TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Integralgleichungsverfahren in der Flüssigkeitstheorie in höheren Dimensionen

zur Erlangung des akademischen Grades

Diplom-Ingeneurin

im Rahmen des Studiums

Technische Physik

eingereicht von

Margit Kranner
Martrikelnummer: 1008876

ausgeführt am Institut für theoretische Physik
der technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. DI Dr. Gerhard Kahl

Wien, am 29.10.2018

(Unterschrift VerfasserIn) (Unterschrift BetreuerIn)

MASTER THESIS

Integral-equation approaches in liquid state theory in higher dimensions

carried out for the purpose of obtaining the degree of

Master of Science

as part of the study of

Technical Physics

submitted by

Margit Kranner
Matriculation number: 1008876

at the Institute of Theoretical Physics,
Technical University of Vienna

under the supervision of

Ao.Univ.Prof. DI Dr. Gerhard Kahl

Vienna, at 29.10.2018

Abstract

In this thesis we will discuss the numerical solution of techniques, that allow the calculation of thermodynamic properties and the specific numerical structure of integral equations, describing simple fluids. The calculation is performed by a program, written in FORTRAN. It uses different specific numerical algorithms to solve the Ornstein-Zernike integral equation in combination with a closure relation. The solution leads to the correlation functions, as well as to the pair distribution functions for more component systems.

As solutions of the OZ-equations are sometimes required in higher dimensions as well, the code was generalised from the original three-dimensional case to higher (odd) dimensions. To verify the adaptations, the numerical solutions for the special case of a hard-sphere potential in different dimensions were compared to the corresponding analytic hard-sphere solutions, available for the Percus-Yevic closure relation. The program was then applied to a binary, symmetric mixture, where the cross interaction was assumed to be soft. This case is of relevance in investigations of glassy systems.

Contents

1. Introduction

The description of the properties of a fluid requires the assumption of a basic interaction between the particles. The commonly used potentials [2] are for example hard-sphere, Lennard-Jones or soft-sphere potentials. In this thesis we will only consider pair potentials $\phi(r)$, i.e. interactions that depend on the distance r between two particles only. The considered system is homogeneous and isotropic. In an obvious generalisation more components can be introduced, which requires a generalisation of $\phi(r)$ to $\phi_{ij}(r)$, with i and j being the indices of the different components.

One important function characterising the structure of the fluid is the pair distribution function $g(r)$, which plays a key role for the calculation of important thermodynamic properties, such as the excess energy U_{exc}, the pressure P or the compressibility χ_T. It contains information about the probability of a particle being located at distance r, assuming that another particle is located at the origin. This probability is normalized with respect to a totally random system of the same density ρ (like an ideal gas), with no interaction between the particles. It can be derived from the direct and the total correlation functions, $c(r)$ and $h(r)$, that on the other hand can be calculated by some integral equations, consisting of the so called Ornstein-Zernike equation, and one so called closure relation. Those relations can be derived by some basic thermodynamic assumptions, together with some approximations, that depend on the considered potential.

The program that we use for the solution of the integral equations uses three different numerical algorithms, namely the algorithm of Gillan [5], the algorithm of Ng [11] and that of Labik, Malijevski and Vonka [12]. For a better understanding of the program, those algorithms will be introduced shortly in chapter 2, together with the most important functions, subroutines and modules of the program itself.

For the main part of this thesis we will use a general functional notation $f_{ij}(r)$ where i, j

specify the different components. As the numerical structure of the algorithms requires finite vectors, we will also use a vectorial notation, with the vector f_{si} instead of the function $f_{ij}(r)$. In this case i denotes the grid point of the argument r, while s describes the running index of the different combinations of components, meaning that for two components a and b, $s = 1$ stands for aa, $s = 2$ represents both ab and ba (as we assume $ab = ba$) and $s = 3$ stands for bb.

The study of higher dimensional functions requires a detailed consideration of the Fourier transformation. The advantage of an isotropic and homogeneous system is, that we are dealing with radially symmetric functions $f_{ij}(r)$, that only depend on the distance r. Therefore, if we restrict ourselves to odd dimensions, the generalisation to higher dimensions reduces to the consideration of the Fast Fourier transformation for radially symmetric functions.
Numeric solutions in higher dimensions will be presented for hard-sphere and soft-sphere potentials.

2. Algorithm

To calculate the pair distribution function $g_{ij}(r)$ we need to know its dependence on the correlation functions $c_{ij}(r)$ and $h_{ij}(r)$ that, on the other hand, depend on the assumed potential $\phi_{ij}(r)$.
$c_{ij}(r)$ represents the direct correlation between the particles, which means that only the direct impact of the considered particle onto another particle is taken into account. It must somehow depend on the density ρ and on the concentration x_m for each component, as well as on the assumed potential. As mentioned in the introduction, the potential tells us how one special particle affects another particle in the distance r. So the direct correlation function will have quite a similar length as the potential.

2.1. Ornstein-Zernike equation

Following the formalism of Ornstein and Zernike [2] in Eq. 2.1, the total correlation functions, $h_{ij}(r)$, are the sum of the direct correlation functions $c_{ij}(r)$ and the indirect correlation functions $\gamma_{ij}(r)$. Thus the $h_{ij}(r)$ cover the direct correlation of two particles, separated by a distance r, and indirect correlations caused by other particles in the ensemble.

$$h_{ij}(r) = c_{ij}(r) + \rho \sum_m k_m \int h_{im}(\vec{r'}) c_{mj}(|\vec{r} - \vec{r'}|) d\vec{r'} \tag{2.1}$$

As the pair distribution function depends on the total correlation function by

$$h_{ij}(r) = g_{ij}(r) - 1 \tag{2.2}$$

we not only need to know the impact of the potential onto the direct correlation function, but also the way how the direct correlation leads us to the total correlation function.

So the impact of one particle at $\vec{r}$ onto another particle at $\vec{r'}$ is given by the direct correlation function $c(|\vec{r}-\vec{r'}|)$ between them, times the total correlation function of the particle at $\vec{r'}$.
As we know that

$$\gamma_{ij} = h_{ij}(r) - c_{ij} \ , \tag{2.3}$$

the function $c(|\vec{r}-\vec{r'}|)$ is the essential part of the indirect influence of a particle at the origin onto a particle at $\vec{r'}$. So the total correlation function at $\vec{r'}$ can be considered as weight for the 'in-between' function defining the indirect interaction of r on r'. Those weighted interactions are then 'summed up' by the integral over $d\vec{r'}$ for each point $\vec{r'}$.
Finally the Ornstein-Zernike equation, that can be considered as convolution between h(r) and $c(r)$, has to be multiplied by the density (and by x_m for more than one component.
As a convolution two functions in r-space becomes a product of the related Fourier-transforms of the functions in k-space, we will only use the Fourier transform of Eq. (2.1), namely

$$\tilde{h}_{ij}(k) = \tilde{c}_{ij}(k) - \rho \sum_m x_m \tilde{h}_{im}(k) \tilde{c}_{mj}(k), \tag{2.4}$$

or alternatively,

$$\tilde{\gamma}_{ij}(k) = \frac{\rho \sum_m x_m \tilde{c}^2_{im}(k)}{1 - \rho \sum_m x_m \tilde{c}_{im}(k)} \ . \tag{2.5}$$

2.2. Closure Relations

Since the the OZ-equation relates to unknown functions, $h(r)$ and $c(r)$, we need an additional relation between these functions in order to solve this equation. These functions, called 'Closure relations' in literature, involve also the potential $\phi(r)$. They can be derived with the formalism of statistical mechanics [2]. A short motivation for some of the most common closure relations will be given here, inspired by Kierfield [3].
For simplicity, the indices i and j will be omitted for the rest of chapter 2. As mentioned above, the total correlation function $h(r)$ can be split into a direct and an indirect contribution. For the pair distribution function $g(r)$ we proceed in the same way.

$$g_{\text{total}}(r) = g_{\text{indirect}}(r) + g_{\text{direct}}(r) \tag{2.6}$$

where $g_{\text{total}}(r)$ is the conventional pair distribution function $g(r)$, that we already introduced. Similar to Eq. (2.2) there is a connection between the indirect correlation function $\gamma(r)$ and $g_{\text{indirect}}(r)$:

$$\gamma(r) = g_{\text{indirect}}(r) - 1 \ . \tag{2.7}$$

So we obtain

$$c(r) = h(r) - \gamma(r) = g(r) - g_{\text{indirect}}(r) \ . \tag{2.8}$$

Furthermore we introduce a mean field two-particle potential $\phi_{\text{MF}}(r)$ that yields the mean distribution function $g(r)$ by the well known Boltzmann distribution:

$$g(r) = e^{\frac{-\phi_{\text{MF}}(r)}{k_B T}} \tag{2.9}$$

This potential $\phi_{\text{MF}}(r)$ is given by the mean force $\vec{F}(|\vec{r} - \vec{r'}|)$ between two particles. It can be derived by the total potential $V_N(r)$, that contains all interactions between the

N particles. We denote one vector $\vec{r_i}$ as coordinate vector of particle i. As the potential $\phi_{\mathrm{MF}}(r)$ is a two-particle potential, we label particle 1 and 2 with r and r' and obtain [3]:

$$\begin{aligned}
\vec{F}(|\vec{r}-\vec{r'}|) = & -\left\langle \nabla_{\vec{r_1}} V_N(\vec{r},\vec{r'},\vec{r_3},...\vec{r_N})\right\rangle_{\vec{r_1}=\vec{r},\vec{r_2}=\vec{r'}} \\
& = -\frac{\int d^3\vec{r_3}...\int d^3\vec{r_N}\nabla_{\vec{r_1}}\ V_N\ \exp(V_N/k_BT)}{\int d^3\vec{r_3}...\int d^3\vec{r_N}\exp(V_N/k_BT)} \\
& = -\frac{1}{k_BT}\ \nabla_{\vec{r}}\ln\left(\int d^3\vec{r_3}...\int d^3\vec{r_N}\ \exp(V_N/k_BT)\right) \qquad (2.10) \\
& = -\frac{1}{k_BT}\ \nabla_{\vec{r}}\ln(\langle\delta(\vec{r}-\vec{r_1})\delta(\vec{r'}-\vec{r_2})\rangle) \\
& = -\frac{1}{k_BT}\ \nabla_{\vec{r}}\ln[g(\vec{r}-\vec{r'})]\ .
\end{aligned}$$

The last equality originates from the definition of the pair distribution function $g(r)$ for a translational invariant radial symmetric potential.
Thus we can say that, based on Eq. (2.10), there is a mean force that fulfils

$$\vec{F}(|\vec{r}-\vec{r'}|) = \nabla_{\vec{r}} k_BT\ln(g(r)), \qquad (2.11)$$

with $|\vec{r}-\vec{r'}| = r$.
The corresponding potential of the mean force $\vec{F}$ is then called ϕ_{MF}.

Now we can consider different approximations for the direct pair potential $\phi(r)$, which leads us to the different closure relations.

1) **Mean-spherical Approximation (MSA)**

For a potential with a hard sphere part and a very weak long range part we can choose

$$\phi(r) \approx \phi_{\mathrm{MF}}(r) \tag{2.12}$$

The long range potential is so weak, that the particles only affect their nearest neighbour.

We therefore obtain

$$g(r) = e^{\frac{-\phi(r)}{k_B T}} \tag{2.13}$$

and finally

$$\begin{aligned} g(r) &= 0 \qquad &, r < \sigma, \quad \phi = \infty \\ c(r) &\approx \tfrac{-\phi(r)}{k_B T} &, r > \sigma, \quad \phi << 1 \end{aligned} \tag{2.14}$$

with σ as the hard sphere diameter.

For 'softer' but also strongly repulsive potentials, the MSA can be extended to the Soft-MSA (SMSA). It separates the potential into a compulsive and a repulsive part, represented by $\phi_1(r)$ and $\phi_2(r)$. The separation is performed at the positive minimum r_0 of the potential as follows:

$$\begin{aligned} r < r_0 : \phi_1(r) &= \phi(r_0) \\ \phi_2(r) &= \phi(r) - \phi(r_0) \\ r > r_0 : \phi_1(r) &= \phi(r) \\ \phi_2(r) &= 0\ . \end{aligned}$$

$$\tag{2.15}$$

Now we can write the SMSA Closure relation as:

$$c(r) = \left[1 - e^{\frac{\phi_1(r)}{k_B T}}\right] g(r) - \frac{\phi_2(r)}{k_B T} \ . \tag{2.16}$$

2) **Percus-Yevick (PY) closure relation**

The structural properties of systems like short range potentials are often described well using the PY- approximation. Here the indirect part of the distribution function, $g_{\text{indirect}}(r)$, can be written similar to Eq. (2.9) as

$$g_{\text{indirect}}(r) = e^{\frac{\phi_{\text{MF}}(r) - \phi(r)}{k_B T}} = g(r) e^{\frac{\phi(r)}{k_B T}} \ . \tag{2.17}$$

The PY- closure relation then reads:

$$c(r) = g(r) - g_{\text{indirect}}(r) = g(r)(1 - e^{\frac{\phi(r)}{k_B T}}) \ . \tag{2.18}$$

3) **Hypernetted-Chain closure relation (HNC)**

The HNC - closure relation starts from Eq. (2.17), but expands the exponential function for small arguments of $[\phi_{\text{MF}}(r) - \phi(r)]$, giving

$$g_{\text{indirect}}(r) = 1 - \frac{\phi_{\text{MF}}(r) - \phi(r)}{k_B T} \tag{2.19}$$

so that

$$c(r) = g(r) - g_{\text{indirect}}(r) = g(r) - 1 + \ln(g(r)) + \frac{\phi(r)}{k_B T} \ . \tag{2.20}$$

Thus one obtains

$$g(r) = e^{-\frac{\phi(r)}{k_B T} + h(r) - c(r)} \ . \tag{2.21}$$

Equation (2.20) and (2.21) hold for long range potentials.

As closure relations are only approximated expressions, a better accuracy can be achieved by using parametrised closure relations. Those relations are hybrids between two closure relations mentioned above, introducing a mixing parameter which has to be fixed.
One usually introduces the mixing function

$$f(\alpha, r) = 1 - e^{-\alpha r} \quad \alpha > 0 \tag{2.22}$$

2.2.1. HMSA

The HMSA is a combination of the HNC and SMSA relations:

$$g(r) = e^{-\frac{\phi_1(r)}{k_B T}} \left(1 + \frac{\exp\left[(h(r) - c(r) - \frac{\phi_2(r)}{k_B T} \right] f(\alpha, r) - 1}{f(\alpha, r)} \right) \tag{2.23}$$

with $\phi_1(r)$ and $\phi_2(r)$ defined above.
As one can see, different choices of α lead to the HNC ($\alpha = \infty$) or to the SMSA ($\alpha = 0$) relations.

2.2.2. Rogers-Young (RY) relation

The RY method is a combination between the RY and the HNC relations [2]:

$$g(r) = e^{-\frac{\phi(r)}{k_B T}} \left(1 + \frac{\exp(h(r) - c(r)) f(\alpha, r) - 1}{f(\alpha, r)} \right) \ . \tag{2.24}$$

$\alpha = \infty$ leads to the HNC relation, while a choice of $\alpha = 0$ reproduces the PY relation.

2.2.3. MHNC

The MHNC relation interpolates between the HNC relation and the exact relation, introducing the so called bridge functions B(r)

$$g(r) = e^{-\frac{\phi(r)}{k_B T} + h(r) - c(r) + B(r)} \; . \tag{2.25}$$

So for $B(r) = 0$ we have a single HNC relation, while we consider Eq. (2.25) for $B(r) \neq 0$ to be exact. Of course we do not know the exact solution, but we can calculate it for some reference systems like the hard sphere potential for example, of which we know an analytic solution.
A further method, called RHNC, optimizes those Bridge functions for a hard sphere potential as described in chapter 3.

2.3. Solving the Integral equations

Now that we collected all the relevant equations, let us recall the essence of this thesis, namely the calculation of the pair distribution function $g(r)$. In order to solve the OZ-equation in combination with a closure relation, we need a numeric algorithm to iteratively solve the two integral equations.

Of course, the program uses discrete vectors f_{si} instead of the functions $f_{ij}(r)$, where s is the index for the number of different combinations of components, replacing ij. (As $f_{ij}(r) = f_{ji}$, we only have $s = 1, 2, 3$ for a two component system, for example.) The index i specifies the grid point index of the underlying vector space. For simplicity we will still use the general functional description here.

All numerical algorithms applied on those equations use essentially the same iteration method. We start with an initial guess for $\gamma(r)$ (for example $\gamma(r)$=0) and calculate $c(r)$ via the Closure relation. Then we transfer from $c(r)$ to k-space, receiving $\tilde{c}(k)$. The process is described in detail in section [3.3.6]. The transformation is necessary to use the OZ-equation in k-space in Eq. (2.5).

We then insert $\tilde{c}(k)$ into the OZ-equation in k-space, Eq. (2.5), and obtain a function $\tilde{\gamma}^{(1)}(k)$. Now we transform $\tilde{\gamma}^{(1)}(k)$ back into r-space and obtain a function $\gamma^{(1)'}(r)$ by another Fourier transformation, where $'$ denotes that $\gamma^{(1)'}(r)$ is the output value. The difference between $\gamma^{(1)}(r)$ and $\gamma(r)$ is then calculated. If the difference is small enough the program stops. Otherwise this circle is iterated, now starting with the function $\gamma^{(1)}(r)$. Iterations are counted by the upper index n. The main difference between the different numerical algorithms is how the new starting value is calculated, using the output value $\gamma^{(n)'}(r)$.

For a basic Picard iteration we have

$$\gamma^{(n+1)}(r) = (1-\beta)\gamma^{(n)'}(r) + \beta\gamma^{(n-1)'}(r) \quad at \quad 0 \leq \beta \leq 1 \tag{2.26}$$

with a fixed value for β. As soon as the functions attempt higher values (according to stronger correlations) the method becomes very slow. Therefore we need faster algorithms with a better convergence. Using the program we can choose between three different algorithms, that, for the sake of completeness, are shortly discussed below.

2.3.1. Algorithm of Ng

Similar to the Picard iteration, the algorithm of Ng [11] uses a linear combination of the old output functions of the loop for the new input. We start from the expression

$$\gamma^{(n+1)}r) = (1 - c_1 - c_2)\gamma^{(n)}(r) + c_1\gamma^{(n-1)}(r) + c_2\gamma^{(n-2)}(r) \tag{2.27}$$

In contrast to the simple Picard iteration, the coefficients c_1 and c_2 are optimized for each circle. This is done as follows:

We define

$$\gamma^{(n)'}(r) = A\gamma^{(n)}(r) \tag{2.28}$$

$$d^{(n)}(r) = \gamma^{(n)'}(r) - \gamma^{(n)}(r) \tag{2.29}$$

where the primed variable $\gamma^{(n)'}(r)$ denotes the solution of the n^{th} iteration, containing closure relation, Fourier transformation, OZ-relation and inverse Fourier transformation, represented by an operator A here. Now we compare

$$A\gamma^{(n+1)}(r) = \gamma^{(n+1)'}(r) = (1 - c_1 - c_2)\gamma^{(n)'}(r) + c_1\gamma^{(n-1)'}(r) + c_2\gamma^{(n-2)'}(r) \tag{2.30}$$

with equation Eq. (2.27) and obtain

$$d^{(n+1)}(r) = \gamma^{(n+1)'}(r) - \gamma^{(n+1)}(r)\ . \tag{2.31}$$

As $d^{(n+1)}(r)$ is the direct measure for the quality of convergence, the optimal coefficients c_1 and c_2 are obtained by minimizing the function $[d^{(n+1)}(r)]^2$. The criterion for the determination of c_1 and c_2 finally reads

$$\begin{aligned}
&\Big((d^{(n)}(r) - d^{(n-1)}(r)), (d^{(n)}(r) - d^{(n-1)}(r))\Big)\, c_1 + \\
&\Big((d^{(n)}(r) - d^{(n-1)}(r)), (d^{(n)}(r) - d^{(n-2)}(r))\Big)\, c_2 = \\
&\qquad \Big(d^{(n)}(r), (d^{(n)}(r) - d^{(n-1)}(r))\Big) \\
&\Big((d^{(n)}(r) - d^{(n-1)}(r)), (d^{(n)}(r) - d^{(n-2)}(r))\Big)\, c_1 + \\
&\Big((d^{(n)}(r) - d^{(n-2)}(r)), (d^{(n)}(r) - d^{(n-2)}(r))\Big)\, c_2 = \\
&\qquad \Big(d^{(n)}(r), (d^{(n)}(r) - d^{(n-2)}(r))\Big)
\end{aligned} \tag{2.32}$$

with the common inner product for functions, defined as

$$(a(r), b(r)) = \int a(r) b(r) dr \quad . \tag{2.33}$$

The only remaining task is to solve a linear equation system for c_1, c_2. Finally the new starting value $\gamma^{(n+1)}(r)$ is calculated by

$$\gamma^{(n+1)}(r) = (1 - c_1 - c_2)\gamma^{(n)'}(r) + c_1 \gamma^{(n-1)'}(r) + c_2 \gamma^{(n-2)'}(r) \tag{2.34}$$

instead of Eq. (2.27), because otherwise all further $\gamma^n(r)$ with $n > 2$ would be linear combinations of the first solutions.

2.3.2. Algorithm of Gillan

The algorithm of Gillan is described in detail in chapter 4, as an example for important differences in higher dimensions. Therefore no detailed consideration is provided in this section. For a closer adjustment to the program, index notation is used there.

2.3.3. Algorithm of Labik, Malijevsky and Vonka (LaMaVo)

The algorithm of Labik, Malijevsky and Vonka [12] separates the considered vector space into a coarse and a fine part, using the Fourier transform of the first taylor expansion for the closure relations, describing $c(r)$. Then a newton iteration is performed before we move to the fine part. The length of the coarse part is typically about $60dr$. The benefit of this separation is the improvement of speed, due to the the shorter vectors involved in the coarse part.

