UCRL-LR-135618

Genetic Algorithms Applied to Nonlinear and Complex Domains

Danny Barash

June 1999

Lawrence Livermore National Laboratory

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

UCRL-LR-135618
Distribution Category UC-99

Genetic Algorithms Applied to Nonlinear and Complex Domains

Danny Barash

Manuscript date: June 1999

LAWRENCE LIVERMORE NATIONAL LABORATORY
University of California • Livermore, California • 94551

Genetic Algorithms Applied to Nonlinear and Complex Domains

by

DANNY BARASH

B.S. (Hebrew University, Jerusalem) 1992
M.S. (University of California, Davis) 1996

DISSERTATION
Submitted in partial satisfaction of the requirements for the degree of
DOCTOR OF PHILOSOPHY

in

Applied Science

in the

OFFICE OF GRADUATE STUDIES
of the
UNIVERSITY OF CALIFORNIA
DAVIS

Approved:

____
Chair

____

____

Committee in Charge

1999

Genetic Algorithms Applied to Nonlinear and Complex Domains

Danny Barash

June 1999

Applied Science

Genetic Algorithms Applied to Nonlinear and Complex Domains

Abstract

The dissertation, titled "Genetic Algorithms Applied to Nonlinear and Complex Domains", describes and then applies a new class of powerful search algorithms (GAs) to certain domains. GAs are capable of solving complex and nonlinear problems where many parameters interact to produce a 'final' result such as the optimization of the laser pulse in the interaction of an atom with an intense laser field. GAs can very efficiently locate the global maximum by searching parameter space in problems which are unsuitable for a search using traditional methods. In particular, the dissertation contains new scientific findings in two areas.

First, the dissertation examines the interaction of an ultra-intense short laser pulse with atoms. GAs are used to find the optimal frequency for stabilizing atoms in the ionization process. This leads to a new theoretical formulation, to explain what is happening during the ionization process and how the electron is responding to finite (real-life) laser pulse shapes. It is shown that the dynamics of the process can be very sensitive to the ramp of the pulse at high frequencies. The new theory which is formulated, also uses a novel concept (known as the (t,t') method) to numerically solve the time-dependent Schrödinger equation. Second, the dissertation also examines the use of GAs in modeling decision making problems. It compares GAs with traditional techniques to solve a class of problems known as Markov Decision Processes. The conclusion of the dissertation should give a clear idea of where GAs are applicable, especially in the physical sciences, in problems which are nonlinear and complex, i.e. difficult to analyze by other means.

Prof. Ann E. Orel
Dissertation Committee Chair

To My Parents, Debora and Isaac

Contents

List of Figures

Acknowledgements

First, I would like to thank my academic advisor Ann Orel for the tremendous support during all stages of my studies. At times of trouble when a medical incident threatened my continuation in the program, her husband Hugh Woodin was recruited to help with a reallocation which allowed my recovery. I thank both of them for the time I spent at the Berkeley Mathematics Department in a most friendly, professional and inspiring environment. The advices and help I received from Ann at all times were invaluable, with a clear intention to put student's sake first.

Second, I would like to thank Bill Hoover for accompanying the final stage of my thesis in a notable way. His authoritative presence was clearly felt and much encouragement I gained from his consent to fill in the gap and oversee my timely graduation. I am grateful for the effort he spent to ensure my studies will end successfully and on schedule.

Third, I would like to thank Roi Baer: then a post-doc at the Berkeley Chemistry Department and now a faculty member at the Hebrew University in Jerusalem. His extraordinary devotion to graduate students and energetic guidance enabled to resolve dynamic stabilization in a remarkably short time. It was in the late hours of the night at the Berkeley "Village", with Ahinoam on his lap, when the (t-t') equations were set in an original way to explain the mystery surrounding atomic stabilization.

Once he took the lead, different pieces of the puzzle were put together to form a coherent picture and a paper submission to Physical Review Letters was inevitable.

I thank Rao Vemuri for the paper we wrote together on genetic algorithms as well as the courses on various aspects of computations. I thank Ken Kulander for early sessions on atomic stabilization and his feedback on the paper with Ann and Roi later on.

I would like to thank distinguished professors at the Department of Applied Science: Garry Rodrigue, Richard Christensen and Fred Wooten from whom I took some of the courses, and Richard Scalettar from the Physics Department at Davis. I had the pleasure to be a guest in group meetings at Lawrence Berkeley Labs hosted by Tom Rescigno, Bill Isaacs and Bill McCurdy. At the Berkeley Mathematics Department, where I spent most of the time during the final two years of my studies, I thank Hugh Woodin, Ted Slaman, Jamie Sethian and Alexandre Chorin for his good advice.

Special thanks to friends from the Department of Applied Science: Hank Alme, Mark Baertschy, Victor Castillo, Rebecca Darlington, Sony Lemoff, Tony McGofna, Steve Moon, Jason Zweiback, Mike Wickett, Dan White, Dan Wake. Exceptional assistance was always available from the administrative staff: Archietta Johnson, Monica Cowart, Estelle Miller, Donna Clifford and Paula Silva.

A collection of special people at Berkeley and Stanford, who were very helpful on various professional matters at different times during my studies, are: Avital Baer, Aviv Nevo, Ronald Parr, Yossi Rubner, Brian Lemoff, Raz Kupferman, Ronny Kimmel, Ian Goldberg, Golan Yona, Nancy Blachman, Gady Costeff and Dov Gorman.

Countless help and support I received from my closest family: Debora, Isaac and Ronit. Finally, I would like to thank the Elzam family from the agricultural town of Aromas, CA. Last but not least, I would like to thank Dov Michaeli, vice president and chief scientist at Aphton cooperation. At troubled times, his advice on the right medications to correct the HMO failure led to my recovery. In addition, the many conversations we had about life in industry will accompany me in the future.

I thank the generous support of the Department of Energy under contract W-7405-Eng-48 which enabled me to pursue my graduate studies as a student-employee at Lawrence Livermore National Laboratories. Support was also provided by the National Science Foundation under grant PHY-97-22136. Computing resources were supported by the National Partnership for Advanced Computational Infrastructure.

I would like to end here with a promise, not to forget the times I was struggling as a graduate student and assisted by good people who understood this state-of-mind.

Danny Barash

Chapter 1

Introduction

1.1 Preface

The scientific findings contained in this dissertation grew from a set of computational experiments, designed to simulate the irradiation of atoms by high-intensity laser pulses. These pulses, with intensities above $10^{13}W/cm^2$, are capable of sending thousands or even millions of photons with an energy of approximately $10^{-13}erg$ through a typical atomic cross section ($10^{-16}cm^2$) during a pulse of a picosecond or less. The pulse interaction with atoms and molecules is interesting to model, because both the target and the frequency of photons are altered as a consequence.

Atomic stabilization is an interesting effect which occurs as a result of high-intensity laser-atom interaction. As the laser intensity gets higher, the probability that an electron is stripped from the core atom and becomes ionized increases with intensity. At first, the increase is linear. However, at intensities higher than $10^{13} W/cm^2$, the pattern changes. This process can be described with four input parameter values (intensity, frequency, pulse shape, duration of the pulse) for computer simulation. Computer simulations are often performed on a simplified model of a hydrogen atom due to time and memory constraints. Still, it is possible to improve on past attempts, by introducing systematic search methods in conjunction with parallel runs. Such systematic schemes can efficiently locate the "interesting" parts of parameter space. That way, useful processing of information can be performed with existing resources of time and memory allocations. These techniques led to my discovering that a high-frequency limit exists when atoms are stabilized by pulses with a short rise time. The ionization was defined as the population which is left after the pulse is over relative to the population in the initial state of the system. A structure was observed when plotting the ionization probability versus intensity. Chapter 4 explains the high-frequency limit structure based on the slope of the pulse rise-time.

The systematic search scheme which was chosen for this task, a genetic algorithm, is described in this chapter. It is implemented throughout this dissertation. It is interesting to examine the following questions regarding the success of genetic algorithms in searching parameter space. First, are there other general stochastic search methods that can perform equally well? Second, are there deterministic search methods that can perform better than genetic algorithms on problems where both can be applied? The answers to these questions point out the strengths and weaknesses of the genetic algorithm methods.

To answer both questions, detailed comparisons with other techniques are performed. In chapter 3, a comparison between genetic algorithms and simulated annealing demonstrates that simulated annealing performs equally well on searching for the optimal frequency to maximize atomic stabilization. In chapter 5, a model problem from the area of complex decision making is introduced along with dynamic programming, a deterministic search technique which is specifically used to solve problems of this type. In chapter 6, genetic algorithms and dynamic programming are compared. Dynamic programming is clearly superior to genetic algorithms for this model problem.

Thus, the thesis is organized in two parts: the first is a *successful* implementation of genetic algorithms, which culminates in the non-adiabatic high-frequency theory described in chapter 4. It is shown that simulated annealing could have been used with an equal amount of success. The second is an *unsuccessful* implementation of genetic algorithms. The conclusion of this dissertation, drawn in chapter 7, is that genetic algorithms can be used as general search schemes. Although genetic algorithms can not compete successfully with deterministic algorithms, they can conveniently be implemented on a variety of problems. However, they represent no conceptual breakthrough in their introduction from the theoretical standpoint, because simulated annealing offers the same type of mechanism without the distraction of new terminology borrowed from biology and economics.

1.2 Genetic Algorithms

Genetic algorithms are search algorithms that use concepts from reproduction and natural selection to produce better solutions (children) from previous solutions (parents). Genetic algorithms were invented by John Holland in the 1960s and were developed by Holland [1] and his students [2] and colleagues at the University of Michigan in the 1960s and the 1970s. Since then, genetic algorithms have given rise to many new applications in a variety of disciplines. Genetic algorithms are a rapidly expanding field.

Much like simulated annealing, the genetic algorithm belongs to the same class of stochastic optimization techniques which offer an alternative to traditional methods such as the gradient search and other calculus-based methods. However, simulated annealing and genetic algorithms utilize a different strategy. In [3], these two techniques are discussed and applied to several model problems.

Although the field has grown to encompass several subfields which are actively being pursued on their own right, the basic concept of optimizing a function using genetic algorithms is the driving force behind all these extensions. *Genetic programming*, evolving a set of computer programs to perform various tasks, as well as *classifier systems*, evolving rule-based systems, are two examples of such active subfields. There are other evolution strategies aside of genetic algorithms. Therefore the whole field (which includes genetic algorithms as a subset) is often referred to as *evolutionary computing.* In this dissertation, however, attention is only given to genetic algorithms in their basic form.

1.3 Genetic Algorithm Basic Terminology

Since genetic algorithms simulate biological evolution, it is useful to introduce some of the terminology borrowed from biology. In a biological system, the structure that encodes the prescription of how the organism is to be constructed is called *a chromosome.* One or more chromosomes may be required to specify the complete organism. The complete set of chromosomes is called *a genotype*, while the resulting organism is called *a phenotype.*

Each chromosome is made up of individual structures called *genes.* Each gene encodes a specific feature of the organism, such as the eye color. The different values of a gene, such as green, blue or hazel colors, are called *alleles.* The location of the gene within the chromosome, *the locus*, is responsible for the characteristic the gene represents.

In a genetic algorithm simulation, chromosomes are represented by *a string* of some variable type. Binary strings, which comprise binary bits, are most common although other representations have been used. However, it should be stressed that at this time, not much is known from the theoretical standpoint about other representations aside of the binary representation (as will be discussed in the theoretical foundations section). In the binary representation, the alleles are zero and one.

Furthermore, in a case where there is only one chromosome per organism, the chromosome and genotype are the same. The phenotype is the solution decoded from the genotype. It is possible to refrain from biological terminology and speak of strings, positions on the string, and values, instead of chromosomes, genes, and alleles.

1.4 Genetic Algorithms in Practice

First, an example problem is solved using the standard genetic algorithm implementation, known as the *simple genetic algorithm.* This simple example should illustrate how genetic algorithms work.

1.4.1 Simple Genetic Algorithm

The example problem which will be used to illustrate the mechanism of a simple genetic algorithm in practice, is a typical problem which is difficult for traditional techniques. Given the following noisy function, find the maximum of that function:

$$f(x) = 1 + \frac{\cos(x)}{(1 + x^2/100)} \tag{1.1}$$

This function (figure 1) contains a global maximum at x=0 and many local maxima which are suboptimal. Traditional hill-climbing techniques will fail miserably to advance towards the right solution, after reaching one of the suboptimal peaks and getting stuck there. The genetic algorithm evolution procedure to find the global maximum is now described step by step.

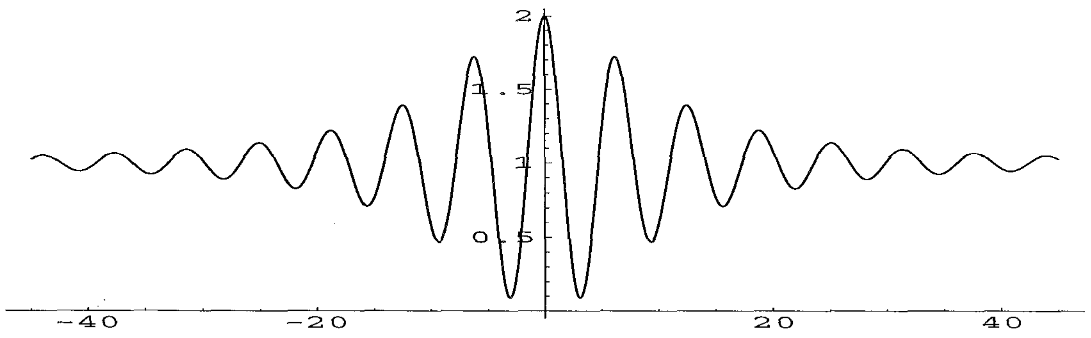


Figure 1.1: An example function to be optimized by a genetic algorithm search

1. Chromosome Representation: Using a binary representation, represent

each chromosome as a ten-digit binary string. Scale the phenotypes to values between -40 and 40. Therefore, the chromosome $[0,0,0,0,0,0,0,0,0,0]$ must decode to the value -40, and the chromosome $[1,1,1,1,1,1,1,1,1,1]$ must decode to the value 40. Note that the binary number 1111111111 is equal to decimal 1023, which means that $2^{10} = 1024$ points are used to represent the interval between -40 and 40. With the chosen discrete representation, chromosome $[0,1,1,1,1,1,1,1,1,1]$ decodes to -0.0391007, the largest negative phenotype. Chromosome $[1,0,0,0,0,0,0,0,0,0]$ decodes to 0.0391007, the smallest positive phenotype.

2. Initialization: Construct a random population with ten individuals. The ten individuals are spread evenly over all space, in the example above ten random points are chosen between -40 and 40.

3. Fitness Function: Based on the quantity to be optimized, construct a *fitness function* that assigns a score (fitness) to each chromosome. Thus, each individual in the population gets a fitness which will determine its fate during the evolutionary process. The fitness function for finding the global maximum in figure 1 is simply the function to be optimized itself: $f(x) = 1 + \cos(x)/(1 + x^2/100)$

4. Parent Selection: Select parents for reproduction. Several methods can be applied, one of which is the *roulette-wheel* method. In this method, the individuals are organized from lowest to highest fitness. Two parents are then selected with a probability in direct proportion to their fitness values, so that individuals with a higher fitness are more likely to be chosen.

5. Reproduction: Apply *crossover* and *mutation* on the parents. These two simple genetic operators, when applied together, have been shown [1] to improve the fitness of the population as a whole in a systematic and efficient manner. There are many different crossover methods. The simplest is a single-point crossover and is described as follows: a pair of chromosomes, such as [1011000101] and [1111010110], are selected as parents' chromosomes and the crossover point is selected to be between the $4th$ and $5th$ locus. The childrens' chromosomes are then formed by combining opposite parts of each parent's chromosome . For the example given above, the children are: [1011010110] and [1111000101]. More precisely, after selecting two parents for mating, a biased coin flip with certain probability of heads will determine whether to proceed with the crossover. If the coin toss is successful, a crossover point is chosen at random and two

children are produced by slicing the genes up to the crossover point from one parent with the genes beyond the crossover point from the other parent. If the coin toss is not successful, the parents themselves are simply returned as the new children.

The crossover mechanism was theoretically shown [1] to generate a population whose overall fitness increases with time, as the desirable features of each parent are combined. It can happen, however, that crossover results in children who are less fit than their parents. By subjecting each of the genes to a small probability of mutation, it is possible occasionally to reverse the results of a bad crossover. Mutation can occur at each locus in a chromosome with some probability, usually very small, to produce variations just as in natural evolution.

6. Populating the New Generation: Build the next generation. A simplistic approach is to mate enough parents so that enough children are produced to completely replace their parents. This technique, called *generational replacement*, allows for the most thorough possible mixing of genes (whether desirable or not) in the new generation. It is possible to counter some of the negative effects of generational replacement by retaining a certain number of the best individuals from the previous generation, a strategy which is often used and is called *elitism.* Therefore at the opposite side of the spectrum from generational replacement, there exists a technique

called *steady state* reproduction which is sometimes used. In this method, a certain number of individuals are replaced with an equal number of children and all other individuals remain unchanged. In the simple genetic algorithm, generational replacement is mostly used.

7. Go to step 3 until convergence criterion is achieved.

For the example above, applying the simple genetic algorithm with a crossover percentage of 0.75, mutation percentage of 0.008 and an initial population of 100 candidates, five generations were enough to reach $y = 1.87132$, which is very close to the global maximum located at $y = 2.0$.

1.4.2 Micro Genetic Algorithms

In this dissertation, a variant of genetic algorithms called *micro genetic algorithms* [4] is often used. The main differences between micro genetic algorithms and simple genetic algorithms are described in this section.

In a simple genetic algorithm, a question which has been addressed in various studies is how one should choose the optimal population size. The basic idea in a micro genetic algorithm, as its name suggests, is to work with the smallest population size possible in order to minimize the number of function evaluations required.

In cases where the function evaluation is expensive, this strategy proves to be effective. However, at the cost of saving time, micro genetic algorithms are less accurate and comprehensive in finding the exact global optimum compared to simple genetic algorithms. In problems in which near-optimality and best-so-far string are more important than the average behavior of the population as a whole and exact optimality, this strategy is recommended and in principle should work more efficiently than the simple genetic algorithm.

The motivation behind micro genetic algorithms is that genetic algorithms generally perform poorly with very small population size, due to insufficient information processing and early convergence to non-optimal results. Therefore, some slight modifications are needed, rather than simply taking the simple genetic algorithm with a small population size. A step by step procedure for the micro genetic algorithm implementation is given below:

1. Randomly select a population size of 5 individuals initially, or 4 random and 1 good string from any previous search. 5 is the minimal population size which can offer enough diversity in the evolution process.

2. Evaluate fitness and determine the best string. Label it as string 5 and carry it to the next generation (elitist strategy).

3. Choose the remaining 4 strings for reproduction (the best string also com-

petes for a copy in the reproduction) based on a deterministic *tournament selection* strategy [2]. In the tournament selection strategy, the strings are grouped randomly and adjacent pairs are made to compete for the final four.

4. Apply a crossover percentage of 1.0. Mutation rate is kept to zero, since enough diversity is introduced after every convergence through new population of strings.

5. Check for nominal convergence (an example would be to count the number of different bits in the whole population from best member, and if less than 5% of number of bits are different, consider population to be converged). If converged go to step 1.

6. Go to step 2.

As was mentioned above, average behavior of the population has no meaning in the micro genetic algorithm. Therefore performance measure, when compared to other genetic algorithm strategies, should solely be based on the best-so-far string rather than on any average performance.

1.4.3 Parallel Implementations

It is well known that genetic algorithms are structured in a way which is highly parallelizable. In problems which are time consuming and computationally very expensive, it is more efficient to utilize a parallel implementation of genetic algorithms.

Among several prototypes which were examined in early studies [5] for designing a parallel genetic algorithm, the simplest is the master-slave prototype which is used throughout this dissertation.

In a master-slave prototype, a single master process coordinates k slave processes (k can be chosen depending on the problem specifics). The master process controls selection, mating, and the performance of genetic operators. The role of the slaves is only to perform function evaluations. Since each slave corresponds to a different function evaluation, the need for communication among processes is avoided (besides a one-time global broadcast of data from the master to all slaves, which is sometimes used for convenience). This makes the scheme *embarrassingly parallel.*

This scheme is known to have two major drawbacks. First, a fair amount of time is wasted if there is a variance in the time of function evaluations. Second, it relies on the health of the master process as far as reliability issues are concerned. In problems which are examined in this dissertation and many others, such drawbacks are of less relevance.

1.5 Theoretical Foundations

In this section, some of the approaches to understand the theoretical foundations of genetic algorithms are briefly described. A complete survey of work on the theory of genetic algorithms can easily fill several volumes. It should be noted however, that genetic algorithms theory is by no means a closed book and there are more open questions than answered ones.

1.5.1 The Schema Theorem

The traditional theory of genetic algorithms (first formulated in [1] and expanded in [2]) assumes that at a very basic level, genetic algorithms work by discovering and recombining good *building blocks* in an efficient way. The idea is that good solutions tend to be made up of good building blocks, which are combination of bit values that confer higher fitness on the strings in which they are present.

To formalize the informal notion of building blocks, the notion of *schemas* (or *schemata*) was introduced. A schema is a set of bit strings made up of ones, zeros and asterisks, the asterisks representing wild cards (don't cares). For example, the schema $H = 1****1$ represents the set of all 6-bit strings that begin and end with 1. The strings that fit this template (e.g., 100111 and 110011) are said to be *instances* of H. The schema H is said to have two defined bits (non-asterisks) or, equivalently, to be of *order* 2. Its *defining length* (the distance between its outermost defined bits) is 5. Traditional genetic algorithm theory is based on schemas being the building blocks that the genetic algorithm processes effectively under the operators of selection, mutation and crossover.

The approximate dynamics of an increase and decrease in schema instances can now be formulated as follows. Let H be a schema with at least one instance present in the population at time t. Let $m(H,t)$ be the number of instances of H at time t, and let $\hat{u}(H,t)$ be the observed average fitness of H at time t (i.e., the average fitness of instances of H in the population at time t). The goal is to calculate $E(m(H,t+1))$, the expected number of instances of H at time $t+1$. With the assumption that selection is carried out in a roulette-wheel fashion, the expected number of offspring of a string x is equal to $f(x)/\bar{f}(t)$, where $f(x)$ is the fitness of x and $\bar{f}(t)$ is the average fitness of the population at time t. Letting $x \in H$ denote that x is an instance of H, the following relationship holds:

$$E(m(H,t+1)) = \sum_{x \in H} f(x)/\bar{f}(t) = (\hat{u}(H,t)/\bar{f}(t))m(H,t) \tag{1.2}$$

since by definition, $\hat{u}(H,t) = (\sum_{x \in H} f(x))/m(H,t)$ for x in the population at time t.

Crossover and mutation can both destroy and create instances of H. In order to obtain a lower bound on $E(m(H,t+1))$, only the destructive effects are considered (the worst case). First, the disruptive effect of crossover is examined. Schema H will survive under a single-point crossover only if one of the offspring is also an instance of schema H. Letting p_c be the probability that a single-point crossover will be applied to a string, $d(H)$ be the defining length of H and l be the length of bit strings in the search space, a lower bound on the probability $S_c(H)$ that H will survive single-point crossover is:

$$S_c(H) \geq 1 - p_c \left(\frac{d(H)}{l-1} \right) \tag{1.3}$$

The lower bound was obtained for the following reasoning. Crossovers occurring within the defining length of H can destroy H (i.e., can produce offspring that are not instances of H), so the fraction of the string that H occupies multiplied by the crossover probability gives an upper bound in the probability that it will be destroyed. Subtracting this value from 1 gives a lower bound on the probability of survival $S_c(H)$. In short, the probability of survival under crossover is higher for shorter schemas.

Second, the disruptive effect of mutation is examined. Letting p_m be the probability of any bit being mutated and $o(H)$ the order of H (i.e., the number of defined bits in H), then $S_m(H)$, the probability that schema H will survive under mutation of an instance of H is:

$$S_m(H) = (1 - p_m)^{o(H)} \tag{1.4}$$

That is, for each bit, the probability that the bit will not be mutated is $1 - p_m$, so the probability that no defined bits of schema H will be mutated is this quantity multiplied by itself $o(H)$ times. In short, the probability of survival under mutation is higher for low-order schemas. Combining the two disruptive effects, one obtains:

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)}{\bar{f}(t)} m(H,t) \left(1 - p_c \frac{d(H)}{l-1}\right) [(1-p_m)^{o(H)}] \tag{1.5}$$

This is known as the Schema Theorem [1,2], or the Fundamental Theorem of Genetic Algorithms. It describes the growth of a schema from one generation to the next and implies that short, low-order schemas whose average fitness remain above average, will receive exponentially increasing numbers of samples (i.e., instances evaluated) over time. The reason for the exponential growth is that the number of samples of those schemas that are not disrupted and remain above average in fitness increases by a factor of $\hat{u}(H,t)/\bar{f}(t)$ at each generation. Why this supports the observation in practice that genetic algorithms are indeed efficient stochastic algorithms, also from the theoretical standpoint, is discussed in the next section.

1.5.2 The Multiarmed Bandit Analogy

The Schema Theorem proves that schemas of short defining length, low order, and above average fitness receive exponentially increasing trials in future generations. This section attempts to answer, using a well-founded problem from statistical decision theory, why this result is important. The basic idea is that the optimal solution to the *two-armed bandit* problem and its extension, the *k-armed bandit* problem, is very similar in form to the exponential allocation of trials obtained from the Schema Theorem to model genetic algorithms. Only a qualitative brief sketch of the results is given in this section, a comprehensive treatment can be found in the genetic algorithm literature [2].

In the two armed bandit problem, a gambler is given N coins with which to play a slot machine having two arms (a conventional Las Vegas slot machine is a *one-armed bandit*). Assuming that one of the arms pays an award of μ_1 with a variance of σ_1^2 and the other arm pays an award of μ_2 with a variance of σ_2^2, where $\mu_1 \geq \mu_2$, a decision should be made as to which arm to play. The difficulty is that the gambler does not know these payoff rates or their variances in advance, it can only be estimated by playing coins on the different arms and observing the payoff obtained on each. A fundamental theme in adaptive systems theory is applicable here, in which the decision maker must make a sequence of decisions about which arm to play and at the same time collect information about which is the better arm. This is known as the trade-off between *exploration* for knowledge and the *exploitation* of that knowledge. The tension between exploration and exploitation can be viewed as the most basic guiding rule in an adaptive system [1], whether genetic algorithms or bandits, and an optimal balance between the two must be found.

