from SIMION introduces discontinuities in function values, making it difficult to handle for classical techniques.

Direct search methods offer a solution to this problem because they do not require any information about the gradient of a function. A direct search algorithm starts with an initial guess for a solution, evaluates the function and the looks for nearby points to find a better solution. One can easily constrain a direct search method to search within a given range of parameter values for a solution.

The generalized pattern search algorithm (GPS) is a direct search algorithm for finding the global minimum of a function. It starts off with an initial guess for the parameters and evaluates the function. The algorithm then looks at nearby points on a mesh for better solutions. If a better solution is found within the neighbourhood of the point, the mesh size is reduced and the process starts over again. If there is no better solution within the nearest neighbours, the mesh is expanded and the neighbours are searched again. This continues until the mesh size at some point is below a threshold value.

## 4.2 Genetic Algorithms

There are many types of unconventional optimization techniques. These methods often draw inspiration from nature. One such group of methods is genetic algorithms (GA) which belongs to the larger group of evolutionary algorithms that are concerned with finding solutions to problems by mimicking natural evolution. The text by Mitchell [7] serves as an excellent introduction to the subject.

Genetic algorithm have been used in a wide range of fields as optimization and modeling tools. They have been used to predict weather, protein folding and evolving antennas for satellites [5]. More related to this thesis is the work by Garcia et. al who successfully used a genetic algorithm to evolve the geometry of a 2D velocity map imaging spectrometer [3].

### 4.2.1 Background

The Genetic algorithm is a method for finding global optima in some parameter space. A GA consists of a set of individuals, a population, that each represent the solution for a problem. Each individual consists of a *genome* that is an encoding of the solution to the problem at hand. In the case of minimizing a function each individual could for example be represented by some parameters

that are later evaluated by the function. An individual's genetic code makes up its *genotype*, while the expression of the genotype, in the form of solving a problem is called *phenotype*. The location of a gene within the genome is called a *locus*. For a GA to work, one should be able to quantify exactly, with a real number, how well an individual solves the problem relative to some other individual. Borrowing terminology from evolutionary biology, this number is called *fitness* and corresponds to the ability of an individual to propagate its genes throughout the population.

The idea behind GA is to initialize a random set of individuals, evaluate them and letting the most fit individuals reproduce and create offspring. In theory, iterating this process, results in more fit offspring for each successive generation of the population.

The space of all possible genotypes, when evaluated, is called a fitness landscape; a term first used by biologist Sewell Wright in 1932 [9]. In the case of a function with two parameters, the fitness landscape would be a surface, with the two parameters on two axes and the fitness on the third. In an $N$-dimensional problem space, the fitness landscape is an ($N$-1)-dimensional hypersurface.

### 4.2.2 Genetic operators

The difference between classical optimization techniques, such as Newton's algorithm or gradient descent, and GA is that the former tend to perform poorly when applied to a function that has many local extremes. Methods based on derivatives of a function tend to become stuck in these local extreme points and may come up with suboptimal solutions to a minimization problem. A GA can solve this by first sampling the solution space sufficiently well so that a wide range of solutions are present in the population. Genetic operators also act on the population in such a way as to increase the diversity of solutions, steering the GA away from local extreme points, and creating more fit individuals.

#### Selection

The selection operator serves the purpose of choosing individuals for reproduction. Generally one wants more fit individuals to have a higher probability of reproducing. Although some care has to be taken to keep some suboptimal solutions present in the population in order not to reach premature convergence, and lose potentially good genetic material.

There exists a plethora of different selection methods that are suitable in different circumstances. One of the more widely used methods is fitness-proportionate selection or "roulette wheel sampling", where the probability for an individual being selected to mate is proportional to its fitness. This can be likened to spinning a roulette wheel where each individual is given a slice of the wheel that corresponds in size to its fitness, hence the name.

#### Mutation

Like its counterpart in biology, the mutation operator introduces random errors in the genome of offspring after reproduction. This leads to a higher diversity in the genetic material of the population and decreases the chance of stagnation or getting stuck in local extreme points. The probability for a mutation occurring at a locus is usually set to a small value, around 0.05. In the limit where the probability for mutation goes to one, the GA becomes a random search of the fitness landscape.

