# Computational Methods in Particle Physics

Reinhard Alkofer

Lecture notes Spring Term 2021

Institute of Physics
University of Graz / NAWI Graz

These notes are based on the manuscript “Computational methods of elementary particle physics” by Markus Huber and on his course of the same name at the University of Graz in the spring term 2017 as well as my respective course in the spring term 2019 and Madeleine Adrien’s respective course in the spring term 2020.

Version: May 26, 2021
(The notes are updated according to the progress of the course.)

# Contents

# 1 Introduction

## 1.1 Computational methods in particle physics: A brief overview

Experimental and theoretical elementary particle physics have been and still are a major driving force in the development of computational methods. Early examples include the development of algorithms for the task of numerical calculations, the first computer algebra systems, efficient automated trigger systems for detectors, the analysis of huge data samples, and last but not least the distribution of data with the internet protocol. As much as particle physics contributed to hardware, software and algorithm development on the one hand, contemporary particle physics were impossible on the other hand without the use of powerful computers.

Therefore, given the plethora of computer-based methods in particle physics a disclaimer with respect to the title of this course is in order: This course will be about one specific example within a specific approach to Quantum Chromo Dynamics (QCD), the theory of the Strong Interactions. Phrased otherwise, within the huge research area called particle physics we will explore a tiny island in this course.

Before going into detail about the topic of this course I will provide a (necessarily incomplete) overview about the most important uses of computers in the research area of particle physics. Computational methods in elementary particle physics include:

- Computer-aided design and operation of particle sources, accelerators and detectors, *cf.* the lecture by Robert Schöfbeck last term, is a necessity due to the complexity of these machines.[1]
- Monte Carlo event generators: High-energy particle collisions as *e.g.* at the LHC produce up to several thousands of particles. Almost all of the primary particles are unstable resonances which decay still in the detector and thus produce even more particles. Therefore, it is not possible to describe such collisions of particles (semi-)analytically even in cases the elementary reactions can be treated in perturbation theory (which is by far not always the case). In order to confront experimental results with theoretical predictions and, even more importantly, to separate searched signals from already known background, simulations of these so-called particle showers are performed. The quantum nature of single interactions, and thus the resulting probability distributions in momenta, spin, etc., are taken into account via (pseudo-)random processes. (NB: Hereby, even modern event generators employ Markov chains, *i.e.*, neglect the history-dependence of processes. This is in general justified by the type of processes considered in collider high-energy physics but it is not the generic case. A typical counterexample is particle production in ultra-strong

[1] As I am a theorist I will not say more on this issue.

fields.) By construction event generators produce large numbers of possible outcomes, and their average values and standard deviations are then compared to the measurements.

- Data analysis: Experiments as those carried out at the LHC produce immense amounts of data, approximately 10-15 PB per year for the LHC. The expected data rate for the High-luminosity LHC is one to two orders of magnitude larger. First, processing and storing such amounts of data requires large computational resources. In the case of the LHC this is handled via a worldwide network of computing centers, the so-called Worldwide LHC Computing Grid. It handles the distribution of the data and provides distributed computing resources. Second, machine learning, mostly in the form of deep reinforcement learning by deep neural networks, is used to identify particle tracks and, in a next step, to classify the data accordingly to the likelihood of containing a particle and/or reaction of interest. Third, modern statistical tools (keyword Bayesian methods) allow then for conclusions whether a discovery has been made or not.

- Perturbation theory: In many areas of elementary particle physics perturbation theory is applied well beyond the leading and next-to-leading (NL) order. At the NNNL ($N^4L$) the task at hand consists of evaluating thousands (millions) of Feynman integrals. Modern techniques in this direction are building on computer algebra systems which automatically generate the corresponding integrals and relate them to several dozen master integrals.

- Lattice quantum field theory: One possible, and by now the most wide-spread formulation, of a quantum field theory is based on functional integrals, the multidimensional generalisations of Feynman's quantum mechanical path integral. Thereby, a high-dimensional integral over the values of the fields at every space-time point needs to be performed. To numerically estimate this integral one approximates space-time by a, typically hypercubic, lattice with lattice spacing $a$ and length $L$ in every of the four directions. Hereby, the continuum limit, in which translational and Lorentz invariance is restored, corresponds to $a \to 0$ and $L \to \infty$. One typically tries to capture this limit by performing calculations for several different lattices and extrapolation of results. One obvious obstacle is the dimension of the integral: On $64^4$ lattice QCD possesses half a million degrees of freedom, and therefore one would have to integrate over a huge number of variables. Hereby, Monte-Carlo integration techniques (*cf.* the lecture by D. Sexty last term) are used to arrive at a stochastic estimate of the corresponding integrals. Lattice calculations in Statistical Physics and Quantum Field Theory belong to the most resource-intensive calculations in science but their predictive power makes them an extremely useful tool, for more details see a corresponding lecture, *e.g.* the one by A. Maas in the last year.

- Functional methods: This kind approach is based on an approximate evaluation of correlation functions, see the lecture by A. Tripolt in the last term. The corresponding numerical task consists typically in the solution of integral and/or integro-differential equations, sometimes supplemented with the numerical evaluation of the spectrum of an operator. Compared to other methods functional approaches require a different kind of computer resources than *e.g.* lattice field theory. Although being less CPU-time intensive most of these methods were not possible without computer algebra systems. In addition, modern applications tend to rest on the availability of memory (RAM) in the multi-GB range.

Within this course we will consider an example based on equations developed by F. Dyson, J. Schwinger, H. Bethe, E. Salpeter and many others for an important by nevertheless relatively simple system. To this end we will solve, admittedly based on some drastic approximations, for the self-energy of quarks, thereby demonstrating the dynamical generation of mass, and for the bound-state amplitude of the lightest hadron, the pion, for more details see the next subsection.

In particle physics, depending on the problem, various architectures are used for computations. For some problems desktop PCs are sufficient. However, even then parallelisation is advantageous due to the increasing number of cores in modern CPUs, *e.g.*, the Intel Skylake Platinum possesses 24 cores. High-performance compute clusters naturally take advantage of parallelised programs. Clusters can have a few dozen nodes, with several CPUs each, up to several million nodes for modern supercomputers. In some cases it is even useful to run on clusters several single-core instances of the same program, *e.g.*, for Monte Carlo methods when the same program is run over and over again to accumulate statistics. Possibilities for parallelisation are the program packages MPI or OpenMP.

In the last decade graphical processing units (GPUs) have become also very important. Programs are written with CUDA or OpenCL. An example for a hybrid-type architecture is Intel's Xeon Phi coprocessor, which offers a highly-parallel multicore environment similar to a GPU.

Various programming languages are used in elementary particles physics. Historically, FORTRAN (short for FORmula TRANSlation) has been always strong and, despite constant rumours to the contrary, it is still widely used. One reason is that many libraries, *e.g.*, LAPACK, used in physics are written in FORTRAN. The fact that FORTRAN was heavily used in physics, or in science in general, is reflected in many 'physics friendly' features, e.g., a complex number type was included early on. Another often used language is C++, an object-oriented and thus very flexible language. For GPU and coprocessor architectures languages like CUDA and OpenCL are required. A modern open-source programming language with free access to compilers for many computer architectures is Julia.

On the symbolic side Mathematica must be mentioned, which was founded by Stephen Wolfram, a particle physicist by training, in 1987. Special purpose packages can add additional functionalities useful for particle physicists. Special mention deserves the program FORM, originally started by Vermaseren (of course, a particle physicist). Its focus is on use in particle physics, where it is extremely fast when it comes to perform large symbolic calculations.

To complete the list, let me mention that for larger and typically cooperative projects additional auxiliary tools exist that can be very useful. This includes *make* (for the compilation of the code), IDEs (integrated development environments), version control systems (*git* can be considered state-of-the-art, but also *subversion* or others offer the basic features), debuggers, memory error detection programs and profilers.

## 1.2 Outline of this course

In this course the methods necessary to calculate a meson mass in QCD will be developed, and as already stated the chosen example is the pion as lightest of all hadrons. This will be done by solving (within this course a simplified version of) the respective bound state equation (BSE), a so-called Bethe-Salpeter equation. This equation needs several other quantities as input. Some of them will be taken from models, another one, the quark propagator, will be calculated in this course from an integral equation called Dyson-Schwinger equation (DSE).

On the technical side this will require numerical integration, function interpolation, implementing iterative methods, and solving an eigenvalue system. The complete code for this project will be developed step by step and should result in a final program.

The following tasks need to be implemented which will be done in individual projects:

1. Functions for numerical integration
2. Functions for interpolation
3. Implementation of the quark propagator
4. Implementation of the BSE
5. Solution of the BSE as an eigenvalue system

Programmes suitable for numeric computation should be used, *e.g.*, FORTRAN, C, C++, Python, and/or Julia.

Tasks should be performed in small groups of two to four, preferentially three, students working closely together.

The exam consists of handing on a documentation by every group and an individual oral exams for every student.

# 2 Basic techniques

## 2.1 Numerical integration

With respect to the numerical evaluation of an integral one distinguishes two cases: The integrand is known analytically but the integral cannot be solved analytically in closed and/or usable form, or the integrand itself is only known numerically. Depending on the problem, one can choose from many methods for numerical integration. Alternatively one speaks of numerical *quadrature*, which can refer to one- or multi-dimensional integration, whereas *cubature* is used for multi-dimensional integration only.

In this section we will first look at some simple integration formulas to explain the basic idea and get some insight about the related errors. Then we will turn towards integration techniques which are actually used in this course.

### 2.1.1 Newton-Cotes formula

A simple integration method is the *Newton-Cotes formula.* To calculate an integral $I$ one replaces the integrand $f(x)$ by an approximating function $p_m(x)$, typically an $m$-th order polynomial:

$$I = \int_a^b dx\, f(x) \approx \int_a^b dx\, p_m(x). \tag{2.1}$$

The approximating function $p_m(x)$ is chosen such that it agrees with $f(x)$ at a fixed number of points. For a polynomial this determines the values for the coefficients $a_i$:

$$p_m(x) = a_0 + a_1 x + \ldots + a_{m-1}x^{m-1} + a_m x^m. \tag{2.2}$$

For practical calculations one splits the integration interval $[a, b]$ into smaller intervals, herein first with equidistant width

$$h = \Delta x = \frac{b-a}{n} \tag{2.3}$$

and defines accordingly the points

$$x_i = a + i\,h, \quad i = 0, \ldots, n. \tag{2.4}$$

The simplest approximation is to use a constant $p_m(x)$. This is called the *rectangular rule.* It is rarely used, because its truncation error is quite large for general functions. We can approximate the function $f(x)$ in the interval $x_i \leq x \leq x_{i+1}$ either by $f_i = f(x_i)$ or $f_{i+1} = f(x_{i+1})$. The value of the integral in the interval $[x_i, x_{i+1}]$ is then $h\,f_i$ and $h\,f_{i+1}$, respectively. The total integral can be calculated correspondingly as

$$I_{\text{left}} \approx h \sum_{i=0}^{n-1} f_i \tag{2.5}$$

or

$$I_{\text{right}} \approx h \sum_{i=0}^{n-1} f_{i+1} = h \sum_{i=1}^{n} f_i. \tag{2.6}$$

These formulas over- or underestimate integrals of functions that increase or decrease monotonically. This behaviour can be improved using the average of the two forms instead:

$$I_r \approx h \sum_{i=0}^{n-1} \frac{f_i + f_{i+1}}{2}. \tag{2.7}$$

As we will see, this is equivalent to the trapezoidal rule.

The *trapezoidal rule* approximates the integrand piecewise by a polynomial of order one: $p_1(x) = a_0 + a_1\, x$. The area under the curve in the interval $[x_i, x_{i+1}]$ is then given by the area of a trapezoid: $h\,(f_i + f_{i+1})/2$. Summing up all intervals yields

$$I_t \approx h \sum_{i=0}^{n-1} \frac{f_i + f_{i+1}}{2} = \frac{h}{2}(f_0 + 2f_1 + \ldots + 2f_{n-1} + f_n). \tag{2.8}$$

The error for a single segment is with this method given by

$$E_{1t} = \int_a^b dx\, f(x) - \frac{f(a) + f(b)}{2}(b - a). \tag{2.9}$$

Using a Taylor expansion around $\bar{x} = (b + a)/2$ with $y = x - \bar{x}$,

$$f(x) = f(\bar{x}) + yf'(\bar{x}) + \frac{y^2}{2!} f''(\bar{x}) + \ldots, \tag{2.10}$$

we can perform the integral:

$$\begin{aligned} \int_a^b dx\, f(x) &= \int_{-\frac{h}{2}}^{\frac{h}{2}} dy \left( f(\bar{x}) + yf'(\bar{x}) + \frac{y^2}{2!} f''(\bar{x}) + \ldots \right) = \\ &= h\, f(\bar{x}) + \frac{1}{24} h^3 f''(\bar{x}) + \ldots . \end{aligned} \tag{2.11}$$

We use the Taylor expansion also at the endpoints of the interval:

$$\begin{aligned} f(a) &= f(\bar{x}) - \frac{h}{2} f'(\bar{x}) + \frac{1}{2}\left(\frac{h}{2}\right)^2 f''(\bar{x}) - \ldots, \\ f(b) &= f(\bar{x}) + \frac{h}{2} f'(\bar{x}) + \frac{1}{2}\left(\frac{h}{2}\right)^2 f''(\bar{x}) + \ldots . \end{aligned} \tag{2.12}$$

Plugging all this into Eq. (2.9) we obtain

$$E_{1t} = \left( h\, f(\bar{x}) + \frac{1}{24} h^3 f''(\bar{x}) + \ldots \right) - \left( h\, f(\bar{x}) + \frac{1}{8} h^3 f''(\bar{x}) + \ldots \right) \approx -\frac{1}{12} h^3 f''(\bar{x}). \tag{2.13}$$

Thus the error is proportional to $f''(\bar{x})$ and $h^3$. When a multisegment trapezoidal rule is used, the individual error estimates need to be summed for obtaining an estimate for the total error:

$$E_t \approx -\frac{1}{12}\left(\frac{b-a}{n}\right)^3 \sum_{i=0}^{n-1} f''(\bar{x}_i), \tag{2.14}$$

where $\bar{x}_i$ is the midpoint between $x_i$ and $x_{i+1}$. Using an average value for the second derivative,

$$\overline{f''} = \frac{1}{n}\sum_{i=0}^{n-1} f''(\bar{x}_i), \tag{2.15}$$

this can be rewritten as

$$E_t \approx -\frac{1}{12}(b-a)\left(\frac{b-a}{n}\right)^2 \overline{f''} = -\frac{1}{12}(b-a)h^2\overline{f''}. \tag{2.16}$$

Thus, the error is $O(h^2)$, since $b-a$ is fixed. Repeating a similar analysis for the rectangular rules shows that the error is $O(h)$:

$$E_r \approx -\frac{1}{2}(b-a)h\overline{f'}. \tag{2.17}$$

*Voluntary exercise:* Derive eq. Eq. (2.17).

To improve the precision of the results, the straightforward approach is to lower the step size $h$. However, the usefulness of this approach is limited by round-off errors becoming too large at some points. Another method is to increase the degree of the employed polynomial $m$. For $m = 2$ this leads to *Simpson's one-third rule*,

$$\begin{aligned}&\int_a^b dx f(x) \approx \\ &(\frac{1}{3}f(x_0) + \frac{4}{3}f(x_1) + \frac{2}{3}f(x_2) + \frac{4}{3}f(x_3) + \ldots + \frac{2}{3}f(x_{n-2}) + \frac{4}{3}f(x_{n-1}) + \frac{1}{3}f(x_n))h \\ &+\mathcal{O}(1/n^4).\end{aligned} \tag{2.18}$$

for $m = 3$ to *Simpson's three-eight rule* and for $m = 4$ to *Boole's rule.* Especially, Simpson's one-third rule can be considered as a very efficient method in case of an integrand without any complications like variations over several orders of magnitude and/or fast oscillations. Assuming an odd value for $n$ a pseudo-code for this rule is:

```
FUNCTION SIMPS(N,H,A)
DIMENSION A(2049)
S=A(1)+A(N)
T=0.D0
DO I=2,N-1,2
T=T+A(I)
ENDDO
S=S+4.D0*T
T=0.D0
DO I=3,N-3,2
T=T+A(I)
ENDDO
SIMPS=H*(S+2.D0*T)/3.D0
```

It several cases it might prove to be useful to choose $n = 2^m + 1$ with $m$ integer.

An alternative possibility is to reduce the error by combining the results from calculations with different step sizes (*Richardson's extrapolation*). For example, consider the errors obtained for two step sizes $h_1$ and $h_2$ with the trapezoidal method:

$$E(h_1) \approx -\frac{1}{12}(b-a)h_1^2\overline{f''}, \tag{2.19}$$

$$E(h_2) \approx -\frac{1}{12}(b-a)h_2^2\overline{f''}. \tag{2.20}$$

Assuming that $\overline{f''}$ is independent of the step size the true value for the integral $I$ can then be obtained from each calculation as

$$I \approx I(h_1) + c\,h_1^2, \tag{2.21}$$

$$I \approx I(h_2) + c\,h_2^2, \tag{2.22}$$

where $I(h)$ is the result obtained by using the step size $h$. Equating these two equations we can calculate the constant $c$:

$$c = -\frac{1}{12}(b-a)\overline{f''} \approx \frac{I(h_2)-I(h_1)}{h_1^2-h_2^2}. \tag{2.23}$$

Plugging this into Eq. (2.22) yields

$$I \approx I(h_2) + \frac{I(h_2)-I(h_1)}{(h_1/h_2)^2-1}. \tag{2.24}$$

One can show that the respective error is $O(h^4)$. A particularly useful choice which allows to re-use function evaluations is $h_2 = h_1/2$.

This process can be used to combine results from calculations with different step sizes. For example, if we have the results $I(h_1)$, $I(h_2)$ and $I(h_3)$, we can combine $I(h_1)$ and $I(h_2)$ to obtain a result $I(h_1, h_2)$ with an error $O(h^4)$, but we can also combine $I(h_2)$ and $I(h_3)$ to obtain a result $I(h_2, h_3)$. Combining the two new results, we can get a result with an error $O(h^6)$. Systematically applying this procedure is known as *Romberg integration.*

### 2.1.2 Gauss quadrature

The Newton-Cotes formula calculates an integral from values at equidistant points. Gauss quadrature does not only use the freedom to fix the weights $w_i$, but also the positions $x_i$ where the function needs to be known. Consequently, Gauss quadrature requires knowledge of the integrand at not equidistant given points. The general formula is

$$\int_{c_1}^{c_2} dx\, g(x)f(x) \approx \sum_{i=1}^{n} w_i f(x_i). \tag{2.25}$$

Due to this enlarged freedom, fewer function evaluations are required compared to the Newton-Cotes method. For a given value of $n$, the formula is exact for polynomials of degree $2n-1$. Note that the function $g(x)$ appears only on the left-hand side. It allows to make the quadrature exact for polynomials up to order $2n-1$ times the function $g(x)$. The values of the positions $x_i$ and of the weights

$w_i$ depend on the choice of $g(x)$. To exemplify the method we will choose $g(x) = 1$. This is called *Gauss-Legendre quadrature*, which is based on Legendre polynomials.

The use of orthonormal functions makes Gauss quadrature especially efficient. This implies that the choice of $g(x)$ also determines the values for the boundaries, $c_1$ and $c_2$. For Gauss-Legendre they are set to $c_1 = -1$ and $c_2 = 1$. Hence, an integral with different boundaries must be transformed to this interval. In principle any transformation from an interval $z \in [a, b]$ is possible that transforms this interval to $x \in [-1, 1]$. For example, the *linear transformation* reads

$$x = \frac{2z - a - b}{b - a}, \tag{2.26}$$

where $z$ is the original variable. The inverse transformation is

$$z = \frac{(b - a)x + a + b}{2}. \tag{2.27}$$

When such a transformation of variables is performed, the Jacobian of the transformation must also be included, and thus for the linear transformation one has

$$dz = \frac{b - a}{2} dx. \tag{2.28}$$

Thus, a general integral is calculated as

$$\int_a^b dz f(z) \approx \frac{b - a}{2} \sum_i^n w_i f(\zeta_i)\,, \quad \zeta_i = z(x_i). \tag{2.29}$$

The Jacobian can be taken into account directly in the weights.

