May 2014

The Pekeris Method for Lithium: Possibilities and Obstructions

Marcel Kreuter
University of Wisconsin-Milwaukee

The pekeris method for lithium: Possibilities and obstructions

by

Marcel Kreuter

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of

Master of Science
in
Mathematics

at

The University of Wisconsin-Milwaukee
May 2014

ABSTRACT

THE PEKERIS METHOD FOR LITHIUM: POSSIBILITIES AND OBSTRUCTIONS

by

Marcel Kreuter

The University of Wisconsin-Milwaukee, 2014
Under the Supervision of Professor Kevin McLeod

It is widely believed that the properties of atoms and molecules are accurately described by the Schrödinger equation, at least in so far as relativistic effects may be neglected. Extracting these properties from the equation in practice, however, can be a highly challenging task. In 1958, Chaim L. Pekeris developed a method for computing the ground state energy of the helium atom. This thesis surveys the possibilities and obstructions that occur when one tries to compute the ground state energy of lithium using Pekeris's method.

Table of Contents

List of Figures

ACKNOWLEDGEMENTS

I would like to thank Prof. Kevin McLeod who introduced me to Quantum Mechanics and who came up with the idea of carrying over Pekeris's method to a higher dimensional system.

Chapter 1

Introduction

Modern Quantum Mechanics describes the interaction of elementary particles in the language of Functional Analysis. Every such physical model has to be evaluated based on its ability to predict the outcome of experiments. It was therefore one of the great achievements of Quantum Mechanics that it was possible to compute the different energy states of the Hydrogen atom analytically and that the resulting values coincided with the empirical formula found by Johann Balmer in the 19th century.

For Helium, the next atom in the periodic table, the approach is not so easy. The extra electron of this element seems to gravely complicate the situation so that an analytic computation such as that for Hydrogen is not possible. Instead, numerical methods are considered to compute at least the state with the lowest energy: the so called ground state. Most scientists use a variational method developed by Egil Hylleraas in the early 20th century. In this method a class of different states is chosen and one tries to minimize the energy among these states. The energy is computed directly, using formulas derived from the mathematical model. These computations include high-dimensional integration which is a hard task both numerically and analytically. The effort required to compute the energy is so great that even today only small systems of particles have been numerically analyzed to a satisfying degree of accuracy via the mathematical model.

In 1958 Chaim L. Pekeris published a paper in which he computed the ground state energy of Helium using a completely different method. Using a sophisticated coordinate transform, he broke down the problem from complicated integration to the computation of a determinant, an easy algebraic task. Even so, the transformation made the problem very lengthy. Therefore the computation itself was then done by the WEIZAC computer of the Weizmann Institute in Israel.

Of course computations like this can be done much quicker today and so it seems odd that even though Pekeris's approach was a huge success, noone seems to have ever taken the effort to approach other problems in the same way. The purpose of this thesis is therefore to survey the possibilities and limitations of Pekeris's method, applied to the next atom, lithium. We will see that a coordinate transform analogous to the one Pekeris used does not exist for lithium and that therefore the biggest problem is to find a suited transform. We will consider several transforms and observe how far Pekeris's method can be accomplished in these cases and where the approaches go wrong.

Chapter 2

Quantum Mechanics and ground states

The main purpose of this chapter is to give a brief introduction to modern Quantum Mechanics and the Functional Analytic background to it. All theorems metioned can be found in [1] and [2].

Let $\mathcal{H}$ be a separable, complex Hilbert space. A one dimensional subspace span(φ) with $\varphi \neq 0$ is called a *state* of a quantum mechanical system. A self-adjoint operator $H \in \mathcal{L}(\mathcal{H})$ is called an *observable* of the system.

Example 2.1 (N-particle system)**.** *We consider N particles with positions $x_1, \ldots, x_N$ in $\mathbb{R}^3$. In this scenario the Hilbert space $\mathcal{H}$ is the space $L^2\left(\mathbb{R}^{3N}\right)$ and thus a state is the span of a complex valued function φ. We will w.l.o.g. assume that $\|\varphi\| = 1$. In this case, the function $|\varphi|^2$ can be interpreted as a probability density. The resulting probability measure tells us what the possibility of finding the particles in certain positions is. This is necessary as Quantum Mechanics denies that it is possible to predict the result of an experiment exactly.*

The particles have a kinetic energy that is assumed to be the sum of the Laplacians with respect to each particle divided by two times the mass of the particle. At the same time, the particles interact by means of potential energy which is given by

$v_{ij}\left(x_i, x_j\right)$ *where the functions* v_{ij} *have to be determined based on the kind of particles that interact. If the particles interact by Coulomb forces,* v_{ij} *is given by* $\frac{e_i e_j}{|x_i - x_j|}$ *where* e_i *is the charge of the i-th particle. The total energy of the system is given by the sum of the kinetic and the potential energy. For an atom, a system with a nucleus and* $N-1$ *electrons, we get*

$$H = \sum_{i=1}^{N} -\frac{\Delta_i}{2m_i} + \sum_{i\neq j=1}^{N} \frac{e_i e_j}{|x_i - x_j|},$$

an operator in $\mathcal{L}\left(L^2\left(\mathbb{R}^{3N}\right)\right)$*. Note that we ignored physical constants which results that this computation is not done in usual units for energy. We usually assume that the mass of the nucleus is infinite and that the mass of each electron as well as its charge is* 1*. The charge of the nucleus will be denoted by* Z *in the same unit as the charge of the electrons and the particle distances* $|x_i - x_j|$ *will be thought of in Bohr radii. With this choice, the unit of energy we are computing is the so called* hartree *in which most computations are done for the sake of simplicity. The resulting operator takes the form*

$$H = \sum_{i=1}^{N-1} -\frac{\Delta_i}{2} + \sum_{i=1}^{N-1} \frac{-Z}{r_i} + \sum_{j\neq i} \frac{1}{r_{ij}},$$

where r_i *and* r_{ij} *denote the electron-nucleus and the electron-electron radii respectively.*