#### Crossover

The crossover operator serves the purpose of mixing the genetic material of two parents to create two offspring. A locus is chosen randomly and the genetic material before and after the locus is exchanged between the two parents to create the offspring. Figure 15 illustrates this process for a single point crossover in a binary encoded genome. The probability for crossover occurring is usually quite large, around 0.95. If crossover does not occur, the parents that are chosen are directly copied to the new generation.

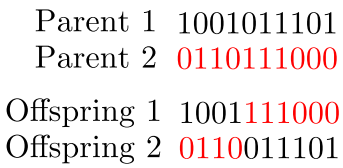


Figure 15: Crossover between two parents to create two offspring.

#### Elitism

Elitism is an operator acting on the whole population. The purpose of elitism is to retain a certain number of highly fit individuals in the population between two successive generations. These individuals are selected and copied to the new generation without undergoing crossover or mutation. This is way to make sure that good solutions are transferred to the next generation and are not lost to mutation or crossover. Elites can still be selected to produce offspring into the same generation that they are copied.

### 4.2.3 Encoding

Probably the most challenging part of implementing a GA is finding an suitable encoding for a problem. In our case the problem consists of finding five, real

valued parameter values, which when evaluated, are able to resolve a set of given energies. The simplest way would be to encode the genome an individual as a set of five real numbers: $\boxed{V_{Bend} \mid V_{Tweak} \mid V_{Ext} \mid A_{Free} \mid A_{Plateau}}$. Although such a simple implementation is appealing and viable, it has some drawbacks when applying the genetic operators; e.g. it would become quite convoluted to find a suitable mutation operator. One could add or subtract some arbitrary value to the selected locus. But the mutation operator should be able to bring about both subtle changes and larger ones, effectively sampling the parameter space.

More common in the field of GA is using a binary encoding, where the problem is translated into a binary code and the genome becomes a string of bits. One of the reasons for using binary encoding is purely historical, another is that the theoretical foundation of GA assumes a fixed-length, fixed-order binary encoding [7]. Although the choice of encoding highly depends on the problem at hand, there are studies that show better performance for binary encoding [1].

In our GA implementation we will use reflected binary encoding also known as a Gray code after its inventor Frank Gray [4]. The Gray code has some advantages and lends itself amenable to a GA implementation. Two successive integer values, encoded as a Gray code, differ in only one bit. In other words; the Hamming distance between two successive Gray encoded values is always one. Table 1 shows the integers 1-7 in binary and Gray code.

| number | Binary encoding | Gray encoding |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |

Table 1: Binary and Gray encoding of integers 1-7.

This property of the Gray code is advantageous because a mutation at one locus in the bit-string genome has a higher probability of shifting the represented value only a small amount. An individual that encodes a solution that is close to a global minimum should be able to take small steps in the fitness-landscape, making an incremental improvement. The Gray code makes this possible.

In some cases a single bit mutation results in larger steps. Looking at table 1, if the single 1 in the Gray encoding of "7" is flipped, the result is a string representing "0". These larger steps are also desirable to bring about larger change and keeping the population diverse.

To represent a sequence of integers $\left(0, 1, 2, ..., 2^N - 1\right)$ a binary string of length $N$ is required. The crossover operator is easier to implement if the bit-string genome has a fixed length. Therefore by our knowledge of the problem domain, we limit the possible parameter values to the range 0-2047, which is more than sufficient for all simulation cases. To encode a single parameter value in the range 0-2047, an 11-bit string is required. Since we have five parameters, the length of the genome is 55 bits. Table 2 shows the genome of an arbitrary individual encoded as integers and translated to a Gray code.

| $V_{Bend} = 386$ | $V_{Tweak} = 81$ | $V_{Ext} = 39$ | $A_{Free} = 2$ | $A_{Plateau} = 3$ |
|---|---|---|---|---|
| 00101000011 | 00001111001 | 00000110100 | 00000000001 | 00000000011 |

Table 2: Gray code for the genome of an individual with five parameters.

#### 4.2.4 Implementation

A genetic algorithm was implemented for finding optimal parameter values that resolve a set of given energies. The genome of the individuals is encoded as a 55-bit Gray code string. The fitness of an individual is equal to one of the resolution measures described in section 1.6. If some particles miss the detector during a simulation, the individual is penalized by adding to the fitness the number of particles that missed the detector. Since the resolution measures are constructed in such a way that a smaller value means better resolution, the problem becomes to minimize the fitness. Roulette wheel selection is used to choose parents that reproduce. The probability for an individual reproducing is inversely proportional to its fitness.