For integrands for which the integration variable stretch over several orders of magnitude a *logarithmic transformation* is advantageous:

$$\begin{aligned} x &= A + B \ln z, \\ z &= e^{(x-A)/B}. \end{aligned} \tag{2.30}$$

The coefficients $A$ and $B$ are determined by the conditions $x(a) = -1$ and $x(b) = 1$ as

$$A = \frac{-\ln(a\,b)}{\ln(b/a)}, \quad B = \frac{2}{\ln(b/a)}. \tag{2.31}$$

The Jacobian is

$$dz = \frac{1}{B} e^{(x-A)/B} dx. \tag{2.32}$$

Practical hint: Never put the transformation into the function evaluation. From the points $x_i$ one best calculates the values $\zeta_i = z(x_i)$ and then evaluate the function to obtain the $f_i = f(\zeta_i)$.

To elucidate the method we consider the two-point formula, viz., the result is exact for polynomials of order 3 and below:

$$\int_{-1}^1 dx f(x) = w_1 f(x_1) + w_2 f(x_2). \tag{2.33}$$

From the requirement that it should be exact for $f(x)$ equal to 1, $x$, $x^2$ and $x^3$ we obtain

$$\begin{aligned}\int_{-1}^{1} dx &= 2 = w_1 + w_2,\\ \int_{-1}^{1} dx\, x &= 0 = w_1\, x_1 + w_2\, x_2,\\ \int_{-1}^{1} dx\, x^2 &= \frac{2}{3} = w_1\, x_1^2 + w_2\, x_2^2\\ \int_{-1}^{1} dx\, x^3 &= 0 = w_1\, x_1^3 + w_2\, x_2^3. \end{aligned} \tag{2.34}$$

These four conditions leads to

$$w_1 = w_2 = 1, \quad x_1 = -x_2 = \frac{1}{\sqrt{3}}. \tag{2.35}$$

Note the symmetry $x_1 = -x_2$. As Simpson's three-eight rule, Eq. (2.33) is exact for polynomials up to order three. However, here less evaluations of the function are required (2 vs. 4).

For higher $n$ the determination of the $x_i$ and $w_i$ is done using Legendre polynomials. They are given by

$$\begin{aligned}P_0(x) &= 1,\\ P_1(x) &= x,\\ P_n(x) &= \frac{2n-1}{n} x\, P_{n-1}(x) - \frac{n-1}{n} P_{n-2}(x), \quad n = 2, 3, \ldots\end{aligned} \tag{2.36}$$

and fulfill the orthogonality relation

$$\int_{-1}^{1} dx P_n(x) P_m(x) = \delta_{nm} \frac{2}{2n+1}. \tag{2.37}$$

The weights $w_i$ are

$$w_i = \frac{2}{(1-x_i^2) P_n'(x_i)^2} = \frac{2(1-x_i^2)}{(n+1)^2 P_{n+1}(x_i)^2}. \tag{2.38}$$

The derivative can be calculated from the recurrence relation

$$(1-x^2) P_n'(x) = (n+1) x P_n(x) - (n+1) P_{n+1}(x). \tag{2.39}$$

From our elementary example it is evident that the values for $x_i$ need to be the zeros of the Legendre polynomial of order $n$: $P_n(x_i) = 0$, and, according to the theory of orthogonal functions, there are exactly $n$ zeros (nodes): $i = 1, \ldots n$. For finding the zeros, the standard Newton method is sufficient which amounts in iterating

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})} \tag{2.40}$$

until $|x_i - x_{i-1}| < \epsilon$. $\epsilon$ must be chosen sufficiently small, *e.g.*, $10^{-12}$. By using a good initial guess $x_0$, this method converges in a few steps. The Legendre polynomials have the advantage that a good initial guess is available

$$x_{n,j}^{(0-guess)} = \cos(\pi(j-0.25)/(n+0.5))$$

where $n$ is the order of the polynomial and $j$ labels the respective nodes.

In principle, tabulated values for the weights and zeros can be used. However, this is not recommended. In particular, more flexibility is achieved when they are calculated in the program, and in most cases of practical relevance the calculation is even faster than data input.

The Gauss-Legendre quadrature has an error estimate of

$$E_{GL} \approx \frac{2^{2n+1}(n!)^4}{(2n+1)((2n)!)^3} f^{(2n)}(\xi), \quad -1 < \xi < 1. \tag{2.41}$$

From this it can be seen that polynomials of order up to $2n-1$ are evaluated exactly, since the derivative $f^{(2n)}$ vanishes in that case. Also, convergence is much faster than for Newton-Cotes.

It is possible to construct the weights and zeros for a generic function $g(x)$ by constructing the respective set of orthogonal polynomials multiplying the weight function $g(x)$ to generate orthogonal functions under the $L^2$-norm (Gegenbauer construction). Some choices result in well-known orthogonal polynomials. The most useful cases are:

| $g(x)$ | Polynomials | Boundaries |
|---|---|---|
| $1$ | Legendre | $[-1,1]$ |
| $1/\sqrt{1-x^2}$ | Chebyshev first kind | $[-1,1]$ |
| $\sqrt{1-x^2}$ | Chebyshev second kind | $[-1,1]$ |
| $e^{-x}$ | Laguerre | $[0,\infty]$ |
| $e^{-x^2}$ | Hermite | $[-\infty,\infty]$ |

Hereby, the use of Chebyshev polynomials provides practical advantages. In addition, a typical application for Gauss-Chebyshev quadrature is the integration over angles. For example, hyperspherical coordinates in four dimension result in a weight $(\sin(\theta))^2$ in the integration over an angle $\theta$. With $u = \cos(\theta)$ the respective integral can be rewritten as

$$\int_0^\pi d\theta \, (\sin(\theta))^2 = \int_{-1}^1 du \, \sqrt{1-u^2}\,, \tag{2.42}$$

and are thus already in the form suitable to use Chebyshev polynomials of the second kind.

The properties of many different sets of orthogonal polynomials can be found in Chapter 22 of Abramowitz & Stegun.

The generalisation to multi-dimensional integrals is straightforward. The sum becomes then a multiple sum, for example:

$$\int_{-1}^1 \int_{-1}^1 dxdy \, f(x,y) = \sum_{i=1}^n \sum_{j=1}^m w_i^x w_j^y f(x_i, y_i). \tag{2.43}$$

Note, however, that if the number of dimensions becomes too large Monte-Carlo integration (not treated in this course) becomes more efficient. Typically, and dependent on the integrand, this happens somewhere between five and eight dimensions.

For many applications a fixed order of the quadrature is sufficient. One can easily find out how many points are needed by increasing their number until the result does not change anymore and stay with that number. Alternatively, adaptive integration algorithms exist that rely on subdividing the integral into smaller integrals until a certain convergence criterion is reached. Ideally one can reuse the points already calculated. *Gauss-Kronrod quadrature* is an example of such an adaptive integration. In this course we will not need such elaborate algorithms.

**Exercise 1:**

How many grid points are needed within the Gauss-Legendre integration, to calculate the integrals

$$
\begin{aligned}
&(i) && \int_{-1}^{1} dx \sqrt{1-x^2} = \frac{\pi}{2}, \\
&(ii) && \int_{0}^{1} dx \ln^2(1+x) = 2\ln^2(2) - 4\ln(2) + 2, \\
&(iii) && \int_{0}^{1} dx \frac{1}{x}\ln(1-x) = -\zeta(2) = -\frac{\pi^2}{6}, \\
&(iv) && \int_{0}^{1} dx \ln^2(1-x) = 2,
\end{aligned}
$$

with a relative precision $10^{-6}$?
How many grid points are needed with the simple trapezoidal, respectively, with the Simpson rule?
Write your own program in a programming language of your choice.

For further reading

- W.H. Press, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes in F90/C++: The art of scientific computing*
- M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*

## 2.2 Interpolation

### 2.2.1 On the use of interpolations in integral equations

Functional methods typically entail the numerical solution of integral or integro-differential equations. To illustrate the need for interpolations it is sufficient to consider the general structure of an integral equation

$$f(x^1,\dots x^m) = g(x^1,\dots x^m) + \int d^m y\, \mathcal{I}[f(y^1,\dots y^m), K(x^1,\dots x^m, y^1,\dots y^m)] \tag{2.44}$$

in which the inhomogeneity $g(x^1,\dots x^m)$ might vanish (homogeneous integral eq.) or not, and in which the integrand $\mathcal{I}[f,K]$ is a linear functional of the function $f$ (linear integral eq.) or a non-linear one. In this course we will treat an inhomogeneous non-linear integral equation and homogeneous linear one.

Although the respective solution strategy is quite different in these two case there is one common feature: Due to the need of numerical integration and the respective discretisation of the continuous variables $\{x^1,\dots x^m\}$ to $m$ discrete sets of values $\{(x^1_1,\dots x^1_{n_1}),\dots(x^m_1,\dots x^m_{n_m})\}$ the function $f$ on the right-hand-side is needed only at discrete values of the variables and consequently the function $f$ on the left-hand-side will also only be calculated at some discrete values of its variables.

There are now three strategies to deal with this situation: First, and simplest, if possible, choose the arguments on the left-hand-side on the same grid as on the right-hand-side. Besides the fact that this is not always possible it is (except a few simple cases) also not recommended because choosing relatively few points results in an inadequate resolution and choosing relatively many points in a waste of CPU time. Second, one might use the integral equation itself to calculate the function for many more arguments on the left-hand-side as used on the right-hand side. This is typically a good strategy except the fact that it lacks the flexibility of the third option which is the use of interpolation.

To start we note that generically one can distinguish between two ways to describe a function known, necessarily in some degree of approximation, at discrete points. The first one is to try to approximate the general features of the available data based on a model function for these data. This is especially useful in case the data are known only within significant error margins or come with some noise and/or systematic error, as it is typical for experimental data or the results of calculations based on Monte-Carlo techniques. Analysing data in such a generic way is typically done by *least-squares regression*, a method which aims coming as close as possible to within the error bands of the data by choosing appropriate sets of parameters in the (quite often theoretically motivated) model function. The second approach requires the approximating function to pass through every known point. This, of course, makes only sense if the data represent the function (or will at the end of the calculation represent the function) very precisely. This second method is called *interpolation*, and in the following we will treat this case.

NB: Once a method for interpolation is defined and all the required parameters are determined one can use the resulting interpolating function also outside the intervals defined by the discrete variables. This is then called *extrapolation.* If no further information about the behaviour of the function outside the domain of the available data is known this is a very delicate procedure, and the corresponding results should always be considered with caution.

For simplicity we will treat first the one-dimensional case before generalising to multi-dimensional cases.

### 2.2.2 Polynomial interpolation

As done for the numerical integration one can use a polynomial to describe data. If $n+1$ points are known, a polynomial of order $n$ is required to make the polynomial pass through the data:

$$f(x) = y \approx a_0 + a_1 x + \ldots + a_n x^n. \tag{2.45}$$

A naive implementation requires a system of equations to be solved to obtain the coefficients $a_i$, namely:

$$f(x_i) = y_i = a_0 + a_1 x_i + \ldots + a_n x_i^n, \quad i = 0, 1, 2, \ldots, n. \tag{2.46}$$

For a well-defined function $f$ this system has a unique solution, but typically solving it directly is not very efficient. Even worse, problems like ill-conditioned matrices might appear.

If another representation for the polynomial is chosen, the determination of the coefficients is easier. *Lagrange interpolation* uses the representation:

$$f(x) = y \approx \sum_{i=0}^{n} b_i \prod_{j=0, j\neq i}^{n} (x - x_j). \tag{2.47}$$

For example, for $n = 2$, this yields

$$f(x) = y \approx b_0(x - x_1)(x - x_2) + b_1(x - x_0)(x - x_2) + b_2(x - x_0)(x - x_1). \tag{2.48}$$

The coefficients can easily be determined from the $f(x_i)$:

$$\begin{aligned} f(x_0) &= y_0 = b_0(x_0 - x_1)(x_0 - x_2), \\ f(x_1) &= y_1 = b_1(x_1 - x_0)(x_1 - x_2), \\ f(x_2) &= y_2 = b_2(x_2 - x_0)(x_2 - x_1). \end{aligned} \tag{2.49}$$

Solving for the $b_i$ and plugging these expressions back into Eq. (2.48) yields

$$f(x) = y \approx y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}. \tag{2.50}$$

For general $n$ the formula is

$$f(x) = y \approx \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}. \tag{2.51}$$

If the function to be described is a polynomial of order $n$ or less, using polynomials is, of course, exact. However, for Lagrange interpolation it is difficult to estimate the related error. Fig. 2.1 reveals that a Lagrange interpolation even for a simple function like $e^{-x}$ leads to significant artefacts. Another disadvantage of the Lagrange method is that results of a computation cannot be reused for interpolation at another

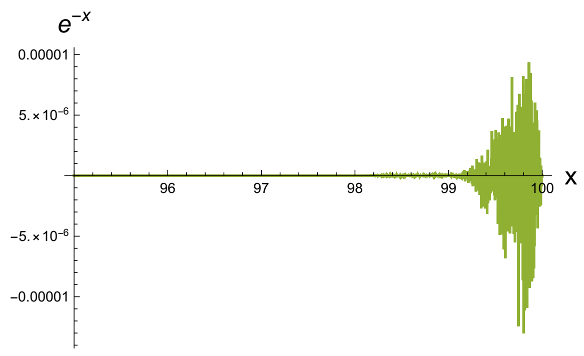


Figure 2.1: Lagrange fit of $e^{-x}$ using 101 points between 0 and 100. Clearly some artifacts are seen for high $x$.

point. Also, if another point is added to the known data, the complete calculation has to be redone (and will lead in many cases to significant differences).

A method that overcomes the last two problems is using a *Newton polynomial.* The approximating polynomial is in this case written as

$$\begin{aligned} f(x) &= y \\ &\approx c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \dots \\ &\quad + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}). \end{aligned} \tag{2.52}$$

The coefficients $c_i$ can be determined recursively starting with a constant interpolation $c_0 = f(x_0)$. Adding the point $x_1$ one determine the coefficient for linear interpolation:

$$c_1(x_1, x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}, \tag{2.53}$$

where we indicated the dependence of the coefficient $c_1$ on the variables $x_0$ and $x_1$. The next point $x_2$ is used to fix the quadratic coefficient as a function of $x_0$, $x_1$ and $x_2$:

$$c_2(x_2, x_1, x_0) = \frac{1}{x_2 - x_0}\left(\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}\right). \tag{2.54}$$

The $i$-th coefficient can be expressed as:

$$c_i(x_i, x_{i-1}, \dots, x_0) = \frac{c_{i-1}(x_i, x_{i-1}, \dots, x_1) - c_{i-1}(x_{i-1}, \dots, x_0)}{x_i - x_0}. \tag{2.55}$$

The recursive definitions of the $c_i$ are known as divided differences and hence this method is also known as Newton's divided-difference interpolation polynomial.

The Newton polynomial, Eq. (2.52), is in structure similar to a Taylor series. The error can thus be estimated from the next order $n + 1$:

$$E^n_{\text{Newton}} \approx c_{n+1}(x_{n+1}, \dots, x_0)(x - x_0)\cdots(x - x_n). \tag{2.56}$$

As one typically already used up all data points for the interpolation, one creates a new "data point" $(x_{n+1}, y_{n+1})$ from the polynomial itself at an arbitrarily chosen point $x_{n+1}$ for an error estimate.

### 2.2.3 Chebyshev interpolation

The typical problem of the above discussed interpolation methods is the fact that the error of interpolation is larger at the ends of the domain. To have a uniformly distributed error one is advised to use orthogonal polynomials as approximating functions. (NB: This is also within the class of polynomial interpolations.) As will become immediately evident Chebyshev polynomials are especially suited.

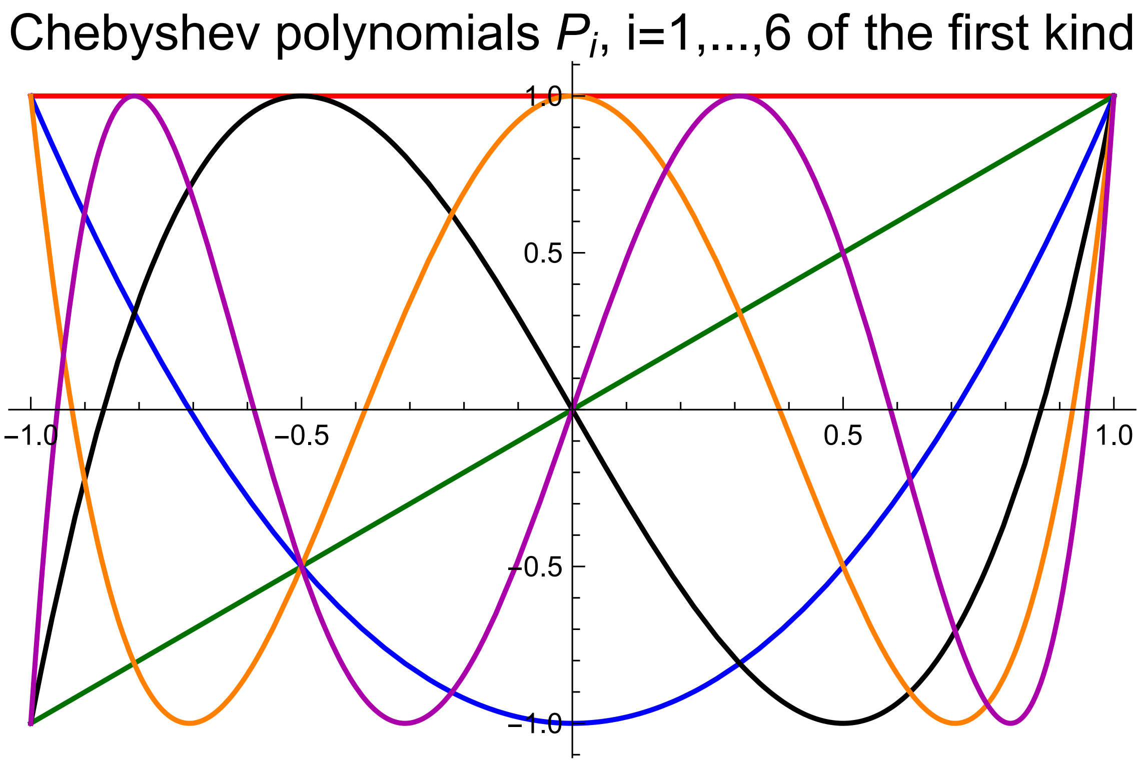


Figure 2.2: The first six Chebyshev polynomials, the number of nodes identifies the order.

**Chebyshev polynomials (of the first kind)**
The recursive definition of Chebyshev polynomials is ($-1 \le x \le 1$):

$$T_0(x) = 1, \tag{2.57}$$
$$T_1(x) = x, \tag{2.58}$$
$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k = 2, 3, \ldots \tag{2.59}$$

Chebyshev polynomials can be expressed in terms of cosines:

$$T_k(x) = \cos(k\,\theta) \tag{2.60}$$

with $\cos\theta = x$. The equivalence can be seen from $T_0(x) = \cos 0 = 1$, $T_1(x) = \cos(\arccos x) = x$ and the trigonometric identity

$$\cos((k+1)\theta) + \cos((k-1)\theta) = 2\cos\theta\cos(k\,\theta), \tag{2.61}$$

which yields the recursive relation Eq. (2.59).

The orthogonality relation of the polynomials is

$$\int_{-1}^{1} dx \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} = \begin{cases} 0 & n \neq m \\ \pi & n = m = 0 \\ \frac{\pi}{2} & n = m \neq 0 \end{cases} \tag{2.62}$$

The $k$ zeros $x_j$ can be determined from the zeros of the cosine at $\pm\pi/2, \pm 3\pi/2, \ldots$ in Eq. (2.60) as

$$k\,\theta = \frac{2j+1}{2}\pi, \quad j = 0, 1, \ldots k-1, \tag{2.63}$$

$$x_j = \cos\left(\frac{2j+1}{2k}\pi\right). \tag{2.64}$$

A special feature of Chebyshev polynomials is that they also possess a discrete orthogonality relation. Labelling with $x_j$, $j = 0, 1, \ldots, k-1$, the zeros of the polynomial $T_k(x)$ then one has for $n, m < k$:

$$\sum_{j=0}^{k-1} T_n(x_j)T_m(x_j) = \begin{cases} 0 & n \neq m \\ k & n = m = 0 \\ \frac{k}{2} & n = m \neq 0 \end{cases} \tag{2.65}$$

Note that Chebyshev polynomials oscillate between $-1$ and 1. The polynomial of order $n$ has $n+1$ extrema which are always $-1$ and 1, see Fig. 2.2.

