Quantum-mechanical systems in traps and density functional theory

by

Jørgen Høgberget

THESIS
for the degree of
MASTER OF SCIENCE

(Master in Computational Physics)

Faculty of Mathematics and Natural Sciences
Department of Physics
University of Oslo

June 2013

Preface

blah blah

Contents

Chapter 1

Introduction

blah blah

Part I

Theory

Chapter 2

Scientific Programming

The introduction of the computer around 1945 had a major impact on the mathematical fields of science. Previously unsolvable problems were now easily solvable. The question was no longer whether or not it was possible, but rather to what precision and with which method. The computer spawned a new branch of physics, *computational physics*, breaching barriers no one could even imagine existed. The first major result of this synergy between science and computers came with the atomic bombs as a result of the Manhattan Project at the end of the second world war **citation needed**.

2.1 Programming languages

Writing a program, or a code, is a list of instructions for the computer. It is in many ways similar to writing human-to-human instructions. You may use different programming languages, such as C++, Python, Java, as long as the reader is able to translate it. The translator, called *compiler*, translates your program from e.g. C++ code into machine code. Other languages as Python are interpreted real-time and therefore require no compilation. Although the latter seems like a better solution, it comes at the price of efficiency, a key concept in programming.

As a rule of thumb, efficiency is inverse proportional to the complexity of the programming language. It is therefore natural to sort languages into different subgroups depending on where they are at the efficiency-complexity scale.

2.1.1 High-level languages

This subgroup of languages are often referred to as *scripting languages*. A script is a short code, often with a specific purpose such as analyzing output by e.g. generating tables and figures from raw data, or gluing together different programs which are meant to be run in sequential order.

For simple jobs as these, taking only seconds to run on modern computers, we do not need an optimized code, but rather an easily read, clutter-free code. The languages which prefer simplicity over efficiency are referred to as *High-level* [1]. Examples of high-level languages are Python, Ruby, Perl, Visual Basic and UNIX shells.

[1] There are different definitions of high-level vs. low-level. You have languages such as *assembly*, which is extremely complex and close to machine code, leaving all machine-independent languages as high-level ones. However, for the purpose of this thesis I will not go into assembly languages, and keep the distinction at a higher level.

Part II

Results

Chapter 3

Results

3.1 Validating the code

3.1.1 Calculation for non-interacting particles

I

Bibliography