In detail the taylor expansion is done as follows:

Considering the closure relation as a general equation, depending on $\phi(r)$ and $\gamma(r)$

$$c(r) = f(\phi(r), \gamma(r)) \tag{2.35}$$

one can formally write the expansion for any point of expansion $c^0(r)$ as

$$c(r) \approx c^0(r) + \left(\frac{\partial f}{\partial \gamma(r)}\right)_{\gamma(r)=\gamma^0(r)} (\gamma(r) - \gamma^0(r)) \ . \tag{2.36}$$

We will understand the meaning of $c^0(r)$ later. For small arguments $(\gamma(r) - \gamma^0(r))$ we can neglect higher derivatives of $f(\phi(r), \gamma(r))$ (with respect to $\phi(r)$). The Fourier transform of Eq. (2.36) in three dimensions then reads

$$\tilde{c}(k) = \tilde{c}^0(k) + \frac{4\pi}{k} \int_0^\infty r \sin(kr) \frac{\partial f}{\partial \gamma(r)}\bigg|_{\gamma(r)=\gamma^0(r)} (\gamma(r) - \gamma^0(r)) dr \ . \tag{2.37}$$

If we insert the inverse Fourier transforms for $\gamma(r)$ and $\gamma^0(r)$ as well, we obtain

$$\tilde{c}(k) = \tilde{c^0}(k) + \frac{4\pi}{k}\int_0^\infty r\sin(kr)\frac{\partial f}{\partial\gamma(r)}\Big|_{\gamma(r)=\gamma^0(r)} dr$$
$$\left(\frac{1}{2\pi^2 r}\int k'\sin(k'r)\gamma(r)dk' - \frac{1}{2\pi^2 r}\int_0^\infty k'\sin(k'r)\gamma^0(r)dk'\right) . \tag{2.38}$$

With respect to the definition of the Fourier transformation for radially symmetric functions and some relations for sine and cosine which will be discussed in detail for the Gillan algorithm in chapter 4, we can rewrite equation (2.38)

$$\tilde{c}(k) = \tilde{c^0(k)} + \int \tilde{p}(k',r)(\tilde{\gamma}(k') - \tilde{\gamma^0}(k'))dr \tag{2.39}$$

with

$$\tilde{p}(k',r) = \frac{k'}{k}\int \frac{\partial f}{\partial\gamma(r)}_{\gamma(r)=\gamma^0(r)}(\cos(r(k-k')) - \cos(r(k+k')))dk' . \tag{2.40}$$

The Newton iteration is performed on the function $F(\tilde{\gamma}(k))$:

$$F(\tilde{\gamma}(k)) = \tilde{\gamma}(k) - F_{OZ}(\tilde{c}(\tilde{\gamma(k)})) \tag{2.41}$$

with $F_{OZ}(\tilde{c}(\tilde{\gamma}(k)))$ as the equation of Ornstein-Zernike in k-space, Eq. (2.5), calculating a new value for $\tilde{\gamma}(k)$.
Now we can iteratively find a solution for $\tilde{\gamma}(k)$ in Eq. (2.41) by searching for the zeroes of $F(\tilde{\gamma}(k)) = 0$:
Due to the general formula of the Newton iteration

$$\tilde{\gamma}^{(n+1)}(k) = \tilde{\gamma}^{(n)}(k) - \frac{F(\tilde{\gamma}^{(n)(k)}}{\frac{\partial F(\tilde{\gamma}^{(n)}(k))}{\partial\tilde{\gamma}^{(n)}(k)}} \tag{2.42}$$

we obtain

$$\tilde{\gamma}^{(n+1)}(k) = \tilde{\gamma}^{(n)}(k) - \frac{F(\tilde{\gamma}^{(n)}(k)}{1 - \frac{\partial F_{OZ}(\tilde{c}^{(n)}(\tilde{\gamma}(k)))}{\partial \tilde{c}(k)} \frac{\tilde{c}(k)}{\tilde{\gamma}(k)}} \; . \tag{2.43}$$

Since the derivative $\frac{\tilde{c}(k}{\tilde{\gamma}(k)}$ in Eq. (2.43) is $\int \tilde{p}(k', r) dk'$ (as one can see from Eq. (2.39)), we only need to insert expression Eq. (2.40) into Eq. (2.43), together with $\Delta\tilde{\gamma}(k) = \tilde{\gamma}^{(n+1)}(k) - \tilde{\gamma}^{(n)}(k)$, to obtain the linear equation system:

$$\Delta\tilde{\gamma}(k) - \frac{\partial F_{OZ}(\tilde{c}^{(n)}(\tilde{\gamma(k)}))}{\partial \tilde{c}(k)} \int \tilde{p}(k', r) dk' \Delta\tilde{\gamma}(k') + F(\tilde{\gamma}^{(n)}(k)) = 0. \tag{2.44}$$

$$\Delta\tilde{\gamma}(k) = \tilde{\gamma}^{(n+1)}(k) - \tilde{\gamma}^{(n)}(k) \tag{2.45}$$

So the procedure starts with an initial $\gamma^{(n=0(i))}(r)$ and a calculation of the *non-separated* function $c^{(0(i))}(r)$ via a closure relation, followed by a calculation of $\tilde{c}^{0(i)}(k)$ and $\tilde{\gamma}^{(0(i))}(k)$ via Fourier transformation. Index n, which is zero at the beginning, counts the steps of newton iterations, while i represents the fine steps. For the very first step, i is zero as well, of course. $c^{(0(i))}(r)$ corresponds to $c^0(\gamma^0(r))$ in Eq. (2.36), while $\gamma^{(0(i))}(r)$ is the point of expansion.
Now we move on to the coarse step, which means that we only use the first 60 positions of all functions that are involved in the process. At first we insert $\tilde{c}^{(0)}(k)$ and $\gamma^{(0)}(r)$ into Eq. (2.40) to calculate $\tilde{p}(k', r)$ for the coarse part.

The second part of the coarse step is represented by the calculation of $\Delta\tilde{\gamma}(k)$ by Eq. (2.44) and Eq. (2.45). Therefore, as $F(\tilde{\gamma}^{(n(i))}(k))$ in Eq. (2.44) contains $F_{OZ}(\tilde{c}^{(n(i))}(k))$, we need to calculate the coarse part of $\tilde{c}^{(n(i))}(k)$ by the Taylor expansion Eq. (2.39), considering $\tilde{\gamma}(k)$ as $\tilde{\gamma}^{n(i)(r)}$ such that:

$$\tilde{c}^{(n(i))}(k) = \tilde{c}^{(0(i))}(k) + \int \tilde{p}(k', r)(\tilde{\gamma}^{(n(i))}(k') - \gamma^{(\tilde{0(i)})}(k'))dr \tag{2.46}$$

So in the first step of each coarse part we set $\tilde{c}^(n(i))(k) = c^{\widetilde{0(i)}}(k)$, because we did not produce $\tilde{\gamma}^{n(i)}(k')$ so far, of course.
Now we can calculate $\Delta\tilde{\gamma}(k)$ and $\tilde{\gamma}^{(n+1)}(k)$ from the linear equation system Eq. (2.44) and Eq. (2.45) and set $\tilde{\gamma}^{(0(i))}(k) = \tilde{\gamma}^{(1(i))}(k)$, or $\tilde{\gamma}^{(n(i))}(k) = \tilde{\gamma}^{(n+1(i))}(k)$ in general. If $\Delta\tilde{\gamma}(k)$ is not small enough, a new coarse Newton iteration cycle starts with the new value $\tilde{\gamma}^{(n+1(i))}(k)$ instead of $\tilde{\gamma}^{(n(i))}(k)$, until $\Delta\tilde{\gamma}(k)$ corresponds to a satisfying value, that has to be defined at the beginning.
The output value is $\tilde{\gamma}^{(n+1(i))}(k)_{coarse}$, that we can call $\tilde{\gamma}^{(0(i+1))}(k)$ for the next step.

Moving on to the fine step, we have to apply the Orstein-Zernike equation on the remaining $N - 60$ positions of the initial $\tilde{\gamma}^{(0(i))}(k)$ and obtain $\tilde{\gamma}^{(0(i+1))}(k)_{fine}$, where N is the number of grid points. Then we can evaluate $\tilde{\gamma}^{(0(i+1))}(k)$ by combining $\tilde{\gamma}^{(0(i+1))}(k)_{coarse}$ and $\tilde{\gamma}^{(0(i+1))}(k)_{fine}$, and transform this function back to r-space.
If $\Delta\gamma(r)$ is small enough as well, the algorithm stops; Otherwise, the whole circle, containing the calculation of the coarse part will be performed again, with the new starting value $\tilde{\gamma}^{(n+1(i))}(k)$.

Obviously the expression for the Fourier transformation we use at Eq. (2.37) will be different in higher dimensions. Fortunately, as explained in chapter [4] and for the same reason as for the Gillan algorithm, it will make no difference, as a full iteration circle totally encloses the coarse circle, so that it is only a matter of speed how fast convergence will be achieved.

As the generalization of the FFT to higher dimensions leads to more complicated functions (see Eq. (4.56) and (4.57)) we will use the same functions for the FFT routine for all dimensions.

3. Description of the program in detail

To make it easier for further users to work with the program for the numerical solution of the OZ -equation, this section will be dedicated to a description of the most important parameters and functions as well as of important units and a short manual how to use it. The program contains six different modules, together with external programs for the potential, as well as three input files.

As mentioned, the program works with discrete vectors instead of functions. For the functions we use N grid points with distance dr covering the relevant part of r-space. Because the FFT-routine needs $N = 2^m$- dimensional vectors as input, we have to choose $N = 2^m$ in the program as well.
Furthermore, dk is fixed via $dr \cdot dk = \pi/N$, as the number of grid points in k-space must be the same as in r-space.

3.1. Potential programs

In a first step we have to run one of the potential programs, producing the desired potential file called 'poten' as an input file for the main program. Therefore the parameters dr, N, ρ, etc., that are input values for the program, have to be set in the potential file as well.

3.1.1. File PHI_LJ.f90

PHI_LJ.f90 produces a Lennard-Jones Potential

$$\phi(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{3.1}$$

for particles separated by a distance r, with an an attractive part r^{-6}, modelling the dipol-dipol interaction as well as Van-der-Waals-forces for bigger distances. The closer the particles get, the more important the repulsive part r^{-12} becomes, due to the Pauli principle. The parameter σ stands for the distance at $\phi(r) = 0$, while ϵ is the depth of the potential at $r = \sigma\sqrt[6]{2}$.

In general dimensionless reduced units are used instead of σ and ϵ to specify the potential, defined as $T^* = \frac{k_B T}{\epsilon}$ as well as $\rho^* = \rho\sigma^3$. Furthermore the program uses the unit a: In three dimensions a is defined as follows:

$$\rho = \frac{N}{V} = \frac{1}{v_p} = \frac{3}{4\pi\, a^3} \Rightarrow \rho^* = \frac{3}{4\pi}\left(\frac{\sigma}{a}\right)^3 . \tag{3.2}$$

v_p is the spherical volume which contains only one particle; a is therefore the radius of this sphere (not to be confused with the size of the particle itself). In higher dimensions v_p has to be calculated for a d-dimensional sphere, so that ρ reads for five dimensions (where $\rho^* = \rho\sigma^5$):

$$\rho = \frac{N}{V} = \frac{1}{v_p} = \frac{15}{8\pi^2\, a^5} \Rightarrow \rho^* = \frac{3}{4\pi}\left(\frac{\sigma}{a}\right)^5 . \tag{3.3}$$

For seven dimensions we find (with $\rho^* = \rho\sigma^7$):

$$\rho = \frac{N}{V} = \frac{1}{v_p} = \frac{105}{16\pi^3\, a^7} \Rightarrow \rho^* = \frac{3}{4\pi}\left(\frac{\sigma}{a}\right)^7 . \tag{3.4}$$

As the program PHI_LJ.f90 asks for the standard parameters such as the energy parameter ϵ, temperature T, σ and density ρ, one has to take that into account by setting each ϵ and ρ to one. For example, if the reduced temperature is given by $T^* = 0.7$ and $\rho^* = 0.85$ one has to set $T = 0.7$, $\sigma^d = 0.85$, $\epsilon = 1$ and $\rho = 1$,with $d = 1, 3, 5$.

Furthermore, the program asks for the number of m, which is related to the number of grid points N via $N = 2^m$ (which should be in the range of $[8, 14]$); The program also requests the grid size dr, as well as the number of components NC. If NC is greater than one, the program asks for separated values T^* and ρ^* for each component, together with some optional additive parameters for distance and energy, which are P1=0 and P2=1 by default. P1 is added to σ, while P2 is multiplied with ϵ.

3.1.2. File PHI_LJ-2comp.f90

PHI_LJ-2comp.f90 can be used for the special case of a two component system [4], with a pair potential for the interaction between the different components, given by

$$\phi_{12}(r) = -\epsilon_{12} \left[\frac{c^2}{x^2 + c^2} \right]^6 \tag{3.5}$$

where $x = \frac{r_{12}}{\sigma}$ (with r_{12} as the difference between component 1 and 2), $c = 3\sigma$ and ϵ_{12} is the interaction energy. As $\phi_{12}(r)$ is attractive, c is chosen such that the range of the attraction is short enough to make sure that only one atom of species 1 can interact with at least one atom of species 2 [4]. The potential between equal components is either the same as for 'PHI_LJ.f90', or can be chosen as a soft-sphere potential instead with

$$\phi_\alpha(r) = \epsilon_\alpha \left(\frac{\sigma_\alpha}{r_\alpha} \right)^\nu ; \alpha = 1, 2. \tag{3.6}$$

(ν is mostly chosen as $\nu = 12$, while d is the dimension of the system and r_α is the difference between particles of the same species 1 or 2.)
The soft sphere model is purely repulsive and can be considered as a high temperature limit of the Lennard-Jones potential, neglecting dipol and Van-der-Waals forces. Again the program uses reduced units, ρ^* and T^*.
So as before ϵ and ρ are set to one, as well as σ. Furthermore the dimensionless variable $\Gamma = \frac{\rho^*}{T^{*d/\nu}}$ is introduced with

$$\frac{\epsilon}{k_B T} = \frac{1}{T^*} = \left(\frac{\Gamma}{\rho^*} \right)^{\nu/d} = \Gamma^{\nu/d} \tag{3.7}$$

The user can choose between metric units and units a. In the program we use $S = \frac{1}{a}$ instead. If metric units are used, one has to change the scaling of ρ in the main program as well. (See section [3.3.1]). For standard units, S is set to one. Finally the program produces the following potential:

$$\begin{aligned}-\phi_{12}(r_{12}) &= -\epsilon_{12}\Gamma^{\nu/d}\left(\frac{(0.3S)^2}{(0.3S)^2+(\frac{r_{12}}{S})^2}\right)^6 \\ \phi_\alpha(r_\alpha) &= \Gamma^{\nu/d}\left(\frac{S}{r_\alpha}\right)^{12} \quad ;\alpha = 1,2\end{aligned} \tag{3.8}$$

As for 'PHI_LJ.f90', the user has to choose dimension, grid size, grid points and ϵ_{12}.

3.1.3. File hardsphere.f90

The program 'hardsphere.f90' produces a hard sphere potential as follows:

$$\phi(r) = \begin{cases} 0; & r \leq \sigma \\ \infty; & r > \sigma\ , \end{cases} \tag{3.9}$$

where σ is the particle diameter. As before, the unit a is chosen, such that $\rho = 1$.
In contrast to the Lennard-Jones potential, temperature and energy are not relevant for hard spheres. We only need information about the density ρ and the particle diameter σ, to specify the system. The commonly used parameter for hard sphere systems is η which can be calculated from ρ and σ:

$$\eta = \frac{\rho \cdot V_d}{2^d} \tag{3.10}$$

V_d describes the volume of a particle in d dimensions with radius $\frac{\sigma}{2}$. With this definition, together with the definition of ρ by the standard measure a, we can evaluate σ:

d=3:	$V_d = \frac{4\pi}{3}\sigma^3$	$\rho = \frac{3}{4\pi a^3}$	$\sigma = 2\eta^{\frac{1}{3}}a$
d=5:	$V_d = \frac{8\pi^2}{15}\sigma^5$	$\rho = \frac{15}{8\pi^2 a^5}$	$\sigma = 2\eta^{\frac{1}{5}}a$
d=7:	$V_d = \frac{16\pi^3}{105}\sigma^7$	$\rho = \frac{105}{16\pi^3 a^7}$	$\sigma = 2\eta^{\frac{1}{7}}a$

Table 1: Important relations between σ, ρ, η and a for hard spheres.

The radius σ is called 'dred' by the program. After the program received the information for dimension d and η, it produces a potential with a discontinuity at σ, according to Eq. (3.9).

3.2. Input Files

Apart from the potential file **poten** produced by one of the potential programs,

3.2.1. File inp_n

The file **inp_n** contains some parameters read by the program **IEM-new.f90** at the beginning.

- 1.line: g, rho, dr, rnu
- 2.line: Scale, par(i);i=1,NrIntTypes
- 3.line: shs

g is a scaling variable used in the subroutine **PHASED** (which is described in section 3.3) as well as in FFT-back transformations, which is set to zero in general.
rho and *dr* are the well known density and grid size, while *rnu* is a scaling factor for the Axilrod-Teller potential, which was not part of this thesis, but can be looked up in the thesis of Christian Libert [1], at page 45. One has to take into account, that ρ is the *total* density here, which means that for two components with $\rho_{1,2} = 1$ we have to set $\rho = 2$ here!
Scale is a scaling factor for the potential in general, with which the potential is multiplied.
par(i) is a vector for the mixing parameter of the HMSA (Eq. (2.23)) and RY (Eq. (2.24)) closure relation between the components.

So for two components we have to specify three different variables, as we always assume $f(12) = f(21)$ for all functions. Three components therefore need six parameters. For RY being a mixture between HNC and PY closure relation, we have to set $par = 0$ to obatin the PY closure and par=∞ for the HNC closure (but 10^{24} should do it as well.) If HMSA is chosen, $par = 0$ leads to SMSA, while par=∞ leads to the HNC closure. Lastly we have shs which is a scaling variable for the RHNC method, which is not relevant for this thesis.

3.2.2. File gaminp

The file **gaminp** contains the initial $\gamma(r_i)$, which is set to zero by the potential programs by default. In **PHI_LJ-new.f90** one can avoid that $\gamma(r_i) = 0$, which can be important for iterative studies, like, for example, when systematically increasing Γ.

3.2.3. File poten

As mentioned, the file **poten** is produced by the potential programs, providing the potential function on the grid.

3.3. Modules

For a better readability, the main program was split into six different modules. We from now on use index notation, with i or j for the grid points and s for the index indicating the component.

3.3.1. Module IEM-new.f90

In the main program, all global parameters and variables are defined. If the user changes the number of components or the number of grid points, those parameters have to be set

explicitly in **IEM-new.f90** as they are not read from **in_p**. The related parameters are m, specifying the number of grid points via $N = 2^m$, $NrTypes$, representing the number of components and the vector $rn(NrTypes)$, containing the density of each component, mostly set to one. Thus for a single component system we have $rn(NrTypes) = (1.)$, while a two component system is specified by $rn(NrTypes) = (1., 1.)$.
Other important global parameters are:

nx	$nx = 2^m$ as the number of grid points
npoint	$npoint = 2^{m+1}$, used in the FFT-routines,for example
NrIntTypes	number of different combinations of components, (because $f(12) = f(21)$ for all functions) $NrIntTypes = \frac{NrTypes}{2}(NrTypes + 1)$
iTBcutoff	cutoff point for the effective potential, if a three component system is considered
iMethod, iAlgorithm, **iTask, iMode,** **iThreebody, iDimension,** **ItBmax, iSucess,**	setting parameters for method, algorithm task, number of components three- or two-body potential, dimension max. number of iterations for the bridge cycle parameter of success with iSuccess=1 if the algorithm succesfully converges and iSuccess=0 otherwise.

isdepth,isldepth,isloop	counting variables for checking the convergence, used by different routines
dq	increment dk in k-space calculated by $dq = \frac{2\pi}{npoint\, dr}$
cR, cQ, GammaR, GammaQ, gR, GammaR_new, Phi, PhiAtt, PhiRep	arrays of dimension (nx,$NrIntTypes$), for the functions c_{is}), γ_{is}),g_{is}) and there Fourier transforms $\tilde{c}_{is}$, $\tilde{\gamma}_{is}$ as well as for the potential function $\phi(r_{is})$. PhiRep and PhiAtt are relevant when a potential split is performed
fmix	mixing paramter for the closure relations representing the function $f(\alpha, r)$ in Eq. (2.22) α is read by **in_p** and stored in $par(i)$ for i being the index for the number of components
typtrans	running index for the number of component combinations set by the subroutine initAlgorithm

Some parameters are set in the modules itself:

MODULE Var_Gillan:	
NrBaF	Number of basis functions for the Gillan algorithm
nod	size of the space between the nodes for the basis functions P_BaF in terms of dr

b(NrBaF,NrBaF)	array for the conjugated basis functions $B_{\alpha\beta}$ to calculate Q_a^i (see Gillan algorithm)
rJacobi	array of the Jacobi matrix
MODULE Var_Derivation:	variables that are generally used for the derivation of functions, used by the module **THERMO-new.f90**
ni	dimension of the derivated array
func	array for the derivated function,
ri	array of arguments of the function
MODULE Var_Thermo:	
UnC_tot, UnC(NrIntTypes)	total excess energy U_{tot}^{ex} and U^{ex} for each component
P_tot, P(NrIntTypes),	total pressure P and P for each component
chi_tot, chi(NrIntTypes),	total compressibility χ and χ for each component
mu_tot, mu(NrIntTypes),	total μ and μ for each component
A_RHNC_tot,	total free Helmholtz energy A and A for each component
A_RHNC(NrIntTypes)	component, relevant if RHNC is chosen as closure relation

MODULE Var__LaMaVo:	
MaxIndexNewton	size of the coarse part
P(0:MaxindexNetwton, 0:MaxIndexNewton, NrIntTypes)	array of the Fourier transform of the derivation of the closure relation
MODULE Var__Bridge	
BridgeR,hQ,d	arrays and vectors relevant for the calculations of bridge functions for the MHNC- and RHNC-Closure relations

The main program **IE** calls only four different subroutines, which are **MENU**, **init__Algorithm**, **init__Method** and **Solve__2**.

