REU-Project: Laser pulse dependence in multiphoton ionization of atomic hydrogen

Klaus Bartschat

Department of Physics and Astronomy,

Drake University, Des Moines, Iowa 50311, USA

(Dated: February 18, 2014)

Abstract

Below are instructions for an REU project of a hydrogen atom in a strong laser field.

I. INTRODUCTION

This project is an extension of the work published in [1], which is a short version of the preprint [2] that has not yet been completed. Some useful information can also be found in [3] and, even though we used the velocity of the dipole operator, in [4].

The basic idea is to study the dependence of the ejected-electron spectrum, in particular the first Autler-Townes [5] doublet on the details of the laser pulse. Some surprising results were found in [2], but more analysis is required in order to figure out what exactly is happening.

II. PROJECT STEPS

We will use the program `work15b-all.f` – or an extension thereof, to be developed by the students. The code uses two input files called `pulse.inp` and `tdse.inp`, respectively. For the description of `pulse.inp` see the subroutine `pulsekb`; some of the input for `tdse.inp` will be described below.

A. Development work

First, I would like you to practice with the 4-cycle pulse that we studied before. The good thing about this test case is that it will run so fast that you do *not* need to use the `-openmp` flag and can just run things interactively on the Drake cluster. There are ways to run interactively also on Stampede, but it's tricky and you need to be careful. If you ever do this, it is very important to ensure that all the environment variables are set properly – otherwise your job may fail with strange error messages.

As the first development project, I would like you to implement my Fourier program directly into the subroutine `pulsekb` and write the output to a file, so that we have it once and forever. That file should be kept for all the runs; just give it a reasonable name and set it up as an output file.

Secondly, I would like you to analyze the ejected electron spectrum. The first column of the file `betas.out` contains the energy (in atomic units) and the second column that

spectrum (dP/dE). Hopefully, the current output is properly normalized, meaning that

$$P_{ion} = \int_0^\infty \frac{dP}{dE}\, dE \tag{1}$$

is the total probability for ionization, obviously with $P_{ion} \leq 1$. You should write a simple integration routine that reads the file `betas.out` and works out this integral. You could use trapezoidal rule, Simpson's rule, or another fancy algorithm you may know of. Whatever you do, however, don't forget that you may need a point at $E = 0.0$. The program can't give you that, but you could extrapolate it using the lowest (few) energy point(s) that you have. Also, you need to go high enough in energy of the ejected electron that cutting off the integral at the highest energy is o.k., i.e., the contribution from the rest is negligible. It should be fine for the cases that I set up, and you can easily check by plotting it. (I would use gnuplot.)

As a check, you should also look at the output produced by subroutine `output`, which is called at the beginning, at the end, and also every `nprint` timesteps. You can specify (within reason) to obtain the overlaps with various hydrogenic states. Chances are that the discrete overlaps (you might want to modify/extend the output to make things more convenient) are converged by $n = 4$. So one would expect that the sum of the squares of these overlaps plus your integral over the ejected electron spectrum (at this point, you'll only have this at the every end) would give a result very close to 1.

Compilation hints: On the Drake cluster (cl2.bartschat.drake.edu) you say (`program.f` is your program):

```
ifort -o program -C -traceback program.f
```

The `-C` option will ensure (at least for a while) that you don't run into a simple array overflow) issue. It will (minorly) slow down the execution, but you can save yourself a lot of trouble by using the option until you are dead sure that your programs work.

On Stampede, you say

```
ifort -o program -openmp program.f
```

B. Sampling Runs

Once all your practice stuff works, we need to do a systematic overview study in order to see how things behave for the case of interest – a 40-cycle, relatively strong pulse. To

run jobs, you need to modify the `run15b.uni` script properly, changing directory names etc. **Always keep the file `wfn.out` in the appropriate directory!!!** This file can be used in a restart, which would save us a lot of time. Watch the disk space – remember that there is a home and a work directory.

We will change the following parameters:

- The frequency (energy) of the pulse (`ww1`): 0.375 a.u. (that is the excitation energy of the $n = 2$ states in H, i.e., we are on resonance) and 0.350 a.u., which is off-resonance and sufficiently far away from the resonance.

- The intensity and hence the amplitude of the electric field: $4.0 \times 10^{14}\,\mathrm{W/cm^2}$ (`ee1 = 1.0676d-1`), $1.0 \times 10^{14}\,\mathrm{W/cm^2}$, and $0.25 \times 10^{14}\,\mathrm{W/cm^2}$. Dropping the intensity by a factor of 4 means that the amplitude should be cut in half.

- The carrier envelope phase (`cep1`): We use 0° (the default) and 90° (this would have to read in via the appropriate `namelist`). For starters, a few spot checks should be run *only* for the highest intensity. If we find a significant dependence on that phase, we will investigate further.

- The pulse shape, in particular the way the pulse is switched on and off, as well as a possible plateau. Most of the runs should be performed with $\sin^2$ on and off (the default s-s below), and a plateau in the middle. Looking at the routine `pulsekb`, we want to run the following cases:

  2-36-2; 3-34-3; 4-32-4; 7-26-7; 10-20-10; 20-0-20

- For $4.0 \times 10^{14}\,\mathrm{W/cm^2}$ and 2-36-2 only, let's investigate the following on/off scenarios:

  s-s; s-t; s-g; t-s; t-t; t-g; g-s; g-t; g-g

The above set already requires quite a few jobs – 9 for the on/off scenarios times 3 intensities times 2 energies = 54 jobs; 6 for the on-plateau-off cases (always s-s) times 3 intensities times 2 energies = 36 jobs; plus a few spot checks for the carrier envelope phase. I am estimating 100 individual jobs total. With the current parameters, each job takes about 4.5 hours on Stampede, so – in principle – the entire job sequence would require to set up 10 jobs with a requested wall-time of 48 hours each, since each of these jobs could take care of 10 individual cases. But:

DO NOT DO THIS RIGHT WAY!!!

You need to practice first and gain some experience in using the machine, setting up and switching directories and input files without error, etc. Experience shows that this not easy – usually it's best to write scripts that set these things up for you automatically.

Depending on what comes out of this, we may want to repeat the analysis for a shorter pulse. I am currently thinking the following:

2-16-2; 3-14-3; 4-12-4; 7-6-7; 10-0-10

and adding an even higher intensity of $1.6 \times 10^{15}\,\mathrm{W/cm^2}$. These jobs should go faster (by a factor of 2), because the pulses are shorter, but I am not sure whether we can get results with the length form of the dipole operator for the very high intensity. It will help that the pulse is shorter, but I will have to check carefully.

III. SUMMARY

That's it for the moment. I will attach some sample files with obvious names and the code that I have so far. For the 4-cycle pulse, `betas-benchmark.out` is the benchmark to compare with. If you have questions, email or skype me.

---

[1] A. N. Grum-Grzhimailo, M. N. Khaerdinov, and K. Bartschat, Phys. Rev. A **88** (2013) 055401.

[2] A. N. Grum-Grzhimailo and K. Bartschat, unpublished (2013).

[3] A. N. Grum-Grzhimailo, A.D. Kondorskiy, and K. Bartschat, J. Phys. B **39** (2006) 4659

[4] A. N. Grum-Grzhimailo, B. Abeln, K. Bartschat, D. Weflen, and T. Urness, Phys. Rev. A **81** (2010) 043408

[5] S. H. Autler and C. H. Townes, Phys. Rev. **100** (1955) 703