A program with a graphical user interface was developed in the programming language C#. The user is able to set limits for the parameters $V_{Bend}$, $V_{Tweak}$, $V_{Ext}$, $A_{free}$, $A_{plateau}$, within which an optimal solution is to be found. The user also defines a set of energies that are to be simulated. Furthermore, the population size, mutation and crossover probability and number of elites can be set by the user.

When the parameter values have been set, a population is initialized with random values selected from within the ranges the user has defined. Each individual is then translated into a SIMION command that is subsequently run. After a run, the output from SIMION is read and the resolution is calculated. In the case of the new resolution measure we get $N - 1$ measures of resolution between $N$ energies. The fitness of the individual is therefore set to the mean value of the resolution between the energies. The user can select if the mean is to be an arithmetic or a geometric mean.

When all individuals in the population have been evaluated, the population is sorted according to fitness. A number of elite individuals are copied to a new population, and retain the possibility of being parents to offspring in the same generation. Then follows the roulette wheel selection and reproduction process.

Two parents are selected and two offspring are formed and mutated accordingly, then transferred to the new population. This process continues until the new population contains the predefined number of individuals. The new population is evaluated and the process is repeated until a satisfactory solution is found. A GA is typically run with a population size of 30-50 individuals for around 50 generations before the population converges to a good solution.

Algorithm 1 shows the pseudocode for our GA.

```
1  Population ← RandomPopulation(Population_size, Parameter_ranges)
2  while StopCondition()
3   EvaluatePopulation(Population)
4   Individual_best ← GetBestSolution()
5   NewPopulation ← ∅
6   NewPopulation ← copyElites(NbrOfElites)
7   while NumberOfIndividualsIn(NewPopulation)<Population_size
8    Parent_1, Parent_2 ← RouletteWheelSelection(Population)
9    Offspring_1, Offspring_2 ← Crossover(Parent_1, Parent_2, P_crossover)
10   Offspring_1 ← Mutate(Offspring_1, P_mutation)
11   Offspring_2 ← Mutate(Offspring_2, P_mutation)
12   NewPopulation ← Offspring_1, Offspring_2
13  end
14  Population ← NewPopulation
15 end
16 return Individual_best
```

Algorithm 1: Pseudocode for the genetic algorithm

### 4.3 The curse of dimensionality

As the number of variables in a problem grows large, the task of optimization becomes more difficult. This is known as the "curse of dimensionality", a term first coined by Richard Bellman in 1961. He initially used it to describe the exponential increase of a hypervolume associated with adding extra dimensions to a space.

In our case we have limited ourselves to a parameter space of five variables. There is no easy way of determining if this problem lends itself to classical optimization. The function we want to minimize is not continuous or differentiable. However, using knowledge of the problem domain we can constrain the parameter values to within some interval that is relevant from a physics point of view. This decreases the space somewhat. However it is still interesting to implement a non-classical algorithm such as a GA. As previously mentioned, we have chosen to limit the problem by choosing three voltage differences that are of main importance and two variables that determine the position and size of a single electrostatic lens. Another important factor is that our function is costly to evaluate; it requires a full simulation for every evaluation. The optimization

algorithm need to sample the parameter space efficiently and reach convergence in a reasonable amount of time or function evaluations.

In theory one could try to find optimal values for each electrode separately. This would give the possibility for having multiple lenses that perhaps provide a better solution when dealing with a large range of energies. One could state with some certainty that finding optimal values for 20 variables is not an easy task, but it is probably a task more suitable for a GA than a classical algorithm.

# 5 Results

This section holds the main results of the thesis. Simulations and optimizations are made and presented. In all cases, if otherwise not stated, the source point pattern is the default pattern, and the source point spacing is 0.8 mm. Simulations are run with 11 source points with particles emanating in 16 different directions from each source point. The free and plateau electrodes are held constant at $A_{free} = 2$ and $A_{plateau} = 3$ in all simulation and optimization cases.