MENU reads all parameters for the choices of closure relation, algorithm, dimension, etc from the command prompt. One can also choose between four different tasks:

- solution of integral-equations
- calculation of thermodynamics
- thermodynamic self-consistency
- phase coexistence

The subroutines **init__Algorithm** and **init__Method** initialize some important parameters for the start, as **init__Method** for exapmple reads **in__p** and **poten** produced by the chosen potential program, and renormalizes ρ to units of a, as it is defined in Eq. (3.2) to Eq. (3.4). As mentioned, if units of a shall not be used, one has to set $\rho = \rho$ instead of $\rho = \frac{3}{4\pi}\rho$.

init__Algorithm sets two frequently used arrays, namely *typtrans*(i, j) and *typtrans*(j, i) as follows:

```
DO i=1,NrTypes
    DO j=i,NrTypes
        k=k+1
        typtrans(i,j)=k
        typtrans(j,i)=k
    ENDDO
ENDDO
```

As one can see from the routine above, *typtrans* is an array of the dimension *NrTypes* x *NrTypes*, filled with a consecutive numbering of the combinations of components. For example for two components, with *NrTypes* $= 2$ we have *typtrans*$(1, 1) = 1$, *typtrans*$(1, 2) =$ *typtrans*$(2, 1) = 2$ and *typtrans*$(2, 2) = 3$. Furthermore **init__Algorithm** normalizes the particle densities $rn(i)$ to

$$\sum_i rn(i) = 1 \tag{3.11}$$

Lastly, **Solve__2** is the superordinate solving structure; It calls the chosen algorithmic subroutines as well as further subroutines required for the chosen task, like **Thermo** for the calculation of thermodynamics, **THD__SC** for thermodynamic selfconsistency, and **PHASED** for phase coexistence. If the closure relation **RHNC** is chosen, **Solve__2** calls the subroutine **RHNC** instead which is also defined in **IEM-new.f90**. **RHNC** runs the chosen algorithm before calling the bridge subroutines **TESTBR** and **BRIDGE** that are described down below.
Solve__2 also calculates the computation time which makes it possible to compare the efficiency of different algorithms.
The variable *iSucess*, set by the algorithmic subroutines is set to one if the algorithm has been successful and is set to zero as long as the iteration cycle does not converge.

If the algorithm exits with $iSucess = 0$, another subroutine named **SOLVER** is called, that rescales the potential by a multiplication with 7/8 and the parameter rnu by $rnu = rnu/2$ to nevertheless achieve convergence. It can be called in two different modes, '1' and '-1'. Then **SOLVER** calls the active algorithm again. It uses the parameters *isloop* and *isdepth* to evaluate the convergence of the system. Each time the program is scaled, *isdepth* will increase or decrease, depending on the mode. If it is greater than 6, the program stops, as well as for $isloop = 1$, which is the case, if a switch of the mode is performed more than two times.

3.3.2. Module STR-new.f90

This module contains most of the subroutines for all different algorithms and closure-relations. **STR-new.f90** is the new version of **SRUCTM.f90** because the subroutines FT_c_RtoQ, FT_Gamma_QtoR and FT_Gamma_RtoQ where generalized to higher dimensions now.

- **SUBROUTINE Potential_Separate** searches for a point in the potential function $\phi(r_{is})$ where $\phi(r_{i+1s})$ is larger than $\phi(r_{is})$. If there is no such point, the program stops with the message: 'NO SEPARATION OF PHI POSSIBLE'. The separation is performed as described in section 2.2 for the SMSA Closure relation.

- **SUBROUTINE init_Gillan** initializes the Gillan algorithm, first called by **init_Algorithm**. It also specifies the values $l1 = nod(NrBaF - 1)$ and $l2 = l1 + nod$, defining the length of the roof functions or the coarse part respectively. As mentioned in section 4.2, there are $n = NrBaF$ different roof functions P^i_β with a node-length of *nod*. Due to the fact that $P^i_\beta = 0$ for $i > l2 + nod$, the Jacobi matrix together with all other relevant functions only need $l2$ as an upper bound instead of the general number of grid points $N = 2^m$.

Furthermore the coordinates $B_{\alpha\beta}$ of the conjugate basis functions Q^i_α are defined here, as their matrix has only a dimension of $NrBaF$ x $NrBaF$.

- **FUNCTION P_BaF(j,i)** is a function, that specifies the roof functions P^i_j, for the Gillan algorithm, explained in section 2.3.2, where i is the running index for the grid points and j specifies the numbering index of the roof functions .

- **SUBROUTINE calc_a** calculates the weights a_α from the whole vector γ_{is} as described by Eq. (4.32) . It requires γ_{is} and the running index for the component number *ityp*. The output is a_α

- **SUBROUTINE calc_DeltaGamma** calculates the function $\Delta\gamma_{is}$ from Eq. (4.62). Its input values are γ_{is}, a_α and a storing vector for $\Delta\gamma_{is}$.

- **SUBROUTINE OZ** calculates $\tilde{\gamma}_{is}$ via the Ornstein-Zernike equation in k-space, using the vector c_{is}. For the special case of a one component system the result is directly given by the equation.

  For the more component case a system of linear equations has to be solved, in order to express the cross-terms $ga\tilde{m}ma_{ij}$ in terms of $\tilde{c}_{ij}$ in Eq. (2.5). For this task we use a standard Gaussian solving routine called **SUBROUTINE solve_LinEquSyst**, defined in the module **FFT.f90**.

  The only input-value for the subroutine **OZ** is the running index k of $\tilde{\gamma}_{is}$ (named 'l' in the program).

- **SUBROUTINE Derivation_OZ** calculated the derivative $\frac{\tilde{\gamma_{is}}}{\tilde{c_{is}}}$ via Eq. (4.53) by solving a linear equation system, such as for the subroutine **OZ**. It needs the function $\tilde{\gamma_{is}}$ and an index indicating the components as an input. The output is $\frac{\tilde{\gamma_{is}}}{\tilde{c_{is}}}$.

- **FUNCTION Derivation_Closure** provides the derivative of the closure relation $\frac{c_{is}}{\gamma_{is}}$, for each of the closure relations. It needs an index for both, the variable r and the components as an input.

- **SUBROUTINE calc_Jacobi** calculates the Jacobi matrix to obtain the new values for $\bar{a}_\alpha$ in the Newton iteration,as described in detail in chapter 4. This subroutine calls the subroutine **Derivation_OZ** and uses the function **Derivation_Closure**. The values for $\frac{\partial\gamma_{sj}}{\partial\gamma_{ti}}$ in expression Eq.(4.50) are stored in the vector *deri*, while for the Jacobi matrix as well as for the inverse Jacobi matrix we use *rJacobi* and *rJacobiInv*. They are both of the dimension $imaxJa = NrBaF$ * $NrIntTypes$ as the Jacobi matrix is of the dimension $NrBaF$ x $NrBaF$ for each component, see Eq. (4.43).

- **SUBROUTINE Gillan** finally performs the Gillan algorithm as described in section 2.3.2. At the beginning the increments of convergence for the coarse and the fine part, $CoarseDev = 10^{-9}$ and $FineDev = 10^{-8}$, are defined. $CoarseDev$ is the increment for Δa_α while $FineDev$ ends the iteration for $\Delta\gamma_i$.
  All important calls of the Gillan algorithm are shown in Figure 3.2.

  Further specifications of convergence are the parameters *isuccount* and $current\_CDEV$. $current\_CDEV$ is the sum of the difference between the weights of the roof functions a_α and $\bar{a_\alpha}$ for each component, used by the Gillan algorithm. If $current\_CDEV$ is greater than 1000, *isuccount* increases by one. If *isuccount* is greater than 3, *iSuccess* is set to zero, and the subroutine ends, followed by a call of **SOLVER(1)** by the surrounding subroutine **solve_2**.

- **SUBROUTINE LaMaVo** is the main subroutine for the algorithm of LaMaVo, as described in section 2.3.3. Similar to the Gillan algorithm, it uses the parameters $CoarseDev$ and $FineDev$ as indicators of convergence.

  In contrast to the Gillan algorithm, $CoarseDev$ is typically set to 10^{-5}, because it is compared to $\Delta\tilde{\gamma}_{is}$ in k-space, which is generally coarser than r-space, due to $dr \cdot dk = \pi/N$.

  A schematic overview of the algorithm is given in Figure 3.3.

  For the calculation of the factors $\tilde{p}_{i,ks}$), specified in Eq. 2.40, another subroutine, **calc_P** is called, as well as **calc_DeltaGammaQ** for the calculation of $\Delta\tilde{\gamma}_{is}$ in k-space.

  We need the old vector for $\tilde{\gamma}_{is}$ and $\tilde{c}_{is}$ as an input, while $\Delta\tilde{\gamma}_{is}$ is the output value. At the beginning of **calc_DeltaGammaQ** the subroutine **calc_cQ_new** calculates the new vector $\tilde{c}_{is}$, which also requests the old vector $\tilde{\gamma}_{is}$, $\tilde{c}_{is}$ as well as the index for k-space and the number of different components. For all subroutines of the LaMaVo algorithm the length of the coarse part is specified by the parameter $MaxIndexNewton$.

  Similar to the subroutine **Gillan**, the subroutine **LaMaVo** uses the parameter $current{-}CDEV$ and *isuccount* as further criteria of convergence: $current\_CDEV$ is the sum of $\Delta\tilde{\gamma}_{is}$ for each component. If this variable is greater than 10, the variable *isuccount* will increase by one and ends the subroutine **LaMaVo**, while setting $iSuccess = 0$. The subroutine will also stop, if the parameter *icount*, which counts the number of cycles performed by the subroutine, is greater than 25.

- **SUBROUTINE Ng** performs the algorithm of Ng as described in section 2.3.1. For the criterion of convergence the variable Dev is defined at the beginning. It will be compared with the variable $current\_Dev$ at the end of each circle, where $current\_Dev$ is the the actual difference between the Gamma-functions.

  If $current\_Dev$ is greater than 4000, or if the parameter *istep*, counting the number of cycles performed by **Ng**, is greater than 400, the algorithm sets $iSucess = 0$ and end immediately.

A schematic representation of **Ng** is shown by Figure 3.4.

For the first round we start with setting $d_{is}^{(n-2)} = d_{is}^{(n-1)} = d_{is}^{(n)} = 0$, as well as $\gamma_{is}^{0} = 0$, specified by Eq. 2.29 and Eq. **??**. After the calculation of the coefficient c_1 and c_2 used by the Eq. 2.34, a single iteration circle is performed on $\gamma_{is}^{(n+1)}$. The new value for $\gamma_{is}^{(n+1')}$ is directly compared with $\gamma_{is}^{(n)}$ and stored in $d_{is}^{(n)}$ after we set $d_{is}^{(n-1)} = d_{is}^{(n)}$ and $d_{is}^{(n-2)} = d_{is}^{(n-1)}$.

- **Subroutines of general use**

  Some subroutines are used by all three integral-equation solver algorithms. As they are very short and easy to understand, they are just listed here for completeness:

  The subroutines **SUBROUTINE Closure_MHNC, Closure_RY** and **Closure_HMSA** calculate g_{is} and need $\phi(r_{is})$, γ_{is} and, for storage, g_{is} or c_{is} (depending on the calling method) as an input. For the choice of 'RHNC' the Closure_MHNC is used as well, but the main solver **Solve_2** calls the optimizing routine 'RHNC' instead of the algorithms, to optimize the bridge functions, see subsection **BR-new.f90**.

  SUBROUTINE Closure sets the closure relation depending on the parameter $iMethod$ by calling the corresponding method, with c_{is} as an output vector.

  SUBROUTINES FT_c_RtoQ, FT_Gamma_QtoR and **FT_Gamma_RtoQ** perform the FFT- transformation for d dimensions. They are described in detail in section 4, as they where generalized two higher (odd) dimensions. They call some basic FFT-routines, **RSA** and **RCA**, to perform either a simple sine or cosine FFT. They need only the running index for the number of components as an input.

SUBROUTINE calc_PDF is called at the end of each algorithm, to finally calculate g_{is}, as the main loop of each algorithm is left with a fixed γ_{is} and c_{is}. So **calc_PDF** calls the chosen closure relation and stores g_{is}.

3.3.3. Module BR-new.f90

BR-new.f90 is the new version of the subroutine **BRIDGE.f90**, containing all relevant subroutines for the calculation of the bridge functions [1]. If either the method **MHNC** or **RHNC** is chosen, the bridge function extrapolates the **HNC** closure relation. Those Bridge diagrams are nearly the same for different potentials and can be derived from a potential with an already existing solution, such as the hard sphere potential. For **RHNC** the bridge functions are optimized by introducing the bridge functions of hard spheres; The diameters of these hard spheres is optimized as follows:
First we need to determine the hard sphere particle diameter σ. As described in the subsection 3.1.3 describing hard spheres , the packing fraction η is used instead of σ, as specified in table 3.1.3. Because a further condition is needed here, the free Helmholtz energy is minimized. So the determination of the bridge functions B_{is} is performed as follows:

- Determination of the free Helmholtz energy of the system and its derivative with respect to η.
- Calculation of the distribution function g_{is} for a fixed B_{is} by solving the integral equations.
- Minimization of the free Helmholtz energy with respect to η
- Cycle as long as convergence for η is reached, meaning that the equation of minimization has to be satisfied by the current η.

The bridge functions are calculated by the subroutine **BRSUB**. As the method of RHNC was not part of this thesis, the routine will not be described in detail here. It can be looked up in the thesis of Christian Libert [1], in chapter 4.2.4. Nevertheless, as the calculation of some coefficients for the bridge functions is performed in k-space, two FFT routines, called **FT__RtoQ** and **FT__QtoR**, that are introduced in **BR-new.f90**, where generalized to higher (odd) dimensions as well.

3.3.4. Module TESTBR.f90

The module **TESTBR.f90** only contains the subroutine **TESTBR** to optimize η in the above sense. It needs η as an input.

3.3.5. Module THERMO-new.f90

The module **THERMO-new.f90** defines all important functions that allow the calculation of the thermodynamic properties; One can choose them at the beginning. This new version of the old subroutine **Thermo.f90**, now contains the calculation of the excess energy U_2^{ex}, the pressure P and the compressibility χ_T, generalized to five and seven dimensions.

- **SUBROUTINE Thermo** is the main subroutine for the calculation of all thermodynamic properties, calling the subroutines for the calculation of U_2^{ex}, P, χ_T, the excess chemical potential μ^{ex} and, for the RHNC closure, the calculation of the free Helmholtz energy A. A schematic presentation of this routine is given by Figure 3.5.

- **SUBROUTINE print_Thermo** prints all final results of the thermodynamic properties.

- **SUBROUTINE U_EXC** calculates the excess energy as specified in Eq. (4.65). First a function f_{is} is defined, containing all r-dependencies of U_2^{ex} in Eq. (4.65), that will be introduced in section 3.3.5. To be specific, the function f_{is} in different dimensions reads:

  d=3:
  $$f_{is} = \phi(r_{is}) g_{is} (i\ dr)^2 \tag{3.12}$$
  d=5:
  $$f_{is} = \frac{4\pi}{3} \phi(r_{is}) g_{is} (i\ dr)^4 \tag{3.13}$$
  d=7:
  $$f_{is} = \frac{8\pi^2}{15} \phi(r_{is}) g_{is} (i\ dr)^6 \tag{3.14}$$
  where the pre factors need to be introduced as an additional factor to the normalization of the fast Fourier transformation in three dimensions. f_{is} is then integrated with the help of the subroutine **DARSIM**, using the Simpson method, defined in section 3.3.6.

- **SUBROUTINE Pressure** uses a method similar to the subroutine **U_EXC**, using now Eq. (4.67).

- **SUBROUTINE Compressibility** calculates the isothermal compressibility χ_T as defined in Eq.(4.75).

- **SUBROUTINE mu_excess and FUNCTION Bridge** calculate of the chemical potential μ. It was not used in this thesis, and was therefore not generalized to higher dimensions. For a derivation of the underlying equations we refer to further literature [2], as well as for the calculation of **Bridge** and **calc_A_RHNC** for the Helmholtz energy.

- **FUNCTION Derivation_f(r,dl)** and **FUNCTION f_interpolated(r)** call the subroutine **DERIV** and **DDVDIF**, both defined in **FFT.f90**, to derive the potentials in **Pressure** and **mu_excess** by r.

3.3.6. Module FFT.f90

The file **FFT.f90** is a collection of some basic routines: Any of those are important routines and functions for integration, derivation or for the FFT-transformations **RSA** and **RCA**. As both, their use and their input values are well described by the comments of the module **FFT.f90** itself, the most important functions are just listed here to provide a better orientation while reading the code:

- **SUBROUTINE RCA(MM, X, IX, Y, IY)**
  Describes a cosine Fast Fourier transformation of a real even function.
  M: Integer number, such that $N = 2^M$, with N being the number of grid points. It limits the grid size of the functions to a square number of 2. This limitation is the consequence of the fact that the FFT- algorithm, which recursively divides the range of the functions by a factor of two.
  X: Input vector of size N+1
  IX: step size for the increase of the input values, X(r*IX+1)
  Y: Output vector of size N+1
  IY: step size for the increase of the output values, Y(k*IY+1)

- **SUBROUTINE RSA(MM, X, IX, Y, IY)**

  Describes a sine fast Fourier transformation of an odd function.

- **SUBROUTINE solve_LinEquSyst(N,R,RES)**

  Solving of a system of linear equations, using the Gauss method.

  N: Dimension of the equation system

  R: Matrix of the dimension N x N

  RES: Vector for the storage of the result

- **SUBROUTINE calc_InverseMatrix(n,Matrix,res)**

  Calculation of an Inverse matrix. The variables are written in small letters, similar to their specification in the program.

  n: Dimension of the equation system

  Matrix: Matrix of the dimension n x n

  res: Vector for the storage of the result

- **SUBROUTINE DARSIM(N,DEL,A,RES)**

  Integration of a function $A(N)$ with N grid points, separated by an increment DEL, with the result stored in $RES(N)$

- **FUNCTION DERIV(F,X,DELTA,ABSCONV,RELCONV,ICONV)**

  Derivative of a function F with arguments X by an increment $DELTA$. The parameters $ABSCONV$ and $RELCONV$ are chosen to define the convergence of the routine. If the numerical difference between the last two steps of the derivation process is bigger than either $ABSCONV$ or $RELCONV$ multiplied with the last result, the parameter $Iconv$ is set to one. Otherwise the routine ends by setting $Iconv$ to zero and by writing the message 'DERIV NON CONVERGE' to the standard output line.

- **FUNCTION DDVDIF(F,X,N,Z,M2)**
  polynomial Interpolation for a function F with known values at X, using Newton interpolation
  F: Array of the given function
  X: Array of the given arguments
  N: Number of grid points
  Z: Argument on which the interpolation of the function should be performed
  M2: Input for the order of interpolations. It can not be chosen bigger than 10.

3.4. Output files

Computation time and thermodynamic properties (if chosen) are shown in the standard output line. Furthermore **IEM-new.f90** produces the output file **gamout**, that contains the functions $\gamma(r)$, the argument r and the array $g(r)$. The direct correlation function $c(r)$ is written to the file **c1**.

