Approximating Solutions of the time independent Schrödinger equation

Gian Laager
August 12, 2022

Maturaarbeit
Kantonsschule Glarus
Betreuer: Linus Romer
Referent: Beat Temperli

Dank

Mein ausserordentlicher Dank gilt Fritz Muster für das Korrekturlesen. Ebenso danke ich Heiri Müller und Otto Normalverbraucher für ihre hilfreichen Korrespondenzen mit mir.

Contents

1. Introduction

1.1. Schrödinger Equation

In 1926 Erwin Schrödinger changed our understanding of quantum physics with the Schrödinger equation. Based on the observations of Plank and Einstein that particles behave like waves he develop a wave equation which describes how the waves move and change in a given potential $V(x)$ or Hamiltonian H.

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x,t)\right]\Psi(x,t)$$

Or more general

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = H\Psi(x,t)$$

The time independent version that is going to be used later ignores the change over time and is much simpler to solve since it is ***only*** an ODE instead of a PDE.

$$E\psi(x) = H\psi(x)$$

or

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2}(x) + V(x)\psi(x) = E\psi(x)$$

Even with the time independent equation it is very difficult to get analytical solutions, because of this there are mainly two approaches to approximate solutions of $\psi(x)$, perturbation theory and WKB approximation. purturbation theory's goal is to give an analytical approximation which means it is very difficult to implement it for a computer. WKB on the other hand is much better since it just uses formulas, while it can still be used for analytical approximations.

1.2. Rust

Rust is one of the younger programming languages and attempts to replace C/C++ which are notoriously difficult to work with. It supports both functional and object oriented para dimes. It is much safer to work with while it promises the performance of C. One of the goals of Rust is fearless concurrency which means everybody should be able to write concurrent code with out dead locks and data races. This means calculations can utilize the full potential of the CPU without countless hours of debugging.

Personally I like programming languages that support functional para dimes when I'm programming something that uses a lot of math since functional programming languages are designed according to mathematical concepts that also govern the problem. This mostly results in rather neat solutions.

Rust as of the time of writing this document is not yet standardized meaning the code provided might no longer be correct with one of the newer Rust versions.

2. Methods

2.1. Approximation Scheme

There are mainly three approximation methods used to solve for the actual wave function itself. There is perturbation theory which breaks the problem down in to ever smaller sub-problems that then can be solved exactly. This can be achieved by adding something to the Hamiltonian operator $\hat{H}$ which can then be solved exactly. *Perturbation theory is inefficient compared to other approximation methods when calculated on a computer* (Van Mourik u.a., 2014, Introduction).

The other is Density functional field theory, it has evolved over the years and is used heavily in chemistry to calculate properties of molecules and is also applicable for the time dependent Schrödinger equation. It is something that might be interesting to add to the program in the future.

The program uses the third method WKB approximation, it is applicable to a wide verity of linear differential equations and works very well in the case of the Schrödinger equation. originally it was developed by Wentzel, Kramers and Brillouin in 1926. It gives an approximation to the eigenenfunctions of the Hamiltonian $\hat{H}$ in one dimension The approximation is best understood as applying to a fixed range of energies as $\hbar$ tends to zero (Hall, 2013, p. 305). This is not a physically correct explanation because one can not assume that $\hbar$ is small since it has units, but there is another interpretation that is physically more valid that will be discussed in the validity section (2.1.2).

2.1.1. Example

This example is from (Wkipedia, 2022, An example).

Lets solve the ordinary differential equation

$$\epsilon^2 \frac{d^2 y}{dx^2} = Q(x)y$$

where $Q(x)$ is an arbitrary function that is not $Q(x) = 0$ and ϵ is small. Note that this example relates to the Schrödinger equation where $\epsilon = -\frac{\hbar^2}{2m}$ and $Q(x) = E - V(x)$.