Holland has performed calculations that show how trials should be allocated between the two arms to minimize expected loss. This results in the allocation of n^* trials to the worse arm and $N - n^*$ trials to the better arm, where n^* is given by the following equation:

$$n^* \cong b^2 \ln \left[\frac{N^2}{8\pi b^4 \ln N^2} \right] \tag{1.6}$$

where $b = \sigma_1/(\mu_1 - \mu_2)$. Turning the equation around, the following equation predicts the number of trials given to the observed better arm:

$$N - n^* \cong N \cong \sqrt{8\pi b^4 \ln N^2} \cdot e^{n^*/2b^2} \tag{1.7}$$

which implies that in order to allocate trials optimally, the gambler should give slightly more than exponentially increasing trials to the observed best arm. Such strategy is unfortunately not realizable, as it requires knowledge of outcomes before they occur. Its importance is by forming a bound on performance that a realizable strategy should try to approach. Since the schema theorem guarantees giving at least an exponentially increasing number of trials to the observed best building blocks, it is therefore found that the genetic algorithm is a realizable yet near optimal procedure for searching among alternative solutions.

Finally, it should be noted that with a genetic algorithm we are no longer solving a simple two-armed bandit problem, but a simultaneous solution of many multiarmed bandits. An extension of the two-armed bandit can be made. It is also worthwhile mentioning that the Schema Theorem was developed using a three-operator (selection, crossover, mutation) genetic algorithm and a binary representation, which is known as the simple genetic algorithm. While there are extensions to the Schema Theorem for treating other types of representations and crossover operators, not much is known from the theoretical standpoint about many genetic algorithm variants which are sometimes used in practice.

1.5.3 The Ising One-Dimensional Model Analysis

Finally, an approach to use methods from statistical mechanics to predict macroscopic features of a genetic algorithm over a course of a run, on a given toy problem [6], is briefly sketched.

The model problem being used is the one-dimensional Ising model. It consists of a vector of adjacent spins, $\vec{S} = (S_1, S_2, ..., S_{N+1})$, where each S_i is either -1 or $+1$ (which naturally leads to a binary representation). Each pair of neighboring spins $(i, i+1)$ is coupled by a real-valued weight J_i. The total energy $E(\vec{S})$ of the spin configuration $\vec{S}$ is:

$$E(\vec{S}) = -\sum_{i=1}^{N} J_i S_i S_{i+1} \tag{1.8}$$

The goal is to find the minimal energy states. The genetic algorithm is then set with the problem of finding an $\vec{S}$ that minimizes the energy with given J_i's (the J_i values were selected ahead of time at random in $[-1, +1]$). A chromosome is a string of $N+1$ spins and the fitness of a chromosome is the negative of its energy. The initial population is generated by choosing such strings at random. At each generation a new population is formed by selection of parents that engage in single-point crossover to form offspring. For simplicity, no mutation was used and a *Boltzmann selection* procedure was chosen to select parents [2].

A way was found to devise a mathematical model, based on a set of equations to describe the effects of selection and crossover on the cumulants, aiming to predict changes in the distribution of energies (the negative of fitnesses) in the population over time. Indeed, the predicted evolution of the first two cumulants and their observed evolution in an actual genetic algorithm run did match remarkably well. The approach is not a general method for predicting genetic algorithm behavior (apart from a special case which is much dependant on the example model). Therefore it can not be used, as devised, to predict any actual genetic algorithm evolution run in common problems which are of real value. However, it serves as an illustration to the statistical nature of genetic algorithms and the fact that a macroscopic (rather than a microscopic) approach, borrowed from statistical mechanics, might be used in the future to assist in some way to the understanding of a genetic algorithm behavior.

1.6 Contribution of this Dissertation

In contrast to analytical solution analyses of difficult physics problems in certain limiting cases, it is almost impossible to cover all of parameter space while performing numerical experiments in a computer simulation. Often, one conducts such experiments by trial-and-error, based on intuition and acquaintance with a given problem.

Beginning with an introduction to genetic algorithms, this dissertation attempts to examine their use as a systematic search tool for finding optimal values in parameter space. In the first part, genetic algorithms are implemented on a nonlinear problem in atomic and molecular physics, which leads to a new theoretical formulation for solving this problem. In the second part, genetic algorithms are attached to a complex decision making problem in artificial intelligence. Lessons can be drawn, based on the success of this approach to make progress with the given problems. These precepts can be useful to researchers who are faced with similar difficult problems in computational physics as well as other scientific fields.

1.6.1 Application of Genetic Algorithms

Genetic algorithms have been applied to a wide range of problems since their invention. To mention only a few applications, genetic algorithms were implemented in radar and communication systems, neural networks, signal processing, geophysics, scheduling problems, protein structure predictions in molecular biology, data mining, etc. In addition to solving practical problems, they have been used in scientific models of natural evolution and ecosystems.

Applications of genetic algorithms to physics problems, in particular to laser physics, are steadily growing. The first use of genetic algorithms in optical control [8] was in designing laser pulses to control the motion of molecules in real time.

In [9], a genetic algorithm is used to search for the laser pulse that best drives an electronic wave packet to a desired target. In [10], genetic algorithms are used for chemical laser modeling.

The two problems chosen for a genetic algorithm implementation in this dissertation are representatives of many other similar problems in which there is a potential for applying such an adaptive approach. An attempt is made to answer questions such as: In which type of problems are genetic algorithms more likely to be implemented successfully? How should such an implementation proceed?

1.6.2 The Non-Adiabatic High Frequency Theory in Atomic Stabilization

Atomic stabilization by high intensity laser field is a difficult nonlinear problem which is computationally expensive. In [11], a comprehensive attempt was made to understand stabilization at high frequencies and finite pulse shapes. While the literature is rich with computational studies which were conducted to model atomic stabilization, the computations tend to be highly sensitive to the pulse shape in the case of finite pulses (pulse shapes with an on and off switching ramps). Some researchers are claiming based on their computational experiments [12,13] that atomic stabilization does not exist. Prior to the simulations covered in this dissertation, numerous computational runs by the author were performed in order to optimize the stabilization effect using two distinct laser frequencies and two different pulse shapes. This approach failed, since for each pulse shape a different stabilization structure was observed. It was evident that the simple case of one color (a single frequency) and one pulse shape is not fully understood.

These initial studies prompted the idea of optimizing the stabilization effect with a fixed pulse shape and different colors using a systematic search method rather than by trial-and-error runs, in which one needs to perform an expensive calculation each time with a different parameter. Instead, parallel genetic algorithms were used [14] automatically to find the optimal frequencies which will produce a stabilization effect. The results of these studies were in contrast to [11]. This led to isolating the conceptual error which was hidden in current literature and based on that, a new formulation to explain finite-pulse stabilization at high frequencies was developed.

In [15], the non-adiabatic high frequency theory was formulated to explain the stabilization effect. Genetic algorithms were no longer used in this study. However, the new theory was developed based on novel computational methods to solve the time-dependent Schrödinger equation. A generalization of the (t,t') method enabled the authors to simply remove the laser frequency explicitly from the calculation at high frequencies. Such new concepts can potentially be used in other related computational studies, as well as providing a new treatment which sheds light on atomic stabilization at high frequencies.

After an introduction to atomic stabilization in chapter 2, Chapter 3 describes the genetic algorithm implementation in detail. Chapter 4 presents the non-adiabatic high frequency theory, following a few introductory sections.

1.6.3 Searching Policy Space in Markov Decision Processes

Markov decision processes are used as a model to solve complex decision making under uncertainty. Problems of this type, which are of interest in robotic navigation and operations research, are known to be computationally expensive. Search methods to solve such problems have been constantly developed over the past 40 years, since the method of dynamic programming was invented [16]. Chapter 5 is an introduction to the terminology and methods used for Markov decision processes.

In this dissertation, an attempt is made to compare between the traditional algorithms used in this field, which are deterministic in nature, and genetic algorithms which are a stochastic search method. While this study was in progress, a related study [17] recommended using genetic algorithms to solve a similar model problem. However, no actual comparisons with traditional techniques were reported in their work. Before proceeding, it should be noted that in an indirect way, genetic algorithms have already been incorporated in an adaptive reinforcement learning procedure to solve, using certain approximations, some huge problems of this type [18]. In such large problems, traditional algorithms fail to work at all. Therefore, it is an interesting question to examine how well genetic algorithms perform when compared to traditional techniques in exact solutions, where both approaches are applicable. Dynamic programming is being developed to treat larger and larger problem sizes [19], building up on traditional thinking. The dissertation does not attempt to treat those problems for which dynamic programming is not yet applicable.

In [20], it is found that in contrast to [17], dynamic programming is a more efficient search technique than genetic algorithms when both approaches are applied to the same model problem. Chapter 6 describes the comparison in detail.

Chapter 2

Introduction to Atomic Stabilization

In this chapter, atomic stabilization phenomenon is introduced. Section 2.1 begins with two discoveries from 1980s laser experiments, which led to a shift in viewpoint of how atoms behave in very strong laser fields. These two effects, above threshold ionization and high harmonic generation, are important and relevant to laser-induced stabilization. Section 2.2 then illustrates how numerical techniques were first used to explain basic phenomena in high-intensity laser atom interaction, following the experiments described in section 2.1. In section 2.3, early predictions of atomic stabilization are described. These theoretical predictions initiated new experiments which are still being carried out at the late 1990s, at the Netherlands, to verify atomic stabilization. Section 2.4 describes the numerical simulations which are used to model atomic stabilization with a finite pulse (which is different from stabilization using continuous-wave laser fields). This type of stabilization is of recent interest in the field and the subject of a lively literature debate which is also referred to in section 2.3. Section 2.5 then summarizes the state of understanding prior to the new findings described in the next two chapters.

2.1 Ionization Experiments with High-Intensity Lasers

For nearly 75 years, following the discovery of the photoelectric effect in 1905 [21], a well-founded theory existed which was adequate to describe photoabsorption processes. With conventional light beams containing a low density of photons, conservation of energy of the form $E_d = E_0 + \hbar\omega$ (where E_d is the energy of the detected electron in a photoabsorption experiment, $E_0 (< 0)$ is the energy of the initial atom, and ω is the frequency of light) was successful in explaining the ionization process by using a simple picture: an electron is ejected from an atom as soon as a photon with an energy greater than E_0 is absorbed. The electron's energy is the difference between the photon's energy and the binding energy to the nucleus which the photon needs to break apart.

However, beginning around 1980, new lasers were developed that were intense enough (intensities above $10^{13} W/cm^2$) to send thousands or even millions of photons through a typical atomic cross section ($10^{-16} cm^2$) during a pulse of a picosecond or less. As a result of atomic physics experiments with these more intense lasers, a different view of photoabsorption was required. The experiments were mostly done on noble gases, such as neon or xenon, at around 15 Torr maximum pressure ($5 \times 10^{17} atoms/cm^3$). These experiments uncovered several surprising effects, which will now be described, that are relevant to atomic stabilization.

The first of these discoveries is called *above-threshold ionization* [22,23]. It was shown that an additional term should be added to the energy conservation mentioned in the previous paragraph, in case of strong laser fields. In early experiments, energy spectrum of the emitted electrons in a strong-field with intensities above $10^{13} W/cm^2$ was examined. Several peaks were seen, equally spaced by the photon energy $\hbar\omega$. This is instead of a single peak in the weak-field, which hints that the atom in a strong field is supersaturated with photons. It was noted in experiments with relatively long pulse lengths that the lowest such peaks may disappear as the intensity is raised. This effect has been attributed to an extended form of the energy conservation, which includes the *ponderomotive potential* E_p. The energy conservation now becomes $E_d = E_0 + N\hbar\omega - E_p$, where N is the number of photons absorbed and $E_p \propto I/\omega^2$ is the average energy associated with the classical oscillations of an ionized electron in the laser field. The spectroscopy of above threshold ionization peaks and their substructure is an active area of study in atomic physics.

The second of these effects is called *high-harmonic generation* [24,25]. Harmonic generation is a term from nonlinear optics meaning the coherent emission of photons with shorter wavelength than the incident photons. The new wavelengths are shorter by an integer factor compared to the incident laser's wavelength. These higher energy photons are typically produced by irradiation of special crystalline materials that can be damaged. Therefore the irradiation cannot be too intense. In contrast to normal experience in nonlinear optics, typically dealing with generation of up to third or fourth harmonics, more than 100 harmonics can be produced by a strong field which are generated by irradiation of gas atoms. This is considered a big number, since it is difficult to produce very-high harmonics. The explanation for this phenomenon is that the above threshold ionization electrons which became supersaturated with photons, rather than breaking free, are recaptured by the atom. When that happens, the electron gives up its energy all at once. The short wavelengths produced by this energy multiplier effect have valuable practical applications in studying the dynamics of molecules, surfaces and materials. Similar to above threshold ionization, high harmonic generation is an effect produced by supersaturating an atom with thousands of photons, and the basic theory of light and matter in a weak field needs to be extended since it no longer holds.

2.2 Numerical Methods to Analyze High-Intensity Laser-Atom Interaction

As was mentioned in the previous section, the phenomena which were observed in the 1980s high-intensity laser experiments could no longer be explained using existing analytical tools such as perturbation theory. Perturbation theory breaks up at around the intensity of $I = 10^{13} W/cm^2$. Since the main body of these experiments have been carried out in the intensity range between 10^{12} and $10^{15} W/cm^2$, with laser pulse durations in the vicinity of $1 - 10$ picoseconds, it was evident that a new framework for analyzing these high-intensity laser atom interaction effects is required.

Beginning in [2], computer programs were able to solve the time- dependent Schrödinger equation for certain simple potentials in low dimensions. In these cases, they permitted essentially exact analyzes of the quantum mechanical response of an electron exposed to both a strong time-dependent laser field and the static coulombic binding potential of the atom.

Non-numerical approximations are also typically incorporated into numerical solutions of the time- dependent Schrödinger equation. For example, the electron is treated as non- relativistic and spinless, and the electric dipole approximation is used. This can be justified rather easily since the optical wavelength is much larger than the size of the atom, so any spatial dependence of the electric field can be neglected, treating it as time-dependent only. A further assumption commonly made is that the electron's motion is so nearly one-dimensional in a strong laser field (along the laser field's polarization direction) that the other two dimensions can often be eliminated without harm. Laser polarization effects and angular distributions of emitted photons and photoelectrons can not be modeled that way, but in all other cases comparisons of $1D$ with $3D$ results and with experiments have shown that the one-dimensional assumption is satisfactory for accurate semi- quantitative modeling of strong-field ionization dynamics.

To illustrate the points described so far, an example is given for the case of harmonic generation. Using all the simplifying assumptions described in the previous paragraph, one can use a simple numerical technique, such as the finite-difference, to solve the one-dimensional time-dependent Schrödinger equation describing a hydrogen atom exposed to a high-intensity laser field:

$$i\frac{\partial\psi(x,t)}{\partial t} = \left[-\frac{1}{2}\frac{\partial^2}{\partial x^2} + xE_0 f(t)\sin(\omega t) - \frac{1}{\sqrt{x^2+1}}\right]\psi(x,t) \tag{2.1}$$

Note that atomic units ($\hbar = m = e = 1$) are used, and the potential is a "soft coulomb" in order to avoid a division by zero using a computer. Analysis such as this one will be covered in great detail in section 2.4, for the atomic stabilization case. Here, after numerically solving for the wavefunction $\psi(x,t)$ on a one-dimensional grid containing 1000 points with a spacing of $dx = 0.2$, the expectation value of the position was numerically extracted from the wavefunction, and the Fourier transform of the resultant oscillatory motion is shown in figure 2.1. The laser parameters which were used in this simulation are a wavelength of $\lambda = 7800\mathring{A}$ and an intensity of $I = 2.0 \cdot 10^{13} W/cm^2$ (above the perturbation limit). The first few harmonics correspond to peaks in the graph. Note that only odd multiples of the fundamental frequency are generated due to symmetry.

Among the leading figures to perform numerical studies, a work which began during the 1980s for understanding high-intensity laser ionization processes, are J.H. Eberly from the University of Rochester, S. Geltman from the University of Colorado at Boulder and K.C. Kulander from the Lawrence Livermore National Laboratories. Work is recently reviewed in Protopapas et al. 1997].

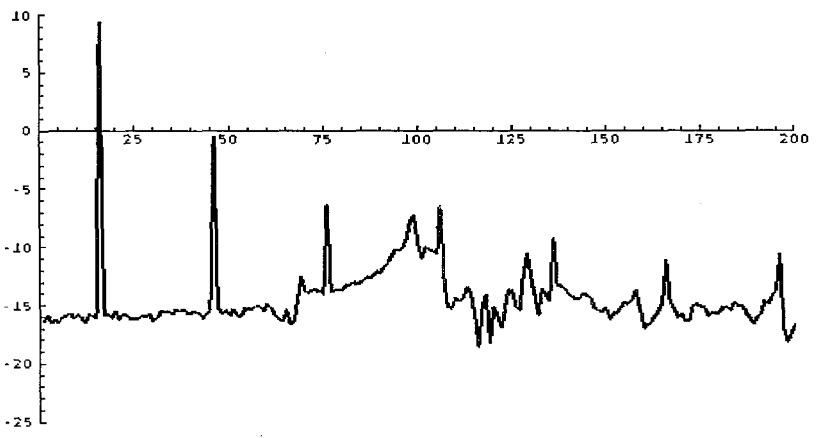


Figure 2.1: Harmonic Generation: an example of the electron dynamics in an intense laser field. In the example above, a finite-difference scheme was used to solve the equation and find the harmonics that were created. The calculations are based on a one-dimensional model of a hydrogen atom that interacts with a high-intensity laser field.

2.3 Atomic Stabilization: From Predictions to Experiments

Unlike other strong field effects, such as above threshold ionization and harmonic generation, in which the theory was required to further develop in order to explain the experimental observations, the opposite is true in the case of atomic stabilization. First came the theoretical predictions and only later the experiments to test these predictions.

In ionization at low laser intensities, the probability that an electron is stripped from the core atom and becomes ionized linearly increases with intensity. This linear behavior can be verified using perturbation theory. The term *atomic stabilization* refers to the possibility that at very high intensities, this pattern reverses itself. As the laser intensity is increased above a critical point, the ionization rate slows down dramatically, the laser itself starts working against the ionization and a new "stabilized" atomic configuration emerges.

In the early predictions of atomic stabilization [27,28], a laser field of asymptotically high frequency and high intensity was assumed. It was shown that under such conditions, the atom remains stable against both ionization and bound-bound transitions because the Hamiltonian that describes the system (including the field-atom interaction) is well represented as being time-independent in a convenient reference frame. Gavrila and co-workers then went on to develop a high-frequency Floquet theory to explain the stable atomic configuration which emerges. At the same time, numerical studies similar to the ones discussed in the previous section were performed and confirmed the stabilization effect. In the next section, such a numerical approach to study atomic stabilization will be described in detail.

M.Gavrila, from the FOM Institute for Atomic and Molecular Physics in Amsterdam, the Netherlands, formulated and organized for several years the majority of work on the theory of atomic stabilization. Much of these studies can be found in his book, *Atoms In Intense Laser Fields* [29]. On the numerical side, especially with regards to the solution of the time- dependent Schrödinger equation in order to examine stabilization for short finite laser pulses, Q. Su and J.H. Eberly have been actively involved in this field [11,30]. S.Geltman [13] recently commented on this work, critically claiming that non-monotonic variation of ionization probability with laser intensity is a normal expectation in any strongly coupled quantum system. However, the vast majority of researchers in the field believe that the effect has been numerically predicted, for finite pulses as well, beyond all doubt (as will be seen in this dissertation which supports [11], much of the confusions arise due to the strong sensitivity of stabilization to the pulse shape). Other contributions to the field are coming from various groups in England and the U.S. [31].

On the experimental side, ongoing work by H.G. Muller and colleagues in the Netherlands exhibit results which are so far consistent with predictions for atomic stabilization. The first experimental evidence for atomic stabilization was reported in [32]. In these experiments the $5g$ circular Rydberg state in neon was prepared using a 1-ps, 286-nm wavelength, circularly polarized laser pulse, and the photoionization yield due to a second, 620-nm laser pulse of either 100-fs or 1-ps was studied. The decrease of the single-photon ionization signal can be explained by the population remaining behind in the 5g state, thus strongly indicating stabilization of this state. In [33], measurements are done using three laser pulses. The first laser pulse excites the circular $5g$ state from the neon ground state. The second pulse is the intense light pulse that leads to ionization (or lack of ionization due to stabilization). The third pulse is a long low-intensity pulse ionizing all of the remaining $5g$ population. The photoionization yields of these three pulses are detected and separated by electron spectroscopy. This extended experimental study was done to rigorously rule out alternative explanations, such as transitions to long-lifetime states that could act as ionization traps. Both the photoionization yield and the remaining population in the $5g$ state were measured. It was found that the photoionization yield does not increase when the pulse peak intensity is increased above $60TW/cm^2$, and that a large fraction of the population remains in the $5g$ state instead. These results are consistent with predictions for atomic stabilization.

2.4 Modeling Atomic Stabilization

In this section, numerical techniques which are used to model atomic stabilization with a finite pulse are introduced. The detailed description will be helpful for the continuation of this modeling in the next chapter.

The starting point for any model of atomic stabilization is the time-dependent Schrödinger equation,

$$i\hbar\frac{\partial\psi}{\partial t} = H\psi \tag{2.2}$$

where ψ is the wavefunction, H is the Hamiltonian and $\hbar$ is a universal constant. There is no loss of generality if the time-dependent Schrödinger equation is solved in atomic units ($\hbar = m = e = 1$).

In a one-dimensional model for the laser-atom interaction, which are justified based on the simplifying assumptions described in section 2.2, the Hamiltonian $H(x,t)$ contains a nonlinear term to describe the linearly polarized oscillating electric field $E(t)$ along with the kinetic energy (second derivative in space) and the potential energy $V(x)$:

$$H(x,t) = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + V(x) + xE_0 f(t)\sin(\omega t) \tag{2.3}$$

To eliminate any external effects which might introduce complications to the system, such as infinite number of bound states in a coulomb potential, it was decided to work with a short-range potential which can be tuned to support a single bound state. This potential, namely $V(x) = -V_0 \exp(-x^2/x_0^2)$, is a model for a negative atomic ion. The potential parameters were set at $V_0 = 0.18$ and $x_0 = 7.0225$, one of several possible combinations so that the potential supports only a single bound state. The tuning procedure is similar to what was done in [34]. In order to compute the initial condition, which is the wavefunction ψ at this bound state, any time-dependent expression is taken out of the time-independent Schrödinger equation and the time-independent Schrödinger equation is solved:

$$\left[-\frac{1}{2}\frac{\partial^2}{\partial x^2} - V_0 \exp\left(-\frac{x^2}{x_0^2}\right)\right]\psi = E\psi \tag{2.4}$$

This is an eigenvalue problem. The eigenfunction corresponding to the only negative eigenvalue (namely, the bound state) constitutes the initial wavefunction, which is the ground state of the system before the laser is switched on to start the ionization process. For the potential function parameters chosen above, the ground state energy (the negative eigenvalue) comes out to be -0.0933 au. The eigenvalue problem was solved using a standard Eispack routine which computes the eigenvalues and eigenvectors of a symmetric tridiagonal matrix [35].

Substituting (2.3) into (2.2), the time-independent Schrödinger equation with an exponentially decaying short-range potential becomes:

$$i\frac{\partial\psi(x,t)}{\partial t} = \left[-\frac{1}{2}\frac{\partial^2}{\partial x^2} + xE_0 f(t)\sin(\omega t) - V_0 \exp\left(-\frac{x^2}{x_0^2}\right)\right]\psi(x,t) \qquad (2.5)$$

This equation will be numerically solved, using finite-difference and Fourier methods, for several laser intensities, i.e. by setting ($E_0 = \sqrt{I_i}$), $i = 1, 2, ...$ and using the laser frequency ω as the input parameter.

Now it is necessary to choose pulse shape $f(t)$ and pulse duration, unless it is decided to vary the shape as a parameter (along with the frequency). As in [11], the commonly used procedure is to switch the laser on and off smoothly, according to $f(t) = \sin^2((\pi/2) \times t/T_1)$ for $0 \leq t \leq T_1$, $f(t) = 1$ while the laser is on ($T_1 \leq t \leq T_2$) and switching off according to $f(t) = \cos^2((\pi/2) \times t/(T_3 - T_2))$ for $T_2 \leq t \leq T_3$. In the numerical simulations reported here, 3-10-3 pulses (3 cycles turn-on, 10 cycles fully on, 3 cycles turn-off) were used. The time period of each cycle is $2\pi/\omega$ (ω is the laser frequency).

Care should be taken with respect to the boundary conditions. In [11,30] simulations were done specifically for a fixed high frequency value which promoted the ground state well into the continuum. For that particular case, the grid was large enough and no forcing boundary conditions were needed. However, in the current studies which include a large range of frequencies, the ground state of the grid walls was found to interfere with the calculations below a certain frequency threshold. This caused a false stabilization peak to appear below this threshold (presumably, a numerical artifact since it disappears as we decrease dx). This prompted the use of either huge grid sizes, which is least desirable, or any other method such as mask functions, absorbing potentials, or a dilation transformation so that the time-dependent wavefunction will vanish in the limits as $x \to \pm\infty$ at each time step in the calculation. Mask functions were chosen, taking a $\sin^{0.2}(x)$ shaped mask function and multiplying with it the wavefunction at the grid edges. In addition, the possibility for a false peak should not be forgotten when working with low frequencies close to the top of the potential well.

Finally, the calculations were done with the following set of numerical parameters in addition to the ones mentioned above. For the spatial grid, $N_x = 1000$ points are taken (or $N_x = 1024$ when Fourier representation is employed). For the mask function region, 100 grid points are used on each side. Grid spacing is $dx = 0.1$. For the time grid, $N_t = 16,000$ steps are used corresponding to a 3-10-3 pulse (total of 16 cycles) in which each cycle contains 1000 time steps. Time spacing is determined by the laser frequency, so that $dt = T/ndt$ where T is the time-period of a cycle obtained from the frequency and $ndt = 1000$ time steps per cycle. Unless noted otherwise, atomic units are used for all these parameters.