The operator H*, the so called* Hamiltonian *is certainly defined on all* C_c^∞*-functions. By a* Kato's theorem H *is an essentially self-adjoint operator and thus its closure is an observable.*

As an observable is meant to describe a physical quantity of a system, it seems unusual to define it as an operator on a Hilbert space. We have to make sense of the output if we measure an observable. Just as with the position of the particles, the energy and any other observables are not known exactly, but only by probability. The probability measure of an observable H is given by $\frac{d(\varphi, P_\lambda \varphi)}{(\varphi,\varphi)}$ where P_λ is the projection valued measure associated with H by the *spectral theorem*. With help of this theorem, the measured energy in an experiment if the system is in the state φ will have the expected outcome

$$E_{\varphi}(H) = \int_{\mathbb{R}} \lambda \frac{d(\varphi, P_{\lambda}\varphi)}{(\varphi,\varphi)} = \frac{(\varphi, H\varphi)}{(\varphi,\varphi)},$$

the so called Rayleigh quotient. Note that this quotient represents a real number as H is self-adjoint.

Now we will look at how a system evolves over time. In classical mechanics, the evolution of a system is governed by Hamilton's equations

$$\begin{aligned} \frac{dp}{dt} &= -\frac{dH(q,p,t)}{dq} \\ \frac{dq}{dt} &= \frac{dH(q,p,t)}{dp}, \end{aligned}$$

with $p, q \in \mathbb{R}^d$ and the Hamilton function $H(q,p,t)$. This has the quantum mechanical equivalent, the so called Schrödinger equation

$$\frac{d}{dt}\varphi(t) = iH\varphi.$$

By *Stone's theorem*, this equation is uniquely solvable with the help of the strongly continuous unitary group $U(t) = \exp(itH)$ generated by H where the exponential is given by the *continuous functional calculus*. The solution is then given by $\varphi(t) = U(t)\varphi$ where φ is the state of the system at time $t = 0$.

With this in mind we can now explain stability of particle systems, a question that classical mechanics failed to answer. Suppose φ is an eigenvalue of H with respect to the eigenvalue λ and that the system is in this state at time $t = 0$. The expected total energy of the system at time $t = 0$ when measured is then given by the Rayleigh quotient

$$\frac{(\varphi, H\varphi)}{(\varphi,\varphi)} = \frac{(\varphi, \lambda\varphi)}{(\varphi,\varphi)} = \lambda,$$

thus λ is interpreted as the total energy of the system in the state φ. Now we want to know what the energy at some time t in the future would be. By the previous results, the state of the system at this time is given by $U(t)\varphi$ and thus we can compute the energy of the system at time t to be

$$\frac{(U(t)\varphi, HU(t)\varphi)}{(U(t)\varphi, U(t)\varphi)}.$$

Note, that the group $U(t)$ and its generator H commute and that $U(t)$ preserves the inner product at any time. Therefore the Rayleigh quotient from above can be evaluated and we get

$$\frac{(U(t)\varphi, HU(t)\varphi)}{(U(t)\varphi, U(t)\varphi)} = \frac{(U(t)\varphi, U(t)H\varphi)}{(U(t)\varphi, U(t)\varphi)} = \frac{(\varphi, H\varphi)}{(\varphi, \varphi)} = \lambda.$$

This means that the total energy of the system does not change. Such a state is therefore called a *stationary state*. The interpretation of this phenomenon is that the system remains at a certain energy level while the state stays the same. Indeed the spectral theorem tells us that $U(t)\varphi = \exp(it\lambda)\,\varphi$ which means that the vectors φ and $U(t)\varphi$ belong to the same one dimensional subspace.

According to this interpretation, the lowest possible energy in a stationary state that a system can have is

$$\inf\left(\sigma_p\left(H\right)\right).$$

The eigenfunction to this eigenvalue is called the *ground state* of the system.

Chapter 3

Pekeris's approach for helium

As a quantum mechanical systems changes from one stationary state into another one with a lower energy, the energy difference between those two states results in the emission of light. The frequency of the light is directly proportional to the energy that is released. In 1885, Johann Jakob Balmer was the first person to find an empirical formula for the spectral lines of the lightest element that exists, hydrogen. He proposed a formula to compute the spectral lines which is today known as the *Balmer formula*. It was a great success for modern Quantum Mechanics when the eigenvalue problem of the hydrogen Hamiltonian could be solved analytically and the resulting spectrum essentially reproduced the Balmer formula.

For atoms with more than one electron, this computation is not so easy. It seems as if the eigenfunctions of the Hamilton operator are so complicated that it is impossible to write them down by hand. Instead, numerical methods where sought to approximate at least the lowest eigenvalue, i.e. the ground state energy. The first person to do so was Egil Hylleraas in his paper [3] for the helium atom. Hylleraas used a variational method that is still used today: A parametrized class of functions $\{\varphi_\alpha; \alpha \in \mathcal{A}\}$ is chosen and the energy is computed directly via the Rayleigh quotient $\frac{(\varphi_\alpha, H\varphi_\alpha)}{(\varphi_\alpha,\varphi_\alpha)}$. Via methods of mathematical optimization, this quotient is minimized. The resulting minimal value is an upper bound for the ground state energy of the system. The main problem with this method is the computation of the Rayleigh quotient, as the inner product is just a short notation for an integral. In the case of helium, this integral has to be computed in three dimensions (two for the radius of each electron and one for the distance between the two electrons). For the next bigger atom, lithium, the integral would be six dimensional, and the number of variables will increase drastically for every new electron. Computing integrals in high dimensions poses big problems both numerically and analytically, and in this case it is also aggravated by the fact that the variables are not independent: Interelectron distances cannot be arbitrarily large if the nucleus-electron radii stay fixed.

In 1958, Chaim L. Pekeris published a paper [4] in which he gave a different approach to the problem. The purpose of this chapter is to summarize and explain Pekeris's approach.