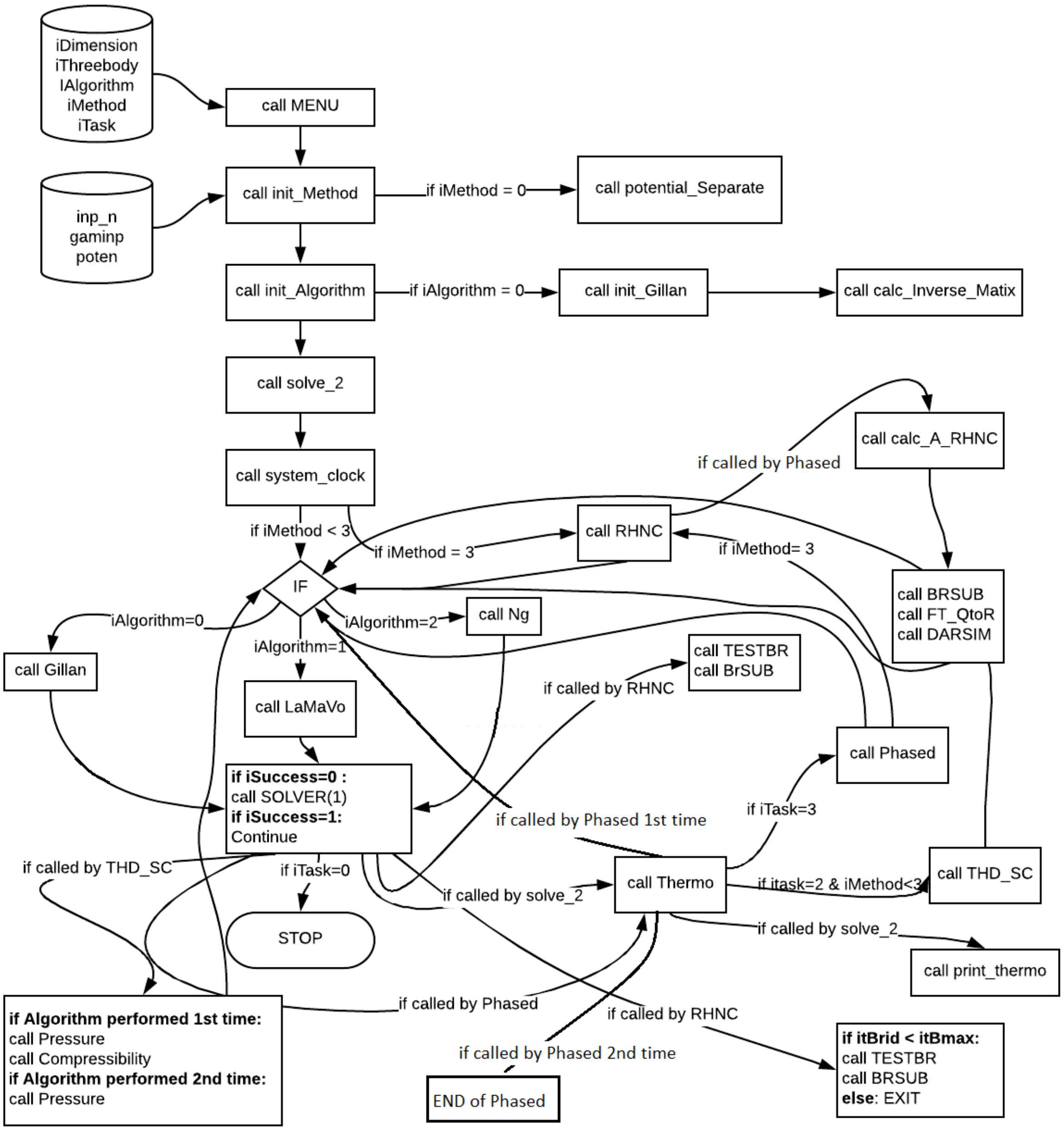

Figure 3.1: Program IE

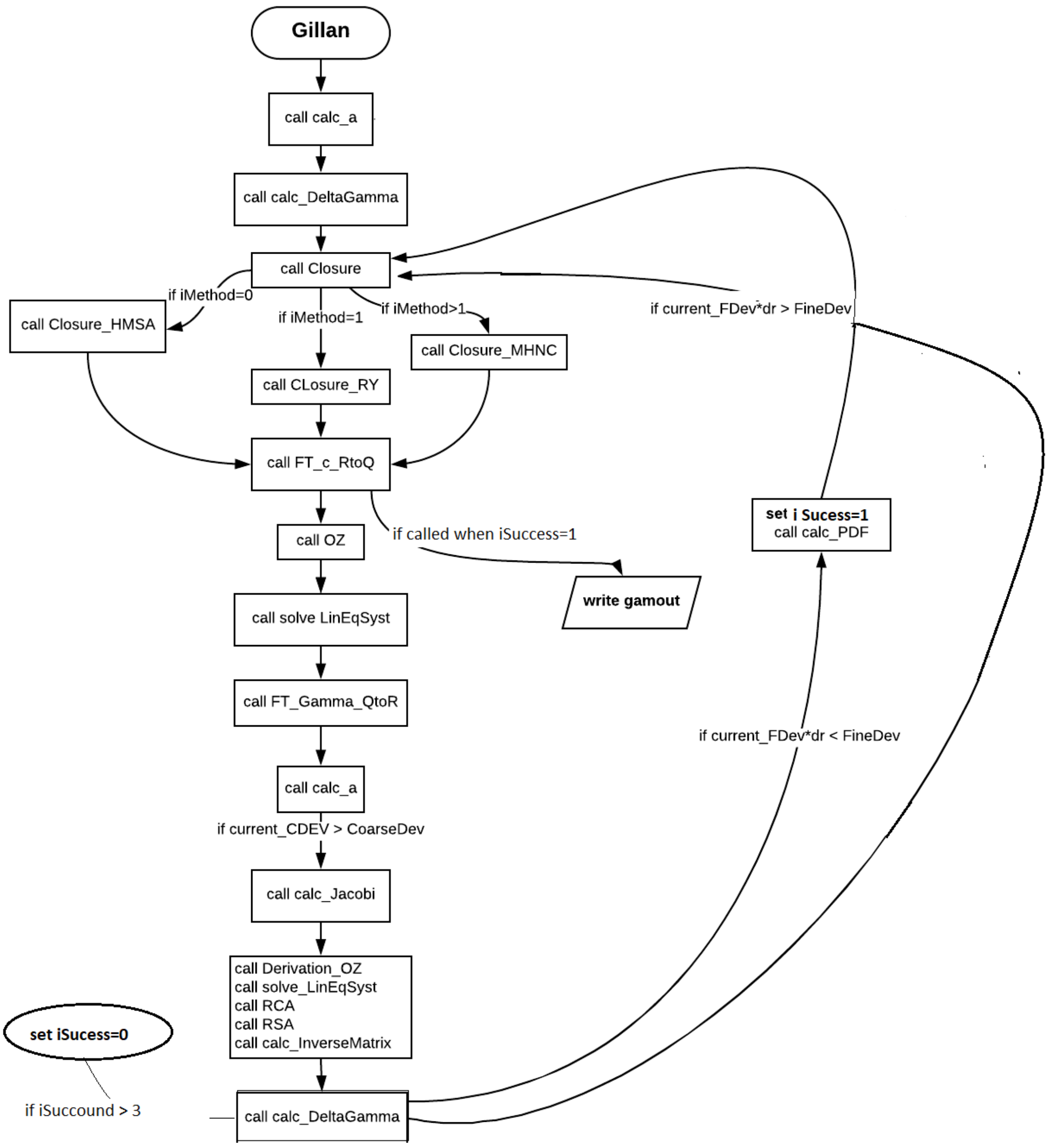

Figure 3.2: Gillan algorithm

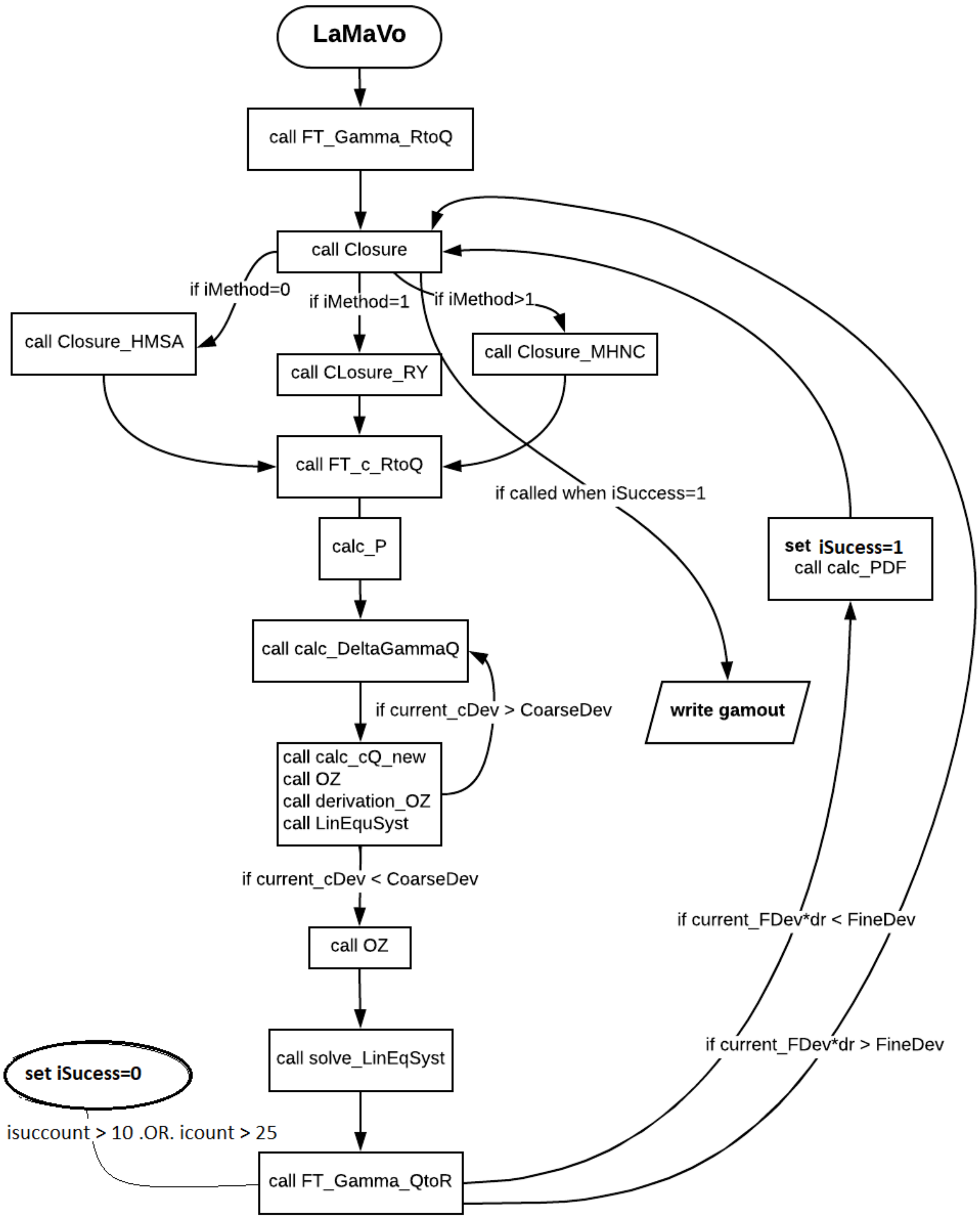

Figure 3.3: Algorithm of LaMaVo

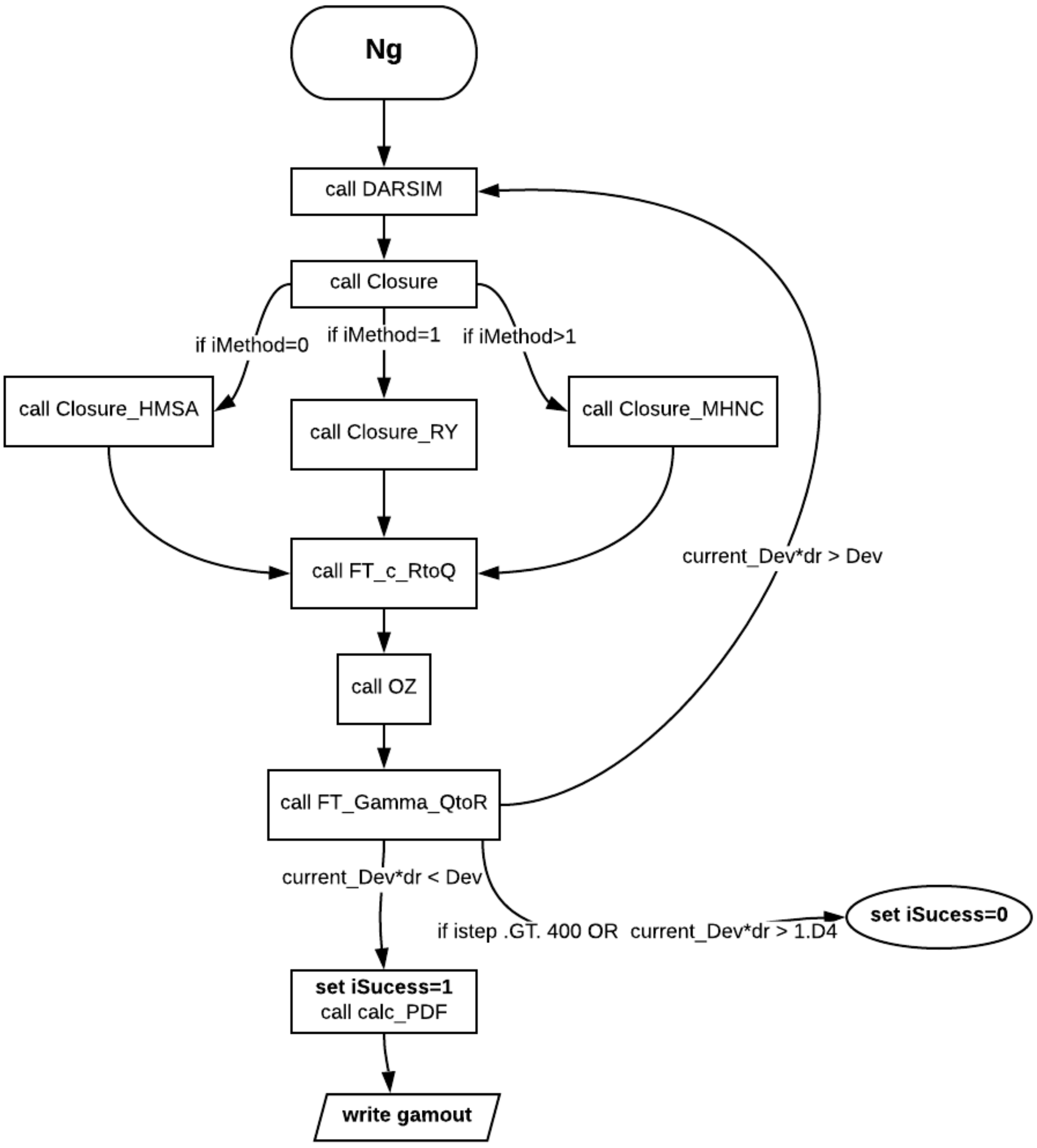

Figure 3.4: Algorithm of Ng

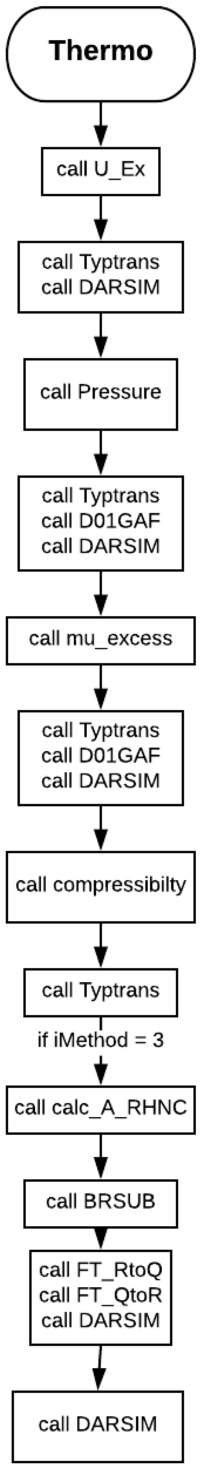

Figure 3.5: subroutine Thermo

4. Generalisation to higher dimensions

Proceeding to higher dimensions, we need to know how the assumption of a five or seven dimensional function have to be taken into account by the program. Luckily we deal throughout with radially symmetric functions, which means that most of the transformations will only apply to functions, which depend on one single argument, namely the distance r. As a result the basic gillan algorithm contains only two subroutines, where the dimensionality of the functions can not be neglected: The Fast Fourier Transformations(FFT) performed in the basic picard circle and the Jacobi Transformation.

4.1. Fast Fourier Transformation in higher dimensions

A general consideration of radially symmetric functions the Fast Fourier Transformation leads to Hankel-transformations, which is the reason why we will focus on odd dimensions only [2].

4.1.1. Hyper-spherical coordinates in $\mathbb{R}^n$

First we have to transfer Euclidean n-dimensional space, with the coordinates

$$\{x_1, x_2, ...x_n\} \tag{4.1}$$

to hyper-spherical coordinates

$$\begin{aligned} \{r, \theta_1, \theta_2, ...\theta_{n-2}, \phi\}\,; \quad & 0 \le \theta_\alpha \le \pi, \quad \alpha \in \{1, ...n-2\} \\ & 0 \le \phi \le 2\pi, \\ & 0 \le r \le \infty \end{aligned} \tag{4.2}$$

with

$$\begin{aligned}
x_1 &= r\cos(\theta_1) \\
x_2 &= r\sin(\theta_1)\cos(\theta_2) \\
&... \\
x_\alpha &= r\sin(\theta_1)\ \ ...\ \ sin(\theta_{\alpha-1})\cos(\phi_\alpha) \\
&... \\
x_{n-1} &= r\sin(\theta_1)\ \ ...\ \ \sin(\theta_{n-2})\cos(\phi) \\
x_n &= r\sin(\theta_1)\ \ ...\ \ \sin(\theta_{n-2})\sin(\phi).
\end{aligned} \tag{4.3}$$

4.1.2. Volume of an n-dimensional sphere

By calculating the functional determinant of the above transformations, we find an expression for the infinitesimal volume element:

$$d^n r = \prod_{\alpha=1}^{n} dx_\alpha = r^{n-1}\prod_{\alpha=1}^{n-2}\sin^{n-\alpha-1}(\theta_\alpha)d\theta_\alpha d\phi. \tag{4.4}$$

This leads us to the volume V_n of an n-dimensional sphere with radius R:

$$V_n = \int_{\phi=0}^{2\pi} d\phi \int_{r=0}^{R} r^{n-1}dr \int_{\theta_1=0}^{\pi}\sin^{n-2}(\theta_1)d\theta_1\ ...\ \int_{\theta_{n-2}=0}^{\pi}\sin^{n-2}(\theta_{n-2})d\theta_{n-2} \tag{4.5}$$

For the explicit integration of Eq.(4.5), we need the following expressions:

$$\begin{aligned}
\int_{\theta=0}^{\pi}\sin^{2l}(\theta)d\theta &= \pi\frac{(2l-1)!}{(2l)!} \\
\int_{\theta=0}^{\pi}\sin^{2l+1}(\theta)d\theta &= \pi\frac{(2l)!}{(2l+1)!} \\
&\qquad\qquad l \in \mathbb{N}.
\end{aligned} \tag{4.6}$$

Together with the following relations of the Gamma-function

$$\begin{aligned} \Gamma(n+1) &= n! \\ \Gamma(1) &= 1 \\ \Gamma\left(\frac{1}{2}\right) &= \sqrt{\pi} \end{aligned} \tag{4.7}$$

we obtain

$$V_n = \frac{S_n R}{n}; \quad S_n = \frac{n\pi^{n/2}}{\Gamma(n/2+1)} R^{n-1}. \tag{4.8}$$

4.1.3. Fourier transforms of radially symmetric functions

For a general function, $f(\vec{r})$, that depends on $\vec{r} \in \mathbb{R}^n$ we define its Fourier transform, $\tilde{f}, (\vec{k})$ via

$$\tilde{f}(\vec{k}) = \int e^{i\vec{k}\vec{r}} f(\vec{r}) d^n r. \tag{4.9}$$

If we consider radially symmetric functions, we are free to choose our coordinate system such that $\vec{k} = (k, 0, ...0)$ is the polar axis:

$$\vec{k}\vec{r} = \sum_{\alpha}^{n} k_\alpha x_\alpha = kr\cos(\theta_1). \tag{4.10}$$

This, together with Eq. (4.4) for the volume element dr, transforms the relation Eq. 4.9 to the following expression of Eq.(4.9)

$$\begin{aligned} \tilde{f}(k) = 2\pi \int_0^\infty r^{n-1} f(r) dr \int_{\theta_1=0}^{\pi} e^{ikr\ \cos(\theta_1)} \sin^{n-2}(\theta_1) d\theta_1 \\ \times \prod_{\alpha=1}^{n-3} \int_{\theta_\alpha}^{\pi} \sin^{n-\alpha-2}(\theta_\alpha) d\theta_\alpha \end{aligned} \tag{4.11}$$

In an attemt to avoid complicated Hankel-transformations, we restrict ourselves to odd dimensional functions, with $n = 2l + 1$.

Together with Eq. (4.8) for the surface $S_n(r)$ we can integrate the product, which appears as a last factor of Eq. (4.11)

$$\tilde{f}(k) = \frac{(2\pi)^{\frac{n-1}{2}}}{(n-3)!} \int_{r=0}^{\infty} f(r) X_n(k,r) r^{n-1} dr;$$

$$X_n(k,r) = \int_{\theta=0}^{\pi} e^{ikr\ \cos(\theta)} \sin^{n-2}(\theta) d\theta. \tag{4.12}$$

We can now explicitly solve the integral for X_n, using the relation

$$\int_{\theta=0}^{\pi} e^{ia\cos(\theta)} \sin^{2\nu}(\theta) d\theta = \sqrt{\pi} \left(\frac{2}{a}\right)^{\nu} \Gamma(\nu + \frac{1}{2}) J_\nu(a) \tag{4.13}$$

where $J_\nu(a)$ is the ν-th order Bessel-function of the first kind, i.e.,

$$J_\nu(a) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\Gamma(m+\nu+1)} \left(\frac{a}{2}\right)^{2m+\nu}. \tag{4.14}$$

By comparing Eqs. (4.13) and (4.12), we find that $\nu = (n-2)/2$. Since n is odd, ν is half-integer. We thus obtain

$$\int_{\theta=0}^{\pi} e^{ikr\ \cos(\theta)} \sin^{n-2}(\theta d\theta) = \sqrt{\pi} \left(\frac{2}{kr}\right)^{\frac{n-2}{2}} \Gamma\left(\frac{n-1}{2}\right) J_{\frac{n-2}{2}}(kr). \tag{4.15}$$

Eq. (4.15) can be simplified by using spherical Bessel-functions $j_\nu(a)$, which are related to the $J_\nu(a)$ via:

$$j_\nu(a) = \sqrt{\frac{\pi}{2a}} J_{\nu+\frac{1}{2}}(a). \tag{4.16}$$

This leads to

$$J_{\frac{n-2}{2}}(kr) = \sqrt{\frac{2kr}{\pi}} j_{\frac{n-3}{2}}(kr). \tag{4.17}$$

Putting these results together, we finally obtain

$$\tilde{f}(k) = \frac{2^{n-1}\pi^{\frac{n-1}{2}}}{(n-3)!}\Gamma\left(\frac{n-1}{2}\right)\int_{r=0}^{\infty} f(r) r^{n-1} \frac{j_{\frac{n-3}{2}}(kr)}{(kr)^{\frac{3-n}{2}}} dr. \tag{4.18}$$

4.1.4. Application of the Fourier transformation to the special cases of d=3,5,7

Using Eq. (4.18) for the special case of three, five and seven dimensions, we find:

$$\begin{aligned}
d = 3: \quad & \tilde{f}(k) = 4\pi \int_0^\infty f(r) \tfrac{\sin(kr)}{k} r dr; \\
d = 5: \quad & \tilde{f}(k) = 8\pi^2 \int_0^\infty f(r) (\tfrac{\sin(kr)}{k^3} r - \tfrac{\cos(kr)}{k^2} r^2) dr; \\
d = 7: \quad & \tilde{f}(k) = 16\pi^3 \int_0^\infty f(r) (3\tfrac{\sin(kr)}{k^5} r - \tfrac{\sin(kr)}{k^3} r^3 - 3\tfrac{\cos(kr)}{k^4} r^2) dr.
\end{aligned} \tag{4.19}$$

The related inverse Fourier transformations read:

$$
\begin{aligned}
d=3: &\qquad f(r)=\tfrac{1}{2\pi^2}\int_0^\infty \tilde{f}(k)\tfrac{\sin(kr)}{r}kdk;\\
d=5: &\qquad f(r)=\tfrac{1}{4\pi^3}\int_0^\infty \tilde{f}(k)(\tfrac{\sin(kr)}{r^3}k-\tfrac{\cos(kr)}{r^2}k^2)dk;\\
d=7: &\qquad f(r)=\tfrac{1}{8\pi^4}\int_0^\infty \tilde{f}(k)(3\tfrac{\sin(kr)}{r^5}k-\tfrac{\sin(kr)}{r^3}k^3-3\tfrac{\cos(kr)}{r^4}k^2)dk.
\end{aligned}
\tag{4.20}
$$