## 5.1 Simulations and visualizations

### Improvement of simulation speed

By comparing the previous way to launch simulations (sequentially with Matlab) to the C# library simulation times decreased by a factor of 50-100. The improvement was more pronounced for larger simulations than smaller ones. The tests were done by running simulations with the same values for the ranges of the voltages with both methods. The computer used for this, was running on a 8-core Intel i7 computer.

### Visualization with tool

Figure 16 shows the type of 2D diagrams that can be plotted after a simulation. The figure also shows the need to find a total measure of resolution for all energies by taking an average.

Figure 17 shows the type of 3D diagrams that can be plotted with the tool. This type of plot can be used as a way to manually optimize two parameters, since the parameter space can be fully visualized.

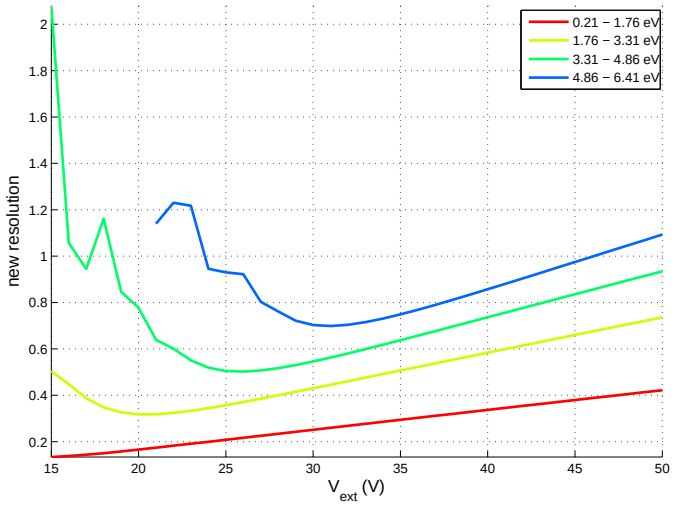


Figure 16: Resolution as a function of $V_{Ext}$ with $V_{Bend} = 200$ and $V_{Tweak} = 20$. Data is not shown for parameter values where particles miss the detector.

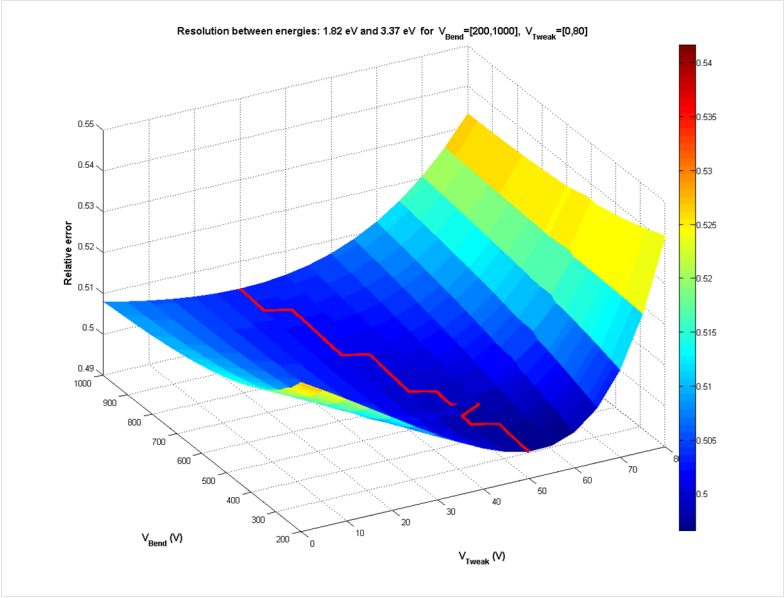


Figure 17: Resolution between two energies as a function of $V_{Bend}$ and $V_{Tweak}$ with $V_{Ext} = 30$ V. Values for $V_{Tweak}$ that give optimal resolution are drawn as a red line on the surface.