Pekeris first performed a coordinate tranform into the so called *perimetric coordinates*

$$\begin{aligned} u &:= \varepsilon\left(r_2 + r_{12} - r_1\right) \\ v &:= \varepsilon\left(r_1 + r_{12} - r_2\right) \\ w &:= \varepsilon\left(r_1 + r_2 - r_{12}\right), \end{aligned}$$

where $\varepsilon := \sqrt{-E}$ for the energy of the ground state E. Those coordinates were introduced in [5] to investigate the Hylleraas method. The perimetric coordinates range freely from 0 to ∞ and are therefore also worth a thought when performing a variational computation.

Pekeris's next step was to choose an Ansatz for the eigenfunction. On physical grounds, the function should decay exponentially with the coordinates, so Pekeris suggested a decay of $\exp\left(-\frac{1}{2}\left(u+v+w\right)\right)$. An orthogonal basis for the space $L^2\left(\mathbb{R}_+^3, \mathcal{B}\left(\mathbb{R}_+^3\right), \exp\left(-\left(u+v+w\right)\right) du\, dv\, dw\right)$ is given by the Laguerre polynomials

$$L_n\left(x\right) := \sum_{k=0}^{n} \binom{n}{k} \frac{\left(-x\right)^k}{k!},$$

so the Ansatz for the eigenfunction was

$$\varphi\left(u,v,w\right)=\exp\left(-\frac{1}{2}\left(u+v+w\right)\right)\sum_{l,m,n=0}^{\infty}A\left(l,m,n\right)L_l\left(u\right)L_m\left(v\right)L_n\left(w\right)$$

with the coefficients $A\left(l,m,n\right)$ to be determined. The downside of the new coordinates and the expansion in Laguerre polynomials is of course that the eigenvalue equation becomes far more complicated. This equation can be looked up in Equation (14) of [4].

Using the definition above, one can easily show that the Laguerre polynomials satisfy the recurrence and differential equations

$$\begin{aligned}
xL_n''\left(x\right)&=(x-1)L_n'\left(x\right)-nL_n\left(x\right)\\
xL_n\left(x\right)&=-(n+1)L_{n+1}\left(x\right)+(2n+1)L_n\left(x\right)-nL_{n-1}\left(x\right)\\
xL_n'\left(x\right)&=nL_n\left(x\right)-nL_{n-1}\left(x\right).
\end{aligned}$$

Substituting these relations into Equation (14) it is possible to remove all occuring variables u,v and w in the coefficients as well as all differentials leaving only coordinate independent coefficients and Laguerre polynomials. As this expression is equal to zero and the Laguerre polynomials are an orthogonal basis, all coefficients must be zero. This yields a recursion relation for the coefficients, which is equation (22) of [4]. Although this equation looks very complicated, it is actually linear and thus, if broken down into a finite sum, the solvability of the equation comes down to a determinant being equal to zero. Pekeris computed this determinant with the WEIZAC computer of the Weizmann Institute and computed the smallest value of ε for which it would be zero. The number $E=-\varepsilon^2$ is an upper bound for the ground state energy and Pekeris's computation improved the results that were given by the variational methods.

The big improvement of Pekeris's method is that the computations are fairly simple compared to the variational methods. Instead of computing highdimensional integrals we only need to compute a determinant, a much easier algebraic task.

Pekeris's computations can and have been redone using modern mathematical software, see for example [6]. Even though this method is much easier to handle than the variational methods, it has never been applied to higher systems like the lithium atom. Literature like [7] mentions a huge numerical effort that is carried out to compute ground states of three electron atoms using the variational method. The main idea of this project was therefore to survey the possibilities and restrictions of Pekeris's method, applied to the lithium atom.

Chapter 4

The convexity problem

The first step in applying Pekeris's approach to the Lithium atom is to find an appropriate coordinate system for the position of the three electrons. It turns out that this task causes a lot of issues. Recall that the perimetric coordinates are given by

$$\begin{aligned} u &:= \varepsilon\left(r_2 + r_{12} - r_1\right) \\ v &:= \varepsilon\left(r_1 + r_{12} - r_2\right) \\ w &:= \varepsilon\left(r_1 + r_2 - r_{12}\right). \end{aligned}$$

These coordinates, which are based on triangular inequalities, have two major benefits: First, the coordinates range independently from 0 to ∞, which plays an important role in the expansion in orthogonal polynomials. Second, the coordinate transform is a linear map which can be inverted easily. This is important when it comes to the computation of the Hamilton operator. So for the Lithium atom, we would favour similar coordinates: freely ranging on an interval and obtained by a linear transformation. It turns out that this is impossible. A nice discussion of this fact is given in [8] by Benjamin P. Carter.

Let D be the region of all possible coordinates $R = (r_1, r_2, r_3, r_{12}, r_{13}, r_{23})$. A point R is in D if and only if three vectors $X_1, X_2, X_3 \in \mathbb{R}^3$ exist such that $r_i = \|X_i\|$ and $r_{ij} = \|X_i - X_j\|$. The task is to find constraints on the numbers r_i and r_{ij} such that $R \in D$ if and only if R obeys these constraints. Suppose these constraints would be linear, then the set D would be convex as any set of points constrained by linear equations and inequalities is a convex subset of $\mathbb{R}^d$. But an easy counterexample shows that D cannot be convex.

Let the three electrons and the nucleus form a rectangle of length a and width b such that $a < b$. Let $c = \sqrt{a^2 + b^2}$ be the length of the diagonals. This configuration corresponds to a point $R = (a, b, c, c, b, a) \in D$. Let $R' \in D$ be the configuration of the rectangle with lengths $R' = (b, a, c, c, a, b)$. Note that this rectangle is congruent to the first rectangle.