Figure (2.2) shows the end-of-pulse ionization vs. intensity plots (stabilization curves) for two different laser frequencies (or, wavelengths). The curves were obtained by solving the time-independent Schrödinger equation for $\psi_f(x, T_d)$, where T_d is the duration of the pulse. After the pulse duration is over, ψ_f is projected onto ψ_i so that $p0 = \mid < \psi_f \mid \psi_i > \mid^2$ is the probability of the final wavefunction being in the ground state. $1 - p0$ is plotted as a function of intensity, after solving the time-independent Schrödinger equation repeatedly for different intensities, as a measure of the ionization rate. Full ionization is achieved at 1.0. For low intensities, one can see the expected linear response which is also predicted using perturbation theory. For high intensities, however, a stabilization structure can be seen whereby the laser interferes with the ionization process and the ionization is suppressed. Perturbation theory breaks down and fails to explain the behavior at such high intensities.

2.5 Understanding Atomic Stabilization

In this section, some of the key points based on which atomic stabilization can be explained are examined. Finally, some aspects which were well understood and some which were not that well understood prior to this dissertation are stated.

It is instructive to first start with a simple analytical analysis, in which the electron interacts only with the laser light, to physically understand the motion of an electron inside an electric field. Examining this zero-order scenario is essential for

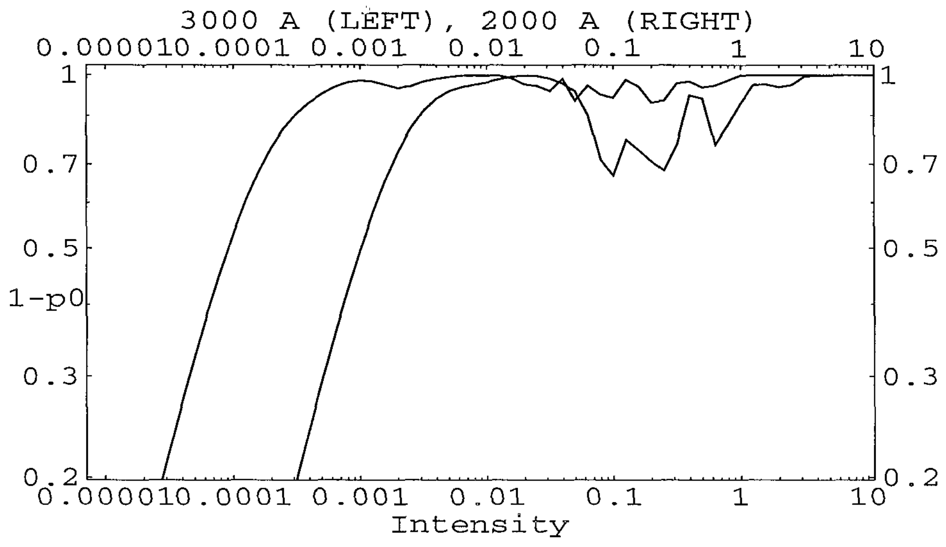


Figure 2.2: End of pulse ionization 1-p0 as a function of laser intensity, for two different frequencies. The 16-cycle pulse that was used consists of a 3-cycle smooth (sine-squared ramp) turn-on, a 3-cycle smooth turn-off and 10-cycle constant laser field inbetween. Ionization suppression at high intensities is evident in both cases. Left curve corresponds to $\omega = 0.152$ ($\lambda = 3000\mathring{A}$) and right curve corresponds to $\omega = 0.228$ ($\lambda = 2000\mathring{A}$). In atomic units, $I = 1.0$ corresponds to $I = 3.5\times10^{16}W/cm^2$.

gaining intuition and constructing the proper foundations being used in what follows.

In a classical picture, a single one-dimensional electron in an electric field (no coulomb potential) obeys the Newtonian equation of motion:

$$m\frac{d^2x}{dt^2} = -eE_0f(t)\sin\omega t \tag{2.6}$$

For simplicity, special units in which $m = e = 1$ are used (see previous section) and the laser light is assumed to be steady with no pulse envelope. Integrating twice, one gets:

$$x = \frac{E_0}{\omega^2}\sin\omega t \tag{2.7}$$

Therefore, the electron is swinging in an oscillatory motion around the center with an amplitude (known as the *quiver amplitude* in the literature) of $\alpha_0 = E_0/\omega^2$. This motion is often called *quiver motion*, since for a realistic choice of parameters such as $\lambda = 7800 \AA$ and $I = 2 \cdot 10^{14} W/cm^2$, one gets $\alpha_0 \sim 80a_0$ which means oscillations with an amplitude around 80 times the size of an atom with a most energetic motion. This simple dynamical picture forms the basis of understanding of many strong-field phenomena. In high-intensity laser experiments, performed on noble gases at around 15 Torr maximum pressure ($5 \times 10^{17} atoms/cm^3$), such an oscillatory motion of the electron contributes to the detected photoelectron energy spectra. It is precisely this motion which is responsible for adding the ponderomotive potential which was discussed in section 2.1.

The above analysis motivates the use of a frame of reference transformation, in which the electron becomes the center of motion rather than the nucleus. This is done by the transformation $x \to x + \alpha(t)$, where $\alpha(t) = \alpha_0 \sin \omega t$. It is known as the Kramers-Henneberger (K-H) transformation, under which the Hamiltonian of section 2.4 becomes:

$$H(x,t) = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + V(x + \alpha(t)) \tag{2.8}$$

i.e., it describes the motion of the electron in an oscillatory potential. In the high frequency limit, this potential may be replaced by its time average:

$$V_{K-H}(x) = \frac{w}{2\pi}\int_0^{\frac{2\pi}{\omega}} V(x + \alpha(t))dt \tag{2.9}$$

and the remaining Fourier components of $V(x + \alpha(t))$ are treated as a perturbation. When α_0 is large, the effective potential has two minima close to $x = \pm\alpha_0$. The corresponding wavefunctions of the bound states are centered near these minima, thus exhibiting a *dichotomy.*

In figure (2.3), it is seen how the cycle-averaged, effective potential in the K-H frame develops a double well as the intensity increases.

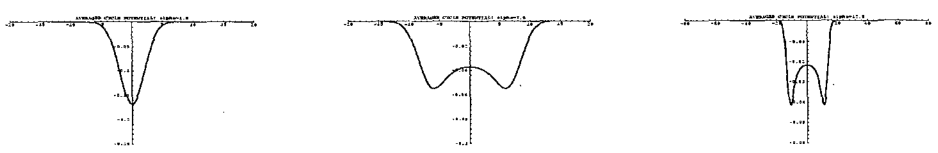


Figure 2.3: Development of a double well in the effective cycle-averaged potential V_{K-H}. α_0 is increased, from left to right: $\alpha_0 = 1.0$, $\alpha_0 = 7.5$, $\alpha_0 = 12.5$

In figure (2.4), a map of the eigenenergies of the averaged cycle potential used throughout this dissertation is plotted as a function of the quiver amplitude α_0.

Figure (2.4) is an attempt to provide a general explanation for atomic stabilization with finite pulses. For short frequencies, a multiphoton process (such as a 3-photon promotion from the ground state) is the mechanism for ionization. As seen in figure

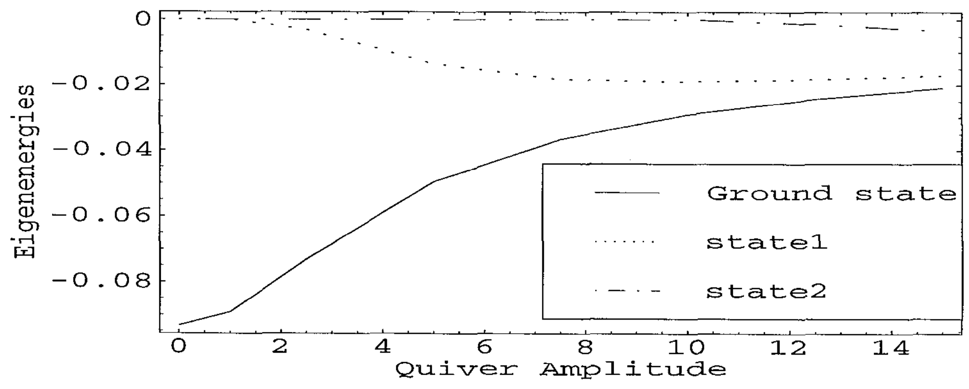


Figure 2.4: Map of the eigenenergies of the cycle-averaged potential used throughout this dissertation. Formation of laser-induced states (dashed lines) in addition to the initial ground state, as the quiver amplitude α_0 is increased, can be seen.

(2.4), as α_0 grows, new laser induced states are formed which are responsible for the interruption in the ionization process. This explains atomic stabilization at low frequencies (e.g., [34]). However, this picture does not provide any explanation of why atomic stabilization occurs at higher frequencies, in which a single strong photon promotes the ground state well above the potential well, into the continuum. In [11,30], attempts are made to relate this type of stabilization to Gavrila's Floquet theory which is valid for infinite pulses (high frequency continuous-wave laser fields). Chapter 4 provides a proper explanation for atomic stabilization at high frequencies for finite pulses.

Chapter 3

Genetic Search for the Optimal Frequency to Stabilize Atoms

This chapter follows step by step, starting from the point reached at the end of section 2.4 (figure 2.2), the details of the computational study which led to illustrate the true physical interpretation surrounding atomic stabilization at high frequencies. First, the genetic algorithm succeeds in locating an optimal frequency to stabilize atoms. Then, it is found that stabilization does not increase as the frequency is increased. Finally, convergence to a high-frequency limit in the stabilization structure is obtained. Chapter 4 will provide the explanation to the atomic stabilization structure at high frequencies.

3.1 Introduction

The first part of this chapter shows the procedure used to find the optimum frequency via genetic algorithms, for which an atomic stabilization effect is maximum. It will be found that this approach actually leads to important results concerning the stabilization structure produced. It is possible to search frequency space to locate this optimum based on the following fitness function:

$$f(\omega) = (p0_{I_{i+1}} - p0_{I_i}) + (p0_{I_{i+2}} - p0_{I_i}) + ... + (p0_{I_{i-1}} - p0_{I_i}) + (p0_{I_{i-2}} - p0_{I_i}) + ... \quad (3.1)$$

The quantities in this expression, resulting from numerically solving the time-dependent Schrödinger equation as was as done in section 2.4, are calculated in parallel using the methodology outlined in section 1.4.3. A systematic search for the optimal frequency ω can then be conducted. Since the search is computationally intensive, three different optimization techniques are compared so that the most efficient one can be recommended for future studies. The three methods are simple genetic algorithms, simulated annealing and micro genetic algorithms.

Section 3.2 describes the genetic algorithm implementation. Section 3.3 reports the results of this study and the comparison between the three optimization methods. Section 3.4 examines the implication of the results, in relation to other studies which were reported in the literature on the same problem and parameters. It then explains the conceptual error in [11] and gives the correct calculation which reaches the high frequency limit. Section 3.5 concludes this chapter with an outlook towards chapter 4, which provides the explanation to the new findings.

3.2 Genetic Algorithms on a parallel platform

This section follows the one-frequency modeling which was outlined in section (2.4) and extends it to a many-frequency modeling, using parallel micro genetic algorithms, to locate the optimal frequency that achieves the most pronounced stabilization effect. Writing the expression (3.1) in more detail, the goal is to maximize:

$$f(\omega) = -N \times p0_{I_i} + p0_{I_{i+1}} + p0_{I_{i+2}} + ... + p0_{I_{i+N/2}} + p0_{I_{i-1}} + p0_{I_{i-2}} + ... + p0_{I_{i-N/2}} \quad (3.2)$$

given $p0_{I_i}$ (I_i is the intensity and $p0$ is the projection to the ground state) and the projections to the ground state of the neighboring intensities, namely $p0_{I_{i+1}}$, $p0_{I_{i+2}}$, ... , $p0_{I_{i-1}}$, $p0_{I_{i-2}}$, ... , where N is the number of neighboring intensities. This quantity is greatest for the most pronounced minimum in the curve, centered at $I = I_i$. It is frequency dependent and maximizing this function guarantees to locate the optimal frequency ω for which the minimum caused by stabilization is most pronounced. Possible ways to perform the search for optimal frequencies are trial-and-error, interval-halving and random search techniques.

There are several reasons why a genetic algorithm was chosen for the implementation. First, it is impossible to know a priori how many minima and maxima are hidden in the fitness function. By trial and error runs it was possible to observe that the function is not simply monotonic and therefore there are optimal frequencies. However, there is a likelihood to miss the best frequency in an unexpected place, since there might be several resonant frequencies. Second, it is laborious and a waste of resources to perform a trial-and-error search when the evaluation of the fitness function is costly. An automated search would be beneficial. Third, in simulations of this kind, an extension to several other laser parameters such as pulse shape and pulse duration should be taken into account for future implementations. While interval-halving and similar techniques (Fibonnaci search, golden search) are simple ways to perform an automated search and in that regard offer an ideal possibility to meet the second requirement, they are extremely inefficient in multi-modal functions (first requirement) and more than one parameter (third requirement).

Since the evaluation of the fitness function is costly, a variant of genetic algorithms which can offer a saving in the number of function evaluations is most desirable, particularly for the simulations in this paper. With micro genetic algorithms (see section (1.4.2)) it might be possible to reduce the number of function evaluations. Therefore, the micro genetic algorithm was implemented in addition to the simple genetic algorithm and simulated annealing for the comparative study done in the next section.

In evaluating the fitness function, it is possible to perform the calculation for several intensities all at once. A parallel platform is most suited for this task, since different intensities can be distributed among several processors that perform the same calculation, each processor working on a different point or set of points in a stabilization curve.

Message passing interface can be used to communicate between the different processors and is currently available on almost all parallel platforms. Since the calculation for each intensity is done independently and only at the very end the fitness function is calculated based on information from all processors, it is desirable to reduce the amount of communication. In this problem, it is possible to refrain completely from any need of communication, assuming that enough processors are available to handle each point in the minimum and surrounding intensities. Therefore, each intensity is assigned to a single processor and a synchronizing barrier makes sure that all processors are done calculating their end-of-pulse probabilities before proceeding onwards. Only then the master processor (proc. 0) collects the results from a file and calculates the fitness function. This procedure is a particular implementation of the master-slave prototype outlined in section (1.4.3).

The various steps in the simulation are as follows:

1. Generate an initial population of frequencies $\omega_1^{(0)}, \omega_2^{(0)}, \omega_3^{(0)}, \ldots$

2. For each $\omega_j^{(0)}$ in the population, solve the time- dependent Schrödinger equation in parallel on (N+1) processors for (N+1) given intensities, I_i,I_{i+1},I_{i+2}, ... ,I_{i-1}, I_{i-2}, ... , to obtain end-of-pulse ionization probabilities $p0_{I_i}$, $p0_{I_{i+1}}$, $p0_{I_{i+2}}$, ... , $p0_{I_{i-1}}$, $p0_{I_{i-2}}$, ... and evaluate the fitness function $f_j^{(0)} = -N \times p0_{I_i} + p0_{I_{i+1}} + p0_{I_{i+2}} + \ldots + p0_{I_{i-1}} + p0_{I_{i-2}} + \ldots$ serially on the master processor.

3. Build next generation $\omega_j^{(1)}$ (in general $\omega_j^{(k)}$ for the kth generation) using crossover and mutation and evaluate new fitness function $f_j^{(1)}$.

4. Stop when $f_{best-so-far}$ does not improve substantially after several generations. Print ω^* corresponding to f^*.

5. Go to step 3.

To summarize, in order to maximize the fitness function, p0 is calculated for each intensity. This operation amounts to solving a single partial differential equation.

For a 16-cycles pulse, this can be achieved in approximately 150 seconds on an Ultra-Sparc. Working on the UC Berkeley network of workstations, which is a collection of Ultra-Sparcs hooked together, it is possible to calculate all p0's at once so that each evaluation function costs around 150 seconds. For the genetic implementation, a genetic algorithm driver written in FORTRAN was used [10]. While a number of processors are used to calculate the fitness function, the master processor by itself is used to advance the simulation from one generation to the next until convergence is achieved.

3.3 Experiments: Genetic Algorithms vs. Simulated Annealing

The previous section described one possible implementation for searching frequency space, which seems to be most suitable for the given task. Beginning with the final results of the search (which is independent of the chosen method for the implementation), this section compares simple genetic algorithms with micro genetic algorithms and simulated annealing.

To begin with, final results from the genetic algorithm search are reported. The genetic algorithm was tested for two cases to identify the optimal frequency to achieve maximum stabilization. In the first case, the goal was to find maximum stabilization around $I = 0.1$ au ($I = 3.5 \times 10^{15} W/cm^2$). Surrounding intensities were taken to be at $I = 1.5 \times 10^{15} W/cm^2, I = 2.5 \times 10^{15} W/cm^2, I = 4.5 \times 10^{15} W/cm^2, I = 5.5 \times 10^{15} W/cm^2$. In the second case, the optimal frequency for finding maximum stabilization around $I = 0.2$ au was calculated by the genetic algorithm. The frequency region is extended between $\lambda = 1500\AA$ to $\lambda = 3500\AA$, since above that would promote the ground state to below the top of the potential well and below that the stabilization effect disappears. The results in figures 3.1, 3.2 were obtained after 250 evaluation functions of the genetic algorithm run, when strict convergence was achieved. This was done with a population size of $N = 50$, stopping the calculation after 5 generations. Genetic algorithm parameters were set to: crossover percentage of 0.5, mutation percentage of 0.02, tournament selection and elitism was used (best individual replicated into next generation). The frequency region between $\lambda = 0$ to $\lambda = 3500\AA$ was divided into 1024 equal segments. In both figures 3.1 and 3.2, the reported optimal frequency was obtained after repeating the computational experiment for ten times.

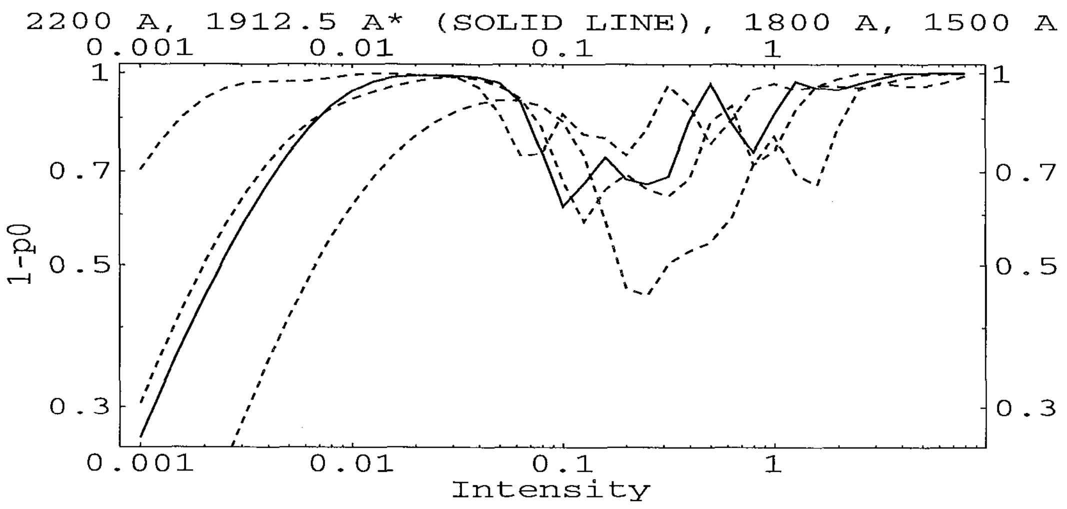


Figure 3.1: End of pulse ionization as a function of laser intensity, for several distinct frequencies. Same parameter values as in figure 2.1 apply. $\lambda = 1912.5\mathring{A}$ corresponds to the best frequency the genetic algorithm found so that a stabilization structure surrounding $I = 0.1$ can be observed.

Several observations are important before initiating a comparative study between different stochastic search strategies. First, with only 50 function evaluations, near optimality was achieved in all repeated runs. That is, after 10 consecutive runs of the genetic algorithm for the second test-case (figure 3.2), the average of the best-so-far-fitness was 0.544 corresponding to the frequency of $\lambda = 2223.4\mathring{A}$ while the lowest and highest values of the best-so-far-fitness corresponded to the frequencies $\lambda = 2209.7\mathring{A}$ and $\lambda = 2242.9\mathring{A}$, respectively. This difference is of no importance from the experimental point of view. Therefore, strict optimality is not always necessary

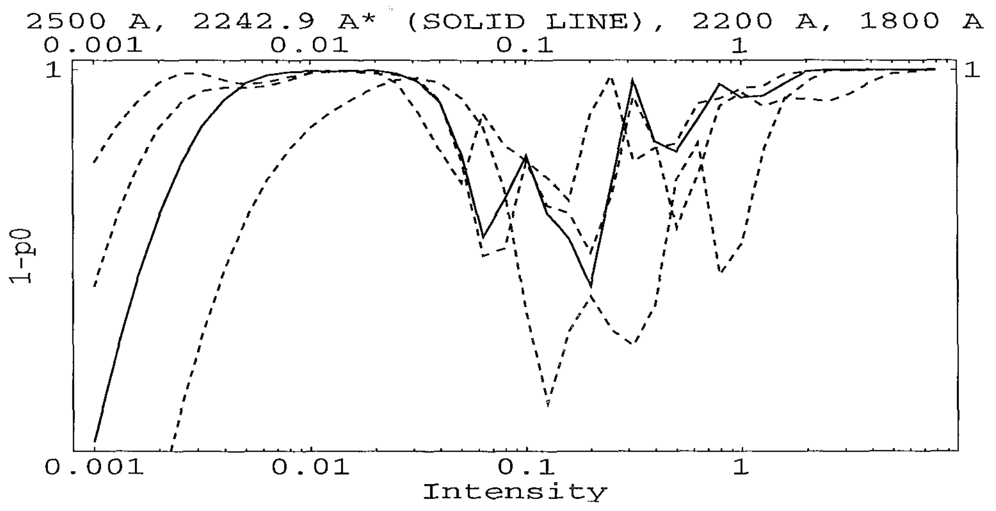


Figure 3.2: Same as figure 3.1, for a different surrounding intensity. $\lambda = 2242.9\mathring{A}$ corresponds to the best frequency the genetic algorithm found for a stabilization structure surrounding $I = 0.2$.

and stopping the calculation after only 50 function evaluations is satisfactory. Second, each function evaluation takes several hours on the parallel platform which was used. While conducting a laboratory experiment, it can not be afforded to spend several days performing repeated runs in the laboratory; instead, the aim will be to use as few function evaluations as possible along with a scheme which is reliable enough to meet its purpose. However, for the comparative analysis conducted in this paper, numerous repeated runs with the genetic algorithm were performed and 664 distinct function evaluations were gathered. This enabled to plot the fitness function (figure 3.3) and continue the handful of runs needed for the statistical analysis using a fourth-order interpolating polynomial, after it was checked that identical results are achieved using the parallel platform and the interpolating function for performing fast function evaluations. It is now possible to check whether micro genetic algorithm or simulated annealing can offer a reduction in computational cost compared to the simple genetic algorithm which is used.

In figure 3.4, micro genetic algorithm performance is compared with both the simple genetic algorithm and simulated annealing for figure 3.2. The number of function evaluations is taken as a measure of computational effort and the best-so-far fitness is the measure of performance. Average performance at each generation is of no interest in micro genetic algorithms, therefore the best-so-far-fitness is the appropriate criterion for the comparison. Population size for the micro genetic algorithm was chosen to be 5 along with a crossover percentage of 0.5. In the micro genetic algorithm, no mutation is performed since a random shuffle guards from fixation. For the simulated annealing approach, a package based on [36] was used. The simulated annealing driver was attached to the stabilization problem of this paper with initial values recommended for the test function in [36]. Then, the parameters were further tuned to best locate the optimal frequency which was achieved by the genetic algorithm, starting from an initial guess of $\lambda = 1500\mathring{A}$. Simulated annealing parameters were set to: initial temperature of 5.0, with a decline in temperature T according to $T(i+1) = RT \times T(i)$, $RT = 0.85$, where i is the ith iteration (each iteration containing 5 function evaluations with the current temperature). With a simulated annealing, unlike genetic algorithms, one starts from a single initial guess within the frequency range. Therefore, 50 runs were performed after dividing the region into 50 equally spaced points so that in the end, all parts of the region will be taken into account.

Examining the average and standard deviation of the 3 methods (figure 3.4), it is seen that they are all reaching the same optimum value in about the same speed and the differences are small. There is no gain in applying micro genetic algorithms for this particular function, although it might be beneficial to implement this scheme along with the simple genetic algorithm for other functions. Simulated annealing also performs well for this particular function. Simple genetic algorithm is most reliable (least standard deviation) and can be safely used to search for the optimal frequency in a systematic way.

3.4 Existence of a High Frequency Limit

The results of the final section indicate that an optimal frequency exists for stabilizing atoms surrounding a given intensity. This frequency is not necessarily high. In figure 7 of [11], it is observed that as the frequency gets higher the degree of stabilization increases. There is no indication of an optimal frequency, and in particular the plot is claimed to be true for any given intensity. Obviously, there is a disagreement

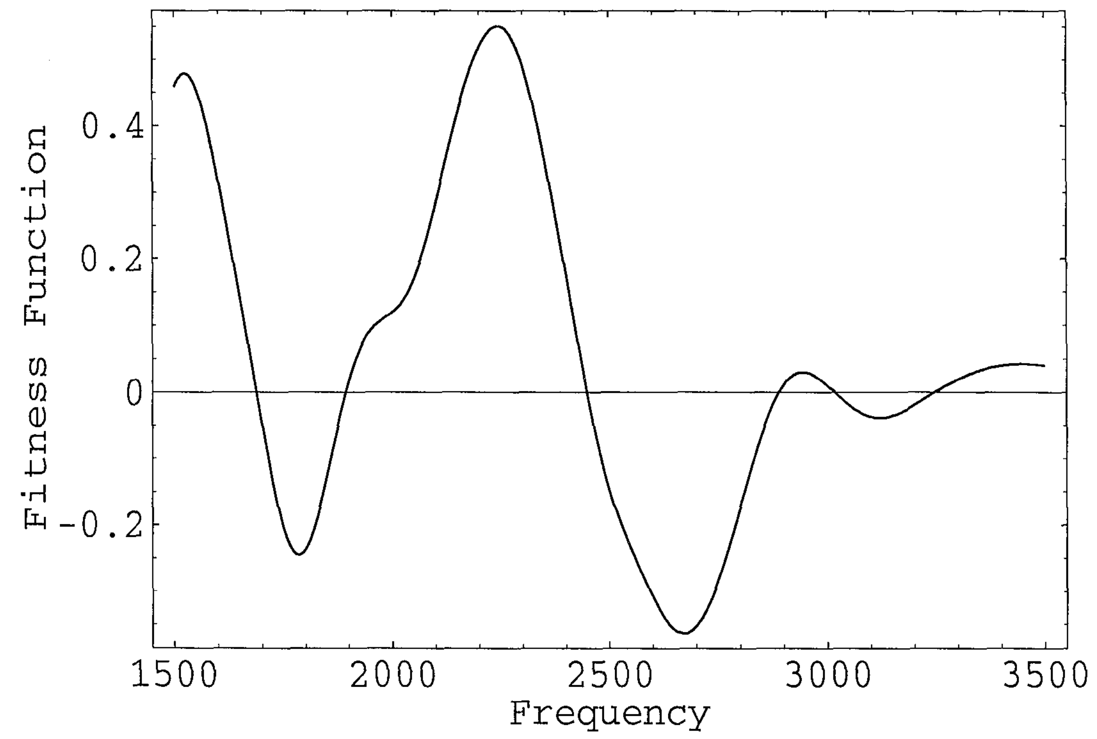


Figure 3.3: A plot of the fitness function, which was not known a priori, based on 664 function evaluations gathered in the course of the genetic algorithm implementation. It is seen that the function is multi-modal in the desired frequency range, and can be considered continuous to a good approximation. Therefore an interpolating function can be used in further function evaluations which are necessary for a comparative analysis.

which needs to be resolved.