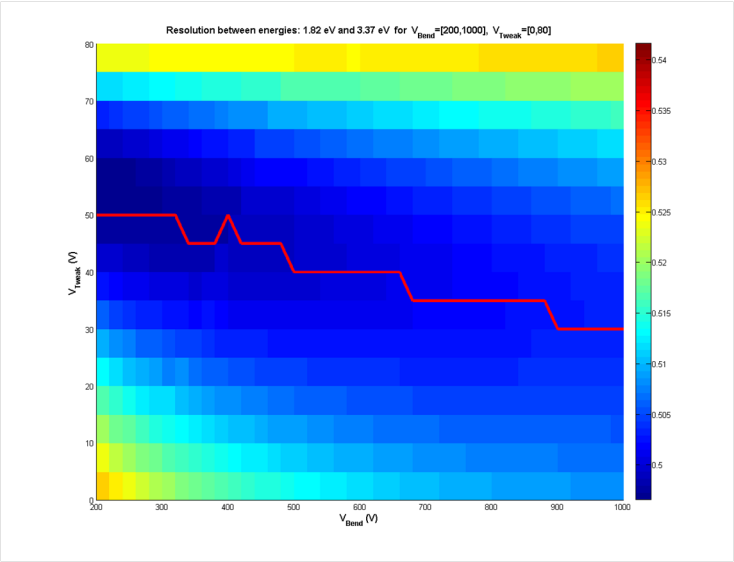


Figure 18: Resolution between two energies as a function of $V_{Bend}$ and $V_{Tweak}$ with $V_{Ext} = 30$ V. Values for $V_{Tweak}$ that give optimal resolution are drawn as a red line on the surface.

## 5.2 Optimization

The GA tool for optimization was used to find optimal settings to resolve the energies belonging to helium. Table 3 shows the lists of energies used in these simulations. They will be referred to by their names followed by the number of energies in the list, e.g. helium 5 etc.

| Helium 5 (eV) | Helium 4 (eV) |
|---|---|
| 0.21 | 0.21 |
| 1.76 | 1.76 |
| 3.31 | 3.31 |
| 4.86 | 4.86 |
| 6.41 | |

Table 3: Energies used in the optimizations

### 5.2.1 Testing GA

To show that the GA worked properly a test was done by attempting a simple optimization. By varying two parameters and keeping the rest constant, we get a unique opportunity of visualizing the fitness landscape as seen by the GA.

A GA was run to optimize $V_{Bend}$ and $V_{Tweak}$ while keeping $V_{Ext}$, $A_{free}$ and $A_{plateau}$ constant. The population size was 20 individuals, mutation probability 0.05, crossover probability 0.95 and two elites in the population. Figure 19 shows the result. This fitness landscape is first simulated and plotted with the tool described in 3.2.

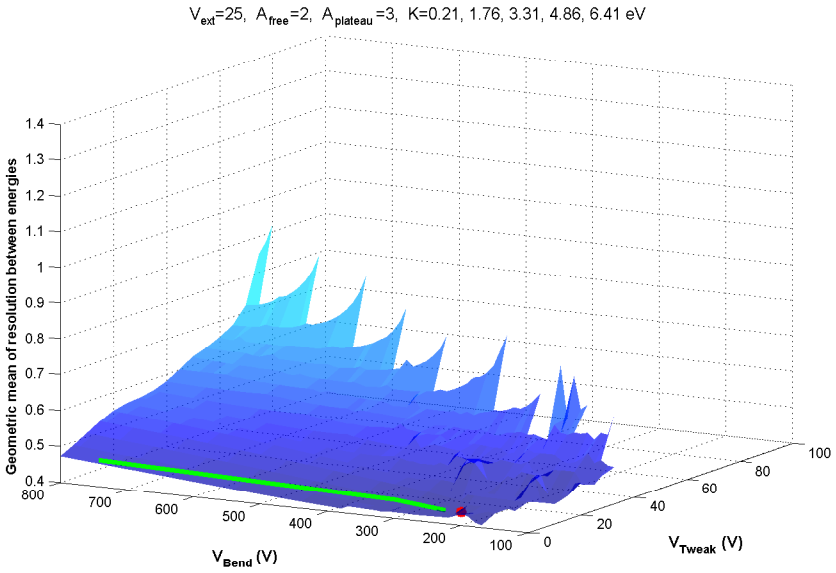


Figure 19: Geometric mean of the resolution between the energies in helium-5 as a function of $V_{Bend}$ and $V_{Tweak}$. Parameter values that lead to particles missing the detector are not shown. The optimal value is marked as a red dot and corresponds to $V_{Bend} = 240$ and $V_{Tweak} = 10$. The green line shows the path taken by a GA. The path shows the best individual for every generation.