Replace y with

$$y(x) = exp(\frac{1}{\delta}\sum_{n=0}^{\infty} \delta^n S_n(x))$$

resulting in the equation

$$\epsilon^2(\frac{1}{\delta^2}(\sum_{n=0}^{\infty}\delta^n S_n'(x) + \frac{1}{\delta}(\sum_{n=0}^{\infty}\delta^n S_n''(x))) = Q(x) \tag{2.1}$$

As an approximation set the upper bound of the sum to be 2 instead of ∞, this then 2.1 can be written as

$$\frac{\epsilon^2}{\delta^2}S_0'^2 + \frac{2\epsilon^2}{\delta}S_0'S_1' + \frac{\epsilon^2}{\delta}S_0'' = Q(x)$$

This might seem worse since there now is a *non* linear differential equation, but as $\delta \to 0$ only the first term remains.

$$\frac{\epsilon^2}{\delta^2}S_0'^2(x) \sim Q(x)$$

This means that $\epsilon \propto \delta$. δ is a constant that can have an arbitrary value so lets set $\delta = \epsilon$. There for

$$S_0'^2 = Q(x)$$

which is the Eikonal equation, with solution

$$S_0(x) = \pm \int_{x_0}^{x} \sqrt{Q(t)}\, dt$$

and for S_1 we get

$$todo$$

2.1.2. Validity

todo

2.2. Newtons Method

Newton's method, also called the Newton-Raphson method, is a root-finding algorithm that uses the first few terms of the Taylor series of a function $f(x)$ in the vicinity of a suspected root (Weisstein, 2022). It makes a sequence of approximations of a root x_n that in sure tent cases converges to the exact value where

$$\lim_{n\to\infty} f(x_n) = 0$$

The sequence is defined as

$$x_0 = a$$
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Visually this looks like figure 2.1 $f(x) = (x-1)(x+1)(x-2)$. The blue line indi-

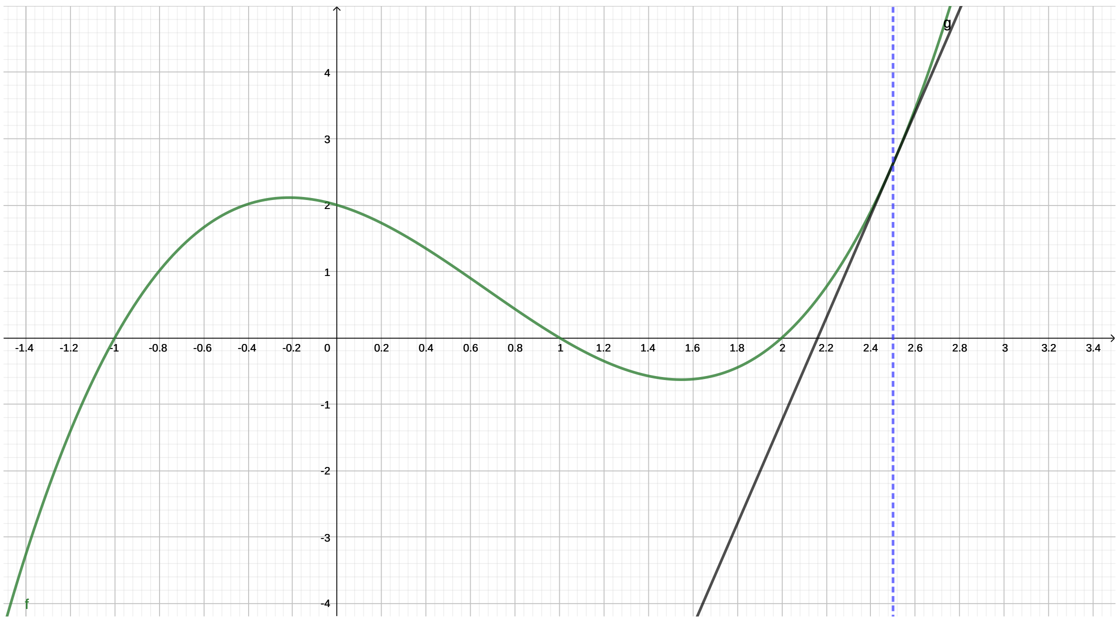

Figure 2.1.: Illustration of Newtons method, $f(x) = (x-1)(x+1)(x-2)$.

