Appendix A

Numerical methods

A.1 The fast Fourier transform [58]

In quantum mechanics the Fourier transform of a (one dimensional) wave function $\psi(x)$ is defined as

$$\tilde{\psi}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} dx \; e^{-ikx} \psi(x) \tag{A.1}$$

and the inverse relation as

$$\psi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} dk \; e^{ikx} \tilde{\psi}(k) \quad . \tag{A.2}$$

One can think of $\tilde{\psi}(k)$ and $\psi(x)$ as begin the same function in two different representations, namely the momentum representation k and coordinate representation x. There are other representation in physics which are connected through a Fourier transform, *e.g.* the time domain and frequency domain. In the following we will explain the basic idea of the fast Fourier transform algorithm using a one dimensional example, but of course in principle this algorithm can be applied in any dimensionality. Let us consider the definition of the Fourier transform of a function $f(t)$ in the time domain. Its Fourier transform $F(\nu)$ in the frequency domain is then

$$F(\nu) = \int_{-\infty}^{+\infty} dt \; e^{-i2\pi\nu t} f(t). \tag{A.3}$$

A.1.1 The discrete Fourier transform

In numerical applications normally one has to deal with discretely sampled data instead of continuous functions. Let $f(t)$ be sampled at N discrete times separated by the time interval Δt

$$f_n = f(t_n) \quad t_n = n\ \Delta t \quad n = 0, 1, 2, ...N-1 \quad . \tag{A.4}$$

Let us also assume that N is even (which is not strictly necessary but most common for a practical application of the FFT). For any finite time interval Δt there is then a so called Nyquist critical frequency ν_{max}, which is the maximal frequency which can be resolved with the chosen discrete data points. ν_{max} is given by the sampling theorem as

$$\nu_{max} = \frac{1}{2\Delta t} \quad . \tag{A.5}$$

If we now want to estimate the Fourier transform of $f(t)$ by using the N sampled points, we can of course again only get N discrete values in the frequency domain. From Eq. (A.5) we know that these frequencies ν_n have to be $-\frac{1}{2\Delta t} \leq \nu_n \leq \frac{1}{2\Delta t}$. Therefore it is convenient to evaluate $F(\nu)$ only at the following discrete values

$$F_j = F(\nu_j) \quad ; \quad \nu_j = \frac{j}{N\Delta t} \quad ; \quad j = -\frac{N}{2}, ..., \frac{N}{2} \quad . \tag{A.6}$$

We can now estimate Eq. (A.3) by a discrete sum:

$$\begin{aligned} F_j = F(\nu_j) &= \int_{-\infty}^{+\infty} dt\ e^{-i2\pi\nu_j t} f(t) \\ &\approx \sum_{n=0}^{N-1} \Delta t\ e^{-j2\pi\nu_j t_n} f(t) \\ F_j &= \Delta t \sum_{n=0}^{N-1} e^{-i2\pi j\ n/N} f_n \quad . \end{aligned} \tag{A.7}$$

Eq. (A.7) is the so called discrete Fourier transform. Note that Eq. (A.7) is periodic in n with a period of N, so that from the $N+1$ values F_i defined in Eq. (A.6) only N are independent, *i.e.*, $F_{-\frac{N}{2}}$ is equal to $F_{\frac{N}{2}}$.

A.1.2 The fast Fourier transform algorithm

By looking at Eq. (A.7) one can see that the direct evaluation of this formula can be written as a matrix vector multiplication

$$\begin{pmatrix} F_{j_0} \\ \vdots \\ F_{j_{N-1}} \end{pmatrix} = \begin{pmatrix} e^{aj_0\ n_0} & \dots & e^{aj_0\ n_{N-1}} \\ \vdots & \ddots & \vdots \\ e^{aj_{N-1}\ n_0} & \dots & e^{aj_{N-1}\ n_{N-1}} \end{pmatrix} \begin{pmatrix} f_{n_0} \\ \vdots \\ f_{n_{N-1}} \end{pmatrix} \tag{A.8}$$

with $a = -i2\pi/N$. This means one would need $N \times N$ multiplications of complex numbers for each transformation (assuming that one computes the matrix elements only one time). This can be reduced to $N\log_2 N$ multiplications by using the Fast Fourier Transform algorithm. The basic idea of the FFT was first given in the Danielson-Lanczos Lemma, which states that a discrete Fourier transform of length N can be rewritten as the sum of two discrete Fourier transforms, each of length $N/2$. One of them containing the even-numbered points e and the other the odd-numbered o. The proof is as follows:

$$\begin{aligned} F_j &= \sum_{n=0}^{N-1} e^{-i2\pi j\ n/N} f_n \\ &= \sum_{n=0}^{N/2-1} e^{-i2\pi j\ 2n/N} f_{2n} + \sum_{n=0}^{N/2-1} e^{-i2\pi j\ (2n+1)/N} f_{2n+1} \\ &= \sum_{n=0}^{N/2-1} e^{-i2\pi j\ n/(N/2)} f_{2n} + e^{-i2\pi j/N} \sum_{n=0}^{N/2-1} e^{-i2\pi j\ n/(N/2)} f_{2n+1} \\ &= F_j^e + e^{-i2\pi j/N} F_j^o \end{aligned} \tag{A.9}$$