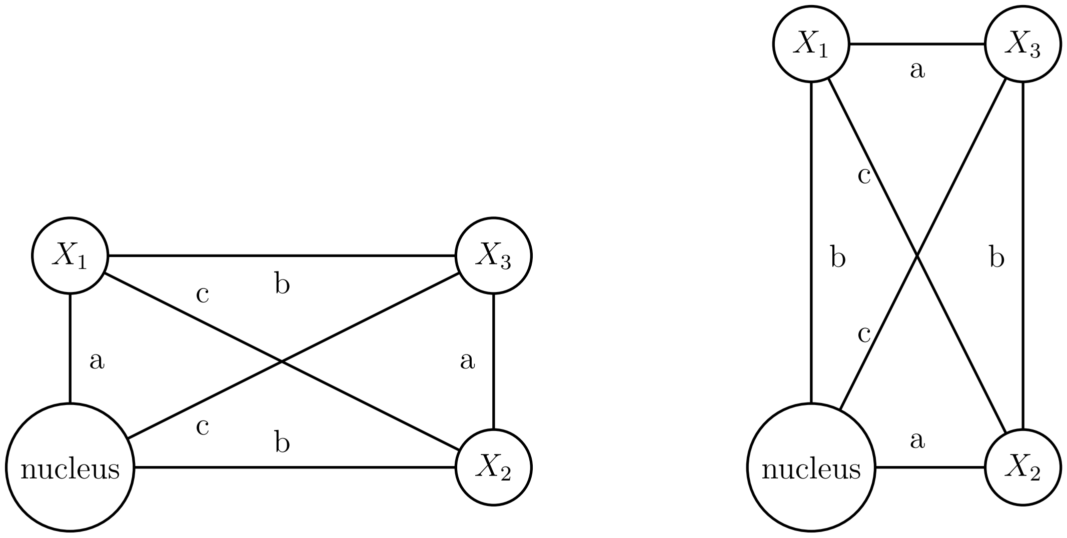

Figure 4.1: Two possible configurations

If D would be convex, then the configuration $\frac{1}{2}(R + R') = (\frac{a+b}{2}, \frac{a+b}{2}, c, c, \frac{a+b}{2}, \frac{a+b}{2})$ would be in D. With the radii r_1, r_2, r_{13} and r_{23} being equal, the radii r_3 and r_{12} cannot be arbitrary long no matter how the particles are located. The maximal value for these radii would be the diagonal of a square $\sqrt{2\left(\frac{a+b}{2}\right)^2} = \sqrt{\frac{(a+b)^2}{2}}$, but as $b > a$, the inequality of arithmetic and geometric means yields that $c^2 = a^2 + b^2 = \frac{a^2+b^2}{2} + \frac{a^2+b^2}{2} > \frac{a^2+b^2}{2} + ab = \frac{(a+b)^2}{2}$, a contradiction.

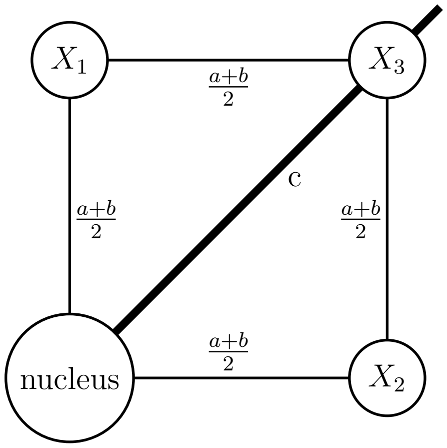

Figure 4.2: Convex combination of the two configurations - An impossible configuration

The lack of a coordinate transform like the one Pekeris used, leads to an inconvenient choice: Either we need to work with a coordinate transform that is exact, but nonlinear and might therefore be very complex, or we choose to work with approximations to the real coordinates that might deliver imprecise results. The next chapters are dedicated to the description of different coordinate systems that we tried and the issues that arise with these choices of coordinates.

Chapter 5

Normal coordinates

As we've seen in the last chapter, an exact coordinate system will always be nonlinear. For the computation of the Hamilton operator, we need the inverse of such a transform, which might be hard to compute. In [8] Carter surveys the so called *normal coordinates*. These coordinates are nonlinear and it is possible to invert them.

Let $D_+ \subset D$ be the set arising from linearly independent points. The set $D \backslash D_+$ is described by the equation $\det(X_1, X_2, X_3) = 0$ and is therefore a one dimensional set with Lebesgue measure zero. It is therefore enough to give a set of coordinates that describe D_+ instead of D. We define the normal coordinates for this set.

Given $\tilde{X}_1, \tilde{X}_2, \tilde{X}_3 \in D_+$ we choose new coordinate axis that result in three vectors X_1, X_2, X_3 respectively. We choose the x-axis of our coordinate system in direction of the vector $\tilde{X}_1$. As $\tilde{X}_1 \neq 0$, this choice is possible and unique. The y-axis is chosen perpendicular to the x-axis in the $\tilde{X}_1 - \tilde{X}_2$ plane. If $\tilde{X}_1$ and $\tilde{X}_2$ are linearly independent, there are exactly two such choices and we choose the one which makes the y-component of X_2 positive. Finally, the z-axis is chosen perpendicular to the first two axis and as before we choose the option that makes the z-component of X_3 positive. With this choice of coordinates, we only have to look at six numbers, describing the positions of the electrons: $\mathrm{X}_1 = \begin{pmatrix} x1 \\ 0 \\ 0 \end{pmatrix}, X_2 = \begin{pmatrix} x2 \\ y2 \\ 0 \end{pmatrix}, X_3 = \begin{pmatrix} x3 \\ y3 \\ z3 \end{pmatrix}$.