Thus the Fourier transforms in the higher dimensional spaces reduce to linear combinations of the sine- and cosine transforms of the function f, and the integrals are weighted by suitable powers of k and r.
For numerical use those equations have to be discretized. FFTs are used in the following modules and subroutines:

Module STR-new.f90

- subroutine FT_c_RtoQ
- subroutine FT_Gamma_QtoR
- subroutine FT_Gamma_RtoQ

Module BR-new.f90

- subroutine FT_RtoQ
- subroutine FT_QtoR

The cases k=0 and r=0 have to be considered separately for both, the direct and the inverse transformations. Here the rule of de l'Hospital is used to calculate the respective Fourier transforms at vanishing r or k. For the three different cases of dimensionality, one obtains for $f(r)=0$:

d=3:

$$\begin{aligned} f(r=0) &= \lim_{k\to 0} 4\pi dr \textstyle\sum_{r=0}^{N} f(k)\frac{\sin(kr)}{k} r \\ &= \lim_{k\to 0} 4\pi dr \textstyle\sum_{r=0}^{N} f(k)\frac{\cos(kr)}{1} r^2 = 4\pi dr \sum_{r=0}^{N} f(k) r^2 \quad . \end{aligned} \tag{4.21}$$

d=5:

For five dimensions the rule of de l´Hospital is a little bit more complicated, one obtains:

$$\begin{aligned} f(r=0) &= \lim_{k\to 0} 8\pi^2 dr \textstyle\sum_{r=0}^{N} f(k)\left(\frac{\sin(kr)}{k^3} r - \frac{\cos(kr)}{k^2} r^2\right) \\ &= \lim_{k\to 0} 8\pi^2 dr \textstyle\sum_{r=0}^{N} f(k)\frac{\sin(kr)r - \cos(kr)r^2 k}{k^3} \quad . \end{aligned} \tag{4.22}$$

Now the Taylor approximations for sine and cosine are used:

$$\sin(kr) \approx kr - \frac{1}{6}k^3 r^3 \tag{4.23}$$

$$\cos(kr) \approx 1 - \frac{k^2 r^2}{2}. \tag{4.24}$$

Thus $f(r=0)$ finally reads:

$$f(r=0) = 8\pi^2 dr \sum_{r=0}^{N} f(k)\frac{r^4}{3} \tag{4.25}$$

d=7:

Similar to the five dimensional case, we obtain an expression Due to the fact, that most of the low-order terms vanish, one more term of the Taylor expansion is needed for both sine and cosine:

$$\sin(kr) \approx kr - \frac{1}{6}k^3r^3 + \frac{1}{120}k^5r^5 \tag{4.26}$$

$$\cos(kr) \approx 1 - \frac{k^2r^2}{2} + \frac{1}{24}k^4r^4. \tag{4.27}$$

We finally obtain:

$$f(r=0) = 16\pi^3 dr \sum_{r=0}^{N} f(k)\frac{r^6}{15} \tag{4.28}$$

Of course we can use Eqs. (4.21), (4.25) and (4.28) for the inverse transformations as well, using instead $\frac{1}{2\pi^2}$, $\frac{1}{4\pi^3}$ and $\frac{1}{8\pi^4}$ as pre-factors for each belonging dimension.

4.2. The Gillan algorithm and the Jacobi transformation in different dimensions

There is another subroutine for which the dimensionality of the problem is of relevance, namely the Jacobi transformation used in the Gillan algorithm. The main idea of the Gillan algorithm [5] is to seprarate $\gamma(r)$ into a coarse and a fine part with orthogonal spacial subspaces. While the coarse part covers the main structure of $\gamma(r)$, the fine one represents small fluctuations around the main part.

As introduced in section 2.3.2, we use in the following index notation and discrete vectors instead of functions, for the reason of a direct comparability with the program. We use i,j and m as counters of grid points in r- or k-space, while s and t stand for the running index of different particle interactions, meaning that combinations of components are counted by this variable. So for two components a and b the running index s could have three different values, namely $s = 1$, representing aa, $s = 2$, representing $ab = ba$ and

$s = 3$, representing bb.

4.2.1. Gillan algorithm

In detail the separation of $\gamma(r)$ in the above sense is performed as follows [5] :

$$\gamma_{si} = \sum_{\alpha} a_{s\alpha} P^i_{s\alpha} + \Delta\gamma_{si} \tag{4.29}$$

$P^i_{s\alpha}$ are the so called basis functions of the coarse part, where the $a_{s\alpha}$ are the expansive coefficients. $\Delta\gamma_{si}$ represents the fine part of $\gamma(r)$. α counts the number of basis functions, that are shaped such that the coarse part just covers a region of relatively small r-values, where the potential is significantly different from zero. As the subspaces of the coarse and the fine part are assumed to be orthogonal, the following equation must be satisfied:

$$\sum_i P^i_{s\alpha} \Delta\gamma_{si} = 0; \forall \alpha, s \tag{4.30}$$

For the basis functions $P^i_{s\alpha}$, that shall be each orthogonal as well, we choose a discrete version of the so called roof functions. Those functions are shaped like a triangle, with its tights meeting in the so called node point. If we have n different basis functions $P^i_{s\alpha}$ ($\alpha = 1, 2, ...n$), we have to choose the location of n different nodes i_n, that all have a height of 1. For example if we choose $i_n = 10$ the first node is located at $i = 10$ (or $r = 0.1$ for $dr = 0.01$).

$$\begin{aligned} P^i_{s\alpha} = \quad & 0 & \text{for} \quad & 1 \leq i \leq i_{s\alpha-2} \\ & (i - i_{s\alpha-2})/(i_{s\alpha-1} - i_{s\alpha-2}) & \text{for} \quad & i_{s\alpha-2} \leq i \leq i_{s\alpha-1} \\ & (i_{s\alpha} - i)/(i_{s\alpha} - i_{s\alpha-1}) & \text{for} \quad & i_{s\alpha-1} \leq i \leq i_{s\alpha} \\ & 0 & \text{for} \quad & i_{s\alpha} \leq i \leq N \end{aligned} \tag{4.31}$$

Figure 4.1 shows a typical example for roof functions, $P^i_{s\alpha}$. Here we use $n = 6$ with a length of 10 sequences of i between the nodes.

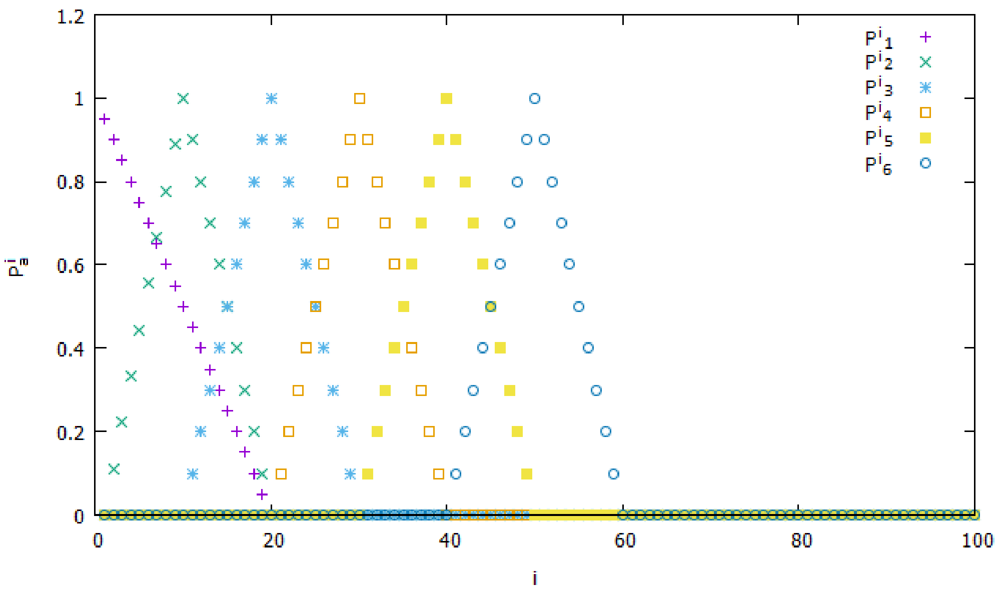

Figure 4.1: Possible choice of roof functions for $n = 6$, $i_n = 10$
Details are given by the descriptions above.

For convergence both $a_{s\alpha}$ and $\Delta\gamma_{si}$ have to converge. The Gillan algorithm in particular tries to accomplish that by performing a Newton-Raphson iteration on the coarse part followed by a basic Picard iteration on the fine part. After choosing an initial value for γ_{si} the algorithm passes the following steps:

1) Picard iteration from γ_{si} to γ'_{si}

2) Calculation of $a'_{s\alpha}$ from γ'_{si} by

$$a'_{s\alpha} = \sum_i Q^i_\alpha \gamma'_i \tag{4.32}$$

where $Q^i_\alpha \gamma'_i$ are the conjugated basis functions, that are derived by:

$$Q^i_{a\alpha} = \sum_{\beta} B_{s\alpha,s\beta} P^i s\beta \tag{4.33}$$

3) Calculation of $d_{s\alpha} = |a_{s\alpha} - a^{'}_{s\alpha}|$, If $d_{s\alpha}$ for each $\alpha = 1, ...n$ is not small enough, a coarse step is performed by a Newton-Raphson step iteration, with the aim to minimize $d_{s\alpha}(a_{s\alpha})$ to find its zeros:

$$\overline{a}_{s\alpha} = a_{s\alpha} - \sum_{t\beta} J^{-1}_{s\alpha,t\beta} d_{t\beta} \tag{4.34}$$

where

$$J^{-1}_{s\alpha,t\beta} = \frac{\partial d_{s\alpha}}{\partial d_{t\beta}} \ . \tag{4.35}$$

Obviously, to find $\overline{a}_{s\alpha}$, one has to calculate $J^{-1}_{s\alpha,t\beta}$, which is done as follows [5] :

The coefficients $B_{s\alpha,s\beta}$ of Eq. (4.33) are chosen such that

$$B^{-1}_{s\alpha,s\beta} = \sum_{i} P^i_{s\alpha} P_{s\beta} \ , \tag{4.36}$$

and hence

$$\sum_{i} Q^i_{s\alpha} P^i_{s\beta} = \delta_{\alpha\beta} \ , \tag{4.37}$$

because the basis functions $P_{s\beta}$ are orthogonal, as mentioned above. As a result $\Delta\gamma_{si}$ is orthogonal to $Q^i_{s\alpha}$ as well, so we obtain the expression

$$a_{s\alpha} = \sum_{i} Q^i_{s\alpha} \gamma_{si} \ . \tag{4.38}$$

Due to the basic chain-rule and Eq. (4.33) to (4.38), the inverse of the Jacobi matrix can be calculated via:

$$J^{-1}_{s\alpha,t\beta} = \frac{\partial d_{s\alpha}}{\partial d_{t\beta}} = \delta_{s\alpha,t\beta} - \sum_{ij} Q^{j}_{s\alpha} \frac{\partial \gamma'_{sj}}{\partial \gamma_{ti}} P^{i}_{t\beta} \tag{4.39}$$

with

$$\frac{\partial \gamma'_{sj}}{\partial \gamma_{ti}} = \sum_{m=0} \frac{\partial \gamma'_{sj}}{\partial \tilde{\gamma}_{sm}} \frac{\partial \tilde{\gamma}_{sm}}{\partial \tilde{c_{tm}}} \frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} \frac{\partial c_{ti}}{\partial \gamma_{ti}} \ . \tag{4.40}$$

4.2.2. Calculation of the Jacobi matrix

The Jacobi matrix has the dimension n x n x $NrIntType$ x $NrIntType$, where $NrIntType$ denotes the number of different component combinations. Typically we choose n between 6 and 12. To calculate each of the four factors in Eq. (4.40) we need explicit expressions [5] for $\gamma'_{sj}(\tilde{\gamma_{sm}})$, $\tilde{\gamma_{sm}}(\tilde{c_{tm}})$, $\tilde{c_{tm}}(c_{ti})$ and $c_{ti}(\gamma_{ti})$.
$c_{ti}(\gamma_{ti})$ can be obtained from closure relations, hence in a first step the derivative of the chosen closure relation by γ_{ti} has to be calculated.
Secondly, the Fourier transform of the OZ-equation yields an expression for $\tilde{\gamma}_{sm}(\tilde{c}_{tm})$. In the one component case we find the expressions:

$$\tilde{\gamma}_m = \frac{\rho \tilde{c}_m^2}{1 - \rho \tilde{c}_m} \tag{4.41}$$

with the derivative:

$$\frac{\partial \tilde{\gamma}_m}{\partial \tilde{c}_m} = (\frac{2\rho \tilde{c}_m}{1 - \rho \tilde{c}_m}) + (\frac{\rho \tilde{c}_m}{1 - \rho \tilde{c}_m})^2 \ . \tag{4.42}$$

For the more-component case the derivation of the OZ-equation is a little bit more complicated. We need to solve a linear equation system, performed by the subroutine **Derivation_OZ**, as described in section 3.3.

Expression $\gamma'_{sj}(\tilde{\gamma_{sm}})$ and $\tilde{c_{tm}}(c_{ti})$ are obtained via the Fourier transformation. This is the reason why there is a formal difference between those expressions in different dimensions. In three dimensions one finds the relation:

$$\frac{\partial \gamma'_{sj}}{\partial \tilde{\gamma}_{sm}} = \frac{dk}{2\pi^2 r_{sj}} k_{sm} \sin(k_{sm} r_{sj}) \tag{4.43}$$

$$\frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = \frac{4\pi dr}{k_{tm}} r_{ti} \sin(k_{tm} r_{ti}) \quad . \tag{4.44}$$

There is basically no difference between k_{sm} and k_{tm} or r_{si} and r_{ti}, because the distinction between the components is only relevant for the functions themselves, but not for their arguments r and k, that are part of the same grid. We will therefore suppress the component indices s and t for the arguments k and r in the following equations. Eq. 4.43 and Eq. 4.44 now leads us to

$$\frac{\partial \gamma'_{sj}}{\partial \tilde{\gamma}_{sm}} \frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = \frac{dr dk r_i}{r_j \pi} \sin(k_m r_i) \sin(k_m r_j) \quad . \tag{4.45}$$

For a better applicability of the algorithm containing the Fast Fourier transformation, we need some basic sine and cosine relations:

$$\cos(x_1) - \cos(x_2) = -2\sin(\frac{x_1 + x_2}{2}) \sin(\frac{x_1 - x_2}{2}) \tag{4.46}$$

$$\cos(x_1) + \cos(x_2) = 2\cos(\frac{x_1 + x_2}{2}) \cos(\frac{x_1 - x_2}{2}) \tag{4.47}$$

$$\sin(x_1) - \sin(x_2) = 2\cos(\frac{x_1 + x_2}{2}) \sin(\frac{x_1 - x_2}{2}) \tag{4.48}$$

$$\sin(x_1) + \sin(x_2) = 2\sin(\frac{x_1 + x_2}{2}) \cos(\frac{x_1 - x_2}{2}) \; . \tag{4.49}$$

In **three dimensions** only Eq. (4.46) is relevant. Choosing $x_1 + x_2/2 = k_m r_i$ and $x_1 - x_2/2 = k_m r_j$ leads to $x_2 = k_m r_i + k_m r_j$ and $x_1 = k_m r_i - k_m r_j$ and therefore:

$$\frac{\partial \gamma'_{sj}}{\partial \tilde{\gamma}_{sm}} \frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = \frac{drdkr_i}{r_j \pi} \cos(k_m r_i - k_m r_j) - \cos(k_m r_i + k_m r_j) \; . \tag{4.50}$$

For the case j=0 (corresponding to r=0) we obtain

$$\frac{\partial \gamma'_{s0}}{\partial \tilde{\gamma}_{tm}} \frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = \frac{2drdkr_i}{\pi} \sin(k_m r_i) k_m \; . \tag{4.51}$$

Here we used Eq. (4.43)along with the rule of de l'Hospital, thus

$$\lim_{j \to 0} \frac{dk}{2\pi^2 r_j} k_m \sin(k_m r_j) = \frac{dk}{2\pi^2 r_j} k_m^2, \tag{4.52}$$

from which Eq. (4.51) straightly follows.

The case of **five dimensions** is a little bit more complicated, but follows essentially the same ideas. The terms now read:

$$\frac{\partial \gamma'_{sj}}{\partial \tilde{\gamma}_{sm}} = \frac{dk}{4\pi^3} \frac{k_m \sin(k_m r_j)}{r_j^3} - \frac{k_m^2 \cos(k_m r_j)}{r_j^2} \tag{4.53}$$

$$\frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = 8\pi^2 dr \frac{r_i \sin(k_m r_i)}{k_m^3} - \frac{r_i \cos(k_m r_i^2)}{k_m^2} \; . \tag{4.54}$$

This leads to

$$\frac{\partial \gamma'_{sj}}{\partial \tilde{\gamma}_{sm}}\frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = \left(\frac{dk}{4\pi^3}\frac{k_m \sin(k_m r_j)}{r_j^3} - \frac{k_m^2 \cos(k_m r_j)}{r_j^2}\right) \left(8\pi^2 dr \frac{r_i \sin(k_m r_i)}{k_m^3} - \frac{r_i \cos(k_m r_i^2)}{k_m^2}\right) . \tag{4.55}$$

A similar consideration as the one for three dimensions leads to:

$$\begin{aligned}\frac{\partial \gamma'_{sj}}{\partial \tilde{\gamma}_{sm}}\frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = \frac{drdk}{\pi}&\frac{r_i^2}{r_j^3 k_m}(\sin(k_m r_j + k_m r_i) + \sin(k_m r_j - k_m r_i))+\\ &\frac{r_i}{r_j^2 k_m}(\sin(k_m r_j + k_m r_i) - \sin(k_m r_j - k_m r_i))+\\ &\frac{r_i^2}{r_j^2}(\cos(k_m r_j + k_m r_i) + \cos(k_m r_j - k_m r_i))+\\ &\frac{r_i}{r_j^3 k_m^2}(\cos(k_m r_j + k_m r_i) - \cos(k_m r_j - k_m r_i))\end{aligned} \tag{4.56}$$

for j=0, corresponding to r=0, we obtain

$$\frac{\partial \gamma'_{s0}}{\partial \tilde{\gamma}_{sm}}\frac{\partial \tilde{c}_{tm}}{\partial c_{ti}} = \frac{2drdk\, k_m^4}{3\pi}\left(\frac{\sin(k_m r_i)}{k_m^3} r_i - \frac{\cos(k_m r_i)}{k_m^2} r_i^2\right) \tag{4.57}$$

To calculate the final sum in Eq. (4.40) we consider $\frac{\partial \tilde{\gamma}_{sm}}{\partial \tilde{c}_{tm}}$ as the Fourier transform of $\frac{\partial \gamma'_{sj}}{\partial \gamma_{ti}}$, as one can see from the following expressions for three and for five dimensions:

d=3:

$$\frac{\partial \gamma'_{sj}}{\partial \gamma_{ti}} = \sum_{m=0}^{N} \frac{drdk r_i}{r_j \pi}(\cos(k_m r_i - k_m r_j) - \cos(k_m r_i + k_m r_j))\frac{\partial \tilde{\gamma}_{sm}}{\partial \tilde{c}_{tm}}\frac{\partial c_{ti}}{\partial \gamma_{ti}} \tag{4.58}$$

and **d=5:**

$$\begin{aligned}\frac{\partial\gamma'_{sj}}{\partial\gamma_{ti}} = \sum_{m=0}^{N}\frac{drdk}{\pi}\Bigg(&\frac{r_i^2}{r_j^3k_m}(\sin(k_mr_j+k_mr_i)+\sin(k_mr_j-k_mr_i))+\\ &\frac{r_i}{r_j^2k_m}(\sin(k_mr_j+k_mr_i)-\sin(k_mr_j-k_mr_i))+\\ &\frac{r_i^2}{r_j^2}(\cos(k_mr_j+k_mr_i)+\cos(k_mr_j-k_mr_i))+\\ &\frac{r_i}{r_j^3k_m^2}(\cos(k_mr_j+k_mr_i)-\cos(k_mr_j-k_mr_i))\Bigg)\frac{\partial\tilde{\gamma}_{sm}}{\partial\tilde{c}_{tm}}\frac{\partial c_{ti}}{\partial\gamma_{ti}}\end{aligned} \quad (4.59)$$

where $\frac{\partial c_{ti}}{\partial\gamma_{ti}}$ can be calculated by the Closure relation. For r=0 we have

d=3

$$\frac{\partial\gamma'_{sj}}{\partial\gamma_{ti}} = \sum_{m=0}^{N}\frac{2drdkr_i}{\pi}\sin(k_mr_i)k_m\frac{\partial\tilde{\gamma}_{sm}}{\partial\tilde{c}_{tm}}\frac{\partial c_{ti}}{\partial\gamma_{ti}} \quad (4.60)$$

and

d=5

$$\frac{\partial\gamma'_{sj}}{\partial\gamma_{ti}} = \sum_{m=0}^{N}\frac{2drdk\,k_m^4}{3\pi}\left(\frac{\sin(k_mr_i)}{k_m^3}r_i-\frac{\cos(k_mr_i)}{k_m^2}r_i^2\right)\frac{\partial\tilde{\gamma}_{sm}}{\partial\tilde{c}_{tm}}\frac{\partial c_{ti}}{\partial\gamma_{ti}}\,. \quad (4.61)$$

So for three dimensions for example, the subroutine just has to calculate the sum by performing two cosine FFTs or one sine FFT for the case r=0 in Eq. (4.60). The inverse of the Jacobi matrix then directly follows from Eq. (4.39).