cates the initial guess which in this case is 2.5 the black line ($g(x)$) is a tangent to $f(x)$ at $(guess, f(guess))$ the next guess will be where the tangent intersects the x-Axis (solution of $g(x) = 0$). This will converge rather quickly compared to other methods such as Regula-Falsi.

```rust
pub fn newtons_method<F>(f: &F, mut guess: f64, precision: f64) -> f64
    where
        F: Fn(f64) -> f64,
{
    loop {
        let step = f(guess) / derivative(f, guess);
        if step.abs() < precision {
            return guess;
        } else {
            guess -= step;
        }
    }
}
```

In rust the sequence is implement with a function that takes a closure `f`, the initial guess `guess` and a stop condition `precision` the function will return if $|\frac{f(x_n)}{f'(x_n)}|$ is less then `precision`.
From the structure of the algorithm it is very tempting to implement it recursively, but by using a loop it is much faster since there are no unnecessary jumps and the precision can (at least in theory) be 0 without causing a stack overflow.

2.3. Derivatives

The precision gained by calculating derivatives would come with a massive overhead both in development and in performance, one would have to implement all the rules of differentiation and create some representation of an actual function to which those rules could be applied. I wrote an implementation of this in Go and not much accuracy can actually be gained.

A much easier approach is to approximate it. The definition of a derivative of a function $f(x)$ is

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$$

The limit can very easily be approximated with

$$\frac{f(x+\frac{\delta}{2})-f(x-\frac{\delta}{2})}{\delta}$$

were δ needs to be small yet not so small that it exceeds the 16 digit precision of double. I chose $\delta = \sqrt{2^{-52}}$, because $\epsilon = 2^{-52}$ is the smallest double precision floating point number where $1+\epsilon \neq 1$. The square root ensures that still enough precision remains. After some further research the function should be implement the same way as described by Boost (C++ library).

```rust
pub fn derivative<F, R>(f: &F, x: f64) -> R
    where
        F: Fn(f64) -> R,
        R: Sub<R, Output=R> + Div<f64, Output=R>,
{
    let epsilon = f64::epsilon().sqrt();
    (f(x + epsilon / 2.0) - f(x - epsilon / 2.0)) / epsilon
}
```

`f64::epsilon()` this is the difference between '1.0' and the next larger representable number (?). This function is implement not only for `f64` but for all types that support subtraction and division by `f64` this means it can also be used for closures with real inputs and complex outputs.

2.4. Integration

The same principles apply to integrals as to derivative it would not be a great benefit to implement an analytic integration system. Integrals would also be much more difficult to implement then derivatives since integrals can not be broken down in to many smaller integrals that can be computed easily instead it needs to be solved as is.

One approach would be to use the same method as with the derivative, take the definition with the limit and use a small value but this method can be improved in this case, since integrals calculate areas under curves a trapeze is more efficient and accurate.

A. Source Code

B. Detailed Calculations

Bildquellen

Wo nicht anders angegeben, sind die Bilder aus dieser Arbeit selbst erstellt worden. Das Titelbild stammt aus .

Bibliography

Hall, Brain C. (2013): *Quantum Theory for Mathematicians*. , Springer New York, NY, 1 Auflage.

Van Mourik, Tanja / Bühl, Michael / Gaigeot, Marie Pierre (2014): *Density functional theory across chemistry, physics and biology*
.

Weisstein, Eric W. (2022): *Newton's Method*
, [Online; accessed 10-August-2022].

Wkipedia (2022): *WKB approximation*
.

Selbständigkeitserklärung

Hiermit bestätige ich, Gian Laager, meine Maturaarbeit selbständig verfasst und alle Quellen angegeben zu haben.

Ich nehme zur Kenntnis, dass meine Arbeit zur Überprüfung der korrekten und vollständigen Angabe der Quellen mit Hilfe einer Software (Plagiaterkennungstool) geprüft wird. Zu meinem eigenen Schutz wird die Software auch dazu verwendet, später eingereichte Arbeiten mit meiner Arbeit elektronisch zu vergleichen und damit Abschriften und eine Verletzung meines Urheberrechts zu verhindern. Falls Verdacht besteht, dass mein Urheberrecht verletzt wurde, erkläre ich mich damit einverstanden, dass die Schulleitung meine Arbeit zu Prüfzwecken herausgibt.

Ort	Datum	Unterschrift