The best solution found by our GA corresponds to $V_{Bend} = 256$ V and $V_{Tweak} = 8$ V. This solution was found after nine generations. At first glance it looks like the GA was not able to find the optimal solution. This has however to do with the step size that we used to simulate and show figure 19. The step size used was 10 V for $V_{Bend}$ and 5 V for $V_{Tweak}$. Another round of simulations was made to zoom in on the area around the previously found solution. This time the step size was reduced to 2 V for both parameters. Figure 20 shows the result of this narrower simulation. The green line that shows the best individual from every generation is taken from the previous GA optimization.

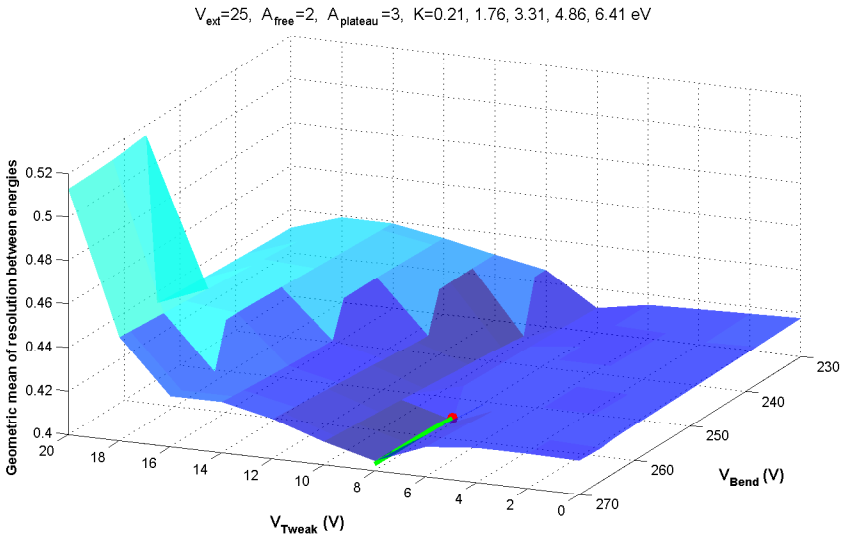


Figure 20: Same setup as in figure 19, but with narrower interval. The optimal value is marked as a red dot and corresponds to $V_{Bend} = 256$ and $V_{Tweak} = 8$. Note that the plot is rotated compared to figure 19.

Figure 20 shows that there is a dip in the fitness landscape, precisely at $V_{Bend} = 256$ and $V_{Tweak} = 8$. This is the solution found by our GA and was previously missed in the simulation because of the step size.

Finding optima in larger parameter spaces will not be as easy as shown in this example and the fitness landscapes will be difficult to visualize and understand. The results from this simple test are however encouraging. They show that our GA is able to find an optimal solution within a bound parameter space.

### 5.2.2 Example optimization

An optimization was attempted, to investigate the performance of the GA, for the energies in the list helium 5. Four rounds of optimizations were made with using the new resolution measure and taking the geometric mean fitness. Another four rounds were performed, using arithmetic mean fitness. This was done to compare the two averages and check for differences.

The GA ran with a population size of 30 individuals, 2 elites, a mutation rate of 0.06 and a crossover rate of 0.95. The population was initiated with random individuals between each round of optimization. The results are shown in table 4. The results are compared to settings found manually by Erik Månsson and Mathieu Gisselbrecht, when performing experiments with helium. The settings $V_{Bend} = 200$, $V_{Tweak} = 25$ and $V_{Ext} = 29$ were found manually and were

determined to be good enough to resolve the three lowest energies in helium 5. Although the two higher energies in the helium 5 list hit the detector they are not focused. During the experiments they used a slightly different geometry for the spectrometer.

Finally the performance of the GA is compared to a that of a pattern search algorithm. The pattern search was done using the global optimization toolbox in Matlab. First a point in the parameter space was chosen randomly. Then a pattern search algorithm was initiated twice at the same random point; once to optimize for geometric mean fitness and a second time for the arithmetic mean, both times using the new resolution measure. This procedure was repeated with four random starting points. The pattern search ran until the mesh size was below $10^{-6}$. Table 5 shows the start and end points that were found for all cases.