In this section, in order to resolve the disagreement, a study is conducted to replicate Su's results. Using Su's potential and parameters, identical results are obtained. It is found that Su's figure 7 is based on a limited set of data points which is misleading and do not indicate the correct behavior at all. Su's figure 7 is recalculated with the full set of data points, which agrees well with the results of the genetic algorithm

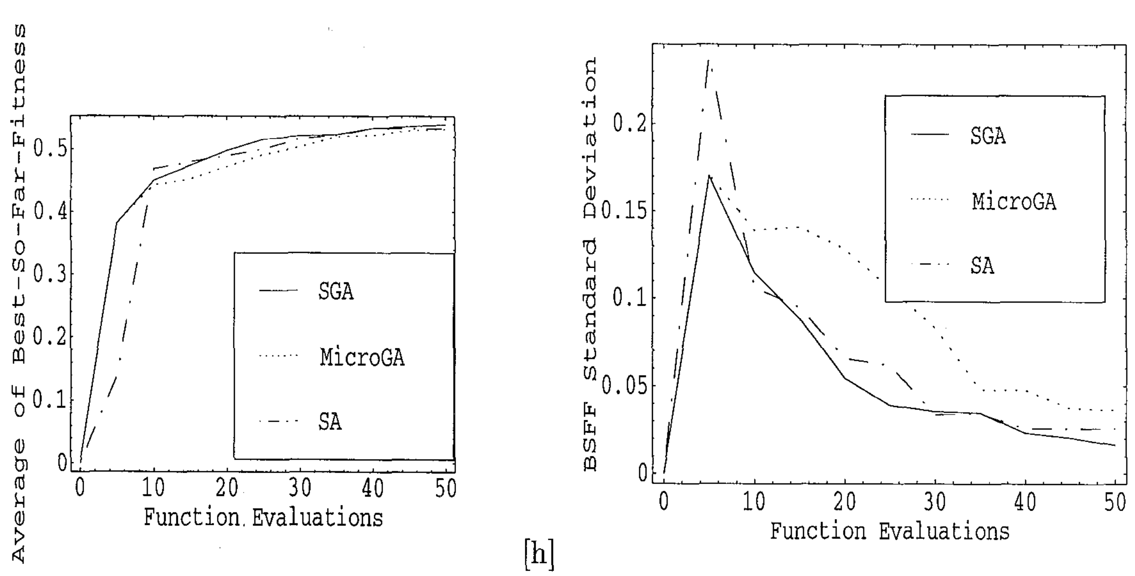


Figure 3.4: Performance comparison of simple genetic algorithm (solid line), simulated annealing (dashed-dotted line) and micro genetic algorithm (dotted line) to find the optimal frequency for the bottom case in figure 3. The average and standard deviation of 50 consecutive runs in each method was taken, and performance of the 3 methods based on these measures show they are almost equally successful in finding the optimum. For the particular function to be optimized in this paper (figure 3.3), the micro genetic algorithm does not seem to result in a reduction of computational cost.

in the previous section. A conceptual error in performing the calculation with only a single intensity is found to obscure the exact behavior. Correcting this error, a high frequency limit for stabilization with a finite pulse is found to exist.

The starting point of the comparison with Su's results is figure 3.5, which is identical to figure 5 in [11]. In this figure, a stabilization structure is examined for the frequency $\omega = 1.0au$. Note that Su is using a one-point delta potential instead of the short-range potential used in sections 2.4, 3.2, 3.3. The exact form of the one-point delta potential is:

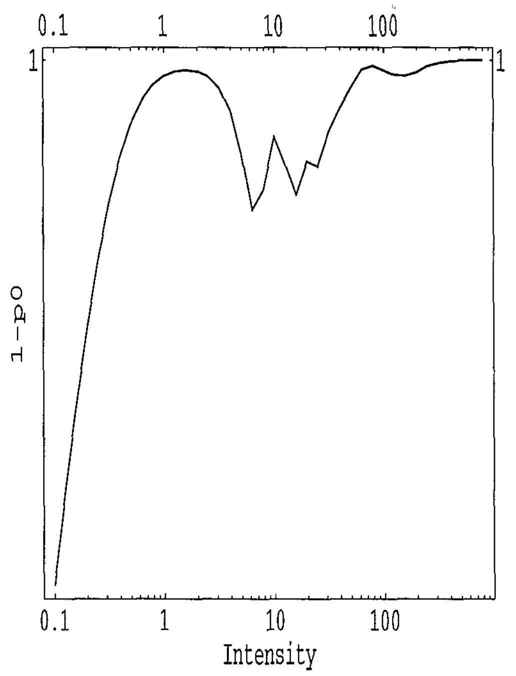


Figure 3.5: Replicating figure 5 of [11], stabilization structure for the frequency $\omega = 1.0$. Original on the right, replica on the left. Same parameters as in the reference were used.

$$V(x) = -B\delta(x) \tag{3.3}$$

Where $B = w \times D$, w is the width of a square well and D is the depth. In the limit $w \to 0$ the square well becomes a delta potential and the potential supports a single bound state energy give by $W_b = -B^2/2$. Only one grid point is used to describe this delta potential.

Since both potentials bound a single state, there is no qualitative difference between them. Since the potential well extends higher up in Su's potential, higher frequencies are needed for the stabilization, but the same effect is obtained in both cases. Note the similarities between this figure and figure 2.2 in the previous chapter.

Next, figure 3.6 is a replica of Su's figure 7. In this figure, it is seen that the degree of stabilization increases as the laser frequency is increased. This also seems to fit the theoretical predictions in [29] for the continuous-wave laser fields (with no ramp). However, in this figure only one intensity is used ($I = 16.0au$). This single intensity can not be used to conclude something about the stabilization effect, since the stabilization curves are shifting to the right as the frequency goes up. Figure 3.7 unveils the correct behavior. It is clear that for $I = 16.0au$ the ionization probability decreases, with no connection to the stabilization effect.

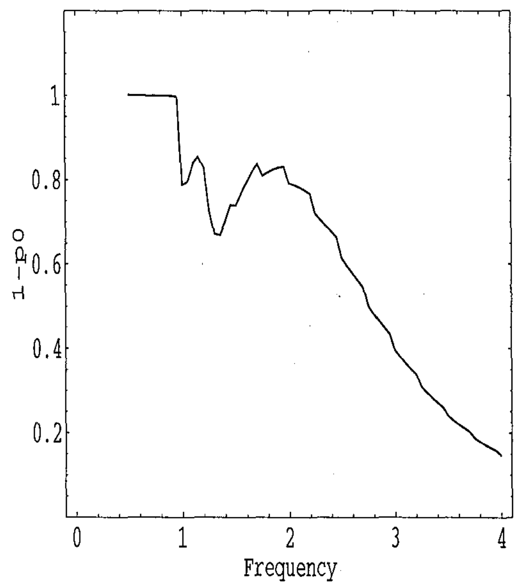


Figure 3.6: Replicating figure 7 of [11]. Original on the right, replica on the left. Ionization probability $(1 - p0)$ is plotted vs. the frequencies (in au) for a single intensity, that of $I = 16.0$. Same parameters as in the reference were used.

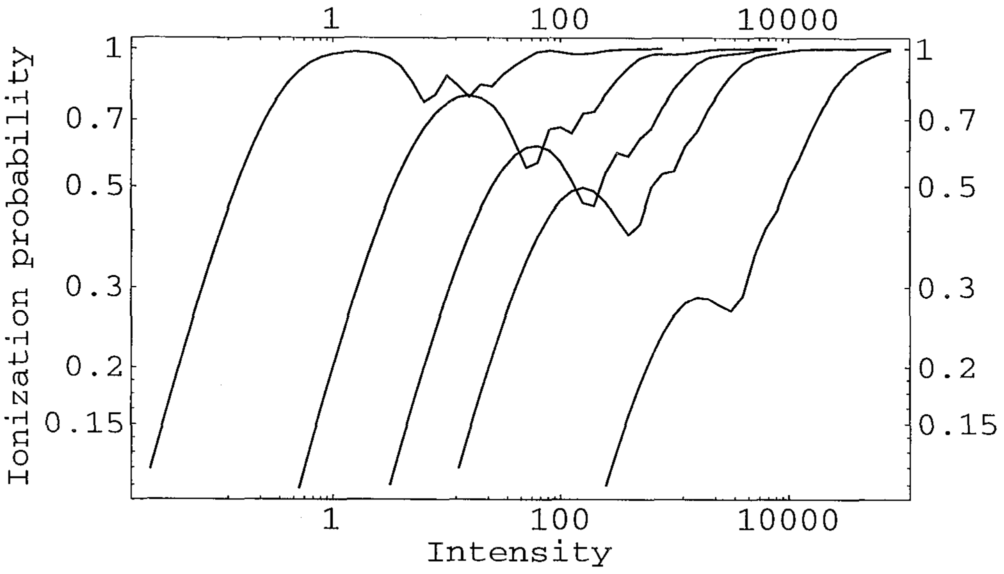


Figure 3.7: The stabilization curves (ionization probability as a function of intensity) for several frequencies, which correspond to figure 3.5. It is clear that a single intensity ($I = 16.0$) can not predict the full stabilization behavior.

Figure 3.7 is disturbing, since the genetic algorithm calculations showed the existence of an optimal frequency which can not be related to the theory [29]. Indeed, in both [11] and [30] the goal was to relate numerical studies with finite pulses to the well-founded theory of atomic stabilization which has been developed in the past 20 years for continuous wave pulses [29]. Therefore, an adequate explanation is to be found. It was discovered that using a different picture, stabilization curves do not move to the right as the frequency is increased. Instead of performing calculations at different intensities, ionization probabilities should be calculated as a function of α_0, which includes information about the laser frequency as well. Following this logic, figure 3.8 is obtained.

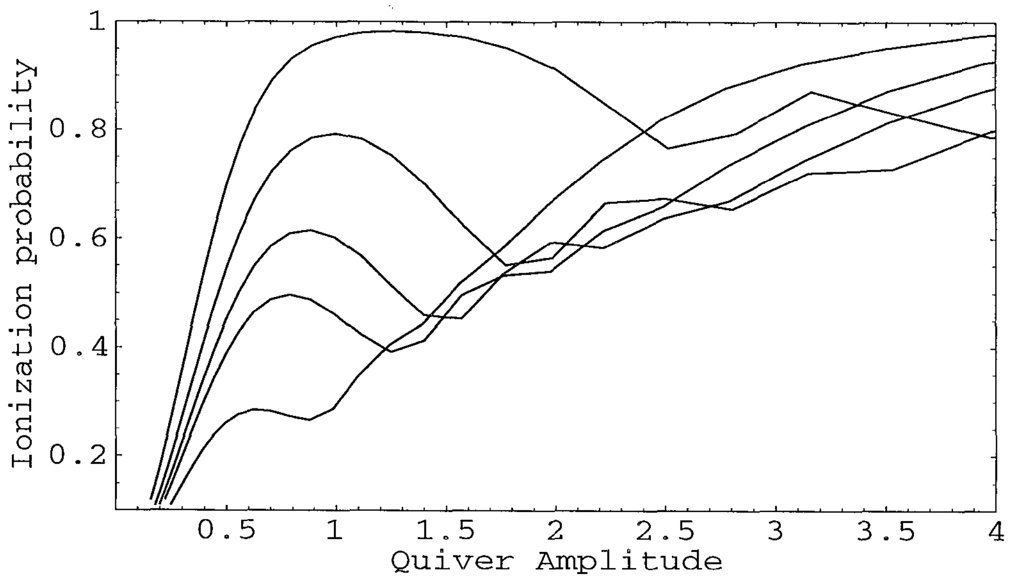


Figure 3.8: The stabilization curves (ionization probability as a function of α_0) for several frequencies, which correspond to figure 3.7. The frequencies from top to bottom (in au) are: $w = 1.0, 2.0, 3.0, 4.0, 8.0$. Note that α_0 is used instead of the laser intensity as the independent variable.

Since with Su's one-point delta potential (3.3) it is impossible to check convergence by decreasing the spatial grid spacing from high to low, because the potential changes as the step size is modified, the short-range potential used throughout this dissertation is preferred. Therefore, going back to the short-range potential used in section 2.4, figure 3.9 is analogous to figure 3.6 and figure 3.10 is analogous to figure 3.7. From now on, all results are generated from the short-range potential. This also enables the use of numerical techniques which converge faster, i.e. that permits using a larger spatial grid spacing. With Su's one-point delta potential, the choice of spatial spacing dx is limited due to the potential.

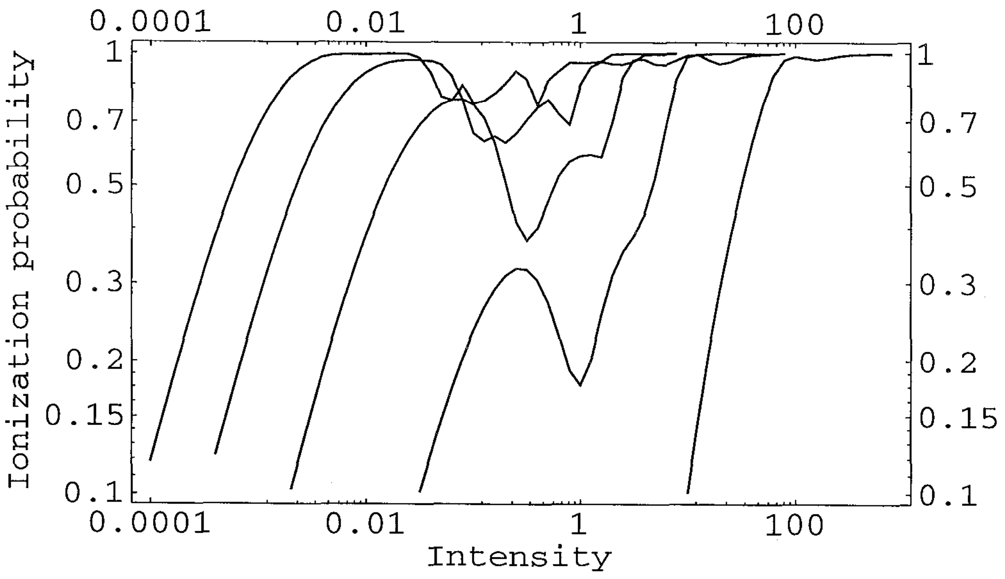


Figure 3.9: The stabilization curves (ionization probability as a function of intensity) for several frequencies, for the short-range potential used in section 2.4.

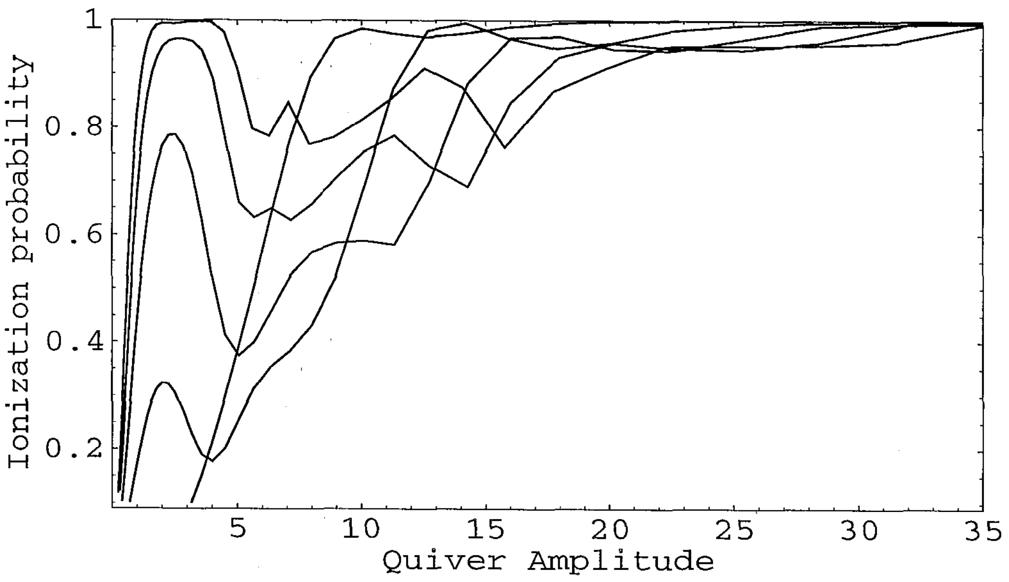


Figure 3.10: The stabilization curves (ionization probability as a function of α_0) for several frequencies, for the short-range potential used in section 2.4. The frequencies from top to bottom (in au) are: $w = 0.2, 0.25, 0.33, 0.5, 1.0$. Note that α_0 is used instead of the laser intensity as the independent variable.

A second important observation is that instead of preserving the number of cycles in the pulse, as Su has done, one needs to preserve the time duration of the pulse, being the physical quantity. Figure 3.11 implements this observation for the same parameters used in figure 3.10. It is found that above a certain frequency, the stabilization structure is converging to the same result. It is the *high frequency limit* of stabilization using finite pulses, which is yet to be verified experimentally.

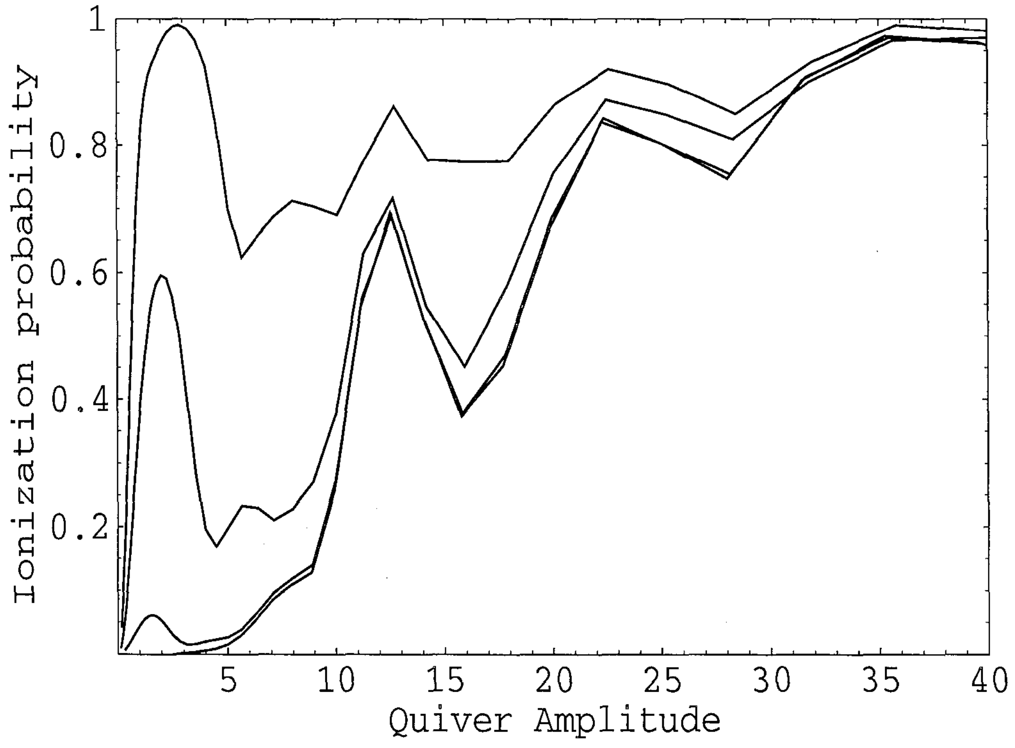


Figure 3.11: The stabilization curves (ionization probability as a function of α_0) for several frequencies, for the short-range potential used in section 2.4. This time, however, time duration of the pulse is preserved as the frequency is increased, instead of number of cycles in a pulse. The frequencies from top to bottom (in au) are: $w = 0.25, 0.5, 1.0, 2.0$. The existence of a high frequency limit is clearly seen, which is the topic of the next chapter.

3.5 Conclusion

In this chapter, through the use of genetic algorithms as a systematic search method, it was possible to obtain a detailed picture of atomic stabilization using finite pulses. Finding an optimal frequency for stabilization was in contrast to [11], in which not enough parameters were used. The discrepancy of the correct results with the known existing theory [29] for continuous-wave pulses led to the idea that one needs to preserve the time duration of the pulse as the frequency is increased. In addition, α_0 should be used as the independent variable instead of the intensity, since it includes information about both the laser intensity and frequency.

The result that was reached, the existence of a high frequency limiting case for atomic stabilization using finite pulses, will be explained in chapter 4.

Chapter 4

Non-Adiabatic High Frequency Theory

The goal of this chapter is to explain the high frequency limit structure found for atomic stabilization. Section 4.1 describes the motivation for applying a (t,t') formalism to connect between the theory of adiabatic stabilization of Gavrila and the stabilization structure seen in figure 3.10. Section 4.2 introduces numerical methods to solve the time-dependent Schrödinger equation. It serves as an important step prior to introducing the (t,t') method, since the (t,t') is based on variable separation and the exploitation of Fourier methods to analyze problems with Hamiltonians which are periodic in time. In section 4.3, the (t,t') method is described. The non-adiabatic high frequency theory is then formulated in section 4.4, using the (t,t'), to bridge the gap between the theory of adiabatic stabilization of Gavrila and the high frequency limit structure of figure 3.10.

4.1 Motivation

Computational studies to model atomic stabilization using finite pulses, i.e. pulses with finite duration turn-on and turn-off ramps to describe the pulse envelope, have been carried out by several researchers including Su, Eberly and Kulander. The most recent studies prior to this dissertation are reported in [11,30,13]. This type of stabilization, often coined "dynamic stabilization", is important to model since realistic pulses in high power laser experiments possess on and off switching ramps. If such pulse shapes can contribute to the stabilization effect, it would be beneficial to understand the reasons and means of such a contribution.

Much work has been conducted for over 20 years to predict and study atomic stabilization by using continuous-wave laser pulses. Gavrila and co-workers developed a high-frequency Floquet theory within the Kramers-Henneberger gauge [28,29] which unifies many of the key points already mentioned in section 2.5 into one theoretical framework. This framework uses the Floquet analysis, in which the effective potential reached after the Kramers-Henneberger transformation (equation 2.8) is Fourier expanded, in order to extract the time-averaged cycle potential (equation 2.9) as the first order contribution. Such an expansion is possible assuming the Hamiltonian is periodic in time. With a continuous-wave laser pulse, this assumption is clearly justified. This type of stabilization, often termed “adiabatic stabilization”, has been modeled extensively. It has been widely accepted as the basis for understanding atomic stabilization seen in continuous-wave laser pulse experiments.

It is therefore not surprising that Su and co-workers attempted to relate “dynamic stabilization” to the well understood “adiabatic stabilization” in their studies. It should be noted that these terms were introduced by researchers in order to distinguish between the two ways of modeling (one with a finite pulse, the other with a continuous-wave pulse). The connection between the two is therefore an important issue which motivated much of Su’s work. This is also the motivation in section 4.4 of this dissertation, bridging between the theory of adiabatic stabilization of Gavrila and the structure obtained in figure 3.10.

However, an important tool that is essential in order to successfully connect between the two models mentioned above, was reported elsewhere [37,38]. With the (t,t’) method, which was introduced in these references, it is possible to Fourier expand not only the time t as was done in the Floquet analysis by Gavrila and co-workers but also the time t’ which exists in the pulse envelope, used in “dynamic stabilization”. The (t,t’) is a numerical method which attempts to treat the time in equal footing as the space. It can be viewed as an extension of an earlier idea, which was first introduced by formulating the Split Operator method [39] and later was carried out in other Fourier based methods [40]. The motivation in the introduction of these methods was to separate between time propagation and space representation, Fourier expanding in space in order to achieve a reduction in computational cost. Therefore, section 4.2 discusses Fourier-based (spectral) methods prior to section 4.3 which introduces the (t,t') method.

4.2 Numerical Methods

In this section, numerical methods to directly solve the time-dependent Schrödinger equation which have been used in atomic stabilization studies are briefly reviewed. The primary goal is to prepare for the introduction of the (t,t') method. The secondary goal is to understand the advantages and limitations of several methods and to recommend a robust method for modeling atomic stabilization.

In an atomic stabilization problem, as well as in other studies of time-dependent processes including above threshold ionization and harmonic generation [41], the Crank-Nicolson method was used with a computational effort of $O(N)$ [42] since the LU decomposition to invert the resulting tridiagonal matrix only costs N operations. Therefore, even the second order difference scheme [43] (an explicit version of the implicit Crank-Nicolson method) will not result in less operations. The Crank-Nicolson method, first used in [26] to solve the time-dependent Schrödinger equation, uses a finite-difference representation with the following propagator:

$$e^{-iH\Delta t} = (1 - iH\Delta t/2)(1 + iH\Delta t/2)^{-1} + O(\Delta t^3) \tag{4.1}$$

Note that in the right-hand side, all spatial derivatives in H are numerically calculated using a finite-difference approximation. For many years, the Crank-Nicolson method dominated this field and was used for practically all purposes, due to its simplicity and prominent features (being second order, norm preserving and unconditionally stable) among other finite-difference competitors.