From these coordinates, we can easily compute the particle distances R via

$$r_i = \|X_i\| \text{ and } r_{ij} = \|X_i - X_j\|,$$

but it is not a-priori clear how to compute the mapping that transforms a set of given radii in D_+ into a set of normal coordinates. The inversion of the above computation can be obtained in the following way:

Let $p_{ij} := X_i \bullet X_j$ $(i, j = 1, 2, 3)$. These numbers are the entries of a symmetrical matrix P which can be written as $P = LL^T$, where L is the lower triangular matrix

$$L = \begin{pmatrix} x_1 & 0 & 0 \\ x_2 & y_2 & 0 \\ x_3 & y_3 & z_3 \end{pmatrix}.$$

It is possible to compute the elements p_{ii} from the set of radii R via $p_{ii} = r_i^2$. The elements p_{ij} can also be computed using the law of cosines $r_{ij}^2 = r_i^2 + r_j^2 - 2r_i r_j \cos\theta$ where θ is the angle between X_i and X_j. This angle can be computed via $X_i \bullet X_j = r_i r_j \cos\theta$. Putting both of these equations together we obtain $p_{ij} = \frac{1}{2}\left(r_i^2 + r_j^2 - r_{ij}^2\right)$. Suppose now that P is a positive matrix, then the matrix L can be computed via the well known Cholesky decomposition algorithm (for a discussion of this algorithm, see for example [9])

$$L_{ij} = \begin{cases} \sqrt{p_{ii} - \sum_{k=1}^{i-1} p_{ik}^2}\,, & \text{if } i = j \\ \frac{1}{p_{jj}}\left(p_{ij} - \sum_{k=1}^{j-1} p_{ik}p_{jk}\right), & \text{if } i > j \\ 0\,, & \text{otherwise} \end{cases}$$

.

In fact, if we try the standard criterion for positivity, we see that P is positive and therefore L can be computed by the above formula. A matrix is positive if and only if the subdeterminants

$$p_{11}, \begin{vmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{vmatrix} \text{ and } \det(P)$$

are all positive numbers. This condition can be interpreted geometrically: The first number is the square of the length of X_1, the second the square of the area of the parallelogram spanned by X_1 and X_2 and the third is the square of the volume of the parallelepiped spanned by X_1, X_2 and X_3. These three numbers are positive for any points in D_+ and thus the matrix P is always positive.

Now that we have the coordinate transform together with its inverse, we can compute the Hamilton operator in the normal coordinates. It is obvious from the formulas used in the Cholesky decomposition that this will result in a rather complicated differential operator. Carter invites the reader to try and write down the equation to gain understanding of how complicated this task is. Our hope was that the computations could be performed with the help of modern mathematical software such as MAPLE. The worksheet 'normalCoordinates' is supposed to compute the equation symbolically, but it turns out that this task is too hard to handle for the current desktop computers, we used. MAPLE runs out of memory, before the computation can be finished. Even if we could compute the Hamilton, we would run into the next problem on our way to perform the steps Pekeris did. The normal coordinates are independent and three range from 0 to ∞ while the three others range from $-\infty$ to ∞. Following Pekeris's steps we would try an expansion in Laguerre and Hermite polynomials for the eigenfunction in the different variables. But now we need to get rid of all derivatives and variables in the coefficients to be able to compute a recursion relation as Pekeris did. The Hermite polynomials obey certain differential and recursion relations just as the Laguerre polynomials do, but with a look at the relations one can clearly see, that it is only possible to handle certain expressions of the form $p(x)L_n(x)$ where p is a polynomial. The normal coordinates are much more complicated and thus it is unlikely, that Pekeris's method can be applied to the eigenvalue problem in this form.

Chapter 6

Frolov's four body perimetric coordinates

The results of the last chapter suggest that exact nonlinear coordinates are too complex to apply Pekeris's method to the Lithium atom. We should find coordinates that approximate D by a set D' without being too complex. In order to get polynomial prefactors, it would be the best to focus on the simplicity of the coordinates rather than on the exactness of the set D'. A set of coordinates that might be able to deliver this is described by Alexei M. Frolov in [10].

Note that the four particles of the Lithium atom form a tetrahydron with four triangular faces. Numerating the three electrons by $1, 2$ and 3 and the nucleus by 0, we can denote the faces by U for the face with vertices $(0, 1, 2)$, T for the face with vertices $(0, 1, 3)$, W for the face with the vertices $(0, 2, 3)$ and S for the face with vertices $(1, 2, 3)$.

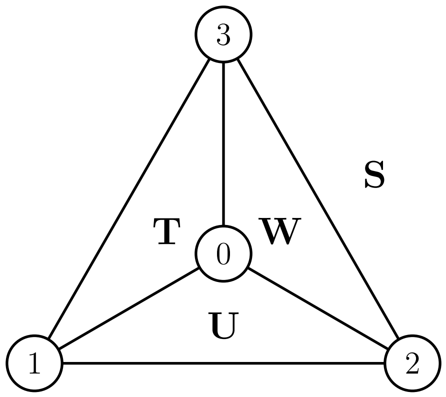

Figure 6.1: The four faces - Stereographic projection

These triangles can be described by the perimetric coordinates, which we will denote in a slightly different form

$$x_1 = \frac{1}{2}(r_2 + r_{12} - r_1)$$
$$x_2 = \frac{1}{2}(r_1 + r_{12} - r_2)$$
$$x_3 = \frac{1}{2}(r_1 + r_2 - r_{12}).$$

With this choice, the inverse coordinates can be easily written as

$$r_1 = x_1 + x_2$$
$$r_2 = x_1 + x_3$$
$$r_{12} = x_1 + x_2.$$

Applying these coordinates to the four faces we get coordinates u_i, t_i, w_i and s_i for the faces U, T, W and S respectively.

$$u_1 = \frac{1}{2}(r_2 + r_{12} - r_1), \qquad u_2 = \frac{1}{2}(r_1 + r_{12} - r_2),\ u_3 = \frac{1}{2}(r_1 + r_2 - r_{12})$$
$$t_1 = \frac{1}{2}(r_3 + r_{13} - r_1), \qquad t_2 = \frac{1}{2}(r_1 + r_{13} - r_3),\ t_3 = \frac{1}{2}(r_1 + r_3 - r_{13})$$
$$w_1 = \frac{1}{2}(r_2 + r_{23} - r_3), \qquad w_2 = \frac{1}{2}(r_3 + r_{23} - r_2),\ w_3 = \frac{1}{2}(r_3 + r_2 - r_{23})$$
$$s_1 = \frac{1}{2}(r_{12} + r_{13} - r_{23}), \qquad s_2 = \frac{1}{2}(r_{23} + r_{13} - r_{12}),\ s_3 = \frac{1}{2}(r_{12} + r_{23} - r_{13})$$

From the inverse coordinates, we can see that six additional constraints must hold.

$$u_1 + u_2 = r_{12} = s_1 + s_3, \quad u_1 + u_3 = r_2 = w_1 + w_3, \; u_2 + u_3 = r_1 = t_2 + t_3$$
$$t_1 + t_2 = r_{13} = s_1 + s_2 \quad t_1 + t_3 = r_3 = w_2 + w_3, \; w_1 + w_2 = r_{23} = s_2 + s_3.$$

With these six constraints, we can reduce the number of necessary coordinates from 12 to 6. All remaining coordinates can be expressed in terms of the six chosen coordinates. The six coordinates can be chosen in many different ways from the coordinates u_i, t_i, w_i and s_i. Frolov suggests to chose the three coordinates describing U as well as one of each of the coordinates describing T, W and S. We will follow Frolov's suggestion and pick the coordinates u_1, u_2, u_3, t_1, w_1 and s_3.