In principle, one can create a standard polynomial by expressing single polynomials via Chebyshev polynomials. For example:

$$1 = T_0(x), \tag{2.66}$$

$$x = T_1(x), \tag{2.67}$$

$$x^2 = \frac{1}{2}(T_0(x) + T_2(x)). \tag{2.68}$$

One could also make use of the continuous orthogonality relation of the polynomials Eq. (2.62) and write it as

$$f(x) = y = \frac{d_0}{2} + \sum_{i=1}^{\infty} d_i T_i(x) \approx \frac{d_0}{2} + \sum_{i=1}^{M-1} d_i T_i(x)\,, \tag{2.69}$$

where the coefficients $d_i$ can be determined from Eq. (2.62):

$$d_i = \frac{2}{\pi}\int_{-1}^{1} dx \frac{f(x)T_i(x)}{\sqrt{1-x^2}}, \quad i = 0, 1, \ldots. \tag{2.70}$$

However, it is most convenient to make use of the discrete orthogonality relation Eq. (2.65). The coefficients can then be calculated as

$$d_i = \frac{2}{N}\sum_{k=0}^{N-1} f(x_k)T_i(x_k) = \frac{2}{N}\sum_{k=0}^{N-1} f\left(\cos\left(\frac{\pi(k+\frac{1}{2})}{N}\right)\right)\cos\left(\frac{\pi\, i(k+\frac{1}{2})}{N}\right) \tag{2.71}$$

and used in Eq. (2.69). At $x = x_k$ this approximation becomes exact.

The expansion in Chebyshev polynomials needs to be truncated for practical calculations:

$$f(x) = y \approx \frac{d_0}{2} + \sum_{i=1}^{M-1} d_i T_i(x), \tag{2.72}$$

A good choice for $M$ can be lower than the number of points $N$. This has the advantage of avoiding typical problems of high order polynomial interpolations. Since the Chebyshev polynomials are always between $-1$ and 1, the sum of the neglected contributions cannot be larger than the sum of the neglected $d_i$. Typically, the coefficients $d_i$ are rapidly decreasing, and the dominant contribution to the error can then be estimated to be $d_M T_M(x)$, which is an oscillatory function that is equally spread in the interval $[-1, 1]$. Thus the error is also equally spread which is a big advantage of Chebysev interpolation. In fact, Chebyshev interpolation yields a polynomial which is very close to the socalled minimax polynomial. This polynomial approximation minimizes the error of the approximation but is hard to calculate.

Instead of implementing Eq. (2.72) directly, it is more convenient to use *Clenshaw's recurrence formula* which allows a direct evaluation of the function[1]:

$$e_{m+1} = e_m = 0, \tag{2.73}$$
$$e_j = 2\,x\,e_{j+1} - e_{j+2} + d_j, \quad j = m-1, m-2, \dots, 1 \tag{2.74}$$
$$f(x) = e_0 = x\,e_1 - e_2 + \frac{d_0}{2}. \tag{2.75}$$

One particular advantage of the Chebyshev expansion is the rapid convergence. In fact, for smooth functions it can be exponentially convergent. The reason is Eq. (2.60), the relation of the Chebyshev polynomials with the periodic cosine functions. Thus the Chebyshev expansion can be seen as mapping the interval $[-1, 1]$ onto a circle. Since the treatment at the endpoints is decisive for the convergence properties, the Chebyshev expansion, having effectively no endpoints, works very well. This is also reflected in the accumulation of points at the boundaries of the interval $[-1, 1]$ and should be compared to the accumulation of points near the boundaries for Gauss quadrature, where it has a similar effect.

Eq. 2.71 requires knowledge of the function $f(x)$. Typically, this function is not known. However, one can still expand an unknown function in terms of Chebyshev polynomials and formulate the equations in terms of the coefficients of the expansions. For example, one can express the dressing function of a propagator using a Chebyshev expansion. Some initial values are determined by an ansatz for the function. Then one can formulate the corresponding integral equation as a minimisation problem for the Chebyshev coefficients: Find those values which fulfil the equation. A standard method to solve this problem is a multidimensional generalisation of Newton's root finding algorithm. (NB: This method is the standard method to solve the propagator Dyson-Schwinger equations of Yang-Mills theory. Its basic idea is also used in the solution for baryons as relativistic three-quark bound states via a Poincaré covariant Faddeev equation.)

[1]For an explanation of downward recurrence see the Appendix C to this chapter.

### 2.2.4 Rational interpolation

Many functions appearing in the amplitudes on Quantum Field Theory are not well approximated by polynomials but can be quite accurately interpolated by rational functions, that is quotients of polynomials

$$R_{n,m}(x) = \frac{P_n(x)}{Q_m(x)} = \frac{p_0 + p_1 x + \ldots + p_n x^n}{q_0 + q_1 x + \ldots + q_m x^m} \,. \tag{2.76}$$

This is due to the fact that rational functions can model better other functions with poles than a polynomial can. In rational interpolation one can furthermore build in the asymptotic behaviour of a function by a suitable choice of the degrees $n$ and $m$. Choosing, e.g., $m = n + 2$ will make $R_{n,m}(x)$ to vanish like $1/x^2$ for $x \to \infty$. In addition, without loss of generality one can choose $q_0 = 1$ such that $R_{n,m}(x = 0) = p_0$.

NB: Making the Taylor series of the rational fit function agree with the Taylor series of the function to be approximated for the first $n+m+1$ terms is called *Padé approximation*. It is a widespread technique in theoretical physics to estimate the locations of the poles of an amplitude.

For rational interpolation there exists a very sophisticated algorithm due to Bulirsch and Stoer, see, e.g., Sect. 3.2. of [Press et al., *Numerical recipes*], for a didactical presentation of a (simplified) version of the Bulirsch-Stoer algorithm. As rational interpolations will not be used during this course this topic will be not further detailed.

NB: If you want to use rational interpolation in MATHEMATICA the best way to proceed is to use a rational function as a model, e.g.,
model = $(p0 + p1\,x)/(1 + q1\,x + q2\,x^2)$, and employ then
FindFit[f,model,{ p0,p1,q1,q2 },x].

### 2.2.5 Spline interpolation

The main problem of polynomial and rational interpolations is the fact that the possibility to increase the precision by increasing the order of the polynomial(s) is very limited. For high orders, oscillations and wiggles will generically appear. A possible way out is the use of spline interpolation which is a piecewise interpolation using a low-order polynomial. Thereby, the connections between two pieces, resp. intervals, are called knots.

The simplest spline is of order 0 and corresponds to a step function: One approximates an interval between two points by the left or right boundary value. This is of course rarely an adequate representation of the true function. A first order spline is also known as *linear interpolation*. In this case, one connects two neighbouring data points $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ and extracts function values in the interval between them from that line:

$$f(x) \approx y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i), \quad x_i \le x \le x_{i+1}. \tag{2.77}$$

To implement this method, one needs first to find the interval where the point $x$ lies. Linear interpolation is very useful and highly efficient if the known data points are very dense and the effect of the discontinuity at the knots is negligible.

A *quadratic spline* interpolates between two points $x_{i-1}$ and $x_i$ via

$$f_i(x) \approx a_i + b_i x + c_i x^2, \quad i = 1, 2, \ldots, n. \tag{2.78}$$

There are three coefficients $a_i$, $b_i$ and $c_i$. In total we have then $3n$ coefficients. The conditions to determine them are:

- At the interior knots, i.e., for $i = 1, 2, \ldots, n-1$, the quadratic polynomials must agree with each other and the data points: $f_i(x_i) = f_{i+1}(x_i)$ must agree. This yields $2n - 2$ conditions.
- At the outer knots, $x_0$ and $x_n$, the polynomials must pass through the data points. This yields 2 conditions.
- The first derivative at the interior knots must be continuous:

  $$f_i'(x_i) = f_{i+1}'(x_i). \tag{2.79}$$

  For the coefficients this means $b_i + 2c_i x_i = b_{i+1} + 2c_{i+1} x_i$, $i = 1, 2, \ldots, n-1$. This yields $n - 1$ conditions.
- For the final condition there are several possibilities. For example, the second derivative at the final point should be zero, which leads to $c_n = 0$.

A widely used variant due to its good relation of precision to effort is *cubic splines.* The polynomial for interpolation between the points $x_{i-1}$ and $x_i$ is

$$f_i(x) \approx a_i + b_i x + c_i x^2 + d_i x^3, \quad i = 1, 2, \ldots, n. \tag{2.80}$$

We require $4n$ conditions to fix the coefficients:

- At the interior knots the polynomials must agree with each other and the data points: $f_i(x_i) = f_{i+1}(x_i)$. This yields $2n - 2$ conditions.
- At the outer knots the polynomials must pass through the data points. This yields 2 conditions.
- The first derivative at the interior knots must be continuous. This yields $n-1$ conditions.
- The second derivative at the interior knots must be continuous: $2c_i + 6d_i x_i = 2c_{i+1} + 6d_{i+1} x_i$. This yields $n - 1$ conditions.
- For the two final conditions several possibilities exist. For example, the second derivative at the endpoints should be zero.

Instead of solving this systems of $4n$ equations, we can determine the coefficients in a simpler way.

Let us start by rearranging the expression for linear interpolation:

$$f(x) = y \approx A(x) y_i + B(x) y_{i+1}, \tag{2.81}$$

where

$$A = \frac{x_{i+1} - x}{x_{i+1} - x_i}, \quad B = 1 - A = \frac{x - x_i}{x_{i+1} - x_i}. \tag{2.82}$$

This is actually nothing else than the Lagrange formula for two points. Now we extend this expression such as to fulfill the requirement of a continuous second derivative (currently it is zero). If we add a term that is zero at the knots, we do not spoil the agreement with the known values at the knots. If we had the values of the second derivatives at the knots, $y_i''$, we could do so by

$$f(x) = y \approx A(x)y_i + B(x)y_{i+1} + C(x)y_i'' + D(x)y_{i+1}'' \tag{2.83}$$

with

$$C(x) = \frac{1}{6}(A(x)^3 - A(x))(x_{i+1} - x_i)^2, \quad D(x) = \frac{1}{6}(B(x)^3 - B(x))(x_{i+1} - x_i)^2. \tag{2.84}$$

To obtain the second derivatives at the knots, we will use the requirement that the first derivatives are continuous at the knots. This will give us equations to determine the $y_i''$. The first derivative is

$$y'(x) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{3A(x)^2 - 1}{6}(x_{i+1} - x_i)y_i'' + \frac{3B(x)^2 - 1}{6}(x_{i+1} - x_i)y_{i+1}''. \tag{2.85}$$

At the knot $x_i$ we use Eq. (2.85) for the interval $[x_{i-1}, x_i]$ (replace $i \to i-1$) and $[x_i, x_{i+1}]$. After working out the details, we arrive at

$$\frac{x_i - x_{i-1}}{6}y_{i-1}'' + \frac{x_{i+1} - x_{i-1}}{3}y_i'' + \frac{x_{i+1} - x_i}{6}y_{i+1}'' = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}}. \tag{2.86}$$

These are $N-1$ linear equations in tridiagonal form for $N+1$ unknowns. Thus we are left with two parameters which have to be fixed.

One already mentioned possibility is to set the second derivatives at the end point to zero: $y_0'' = y_N'' = 0$. This is called *natural spline.* They provide an approximation in the first and last intervals that is flatter than the true curve. If one uses the for the second derivatives at the end points the values from their neighbours, $y_0'' = y_1''$ and $y_N'' = y_{N-1}''$, the approximation is quadratic in the outer intervals and the curvature is larger than that of the true curve. There are various other possibilities, for example, linear extrapolation for $y_0''$ and $y_N''$, or periodic boundary conditions for the first and the second derivative.

**Voluntary exercise:** Work out the missing steps from Eq. (2.83) to Eq. (2.86).

Eq. (2.86) is in tridiagonal form and can be solved in $O(N)$ steps by back-substitution. The values for the second derivatives need to be calculated only once and can then be reused.

**Solution of a tridiagonal equation**
The following system of equations is called a tridiagonal equation:

$$\begin{pmatrix} b_0 & c_0 & & & & \\ a_1 & b_1 & c_1 & & & \\ & a_2 & b_2 & c_2 & & \\ & & & \ddots & & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix} \begin{pmatrix} z_0 \\ z_1 \\ \\ \vdots \\ \\ z_n \end{pmatrix} = \begin{pmatrix} r_0 \\ r_1 \\ \\ \vdots \\ \\ r_n \end{pmatrix} \tag{2.87}$$

It is solved by iteratively eliminating the $z_i$. First, we multiply the first equation with $a_1/b_0$. We can then subtract it from the second equation:

$$\begin{array}{cccccc} a_1 z_0 & +c_0\frac{a_1}{b_0}z_1 & & = & r_0\frac{a_1}{b_0} \\ a_1 z_0 & +b_1 z_1 & +c_1 z_2 & = & r_1 \\ \hline & z_1\left(b_1 - \frac{c_0 a_1}{b_0}\right) & +c_1 z_2 & = & r_1 - \frac{r_0 a_1}{b_0} \end{array} \tag{2.88}$$

We rewrite this by defining

$$\beta_0 = b_0, \quad \rho_0 = r_0, \tag{2.89}$$

$$\beta_1 = b_1 - \frac{a_1 c_0}{\beta_0}, \quad \rho_1 = r_1 - \frac{a_1 \rho_0}{\beta_0}. \tag{2.90}$$

For the elimination of $z_1$ we multiply the second equation with $a_2/\beta_1$:

$$\left(b_2 - \frac{a_2}{\beta_1}c_1\right) z_2 + z_3 c_2 = r_2 - \frac{a_2}{\beta_1}\rho_1. \tag{2.91}$$

Again we introduce new variables:

$$\beta_2 = b_2 - \frac{a_2}{\beta_1}c_1, \quad \rho_2 = r_2 - \frac{a_2}{\beta_1}\rho_1. \tag{2.92}$$

If we iterate this process until the end of the system of equations, we obtain

$$\begin{array}{ccccccc} \beta_0 z_0 & +c_0 z_1 & & & & = & \rho_0 \\ & \beta_1 z_1 & +c_1 z_2 & & & = & \rho_1 \\ & & \beta_2 z_2 & +c_2 z_3 & & = & \rho_2 \\ & & & \ddots & & & \\ & & & \beta_{n-1} z_{n-1} & +c_{n-1} z_n & = & \rho_{n-1} \\ & & & & +\beta_n z_n & = & \rho_n \end{array} \tag{2.93}$$

The $\beta_i$ and $\rho_i$ are given by

$$\beta_i = b_i - \frac{a_i}{\beta_{i-1}}c_{i-1}, \quad \rho_i = r_i - \frac{a_i}{\beta_{i-1}}\rho_{i-1}, \quad i = 1, 2, \ldots, n. \tag{2.94}$$

$z_n$ can be calculated as

$$z_n = \rho_n / \beta_n \tag{2.95}$$

and $z_{n-1}$ as

$$z_{n-1} = \frac{\rho_{n-1} - c_{n-1} z_n}{\beta_{n-1}}. \tag{2.96}$$

The general result is

$$z_{n-j} = \frac{\rho_{n-j} - c_{n-j} z_{n-j+1}}{\beta_{n-j}}, \quad j = 1, 2, \ldots, n. \tag{2.97}$$

The $N-1$ equations from Eq. (2.86) are mapped to the $n$ equations of Eq. (2.87) as follows ($i = 1, 2, \dots, N-1$):

$$z_i \leftrightarrow y_i'' \tag{2.98}$$

$$r_i \leftrightarrow \frac{y_{i+1}-y_i}{x_{i+1}-x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{2.99}$$

$$a_i \leftrightarrow \frac{x_i - x_{i-1}}{6} \tag{2.100}$$

$$b_i \leftrightarrow \frac{x_{i+1} - x_{i-1}}{3} \tag{2.101}$$

$$c_i \leftrightarrow \frac{x_{i+1} - x_i}{6} \tag{2.102}$$

The values for $y_0''$ and $y_N''$ are determined from the extra conditions we want to implement by adding equations for $i = 0$ and $i = N$ appropriately.

**Voluntary exercise:** Write down the system of equations for natural splines and for $y_0'' = y_1''$ and $y_N'' = y_{N-1}''$.

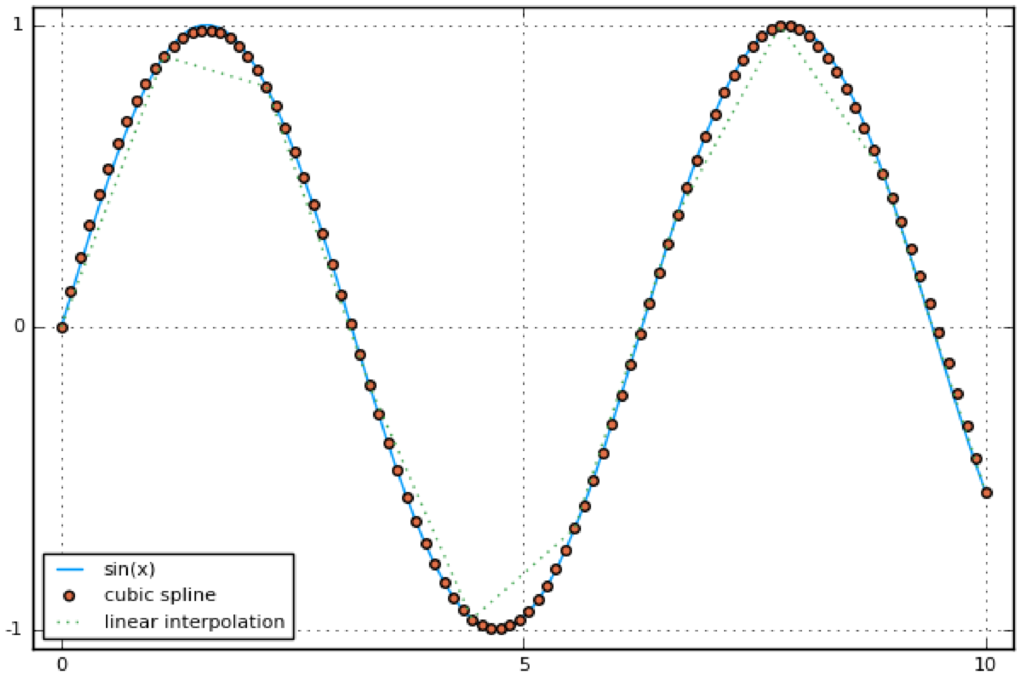


Figure 2.3: Linear and cubic splines of $\sin(x)$ based on (only!)10 points. Note the larger curvatures close to the boundaries which is due to the choice $y_0'' = y_1''$, $y_N'' = y_{N-1}''$.

An implemention of a cubic spline interpolation consists then of the following steps:

0. Calculate the second derivatives (only the first time).
1. Find the appropriate interval $[x_{i-1}, x_{i+1}]$.
2. Use Eq. (2.83) to interpolate.

An example that compares linear and cubic splines is shown in Fig. 2.3. Even though the given number of points is very low, the cubic spline recovers the characteristics of the function quite accurately.

### 2.2.6 Multidimensional interpolation

Interpolation in more than one dimension becomes quickly much more complicated. We will treat here the case of bilinear interpolation, i.e., linear interpolation in two dimensions, and will shortly explain the idea of further possibilities.

First of all one has to distinguish two cases concerning the known function values (input data): Do we have input data on a regular grid or is the data scattered? We will only treat the former case here, as we can always setup the solution of an integral equation in this way.

Furthermore we will restrict ourselves now to two dimensions. In two dimensions the data is represented by values $y_{ij}$, where $i$ and $j$ label the points in the two directions. Thus we also have two arrays $x_{1i}$ and $x_{2j}$. The procedure for interpolation is the same as in one dimension: Find the region, in this case the square, where the evaluation point $(x_1, x_2)$ is located:

$$x_{1i} \leq x_1 \leq x_{1(i+1)}, \tag{2.103}$$

$$x_{2j} \leq x_2 \leq x_{2(j+1)}, \tag{2.104}$$

and apply the interpolation formula,

$$y(x_1, x_2) = (1-t)(1-u)y_{ij} + t(1-u)y_{(i+1)j} + t\,u\,y_{(i+1)(j+1)} + (1-t)u\,y_{i(j+1)}, \tag{2.105}$$

where

$$t = \frac{x_1 - x_{1i}}{x_{1(i+1)} - x_{1i}}, \tag{2.106}$$

$$u = \frac{x_2 - x_{2j}}{x_{2(j+1)} - x_{2j}}. \tag{2.107}$$

Furthermore, $y_{ij} = y(x_{1i}, x_{2j})$. When implementing bilinear interpolation one can reuse functions from the one-dimensional case.