The parameter space was the same for both algorithms. The parameters were bounded to: $V_{Bend} = [100, 800]$, $V_{Tweak} = [0, 100]$ and $V_{Ext} = [0, 100]$. Although both algorithms optimized for the mean of the new resolution measure, the old total relative resolution is also presented, for the same solutions, for comparison.

| | Voltages | | | Fitness | | | |
|---|---|---|---|---|---|---|---|
| | | | | **new** | | **old** | |
| | **Bend** | **Tweak** | **Ext** | **Geom** | **Arithm** | **Geom** | **Arithm** |
| Manual | 200 | 25 | 29 | 0.4543 | 0.4973 | 0.2348 | 0.2718 |
| Optimizing for geometric mean | | | | | | | |
| **Search 1** | 352 | 2 | 25 | 0.4071 | 0.4500 | 0.201 | 0.2394 |
| **Search 2** | 214 | 9 | 26 | 0.4017 | 0.4328 | 0.2044 | 0.2437 |
| **Search 3** | 297 | 3 | 26 | 0.4051 | 0.4375 | 0.2044 | 0.2446 |
| **Search 4** | 241 | 9 | 25 | 0.4002 | 0.4396 | 0.2050 | 0.2381 |
| **Std** | 61.34 | 3.78 | 0.58 | 0.0031 | 0.0073 | 0.0025 | 0.0032 |
| Optimizing for arithmetic mean | | | | | | | |
| **Search 1** | 222 | 5 | 28 | 0.4208 | 0.4467 | 0.2175 | 0.2575 |
| **Search 2** | 249 | 7 | 26 | 0.4029 | 0.4363 | 0.2046 | 0.2441 |
| **Search 3** | 272 | 1 | 27 | 0.4111 | 0.4386 | 0.2113 | 0.2511 |
| **Search 4** | 231 | 8 | 26 | 0.4016 | 0.4328 | 0.2043 | 0.2439 |
| **Std** | 22.07 | 3.10 | 0.96 | 0.0088 | 0.0059 | 0.0063 | 0.0065 |

Table 4: Results from optimization of helium 5 energies with a genetic algorithm.

| | | Voltages | | | Fitness | |
|---|---|---|---|---|---|---|
| | | **Bend** | **Tweak** | **Ext** | **New** | **Old** |
| **Search 1** | start point | 400 | 50 | 40 | | |
| | end geometric | 136 | 31 | 30 | 0.4625 | 0.2562 |
| | end arithmetic | 156 | 23 | 28 | 0.4808 | 0.2697 |
| **Search 2** | start point | 670 | 90 | 15 | | |
| | end geometric | 101 | 72 | 40 | 0.5725 | 0.3625 |
| | end arithmetic | 101 | 75 | 41 | 0.6164 | 0.3944 |
| **Search 3** | start point | 295 | 55 | 96 | | |
| | end geometric | 149 | 24 | 28 | 0.4383 | 0.2530 |
| | end arithmetic | 155 | 22 | 28 | 0.4798 | 0.2661 |
| **Search 4** | start point | 382 | 51 | 43 | | |
| | end geometric | 181 | 12 | 26 | 0.4065 | 0.2063 |
| | end arithmetic | 103 | 51 | 35 | 0.5565 | 0.3407 |
| **Std geometric** | | 33.1 | 26.04 | 6.22 | 0.072 | 0.066 |
| **Std arithmetic** | | 30.9 | 25.36 | 6.27 | 0.066 | 0.062 |

Table 5: Results from optimization of helium 5 energies with pattern search algorithm.

# 6 Conclusions

The results from the optimization with GA shows that there is no noticeable difference between optimizing for arithmetic and geometric mean of the resolution. The idea behind using the geometric mean is that it would lend smaller importance to outlier values. Both settings found about the same range of values for $V_{Bend}$, $V_{Tweak}$ and $V_{Ext}$, with comparable standard deviations. When optimizing for the new resolution measure and taking either the geometric or arithmetic mean of the resolution between the energies, the other type of mean is also displayed in table 4. When optimizing for the arithmetic mean, the same result, calculated as a geometric mean can be compared to the case where we actually optimized for the geometric mean. The result shows that there is no difference in the quality of solutions between the two. In both cases the GA found solutions with better resolution values than the manually found solution.