As mentioned above, the six remaining coordinates can be expressed in terms of the chosen ones, e.g. $w_3 = u_1 + u_3 - w_1$. Note that w_3 is a positive number so this yields an additional constraint $w_1 \leq u_1 + u_3$. In fact we get six constraints, which we will not reproduce here. The constraints and their derivation can be seen in equations (18) and (19) of [10]. Note that these constraints are necessary but not sufficient, for if they were, the set D would be described by linear equations and inequalities, but that cannot be true as we have seen earlier. Thus this paper is in error when it claims that the six coordinates describe all possible configurations in an arbitray four-body system. We will nevertheless use these coordinates as a model and hope that the coordinates are not too far off from the actual set D.

The reason we do not care about the six constraints of equation (19) is that we want to drop these constraints in order to be able to use Pekeris's method. For this method the variables need to range freely from 0 to ∞, so for our model, we will look at the set D', described by the six linear coordinates above without any further constraints.

The MAPLE worksheet 'frolovCoordinates' computes the Hamilton operator in these new variables which are renamed to $x_1, \ldots, x_6$. The result is a 4884 term partial differential equation on which we can now apply Pekeris's method. Following Pekeris's steps, the Ansatz for the eigenfunction is

$$\varphi(x) = \exp\left(-\frac{1}{2}(x_1 + x_2 + x_3 + x_4 + x_5 + x_6)\right)$$
$$\sum_{k,l,m,n,o,p=0}^{\infty} A(k,l,m,n,o,p)\, L_k(x_1)\, L_l(x_2)\, L_m(x_3)\, L_n(x_4)\, L_o(x_5)\, L_p(x_6).$$

Using the relations between the Laguerre polynomials, we want to get rid of all derivatives and variables to get a recurrence relation just as Pekeris did. Unfortunately, this is not possible for these coordinates. In the next chapter, we will analyze the problem that arises.

Chapter 7

Prefactors of the perimetric coordinates

Recall the recurrence and differential equation satisfied by the Laguerre polynomials.

$$\begin{aligned} xL_n''(x) &= (x-1)L_n'(x) - nL_n(x) \\ xL_n(x) &= -(n+1)L_{n+1}(x) + (2n+1)L_n(x) - nL_{n-1}(x) \\ xL_n'(x) &= nL_n(x) - nL_{n-1}(x)\,. \end{aligned}$$

These equations were used to eliminated all derivatives as well as all variables in Pekeris's original approach. Note that these relations are not able to deal with all possible prefactors and derivatives that might occur. It is true that any expression of the form $p(x)L_n(x)$ with a polynomial p can be reduced to a sum of Laguerre polynomials with prefactors that are independent of the variable x. In the same way, we can handle expressions of the form $p(x)L_n'(x)$ if the polynomial p has no constant term and expressions of the form $p(x)L_n''(x)$ if the polynomial p has no constant term and no term of order one. For these expressions, we can inductively compute equivalent forms. But the expressions xL_n'' and L_n' cause serious problems to the approach as they cannot be simplified to a sum of Laguerre polynomials with prefactors independent of x. Looking at the first relation, we can make up a new relation that helps us with this problem.

$$xL_n''(x) + L_n'(x) = xL_n'(x) - nL_n(x)$$

This equation tells us that the term $xL''_n(x) + L'_n(x)$ can be simplified into a convenient expression containing L_n and $xL'_n(x)$, which can then be simplified further. This means that in order to get rid of all terms xL''_n and L'_n we must ensure that the prefactors of the two expressions in the examined equation are the same ones. In Pekeris's approach, this was the case and he was able to eliminate all variables and derivatives as desired (see equation (14) of [4]). But with Frolov's coordinates for the Lithium atom, this condition is not satisfied and thus the approach fails. We now want to examine if it is possible to alter the coordinates in a way that the prefactors of xL''_n and L'_n match. In order to do this, it is helpful to take a look at Pekeris's work first.

The perimetric coordinates were first introduced by Coolidge and James in [5] to examine the convergence of Hylleraas' variational method. In this paper, they introduce the coordinates as

$$\begin{aligned} u &:= \delta\left(r_2 + r_{12} - r_1\right) \\ v &:= \delta\left(r_1 + r_{12} - r_2\right) \\ w &:= \delta\left(r_1 + r_2 - r_{12}\right), \end{aligned}$$

with a prefactor δ. For his work, Pekeris changed the prefactors from (δ, δ, δ) to $(\varepsilon, \varepsilon, 2\varepsilon)$. It is not a-priori clear why he chose the coordinates in this way, but the discussion above suggests that the reason might be that this choice of coordinates results in matching prefactors for xL''_n and L'_n. It was a surprise for us to see that the original perimetric coordinates would have worked in the same way as Pekeris's perimetric coordinates did. The MAPLE worksheets 'PekerisPrefactors' is an altered version of the worksheets we have developed so far. This worksheet computes the Hamilton operator for Helium in the perimetric coordinates with arbitrary prefactors a, b and c and compares the prefactors of xL''_n and L'_n in all three variables $x = u, v, w$. The surprising result is that any choice of the parameters a, b and c yields matching prefactors.