During the early 1980s, new methods were proposed [39,40] for separating time and space (which are intimately connected in the Crank-Nicolson) and use a more efficient spatial representation to treat space. In Feit's split operator method, time propagation is performed by approximating the propagator as follows:

$$e^{-iH\Delta t} = e^{-iK\Delta t/2} e^{-iV\Delta t} e^{-iK\Delta t/2} + O(\Delta t^3) \tag{4.2}$$

where $H = V + K$, H is split into the potential term V and the kinetic term K which includes all spatial derivatives. The spatial derivatives will then be numerically calculated by performing a fast Fourier transform on the wavefunction ψ, multiplying the result by a diagonal matrix, then transforming back to configuration space by an inverse fast Fourier transform. This method was found to be superior in many problems which allowed a better spatial representation using a Fourier representation instead of the finite-difference (meaning a reduction of N, the number of grid points in space, compared with what is needed for accurate finite-differencing). In these problems, one can sometimes reduce N by a factor of 5 (for example, in the simple harmonic oscillator potential). A point can be reached in which the cost for a Fourier representation, being $O(NlogN)$, can be superior to the $O(N')$ in finite-difference representation provided N is much smaller than N'. Note that as far as the time propagation, both the Crank-Nicolson and the split operator method are second order in time. While the spatial variable has been dealt with efficiently in the split operator method, both are equally inefficient with regard to the time propagation.

To treat the time propagation problem, a new scheme was proposed [44] which unfortunately works for only time-independent Hamiltonians without further modifications. An efficient propagator for stationary Hamiltonians was found by using the Chebychev expansion:

$$e^{-iH\Delta t} \approx \sum_{n=0}^{N} a_n \varphi_n(-iH\Delta t) \tag{4.3}$$

where a_n are the expansion coefficients and φ_n are the complex Chebychev polynomials. By using a global solution to the time variable, rather than a local finite-difference in time, the inefficiency in time propagation which exists in both the Crank-Nicolson and spectral methods (such as the split operator method) can be improved. However, in the Crank-Nicolson both space and time are intimately connected to each other in a way that dictates a local representation for both. With the spectral method, since time is separated from space, in contrast to the Crank-Nicolson it is advantageous in some problems to transform from an explicitly time-dependent Hamiltonian to a stationary one. That way, one can use both a Chebychev propagator in time (being an efficient propagator) and a Fourier representation in space (being an efficient representation) as the recommended method for work to solve the problem.

Using a hybrid "Chebychev-Fourier" scheme proposed by Baer, which was specifically implemented to model atomic stabilization in [15], it is possible to use $dx = 2.0$ and $dt = \pi/2\omega$, which are much bigger spacings then what was used in previous studies such as [11,30]. It solves the same problem and gives the same results in wave-packet calculations. In this proposed scheme, the wavepacket calculation is done in the Kramers-Henneberger frame, so that in each time step the instantaneous cycle-averaged (equation 2.9) potential is calculated before the propagation step. As far as speed considerations, this method even surpasses the Crank-Nicolson due to the large spacings allowed. As far as convergence and accuracy considerations, not only does this scheme give a solid 4-digit convergence, which can not be achieved using either the Crank-Nicolson or the split operator method, it is also recommended as an important tool for the diagnostic of the results since the evolution of the cycle-averaged potential in time can be conveniently calculated and followed by using this method.

4.3 (t,t') Formalism

There are many physical applications besides atomic stabilization, in which the Hamiltonian is explicitly time-dependent. The common solution for propagation in these explicitly time-dependent problems is to use very small grid spacing in time, such that within each time step the Hamiltonian $H(t)$ is almost stationary. Considering that these calculations are demanding with respect to computer time, whereas global expansion methods such as the Chebychev propagator are useful essentially only in time-independent Hamiltonians, alternative schemes are desirable in order to treat the time-coordinate accurately and efficiently.

In the previous section, a problem-dependent solution was found to treat the Hamiltonian as stationary by calculating the cycle-averaged potential (working in the Kramers-Henneberger frame) at each time step. A general approach to transform the explicitly time-dependent Hamiltonian into a stationary one was introduced by Howland [45] in quantum scattering theory and later was discussed [46] in analogy to the stationary formalism used in classical mechanics. The first to implement it in a numerical study was Moiseyev [47] who applied the stationary formalism to the calculation of state-to-state transition probabilities for complex-scaled time-periodic Hamiltonians. The formalism was then developed [37,38] to be applied to general time-dependent Hamiltonians using a Chebychev propagator in time.

The main idea behind the (t,t') formalism is to add another dimension to the problem, supplementing the time and space grids, thus enabling the Hamiltonian to be stationary so that an efficient propagation method can be used. This formalism can reduce the overall computational cost in some problems, depending on how much is gained, since adding a new dimension obviously requires more work. In addition, it can also be used as a powerful formalism (e.g. next section), in particular problems in which one time is much slower than the other. An ideal candidate is a laser pulse interacting with molecules, in which the laser frequency is so high with respect to the slowly varying pulse shape that it is possible to neglect the high frequency oscillations which are produced as a consequence. Simplifying assumptions can then lead to neglecting time t with respect to time t'. One is then left with an extremely efficient scheme at no cost (i.e., without adding any extra dimension).

The first step in the (t,t') formalism is to add a grid in a new coordinate t'. The relation between the embedded wavefunction and the usual one subject to an initial state $\Psi(x, 0)$ is defined as:

$$\Psi(x,t) = \int_{-\infty}^{\infty} \delta(t' - t)\Phi(x,t',t)dt' \tag{4.4}$$

where t' acts like an additional coordinate in the generalized Hilbert space and $\Phi(x,t',t)$ is the new wavefunction to be solved. Note that by definition, $\Phi(x,t',0) = \Psi(x,0)$ is the initial condition. $\Phi(x,t',t)$ is the solution to the time-dependent

Schrödinger equation represented by the (t,t') formalism:

$$i\hbar\frac{\partial}{\partial t}\Phi(x,t',t) = \mathcal{H}(x,t')\Phi(x,t,t') \tag{4.5}$$

The $\mathcal{H}(x,t')$ operator is defined for a general time dependent Hamiltonian by:

$$\mathcal{H}(x,t') = H(x,t') - i\hbar\frac{\partial}{\partial t'} \tag{4.6}$$

Note that this Hamiltonian is t-independent. The formal solution of equation (4.5) is then given by:

$$\Phi(x,t,t') = e^{-it\mathcal{H}(x,t')/\hbar}\Psi(x,0) \tag{4.7}$$

The solution to the original problem is simply $\Psi(x,t) = \Phi(x,t' = t)$. Note that it is now possible to use a Chebychev propagator to advance in t-space, and a Fourier expansion in the t'-space grid to treat t' and x on an equal footing. Each of the 3 coordinates has its own grid and method, independent of the other. Thus, this can be viewed as an extension to the main idea behind the spectral methods which were formulated in the early 1980's for use in quantum mechanics: *separation of variables* [48][1]

4.4 Non-Adiabatic High Frequency Formulation

The difference between adiabatic stabilization and dynamic stabilization was discussed in section 4.1. In this section, the theory needed to explain dynamic stabiliza-

[1]Gilbert Strang was the first to derive a split operator scheme for fluid dynamics, during the 1960s.

tion is formulated, using the tools which were introduced in sections 4.2 and 4.3.

The electron dynamics in an intense laser field is described by the Schrödinger equation in the Kramer-Henneberger [49] gauge (see equation 2.8):

$$i\hbar\frac{\partial}{\partial t}\psi(x,t) = \left(\frac{p^2}{2m_e} + V(x+\alpha(t))\right)\psi(x,t) \tag{4.8}$$

where $p = -i\hbar\frac{\partial}{\partial x}$ and $\alpha(t)$ is the instantaneous displacement (see section 2.5).

Consider a short electromagnetic pulse with displacement of the form $\alpha(t) = \alpha_0 f(t)\sin\omega t$. Here, α_0 is the maximal displacement and the function f(t) is the pulse envelope of finite duration starting from $t = 0$ and ending at $t = T_f$. For an electron initially in the ground state $\psi_0(x)$ the high frequency limit is studied by numerically integrating equation (4.8) with a one-dimensional short-range potential (see section 2.4):

$$V(x) = -Bexp(-x^2/2\sigma^2) \tag{4.9}$$

The potential is constructed to support a single bound state (the value of $B = 0.187$ au and $\sigma = 1.8$ au are chosen to yield a bound state energy of approximately -0.1 au). The total ionization probability, as a function of α_0 and ω was calculated for a sine-square envelope (see section 2.4), characterized by a ramp time T_{on} and flat time T_{flat}:

$$f(t) = \begin{cases} \sin^2(\pi \cdot t/2T_{on}) & t \leq T_{on} \\ 1 & T_{on} < t \leq T_r \\ \cos^2(\pi(t-T_r)/2T_{on}) & T_r < t \leq T_f \end{cases}$$

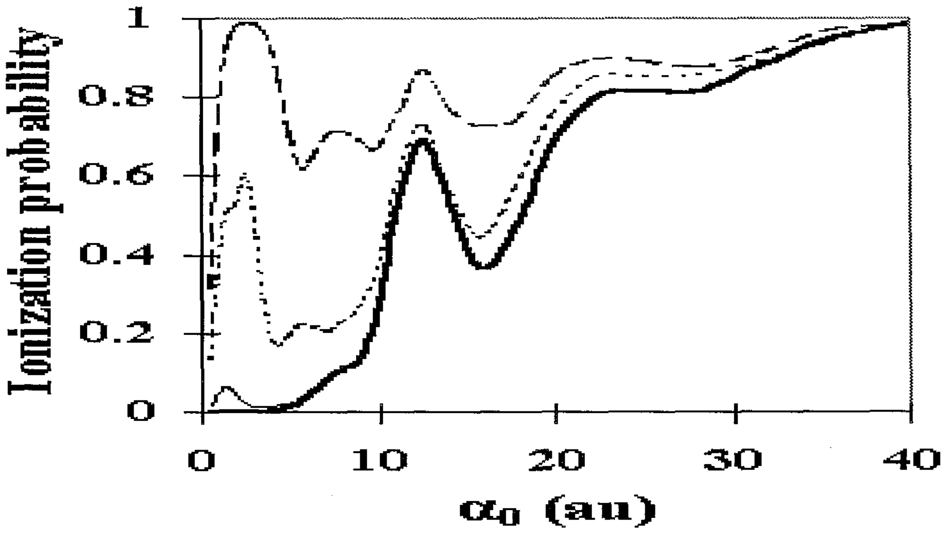


Figure 4.1: Total ionization probability vs. α_0, for a $\sin^2()$ pulse shape with $T_{on} = 94.25$, $T_{flat} = 251.3$ au, and $\omega = 0.25$ (dashed line), 0.5 (dotted), 1.0 (solid) and 2.0 (thick line).

Where $T_r = T_{on} + T_{flat}$. The wave-packet calculations were done within the Kramers-Henneberger gauge, using the Chebychev expansion method that was recommended in section 4.2, with a negative imaginary potential at the grid asymptotes [50]. A grid spacing of $dx = 2.0$ au and $dt = 2\pi\omega^{-1}/4$ gave a solid 4-digit convergence. The shifts of the potential in the Kramers-Henneberger frame are performed using Fast-Fourier Transform based on the identity $V(\hat{x}+\alpha) = e^{-i\hat{P}\alpha/\hbar}V(\hat{x})e^{+i\hat{P}\alpha/\hbar}$. The calculation results are shown in figure (4.1) where the end of pulse ionization is plotted as a function of the maximal displacement α_0 for several frequencies. All other envelope parameters are kept constant. Note that this figure is visually identical to figure (3.10), in which the split operator method was used with a much smaller grid spacing of $dx = 0.1$. As the frequency is raised the ionization profile converges to a limit - the high frequency limit. Unlike the continuous-wave case where the limit is zero according to the theory of adiabatic stabilization of Gavrila, here there is a structure. The ionization initially rises monotonically with α_0 until it reaches a local maximum of 0.7 probability at $\alpha_0 = 12.0$ au. As α_0 is further increased beyond this value, the ionization probability rapidly decreases reaching a local minimum at $\alpha_0 = 16.0$ au. Beyond this the ionization increases once more as α_0 is increased.

In order to understand these results, the (t,t') formalism is generalized as in [38], treating the fast time t as a dynamic variable and defining a slow-time parameter τ and the τ-dependent Hamiltonian:

$$H(\tau) = \frac{p^2}{2m_e} + V(x + \alpha_0 f(\tau)\sin(\omega t)) - i\hbar\frac{\partial}{\partial t} \tag{4.10}$$

With this Hamiltonian, as in the usual (t-t') formalism, the Schrödinger equation becomes:

$$i\hbar\frac{\partial}{\partial \tau}\psi(x,t;\tau) = H(\tau)\psi(x,t;\tau) \tag{4.11}$$

This equation is solved with the initial condition $\psi(x,t;0) = \psi_0(x)$. The full solution of equation (4.11) is then obtained as $\psi(\tau) = \psi(t=\tau,\tau)$.

The extended Hamiltonian $H(\tau)$ is periodic in t, allowing the following Fourier expansion:

$$V(x+\alpha_0 f(\tau)\sin(\omega t)) = \sum_n e^{in\omega t} V_n(x,\tau) \tag{4.12}$$

where the expansion coefficients are:

$$V_n(x,\tau) = \frac{1}{2\pi}\int_0^{2\pi} e^{-in\phi} V(x+\alpha_0 f(\tau)\sin(\phi))d\phi \tag{4.13}$$

Note that this is the starting point of a Floquet approach, in which for a periodic Hamiltonian in time, it is possible to perform a Fourier expansion in order to eliminate the time dependence, thus replacing the time-dependent Schrödinger equation with the time-independent one. Here, the adiabatic-Floquet approach is used to treat the slow time τ. Consider the instantaneous eigenfunctions and eigenvalues:

$$H(\tau)\psi_j(x,t;\tau) = \varepsilon_j(\tau)\psi_j(x,t;\tau) \tag{4.14}$$

Because of the periodicity in t the eigenfunctions are written as:

$$\psi_j(x,t;\tau) = \sum_n e^{-in\omega t}\varphi_n^{(j)}(x;\tau) \tag{4.15}$$

where the components $\varphi_n^{(j)}(\tau)$ are the instantaneous Floquet states, satisfying:

$$\sum_m \left\{ \left(\frac{p^2}{2m} - n\hbar\omega\right)\delta_{nm} + V_{n-m}(x;\tau)\right\}\varphi_m^{(j)}(\tau) = \varepsilon_j(\tau)\varphi_n^{(j)}(\tau) \tag{4.16}$$

where $\varepsilon_j(\tau)$ are the instantaneous quasi-energies. The time dependent wave packet is expanded in the instantaneous Floquet basis:

$$\psi(x,t;\tau) = \sum_j c_j(\tau)e^{-i/\hbar\int_0^\tau \varepsilon_j(\tau')d\tau'}\psi_j(x,t;\tau) \tag{4.17}$$

Plugging this form into equation (4.11) leads to the equation:

$$\dot{c}_j = \sum_k \left\langle \psi_j | \frac{\partial}{\partial \tau} \psi_k \right\rangle_{x,t} e^{-i/\hbar \int_0^\tau (\varepsilon_k - \varepsilon_j) d\tau'} c_k(\tau) \tag{4.18}$$

Using the Hellmann-Feynman theorem and equation (4.15), one obtains for the non-adiabatic transition coefficients ($k \neq j$):

$$(\varepsilon_k - \varepsilon_j) \left\langle \psi_j | \frac{\partial}{\partial \tau} \psi_k \right\rangle_{x,t} = \left\langle \varphi_m^{(j)} | \dot{V}_{n-m} | \varphi_n^{(k)} \right\rangle \tag{4.19}$$

Where:

$$\dot{V}_l(x, \tau) = \frac{\alpha_0 \dot{f}(\tau)}{2i} (\nabla V_{l+1} - \nabla V_{l-1}) \tag{4.20}$$

Note that non-adiabatic transitions are possible only when the envelope itself is changing, as clearly seen in equation (4.20). The strength of transitions also depends on the gradient of the Fourier components. It is seen clearly that for a continuous wave pulse, where the pulse envelope is constant the transition rate is zero, as is the case for the theory of adiabatic stabilization of Gavrila. Note, however, that this observation does not mean there is no ionization, because in general the eigenvalues ε_j may be complex with negative imaginary parts. In the adiabatic picture it is meaningful to differentiate between "adiabatic ionization" which is caused by a complex energy and non-adiabatic ionization caused by non-adiabatic transitions.

The theory above makes no specific assumptions and can be considered exact. When the laser frequency ω is very high it is possible to make simplifying approximations. Following reference [29] p.435 one can neglect the rapidly oscillating parts of the eigenfunctions ψ_j. One can therefore set:

$$\psi_j = \varphi_0^{(j)} \;\; ; \;\; \varphi_m^{(j)} = 0 \;\; (m \neq 0) \tag{4.21}$$

In this case the eigenvalues $\varepsilon_j(\tau)$ corresponding to bound states are all real. The equation for the coefficients is:

$$\dot{c}_j = \sum_k T_{jk} c_k = \sum_{k \neq j} \frac{\langle \varphi_0^{(j)} | \dot{V}_0 | \varphi_0^{(k)} \rangle}{\varepsilon_j - \varepsilon_k} e^{-i/\hbar \int_0^\tau (\varepsilon_k - \varepsilon_j) d\tau'} c_k(\tau) - \langle \varphi_0^{(j)} | \dot{\varphi}_0{}^{(j)} \rangle c_j(\tau) \tag{4.22}$$

where:

$$\dot{V}_0(x,\tau) = \dot{f}(\tau) \alpha_0 \cdot \nabla \left\{ \frac{1}{2\pi} \int_0^{2\pi} V(x + \alpha_0 f(\tau) \sin(\phi)) \sin(\phi) d\phi \right\} \tag{4.23}$$

Equation (4.22) is an adiabatic equation for the slow Schrödinger equation:

$$i\hbar \frac{\partial}{\partial \tau} \Psi(\tau) = \left\{ \frac{p^2}{2m_e} + V_0(x,\tau) \right\} \Psi(\tau) \tag{4.24}$$

This equation is independent of the frequency and yields the "high frequency limit" shown as the bold face line in figure (4.1). Because the potential $V_0(x,\tau)$ is of the same parity as the atomic potential $V(x)$, only even-even and odd-odd transitions are possible in the high frequency limit.

Equation (4.22) may be solved directly for the adiabatic population amplitudes. Here we ignored the odd adiabatic states which are decoupled. During the pulse rise and pulse decay, the M lowest eigenstates were calculated (the "adiabatic states") of the instantaneous Hamiltonian of equation (4.24). When the pulse flat time is very long the adiabatic approach is more efficient than solving equation (4.24). If only the M lowest adiabatic states are important (where M is much smaller than the number of grid points), the transition matrix in equation (4.22) is a small $M \times M$ anti-Hermitian matrix and the evolution of the coefficients for the time step can be efficiently performed by diagonalizing it. It was found that 12 states were important for quantitative agreement with the full wave packet results. However, as will shortly be shown, only 2 states are needed for understanding the general features of the stabilization.

The numerical stability of the evolution within the adiabatic scheme is sensitive to the way one chooses the phases of the adiabatic states: the phase of the n^{th} eigenstate $\varphi_n(\tau + \Delta\tau)$ should be adjusted so that $\langle \varphi_n(\tau) | \varphi_n(\tau + \Delta\tau) \rangle = \rho$ is a real positive number. There is a possibility of accumulating geometric Berry phases [51] at each state. If this happens the eigenstate at the end of the pulse has a Berry phase of $\Omega_B = \pi$ relative to the ground state at time t=0, in other words: $\langle \varphi_n(0) | \varphi_n(T_f) \rangle = -1$. In the calculations for the set of parameters chosen in this study, no such phase effect has been encountered.

The population of the 5 lowest eigenstates for various ramp times during the rise and decay of a $\alpha_0 = 16au$ pulse is shown in the Appendix. Figure (4.2) highlights some of the times which are crucial for observing the most prominent feature, the interplay between the two lowest adiabatic states.

The prominent feature, becoming apparent at $\alpha_0 = 12au$, is the strong population transfer between the $n = 0$ and the $n = 1$ states. This effect becomes dominant for yet larger $\alpha_0's$. Thus, under the influence of a strong laser field the $n = 1$ state acts as a trap against ionization, explaining the high frequency stabilization starting at $\alpha_0 \approx 12au$ and reaching a maximum suppression at $\alpha_0 \approx 16au$ (see figure 4.1). Beyond $\alpha_0 \approx 16au$ ionization suppression degrades as the $n = 1$ state population is high and some transfer to continuum states occur.

The reason for the trapping effect is the creation of a laser induced resonance state as shown in figure (4.3). It is possible to estimate the magnitude of displacement α_{res} at which the resonance appears. Define $A = \int_{-\infty}^{\infty} V_0(r, \alpha)dr$ and assume the cycle-averaged potential (equation 2.9) may be approximated as a square well of length $L = 2\alpha$. The resonance appears when the well depth A/L is equal to the energy of the second state in a square well $E_2 = 2\hbar^2\pi^2/m_eL^2$ thus: $\alpha_{res} \approx \hbar^2\pi^2/m_eA$. In our potential this equation correctly estimates the appearance of the resonance at $\alpha_0 = 12au$.

The Floquet-adiabatic framework has allowed a straightforward explanation of dynamic stabilization in short-ranged potentials caused by the appearance of a laser-

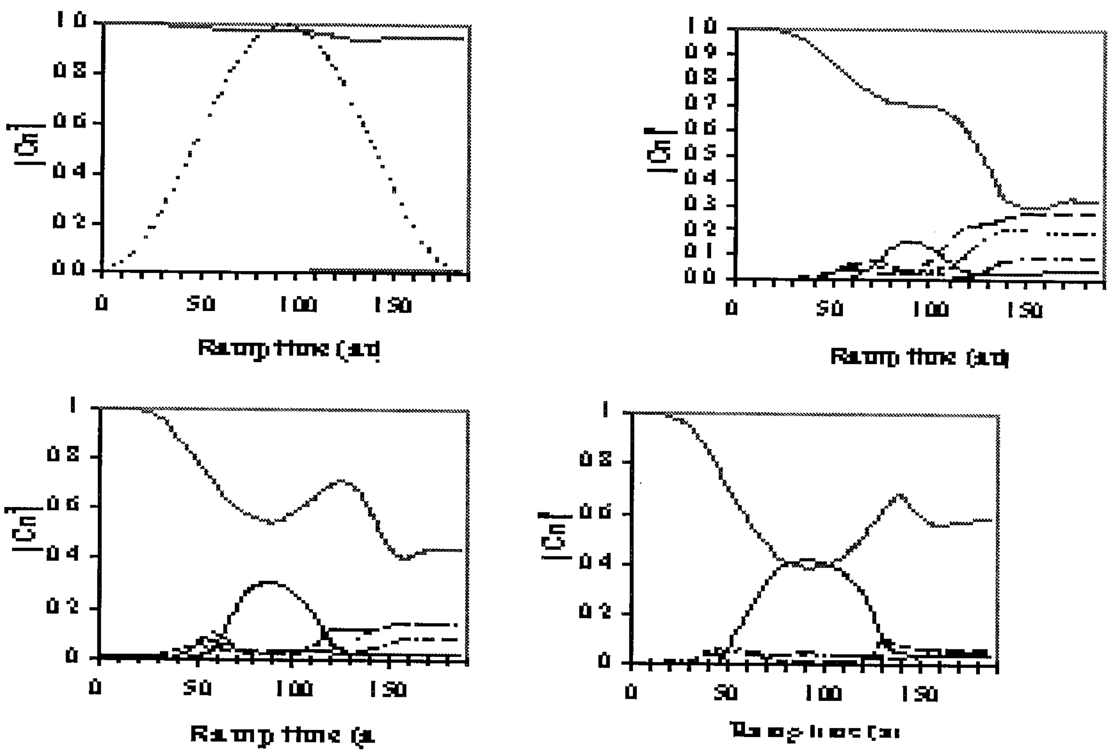


Figure 4.2: Population of the lowest 5 adiabatic states as a function of pulse rise and decay time for 4 cases, starting from left to right: $\alpha_0 = 6au$, $\alpha_0 = 12au$, $\alpha_0 = 14au$, $\alpha_0 = 16au$. Pulse ramp forms are shown as a dotted line in the $\alpha_0 = 6au$ figure. The $n = 0$ and $n = 1$ lines are captioned while the $n = 2$ is a dashed line, $n = 3$ is dotted and $n = 4$ dot-dashed. In all cases the $n = 0$ state starts with $|C_0|^2 = 1$ at $t = 0$ and loses population to excited states which are all positive energy states.

induced resonance state that traps ionized population. A fundamental difference between adiabatic and dynamic stabilization is that the former is caused by the suppression of non-adiabatic transitions while the latter exhibits strong transitions and the suppression of ionization is caused by trapping into a laser induced resonance state. Pulses must also be turned off for a complete understanding of dynamic stabilization. The pulse can also have a "flat" part where no non-adiabatic transitions occur. Still the length of the flat part has an influence on the total ionization, be-

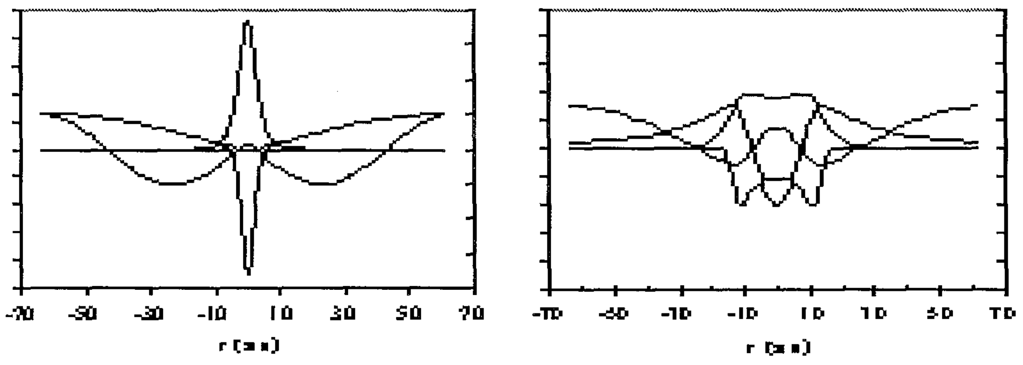


Figure 4.3: The shape of the dressed potential and the first 3 even adiabatic states at $\alpha = 0au$ (left) and $\alpha = 12.5au$ (right). It is seen that while the n=0 state is bound and n=2 is a continuum state in both cases, the n=1 state changes character from a continuum state to a localized resonance.

cause it determines the quantum phases with which components remix as the pulse is turned off. Dynamic stabilization in our case is a manifestation of light induced atomic resonance states. Two questions which deserve detailed treatment in future work are: what is the role of laser induced states when longer ranged potentials are present, and when lower frequencies are used. In addition, an exciting new direction is the construction of polarized pulses (interacting with electrons in more than one dimension) where a geometric Berry phase effect [51] can be measured on these light induced resonances.