The disadvantage of bilinear interpolation is again that the derivatives at the boundaries of the squares are not continuous. Requiring smoothness of the derivatives (and cross derivatives $\partial_{x_1}\partial_{x_2}$) leads to *bicubic interpolation*. However, the values for the derivatives need to be supplemented separately. If they are not known analytically, one can use, for example, finite differences as approximation.

Alternatively, one can use *bicubic splines*. They are easier to implement but slower. Here the idea is to perform splines in dimension one at every point $(x_{1i}, x_2)$. This yields a vector of values on which a second spline is performed. This, however, requires to repeatedly construct the splines which costs computing time.

In several cases one wants to interpolate functions of two variables which vary only slowly in one of the two variables. E.g., in (hyper-)spherical coordinates dependencies on an angle can be weak. Then the method of choice with a good precision-to-effort ratio is to combine a cubic spline in the radial variable with a linear interpolation in the angle variable.

## Appendix

### A Chebyshev polynomials of the second kind

As the Chebyshev polynomials of the first kind the Chebyshev polynomials of the second kind are defined on the interval $[-1,1]$. Note, however, the different weight function required to make them orthogonal polynomials, *c*f. table on page 12.
The recursive definition of Chebyshev polynomials of the second kind, $U_n(x)$, is given by :

$$U_0(x) = 1, \tag{2.108}$$
$$U_1(x) = 2x, \tag{2.109}$$
$$U_k(x) = 2xU_{k-1}(x) - U_{k-2}(x), \quad k = 2,3,\ldots \tag{2.110}$$

Note that the recursion relation is identical to the one for Chebyshev polynomials of the first kind, see 2.2.3. The difference in the polynomials comes about due to $U_1(x) = 2x = 2T_1(x)$ , and this additional factor 2 propagates through all $U_n(x)$, $n = 2,3,4\ldots$.
The orthogonality relation of the polynomials is

$$\int_{-1}^{1} dxU_n(x)U_m(x)\sqrt{1-x^2} = \left\{ \begin{array}{ll} 0 & n \neq m \\ \frac{\pi}{2} & n = m \end{array} \right. \tag{2.111}$$

The $k$ zeros $x_j$ are

$$x_j = \cos\left(\frac{j+1}{k+1}\pi\right). \tag{2.112}$$

The discrete orthogonality relation for $n, m < k$ is

$$\sum_{j=0}^{k-1}(1-x_j^2)U_n(x_j)U_m(x_j) = \left\{ \begin{array}{ll} 0 & n \neq m \\ \frac{k+1}{2} & n = m \end{array} \right. \tag{2.113}$$

where $x_j$ are the zeros of the polynomial $U_k(x)$, $j = 0,1,\ldots,k-1$.
A function can be expanded in terms of Chebyshev polynomials of the second kind as

$$f(x) = \sum_{i=0}^{n} c_i U_i(x). \tag{2.114}$$

The coefficients can be calculated from the discrete orthogonality relation:

$$c_i = \frac{2}{n+2}\sum_{j=0}^{n}(1-x_j^2)f(x_j)U_i(x_j)\,, \tag{2.115}$$

where the $x_j$, $j = 0,\ldots n$ are the zeros of $U_{n+1}(x)$, $U_{n+1}(x_j) = 0$.

**Voluntary exercise:** Derive (2.115).

**B Modified Bessel functions**

As during this course we will need some of the modified Bessel functions of integer order a few of their properties will be briefly summarised here. More details can be found in refs. [Abramowitz & Stegun, *Handbook of Mathematical Functions*, p. 374 - 379] and [Press et al., Numerical Recipes].
The solutions of the differential equation

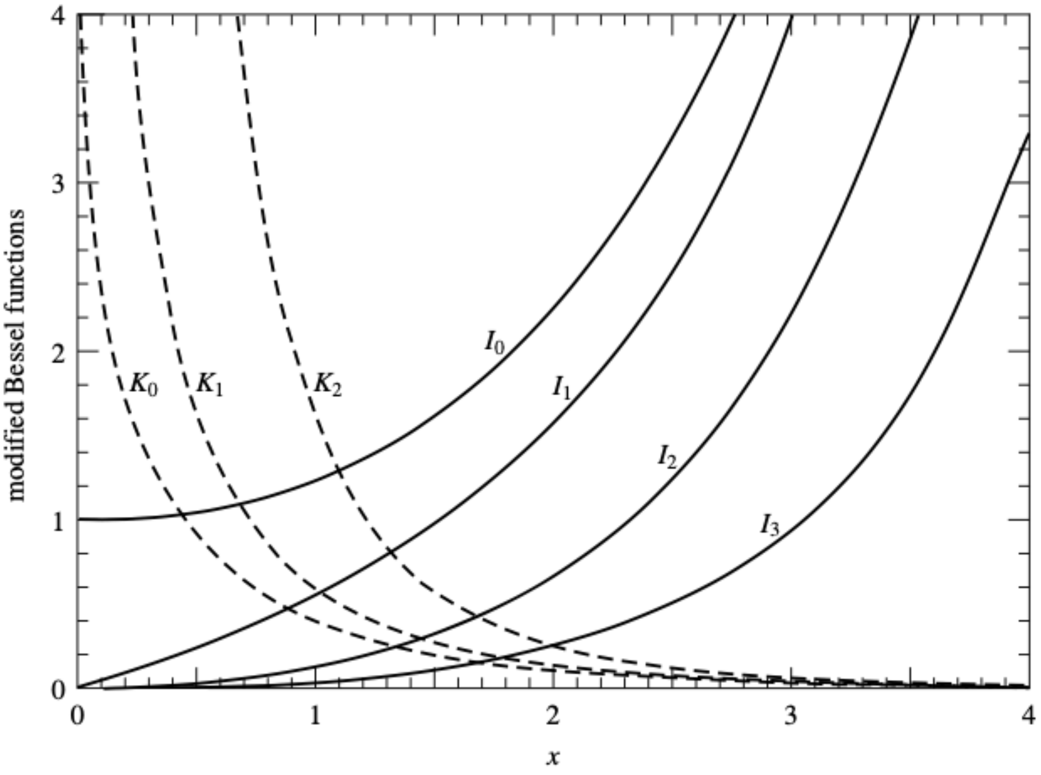
$$x^2 f''(x) + x f'(x) - (x^2 + n^2) f(x) = 0\,, \quad n = 0, 1, 2, \ldots, \tag{2.116}$$

are denoted by $I_n(x)$ and $K_n(x)$. The complex functions $I_n(z)$ are entire functions of the complex variable $z$. (The functions $K_n(z)$ possess a cut along the negative real axis. For both sets of functions an analytic continuation of $n$ to $\nu$ exists such that they are for $z \neq 0$ analytic functions in $\nu$.)
The modified Bessel functions $I_n(x)$ and $K_n(x)$ are equivalent to the usual Bessel functions $J_n(x)$ and $Y_n(x)$ evaluated for purely imaginary arguments, in particular one has $I_n(x) = (-i)^n J_n(ix)$. Whereas the $K_n(x)$ diverge for $x \to 0$ one has

$$I_n(x) = \frac{1}{n!}\left(\frac{x}{2}\right)^n + \ldots, \tag{2.117}$$

*i.e.*, practically the same behaviour as the Bessel function $J_n(x)$. For $x \gg n$ the behaviour is dominated by an exponential

$$I_n(x) = \frac{1}{\sqrt{2\pi x}} e^x\,, \quad K_n(x) = \frac{\sqrt{\pi}}{\sqrt{2x}} e^{-x}\,. \tag{2.118}$$



Modified Bessel functions (Source: Numerical Recipes).

We note already here that in one of the problems to be discussed the functions $I_{0,1,2}(x)$ appear, however, it is more precisely the combination $e^{-x}I_{0,1,2}(x)$ which has to be evaluated. Therefore, it is recommended that you are going to write your own routine to evaluate $e^{-x}I_{0,1,2}(x)$. Once the exponential factor has been removed for large $x$ the functions $I_0(x)$ and $I_1(x)$ can be calculated by simple polynomial approximations for small and large $x$ each, the coefficients can be found in [Abramowitz & Stegun, *Handbook of Mathematical Functions*, p. 378]. The remaining function $I_2(x)$ can be calculated by a straightforward application of the recurrence relation $I_2(x) = -(2/x)I_1(x) + I_0(x)$.

With respect to the recurrence relations

$$I_{n+1}(x) = -\left(\frac{2n}{x}\right) I_n(x) + I_{n-1}(x)\,, \tag{2.119}$$

$$K_{n+1}(x) = +\left(\frac{2n}{x}\right) K_n(x) + K_{n-1}(x)\,, \tag{2.120}$$

a remark is in order. As all other sets of functions fulfilling a second-order differential equation there are directions in the complex plane in which the functions are exponentially enhanced, resp., suppressed.
These functions typically also respect recurrence relations, *cf.* the Legendre and Chebychev polynomials discussed before. Both properties taken together lead to the fact that if an upward recursion is numerically stable (*i.e.,* error propagation is suppressed) the downward recursion is numerically unstable (*i.e.,* error propagation is enhanced), and vice versa. For the modified Bessel functions the naïve upward recursion is always unstable. For the functions $K_n$ this presents hardly any problem because the $K_n$ are increasing anyhow strongly with increasing $n$. However, for the functions $I_n(x)$ an upward recursion is very fast failing due to numerical instability, one need the technique of downward recurrence, see Appendix C below.

**C Up- and downward recurrence**

Recurrence relations of the form

$$f_{n+1}(x) = c_n^{(n)} f_n(x) + c_{n-1}^{(n)} f_{n-1}(x) \tag{2.121}$$

are typically, as $f_0(x)$ and $f_1(x)$ can be easily calculated, evaluated in the way they are written. This is called upward recurrence. Quite often they can applied in a straightforward manner but almost equally often they fail due to numerical instability. The above discussed modified Bessel functions provide one example in which upward recurrence fail for all real values of $x$. More common are cases in which the stability direction of a recurrence relation depend on the value of the argument, *e.g.*, for a spherical Bessel function $j_l(x)$ an upward recurrence is stable for $x > l$ but unstable for $x < l$.
In such cases one can apply the technique of downward recursion. (NB: Note that Clenshaw's recurrence formula is also a downward recursion.) To this end one "inverts" the recurrence relation into the form

$$f_{n-1}(x) = d_n^{(n)} f_n(x) + d_{n-1}^{(n)} f_{n-1}(x) \tag{2.122}$$

in order to calculate from it $f_N(x)$. For some large $M \gg N$ one starts with $f_M(x) = 0$ and $A\, f_{M-1}(x) = 1$ with some yet unknown constant $A$ to apply the recurrence downwards. Employing the recurrence in the downward direction from some arbitrary starting value for an upward-unstable recurrence will put us onto the correct solution modulo a constant pre-factor. To determine the latter we run, so-to-say with the constant $A$ in the cache, down to $f_0(x)$ and use then the known value of $f_0(x)$ to calculate the constant $A$ which in turn then allows to determine $f_N(x)$. The procedure is straightforward except two subtleties.
First, how large should $M$ be? This can be determined from the asymptotic forms of the functions $f_n(x)$. For $f_n(x) \propto x^n + \ldots$ in the region of interest one best chooses $M$ by adding to $N$ a term *const.* $\times \sqrt{N}$ where the constant provides roughly the number of significant figures of accuracy. Therefore, choosing this constant to be of order 10 is a good choice.
Second, during recursion, especially for large values of $N$ and thus $M$ the calculated numbers can get very small or very large. Then it is recommendable to "renormalise" the function values. Suppose $f_n(x)$ becomes larger than $10^{10}$ then it is helpful to multiply $f_{n+1}(x)$ and $f_n(x)$ with $10^{-10}$ before calculating $f_{n-1}(x)$ Note that in this step $A$ needs to be rescaled accordingly. (NB: Such "renormalisations" are also recommendable for long upward recurrences although it is much less often needed.

**D Double-exponential integration**

The double-exponential integration method, sometimes also called tanh-sinh-rule, is very useful if, after a suitable variable transformation to a closed interval or region, the integrand possesses endpoint singularities, infinite derivatives, or similar unusual features.
For simplicity the method is briefly reviewed for a one-dimensional integral over the interval [-1,1]. With denominating the step size by $h$ the rule is implemented for an odd number $N$ of mesh points by:

$$\begin{aligned} \int_{-1}^{1} dx\, f(x) &\approx \sum_{k=-(N-1)/2}^{(N-1)/2} w_k f(x_k)\,, \\ x_k &= \tanh\left(\frac{\pi}{2}\sinh(kh)\right)\,, \\ w_k &= h\frac{\pi}{2}\frac{\cosh(kh)}{\cosh^2(\frac{\pi}{2}\sinh(kh))}. \end{aligned} \tag{2.123}$$

One may view this method also as a change of variables from $x \in (-1,1)$ to $t \in (-\infty,\infty)$ with the transformation rule $x = \tanh\left(\frac{\pi}{2}\sinh(t)\right)$ and equal spacing in the variable $t$.
This method has the following advantages:

- Potential endpoint singularities of $f(x)$ at $x = \pm 1$ are mapped to $t = \pm\infty$, and the value of the function is there doubly-exponentially suppressed (therefore the name) and in practically all cases removed.
  (NB: For smooth integrands it is not as efficient as Gauß integration.)
- It can be used in a progressive manner, one can halve the step size, while reusing the already evaluated function values, in order to estimate the precision.
- Cost of calculating weights is $\mathcal{O}(N^2 \log^2 N)$ as compared to $\mathcal{O}(N^3 \log N)$ for Gauß integration.
- Contained in the C++ library "Boost", see www.boost.org.

NB: The method in the corresponding arbitrary precision version is used for producing conjectures by numerical evaluation ("experimental mathematics"). Hereby precisions up to 20000 digits are used.

**Exercise 2:**

In the following we consider the function

$$f(x) = \frac{0.2}{x + 0.5} + 0.002\,,$$

and two maps of the interval $x \in [0, \infty)$ to the interval $z \in [-1, 1]$:

$$z = \left(\frac{x-1}{x+1}\right)^m \qquad m = 1, 3.$$

Furthermore, we only use values of the function $f_k := f(x(z_k)$ on an equidistant grid in $z$, $z_k = -1 + 2k/N$, $k = 1, \ldots N-1$, for different values of $N$.

a.) Write a numerical program which interpolates this function linearly in between the grid values $z_k$ at mid values $(z_k + z_{k+1})/2$ and transform back to the variable $x$. Plot the function together with the interpolated values for different values of $N$.

b.) Perform the same task but this time with cubic spline interpolation. For this you have to do the following:

   (i) Write a solver for a tridiagonal equation.

   (ii) Use the tridiagonal solver to estimate the second derivatives w.r.t. the variable $z$. At the boundaries one can assume the second derivatives to be constant, $f_0'' = f_1''$ and $f_{N-1}'' = f_N''$.

   (iii) Use the interpolation formula $f(x(z)) \approx A(z)f_k + B(z)f_{k+1} + C(z)f_k'' + D(z)f_{k+1}''$ with the respective coefficients $A, B, C$ and $D$.

   (iv) Transform back to the variable $x$.

c.) In the third version use Chebychev interpolation. Again this requires four subtasks:

   (i) Write a function to calculate the Chebychev polynomials using the recurrence relation as given in the lecture.

   (ii) Determine the coefficients $d_k$ using the discrete orthogonality relation, see the lecture for the corresponding formula.

   (iii) Use Clenshaw's recurrence formula as given in the lecture for the interpolation.

   (iv) Transform back to the variable $x$.

# 3 Generating Functionals and Functional Equations in Quantum Field Theory

## 3.1 Basic Aspects of Generating Functionals and Functional Equations

In theoretical physics, and hereby especially in the context of quantum theory, one typically considers functionals which are maps from a space of functions to the field of real or complex numbers. Within this course I will use the (wide-spread) notation that a function is written as $f(x)$, respectively, $f(\vec{x})$, $\vec{x} = (x_1, \dots x_N)$, $N < \infty$, and a functional as $F[f]$.

An illustrative example is the action $S[q_1, \dots q_n]$ with the $q_i$ being coordinates or $S[\phi, \Psi, A^\mu]$ where $\phi(x)$ is a scalar, $\Psi(x)$ a Dirac and $A^\mu(x)$ a vector field.

A functional equation is in general a relation between functions in implicit form. A simple example is provided by the equation $f(x+y) = f(x)f(y)$ for a real function $f$ of a real variable $x$. This equation is obviously fulfilled by $f(x) = e^x$. A functional equation of this type is called an algebraic function equation.

In quantum field theory, quantum many-body systems and statistical physics functional equations appear as partial differential, integral and/or integro-differential equations for correlation functions. For interacting systems one encounters in general an infinite hierarchy of functional equations. We will discuss an important case in Sect. 3.4.

The derivation of functional equations for the correlation functions is most transparent in the formalism based on Functional Integrals but with a little bit of more work one can derive them also in other formalisms like the canonical operator formalism, in Stochastic Quantisation, etc.. To this end one defines a Generating Functional expressed as a Functional Integral which "sums" all possible field configurations. In Quantum Mechanics the corresponding integral is formulated as an integral over all possible paths of a particle. (Hence, this quantity is also known under the name Path Integral.) Besides the Schrödinger and the Heisenberg picture it provides a third type of formalism for quantum theory, which also is referred to as "Feynman's sum over all possible histories".

Before discussing Generating Functionals it is worthwhile to note that Generating Functions are a wide-spread tool in mathematics, *cf.* Table 22.9 of Abramowitz and Stegun. An example is given by

$$g(t,x) = e^{tx} \cosh(t\sqrt{1-x^2}) \tag{3.1}$$

which is called generating function, more precisely exponential generating function, of Chebychev polynomials due to the expansion:

$$g(t,x) = \sum_{n=0}^{\infty} T_n(x) \frac{t^n}{n!} \,. \tag{3.2}$$

Of course, the dependence on $t$ can be generalised to a dependence on many variable $t_i$, $i = 1, \dots N < \infty$. Performing the limit from many $t_i$ defined on a mesh to a continuous variable dependence $t(y)$ one arrives at a Generating Functional.

## 3.2 Generating Functionals in Statistical Physics

In Statistical Physics calculations for a canonical ensemble start in general from a partition function

$$Z(T,V,N) = \mathrm{Tr}\; e^{-\hat{H}/T} \tag{3.3}$$

where $T$ denotes the temperature, $V$ the volume and $N$ the particle number. Here, $k_B = 1$ and temperatures are measured in energy units. $\hat{H}$ is the Hamilton operator of the (quantum) many-body system under consideration. The trace is to be understood as the sum over all eigenvalues of the Hamilton operator $\hat{H}$. To distinguish traces and determinants of operators (also called functional traces and determinants) from the mathematically much more harmless matrix traces and determinants, functional traces and determinants will be denoted by "Tr" and "Det" whereas matrix traces and determinants will written as "tr" and "det".

The free energy is basically the logarithm of the partition function,

$$F(T,V,N) = -T \log Z(T,V,N) \,. \tag{3.4}$$

The internal energy is then obtained via a Legendre transformation

$$E(S,V,N) = F(T,V,N) + TS \,. \tag{3.5}$$

Note that this formula is valid for equilibrium states in which the entropy $S$ is maximal. The internal energy is then a function of the entropy $S$ instead of the temperature $T$. The construction guarantees that for a bounded Hamiltonian one obtains an internal energy which is a convex function of the entropy, and thermodynamic consistency is consequently built-in (if the system is treated exactly or in a consistent approximation).