It is indeed interesting to see that the prefactors of the perimetric coordinates can be chosen arbitrarily but at the same time this result is unsatisfactory as it suggests that a change of the prefactors of Frolov's coordinates might not yield a different result from what we've seen before. The MAPLE worksheet 'FrolovPrefactors' computes the prefactors of xL_n'' and L_n' for an arbitrary choice of prefactors a, b, c, d, e and f for the six coordinates that Frolov derived. Using this worksheet, we can derive several equations that can never be satisfied by the prefactors. The worksheet gives two examples: A comparison of coefficients for the first coordinate shows, that $e = -f$ where both e and f should be positive numbers. The other comparisons for the fifth coordinate yield that a has to be zero. These contradictions show that the problem of the coordinates are in fact not the prefactors. While for Pekeris's coordinates every prefactor worked, the new coordinates do not seem to be suited to attack the problem in the same way.

Chapter 8

Outlook and open questions

So far, none of the attempts was able to transfer Pekeris's approach to the Lithium atom. There are other things that might be tried in order to get a result. The problem described in the last section could be avoided by multiplying the whole equation with $\prod_{i=1}^{6} x_i^2$. Note that after this operation there are no first or second derivatives of Laguerre polynomials left that do not have a prefactor of at least x^2. This means that all such expressions can be substituted by the desired form. The problem with this approach is that the relations between the Laguerre polynomials start to become extremely long. Instead of substitutions for terms of the form $x^3L(x)$ we now need substitutions for terms of the form $x^5L(x)$. MAPLE can quickly compute such relations but they would make the expression computed by the worksheet 'FrolovCoordinates', which already consists of 4884 terms, much longer. MAPLE would have massive problems in computing the coefficients that are needed to perform Pekeris's method. This task must be decomposed into several smaller computations that might lead to the desired recurrence relation. So far, we were not able to perform this computation.

If the described approach works, one must ask the question how accurate the result is. In this applied problem, the experimental values can serve as a reference but not as the sample solution in the mathematical sense. It would therefore be nice to know how much Frolov's linear approximations influence the result and of course how much it is biased by the omitted boundaries for the variables. The computed solution can only be seen as mathematically accurate if these influences are small enough.

While working on this project, we came along many different coordinates that might be used to perform Pekeris's approach. We described the normal coordinates which were accurate but too complicated and Frolov's coordinates which were easy too handle but not very accurate and eventually failed because the right terms in the differential equation did not cancel out. There are other coordinates which are in between these two approaches. One example are the coordinates p_{ij} which were used to compute the inverse transform of the normal coordinates. These coordinates are exact, nonlinear, but not as complicated as the normal coordinates. The inverses of these coordinates contain roots but only one per inverse which might make them easier to work with. They are not independent of each other but need to obey fewer restrictions than Frolov's coordinates do. Dropping these restrictions might not be as severe as the restrictions we dropped while working with Frolov's coordinates. For a precise discussion of these coordinates, see [8]. Another set of coordinates that might be helpful are the nonlinear coordinates described by P. S. C. Wang. For a reference, see [8]. These coordinates range independently from 0 to ∞, but they are not exact as they are derived purely from the triangular inequalities which does not suffice as we have seen earlier. The problem we had with these coordinates is that they are not as easy to invert as the p_{ij} and therefore might be too complicated to apply Pekeris's method.

Although it seems that there is no set of coordinates available in the short run that can transfer Pekeris's approach to higher dimensional systems, it is indeed worth a try to search for them. If found, they would provide a computationally less complex alternative to the variational methods that are used today.

Bibliography

[1] M. Reed, B. Simon, *Methods of Modern Mathematical Physics I: Functional Analysis* Academic Press, 1980

[2] M. Reed, B. Simon, *Methods of Modern Mathematical Physics II: Fourier Analysis, Self-Adjointness* Academic Press, 1975

[3] E. Hylleraas, *Über den Grundzustand des Heliumsatoms* Zeitschrift für Physik, Band 48, 1928

[4] C. L. Pekeris, *Ground State of Two-Electron Atoms* Physical Review, Volume 112, Number 5, 1958

[5] A. S. Coolidge, H. M. James, *On the Convergence of the Hylleraas Variational Method* Physical Review, Volume 51, 1937

[6] C. Koutschan, D. Zeilberger, *The 1958 Pekeris-Accad-Weizac Ground-Breaking Collaboration that computed the Ground States of Two-Electron Atoms (And its 2010 Redux)*, The Mathematical Intelligencer, 2011

[7] V. V. Albert, *Few-Parameter Exponentially Correlated Wavefunctions for the Ground State of Lithium*, Master Thesis, 2010

[8] B. P. Carter, *Four-Body S-State Coordinates*, The Journal of Chemical Physics, Volume 49, Number 8, 1968

[9] G. Allaire, S. M. Kaber, *Numerical linear algebra*, Springer, 2008

[10] A. M. Frolov, *Four-body perimetric coordinates*, Journal of Physics A: Mathematical and General, Volume 39, Number 50, 2006

Appendix. MAPLE worksheets

These worksheets can serve as a reference to the different approaches described in the thesis.