We deliberately abstain from a consideration of seven dimensions here, as there is no difference between the use of the calculation of the Jacobi matrix for three or for higher dimensions concerning both the Gillan and the LaMaVo algorithm. This is because the fine part is always a further control sequence for the coarse part. So the coarse part just needs to produce a satisfying $d_{s\alpha}$ to provide a useful starting value for the fine part and it is only a matter of speed, when the calculated $\bar{a}_{s\alpha}$ reaches $a'_{s\alpha}$.

Because the derivation of $\frac{\partial\gamma'_{sj}}{\partial\gamma_{ti}}$ is much slower using Eq. (4.56) and Eq. (4.57) the same normalized transformation is used for all dimensions in the Jacobi-subroutine.

4) After one iteration of the coarse part we set $a_{s\alpha} = \bar{a}_{s\alpha}$ and calculate $d_{s\alpha}$. If this quantity is not small enough, another iteration of the coarse part has to be performed until the coarse part has converged.

5) When the previous step has been successfully achieved convergence, $\Delta\gamma'_{si}$ is calculated via

$$\Delta\gamma'_{si} = \gamma'_{si} - \sum_{\alpha} P^i_{s\alpha} a_{s\alpha} \; ; \tag{4.62}$$

This quantity is compared to the initial value of $\Delta\gamma_{si}$. If the difference is not small enough, a new Picard iteration is performed and the loop starts from step 1) again.

4.3. Thermodynamics

Now we move on to the equations to calculate the thermodynamic properties in higher dimensions. To be more specific, we focus on the excess energy U_2^{ex}, the pressure P, and the compressibility χ_T, which play an important role for the specification of different materials.

4.3.1. Excess energy

For U_2^{ex} we find the equation [2]:

$$\frac{U_2^{ex}}{N} = \frac{\rho}{2}\sum_{i,j} x_i x_j \int \phi_{2ij}(\vec{r}) g_{ij}(\vec{r}) d\vec{r} \tag{4.63}$$

For the exact derivation of Eq. (4.63) we refer to more detailed literature [2]. x_i and x_j are the concentrations for each component i and j, while ρ is the total density. $\phi_{2ij}(\vec{r_{ab}})$ is the pair potential

$$V_{ij}(r) = \frac{1}{2} \sum_{a,b;a \neq b}^{N} \phi_{2ij}(r_{ab}) \tag{4.64}$$

All potientials defined in section 3.1 are pair potentials, only interacting between two particles each. As we deal with radial symmetric functions, the integration over a d-1-dimensional spherical shell can be performed easily, so that (for a general) d-dimensional problem, Eq. (4.63) reads:

$$\frac{U_2^{ex}}{N} = \rho \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})} \sum_{i,j} x_i x_j \int_0^\infty r^{d-1} \phi_{2ij}(r) g_{ij}(r) dr \tag{4.65}$$

Where $\Gamma(\frac{n}{2})$ is the common Gamma-function.

4.3.2. Pressure

For the pressure P we have to start from the equation

$$\frac{\beta P}{\rho} = 1 - \frac{1}{6} \beta \rho \sum_{i,j} x_i x_j \int \vec{r} \phi'_{2ij}(\vec{r}) g_{ij}(\vec{r}) d\vec{r} \tag{4.66}$$

where u'_{2ij} is the derivative of the pair potential after r. As before we just need to integrate over a d-1 spherical shell to obtain

$$\frac{\beta P}{\rho} = 1 - \frac{1}{3} \beta \rho \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})} \sum_{i,j} x_i x_j \int_0^\infty r^d \phi'_{2ij}(r) g_{ij}(r) dr \tag{4.67}$$

For a three pair potential we obtain

$$\frac{\beta P}{\rho} = 1 - \frac{1}{36}\beta\rho^2 \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})} \int_0^\infty \int_0^\infty \int_{-1}^1 r^d \phi_3'(r_{12}, r_{13}, x) g(r_{12}, r_{13}, x) r_{12}^d, r_{13}^d dr_{12} dr_{13} dx \tag{4.68}$$

4.3.3. Compressibility

For the calculation of the compressibility χ_T the program needs no changes at all, as one can see from the following equations:

$$1 + \rho \sum_{ij} x_i x_j \int (g_{ij}^{(2)}(\vec{r}) - 1) d\vec{r} = \frac{\langle N^2 \rangle - \langle N \rangle^2}{\langle N \rangle} = \rho k_B T \chi_T \tag{4.69}$$

g_{ij}^2 is the two-particle distribution function, with the n-particle distribution function generally defined as

$$\rho^n g_{ij}^{(n)}(\vec{r^n}) = \rho_{ij}^{(n)}(\vec{r^n}) \tag{4.70}$$

with $\rho_{ij}^{(n)}$ as the n-particle density defined as

$$\rho_{ij}^{(n)}(\vec{r^n}) = \sum_{N \geq 2}^{\infty} P(N) \rho_{N,ij}^{(2)}(\vec{r^n}) \tag{4.71}$$

where $P(N)$ is the probability to find a system with exactly N particles (as the derivation takes place in the grand canonic ensemble) and $\rho_{N,ij}^{(2)}(\vec{r^n})$ represents the probability to find n-particles in a space $d\vec{r^n}$ around $\vec{r^n}$. The notation $\vec{r^n}$ is needed here to clarify that the functions could generally be more than only 2-particle functions, as they initially depend on $r_1, r_2, ...r_N$, of course.
For a detailed derivation of Eq. (4.69) - (4.71) I refer to more detailed literature [2].

Because of the equation for the structure factor $\tilde{S}(\vec{k})$

$$\tilde{S}(\vec{k}) = 1 + \rho \tilde{h}(\vec{k}) \tag{4.72}$$

as well as for the general relations

$$h_{ik}(r) = g_{ik}(r) - 1 \tag{4.73}$$

$$\gamma_{ik}(r) = h_{ik}(r) - c_{ik}(r) \tag{4.74}$$

and the Fourier transform of the OZ-equation (2.4), a comparison between Eq. (4.72) and Eq.(4.69) leads to

$$\tilde{S}(0) = \frac{1}{1 - \rho \sum_{ij} x_i x_j \tilde{c_{ij}}(0)} = \rho k_B T \chi_T \tag{4.75}$$

As $\tilde{c_{ij}}(0)$ will already be calculated when the program calls the subroutine 'Compressibility' to calculate χ_T, we do not need any changes here, because our general FFT already works for higher dimensions.

5. Validation of the solutions for hard-spheres by a comparison of analytic to numerical results, using c(r)

To check the accuracy and reliability of the code in higher dimensions, we applied it to hard-spheres, using the Percus-Yevick approximation for which analytic results for $c(r)$ in odd dimensions are accessible [6].

The following sections show a comparison of the analytic and the calculated solution for $c(r)$, considering different values for η.

5.1. Three dimensions

Following Leutheusser [6], and assuming σ=1, the analytic solution for $c(r)$ reads in three dimensions:

$$c(r) = c_0 + c_3 r + c_5 r^3; \quad 0 \leq r \leq 1 \tag{5.1}$$

with

$$\begin{aligned} c_0 &= 4Q_1^2 \\ c_3 &= 6\eta Q_0^2 \\ c_5 &= \frac{1}{2}\eta c_0 \end{aligned} \tag{5.2}$$

and

$$\begin{aligned} Q_0 &= -\frac{1+\frac{\eta}{2}}{(1-\eta)^2} \\ Q_1 &= \frac{\frac{1}{2}+\eta}{(1-\eta)^2} \end{aligned} \quad . \tag{5.3}$$

As described in section 3.1.3, we use η as input parameter. Therefore we have to calculate σ by the equations specified in table 1, to produce the hard-sphere potential. The following figures show a selection of the comparisons of the analytic and the numerical solution for different values of dr and N. According to Figure 5.1 and 5.2, a larger number of grid points does not lead to a different solutions (as long as the potential-vector is long enough to cover the discontinuity and a significant number of zeroes, of course). A comparison between Figure 5.2 and 5.3 shows, that an increase of η leads to a larger difference between the numerical and the analytic solution. Fortunately a smaller increment dr leads to a higher accuracy, as Figure 5.3 and 5.4 finally observe.

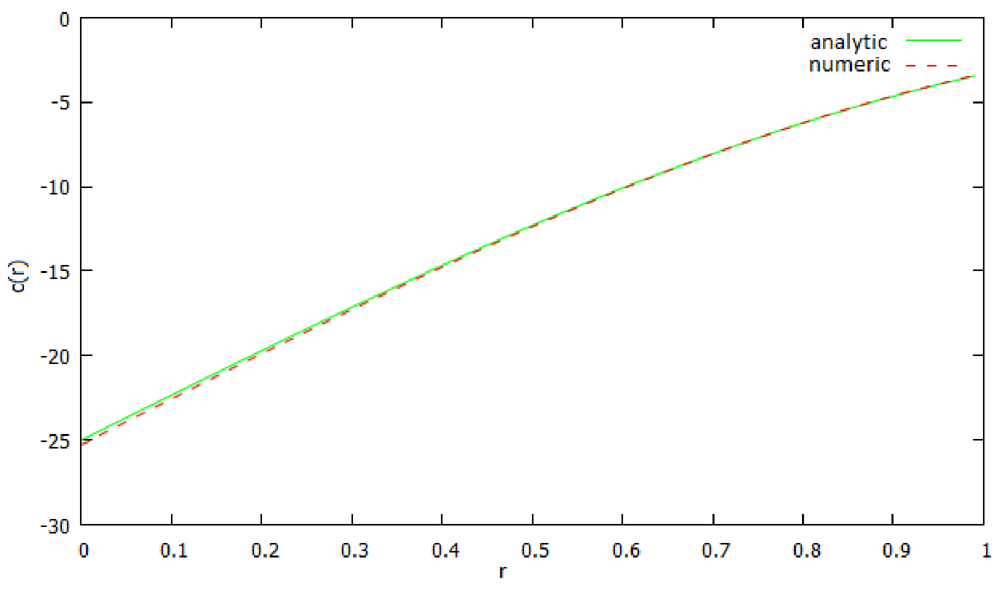

Figure 5.1: Comparison of analytic (solid, green line) and calculated (dashed, red line) solution in three dimensions, for η=0.4, dr=0.01, N=2^{10}.

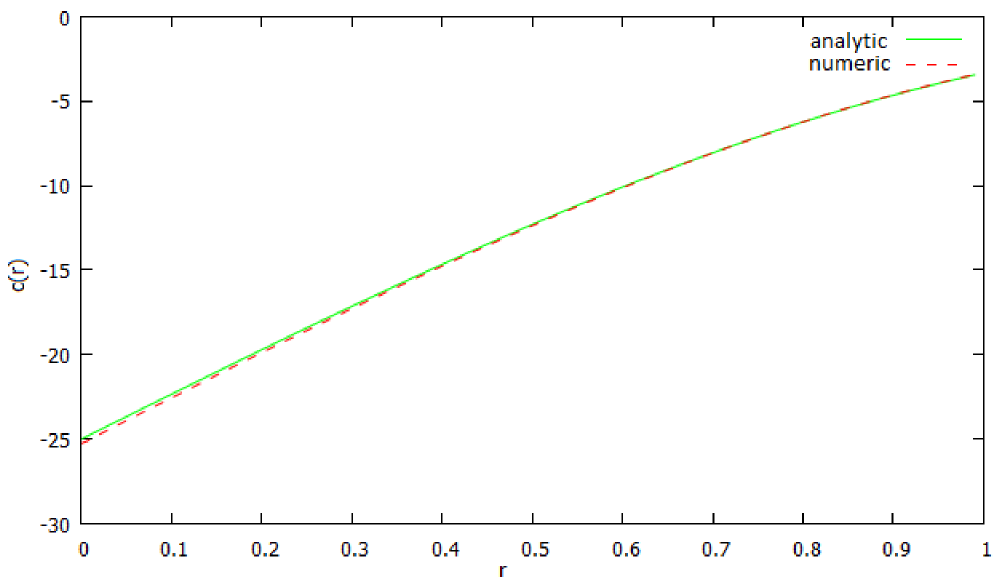

Figure 5.2: Comparison of analytic (solid, green line) and calculated (dashed, red line) solution in three dimensions, for η=0.4, dr=0.01, N=2^{12}.

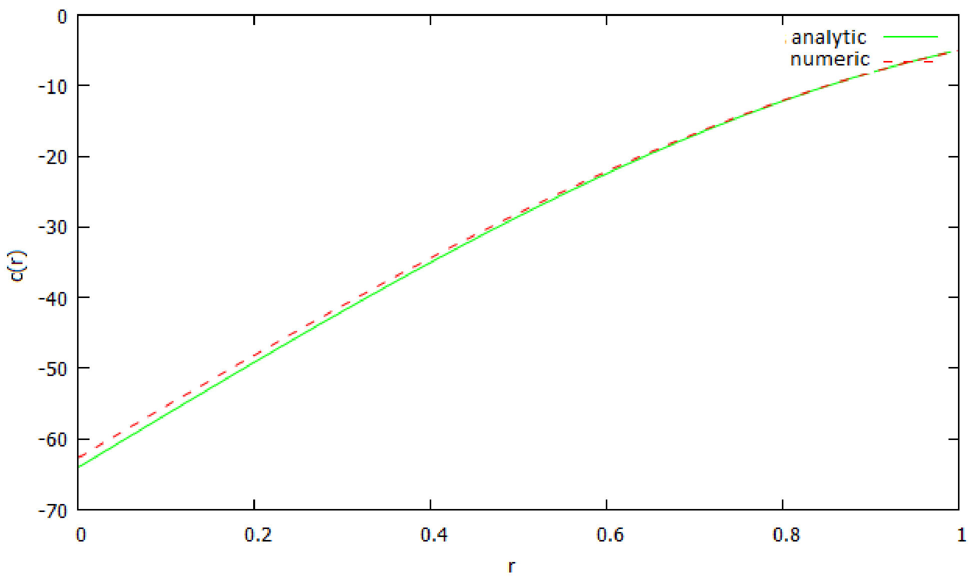

Figure 5.3: Comparison of analytic (solid, green line) and calculated (dashed, red line) solution in three dimensions, for η=0.5, dr=0.01, N=2^{12}.

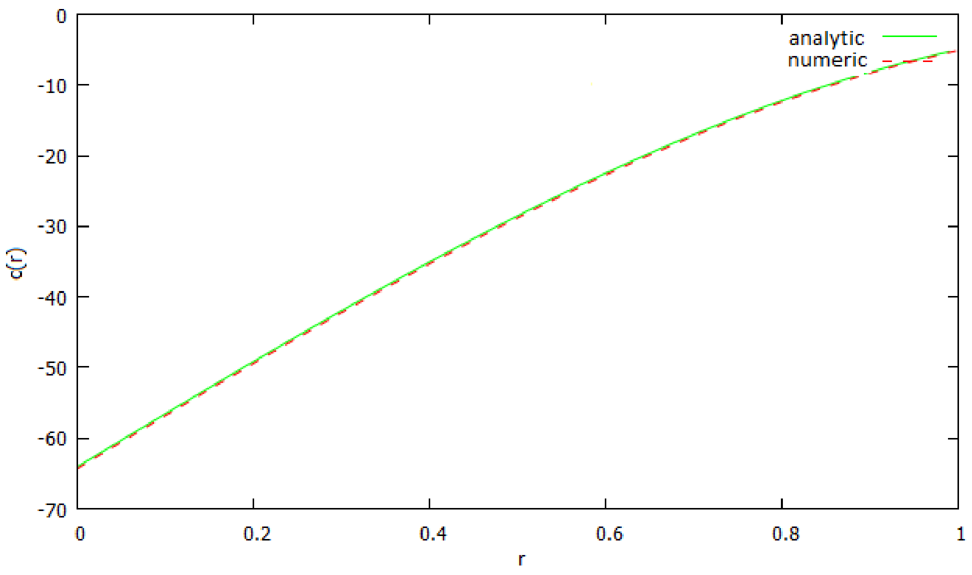

Figure 5.4: Comparison of analytic (solid, green line) and calculated (dashed, red line) solution in three dimensions, for η=0.5, dr=0.005, N=2^{14}.

5.2. Five dimensions

Proceeding to higher dimensions (i.e. d=5 and d=7 dimensions) [6], the analytic solutions for $c(r)$ in higher dimensions become quite complicated. Therefore they were produced in MATHEMATICA by a program provided by A. Santos [9].

A comparison of Figure 5.5 and 5.6 shows again, that for a smaller dr the numeric solution comes quite close to the analytic solution.

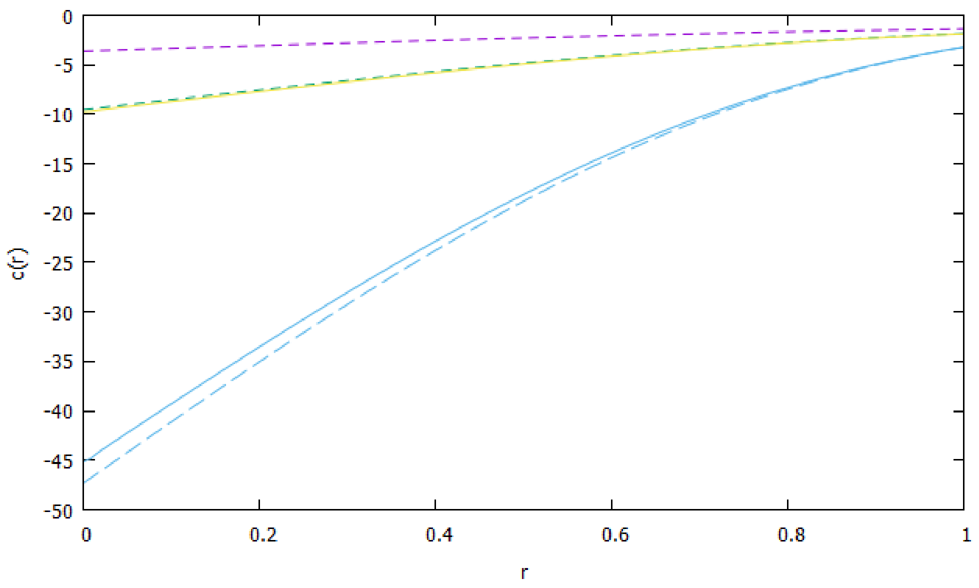

Figure 5.5: Comparison of analytic and calculated solution in five dimensions, for η=0.05, 0.1, 0.2 (from top to bottom), dr=0.01, N=2^{12}; The dashed line represents the analytic solution, while the solid line represents the numeric solution.

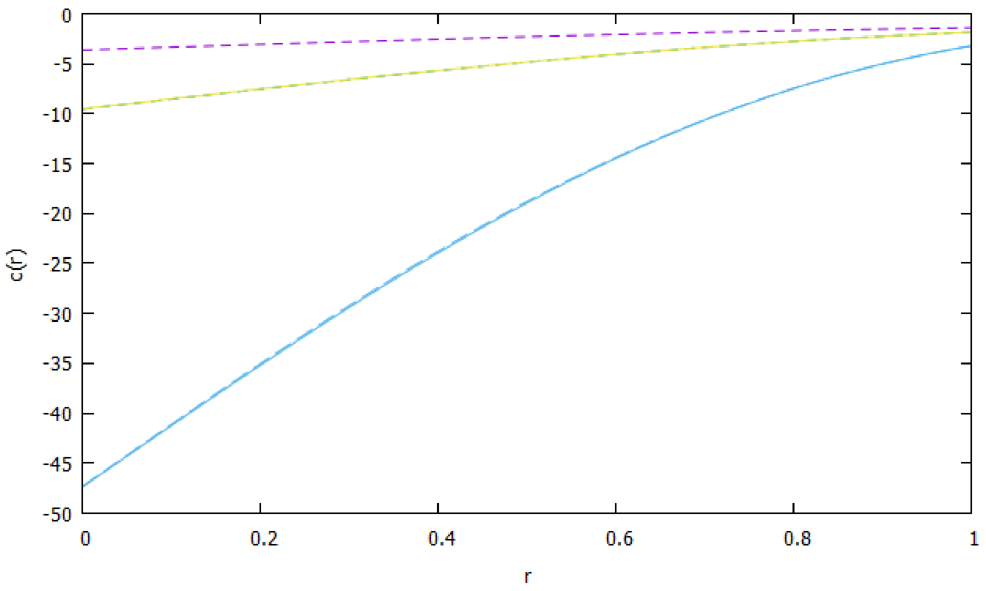

Figure 5.6: Comparison of analytic and calculated solution in five dimensions, for η=0.05,0.1,0.2 (from top to bottom), dr=0.001, N=2^{12}; The dashed line represents the analytic solution, while the solid line represents the numeric solution.

5.3. Seven dimensions

As for five dimensions, Figure 5.7 compares the analytic and the numeric solution for different values of η. A higher accuracy could have been achieved by using a smaller increment dr, together with a higher number of grid points N. Unfortunately (as η=0.07 is comparatively high for seven dimensions) the program does not converge for $dr = 0.001$ anymore.

To validate the adjustments for the thermodynamics as well, the numeric solution for the compressibility χ_T was compared to the analytic solution for the structure factor S(q) at q=0, taken from a publication of Robles, López de Harob & Santos [9].

The results are shown in Table 2.

η	χ_T	$S(0)$
0.01	0.37873	0.37582
0.02	0.19136	0.18627
0.03	0.11058	0.11438
0.04	0.07071	0.06863
0.05	0.04776	0.04902
0.06	0.03483	0.03595
0.07	0.02491	0.02288

Tab.2: Comparison of χ_T and $S(0)$ of hard spheres in seven dimensions, using the PY-approximation.