Next we consider a spin system, and we couple an external magnetic field to use the trick to calculate thermodynamic quantities by deriving with respect to the external field, e.g., for the local magnetisation

$$m_i = -\frac{1}{V} \left\langle \frac{\partial \hat{H}}{\partial h_i} \right\rangle \,, \tag{3.6}$$

where we exploited that for fixed spin locations $x_i$ only the external magnetic field at these positions, $h_i := h(x_i)$, is required. To derive a two-point correlations function we can couple the external magnetic field by substituting

$$\hat{H} \to \hat{H} - \sum_l \sigma_l h_l \tag{3.7}$$

in the partition function

$$Z(T,V,N;\{h_l\}) = \sum_{\text{all configurations}} e^{-(\hat{H}-\sum_l \sigma_l h_l)/T} \,. \tag{3.8}$$

The two-point correlation function can then calculated by using

$$\langle \sigma_l \sigma_k \rangle \propto \frac{\partial}{\partial h_l}\frac{\partial}{\partial h_k}\, Z|_{h_i=0} =: G(l,k) = G(l-k)\,, \tag{3.9}$$

where in the last step we have used translational invariance. Obviously, this works not only for the two-point function but for any $n$-point function, and thus $Z$ is a (high-dimensional) generating function for the correlation functions

$$Z(T,V,N;\{h_l\}) = \sum_{n=0}^{\infty} \frac{1}{n!} G(l_1,\dots l_n) h_{l_1}\dots h_{l_n}\,. \tag{3.10}$$

Besides $F(T,V,N;\{h_l\}) \propto \log Z$ being the free energy it is also a generating function:

$$F(T,V,N;\{h_l\}) = \sum_{n=0}^{\infty} \frac{1}{n!} G_{conn}(l_1,\dots l_n) h_{l_1}\dots h_{l_n}\,, \tag{3.11}$$

where the $G_{conn}(l_1,\dots l_n)$ are the cumulants with respect to the correlations $G(l_1,\dots l_n)$, e.g.,

$$G_{conn}(l_1,l_2) = \langle \sigma_{l_1}\sigma_{l_2}\rangle - \langle \sigma_{l_1}\rangle\langle \sigma_{l_2}\rangle\,. \tag{3.12}$$

The index *conn* is here for connected correlation function, a name which is only evident in the language of Feynman diagrams, see the lecture Quantum Field Theory.

Going to the thermodynamic limit of infinitely many spins the free energy is not any more a function of the $\{h_l\}$ but a functional. Taking also the continuum limit we write then $F[h(x)]$. From the free energy $F[h(x)]$ one can obtain via a Legendre transformation a very useful effective energy functional

$$\Gamma[S(x)] = -F[h(x)] + \int d^dx\, S(x)h(x)\,, \tag{3.13}$$

where $S(x) = \langle \sigma(x)\rangle_{h(x)}$ is the averaged spin field in the presence of an external magnetic field. Using $\frac{\delta}{\delta f}$ as notation for functional derivatives one has

$$h(x) = \frac{\delta \Gamma[S]}{\delta S(x)}\,. \tag{3.14}$$

$\Gamma$ is a Generating Functional for the so-called vertex functions describing interactions including the thermodynamic fluctuations

$$\Gamma[S] = \sum_{n=0}^{\infty} \frac{1}{n!}\Gamma(x_1,\dots x_n) S(x_1)\dots S(x_n) \tag{3.15}$$

with

$$\Gamma(x_1,\dots x_n) = \frac{\delta}{\delta S(x_1)}\cdots\frac{\delta}{\delta S(x_n)}\,\Gamma[S]|_{S(x)=0} \tag{3.16}$$

which can be understood as continuum limit of

$$\Gamma(\{S_l\}) = \sum_{n=0}^{\infty} \frac{1}{n!}\Gamma(l_1,\dots l_n) S_{l_1}\dots S_{l_n} \tag{3.17}$$

with

$$\Gamma(l_1,\dots l_n) = \frac{\partial}{\partial S_{l_1}}\cdots\frac{\partial}{\partial S_{l_n}}\,\Gamma[\{S\}]|_{S_{l_i}=0}\,. \tag{3.18}$$

## 3.3 Generating Functional for lattice field theory and continuum field theory

In (classical) Statistical Physics this technique provides correlation functions such that from the related expectation values one calculates directly probabilities. Keeping in mind that in a quantum theory these steps will lead to probability amplitudes (the absolute squares of which are then probabilities) we can otherwise employ the same technique in order to calculate correlation functions in a quantum field theory or a quantum many-body system. Then the quantum fluctuations are taken into account when summing configurations, and the weight of a configuration changes from the Boltzmann factor $e^{-H/T}$ to the weight of a Feynman functional integral $\exp(iS/\hbar)$.

Here I will make a big step and assume that at the level of the action a Wick rotation (for its definition see the lecture Quantum Field Theory) has been performed. This maps a spacetime with $d_S$-dimensional space to a $d_S+1$-dimensional Euclidean space, and the oscillatory weight $\exp(iS/\hbar)$ to an exponential one, $\exp(-S_E/\hbar)$.

To understand how the different functionals have to be understood we temporarily approximate the $d_S+1$-dimensional Euclidean space by a (hyper-)cubic lattice but now on the lattice sites one does not have spins as in the Ising or Heisenberg statistical physics model but a field $\phi(x)$. For the illustrative purpose aimed at here it is sufficient to treat a real-valued field, and at every lattice site one has $\phi_l := \phi(x_l) \in (-\infty, \infty)$.

Exploiting that the integral $\int \prod_{l=1}^N d\phi_l$ is equivalent to a sum over all configurations, and coupling on every lattice site a “source” term to the field one defines (for simplicity $\hbar = 1$ is used)

$$Z(\{j_l\}) = \sum_{\text{all configurations}} e^{-S_E + \sum_l j_l \phi_l} , \tag{3.19}$$

which is now clearly a Generating Function for the correlation function describing the autocorrelations of the field $\phi$. In analogy to the free energy one defines the Generating Function of the connected correlation functions via the relation

$$Z(\{j_l\}) = \exp\left(W(\{j_l\})\right) , \tag{3.20}$$

and evidently $W(\{j_l\})$ generates the connected correlation functions. E.g., a simple calculation yields

$$G_{conn}(l-k) = \frac{\partial W}{\partial j_l \partial j_k} = \frac{\partial Z}{\partial j_l \partial j_k} - \frac{\partial Z}{\partial j_l}\frac{\partial Z}{\partial j_k} . \tag{3.21}$$

In a next step one obtains an effective quantum action (which includes, in principle, all quantum fluctuations!) by a Legendre transformation

$$\Gamma(\{\Phi_l\}) = -W(\{j_l\}) + \sum_l \Phi_l j_l , \tag{3.22}$$

where the $\Phi_l$ are the expectation values of the $\phi_l$ in the presence of the sources $j_l$.

Taking the continuum limit, i.e., performing the limit of lattice spacing to zero $a \to 0$,

$$\phi_l \to \phi(x) , \quad j_l \to j(x) \quad \sum_l \to \int d^d x = \int d^4 x , \tag{3.23}$$

the Generating Functions become Generating Functionals

$$Z[j] = \int \mathcal{D}\phi e^{-S_E[\phi]+\int d^4x j(x)\phi(x)}\,, \quad \mathcal{D}\phi = \lim_{N\to\infty}\int \prod^N d\phi_l \tag{3.24}$$

such that

$$G^{(n)}(x_1, \dots x_n) = \left.\frac{\delta^n Z}{\delta j(x_1)\dots\delta j(x_n)}\right|_{j=0}\,, \tag{3.25}$$

and

$$G^{(n)}_{conn}(x_1, \dots x_n) = \left.\frac{\delta^n W}{\delta j(x_1)\dots\delta j(x_n)}\right|_{j=0}\,. \tag{3.26}$$

Furthermore,

$$\Gamma[\Phi(x)] = -W[j(x)] + \int d^4x \Phi(x) j(x)\,, \tag{3.27}$$

with

$$\Phi(x) = \langle\phi(x)\rangle_j = \frac{\delta W}{\delta j(x)}\,. \tag{3.28}$$

The functional derivatives of the effective action $\Gamma[\Phi(x)]$ defines the vertex functions (also sometimes called proper correlation functions or one-particle irreducible correlation functions):

$$\Gamma^{(n)}(x_1, \dots x_n) = \left.\frac{\delta^n \Gamma}{\delta\Phi(x_1)\dots\delta\Phi(x_n)}\right|_{\Phi=0}\,. \tag{3.29}$$

**Voluntary exercise:** Prove that the connected two-point function $G^{(2)}_{conn}(x_1, x_2)$ is the inverse of the two-point vertex function $\Gamma^{(2)}(x_1, x_2)$,

$$G^{(2)}_{conn}(x_1, x_2) = \left(\Gamma^{(2)}(x_1, x_2)\right)^{-1}\,.$$

NB:
1. Due to its special role in the theory the connected two-point function $G^{(2)}_{conn}(x_1, x_2)$ has a special name, it is called the propagator $G^{(2)}_{conn}(x_1, x_2) = D(x_1, x_2)$. In a homogeneous system it depends only on the difference $x_1 - x_2$, in a homogeneous and isotropic system only on the distance, $D(x_1, x_2) \to D(|x_1 - x_2|)$.
2. One can define also all these function only as functional derivatives without setting the sources to zero. If we use such a function in this lecture it will be denoted by a superscript $j$, e.g., $D(x, y)^j$.

*E.g.*, for a free scalar Klein-Gordon field the Euclidean action is of the form

$$S_E[\phi] = \frac{1}{2}\int d^4x\, \phi(x)(-\partial^2 + m^2)\phi(x) + \int d^4x\, \mathcal{L}_{int}\,, \tag{3.30}$$

where $-\partial^2 = -(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial t^2})$. For a self-interacting scalar field one has

$$\mathcal{L}_{int} = \frac{\lambda_3}{3!}\phi^3 + \frac{\lambda_4}{4!}\phi^4 + \dots \tag{3.31}$$

Sometimes it is useful to transform from coordinate to momentum-dependent fields

$$\phi(p) = \int d^d x e^{-ipx}\phi(x)\,, \tag{3.32}$$

where we have generalised from 4 to $d$ dimensions.

**Voluntary exercise:** Show by using the inverse transformation

$$\phi(x) = \int \frac{d^dp}{(2\pi)^d} e^{ipx}\phi(p) \tag{3.33}$$

that the action (3.30) is written in momentum space as

$$\begin{aligned} S[\phi] =& \frac{1}{2}\int \frac{d^dp}{(2\pi)^d}\phi(p)(p^2+m^2)\phi(-p) + \frac{\lambda_3}{3!}\int \frac{d^dp d^dq}{(2\pi)^{2d}}\phi(p)\phi(q)\phi(-p-q) \\ &+ \frac{\lambda_4}{4!}\int \frac{d^dpd^dqd^dk}{(2\pi)^{3d}}\phi(p)\phi(q)\phi(k)\phi(-p-q-k) + \dots . \end{aligned} \tag{3.34}$$

## 3.4 Dyson-Schwinger equations

Dyson-Schwinger equations (DSEs) are the equations of motion of correlation functions. They can be derived in various forms, e.g., as differential equations or as integral equations. Here the latter form will be derived from the functional integral.

The starting equation is the integral of a total derivative, which vanishes:

$$0 = \int \mathcal{D}\phi \frac{\delta}{\delta\phi(x)} e^{-S+\int dy\phi(y)J(y)} = \int \mathcal{D}\phi \left(-\frac{\delta S}{\delta\phi(x)} + J(x)\right) e^{-S+\int dy\phi(y)J(y)} \tag{3.35}$$

We can pull the source outside the integral. We can also do this with the derivative of the action, if we replace all fields by derivatives with respect to the sources:

$$0 = \left(-\left.\frac{\delta S}{\delta\phi(x)}\right|_{\phi(x')=\delta/\delta J(x')} + J(x)\right) Z[J]. \tag{3.36}$$

This is the master equation for full correlation functions.

Employing further derivatives with respect to sources yields the DSEs of the full Green functions, but our goal are the DSEs for the vertex functions, i.e., the correlation functions generated by the effective action $\Gamma$. So instead of applying further derivatives, we continue by substituting $Z[J]$ with $e^{W[J]}$ and using

$$e^{-W[J]}\left(\frac{\delta}{\delta J(x)}\right)e^{W[J]} = \frac{\delta W[J]}{\delta J(x)} + \frac{\delta}{\delta J(x)}. \tag{3.37}$$

After multiplication of Eq. (3.36) from the left with $e^{-W[J]}$ we find

$$-\left.\frac{\delta S}{\delta\phi(x)}\right|_{\phi(x')=\frac{\delta W[J]}{\delta J(x')}+\frac{\delta}{\delta J(x')}} + J(x) = 0. \tag{3.38}$$

This is the generating DSE for connected correlation functions. The DSEs of connected Green functions are obtained by acting with further source derivatives on Eq. (3.38). To get the corresponding version for vertex functions we perform the Legendre transformation of $W[J]$ with respect to the source $J$. Thereby, $\delta W[J]/\delta J(x)$ changes to $\Phi(x)$ and $\delta/\delta J(x)$ becomes

$$\begin{aligned} \frac{\delta}{\delta J(x)} &= \int dz \frac{\delta\Phi(z)}{\delta J(x)}\frac{\delta}{\delta\Phi(z)} = \int dz \frac{\delta}{\delta J(x)}\frac{\delta W[J]}{\delta J(z)}\frac{\delta}{\delta\Phi(z)} \\ &= \int dz \frac{\delta^2 W[J]}{\delta J(x)\delta J(z)}\frac{\delta}{\delta\Phi(z)} = \int dz D(x,z)^J \frac{\delta}{\delta\Phi(z)}, \end{aligned} \tag{3.39}$$

where $D(x,z)^J = \frac{\delta^2 W[J]}{\delta J(x)\delta J(z)}$ is the connected two-point function in the presence of the source $J(x)$. This yields

$$-\left.\frac{\delta S}{\delta\phi(x)}\right|_{\phi(x')=\Phi(x')+\int dz D(x',z)^J\,\delta/\delta\Phi(z)} + \frac{\delta\Gamma[\Phi]}{\delta\Phi(x)} = 0, \tag{3.40}$$

which is the master equation for vertex correlation functions. All DSEs for vertex Green functions can be derived from it by further differentiations with respect to the fields.

**Worked-out example: Scalar field with self-interactions**

The use of this master formula is exemplified for the scalar theory from (3.30). For making the respective combinatorics evident it is advantageous to express the action as

$$\begin{aligned} S[\phi] = &\frac{1}{2}\int dxdy\phi(x)S^{(2)}(x,y)\phi(y) - \frac{1}{3!}\int dxdydzS^{(3)}(x,y,z)\phi(x)\phi(y)\phi(z)- \\ &-\frac{1}{4!}\int dxdydzduS^{(4)}(x,y,z,u)\phi(x)\phi(y)\phi(z)\phi(u). \end{aligned}\tag{3.41}$$

The minus signs follow from the convention for vertices. The bare two-point function is given by

$$S^{(2)}(x,y) = \left.\frac{\delta^2 S}{\delta\phi(x)\delta\phi(y)}\right|_{\phi=0} = \delta(x-y)(-\partial^2+m^2) \tag{3.42}$$

and the bare vertices by

$$S^{(3)}(x,y,z) = -\left.\frac{\delta^3 S}{\delta\phi(x)\delta\phi(y)\delta\phi(z)}\right|_{\phi=0} = \lambda_3\delta(x-y)\delta(x-z), \tag{3.43}$$

$$S^{(4)}(x,y,z,u) = -\left.\frac{\delta^4 S}{\delta\phi(x)\delta\phi(y)\delta\phi(z)\delta\phi(u)}\right|_{\phi=0} = \lambda_4\delta(x-y)\delta(x-z)\delta(x-u). \tag{3.44}$$

The first term in Eq. (3.40) then becomes

$$\begin{aligned} &\left.\frac{\delta S}{\delta\phi(x)}\right|_{\phi(x')=\Phi(x')+\int dz D(x',z)^J\,\delta/\delta\Phi(z)} = \\ &= \Bigg(\int duS^{(2)}(x,u)\phi(u) - \int dudv\frac{1}{2}S^{(3)}(x,u,v)\phi(u)\phi(v)- \\ &\quad -\frac{1}{3!}\int dudvdwS^{(4)}(x,u,v,w)\phi(u)\phi(v)\phi(w)\Bigg)\Bigg|_{\phi(x')=\Phi(x')+\int dz D(x',z)^J\,\delta/\delta\Phi(z)} = \\ &= \int duS^{(2)}(x,u)\Phi(u)- \\ &\quad -\frac{1}{2}\int dudvS^{(3)}(x,u,v)\left(\Phi(u)+\int dzD(u,z)^J\,\delta/\delta\Phi(z)\right)\Phi(v)- \\ &\quad -\frac{1}{3!}\int dudvdwS^{(4)}(x,u,v,w)\left(\Phi(u)+\int dzD(u,z)^J\,\delta/\delta\Phi(z)\right) \\ &\qquad \times\left(\Phi(v)+\int dyD(v,y)^J\,\delta/\delta\Phi(y)\right)\Phi(w). \end{aligned}\tag{3.45}$$

The derivatives act on field and propagators as follows:

$$\frac{\delta}{\delta\Phi(y)}\Phi(x) = \delta(x-y), \tag{3.46a}$$

$$\begin{aligned}
\frac{\delta}{\delta\Phi(x)}D(y,z)^J &= \frac{\delta}{\delta\Phi(x)}\left(\frac{\delta^2\Gamma[\Phi]}{\delta\Phi(y)\delta\Phi(z)}\right)^{-1} = \\
&= -\int dz_1 dz_2 \left(\frac{\delta^2\Gamma[\Phi]}{\delta\Phi(y)\delta\Phi(z_1)}\right)^{-1} \\
&\quad\times\left(\frac{\delta^3\Gamma[\Phi]}{\delta\Phi(z_1)\delta\Phi(x)\delta\Phi(z_2)}\right)\left(\frac{\delta^2\Gamma[\Phi]}{\delta\Phi(z_2)\delta\Phi(z)}\right)^{-1} = \\
&= \int dz_1 dz_2 D(y,z_1)^J \Gamma^{(3)}(z_1,x,z_2)^J D(z_2,z)^J,
\end{aligned} \tag{3.46b}$$

$$\frac{\delta}{\delta\Phi(x)}\Gamma^{(n)}(y_1,\ldots,y_n)^J = -\frac{\delta\Gamma[\Phi]}{\delta\Phi(x)\delta\Phi(y_1)\cdots\delta\Phi(y_n)} = \Gamma^{(n+1)}(x,y_1,\ldots,y_n)^J. \tag{3.46c}$$

For completeness the derivative of an n-point function was included. The second relation follows from the matrix relation

$$\delta(MM^{-1}) = 0 \qquad \Rightarrow \qquad \delta M^{-1} = -M^{-1}(\delta M)M^{-1}. \tag{3.47}$$

We can use eqs. (3.46) in Eq. (3.45) and get

$$\begin{aligned}
&\left.\frac{\delta S}{\delta\phi(x)}\right|_{\phi(x')=\Phi(x')+\int dz D(x',z)^J\,\delta/\delta\Phi(z)} = \\
&= \int du S^{(2)}(x,u)\Phi(u) - \\
&\quad -\frac{1}{2}\int du dv S^{(3)}(x,u,v)\left(\Phi(u)\Phi(v) + D(x,v)^J\right) - \\
&\quad -\frac{1}{3!}\int du dv dw S^{(4)}(x,u,v,w)\Bigg(\Phi(u)\Phi(v)\Phi(w) + 3\Phi(u)D(v,w)^J + \\
&\qquad + \int dz D(u,z)^J \int dv_1 dv_2 D(v,v_1)^J \Gamma^{(3)}(v_1,z,v_2)^J D(v_2,w)^J\Bigg),
\end{aligned} \tag{3.48}$$

where the symmetry of the $S^{(i)}$ under exchange of their arguments was used.
Now we can derive the DSE for the two-point function by plugging this into Eq. (3.40) and differentiating with respect to $\Phi(y)$:

$$\begin{aligned}
\Gamma^{(2)}(x,y) &= S^{(2)}(x,y) - \\
&- \int du S^{(3)}(x,y,u)\Phi(u) \\
&- \frac{1}{2}\int du dv S^{(3)}(x,u,v)\int dz_1 dz_2 D(u,z_1)^J \Gamma^{(3)}(z_1,y,z_2)^J D(z_2,v)^J
\end{aligned}$$

$$
\begin{aligned}
&-\frac{1}{2}\int dudvS^{(4)}(x,y,u,v)\Phi(u)\Phi(v) - \frac{1}{2}\int dudvS^{(4)}(x,y,u,v)D(u,v)^J -\\
&-\frac{1}{2}\int dudvdwS^{(4)}(x,u,v,w)\Phi(u)\\
&\quad\times\int dv_1dv_2D(v,v_1)^J\Gamma^{(3)}(v_1,z,v_2)^JD(v_2,w)^J-\\
&-\frac{1}{3!}\int dudvdwS^{(4)}(x,u,v,w)\int dzD(u,z)^J\\
&\quad\times\int dv_1dv_2D(v,v_1)^J\Gamma^{(4)}(y,v_1,z,v_2)^JD(v_2,w)^J-\\
&-\frac{1}{2}\int dudvdwS^{(4)}(x,u,v,w)\int dzdz_1dz_2D(u,z_1)^J\Gamma^{(3)}(z_1,y,z_2)^JD(z_2,z)^J\times\\
&\quad\times\int dv_1dv_2D(v,v_1)^J\Gamma^{(3)}(v_1,z,v_2)^JD(v_2,w)^J. \qquad (3.49)
\end{aligned}
$$

We can set the sources to zero and the first terms in the second and third lines vanish. The last step is the Fourier transformation to momentum space:

$$
\begin{aligned}
&\int dxdye^{ip(x-y)}\Gamma^{(2)}(x,y) = \int dxdye^{ip(x-y)}\int\frac{drds}{(2\pi)^{2d}}e^{-i(rx+sy)}\Gamma^{(2)}(r,s) =\\
&\int\frac{drds}{(2\pi)^{2d}}(2\pi)^d\delta(p-r)(2\pi)^d\delta(p+s)\Gamma^{(2)}(r,s) = \Gamma^{(2)}(p,-p) = D^{-1}(p) =\\
&S^{(2)}(p,-p) - \int\frac{dq}{(2\pi)^d}S^{(3)}(p,p+q,-q)D(p+q)\Gamma^{(3)}(-p,q,-p-q)D(q)-\\
&\quad-\frac{1}{2}\int\frac{dq}{(2\pi)^d}S^{(4)}(p,-p,q,-q)D(q)-\\
&\quad-\frac{1}{3!}\int\frac{dqdr}{(2\pi)^{2d}}S^{(4)}(p,q,r,-p-q-r)D(q)D(r)\\
&\quad\times\Gamma^{(4)}(-p,-q,-r,p+q+r)D(-p-q-r)-\\
&\quad-\frac{1}{2}\int\frac{dqdr}{(2\pi)^{2d}}S^{(4)}(p,-p-q,r+q,-r)D(p+q)\\
&\quad\times\Gamma^{(3)}(-r-q,r,q)D(r)\Gamma^{(3)}(-p,p+q,-q)D(q)D(r+q). \qquad (3.50)
\end{aligned}
$$

A diagrammatical representation is given in Fig. 3.1.
DSEs for higher n-point functions can be derived by applying further derivatives to Eq. (3.49), setting the sources to zero and doing the Fourier transformation.