We can now apply the Danielson-Lanczos Lemma again to split these two discrete Fourier transforms into four each of the length $N/4 \rightarrow F_j^{ee}, F_j^{eo}...$, and again and again, until we reach the point, where we have N discrete Fourier transforms of the length 1. Such a discrete Fourier transform is just the copying of an input number into an output number:

$$F_j^{eeoeeoo...eeooe} = f_n \quad . \tag{A.10}$$

So for each F_j a reordering of the input has to be done and then the transforms of length 2,4,8,...N have to be calculated by multiplying with the right factors like $e^{-i2\pi j/N}$ in Eq. (A.9). This obviously needs to be done $\log_2 N$ times, resulting in a total numerical effort of $N\log_2 N$.

A.2 The split operator technique

The solution of the time-dependent (for simplicity one dimensional) Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}\psi(x,t) = \hat{H}\psi(x,t) \tag{A.11}$$

is given for a time-independent Hamilton operator $\hat{H} = \hat{T} + \hat{V}$ by the time evolution operator $\hat{U}(t,t_o)$ as

$$\hat{U}(t,t_o) = e^{-i\hat{H}\ (t-t_o)/\hbar} \quad . \tag{A.12}$$

The split operator propagator from Feit-Fleck [50] now splits the exponential function of $\hat{H}$ (although $[\hat{T},\hat{V}] \neq 0$) into exponential functions of $\hat{T}$ (the kinetic energy operator) and $\hat{V}$ (the potential energy operator). If this is done as

$$e^{-i\hat{H}\Delta t/\hbar} = e^{-i\hat{T}\Delta t/(2\hbar)} \cdot e^{-i\hat{V}\Delta t/\hbar)} \cdot e^{-i\hat{T}\Delta t/(2\hbar)} + \mathcal{O}(\Delta t^3) \tag{A.13}$$

for a finite time step Δt, it can be shown that the error introduced scales as the cube of the time step length. If one now uses Dirac δ functions located at equally spaced grid points x_i to expand $\psi(x,t)$ the potential energy operator is diagonal. The kinetic energy operator is diagonal in momentum representation, which is connected through a Fourier transform to the coordinate representation (see section A.1). This means that $e^{-i\hat{T}\Delta t/(2\hbar)}$ can be easily applied in momentum representation and $e^{-i\hat{V}\Delta t/\hbar}$ in coordinate representation, respectively. One needs one FFT and one inverse FFT for each propagation step. Eq. (A.13) can also be used for time-dependent potentials $V(t)$, then one needs to reevaluate the potential energy operator after each time step.

A.3 The Euler integration method [58]

The simplest way to solve an initial value problem for a first order differential equation like

$$\frac{dy}{dx} = z(x, y) \tag{A.14}$$

is the so called Euler method (note that any ordinary differential equation can be rewritten as a set of coupled first order differential equations). In an initial value problem we start from an initial value y_i and want to calculate the final value y_f. In the Euler method one now rewrites dx as a finite step Δx and multiplies Eq. (A.14) by Δx. Then one gets an approximation for the change in y_n when the independent variable x is increased by one step Δx, *i.e.,* from x_n to $x_{n+1} = x_n + \Delta x$. Now one can subsequently calculate the values of y_{n+1} starting from y_i until one reaches y_f as follows:

$$y_{n+1} = y_n + z(x_n, y_n)\Delta x \quad . \tag{A.15}$$

A.4 The Runge Kutta 4th order method [58]

The basic idea of the Runge Kutta method is the same as for the Euler method Eq. (A.15). In the Euler method the step's error is only one power smaller than the correction [$\mathcal{O}(\Delta x^2)$]. The Runge Kutta method uses not only one step but one or several trial steps within the interval to determine $z(x, y)$ for different points in the interval Δx. Adding up the right combinations of these trial steps can eliminate the higher order error terms. In the forth-order Runge Kutta formula one gets $\mathcal{O}(\Delta x^5)$ by applying the following

algorithm:

$$\begin{aligned} y_{s1} &= z\left(x_n, y_n\right)\Delta x \\ y_{s2} &= z\left(x_n + \frac{\Delta x}{2}, y_n + \frac{y_{s1}}{2}\right)\Delta x \\ y_{s3} &= z\left(x_n + \frac{\Delta x}{2}, y_n + \frac{y_{s2}}{2}\right)\Delta x \\ y_{s4} &= z\left(x_n + \Delta x, y_n + y_{s3}\right)\Delta x \\ y_{n+1} &= y_n + \frac{y_{s1}}{6} + \frac{y_{s2}}{3} + \frac{y_{s3}}{3} + \frac{y_{s4}}{6} \quad . \end{aligned}$$