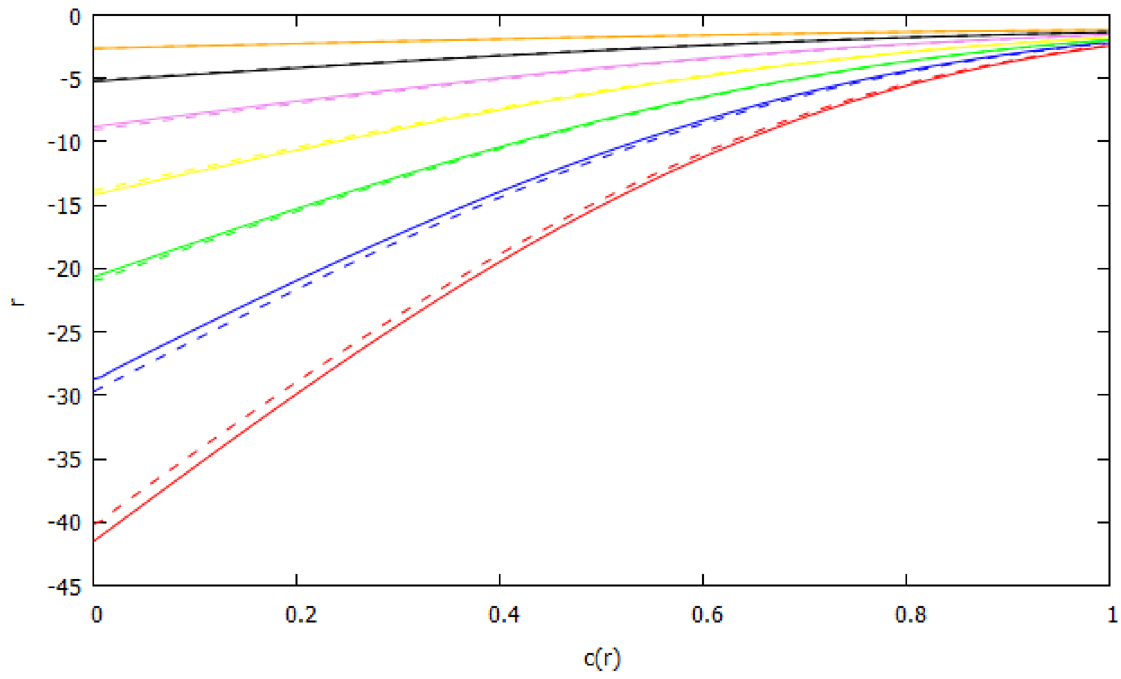

Figure 5.7: Comparison of analytic and calculated solution in seven dimensions, for η=0.01 - 0.07 (from top to bottom), dr=0.01, N=2^{12}; The dashed line represents the analytic solution, while the solid line represents the numeric solution for all cases but the last one with $\eta = 0.07$, where the lines are switched.

6. Results for a binary soft sphere mixture

Inspired by the studies of Jean-Marc Bomont, Jean-Pierre Hansen and Giorgio Pastore [4], [7], [8], a two-component system with a soft sphere potential was considered, due to Eq. (3.5) and Eq. (3.6). The idea of their work (on which I am not going to refer to in detail) is a systematic increase of the Γ, specifying the energy of the soft sphere potential, as defined in section 3.1.2. Basically, the output function $\gamma(r)_{out}$ for a certain Γ, with typical values between $\Gamma = 1$ and $\Gamma = 1.7$, is chosen as a new input $\gamma(r)_{inp}$. Comparatively high values of Γ can be reached by that method, as the algorithm would normally not converge for a very high Γ and $\gamma(r)_{inp} = 0$.

The interesting feature of the specific system considered in the studies of Jean-Marc Bomont, Jean-Pierre Hansen and Giorgio Pastore [7] is, that at a certain Γ, the value of the pair correlation function $g_{12}(0)$ at $r = 0$ is characterized by a certain increase, while for a smaller Γ-value, $g_{12}(0)$ increases much slower.

At this point we have to remember, that we are dealing with an attractive potential between the different components, due to equation Eq.(3.5).
On the other hand, the soft sphere potential between two equal particles is purely repulsive. As a lot of other particles of both sorts will be located between them, the combination of all repulsive forces can be seen as a barrier between one particle of sort 1 and another particle of sort 2. As an increase of Γ in Eq. (3.8) leads to a higher potential for both, the attractive and the repulsive part, the attractive energy will be high enough for the particles to overcome the barrier at a certain point, leading to significantly higher values of the correlation function $c_{12}(r)$, with a higher value of $g_{12}(r)$ as a result. Of course, the effect is particularly strong for small values of r_{12}, when the particles are very close. Therefore the value of $g_{12}(0)$ is of special interest. Further research dealing with that effect can be looked up for example in publications of J.M. Bomont, J.P. Hansen and G. Pastore [4], [7], [8].

6.1. Results in three dimensions

At first, we have to choose an increment $\Delta\Gamma$, that determines how fast Γ increases. According to the studies of Bomont, Hansen ans Pastore [4], [7], [8], who worked with another version of the program, the jump in $g_{12}(0)$ occurs between $\Gamma = 1.76$ and $\Gamma = 1.77$, which was reproduced in our calculations in Figure 6.1. An increment of $\Delta\Gamma = 0.01$ is too large for the current program. For higher values of Γ we have to choose values of $\Delta\Gamma$ as small as $\Delta\Gamma = 0.0001$ in order to achieve convergence.

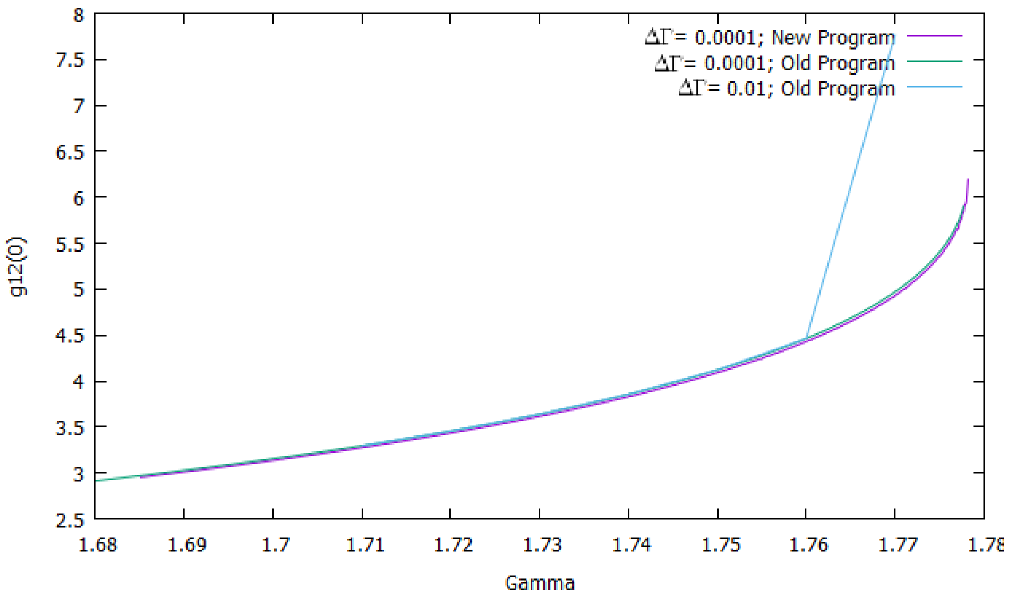

Figure 6.1: Comparison of $g_{12}(0)$ as a function of Γ, for different values of Γ (as labelled), for the current program and an older version of the program.

As one can see in Figure 6.1, the jump happens between $\Gamma = 1.76$ to $\Gamma = 1.78$, and is much lower than for $\Delta\Gamma = 0.01$. To make sure, that the difference was not produced by an error of one of the programs itself, a calculation with $\Delta\Gamma = 0.0001$ was applied on the old program as well. After all, the data obtained for the same value of $\Delta\Gamma$ are coincident for both the old and the new program.
To find out whether one of the solutions might be unphysically, the Structure factor $S(k)$, as well as the function $g_{12}(r)$ were calculated for each of the Γ-values.
Without going into too much detail, the Structure function as it is defined in Eq. (4.72), is related to the scattering cross section, assumed, that we are dealing with identical point particles, producing a delta-function for the density function $\rho(r)$.

As we can see in Figure (6.2) - (6.4), the pair distribution function g_{12} does only show significant differences for very low r. Therefore $S(k)$ in Eq. (4.72) does not change for different values of Γ, because as a function of the Fourier transform of $h(r)$ it combines all functions of the entire r-space, which only differ for very low values of r. (See Figure (6.5) -(6.7)).
A direct comparison between the relevant values for Γ, concerning different increments, is shown in Figure (6.8) - (6.10).

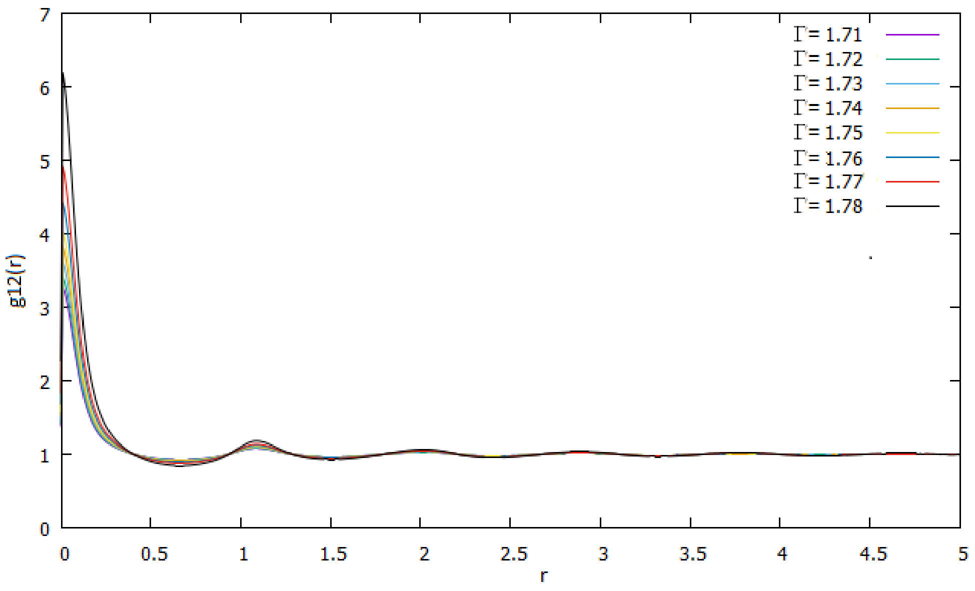

Figure 6.2: $g_{12}(0)$ as a function of Γ (as labelled), calculated by the new program, using $\Delta\Gamma = 0.0001$.

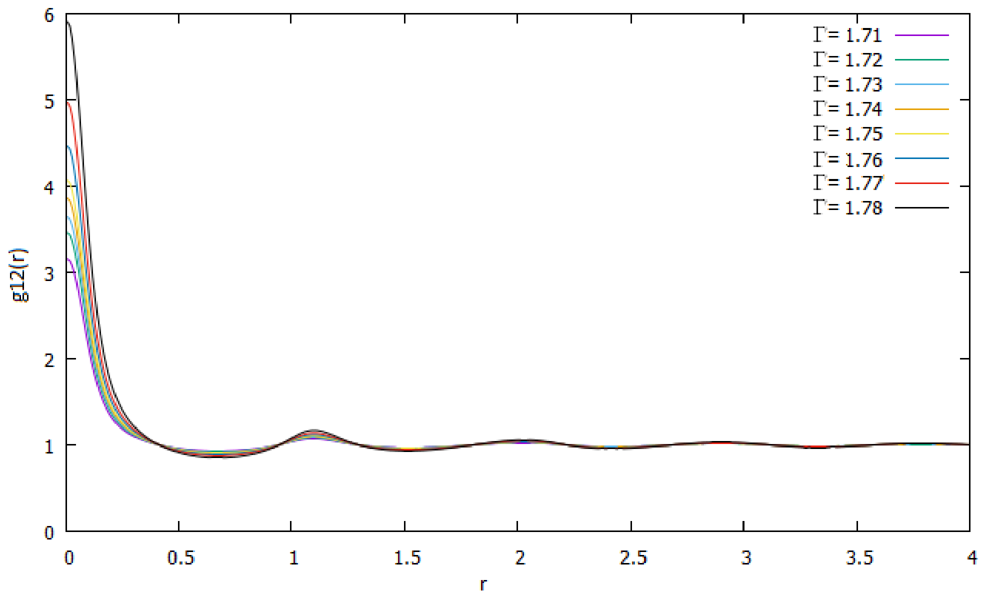

Figure 6.3: $g_{12}(r)$ as a function of Γ (as labelled), calculated by the old program, using $\Delta\Gamma = 0.0001$.

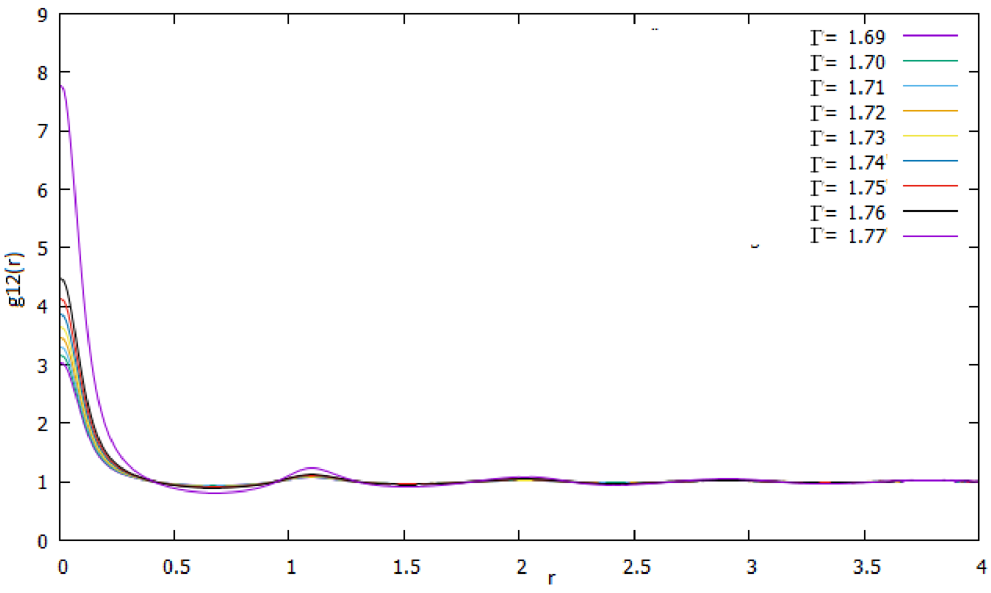

Figure 6.4: $g_{12}(r)$ as a function of Γ (as labelled), calculated by the old program, using $\Delta\Gamma = 0.01$.

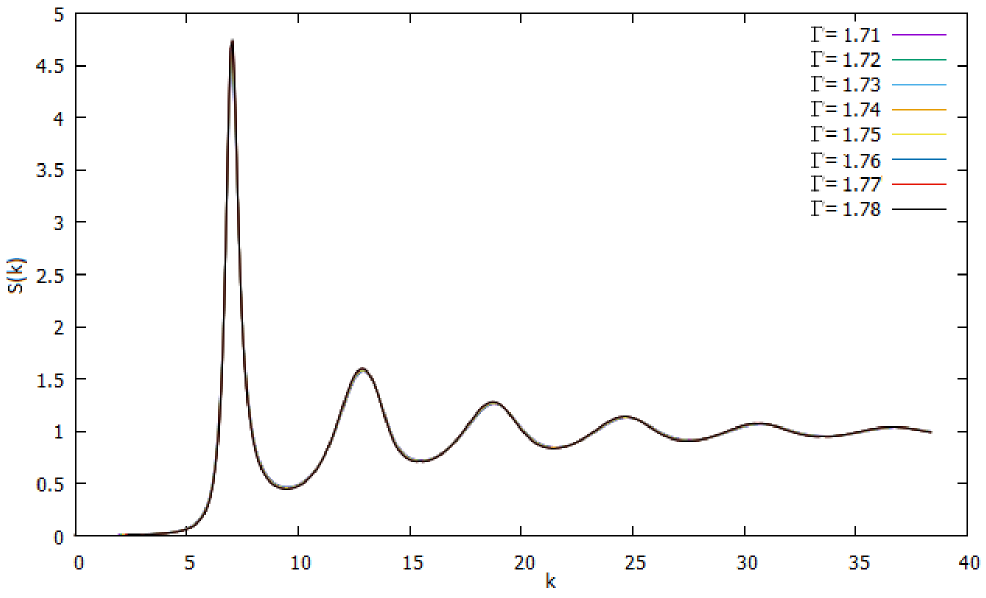

Figure 6.5: $S(k)$ as a function of Γ (as labelled), calculated by the new program, using $\Delta\Gamma = 0.0001$.

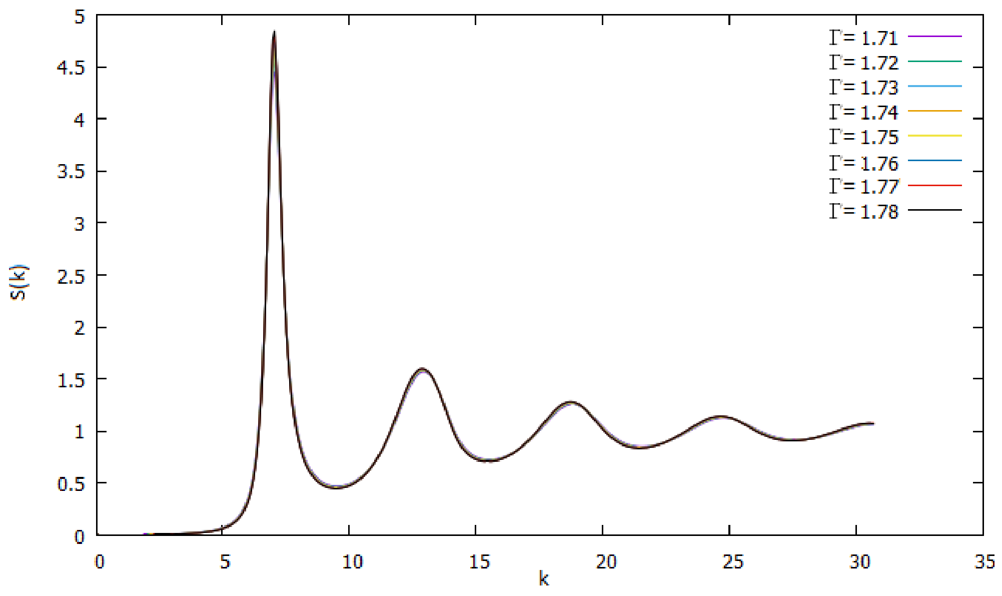

Figure 6.6: $S(k)$ as a function of Γ (as labelled), calculated by the old program, using $\Delta\Gamma = 0.0001$.

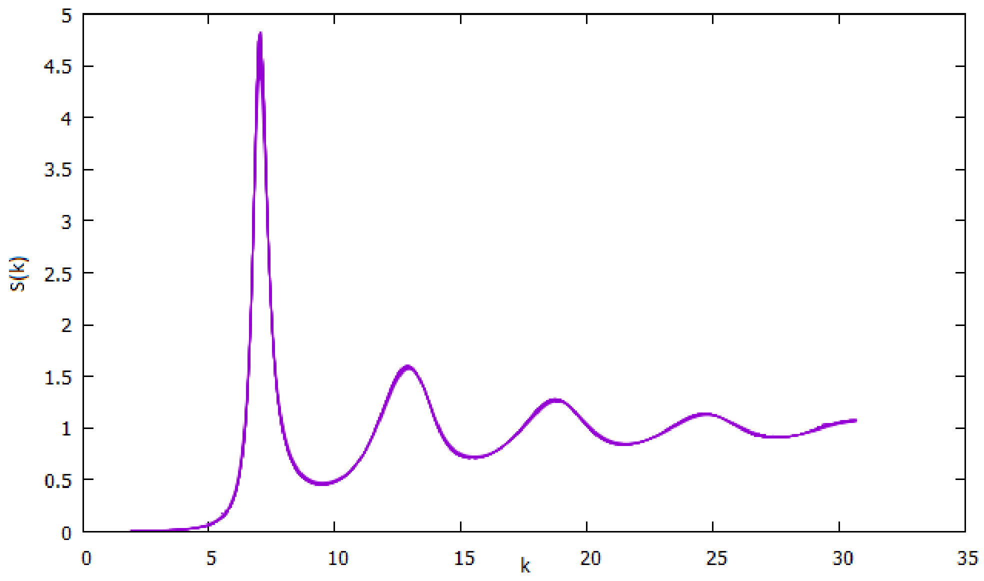

Figure 6.7: $S(k)$ as a function of Γ (as labelled), calculated by the old program, using $\Delta\Gamma = 0.01$.

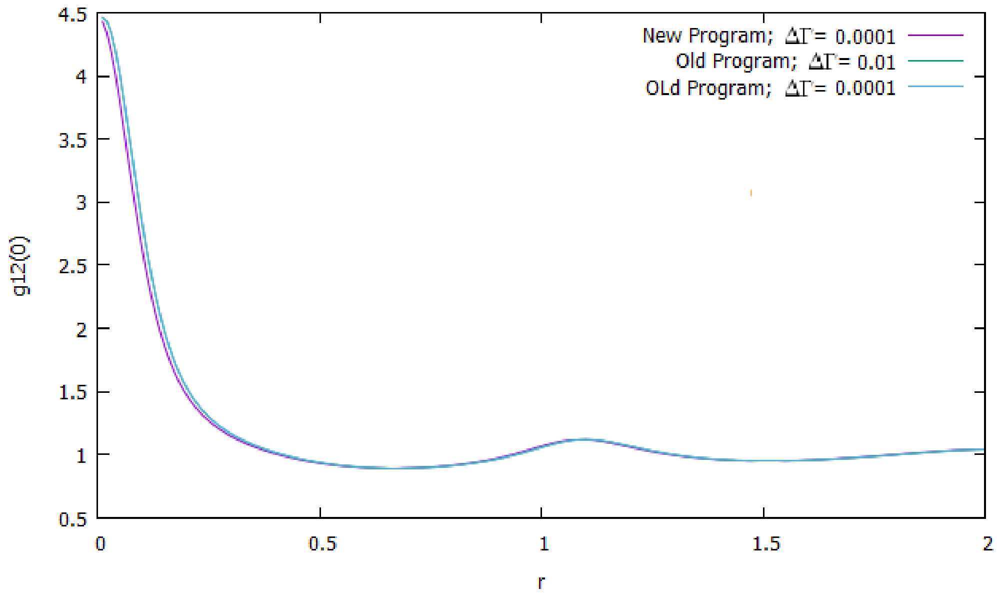

Figure 6.8: $g_{12}(r)$ as a function of Γ (as labelled), for old and new program, using $\Gamma = 1.76$.