In the case of broken symmetry, *i.e.*, if the vacuum expectation value of $\phi$ is not vanishing, setting the field values to the vacuum expectation value, $\langle\phi\rangle_{J=0} =: \overline{\Phi}$, yields an infinite series for the $n$-point vertex functions:

$$
\begin{aligned}
&-\left.\frac{\delta^n\Gamma[\Phi]}{\delta\Phi(x_1)\cdots\delta\Phi(x_n)}\right|_{J=0}\\
&=\Gamma^{(n)}(x_1,\dots,x_n)^{J=0} + \int dy\Gamma^{(n+1)}(x_1,\dots,x_n,y)^{J=0}\overline{\Phi}(y)\\
&+\quad\frac{1}{2}\int dydz\Gamma^{(n+2)}(x_1,\dots,x_n,y,z)^{J=0}\overline{\Phi}(y)\overline{\Phi}(z)+\dots\,. \qquad (3.51)
\end{aligned}
$$

Consequently, in an equation like Eq. (3.49) the terms containing fields will not vanish.

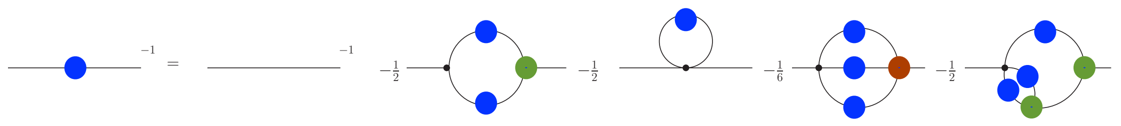


Figure 3.1: Shown is a diagrammatical representation of the DSE of the scalar two-point function Eq. (3.50). A blue circle indicates the fully dressed propagator, a line without a circle the bare propagator, a small black circle a bare vertex function, a green circle a fully dressed 3-point and a red circle a fully dressed 4-point function.
The diagrams on the right-hand side have names (from left to right): polarization, tadpole, sunset and squint diagram.

## 3.5 On the underlying structure of Dyson-Schwinger equations

For most practical purposes the derivation of DSEs is quite cumbersome. One reason is simply the growth of the number of diagrams when one proceeds to higher vertex functions. Another one is the complications introduced when several different fields and/or multi-component fields are present, because then the $n$-point functions become matrix-valid functions. A convenient way to deal with multiple fields is by using a "superfield" $\phi_i$ with one "superindex" $i$, which represents the coordinate (or momentum) dependence, all discrete indices, and the field type. In the scalar theory this field stands for $\phi(x)$, whereas in Quantum Chromo Dynamics (QCD) in a linear covariant gauge it represents the set of gluon, ghost and quark fields: $\{A^a_\mu(x), c^a(x), \bar{c}^a(x), q^j(x), \bar{q}^j(x)\}$, see the next subsection. Repeated indices are summed and integrated over as appropriate, i.e., this includes also a sum over all fields.

The usefulness of this notation becomes apparent in the relation between the propagators and two-point functions for non-vanishing external sources ($\Phi_i$ is the averaged field, $\langle\phi_i\rangle$):

$$\delta_{ij} = \frac{\delta\Phi_i}{\delta\Phi_j} = \frac{\delta J_k}{\delta\Phi_j}\frac{\delta\Phi_i}{\delta J_k} = \frac{\delta\Gamma[\Phi]}{\delta\Phi_j\delta\Phi_k}\frac{\delta W[J]}{\delta J_k\delta J_i} = \Gamma^J_{jk}D^J_{ki}. \tag{3.52}$$

The index $k$ includes a summation over all fields. The importance of this can be seen in the generalization of Eq. (3.46b):

$$\frac{\delta}{\delta\Phi_i}D^J_{jk} = \frac{\delta}{\delta\Phi_i}\left(\frac{\delta^2\Gamma[\Phi]}{\delta\Phi_j\delta\Phi_k}\right)^{-1} =$$
$$= -\left(\frac{\delta^2\Gamma[\Phi]}{\delta\Phi_j\delta\Phi_m}\right)^{-1}\left(\frac{\delta^3\Gamma[\Phi]}{\delta\Phi_m\delta\Phi_i\delta\Phi_n}\right)\left(\frac{\delta^2\Gamma[\Phi]}{\delta\Phi_n\delta\Phi_k}\right)^{-1} = D^J_{jm}\Gamma^J_{min}D^J_{nk}. \tag{3.53}$$

It means that the derivative of a propagator consists of a sum of three-point functions to which two propagators are attached. Graphically this is depicted in Fig. 3.2, where the superfield is represented by double lines. There also the following differentiation rules are shown:

$$\frac{\delta}{\delta\Phi_i}\Phi_j = \delta_{ij}, \tag{3.54}$$

$$\frac{\delta}{\delta\Phi_i}\Gamma^J_{j_1\cdots j_n} = -\frac{\delta^{n+1}\Gamma[\Phi]}{\delta\Phi_i\delta\Phi_{j_1}\cdots\delta\Phi_{j_n}} = \Gamma^J_{ij_1\cdots j_n}. \tag{3.55}$$

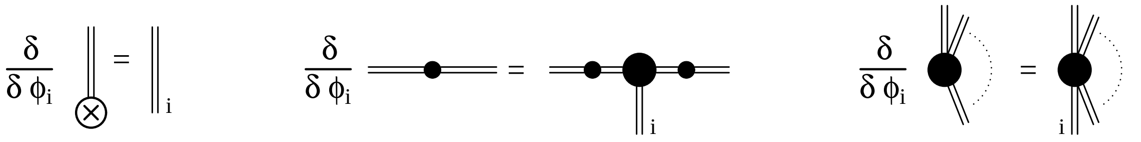


Figure 3.2: Diagrammatic rules for differentiating an external field, a propagator or a vertex. The circle with the cross denotes an external field, small blobs denote dressed propagators, and big blobs dressed 1PI vertices. The double line represents the super-field $\Phi$.

The starting equation for the DSEs of 1PI functions reads

$$-\frac{\delta S}{\delta\phi_i}\bigg|_{\phi_i=\Phi_i+D^J_{ij}\,\delta/\delta\Phi_j} + \frac{\delta\Gamma[\Phi]}{\delta\Phi_i} = 0, \tag{3.56}$$

It is helpful to have this equation worked out for generic fields, that means we just use the super field for the derivatives. Doing so we take into account all possibilities and can use this as starting point for individual cases. This equation is depicted in Fig. 3.3. The ensuing two-point DSE is shown in Fig. 3.4.

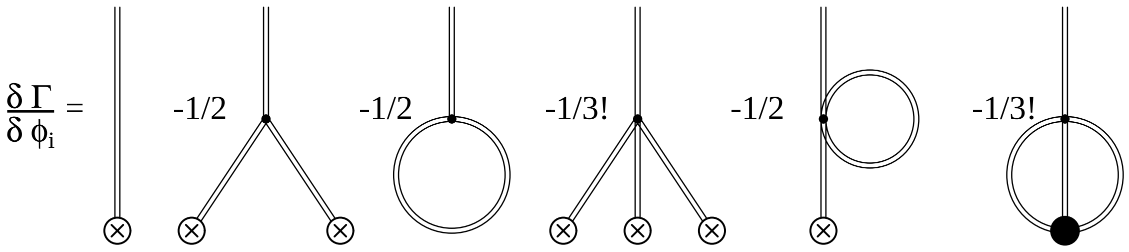


Figure 3.3: The DSE for 1PI functions. Crosses in circles denote external fields. All internal propagators are dressed and the big blob denotes a dressed 1PI vertex function.

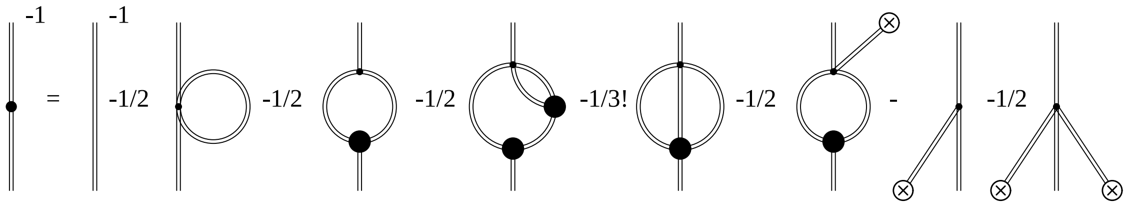


Figure 3.4: The DSE for a generic two-point function, sources not set to zero yet.

## 3.6 QCD in linear covariant gauges and in the Landau gauge

QCD is a non-Abelian gauge theory build on the structure group SU(3) with Dirac fermions in the fundamental representation (quarks) coupling to the gauge fields (gluons) in the adjoint representation of the Lie algebra su(3), respectively. The corresponding charges are called colours: red, green and blue for the quarks as well as three anti-colours for the antiquarks. Gluons appear in the eight traceless colour combinations. As QCD will be treated in detail in the lecture *Quantum Field Theory 2: Gauge Fields* only some basic features will be introduced here.

The Lagrangian density of QCD consists of two separately gauge-invariant terms:

$$\mathcal{L}_{YM} = \frac{1}{2}\,\mathrm{tr}_C\left(F_{\mu\nu}F_{\mu\nu}\right)\,, \tag{3.57}$$

$$\mathcal{L}_{QCD} = \mathcal{L}_{YM} + \sum_f \bar{q^{(f)}}(-i\not{D} + m^{(f)})q^{(f)}\,, \tag{3.58}$$

wherein the first term, $\mathcal{L}_{YM}$, encodes the dynamics of the gauge fields $A_\mu$, i.e., the gluons, and the second term in $\mathcal{L}_{QCD}$ describes the propagation of the quarks and their interaction with gluons via the covariant derivative $D_\mu$:

$$D_\mu = \partial_\mu + i\,g\,A_\mu\,. \tag{3.59}$$

Hereby, we consider several flavours of quarks, from the six flavours of the standard model, $f = u, d, s, c, b, t$, we will consider here only the lightest ones, $u$ and $d$. The masses $m^{(f)}$ originate from the coupling to the Higgs field and the Higgs vacuum expectation value. Although in nature one has $m^{(u)} \neq m^{(d)}$ (within measurement errors one has $m^{(d)} = 2m^{(u)}$) we will work in this course in the so-called isospin limit $m^{(u)} = m^{(d)} = m$. We note here already that the so-called chiral limit $m \to 0$ is of special theoretical interest.

The matrix-valued quantity $F_{\mu\nu}$ is called the field strength tensor, and it is given by

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu + i\,g\,[A_\mu, A_\nu] \tag{3.60}$$

for the matrix-valued gauge field $A_\mu$ being elements of the adjoint representation of the Lie algebra su(3). Correspondingly, $\mathrm{tr}_C$ denotes the trace over the colour indices.

The Lie algebra su(3) is spanned by (hermitian) generators chosen here to be eight $3 \times 3$ matrics, $t^a_{ij}$, $a = 1 \dots 8$, $i, j, = 1, 2, 3$ (r,g,b). By definition of the structure constant $f^{abc}$ they obey the commutation relations

$$[t^a, t^b] = i f^{abc} t^c\,, \tag{3.61}$$

and we choose the standard normalisation $T_f = 1/2$ in

$$\mathrm{tr}_C(t^a t^b) = T_f \delta^{ab}\,. \tag{3.62}$$

The decomposition of the gauge field is

$$A_\mu = A^a_\mu t^a \tag{3.63}$$

and similarly for the field strength tensor:

$$F_{\mu\nu} = F^a_{\mu\nu} t^a \,, \tag{3.64}$$
$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu - g\, f^{abc} A^b_\mu A^c_\nu \,. \tag{3.65}$$

In components the Yang-Mills (YM) Lagrangian reads

$$\mathcal{L}_{YM} = \frac{1}{4} F^a_{\mu\nu} F^a_{\mu\nu} \,. \tag{3.66}$$

The covariant derivative in the fundamental representation is in this decomposition written as $D_{\mu,ij} = \delta_{ij}\partial_\mu + i\, g\, A^a_\mu t^a_{ij}$.

The formulation in terms of gauge fields allows for a formulation with local interactions only, however, it comes at the cost of including unphysical degrees of freedom. In Quantum Electrodynamics (QED) we encountered already such a situation, the photon having two physical polarisations whereas the gauge field as a vector field carries a Lorentz index taking four different values. Employing gauge fixing one can eliminate the unphysical degrees of freedom, e.g., in a linear covariant gauge the two unphysical components, the time-like and the longitudinal photon, cancel in all physical $S$-matrix elements. As QCD is in contrast to QED a non-Abelian gauge theory the corresponding mechanism is much more subtle.

NB: Gauge fixing implies the choice of a coordinate system on a geometrical structure called fibre bundle. For a gauge theory one considers the principal fibre bundle, the fibre being the structure group. The gauge fixing condition determines a sub-manifold on the principal fibre bundle, the section. In case the structure group is a compact Lie group and thus a topologically non-trivial manifold a well-defined global section is impossible. This is called the Gribov-Singer ambiguity. In this course we will ignore this problem, it will be treated in the lecture *Quantum Field Theory 2: Gauge Fields*.

In the following we will discuss gauge invariance under the simultaneous transformation of the gluon and the quark fields. The YM Lagrangian is invariant under a gauge transformation of the gluon field defined by

$$A^U_\mu(x) = U(x)\, A_\mu(x)\, U(x)^{-1} + \frac{i}{g} (\partial_\mu U(x))\, U(x)^{-1} \,, \tag{3.67}$$

where the group-valued unitary matrices $U(x) \in \mathrm{SU}(3)$ are given by

$$U(x) = e^{i\, g\, \omega(x)} \tag{3.68}$$

with $\omega(x)$ being Lie-algebra-valued gauge parameters. It is instructive to use the decomposition $\omega(x) = \omega^a(x) t^a$. The r.h.s. of eq. (3.67) can be expanded for "small" gauge parameters, in an infinitesimal form for the components $A^a_\mu$ the gauge transformation is then given by

$$A^a_\mu \to A^a_\mu + \delta A^a_\mu = A^a_\mu - \partial_\mu \omega^a - g\, f^{abc} \omega^b A^c_\mu = A^a_\mu - D^{ab}_\mu \omega^b, \tag{3.69}$$

where the covariant derivative in the adjoint representation $D^{ab}_\mu$ is defined as

$$D^{ab}_\mu = \delta^{ab}\partial_\mu + g\, f^{abc} A^c_\mu \,. \tag{3.70}$$

Due to gauge invariance the differential operator defining the tree-level gluon two-point function possesses zero modes, and therefore this operator cannot be inverted to obtain the tree-level propagator. Similar to QED one introduces via implementing a gauge fixing condition with the help of a Lagrange multiplier. Herein, we will restrict ourselves to linear covariant gauges,

$$\partial^\mu A^a_\mu = 0\,, \tag{3.71}$$

which implies a restriction of the complete gluon field configuration space to the subspace of four-dimensionally transverse gluon fields. Contrary to the case of QED (in flat spaces, resp., spacetimes) the restriction of the functional integration to this subspace implies a non-trivial Jacobi determinant, the Faddeev-Popov determined. Localising this determined by expressing it as an integral over anti-commuting (Grassmann) variables introduces negative-definite and thus unphysical fields, the Faddeev-Popov ghost and anti-ghost fields, $c^a$ and $\bar{c}^a$. Within perturbation theory it can be shown algebraically that a quartet of unphysical fields, the time-like and the longitudinal gluon, the ghost and the antighost, cancel in the $S$-matrix elements involving transverse gluons and quarks.

Implementing the gauge fixing condition implies then

$$\mathcal{L}_{YM} \to \mathcal{L}_{YM} + \mathcal{L}_{gf}. \tag{3.72}$$

with the gauge-fixing Lagrangian given by

$$\mathcal{L}_{gf} = \frac{1}{2\xi}(\partial_\mu A^a_\mu(x))^2 - \bar{c}^a(x)\,\left(-\partial^\mu D^{ab}_\mu\right)\,c^b(y)\,. \tag{3.73}$$

Hereby, the inverse of the gauge fixing parameter $\xi$ acts as a Lagrange multiplier.

The simultaneous gauge transformation on gluon and quark fields, the latter being

$$q(x) \to U(x)q(x)\,, \quad \bar{q}(x) \to \bar{q}(x)U^\dagger(x)\,, \tag{3.74}$$

leaves $\mathcal{L}_{QCD}$ invariant. (NB: For SU(N) $U^\dagger(x) = U^{-1}(x)$.)

*Voluntary exercise:* Show the gauge invariance of $\mathcal{L}_{QCD}$. Prove for $\mathcal{L}_{YM}$ and the quark term, respectively, in a first step that the gauge transformations on gluon fields imply $F_{\mu\nu} \to U(x)F_{\mu\nu}U^\dagger(x)$ and $D_\mu \to U(x)D_\mu U^\dagger(x)$.

The limit $\xi \to 0^+$ is called Landau gauge. It implements the transversality condition for gluons strictly, i.e., including the quantum fluctuations of the gluon field. Therefore, in the Landau gauge one gains the advantage that only correlation functions with transverse gluons as ‘external legs‘ need to be considered.

In the following we will assume that we made a Wick rotation at the level of the action. Phrased otherwise, instead of a quantum field theory in four-dimensional Minkowski spacetime we consider an “analytically-continued” theory employing an underlying four-dimensional Euclidean space. Besides other (wanted) effects this leads to changes for the Dirac matrices, the details of these changes will be elucidated in the following.

## 3.7 Elementary correlation functions of QCD in the Landau gauge

The Lagrangian of QCD in the Landau gauge, if ordered according to number of fields, possesses seven terms which are the (i) gluon–gluon, (ii) anti-ghost–ghost, (iii) anti-quark–quark, (iv) anti-ghost–ghost–gluon, (v) 3-gluon, (vi) 4-gluon, (vii) anti-quark–quark–gluon terms. The corresponding correlation functions are:

(i) (inverse) gluon propagator,

(ii) (inverse) ghost propagator,

(iii) (inverse) quark propagator,

(iv) ghost-gluon vertex,

(v) 3-gluon vertex,

(vi) 4-gluon vertex, and

(vii) quark-gluon vertex.