Chapter 5

Introduction to Markov Decision Processes

This chapter serves as an introduction to Markov decision problems. It also describes standard methods (i.e., dynamic programming) for solving Markov decision problems. In the next chapter, the implementation of these methods will be compared to a genetic algorithm implementation on a given model problem.

Section 5.1 introduces the field of complex decision making, and the standard way to model such problems as Markov decision processes. Section 5.2 provides the basic terminology. In section 5.3, the Bellman equation is derived. Section 5.4 describes Dynamic Programming, an approach to solve the Bellman equation. Finally, section 5.5 motivates the comparison done in the next chapter, by reviewing the basic approaches used for solving Large-Scale Markov decision problems which are of practical interest.

5.1 Modeling Decision Making as a Markov Decision Process

Computational issues involved in making decisions are of interest in a variety of fields and areas of application. From inventory control problems in operations research to robotic navigation in artificial intelligence, a wide range of problems can be tackled and solved using efficient search strategies.

In general, when confronted with a decision to make, there are a number of different alternatives (actions) one can choose from. Choosing the best action often requires long term thinking, as opposed to the immediate effects of an action. In problems of this type, it is desired to choose the action that makes the right tradeoffs between the immediate rewards and the future gains. People do such reasoning on a daily basis, and a Markov decision process is a way to model problems so that this rational can be automated. If one can model the problem as a Markov decision process, then there are a number of algorithms that will automatically solve the decision problem.

In the next section, after introducing the basic terminology, the Markov assumption will be explained. This simplifying assumption allows the examination of a wide range of algorithms to solve decision making problems, which are otherwise too complicated to solve in a systematic way.

5.2 Basic Terminology: States, Actions, Policies

The four components of a Markov decision problem are: a set of states, a set of actions, the effects of the actions and the immediate value of the actions. These components will be discussed in this section, as well as other basic terminology which is used in this field.

The decision-making problem is framed from the perspective of an *agent* that is situated in an environment. An agent can be a person, a robot, a control program in a factory, or the like. For such an agent, the world is composed of different *states.* The agent can choose among various different *actions*, which are a set of possible alternatives. The problem is to know which of these actions to perform for a particular state of the world.

When deciding between various actions, it is often the case that there is some idea of how they will affect the current state. The *transitions* specify how each of the actions change the state. The most powerful aspect of the Markov decision process is that the effects of an action can be probabilistic (e.g., performing an action 'a1' in state 's1', results in the state 's2' 70% of the time and state 's3' 30% of the time). In order to measure the value of an action, *immediate rewards* are often used, that is the immediate value for performing each action in each state.

It is now possible to formally define a *Markov decision process* as a 4-tuple, (S,A,T,R). S is a set of states, A is a set of actions, T is a transition model for the system which is a mapping from $S \times A \times S$ into probabilities in $[0, 1]$. For the states $s, s' \in S$ and an action $a \in A$, $T(s, a, s') = P(s'|s, a)$, the probability that the next state will be in s' when an action a is taken in state s. R is a reward function that maps from $S \times A \times S$ to real-valued rewards.

Finally, the solution to a Markov decision problem is called a *policy*, denoted by π. It simply specifies the best action to take for each of the states. Although the policy is what one is after when solving a Markov decision problem, what is actually computed is known as the *value function.* A value function, denoted by V, is similar to a policy except that instead of specifying an action for each state, it specifies a numerical value for each state (which is also known as the *utility value* of that state).

At this point, it is instructive to clarify the *Markov assumption.* When previously defining transitions, it was enough to specify the resulting next state for each starting state and action. This assumes that the next state depends only upon the current state (and action). There are situations where the effects of an action might depend not only on the current state, but upon the last few states. The assumption made by the Markov decision process model is that the next state is only determined by the current state (and action).

5.3 Bellman Equation

In this section, the Bellman equation is given which is the basis for dynamic programming, an approach to solving sequential decision problems developed in the late 1950s by Richard Bellman [16]. In addition, two *value determination* methods are described. Value determination is the step in which utility values are determined from a given policy and is common to all methods for solving Markov decision problems.

Algorithms to solve Markov decision problems work by assigning values to states and manipulating these values with the goal of finding the optimal policy. Any policy, π, defines a value function, V_π, over all states in the model. Given an additive value function, the value of a state can be expressed in terms of the value of its successors:

$$V_\pi(s) = R(s, \pi(s)) + \beta \sum_{s'} T(s, a, s') V_\pi(s') \tag{5.1}$$

Equation (5.1) is known as the *Bellman equation* [16]. It says that the value of state s under policy π is the immediate reward received in state s plus the expected value of the succeeding state. $0 \leq \beta < 1$ is known as the *discount factor.* Discounting arises so that the sum in the equation above will converge to a finite amount in infinite-horizon models, as well as other reasons mentioned in [52,53]. In the next chapter, the issue of discounting will be revisited.

The optimal policy will be labeled as π^* and optimal value function as V^*. Several theorems can be proved at this point about uniqueness and optimality (see proofs in [52). For example, there exists a unique, optimal value function, V^*, (Blackwell, 1962) resulting from any optimal policy, π^*, such that:

$$V^*(s) = R(s, \pi^*(s)) + \beta \sum_{s'} T(s, a, s')V^*(s') \tag{5.2}$$

Since the optimal policy assigns the best action to every state, the goal is to find an optimal policy π^*, characterized by actions a, such that:

$$V^*(s) = \max_a [R(s, a) + \beta \sum_{s'} T(s, a, s')V^*(s')] \tag{5.3}$$

A basic step in every method which attempts to find the optimal policy is the value determination step. Given a policy, the goal is to compute the values at all states. The value determination step can be implemented in several ways. Two common ways will be described here, the second one being used in the next chapter. Other ways, such as the method of temporal differences [55] which determines V_π through experience with the environment, are usually applied to large problems and will not be discussed here.

The first way is known as *successive approximation.* It starts with an arbitrary value function, V^0, and finds the desired value function using an iterative procedure. Specifically, one calculates V^i from V^{i-1}. If one labels the right-hand-side of the Bellman equation (5.1) by the operator J, such that the Bellman equation becomes $V_\pi(s) = J_\pi(s)V_\pi$, it is possible to iterate as follows:

$$V^i \leftarrow J_\pi V^{i-1} \tag{5.4}$$

With each application of J_π, V^i gets closer to V_π.

The second way, which can also exploit the sparseness in the system, is by directly solving a set of linear equations. The Bellman equation for policy π can be written for each state in S, producing the following linear system of equations in a matrix form:

$$\left(\begin{array}{ccc|c} \beta T(s_1, \pi(s_1, s_1)) - 1 & \beta T(s_1, \pi(s_1, s_2)) & \dots & R(s_1) \\ \beta T(s_2, \pi(s_2, s_1)) & \beta T(s_2, \pi(s_2, s_2)) - 1 & \dots & R(s_2) \\ \vdots & \vdots & \ddots & \vdots \end{array}\right)$$

The solution to this system of equations produces V_π. If n is the number of variables in the system, it can be solved using Gaussian Elimination in $O(n^3)$ time or closer to $O(n^2)$ with methods that exploit sparseness in the matrix.

5.4 Dynamic Programming

Based on the Bellman equation, two well known iterative procedure which aim at finding the optimal policy π^* are *value iteration* and *policy iteration.*

5.4.1 Value Iteration

Value iteration is very similar to the method of successive approximation, described in the previous section, for the value determination step. Basically, the Bellman equation (5.1) can be applied repeatedly. Value iteration starts with an arbitrary value function, V^0. Instead of applying J_π for a particular policy, as in successive approximation, it applies the general J operator which contains a maximum over all actions. Thus,

$$V^i \leftarrow JV^{i-1} \tag{5.5}$$

As $i \to \infty$, the values in each of the states will converge to stable values given certain conditions on the environment. A handful of theorems exist regarding the property of the J operator and the rate of convergence (see [52]). Also, value iteration can be generalized to the case where not all states are updated at once. This modification is known as the *asynchronous value iteration*, more details can be found in [56]. A step by step procedure for the standard value iteration algorithm is given below:

1. Start with an arbitrary evaluation function V.

2. Repeat until the error bound of the evaluation function is less than ϵ: For each state s in S,

$$V(s) := \arg\max_a [R(s,a) + \beta \sum_{s'} T(s,a,s')V(s')] \tag{5.6}$$

3. Extract an ϵ-optimal policy from the evaluation function as follows. For each state s in S,

$$\pi(s) := \arg\max_a [R(s,a) + \beta \sum_{s'} T(s,a,s')V(s')] \tag{5.7}$$

5.4.2 Policy Iteration

Since the optimal policy is often not very sensitive to the exact utility values, an alternative way to find the optimal policies can be devised. The policy iteration algorithm works by picking a policy, then calculating the utility of each state given that policy. It then updates the policy at each state using the utilities of the successor states, and repeats until the policy stabilizes. The step in which utility values are determined from a given policy, the value determination step, was outlined in the previous section. For small state spaces, value determination using exact solution methods (i.e., solving the linear system of equations as specified at the end of section 5.3) is often the most efficient approach. The basic idea behind policy iteration, as compared to value iteration, is that the value determination step should be simpler than value iteration because the action in each state is fixed by the policy.

A step by step procedure for the policy iteration algorithm is given below:

1. Start with an arbitrary policy π.

2. Repeat until the policy does not change:

   (a) Compute the evaluation function $V(\pi)$ for policy π by solving the linear

   system of equations (as outlined at the end of section 5.3).

   (b) For each state s in S,

$$\pi(s) := \arg\max_a [R(s,a) + \beta \sum_{s'} T(s,a,s')V(s')] \tag{5.8}$$

   Resolve ties arbitrarily, but give preference to the currently selected action.

Thus, step 2(a) is the value determination step which can be done more efficiently than in the case of value iteration, and step 2(b) is the *policy improvement* step. This decomposition of the policy iteration algorithm should be remembered in the next chapter, when comparing the dynamic programming approach with the genetic algorithm approach, since step 2(a) is identical in both methods.

5.5 Large-Scale Markov Decision Problems

This section gives a brief overview of the research effort for solving Large-Scale Markov decision problems. The emphasis is mainly on two different approaches to tackle large problems: one which uses genetic algorithms as its backbone, and the other which attempts to use dynamic programming modified to handle a large state space.

Problems such as the 4x3 model problem, which will be examined in the next chapter, are beneficial only if they enable a better understanding of larger, practical real-life problems. It will be seen that solving the 4x3 model problem can provide insights for solving larger problems. However, real-life problems are much bigger in size: they might contain hundreds or thousands of different states. Unfortunately, while many problems can be modeled as Markov decision problems, not all of these problems can be solved easily within this framework. Standard methods to solve a Markov decision process model with no further modifications require unreasonable amounts of memory and run-time. This section briefly discusses some ongoing research designed to solve Large-Scale Markov decision problems.

An approach which motivates the comparison between genetic algorithms and dynamic programming, being done in chapter 6, is the *symbiotic, adaptive neuro-evolution* [57]. It belongs to the general class of *reinforcement learning methods*, which are learning methods based on learning an input-output mapping through a process of trial and error designed to maximize a scalar performance index. The Symbiotic, adaptive neuro-evolution was designed as a method for forming decision strategies in domains where it is not possible to generate training data for normal *supervised learning* (in a supervised learning, there is an external teacher, having knowledge of the environment that is represented by a set of input-output examples. Supervised learning algorithms include the *least-mean-square* algorithms [58] and the *backpropagation* algorithm [59], both are classic examples of a neural network). Symbiotic, adaptive neuro-evolution maintains a population of possible strategies, evaluates the goodness of each from its performance in the domain, and unlike supervised learning uses an evolutionary algorithm to generate new strategies. The evolutionary algorithm modifies the strategies through genetic operators like selection, crossover, and mutation [2].

More details on the symbiotic, adaptive neuro-evolution can be found in [60]. Active researchers in the field who are using this approach for problems with large state spaces are D.E. Moriarty, from the Information Science Institute at the University of Southern California, and P. Langley, from the Daimler-Benz Research and Technology Center at Palo Alto, California. Problems which are of interest in this context, for example, include learning cooperative lane selection strategies for highways and traffic management control.

For such applications, the state space is so large that it is no longer possible to represent a policy as a simple vector mapping states to actions. The length of such a vector becomes too large, value determination steps become tremendously expensive, therefore building an accurate model becomes impractical. In these extreme conditions, policy iteration in its current form can not be used. Therefore, the only approach which is known at present to try and tackle huge state spaces is to use a compact representation of a policy, by mapping a partial description of the state to actions. The fitness of the policy representation can then be evaluated by measuring the actual (or simulated) performance from some initial condition. This avoids assigning a value to every state, and performance is improved by using genetic algorithms as the backbone of the symbiotic, adaptive neuro-evolution algorithm. No comparison to dynamic programming methods is possible in simulations of this kind.

On the other hand, much research effort is invested for developing ways to solve large Markov decision problems by using standard methods (i.e. dynamic programming) through hierarchy and decomposition. One idea is to use a method called *hierarchies of abstract machines*, which allows the incorporation of prior knowledge into the search for good policies. The knowledge contained in hierarchies of abstract machines is used to transform a large Markov decision problem into a smaller one that makes optimal use of the knowledge provided. Another idea is to decompose a large Markov decision problem into independent, or nearly independent subproblems. These subproblems can be solved separately by devising a cache of solutions for each subproblem, then finding the optimal way to combine the cached solutions. These decomposition methods provide a framework for the transfer of knowledge across problems with similar structures. Hierarchies of abstract machines and decomposition algorithms are complementary to each other and can be combined in novel ways.

More details on solving large Markov decision problems by Hierarchical Control and Learning can be found in [19]. The development of methods for large state spaces that are variations of policy iteration is of interest to several researchers, including R. Parr and D. Koller from the Robotics Laboratory at Stanford University, as well as groups at the University of California at Berkeley, Brown University, Duke University, University of Massachusetts at Amherst, and the University of British Columbia in Vancouver, Canada among many others. Another major effort which is related to this field of research, but will not be discussed here, is the development of methods to solve *partially observable Markov decision problems* [61,62].

Chapter 6

Genetic Search in Policy Space for Solving Markov Decision Processes

Following the introduction to Markov decision processes, as was presented in the previous chapter, the goal of this chapter is to examine the use of genetic algorithms to solve Markov decision problems.

In an earlier paper [17] on using genetic algorithms to solve a small sized finite Markov decision problem, no comparisons with existing techniques were reported. Therefore conclusions related to the success of the approach should be taken cautiously. In this chapter a comparison is made between an efficient genetic algorithm implementation and policy iteration, which is the most closely related iterative method to genetic algorithms, on an extended yet small sized infinite-horizon Markov decision problem. Such a comparison enables a better understanding of the strengths and weaknesses of a genetic algorithm approach. These lessons can prove useful when moving towards real-world applications with large state and action spaces, as well as more complicated model structures which are computationally intractable for existing techniques.

The chapter is divided as follows. Section 6.1 presents the model problem used for the comparison and the dynamic programming approach with which the genetic algorithm is compared. This section is further extended to include a discussion on discounting which will later reappear in the genetic algorithm implementation. Section 6.2 initiates the genetic algorithm approach by describing the binary representation of policies. In section 6.3, the issue of choosing the right fitness function evaluation for the genetic algorithm is examined. Section 6.4 compares several genetic algorithm evolution strategies and the most efficient one is then compared with the policy iteration algorithm. In section 6.5, conclusions are drawn with an eye towards the implementation of a genetic algorithm in larger problems.

6.1 Model Problem Presentation

The following model problem is taken from [53]. An agent is situated in a 4x3 grid-world model environment. The start state is in (1,1) and the goal state is in (4,3). The goal is to find an optimal Markov decision problem policy to reach the goal state using a set of four available actions: North, South, East, and West. The environment terminates when the agent reaches one of the states marked +1 or -1. It is also assumed that the agent knows which state it is in initially, and that it knows the effects of all of its actions on the state of the world. It is possible to introduce additional complications, in which not enough information is provided to determine the state or the associated transition probabilities. Such an extended version of the problem belongs to the class of partially observable Markov decision problems, and will not be treated here. The model environment in our case, a typical Markov decision problem is:

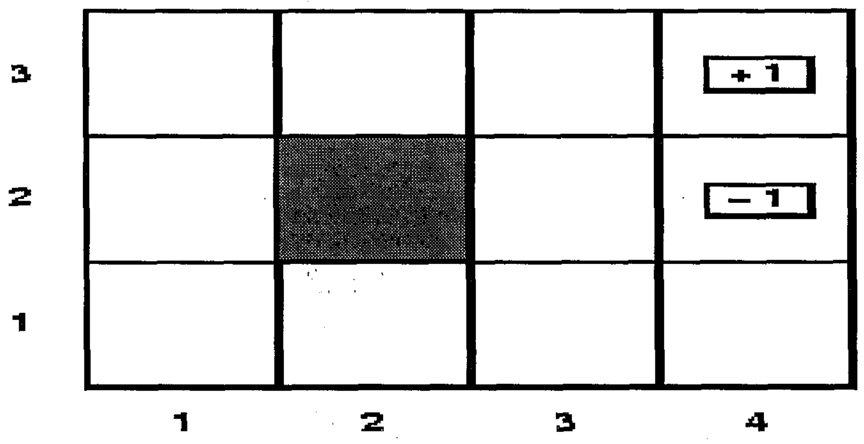


Figure 6.1: The 4x3 grid-world from Russell & Norvig [53]

The rules for this model problem are as follows: each action moves one square in the intended direction with probability 0.8, but the rest of the time the action moves the agent at right angles to the intended direction. For example, from the start square (1,1), the action North moves the agent to (1,2) with probability 0.8, but with probability 0.1 it moves moves East to (2,1) and with probability 0.1 it moves West, bumps into the wall and stays in (1,1). Whenever bumping into a wall, there is no change in the position. The reward R, associated with states, is a penalty of $-1/25$ for passing each state (long and repeated routes are not encouraged) except for +1 in (4,3) and -1 in (4,2).

The following notation will be used to describe the problem, its optimal policy solution and values of the states for the given optimal policy:

$$\begin{pmatrix} \star & \star & \star & +1 \\ \star & W & \star & -1 \\ \star & \star & \star & \star \end{pmatrix} \Rightarrow \begin{pmatrix} \rightarrow & \rightarrow & \rightarrow & +1 \\ \uparrow & W & \uparrow & -1 \\ \uparrow & \leftarrow & \leftarrow & \leftarrow \end{pmatrix}$$

$$\begin{pmatrix} 0.812 & 0.868 & 0.918 & +1 \\ 0.762 & W & 0.660 & -1 \\ 0.705 & 0.655 & 0.611 & 0.388 \end{pmatrix}$$

In this notation, W stands for wall, arrows $\rightarrow, \uparrow, \downarrow, \leftarrow$ are used to label all possible actions, while $\star$ denotes any of these actions. The solution to this problem, also given above, will now be derived following a description of the dynamic programming approach.

In a more abstract mathematical formulation, a Markov decision process is a 4-tuple, (S, A, T, R). S is a finite set of states s, A is a finite set of actions a. T is the transition model for the system, a mapping from $S \times A \times S$ into probabilities in $[0, 1]$, in particular $T(s, a, s') = P(s' \mid s, a)$. R is the reward function which maps from $S \times A \times S$ to real-valued rewards.

A policy π for a Markov decision problem is a mapping from states in S to actions in A. The value function V maps from elements of S to real values. The Bellman equation (see section 5.3) from dynamic programming establishes a relationship between $V_\pi(s)$ and V_π for other states in the model:

$$V_\pi(s) = R(s, \pi(s)) + \beta \sum_{s'} T(s, a, s') V_\pi(s') \tag{6.1}$$

Where $0 \leq \beta < 1$ is known as the discount factor. The optimal policy will be labeled as π^* and optimal value function as V^*. Since the optimal policy assigns the best action to every state, the goal is to find an optimal policy π^*, characterized by actions a, such that:

$$V^*(s) = \max_a [R(s, a) + \beta \sum_{s'} T(s, a, s') V^*(s')] \tag{6.2}$$

Two well known iterative procedures which aim at finding the optimal policy, based on the above equation, are value iteration and policy iteration (see section 5.4). Since it is more logical for a genetic algorithm implementation to search in policy space, the policy iteration algorithm is used for the comparison.

The value determination step can be implemented by either a successive approximation, or directly by solving a set of linear equations (see section 5.3). The latter can exploit the sparseness in the system, hence it is chosen for the model problem implementation. The Bellman equation for policy π can be written for each state in S, producing a linear system of equations in a matrix form (see end of section 5.3). To illustrate the value determination step, the linear system which results from taking the optimal policy in the model problem, the one which produces the final values of the states solution, is:

$$\left(\begin{array}{ccccccccccc|c}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1.0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1.0 \\
0 & 0 & 0.8 & -0.9 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0.04 \\
0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & -0.9 & 0.8 & 0.04 \\
0 & 0 & 0 & 0 & 0 & -0.9 & 0.1 & 0 & 0 & 0.8 & 0 & 0.04 \\
0 & 0.8 & -0.9 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.04 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.8 & 0.8 & 0 & 0.04 \\
0.8 & -0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.04 \\
0 & 0 & 0 & 0 & 0.1 & 0 & 0 & -0.9 & 0.8 & 0 & 0 & 0.04 \\
0 & 0 & 0 & 0 & -0.8 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0.04 \\
-0.9 & 0.1 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.04
\end{array}\right)$$

Attention is now turned to the discount factor β. In [53], for simplicity reasons, the Bellman equation does not contain discounting. While setting $\beta = 1$ does not cause a problem in the original model, consider the following slight modifications to the model:

$$\begin{pmatrix} \uparrow & W & \star & +1 \\ \star & \star & \star & -1 \\ \star & \star & \star & \star \end{pmatrix} or \begin{pmatrix} \uparrow & \uparrow & W & +1 \\ \star & W & \star & -1 \\ \star & \star & \star & \star \end{pmatrix}$$

In both cases the policy iteration, which without loss of generality starts in the model problem with an initial policy in which all arrows are lined upwards, will fail in the first value determination step. Without discounting, conservation of probability for each state s_i holds:

$$\sum_{k=1}^{N} T(s_i, \pi(s_i, s_k)) = 1 \tag{6.3}$$

So that the sum of each row in the left-hand side of the system amounts to zero. In two cases, this can lead to no solution of the linear system and therefore $0 \leq \beta < 1$ is required to break the conservation. In the first case, corresponding to the left modification of the above model, one state interacts only with itself. The state (1,3) will produce a Bellman equation which has no solution. In the second case, shown in the right, only two states interact with each other. The states (1,3), (2,3) will create two Bellman equations that contradict each other, which again results in no solution. To avoid destructive two-states interactions as well as one state interacting with itself, a discount factor $\beta = 0.99$ is added from now on throughout the chapter to the model problem. However, at the price of fixing cases of destructive interferences, the discounted model is different than the original model and the optimal policy, when calculated, will differ. For the $-1/25$ penalty in R, as in [53], the optimal policy for the state (3,1) will cease to be conservative: it will no longer recommend taking the long way around in order to avoid the risk of entering (4,2). Therefore, along with introducing $\beta = 0.99$, the penalty is decreased to $-1/50$, resulting in the same optimal policy as in the original model.

The above discussion on discounting is highly relevant to the genetic algorithm implementation and will further be discussed when formulating the fitness function.

6.2 Representation

The first task in a genetic algorithm is to code the decision variables into binary strings. It is customary to work with a binary representation for which the schema theorem (see section 1.5.1) applies. Therefore, the search space which is the most natural to use in a Markov decision problem is policy space, rather than values of states which are a collection of real numbers.

In the model problem, there are four possible actions in A. These can be coded into $10, 11, 00, 01$ for $\rightarrow, \uparrow, \downarrow, \leftarrow$, respectively. A policy can then be constructed by concatenating the various actions at each state, to form a string of binary bits of length $2 * N$, where N is the number of states. Such a binary string to represent a policy is called a chromosome.

Out of eleven states in the 4x3 example, two are forced to choose a prescribed action. States $(4, 2), (4, 3)$ will always choose their action to be $\uparrow$, therefore each chromosome is of length $2 * 9 = 18$. Indexing is chosen to start at state (1,1), then move right to (4,1), then upwards to (1,2), then right to (3,2), repeatedly until termination at (3,3). For example, the optimal policy π^* is represented by the chromosome [110101011111101010]. Solely for investigative purposes (figure 6.2 in the next section), a real number representation of chromosomes was included in the simulation. For that reason, all possible binary combinations ($2^{18} = 262144$) were taken between -13.1072 and 13.1072, with a grid spacing of 0.0001. In that case, the optimal policy π^* amounts to 8.80 when converted to a real-number.

6.3 Fitness Function Evaluation

For each representation of a policy by a chromosome, a fitness value is assigned. Constructing the best fitness function, which maps from policies to fitness values, so that the genetic algorithm evolves efficiently, is of central importance in the implementation.

The value determination step of policy iteration produces a vector V of N elements, each of which contains the value $V(s_i)$ at state s_i. Taking the separable value function to be additive, the fitness value of a policy π can then be formulated by summing over all elements of V:

$$f(\pi) = (1/N)\sum_{i=1}^{N} V(s_i) \tag{6.4}$$

In a Markov decision problem, in general, an objective function has several choices. Among them are expected average reward (reward per step), expected total undiscounted reward and expected total discounted reward. For reasons already mentioned earlier, expected total discounted reward is mostly favored for the genetic algorithm implementation when using value determination. In addition to the previous example in the presentation section, where two states are interacting with each other to produce no solution in a slightly modified version of the original model when using policy iteration, all policy space is vulnerable in a genetic algorithm search. Below is an example of a configuration in the original model, which does not arise in policy iteration, that needs to be avoided in the genetic algorithm implementation by way of discounting:

$$\begin{pmatrix} \star & \star & \star & +1 \\ \downarrow & W & \star & -1 \\ \leftarrow & \star & \star & \star \end{pmatrix}$$

The two states which are interacting destructively in this example are (1,1) and (1,2). Therefore, $V(s_i)$ is chosen to be calculated by value determination using a discounted model.