The GA found quite similar values for $V_{Ext}$, but more varying values for $V_{Bend}$ and $V_{Tweak}$. This is because there is a trade off between high values for $V_{Bend}$ and low values for $V_{Tweak}$. This is easy to understand because both parameters tend to shrink the image on the detector. So a higher $V_{Bend}$ can be compensated for with a lower $V_{Tweak}$ to give a similar resolution value. Although for different energies, this trade off can be seen in figure 18.

The result for the pattern search algorithm were, as expected, very sensitive to the initial starting point. In the four cases the pattern search did not find

parameter values that were better than the ones found by the GA. The pattern search is probably still useful if an informed guess is made for the start values instead of choosing random ones. It could for example be used to find an even better optimum near some experimentally found value. This test was however done to compare the performance when searching for a global optimum. Since the GA starts with a random population, so should the pattern search start at random point so a comparison can be made.

### Outlook

This work was about developing tools to use when working with the spectrometer. The tools have undergone some testing, but a lot remains to be done. Unfortunately, within the time limit of a bachelor thesis, it is not possible to extensively test and improve the tools. The infrastructure is now in place and further improvements can surely be made. A few things things can be mentioned with regard to this.

It seems that the new resolution measure tends to lead to small values for $V_{Tweak}$. Low values for $V_{Tweak}$ lead to a larger, but less focused image on the detector. It could be that this is a result of the way we set the limitation of the intrinsic detector resolution ($\sigma = 0.5$ ns). One way to test if this affects the results would be to decrease the value of $\sigma$ to something smaller, around 0.1 ns, or perhaps increasing the source point spacing to 1.6 mm. If it is the case that adding $\sigma$ to the transformed data skews the result in favor of small values for $V_{Tweak}$, then one could try a different way to include $\sigma$ when transforming the data, for example $\sigma \cdot \cos|\alpha|$ or $\sigma \cdot (\cos|\alpha| + e \cdot \sin|\alpha|)$.

Perhaps the five energies used for optimization had a range that was too large. It can be suspected that the difference between the smallest and largest energies in the list is too large to find an optimal resolution for both. If there was more time, it would be interesting to try an optimization for three or four of energies and compare that to the manually found solution.

We have attempted to optimize for three parameters only, but the program is capable of handling varying $A_{free}$ and $A_{plateau}$ as well. It would not be a big step to expand the program to handle variable values for all 20 electrodes in the spectrometer. This would open for possibilities of more complex solutions, with multiple lenses, that can perhaps give good resolutions between energies in a wider range.

But it is not only a problem of optimization. There is a point to having a small number of parameters so that a human experimenter can understand and fine tune the results of an optimization so they can work in practice. But it is also a matter of resources; it would be costly and cumbersome to include many more computer controlled voltage regulators that would be needed to control each electrode separately.

# References


[1] Michalewicz Cezary, Janikow Zbigniew. An experimental comparison of binary and floating point representations in genetic algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[2] J. M Dahlström. *Light-Matter Interaction on the Attosecond Timescale.* PhD thesis, Lund University, 2011.

[3] Gustavo A. Garcia, Laurent Nahon, Chris J. Harding, Elisabeth A. Mikajlo, and Ivan Powis. A refocusing modified velocity map imaging electron/ion spectrometer adapted to synchrotron radiation studies. *Review of scientific instruments.*, 76(5):053302–053302–11, 2005.

[4] Frank Gray. Pulse code communication, u.s. patent 2,632,058, March 1953.

[5] Jason Lohn, Gregory Hornby, and Derek Linden. An evolved antenna for deployment on nasa's space technology 5 mission. In *Genetic Programming Theory and Practice II*, volume 8 of *Genetic Programming*, pages 301–315. Springer US, 2005.

[6] J Mauritsson, J M Dahlström, E Mansten, and T Fordell. Sub-cycle control of attosecond pulse generation using two-colour laser fields. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 42(13):134003, 2009.

[7] Melanie Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

[8] Marco Swoboda. *Attosecond Wave Packet Metrology.* PhD thesis, Lund University, 2010.

[9] Sewell Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proc of the 6th International Congress of Genetics*, 1932.