NB: Due to their special role in the renormalisation of QCD they are called the seven primitively divergent correlation functions.

In this course we will ignore all complications introduced by the so-called ghost fields. (These fields, their appearance and their role in the theory will be dealt with in the lecture *Quantum Field Theory II: Gauge Theories*.)

For this section we consider one specific flavour of quarks. It will become evident that one has a quark propagator and a quark-gluon vertex function for every quark flavour separately if one considers only the Strong Interaction, i.e., QCD.

Furthermore, as in the quark propagator DSE the 3- and 4-gluon vertex functions appear only indirectly via the gluon propagator and the quark-gluon vertex we will consider the gluon and quark propagators which are given by

$$D^{ab}_{\mu\nu}(x,y) = \left.\frac{\delta^2 W}{\delta J^a_\mu(x)\delta J^b_\nu(y)}\right|_{J=0} = \left.\left(\frac{\delta^2\Gamma}{\delta A^a_\mu(x)\delta A^b_\nu(y)}\right)^{-1}\right|_{\phi=0}, \tag{3.75}$$

$$S^{ij}(x,y) = \left.\frac{\delta^2 W}{\delta\bar{\eta}^i(x)\delta\eta^j(y)}\right|_{J=0} = \left.\left(\frac{\delta^2\Gamma}{\delta\bar{q}^i(x)q^j(y)}\right)^{-1}\right|_{\phi=0}, \tag{3.76}$$

where $J^a_\mu(x)$, $\eta^i(x)$ and $\bar{\eta}^i(x)$ are the sources for the gluon, anti-quark and quark fields, respectively, as well as the quark-gluon vertex defined by

$$\Gamma^{A\bar{q}q,aij}_\mu(x,y,z) = -\left.\frac{\delta^3\Gamma}{\delta A^a_\mu(x)\delta\bar{q}^i(y)q^j(z)}\right|_{\phi=0}. \tag{3.77}$$

To calculate the respective tree-level quantities we replace in these three relations $\Gamma$ by $S$. A (short) calculation yields:

$$\left.\frac{\delta^2 S}{\delta A^a_\mu(x)\delta A^b_\nu(y)}\right|_{\phi=0} = \delta^{ab}\Big(-g_{\mu\nu}\partial^2 + \partial_\mu\partial_\nu(1-\xi^{-1})\Big)\delta(x-y), \tag{3.78}$$

$$\left.\frac{\delta^2 S}{\delta \bar{q}^i(x) q^j(y)}\right|_{\phi=0} = \delta^{ij}\Big(-\not{\partial} + m\Big)\delta(x-y), \tag{3.79}$$

$$-\left.\frac{\delta^3 S}{\delta A^a_\mu(x)\delta \bar{q}^i(y) q^j(z)}\right|_{\phi=0} = i\,g\,t^{a,ij}\gamma_\mu \delta(x-y)\delta(x-z). \tag{3.80}$$

After a Fourier transformation, or alternatively Fourier-transforming the action and perform derivatives w.r.t. momentum-dependent fields, we obtain

$$\widetilde{\Gamma}^{(0)\,ab}_{\;\;\mu\nu}(p,q) = \delta^{ab}\left(p^2 g_{\mu\nu} - (1-\xi^{-1})p_\mu p_\nu\right)(2\pi)^4\delta(p+q), \tag{3.81}$$

$$\widetilde{\Gamma}^{(0)\,ij}(p,q) = \delta^{ij}\left(i\not{p} + m\right)(2\pi)^4\delta(p+q), \tag{3.82}$$

$$\widetilde{\Gamma}^{(0)\,a\,ij}_{\;\;\mu}(k;p,q) = i\,g\,\gamma_\mu t^a_{ij}(2\pi)^4\delta(p+q+k). \tag{3.83}$$

For the quark two-point function $p/q$ are the anti-quark/quark momenta, for the quark-gluon vertex $p/q/k$ are the anti-quark/quark/gluon momenta. All momenta are defined as incoming. As we employ the Euclidean version of the theory we have $g_{\mu\nu} = \delta_{\mu\nu}$. Note that $a, b$ are adjoint and $i, j$ fundamental colour indices. Note that $\widetilde{\Gamma}^{(0)\,ij}$ and $\widetilde{\Gamma}^{(0)\,a\,ij}_{\;\;\mu}$ are Dirac matrices, the corresponding indices are only suppressed in this (usual) notation.

In the following we split off the factors of $(2\pi)^4\delta(\cdot)$ since they will effectively drop out of functional equations like the DSEs: All except one vanish by integration, reducing the effective number of momentum integrations to the number of loops, and one overall factor, representing momentum conservation, appears in front of all expressions and can be canceled. Thus we will work with

$$\Gamma^{(0)\,ab}_{\;\;\mu\nu}(p) = \delta^{ab}\left(p^2 g_{\mu\nu} - (1-\xi^{-1})p_\mu p_\nu\right), \tag{3.84}$$

$$\Gamma^{(0)\,ij}(p) = \delta^{ij}\left(i\not{p} + m\right), \tag{3.85}$$

$$\Gamma^{(0)\,a\,ij}_{\;\;\mu}(k;p,q) = i\,g\,\gamma_\mu t^a_{ij}. \tag{3.86}$$

The tree-level propagators are then

$$D^{(0)\,ab}_{\;\;\mu\nu}(p) = \left(\Gamma^{(0)\,ab}_{\;\;\mu\nu}(p)\right)^{-1} = \delta^{ab}\left(g_{\mu\nu} - (1-\xi)\frac{p_\mu p_\nu}{p^2}\right)\frac{1}{p^2}, \tag{3.87}$$

$$S^{(0)\,ij}(p) = \left(\Gamma^{(0)\,ij}(p)\right)^{-1} = \delta^{ij}\frac{-i\not{p} + m}{p^2 + m^2}. \tag{3.88}$$

The dressed quantities have additional dressing functions encoding their deviation from the tree-level behaviour. For the propagators they only supplement the existing tensor structure, while for the quark-gluon vertex more tensor structures are possible:

$$D^{ab}_{\mu\nu}(p) = \delta^{ab}\left[\left(g_{\mu\nu} - \frac{p_\mu p_\nu}{p^2}\right)\frac{Z(p^2)}{p^2} + \xi\frac{p_\mu p_\nu}{p^4}\right], \tag{3.89}$$

$$S^{ij}(p) = \delta^{ij}\frac{-iA(p^2)\not{p} + B(p^2)}{p^2A^2(p^2)^2 + B^2(p^2)}, \tag{3.90}$$

$$\Gamma^{a\,ij}_\mu(k;p,q) = i\,g\,t^a_{ij}\sum_{i=1}^{12}\tau^i_\mu(k;p,q)h^i(k;p,q). \tag{3.91}$$

Note the splitting of the gluon propagator into a transverse part, which is characterized by the dressing function $Z(p^2)$, and a longitudinal part proportional to the gauge fixing parameter $\xi$. The latter does not receive any corrections as a consequence of gauge symmetry. (NB: A so-called Slavnov-Taylor identity can be derived from gauge symmetry that ensures this.) As already stated, in the following we will work in the Landau gauge which corresponds to setting $\xi = 0$ and retaining only the transverse part of all correlation functions.

This then reduces the 12 tensor structures of the quark-gluon vertex to eight transverse ones fulfilling $k^\mu\tau^i_\mu = 0$. Some of these tensor structures are needed to describe hadronic physics correctly. In this course we will avoid the complications which come with the related plethora of terms in a DSE kernel.

Note that the tree-level limit implies momentum-independence: $Z(p^2) \to 1$, $A(p^2) \to 1$ and $B(p^2) \to m$. The functions $Z$ and $A$ are dimensionless, the function $B$ has mass (energy) dimension one. As for the quark-gluon vertex, choosing $\tau^1_\mu = \gamma_\mu$, the tree-level limits are $h^1(k;p,q) \to 1$ and all other $h^i \to 0$.

NB: Given that we work in Landau gauge the better choice for $\tau^1_\mu$ would be the transversely projected Dirac matrix vector $\gamma^T_\mu = (g_{\mu\nu} - k_\mu k_\nu/k^2)\gamma^\nu$.

We have now all ingredients to derive the quark propagator DSE.

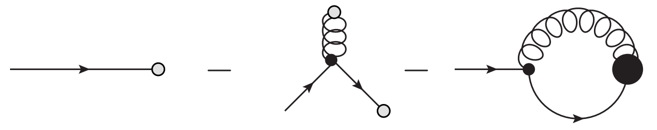
## 3.8 The quark propagator Dyson-Schwinger equation

Also here it is sufficient to consider one specific flavour of quarks, $\psi \in \{u, d, s, c, b\}$.

The weak and the Higgs-Yukawa interactions of the top quark are already so strong that a consideration within QCD alone makes no sense.

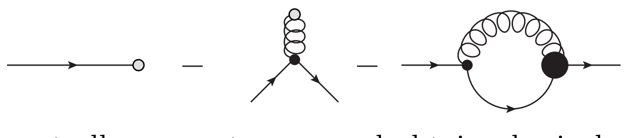
We start with the master equation where the first derivative was done w.r.t. the quark field (the gray blob denotes an external field):

$\frac{\delta\Gamma}{\delta\psi} =$



Applying the differentiation rules to perform an anti-quark field derivative, we arrive at:

$\frac{\delta^2\Gamma}{\delta\bar\psi\delta\psi} =$



We now set all sources to zero and obtain physical propagators and vertices. In diagrammatic form the quark propagator DSE is depicted in Fig. 3.5. Plugging in

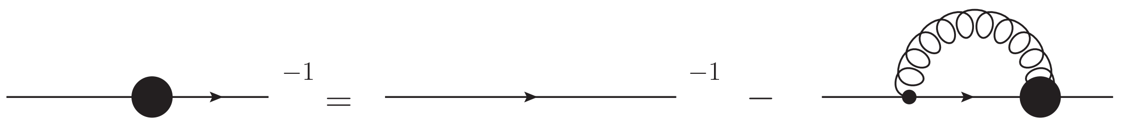


Figure 3.5: The quark propagator DSE. The thin/thick blob denotes a bare/dressed quark-gluon vertex. Internal propagators and the propagator with a blob are dressed. Continuous/Wiggled lines denote quarks/gluons.

the Feynman rules, the equation reads

$$\left(S^{ij}\right)^{-1}(p) = \left(S^{(0)\,ij}\right)^{-1} - \Sigma^{ij}(p), \tag{3.92}$$

$$\Sigma^{ij}(p) = \int \frac{d^4q}{(2\pi)^4} \Gamma^{(0)\,a\,ki}_{\ \ \mu}(k;-q,p) S^{kl}(q) \Gamma^{b\,jl}_{\nu}(-k;-p,q) D^{ab}_{\mu\nu}(k)\,, \tag{3.93}$$

with $k^\mu = q^\mu - p^\mu$. The Dirac-matrix-valued quantity $\Sigma^{ij}(p)$ plays the role of a quark self-energy. As we will see soon it will add to the kinetic term and to the mass term differently thereby verifying the general form (3.90). Using $\Gamma^{(0)\,a\,ki}_{\ \ \mu} = i\,g\,\gamma_\mu t^a_{ki}$, and making for the dressed quark-gluon vertex the (admittedly over-simplifying) ansatz

$$\Gamma^{b\,jl}_{\nu}(k;p,q) = ig\gamma_\nu t^b_{jl} \Gamma(k;p,q) \tag{3.94}$$

we can treat the colour and the Dirac indices in two steps. First, we multiply both sides of the DSE by $\delta^{ij}$, note that $D^{ab}_{\mu\nu}(k) \propto \delta^{ab}$, $S^{kl} \propto \delta^{kl}$ and use

$$\delta^{ij}\delta^{ab} t^a_{ki} t^b_{jl} \delta^{kl} = \mathrm{tr}\, t^2 = \frac{N_c^2 - 1}{2}\,. \tag{3.95}$$

From the left-hand side we obtain a factor of $\delta^{ij}\delta^{ij} = N_c$ so that in total the color factor in front of the integral is $C_F = (N_c^2 - 1)/2N_c$. This result is expected from group theory, $C_F$ being the second Casimir invariant of the fundamental representation of SU($N_c$). For $N_c = 2$, $C_F = 3/4$ ($=s(s+1)$), for the physical case $N_c = 3$ one has $C_F = 4/3$.

Using this we can write the colour-projected self-energy as

$$\Sigma(p) = -C_F\, g^2 \int \frac{d^4q}{(2\pi)^4} \gamma_\mu S(q) \gamma_\nu D_{\mu\nu}(k) \Gamma(-k;-p,q). \tag{3.96}$$

In a next step we take care about the Dirac-matrix properties. To this end we note that the quark propagator $S(q) \propto -iA(q^2)\not{q} + B(q^2)$, i.e., we expect one term proportional to $q^\rho\gamma_\rho$ and one proportional to the unit matrix. Starting with the latter and noting that $D_{\mu\nu}(k) \propto \left(g_{\mu\nu} - \frac{k_\mu k_\nu}{k^2}\right)$ the term of interest is

$$\gamma_\mu B(q^2) \gamma_\nu \left(g_{\mu\nu} - \frac{k_\mu k_\nu}{k^2}\right)\,. \tag{3.97}$$

Using $\mathrm{tr}\,\gamma_\mu\gamma_\nu = 4g_{\mu\nu}$ and $g_{\mu\nu}\left(g_{\mu\nu} - \frac{k_\mu k_\nu}{k^2}\right) = 3$ gives then together with $\mathrm{tr}\,S^{-1}(p) = 4B(p^2)$ as well as $\mathrm{tr}\,(S^{(0)})^{-1}(p) = 4m$ an equation for $B(p^2)$ from the trace of the DSE, see below.

However, as $\mathrm{tr}\,\gamma_\mu \not{q} \gamma_\nu = q^\rho \mathrm{tr}\,\gamma_\mu \gamma_\rho \gamma_\nu = 0$, how does one get an equation for $A$? To this end we multiply the DSE on both sides with $-i\not{p}/p^2$ and trace then the resulting equation. Under the integral then the term with four Dirac matrices will deliver the one contribution, and using $\mathrm{tr}\,\gamma_\mu\gamma_\nu\gamma_\rho\gamma_\sigma = 4(g_{\mu\nu}g_{\rho\sigma} - g_{\mu\rho}g_{\nu\sigma} + g_{\mu\sigma}g_{\nu\rho})$ some more algebra is needed to arrive at the equations

$$A(p^2) = 1 + \Sigma_A(p^2), \tag{3.98}$$

$$B(p^2) = m + \Sigma_B(p^2), \tag{3.99}$$

$$\Sigma_A(p^2) = \tag{3.100}$$
$$\frac{C_F\, g^2}{p^2} \int \frac{d^4q}{(2\pi)^4} \frac{A(q^2)}{q^2A^2(q^2) + B^2(q^2)} \left(p\cdot q + 2\frac{k\cdot p\, k\cdot q}{k^2}\right) \frac{Z(k^2)}{k^2}\Gamma(-k;-p,q),$$

$$\Sigma_B(p^2) = 3C_F\, g^2 \int \frac{d^4q}{(2\pi)^4} \frac{B(q^2)}{q^2A^2(q^2) + B^2(q^2)} \frac{Z(k^2)}{k^2}\Gamma(-k;-p,q) \tag{3.101}$$

with $k = q - p$.

*Voluntary exercise:* Derive the expressions for the selfenergies $\Sigma_A(p^2)$ and $\Sigma_B(p^2)$.

Before we can solve these two coupled equations, we need to specify the gluon dressing function $Z(k^2)$ and the quark-gluon vertex dressing $\Gamma(-k;-p,q)$. We will combine these two quantities into an effective interaction:

$$g^2\frac{Z(k^2)}{k^2}\Gamma(-k;-p,q) \to 4\pi^2 D \frac{k^2}{\omega^2} e^{-k^2/\omega^2} \tag{3.102}$$

which constitutes then the model to be employed in the numerical study of the quark DSE. Hereby, $D$ parameterises the strength of the interaction, and $\omega$ sets the scale. Note that $D$ is of mass dimension -2. Later on we will employ a division of all dimension-full quantities by $\omega$ to arrive at equations containing only dimension-less quantities. In particular, the dimension-less combination $D\omega^2$ determines the strength of our model interaction.

Using also $C_F = 4/3$ we have then

$$A(p^2) = 1 + \frac{16\pi^2 D}{3p^2} \int \frac{d^4q}{(2\pi)^4} \frac{A(q^2)}{q^2A^2(q^2) + B^2(q^2)} \left(p\cdot q + 2\frac{k\cdot p\, k\cdot q}{k^2}\right) \frac{k^2}{\omega^2} e^{-k^2/\omega^2}, \tag{3.103}$$

$$B(p^2) = m + 16\pi^2 D \int \frac{d^4q}{(2\pi)^4} \frac{B(q^2)}{q^2A^2(q^2) + B^2(q^2)} \frac{k^2}{\omega^2} e^{-k^2/\omega^2}, \tag{3.104}$$

with $k = q - p$.

In a next step we express $k^2$, $k\cdot p$ and $k\cdot q$ in terms of $p^2$, $q^2$ and $z = p\cdot q/\sqrt{p^2q^2}$:
With $|p| = \sqrt{p^2}$ and $|q| = \sqrt{q^2}$ one has
$k^2 = p^2 + q^2 - 2|p|\,|q|z$ and
$k^2 p\cdot q + 2k\cdot p\, k\cdot q = -2p^2q^2 + 3(p^2+q^2)|p|\,|q|z - 4p^2q^2z^2$ .

The four-dimensional integral measure will be taken in hyperspherical coordinates:

$$q = |q| \begin{pmatrix} \sin\theta_2 \sin\theta_1 \cos\phi \\ \sin\theta_2 \sin\theta_1 \sin\phi \\ \sin\theta_2 \cos\theta_1 \\ \cos\theta_2 \end{pmatrix}, \quad d^4q = \frac{dq^2}{2} q^2 \, d\phi \, d\theta_1 \, \sin\theta_1 \, d\theta_2 \, \sin\theta_2^2 \, , \tag{3.105}$$

in which $dq\,|q|^3 = \frac{1}{2} dq^2\, q^2$ has been used.

The external momentum is put in the four-direction:

$$p = |p| \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} . \tag{3.106}$$

Then the integrals over $\phi$ and $\theta_1$ can be performed analytically. The final integral measure is then

$$\int d^4q = 4\pi \int_0^\infty \frac{dq^2 q^2}{2} \int_0^\pi d\theta_2 \sin^2\theta_2 = 4\pi \int_0^\infty \frac{dq^2 q^2}{2} \int_{-1}^1 dz \sqrt{1-z^2} \, , \tag{3.107}$$

with $z = \cos\theta_2$. The integral over $\theta_2$ can be performed in case of our simple model analytically.

It is advantageous to introduce dimensionless variables $x = p^2/\omega^2$ and $y = q^2/\omega^2$ (including $dq^2 = \omega^2 dy$). Rewriting the integral equations into ones for $A(x)$ and $B(x)$ we arrive at the form we will solve numerically.

*Voluntary exercise:* Derive the explicit form of the integral equations for $A(x)$ and $B(x)$.
Solution: See Exercise 3 below.

In practice, we proceed as follows:

- Set starting values for the dressing functions $A(x$ and $B(x)$.
- Iterate the equation:
  1. Calculate the integrals (i.e., the self-energies).
  2. Calculate the dressing functions.
  3. Calculate from point-wise difference to the previous dressing functions a quantity to be used in a convergence criterion.
  4. Go to 1. until the convergence criterion is fulfilled.

The quark propagator DSE is not very sensitive to the choice of starting values. Using constant starting values, *e*.g., $A(x) = 1$ and $B(x) = 0.01$ is sufficient.

The convergence criterion should determine when the iteration does not change the dressing functions on a noticeable level. For example, the sum of absolute differences or the sum of relative differences should be below a given value, *e*.g., $10^{-8}$.