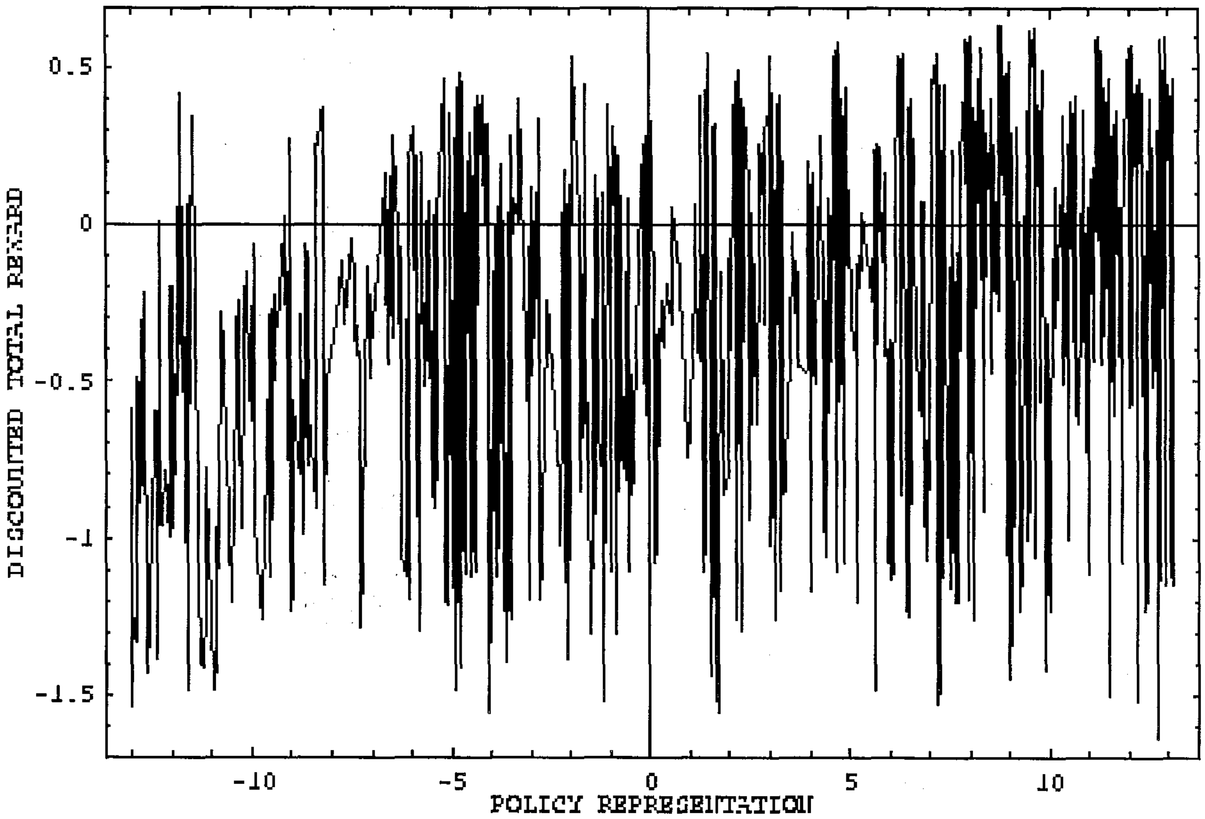


Figure 6.2: Fitness function used in the genetic algorithm simulation

Figure 6.2 illustrates how the fitness function, given in equation (6.4), is noisy and unpredictable. Policies are represented in the horizontal axis by converting binary strings into real numbers between -13.1072 and 13.1072 (although throughout the genetic algorithm evolution, they obviously evolve as chromosomes in a binary form). It is therefore evident that for searching the whole policy space, without favoring any certain subsections of the space initially, a genetic algorithm is perhaps the most efficient technique to locate the global maximum in order to find the optimal policy.

6.4 Evolution Strategies

Finally, an initial population of random policies and their respective fitness values are evolved for several generations using selection, reproduction, crossover and mutation, until convergence to the optimal policy. Figure 6.3 illustrates some of the genetic algorithm terminology used so far (see chapter 1 for exact details and definitions).

The genetic operators which are used to solve the model problem (i.e. crossover and mutation), as well as the chosen genetic algorithm parameter values, are the ones which are commonly used [2] to search for a global maximum in a noisy function similar to the one in figure 6.2. A modern genetic algorithm implementation by [10], which is freely distributed, was modified to handle the 4x3 model-environment, as well as representation of policies and value determination by means of Gaussian Elimination for the fitness function evaluation.

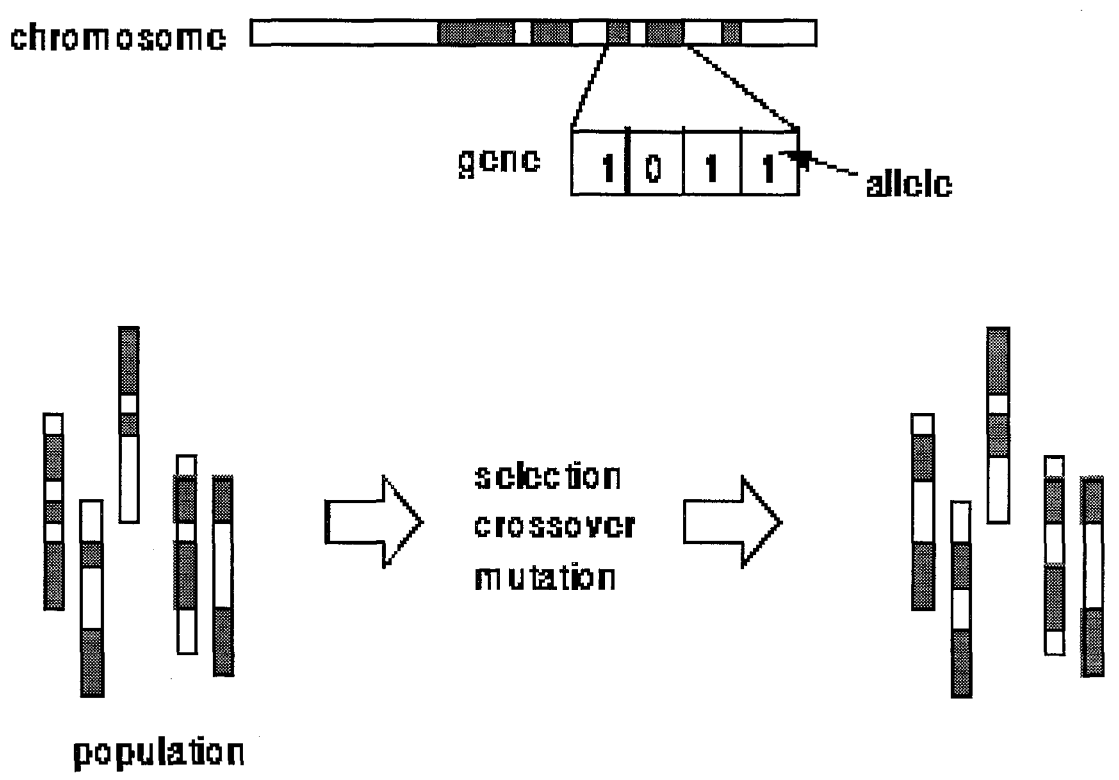


Figure 6.3: Genetic algorithm basic terminology

The above implementation includes a variant of genetic algorithms which allows a small population size to be evolved. Typically the population size is set to be at least $N_p = 50$. However, an approach called micro genetic algorithm (see section 1.4.2) works with a small population size of $N_p = 5$. In both simple genetic algorithms and micro genetic algorithms, selection is done based on tournament selection and uniform crossover is used. However, no mutations are necessary in a micro genetic algorithm since there is a constant infusion of new schema at regular intervals. It is also worthwhile mentioning that sensitivity experiments on this problem, by slightly modifying basic genetic algorithm parameters, yielded no better results.

In figure 6.4, the simple genetic algorithm for two typical population sizes ($N_p = 50$, $N_p = 100$) and the micro genetic algorithm ($N_p = 5$) are compared in terms of function evaluations (the number of generations required for convergence, multiplied by population size). It turns out that for searching policy space in the 4x3 model problem, the micro genetic algorithm works best performancewise. Convergence to the optimal policy was achieved after 8 generations for $N_p = 100$, 13 generations for $N_p = 50$ and 56 generations for $N_p = 5$, resulting in the least amount of function evaluations ($56 * 5 = 280$) when the micro genetic algorithm was used.

Using the policy iteration algorithm, in which the same value determination as in the genetic algorithm is performed in each iteration, only 5 iterations were needed to reach the optimal policy. Policy improvement is therefore surprisingly more efficient than a genetic algorithm for a problem with a small state space.

Finally, it is instructive to examine the intermediate steps before reaching the optimal policy in both methods. In policy iteration, after the first iteration (20% of total calculation), the policy at hand is quite close to the optimal: only three actions in three states are needed to be adapted in consequent steps. No 'illogical' policies

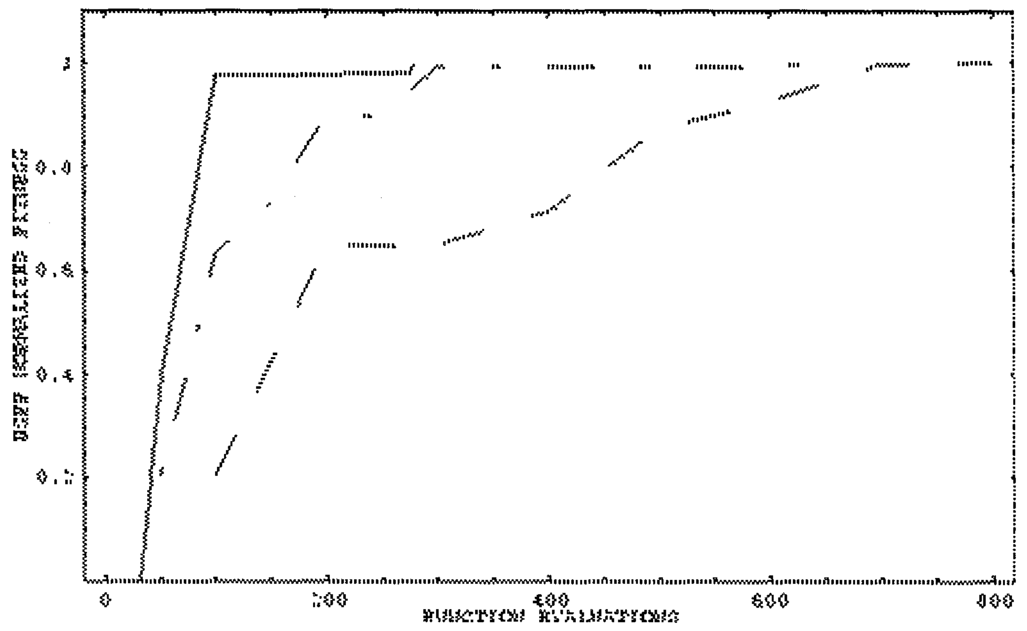


Figure 6.4: Best-So-Far-Fitness genetic algorithm performance evaluation: $N_p = 100$ (dashed), $N_p = 50$ (dot-dashed), $N_p = 5$ (solid).

are considered, such as an action somewhere pointing to a wall with no apparent reason. In addition, policies are consistently improving at each step. In the micro genetic algorithm, after 11 generations (roughly 20% of total calculation), the best-so-far policy differs from the optimal one by four actions (three of which point to a wall), clearly not as good as policy iteration in the early stage. Also, at a later generation it goes downhill, from what seems to be a good policy to a lesser one, in order to emerge with an improved policy after one more generation. This behavior is attributed to the different nature of the two methods. With respect to certain 'illogical' actions, perhaps restricting the search space in the genetic algorithm implementation rather than leaving it flexible can yield a better performance, if such a strategy proves to be cost effective.

6.5 Conclusion

In this section, conclusions are drawn which are related to the comparison between genetic algorithms and dynamic programming on the model problem.
A genetic algorithm approach was implemented to solve a Markov decision process model problem with small state and action spaces. It was found that a discounted model is required in order to search policy space using value determination steps. The approach was optimized to perform in the most efficient manner, using a genetic algorithm variant called micro genetic algorithm.

Comparison with the policy iteration algorithm reveals that for relatively small problem sizes, policy iteration outperforms the genetic algorithm. Larger examples reveal that the inefficiency of genetic algorithms become even more pronounced, since policy iteration converges in a surprisingly small number of iterations. Policy iteration only examines a small portion of policy space right from the beginning whereas the genetic algorithm starts initially with a random distribution covering all space. Therefore, the workload the genetic algorithm is facing is by far heavier than that of policy iteration.

On the other hand, it should be noted that in very large state spaces, it is no longer possible to represent a policy as a simple vector mapping states to actions. As was described in section 5.5, in such cased only the symbiotic, adaptive neuro-evolution reinforcement learning method is known to be applicable. No comparison to dynamic programming methods is possible in simulations of this kind.

To conclude, based on the comparison in this chapter, it is unlikely that searching policy space by using genetic algorithms can offer a competitive approach in cases where the policy iteration algorithm can be implemented. Evolutionary algorithms offer an approach, combined with neural networks, to search very large state spaces in cases where dynamic programming methods are no longer valid. While further advancement on the evolutionary track is to be expected, it remains a challenge to develop methods which are based on policy iteration for making decisions in large state spaces.

Chapter 7

Conclusions and Future Work

In the previous chapters, genetic algorithms were implemented for two different problems. In the first problem, genetic algorithms were *successful* at identifying an optimal frequency. It consequently led to explaining dynamic stabilization by the appearance of laser-induced resonance states which trap ionized population. The second problem was taken from the area of artificial intelligence. The goal was to check how genetic algorithms are able to compete with a traditional technique, deterministic in nature, which has been used for many years to identify an optimal policy in a Markov Decision Process. Although genetic algorithms did find the correct solution, dynamic programming proved to be significantly more efficient when both methods are applicable. For that problem, genetic algorithms proved to be *unsuccessful* compared to an alternative method which can be used.

In the first section, conclusions are drawn and recommendations are given regarding which problems are potentially suited for a successful genetic algorithm implementation. The last section suggests future work concerning the implementation of genetic algorithms on various problems.

7.1 Conclusion

In the first part of this dissertation, the dynamic stabilization effect was explained. It was shown that for a short-ranged potential having a single bound state, ionization suppression is caused by the appearance of a laser induced resonance state, which is coupled by the pulse turn on/off ramp to the ground state and traps ionizing flux. The relation between dynamic stabilization and the well understood adiabatic stabilization was found.

Genetic algorithms are general methods for solving search problems. The same methodology can be implemented on a variety of problems. In the second part of this dissertation, a comparison was done between genetic algorithms and dynamic programming which is a method designed to work on particular problems. Genetic algorithms were shown to be inefficient compared to dynamic programming.

Thus, it is unlikely that a general method such as genetic algorithms can compete well with problem-specific methods, designed to exploit the structure in a problem, when implemented on the same model-problem. It can be argued that for each problem, genetic algorithms can be tuned for best performance. Depending on the problem, a fitness function can be constructed to exploit knowledge, a good representation can be chosen and the genetic algorithm parameters can be optimized. While all these issues are important and should be carefully implemented to maximize efficiency, the mechanism itself is stochastic in nature and general. Therefore, the flexibility in the implementation is limited compared to a method for which the mechanism itself is designed to exploit structure and symmetry. The conclusion is that in many problems for which specific non-stochastic algorithms were designed, it is expected that even the best genetic algorithm implementation will not be able to compete against such methods when applied to the same model-problem.

However, there are problems for which it is difficult to come up with specific methods since the information about the problem is limited or obscured. In such problems, general stochastic methods such as genetic algorithms or simulated annealing are a way to make progress. Genetic algorithms can then offer a convenient alternative since they are flexible and easy to use. Conceptually, simulated annealing offers a very similar mechanism without the need to invent new terminology based on biological evolution in order to describe the algorithm.

7.2 Future Work

Implementation of genetic algorithms for a variety of problems involves both software and hardware considerations. Their first application in atomic and molecular physics [8] was done by incorporating genetic algorithms as part of an experimental apparatus. Laboratory experiments were then fed to the computer in order to design a new sequence of experiments.

Since then, many laboratories are using genetic algorithms in real-time during experiments in order to automatically construct new ones. In addition to laser control experiments, they were used in low energy electron diffraction experiments [63] and their use in processing experimental data is expected to increase in the future. At Hewlett-Packard Laboratories, a high-performance hardware implementation of a genetic algorithm is being designed [64]. A $2200\times$ speed-up over software emulation on a $100MHz$ workstations was reached.

In the software implementation side of a genetic algorithm, a real-number representation instead of the binary representation [65] is expected to be used in an increasing number of applications. Extensions to genetic algorithms such as genetic programming [66] can be used to search in the space of programs to locate which program is best to perform a certain task. These extensions will keep developing in order to solve a variety of real-world problems.

In laser-atom interactions, calculations which were applied in this dissertation to a simplified one-dimensional model of a hydrogen atom can be extended to higher dimensions and molecules. A genetic algorithm can be implemented to search in multi-dimensional parameter space which includes, in addition to frequency and intensity, the pulse shape and duration. Apart from atomic stabilization, different effects can be studied such as high-harmonic generation using different colors (distinct frequencies). Genetic algorithms can then assist in optimal control problems, such as constructing an optimal pulse to drive a chemical reaction down a preferred pathway.

— THE END —

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Bibliography

1. Holland J.H. 1975. *Adaptation in Natural and Artificial Systems.* Cambridge, MA.: MIT Press.

2. Goldberg D.E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA.: Addison-Wesley.

3. Davis L., Ed. 1987. *Genetic Algorithms and Simulated Annealing.* London : Pitman ; Los Altos, CA. : Morgan Kaufmann Publishers.

4. Krishnakumar K. 1989. Micro-Genetic Algorithms for Stationary and Non- Stationary Function Optimization. *SPIE: Intelligent Control and Adaptive Systems* 1196:289.

5. Grefenstette J.J. 1981. Parallel Adaptive Algorithms for Function Optimization. *Technical Report No. CS-81-19.* Nashville: Vanderbilt University, Computer Science Department.

6. Prugel-Bennett A., Shapiro J.L. 1994. An Analysis of Genetic Algorithms using Statistical Mechanics. *Physical Review Letters* 72:1305.

7. Goldberg D.E. 1990. A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing. *Complex Systems* 4:445.

8. Judson R.S., Rabitz H. 1992. Teaching Lasers to Control Molecules. *Physical Review Letters* 68:1500.

9. Krause J.L. et al 1997. Quantum Control in Quantum Wells. *Physical Review B (Condensed Matter)* 57:9024.

10. Carroll D.L. 1996. Chemical Laser Modeling with Genetic Algorithms. *American Institute of Aeronautics and Astronautics* 34(2):338.

11. Su Q., Irving B.P., Johnson C.W., Eberly J.H. 1996. Stabilization of a One-Dimensional Short-Range Model Atom in Intense Laser Fields. *Journal of Physics B: At.Mol.Opt.Phys* 29:5755.

12. Geltman S. 1995. Are Atoms Stabilized by Ultraintense Lasers? *Chemical Physics Letters* 237:286.

13. Geltman S. 1999. Comment on 'Stabilization of a One-Dimensional Short-Range Model Atom in Intense Laser Fields'. *Journal of Physics B: At.Mol.Opt.Phys* 32:853.

14. Barash D., Orel A.E., Vemuri R.V. 1998. A Genetic Search in Frequency Space for Stabilizing Atoms by High-Intensity Laser Fields. *Journal of Computing and Information Technology*, Submitted.

15. Barash D., Orel A.E., Baer R. 1999. Laser Induced Resonance States as a Cause for Dynamic Suppression of Ionization in High Frequency Short Pulses. *Physical Review Letters*, Submitted.

16. Bellman R. 1957. *Dynamic Programming.* Princeton, N.J.: Princeton University Press.

17. Chin H.H., Jafari A.A. 1998. Genetic Algorithm Methods for Solving the Best Stationary Policy of Finite Markov Decision Processes. In *Proceedings of the Thirtieth Southeastern Symposium on System Theory.* Morgantown, West Virginia.: IEEE Press.

18. Moriarty D., Langley P. 1998. Learning Cooperative Lane Selection Strategies for Highways. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence.* Madison, WI.: AAAI Press.

19. Parr R. E. 1998. *Hierarchical Control and Learning for Markov Decision Processes.* Ph.D. diss., Dept. of Computer Science, University of California at Berkeley.

20. Barash D. 1999. A Genetic Search in Policy Space for Solving Markov Decision Processes. In *AAAI Spring Symposium on Search Techniques for Problem Solvingunder Uncertainty and Incomplete Information.* Stanford, CA.: AAAI Press.

21. Einstein A. 1905. *Ann. Phys.* 17:132.

22. Agostini P. et al. 1979. Free-Free Transitions Following Six-Photon Ionization of Xenon Atoms. *Physical Review Letters* 42:1127.

23. Kruit P. et al. 1983. Electron Spectra from Multiphoton Ionization of Xenon at 1064, 532, and 355 nm. *Physical Review A (Atomic, Molecular and Optical Physics)* 28:248.

24. McPherson A. et al. 1987. Studies of Multiphoton Production of Vaccum-Ultraviolet Radiation in the Rare Gases. *Journal of the Optical Society of America B* 4:595.

25. Ferray M. et al. 1988. Multiple-Harmonic Conversion of 1064nm Radiation in Rare Gases. *Journal of Physics B: At.Mol.Opt.Phys* 21:L31.

26. Goldberg A., Schey H.L. 1967. Computer-Generated Motion Pictures of One-Dimensional Quantum-Mechanical Transmission and Reflection Phenomena *American Journal of Physics* 35, 3:177.

27. Gersten J.I., Mittleman M.H. 1976. The Shift of Atomic States by Laser Field. *Journal of Physics B: At.Mol.Opt.Phys* 9:2561.

28. Gavrila M., Kaminski J.Z. 1984. Free-Free Transitions in Intense High-Frequency Laser Fields. *Physical Review Letters* 52:613.

29. Gavrila M., Ed. 1992. *Atoms in Intense Laser Fields.* Boston, MA.: Academic Press.

30. Su Q., Irving B.P., Eberly J.H. 1997. Ionization Modulation in Dynamic Stabilization. *Laser Physics* 7(3):568

31. Protopapas M., Keitel C.H., Knight P.L. 1997. Atomic Physics with Super-High Intensity Lasers. *Reports on Progress in Physics.* 60(4):389.

32. de Boer M.P. et al. 1993 Indications of Adiabatic Stabilization in Neon. *Physical Review Letters* 71:3263.

33. Van Druten N.J. et al. 1997. Adiabatic Stabilization: Observation of the Surviving Population. *Physical Review A (Atomic, Molecular and Optical Physics)* 55:622.

34. Bardsley J.N., Szoke A., Comella M.J. 1988. Multiphoton Ionisation from a Short-Range Potential by Short-Pulse Lasers. *Journal of Physics B: At.Mol.Opt.Phys* 21:3899.

35. Smith B.T. et al. 1976. *Matrix Eigen-System Routines - Eispack Guide.* Springer-Verlag.

36. Goffe W.L., Ferrier G.D., Rogers J. 1994. Global Optimization of Statistical Functions with Simulated Annealing. *Journal of Econometrics* 60:65.

37. Peskin U., Kosloff R., Moiseyev N. 1994. The Solution of the Time Dependent Schrödinger Equation by the (t,t') Method: The Use of Global Polynomial Propagators for Time Dependent Hamiltonians. *Journal of Chemical Physics* 100(12):8849.

38. Yao G., Wyatt R.E. 1994. Stationary Approaches for Solving the Schrödinger Equation with Time-Dependent Hamiltonians. *Journal of Chemical Physics* 101(3):1904.

39. Feit M.D., Fleck J.A., Steiger A. 1982. Solution of a Schrödinger Equation by a Spectral Method. *Journal of Computational Physics* 47:412.

40. Kosloff R., Kosloff D. 1983. A Fourier Method Solution for the Time Dependent Schrödinger Equation. *Journal of Chemical Physics* 79:1823.

41. Kulander K.C., Schafer K.J., Krause J.L. 1992. Time-Dependent Studies of Multiphoton Processes. In *Gavrila M.,Ed.; Atoms in Intense Laser Fields.* 247.

42. Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. 1992. *Numerical Recipes, Second Edition.* Cambridge, England; New-York: Cambridge University Press.

43. Askar A., Cakmak A.S. 1978. Explicit Integration Method for the Time-Dependent Schrödinger Equation for Collision Problems. *Journal of Chemical Physics* 68:2794.

44. Tal-Ezer H., Kosloff R. 1984. An Accurate and Efficient Scheme for Propagating the Time-Dependent Schrödinger Equation. *Journal of Chemical Physics* 81:3697.

45. Howland J.S. 1974. Stationary Scattering Theory for Time-Dependent Hamiltonians. *Math. Ann.* 207:315.

46. Pfeifer P., Levine R.D. 1983. A Stationary Formulation of Time-Dependent Problems in Quantum Mechanics. *Journal of Chemical Physics* 79:5512.

47. Peskin U., Moiseyev N. 1993. The Solution of the Time-Dependent Schrödinger Equation by the (t,t') Method: Theory, Computational Algorithm and Applications. *Journal of Chemical Physics* 99:4590.

48. Strang G. 1986. *Introduction to Applied Mathematics.* Wellesley, MA.: Wellesley-Cambridge Press.

49. Henneberger W.C. 1968. Perturbation Method for Atoms in Intense Light Beams. *Physical Review Letters* 21:838.

50. Neuhauser D., Baer M. 1989. The Time-Dependent Schrödinger Equation: Application of Absorbing Boundary Conditions. *Journal of Chemical Physics* 90:4351.

51. Berry M.V., 1984. Quantal Phase Factors Accompanying Adiabatic Changes. *Proceedings of the Royal Society of London A* 392:45.

52. Puterman M.L. 1994. *Markov Decision Processes.* New York, NY.: John Wiley and Sons.

53. Russell S.J., Norvig P. 1995. *Artificial Intelligence: A Modern Approach.* Upper Saddle River, N.J.: Prentice Hall.

54. Blackwell D. 1962. Discrete Dynamic Programming. *Annals of Mathematical Statistics* 33:719.

55. Sutton R.S. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning.* 3:9.

56. Bertsekas D.C., Tsitsiklis J.N. 1989. *Parallel and Distributed Computation : Numerical Methods.* Englewood Cliffs, N.J. : Prentice Hall.

57. Moriarty D.E., Miikkulainen R. 1996. Efficient Reinforcement Learning Through Symbiotic Evolution. *Machine Learning* 22:11.

58. Widrow B., Hoff M.E. 1960. Adaptive Switching Circuits. *IRE WESCON Convention Record.*

59. Werbos P.J. 1974. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.D. diss., Harvard University. Cambridge, MA.

60. Moriarty D.E. 1997. *Symbiotic Evolution of Neural Networks in Sequential Decision Task.* Ph.D. diss., Department of Computer Science, The University of Texas at Austin.