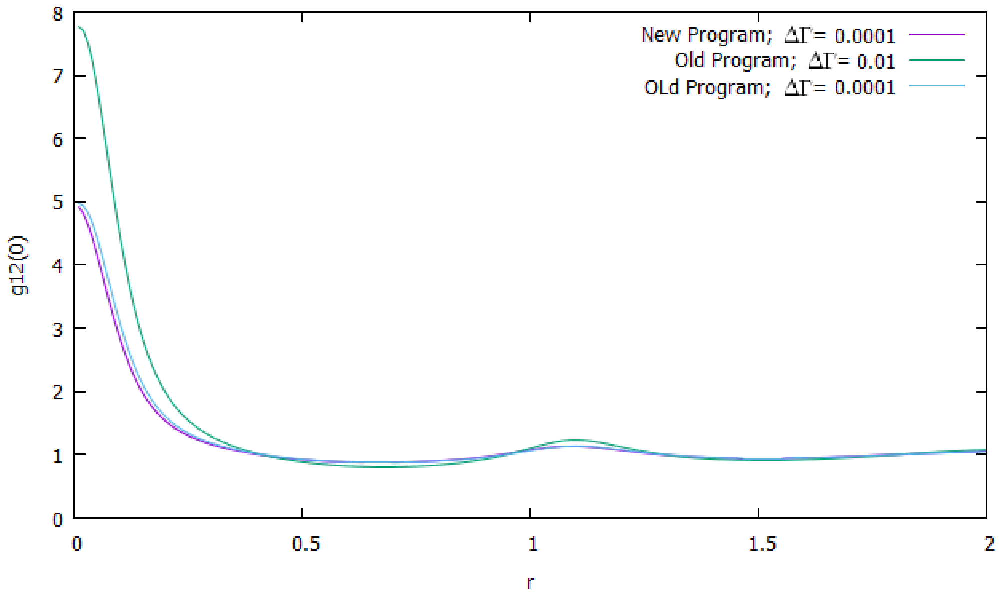

Figure 6.9: $g_{12}(r)$ as a function of $\Delta\Gamma$ (as labelled), for old and new program, using $\Gamma = 1.77$.

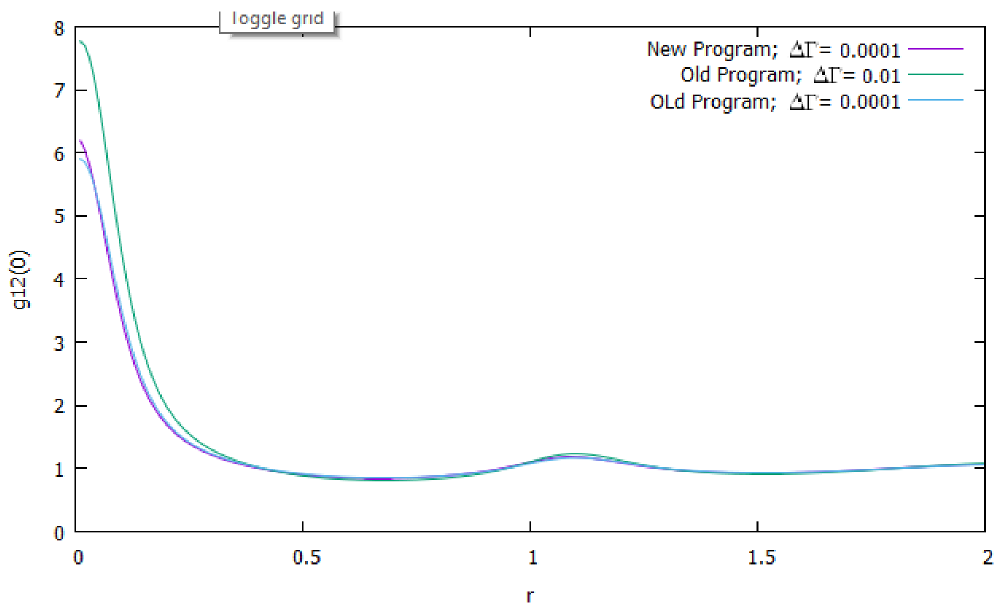

Figure 6.10: $g_{12}(r)$ as a function of $\Delta\Gamma$ (as labelled), for old and new program at $\Gamma = 1.77$ or $\Gamma = 1.78$ for the new program respectively.

6.2. Results in higher dimensions

In higher dimensions we can access much higher values for Γ as one can see in Figure 6.11 for five dimensions, and 6.12 for seven dimensions. A comparison between the functions $g_{12}(r)$ for different values of Γ is given in Figures 6.13 and 6.14.

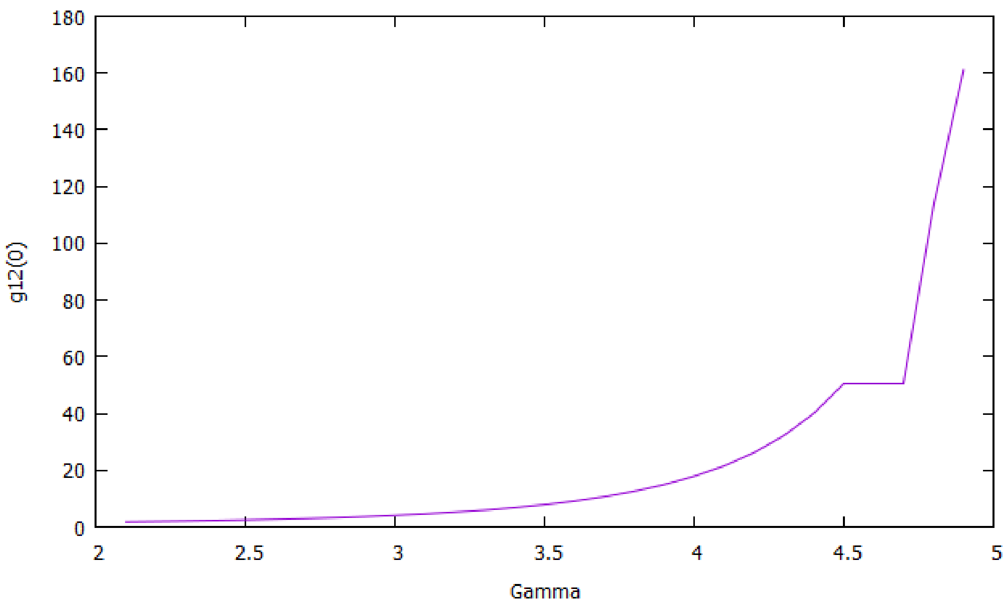

Figure 6.11: $g_{12}(0)$ as a function of Γ in five dimensions, calculated by the new program, using $\Delta\Gamma = 0.01$.

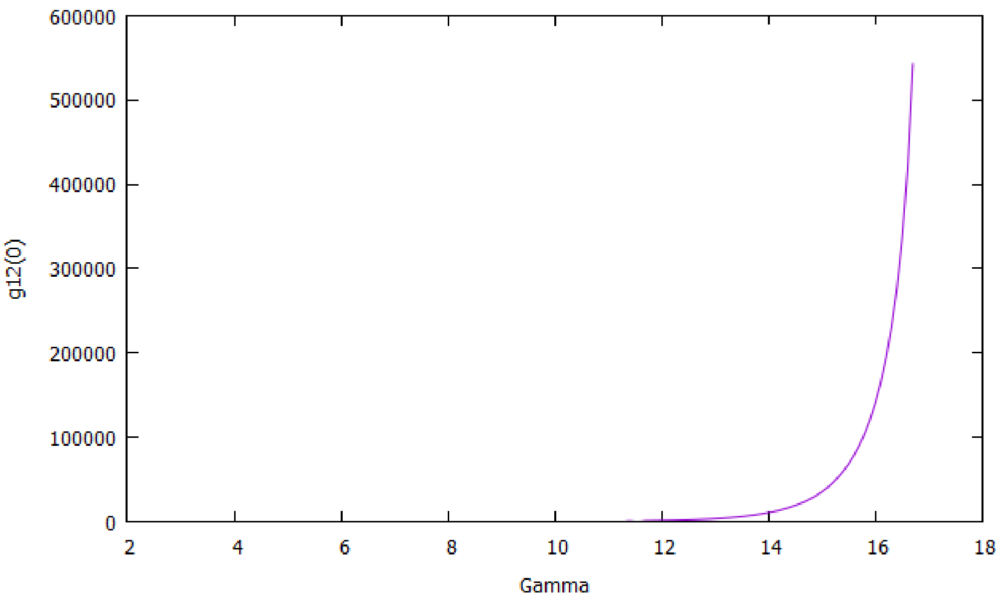

Figure 6.12: $g_{12}(0)$ as a function of Γ in seven dimensions, calculated by the new program, using $\Delta\Gamma = 0.01$.

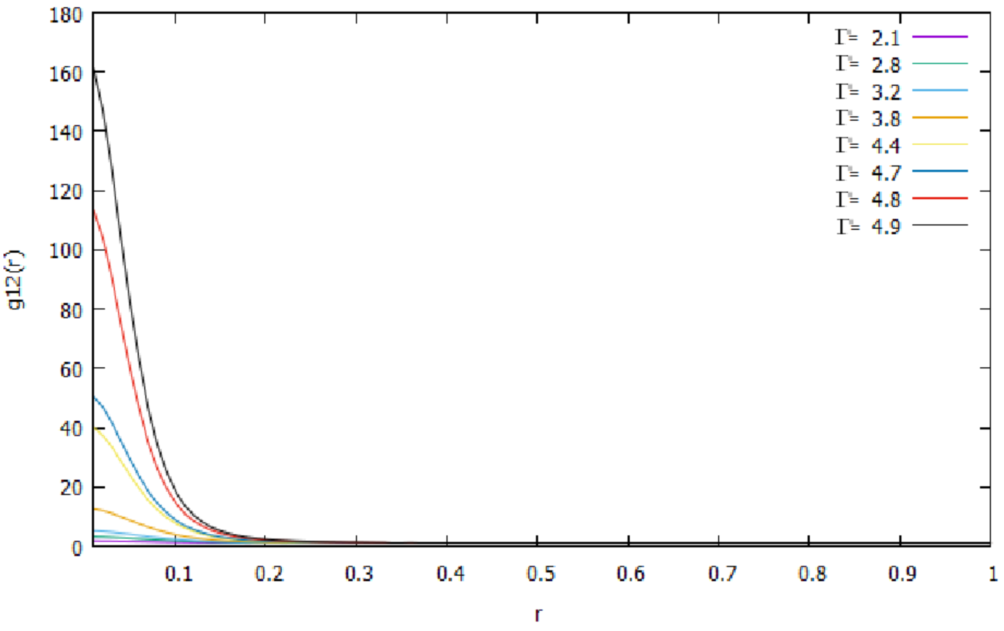

Figure 6.13: $g_{12}(0)$ as a function of Γ in five dimension (as labelled), calculated by the new program, $\Delta\Gamma = 0.01$, with $\Gamma = 2.1 - 4.9$.

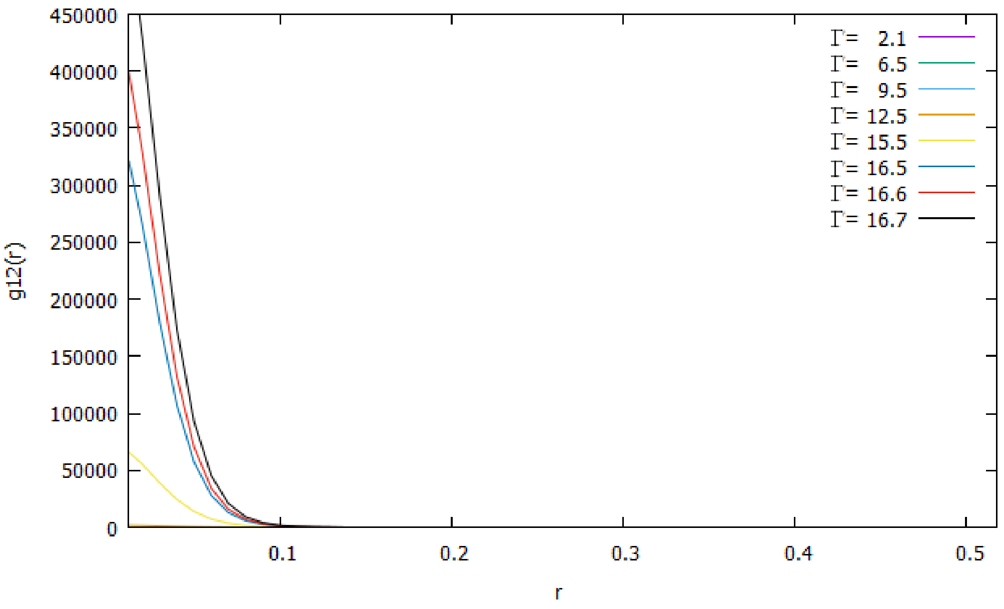

Figure 6.14: $g_{12}(r)$ as a function of Γ in seven dimensions (as labelled), calculated with the new program, using $\Delta\Gamma = 0.01$, with $\Gamma = 2.1 - 16.7$.

Conclusion

In this thesis we introduced the generalisation of integral equation techniques, consisting of the Ornstein-Zernike equation and a closure relation, to higher (odd) dimensions, namely to five and seven dimensions. At first we compared the analytic solution of the direct correlation function $c(r)$, available for hard-spheres, using the Percus-Yevick approximation, to the numeric solution, calculated by the program. In three dimensions the difference $\Delta c(r)$ between the numeric and the analytic solution varies between $\Delta c(0) = 0.05$ for $\eta = 0.4$ and $\Delta c(0) = 0.1$ for $\eta = 0.5$, if the increment of the grid size dr is chosen as $dr = 0.01$. If we improve the resolution by choosing $dr = 0.005$, the difference becomes much smaller, with $\Delta c(0) = 0.001$ for $\eta = 0.5$.

In five dimensions, when choosing $dr = 0.01$, we find $\Delta c(0) = 0.1$ for $\eta = 0.05$, up to $\Delta c(0) = 1.8$ for $\eta = 0.2$. Fortunately, those comparatively high inaccuracies can be compensated by choosing $dr = 0.001$. In this case we find $\Delta c(0) = 0.001$ for $\eta = 0.05$ and $\Delta c(0) = 0.01$ for $\eta = 0.2$. For $dr = 0.01$ the difference between the numeric and the analytic solution ranges between $\Delta c(0) = 0.05$ for $\eta = 0.01$ and $\Delta c(0) = 1.5$ for $\eta = 0.7$.

Secondly, the numeric solution for the pair distribution function $g_{12}(r)$ for a binary system were compared to the solution of another program version. Here the equal particles interact via a soft sphere potential, while different particles interact via an attractive potential. In particular, we studied the increase of $g_{12}(0)$ with the systematic increase of the parameter Γ, specifying the soft sphere potential. Depending on the increment $\Delta\Gamma$ we observed a substantial increase of $\Delta g_{12}(0)$ at $\Gamma \approx 1.76$ with both, the old and the new program versions.

Depending on the increment $\Delta\Gamma$, we obtained different values for the increase of $\Delta g_{12}(0)$. While $\Delta\Gamma = 0.0001$ produces a smooth increase of $\Delta g_{12}(0)$, the choice of $\Delta\Gamma = 0.01$ leads to a much higher $g_{12}(0)$ at $\Gamma = 1.77$.

A. Appendix

A.1. Code

For the generalisation to higher dimensions, a new variable, *idimension*, was introduced in the module **IEM-new.f90**. It specifies the dimension of the system. To normalize the density ρ with respect to the dimension, a few lines were added to the subroutine **init_Method**:

```
if(iDimension .EQ. 3) then
rho=3.D0/4.D0/pi*rho
end if
if(iDimension .EQ. 5) then
rho=15.D0/8.D0/pi**2*rho
end if
if(iDimension .EQ. 7) then
rho=105.D0/16.D0/pi**3*rho
end if
```

Furthermore, the following subroutines where adapted:
In MODULE **STR-new.f90**: **FT_Gamma_RtoQ**, **FT_c_RtoQ**, **FT_Gamma_QtoR**
In MODULE **BR-new.f90**: **FT_RtoQ**, **FT_QtoR**
In MODULE **THERMO-new.f90**: **U_EXC**, **Pressure**

As the adaptations are quite similar, we we will exemplify the generalisation by just one of them:

```
!==================================================
      SUBROUTINE FT_Gamma_RtoQ(ityp)
!==================================================

USE Global_Variables
IMPLICIT DOUBLE PRECISION (a-h,o-z)
DOUBLE PRECISION ::  fr(0:nx),fq(0:nx),fqs1(0:nx),fqc2(0:nx)
DOUBLE PRECISION ::  frs(0:nx),fqs(0:nx),frc(0:nx),fqc(0:nx),frs1(0:nx)
DOUBLE PRECISION::  frs2(0:nx),fqs2(0:nx),frc1(0:nx),fqc1(0:nx),frc2(0:nx)
!-------------------------
if(iDimension .EQ. 3) then
pi=4.D0*ATAN(1.D0)
coef=2.D0*dr*dr*npoint*SQRT(2.D0**(m-1))
Gam0=0.D0

DO i=0,nx
      r=i*dr
      fr(i)=GammaR(i,ityp)*r
      Gam0=Gam0+r*r*GammaR(i,ityp)
ENDDO

CALL RSA(m,fr(0),1,fq(0),1)
GammaQ(0,ityp)=Gam0*4.D0*pi*dr
GammaQ(nx,ityp)=0.D0
DO i=1,nx
      GammaQ(i,ityp)=fq(i)*coef/i
ENDDO
end if

if(iDimension .EQ. 5) then
pi=4.D0*ATAN(1.D0)
coef=dr*8.D0*pi**2*SQRT(2.D0**(m-1))
Gam0=0.D0

DO i=0,nx
      r=i*dr
      frs(i)=GammaR(i,ityp)*r
      frc(i)=GammaR(i,ityp)*r**2
      Gam0=Gam0+r**4*GammaR(i,ityp)/3.D0
ENDDO

CALL RSA(m,frs(0),1,fqs(0),1)
CALL RCA(m,frc(0),1,fqc(0),1)
GammaQ(0,ityp)=Gam0*8.D0*pi**2*dr
GammaQ(nx,ityp)=0.D0
DO i=1,nx-1
q=i*dq
      GammaQ(i,ityp)=fqs(i)*coef/q**3-fqc(i)*coef/q**2
ENDDO
end if
```

```
if(iDimension .EQ. 7) then
pi=4.D0*ATAN(1.D0)
dr=0.01
dq=2.D0*pi/(DBLE(npoint)*dr)
coef=dr*16.D0*pi**3*SQRT(2.D0**(m-1))
Gam0=0.D0

DO i=0,nx
     r=i*dr
     frs1(i)=GammaR(i,ityp)*3.D0*r
     frs2(i)=GammaR(i,ityp)*r**3
     frc(i)=GammaR(i,ityp)*3.D0*r**2
     Gam0=Gam0+(r**6)*GammaR(i,ityp)/15.D0
ENDDO

CALL RSA(m,frs1(0),1,fqs1(0),1)
CALL RSA(m,frs2(0),1,fqs2(0),1)
CALL RCA(m,frc(0),1,fqc(0),1)
GammaQ(0,ityp)=Gam0*16.D0*pi**3*dr
GammaQ(nx,ityp)=0.D0
DO i=1,nx-1
q=i*dq
     GammaQ(i,ityp)=fqs1(i)*coef/q**5-fqs2(i)*coef/q**3-fqc(i)*coef/q**4
ENDDO
end if

!==================================================
!==================================================
     END SUBROUTINE FT_Gamma_RtoQ
!==================================================
```

A.2. Literature

[1] Christian Libert (1997). *Integralgleichungsverfahren in der Flüssigkeitsphysik.* TU Wien. Vienna, Austria

[2] J.P. Hansen & I.R. McDonald (1986). *Theory of simple Liquids, fourth edition.* Universit´e Pierre et Marie Curie: Paris, France & University of Cambridge, UK. Academic Press

[3] J. Kierfield. (22.5.2018). *Weiche und biologische Materie, page 52-54.* Access on 28.7.2018,
http://t1.physik.tu-dortmund.de/files/kierfeld/teaching/LectureNotes/kierfeld_SoftBio1.pdf

[4] Bomont, Jean-Marc & Hansen, J-P & Pastore, Giorgio. (2015). *Hypernetted-chain investigation of the random first-order transition of a Lennard-Jones liquid to an ideal glass.* Physical Review E. 92. 042316. 10.1103/PhysRevE.92.042316.

[5] M.J. Gillan (1979) A new method of solving the liquid structure integral equations, Molecular Physics: An International Journal at the Interface Between Chemistry and Physics, 38:6, 1781-1794, DOI: 10.1080/00268977900102861

[6] E.Leutheusser (1984). *Exact solution of the Perkus-Yevik equation for hard sphere core fluid in odd dimensions.* Physik-Department der technischen Universität München: North Holland, Amsterdam

[7] Bomont, Jean-Marc & Pastore, Giorgio & Hansen, J-P. (2014). *Probing the pair structure of supercooled fluids by integral equations: Evidence for an equilibrium liquid-ideal glass transition?.* EPL (Europhysics Letters). 105. 10.1209/0295-5075/105/36003.

[8] Jean-Marc Bomont & Jean-Pierre Hansen & Giorgio Pastore. *An investigation of*

the liquid to glass transition using integral equations for the pair structure of coupled replicae. Journal of Chemical Physics, American Institute of Physics, 2014, 141 (17), 10.1063/1.4900774. hal-01516069

[9] M. Robles, M. López de Harob & A. Santos (2007). *Percus-Yevick theory for the structural properties of the seven-dimensional hard-sphere fluid.*The Journal of Chemical Physics 126, 016101. doi: 10.1063/1.2424459

[10] Rohrmann, René & Santos, Andres. (2007). *Structure of hard-hypersphere fluids in odd dimensions.* Physical review. E, Statistical, nonlinear, and soft matter physics. 76. 051202. 10.1103/PhysRevE.76.051202.

[11] Kin-Chue Ng(1974), J.Chem. Phys. 61, 2680

[12] S. Labik, A. Malijevsky & P. Vonka (1985), Mol. Physics 56, 709