**Exercise 3:**

Consider the coupled two integral equations for the quark propagator functions $A(p^2)$ and $B(p^2)$, resp., $A(x)$ and $B(x)$, derived above:

$$\begin{aligned} A(x) &= 1 + D\omega^2 \int_0^\infty \frac{dy\, y\, A(y)}{(yA^2(y) + B^2(y)/\omega^2)} \\ &\times \frac{2}{\pi} \int_{-1}^{1} dz \sqrt{1-z^2} \left[ -\frac{2}{3} y + \left(1 + \frac{y}{x}\right) \sqrt{xy}\, z - \frac{4}{3} y z^2 \right] e^{-\left(x+y-2\sqrt{xy}\, z\right)}, \end{aligned} \tag{3.108}$$

$$\begin{aligned} B(x) &= m + D\omega^2 \int_0^\infty \frac{dy\, y\, B(y)}{(yA^2(y) + B^2(y)/\omega^2)} \\ &\times \frac{2}{\pi} \int_{-1}^{1} dz \sqrt{1-z^2} \left[ x + y - 2\sqrt{xy}\, z \right] e^{-\left(x+y-2\sqrt{xy}\, z\right)}, \end{aligned} \tag{3.109}$$

where $x = p^2/\omega^2$, $y = q^2/\omega^2$ and $z = p \cdot q / \sqrt{p^2 q^2}$.

a.) In order to use them in bound-state equations these functions needs to be analytically continued in the external variable $x$. At first sight, it appears that $A(x)$ and $B(x)$ could not be consistently continued because the cut along the negative $x$–axis (associated with $\sqrt{x}$) would yield different results when continuing in the upper or the lower half–plane, and it would be impossible to resolve the ambiguity in $\sqrt{x} \to \pm i\sqrt{\xi}$ when continuing $x \to -\xi$.
Explain why this is not a problem due to a specific symmetry of the integrands in eqs. (3.108) and (3.109).

b.) Calculate analytically in eqs. (3.108) and (3.109) the angular ($z$) integrals for real and positive $x$.

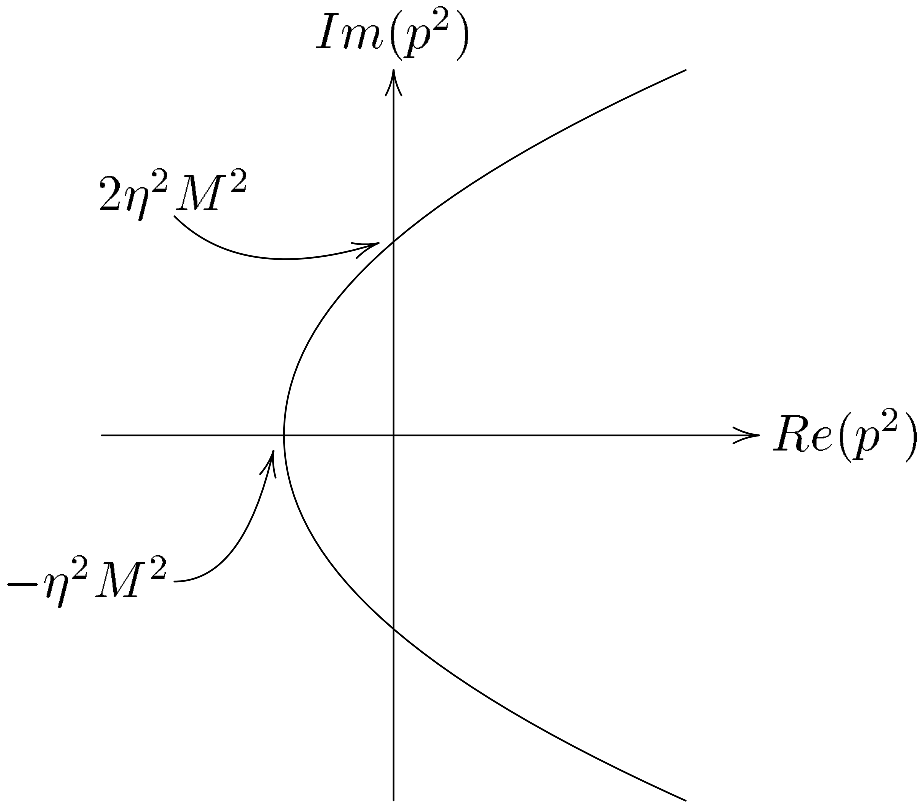
c.) Write a program to calculate the modified Bessel functions $I_n(x)$.

d.) Derive the one-dimensional integral equations for $A(x)$ and $B(x)$ by using the result of b.
Provide an argument why one should not use a subroutine or function for the modified Bessel functions $I_n(x)$ but for exponentially rescaled functions $\mathcal{I}_n(x) = \exp(-|x|) I_n(x)$ instead to solve the integral equations with a high precision on the real positive half-axis.

**Exercise 4:**

Solve numerically the integral equations for $A$ and $B$ on the real positive half-axis by iteration for $\omega = 0.5$ GeV, $D = 16$ GeV$^{-2}$ and $m_0 = 0$, 0.005 and 0.115 GeV. NB: Use a suitable variable transformation and numerical integration routine from the ones discussed in Chapter 2.

Calculate from the integral equations $A$ and $B$ for the below shown parabola-bounded region in the complex plane for $\eta = 1/2$ and $M \approx B(0)/10$.



Hint:
In Fig. 3.6 the quark propagator functions $A(p^2)$ and $B(p^2)$ as well as the mass function $M(p^2) = B(p^2)/A(p^2)$ along the positive real space-like axis, $p^2 > 0$, are shown.

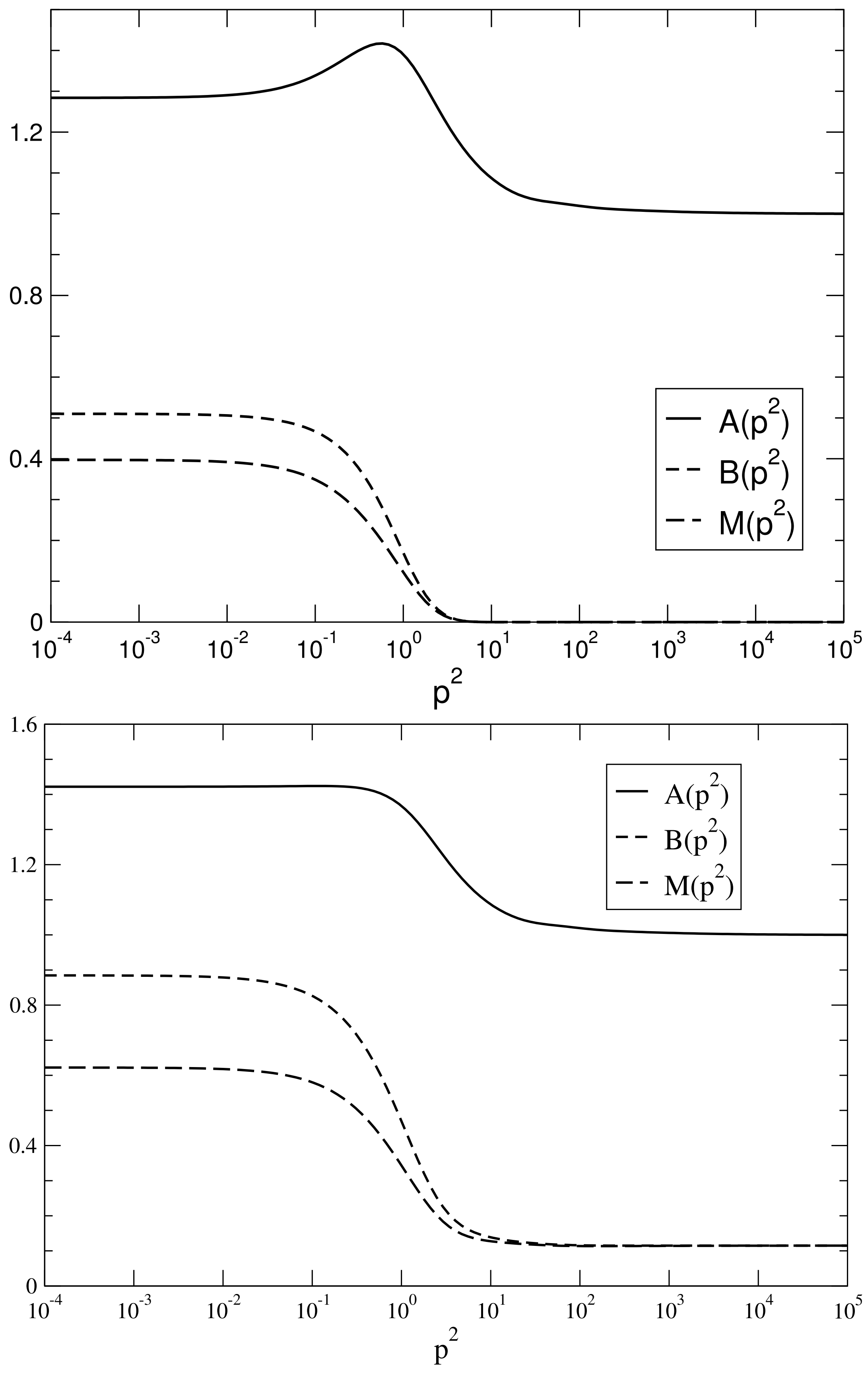


Figure 3.6: Plot of the space-like quark propagator functions as a function of the momentum squared. The parameters are $\omega = 0.5\text{GeV}$, $D = 16.0\text{GeV}^{-2}$. Upper panel: $m_0 = 0$, lower panel: $m_0 = 0.115$ GeV. All units are $GeV$. [RA et al., Phys. Rev. **D**65 (2002) 094026 [arXiv:hep-ph/0202053]].

# 4 Bound state equations

Two-body bound states in non-relativistic classical and quantum mechanics are typically treated by transforming to center-of-mass and relative coordinates as well as to the respective momenta, the total momentum and the relative momentum. The center-of-mass motion is in most cases of interest free, and the relative motion is mapped to the motion of one particle in a central potential with the mass being the reduced mass calculated from the masses of the two particles.

Prominent examples are Kepler's orbits of planetary motion, or the solution of the Schrödinger equation for the hydrogen atom with then relativistic and field-theoretical effects treated in perturbation theory.

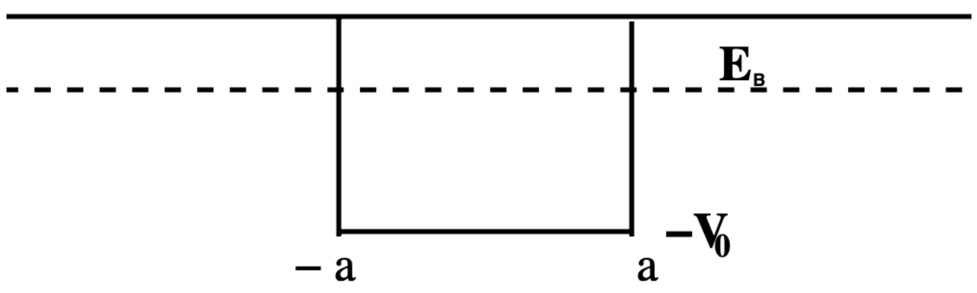
This kind of investigating bound states is restricted to cases in which (i) the binding energy is much smaller than the constituents' masses and (ii) no decay or annihilation channel for the constituents is open. Phrased otherwise, (i) a deeply bound state with $E_B = \mathcal{O}(m)$ is substantially influenced by particle–anti-particle pair components in its quantum amplitude and (ii) reflects the fact that positronium or muonic hydrogen behaves qualitatively different than the usual hydrogen atom.

Therefore, another type than a (single-particle) equation like the Schrödinger or Dirac equation is needed to describe bound states with the properties (i) and/or (ii) correctly. In particular, retardation effects and virtual particles in the two-particle interaction will become important and will prevent a mapping to a potential problem.

NB: Already the three-body problem in Classical / Quantum Mechanics is in general very challenging and requires sophisticated techniques.

## 4.1 Bound states and poles of transmission amplitudes in Quantum Mechanics

"Simplest" quantum mechanical bound state problem: Consider one-dimensional Schrödinger equation for potential well.



Such a potential has finitely many (but at least one) bound states with discrete energy levels $E_n^B < 0$. The corresponding conditions for a bound state read

$$\tan qa = \frac{\kappa}{q} \quad \text{or} - \cot qa = \frac{\kappa}{q} \,,$$

with $\kappa = \sqrt{-2mE_n^B}$ and $q = \sqrt{2m(E_n^B + V_0)}$.

On the other hand, there are infinitely many scattering states with $E > 0$. The scattering off such a potential is most efficiently described by the transmission amplitudes $S(E)$ the absolute square of which describes the probability to detect a particle on the opposite side of an incident (normalised) wave-packet, respectively, a plane wave with wave number $k = \sqrt{2mE}$.

The transmission amplitudes $S(E)$, respectively, $S(k)$, can be analytically continued to complex values, in particular to $k = \pm i\kappa$. The denominator of $S$ vanishes if

$$\tan qa = \frac{-ik}{q} \quad \text{or} \cot qa = \frac{ik}{q} \,,$$

Phrased otherwise, these amplitudes $S$ possesses poles at the locations of the bound state energies, $E = -E_n^B$. This is easy to understand on physical grounds: At these energies one has a non-vanishing wave function without an incident wave.

[See, *e.g.*, the discussion in F. Schwabl, QM, Sect. 3.7.1.]

This can be generalised to all Quantum Theories including Quantum Field Theory in four-dimensional spacetime. Clearly, the role of the transmission amplitudes $S(E)$ is taken by the $S$-matrix. Following LSZ scattering theory, it is evident that such poles can only originate from the respective correlation functions which are, mathematically, Green functions. As we know already from the elementary propagators, a pole in such a Green function correspond to a propagating state and may then represent a bound state.

NB: This propagating state may or may not belong to the physical asymptotic state space. However, we will treat one problem at a time.

Although we have now a consistent ansatz for the search of a bound-state equation there are clearly further considerations to be taken into account. First, bound state properties can only be determined from non-perturbative calculations: Any finite polynomial in the coupling, even if multiplied with loop integrals, will not lead to the searched change of the analytic structure for the tree-level function. In general, existing poles will be shifted but no new ones will be created. At least, an infinite (re-)summation is needed. Also this is known from Quantum Mechanics: One can get the Born series from the Lippmann-Schwinger equation (i.e., the integral equation form of the Schrödinger equation) by an expansion, to obtain the Lippmann-Schwinger equation back one needs to (re-)sum infinitely many terms.

For a two-body bound state one needs to consider a four-point function reflecting the 2$\to$2 scattering process, for a three-body bound state the six-point function, etc.. In this course we will restrict to a $\bar{q}q$ bound state, and thus we consider the four-point function with a $\bar{q}q$ pair with momenta $p_1$ and $p_2$ in and a $\bar{q}q$ pair with momenta $p_3$ and $p_4$ out. In Quantum Mechanics we considered the energy and/or the momentum as variable, in the context of relativistic Quantum Field Theory it is, of course, advantageous to use Lorentz-invariant variables as, *e*.g., the Mandelstam variables

$$\begin{aligned} s &= (p_1+p_2)^2 = (p_3+p_4)^2\,, && (4.1)\\ t &= (p_1+p_3)^2\,, && (4.2)\\ u &= (p_1+p4)^2\,. && (4.3)\end{aligned}$$

Given the way we set up the problem the task is now to search for a pole or poles of the four-point function in $s$.

As we will see, the result of the Dyson-Schwinger equation will enter which, however, was solved after performing a Wick rotation to Euclidean space. However, the bound state will appear for physical momenta. In our notation this implies that there exist a reference frame for which $p_1 + p_2 =: P = (iM_B, \vec{0})$ for the bound state with mass $M_B$. The pole of the four-point function $G^{(4)}$ is then located at

$$P^2 = s = -M_B^2\,. \tag{4.4}$$

It is obvious that this pole has to appear in the interaction part of $G^{(4)}$ as the non-interacting part is given by

$$G_0^{(4)}(p_1,p_2,p_3,p_4) = S(p_1)\delta(p_1-p_4)S(p_2)\delta(p_2-p_3)$$

and will this reflect poles only at the poles of the quark propagators $S(p_i)$.

As mentioned above we will have to sum a single interaction part of $G^{(4)}$ infinitely often. A self-consistent solution of an integral equation amounts to a re-summation, typically in the form of a geometric series, $1 + x + x^2 + \ldots = \frac{1}{1-x}$ which also nicely demonstrates the emergence of a pole, in this case in the limit $x \to 1$.

NB: Until approximately 20 years ago the numerical solution of a non-perturbative four-point function was out of reach, and for the first such calculations based on a Dyson-Schwinger approach supercomputers were used, mostly for the memory (RAM) requirements. With a today's desktop such a calculation (e.g., within a master thesis) is straightforwardly feasible.

## 4.2 Derivation of the meson Bethe-Salpeter equation

Within functional approaches several ways of phrasing a bound state equation are known. In this course we will restrict ourselves to the "classic" relativistic bound state equation derived by Bether and Salpeter in 1951. To this end one exploits our pre-knowledge and considers the four-point function only at or close to the assumed pole at $P^2 = -m_B^2$. The generic structure of the four-point function will be of type

$$G^{(4)} = G_0^{(4)} + G_0^{(4)} T^{(4)} G_0^{(4)} \tag{4.5}$$

where $G_0^{(4)}$ is the product of two dressed quark propagators $S$, see above, and the scattering matrix $T^{(4)}$ contains all possible 2→2 scattering processes in the considered channel. It is related to the four-quark scattering kernel (two-particle irreducible kernel) $K^{(4)}$ via re-summation

$$T^{(4)} = K^{(4)} + K^{(4)}G_0^{(4)}K^{(4)} + K^{(4)}G_0^{(4)}K^{(4)}G_0^{(4)}K^{(4)} + \ldots. \tag{4.6}$$

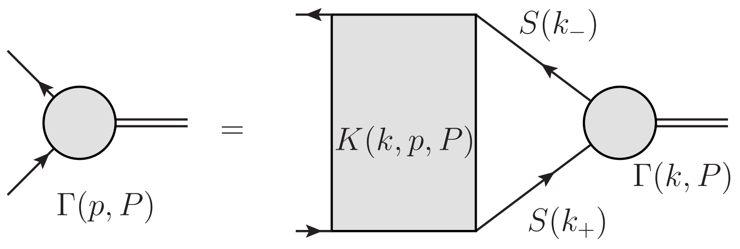


Figure 4.1: The homogeneous BSE.

This equation can be rewritten as

$$T^{(4)} = (1 + T^{(4)} G_0^{(4)}) K^{(4)} , \tag{4.7}$$

which is known as Dyson equation or inhomogeneous Bethe-Salpeter equation (BSE). To derive the BSE (sometimes also called homogeneous BSE) we plug in our assumption that $T^{(4)}$ possesses a simple pole at $P^2 = -m_B^2$ and perform a Laurent expansion in $P^2 = (p_1 + p_2)^2 = (p_3 + p_4)^2$ around $-m_B^2$:

$$T^{(4)}(p_1, p_2, p_3, p_4) = \frac{c_{-1}}{P^2 + M_B^2} + c_0 + c_1(P^2 + M_B^2) + \ldots . \tag{4.8}$$

The first term will provide the wanted equation, and the second term the normalisation, see later. All higher terms vanish for $P^2 \to -M_B^2$. The momentum-dependence of the coefficient $c_{-1}$ can be factorised,

$$c_{-1}(p_1, p_2, p_3, p_4) \propto \Gamma(P; p_1 - p_2)\overline{\Gamma}(P; p_3 - p_4)$$

leading to

$$T^{(4)} \xrightarrow{P^2 \to -M_B^2} \mathcal{N} \frac{\Gamma\,\overline{\Gamma}}{P^2 + M_B^2} . \tag{4.9}$$

$\Gamma$ obviously attains the meaning of a bound-state amplitude, and $\mathcal{N}$ is a normalization factor (to be determined later from $c_0$). Plugging this into Eq. (4.7) and taking the most singular terms only, we obtain

$$\Gamma = K^{(4)} G_0^{(4)} \Gamma . \tag{4.10}$$

A graphical depiction of this equation is shown in Fig. 4.1.

Note that in our structure-oriented equation we suppressed all indices.

**Exercise 5:**

Within this exercise the aim is to solve the simplest non-trivial approximation of the pion Bethe-Salpeter (BS) equation. The employed notation here follows the lecture.

a.) Demonstrate that the arguments of $S(q_\pm)$ in the BS equation are within a parabola-bounded region in the complex plane, *i.*e., provide the set of values of $(q \pm P_{os}/2)^2$ for $q^2 \in (0, \infty)$?

b.) Keeping only the tensor $\tau_1 = \gamma_5$ perform the Dirac algebra on the r.h.s. of the BS equation.

c.) In the approximation of keeping only (i) $\tau_1$ as Dirac tensor structure and (ii) only the lowest Chebychev moment solve the resulting integral equation by generalising it to an eigenvalue problem as explained in the lecture.