61. Monahan G.E. 1982. A Survey of Partially Observable Markov Decision Processes. *Management Science* 28:16.

62. Lovejoy W.S. 1991. A Survey of Algorithmic Methods for Partially Observed Markov Decision Processes. *Annals of Operations Research* 28:47.

63. Doll R., Van Hove M.A. 1996. Global Optimization in LEED Structure Determination Using Genetic Algorithms. *Surface Science* 355: L393.

64. Shackleford B. et al. 1997. A High-Performance Hardware Implementation of a Survival-Based Genetic Algorithm. In *Progress in Connectionist-Based Information Systems. Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems.* Singapore: Springer-Verlag Singapore.

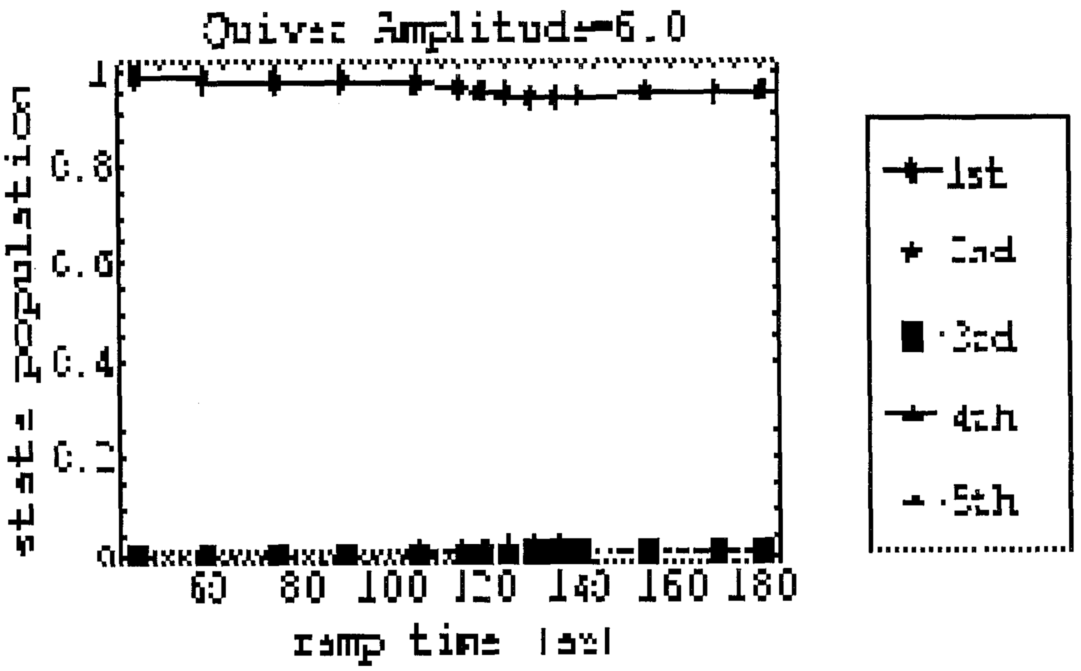
65. Michalewicz Z. 1994. *Genetic Algorithms + Data Structures = Evolution Programs, Second Edition.* Berlin ; New York : Springer-Verlag.

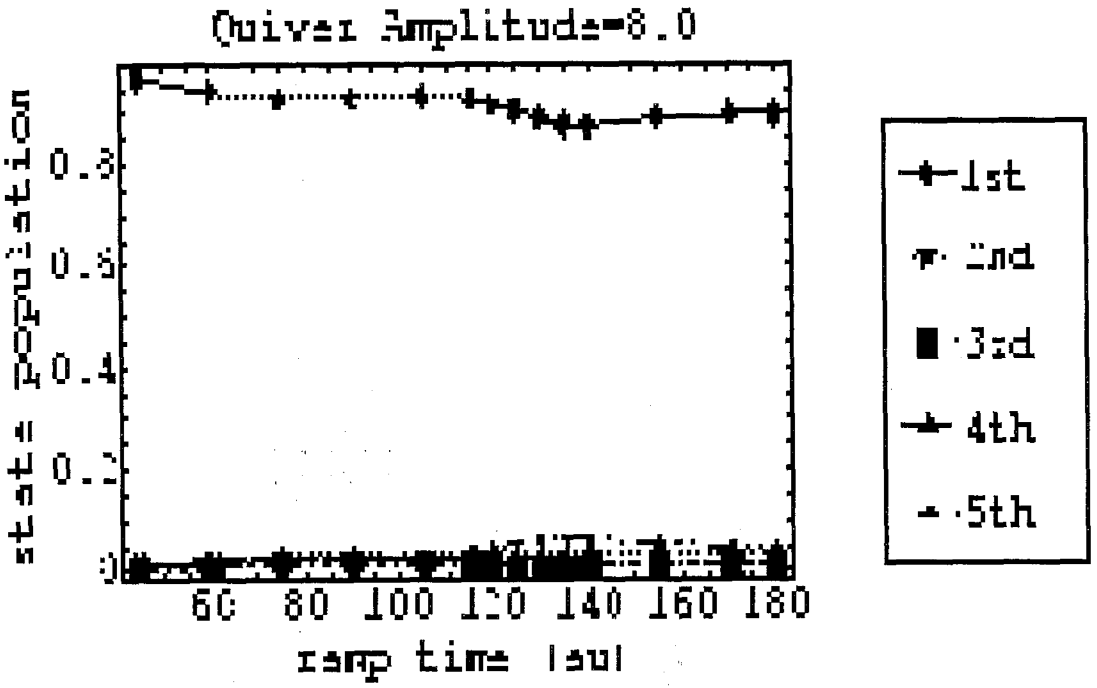
66. Koza J.R. 1992. *Genetic Programming : On the Programming of Computers by Means of Natural Selection.* Cambridge, MA. : MIT Press.

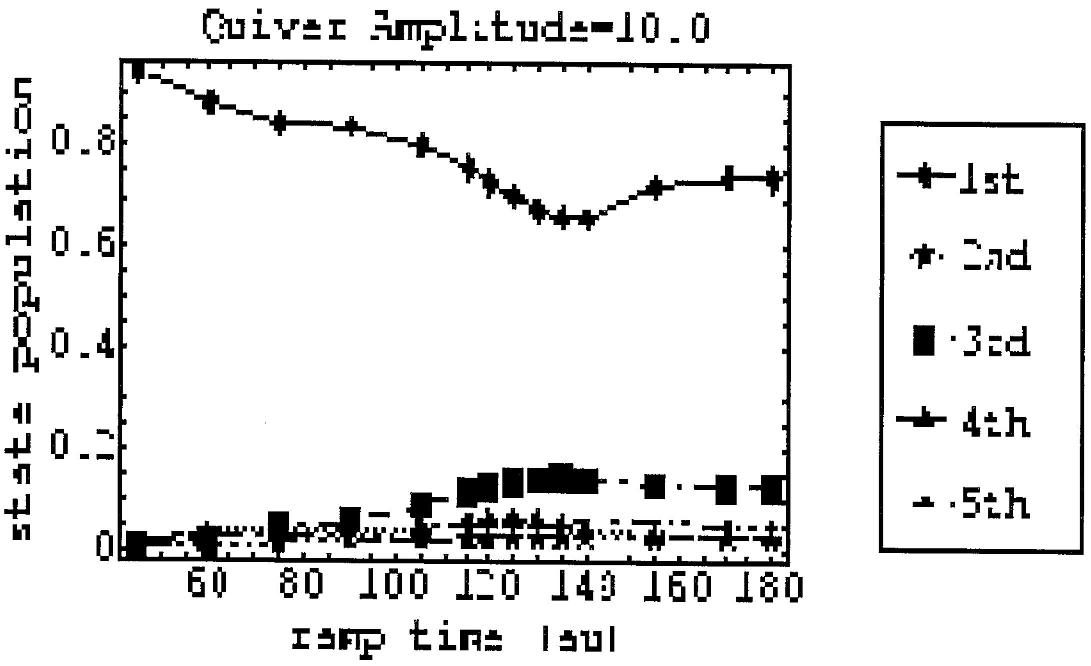
Appendix A

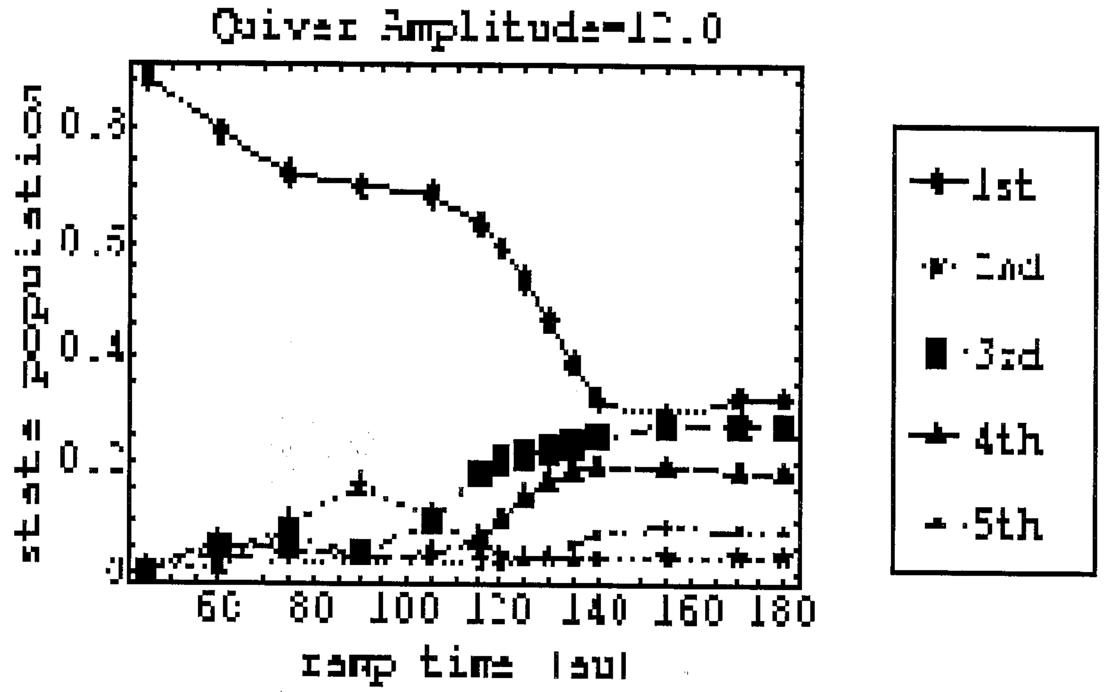
Appendix

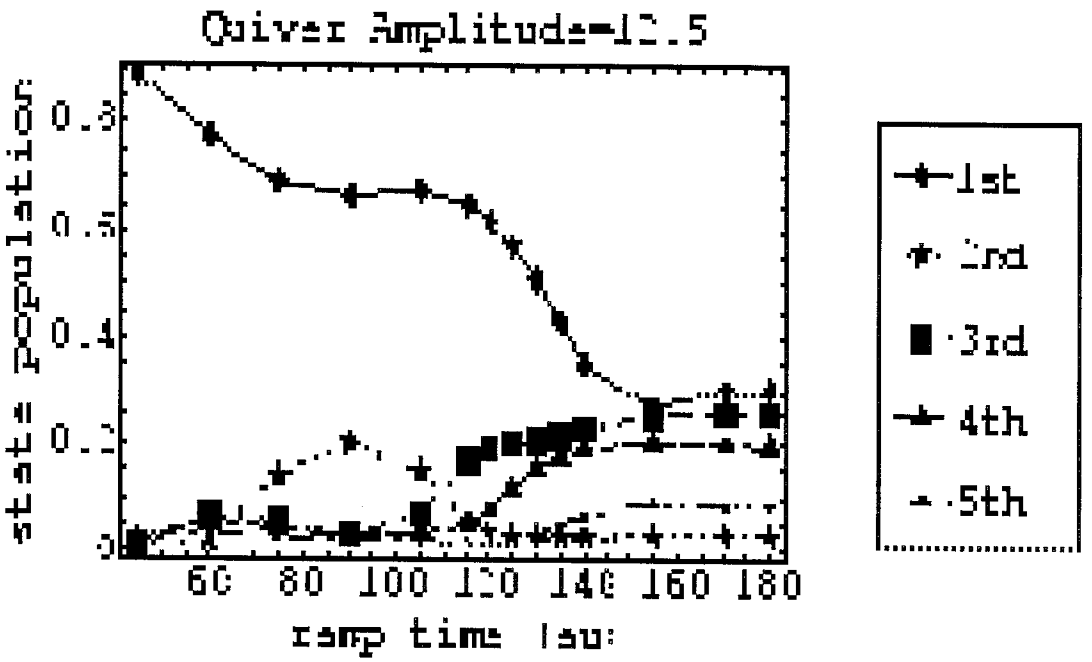
The population of the 5 lowest eigenstates for various ramp times during the rise and decay of the pulse used in chapter 4 are shown in this appendix. Figure 4.2 highlights some of the important times. Note that only the turn-on and turn-off of the pulse are contributing to population change, therefore times when the pulse envelope is flat were taken out.

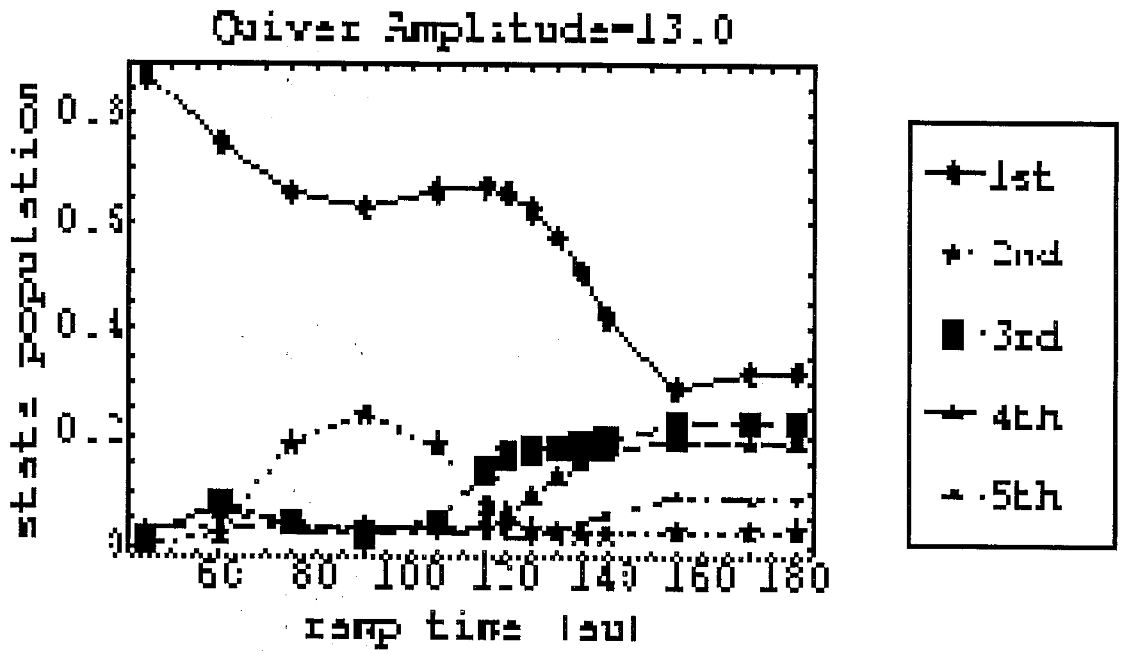

Quiver Amplitude=6.0
1
0.8
0.6
0.4
0.2
0
60
80
100
120
140
160
180
ramp time [au]
state population
1st
2nd
3rd
4th
5th

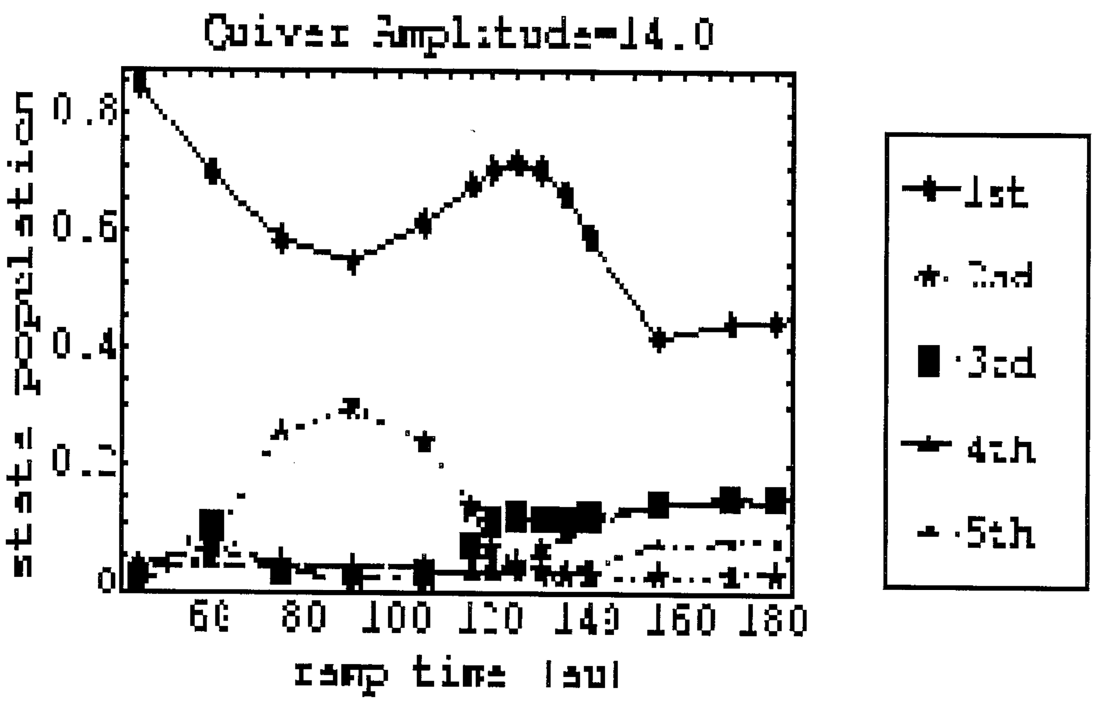

Quiver Amplitude=8.0
0.8
0.6
0.4
0.2
0
60
80
100
120
140
160
180
ramp time [au]
state population
1st
2nd
3rd
4th
5th


Quiver Amplitude=10.0
state population
0
0.2
0.4
0.6
0.8
60
80
100
120
140
160
180
ramp time (au)
1st
2nd
3rd
4th
5th

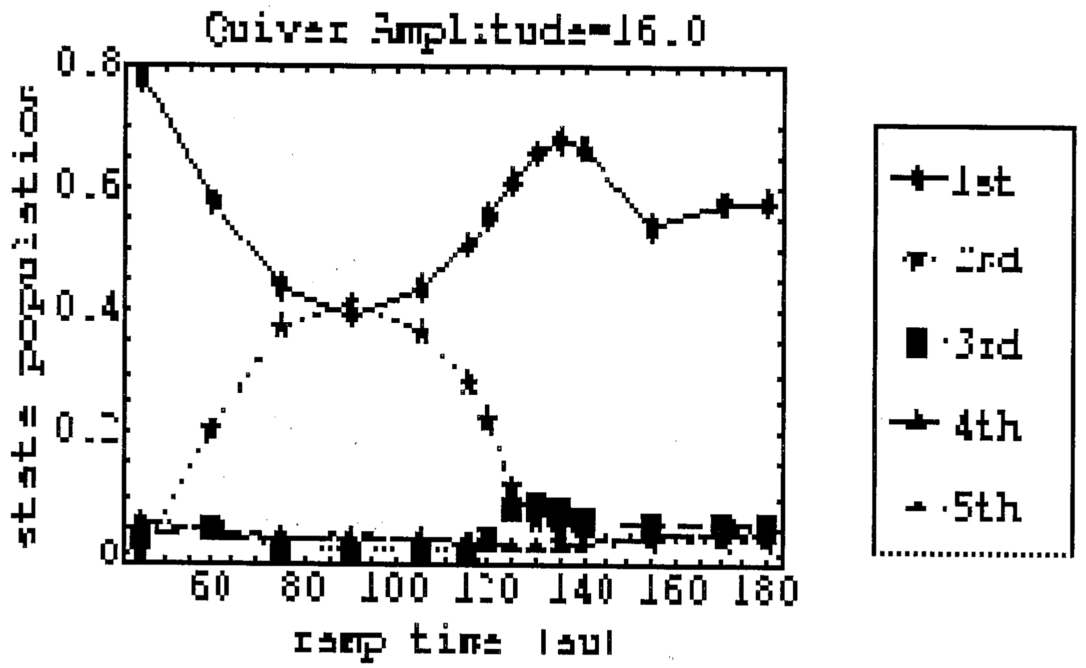

Quiver Amplitude=12.0
state population
0
0.2
0.4
0.6
0.8
60
80
100
120
140
160
180
ramp time (au)
1st
2nd
3rd
4th
5th


Quiver Amplitude=12.5
state population
0
0.2
0.4
0.6
0.8
60
80
100
120
140
160
180
ramp time (au)
1st
2nd
3rd
4th
5th

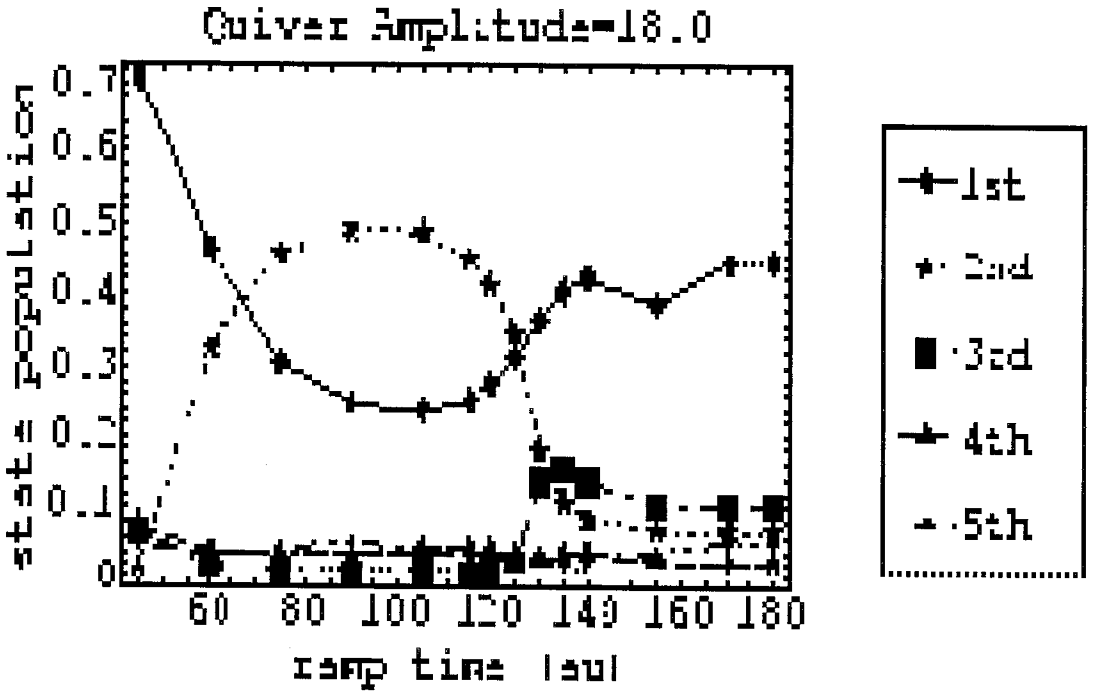

Quiver Amplitude=13.0
state population
0
0.2
0.4
0.6
0.8
60
80
100
120
140
160
180
ramp time (au)
1st
2nd
3rd
4th
5th

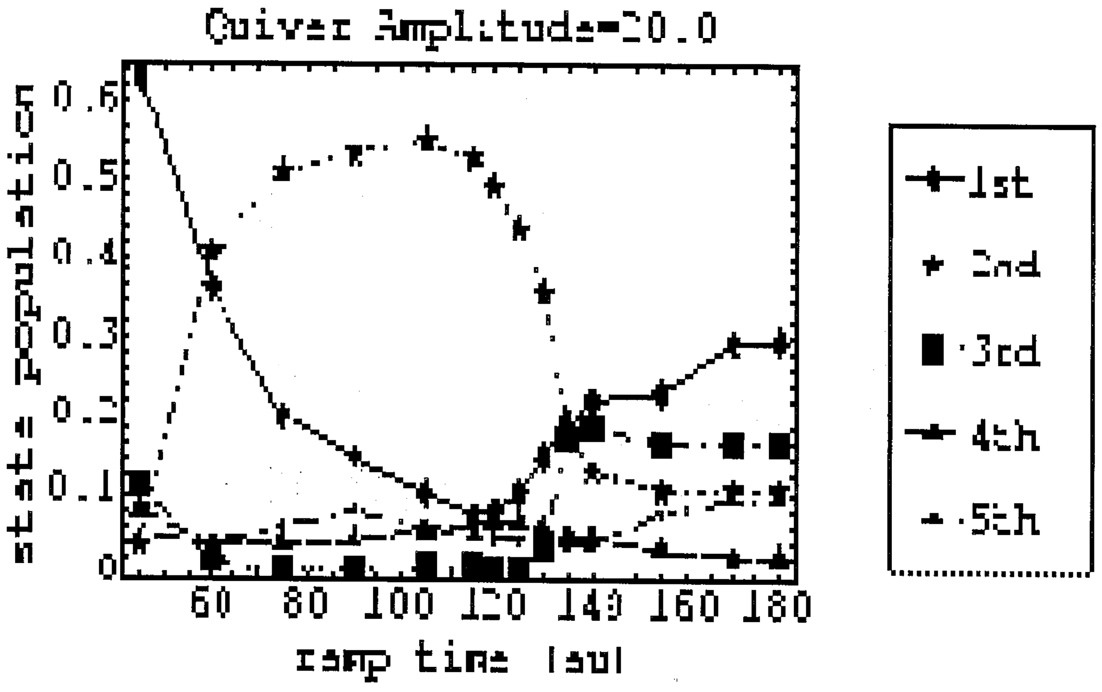


Quiver Amplitude=14.0
state population
ramp time (au)
1st
2nd
3rd
4th
5th



Quiver Amplitude=16.0
state population
ramp time (au)
1st
2nd
3rd
4th
5th


Quiver Amplitude=18.0
state population
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
60
80
100
120
140
160
180
ramp time (au)
1st
2nd
3rd
4th
5th


Quiver Amplitude=20.0
state population
0
0.1
0.2
0.3
0.4
0.5
0.6
60
80
100
120
140
160
180
ramp time (au)
1st
2nd
3rd
4th
5th