NormalCoordinates

```
> restart:
> apply(psi, a, b, c, d, e, f):
> #compute the laplacian in spherical coordinates
> fr1(x1, x2, x3, x4, x5, x6, x7, x8, x9) := sqrt(x1^2+x2^2+x3^2):
> fr2(x1, x2, x3, x4, x5, x6, x7, x8, x9) := sqrt(x4^2+x5^2+x6^2):
> fr3(x1, x2, x3, x4, x5, x6, x7, x8, x9) := sqrt(x7^2+x8^2+x9^2):
> fr12(x1, x2, x3, x4, x5, x6, x7, x8, x9) :=
  sqrt((x4-x1)^2+(x5-x2)^2+(x6-x3)^2):
> fr13(x1, x2, x3, x4, x5, x6, x7, x8, x9) :=
  sqrt((x7-x1)^2+(x8-x2)^2+(x9-x3)^2):
> fr23(x1, x2, x3, x4, x5, x6, x7, x8, x9) :=
  sqrt((x7-x4)^2+(x8-x5)^2+(x9-x6)^2):
> with(LinearAlgebra):
> with(VectorCalculus):
> eq := simplify((1/2)*Laplacian(psi(
  fr1(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr2(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr3(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr12(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr13(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr23(x1, x2, x3, x4, x5, x6, x7, x8, x9)),
  [x1, x2, x3, x4, x5, x6, x7, x8, x9])+
  2*(E+Z/r1+Z/r2+Z/r3-1/r12-1/r13-1/r23),
  {x1^2+x2^2+x3^2 = r1^2, x4^2+x5^2+x6^2 = r2^2,
  x7^2+x8^2+x9^2 = r3^2, (x4-x1)^2+(x5-x2)^2+(x6-x3)^2 = r12^2,
  (x7-x1)^2+(x8-x2)^2+(x9-x3)^2 = r13^2,
  (x7-x4)^2+(x8-x5)^2+(x9-x6)^2 = r23^2}):
> eqsimple := sort(collect(simplify(eq, assume = positive), D)):
> #transform into normal coordinates
> apply(F, a, b, c, d, e, f):
> phi(a, b, c, d, e, f) := exp(-(1/2)*a-(1/2)
  *b-(1/2)*c-(1/2)*d-(1/2)*e-(1/2)*f)*F(a, b, c, d, e, f):
> #compute the normal coordinates using the Cholesky decomposition
> g11 := simplify(sqrt(r1^2), assume = positive):
> g22 := simplify(sqrt(r2^2-g21^2), assume = positive)
> g31 := simplify((1/2)*(r3^2+r1^2-r13^2)/r1, assume = positive):
> g32 := simplify((1/2*(r3^2+r2^2-r23^2)-g31*g21)/g22,
  assume = positive):
> g33 := simplify(sqrt(r3^2-g31^2-g32^2), assume = positive):
> x1(r1, r2, r3, r12, r13, r23) := r1:
> x2(r1, r2, r3, r12, r13, r23) := -(1/2)*(-r1^2-r2^2+r12^2)/r1:
> y2(r1, r2, r3, r12, r13, r23) := (1/2)*sqrt(2*r2^2*r1^2-r1^4
  +2*r12^2*r1^2-r2^4+2*r12^2*r2^2-r12^4)/r1:
> x3(r1, r2, r3, r12, r13, r23) := -(1/2)*(-r3^2-r1^2+r13^2)/r1:
> y3(r1, r2, r3, r12, r13, r23) := -(1/2)*(-r3^2*r1^2-r2^2*r1^2
  +2*r23^2*r1^2+r3^2*r2^2-r3^2*r12^2+r1^4-r12^2*r1^2-r13^2*r1^2
  -r2^2*r13^2+r13^2*r12^2)/(r1*sqrt(2*r2^2*r1^2-r1^4+2*r12^2*r1^2
  -r2^4+2*r12^2*r2^2-r12^4)):
> z3(r1, r2, r3, r12, r13, r23) := ((r12^2*r2^2*r1^2-r3^2*r2^2
  *r12^2-r13^2*r3^2*r2^2-r13^2*r3^2*r12^2-r2^2*r13^2*r12^2+r23^2
  *r3^2*r2^2-r23^2*r3^2*r12^2-r23^2*r13^2*r2^2+r23^2*r13^2*r12^2
  -r3^2*r12^2*r1^2+r3^2*r12^4+r3^4*r12^2+r13^2*r3^2*r1^2-r2^2
  *r13^2*r1^2+r2^4*r13^2+r2^2*r13^4-r23^2*r3^2*r1^2-r23^2*r2^2
  *r1^2-r23^2*r12^2*r1^2-r23^2*r13^2*r1^2+r23^4*r1^2+r23^2*r1^4)
  /(-2*r2^2*r1^2+r1^4-2*r12^2*r1^2+r2^4-2*r12^2*r2^2+r12^4))^(1/2):
> #compute the operator in the normal coordinates step by step
> #computation in one step runs out of memory
> eqnormal1 := (1/2)*(4*E*r1*r12*r13*r2*r23*r3
  *(1/(r1*r12*r13*r2*r23*r3))):
> eqnormal2 := simplify((1/2)/r1r12r13r2r23r3*(-r1^3*r12*r2*r23
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3, r13))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2, r13^2 =
  (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal3 := simplify((1/2)/r1r12r13r2r23r3
  *(-r1^3*r13*r23*r3*(diff(phi(
  x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2, r12))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2, r13^2 =
  (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal4 := simplify((1/2)*r1^2*r12*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r1, r13))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+
  (y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal5 := simplify((1/2)*r1^2*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r1, r12))
  /r1r12r13r2r23r3,{r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal6 := simplify((1/2)/r1r12r13r2r23r3
  *(-r1*r12^3*r2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)),r13, r23))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2
  +y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal7 := simplify((1/2)*r1*r12^2*r13*r2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r12, r23))
  /r1r12r13r2r23r3,{r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal8 := simplify((1/2)*r1*r12^2*r13*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2, r12))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
```

```
> eqnormal9 := simplify((1/2)*r1*r12^2*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r12, r13))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal10 := simplify((1/2)*r1*r12*r13^2*r2*r23
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3, r13))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2
  +y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal11 := simplify((1/2)*r1*r12*r13^2*r2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)),
  r13, r23))/r1r12r13r2r23r3, {r1 = x1, r12^2 =
  (x1-x2)^2+y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2,r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal12 := simplify((1/2)/r1r12r13r2r23r3
  *(-r1*r12*r13*r2^3*(diff(phi(
  x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3, r23))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal13 := simplify((1/2)*r1*r12*r13*r2^2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2, r23))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2
  +y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal14 := simplify((1/2)*r1*r12*r13*r2*r23^2
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3, r23))
  /r1r12r13r2r23r3, {r1 = x1,
  r12^2 = (x1-x2)^2+y2^2,r13^2 = (x1-x3)^2
  +y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal15 := simplify((1/2)*r1*r12*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r1, r1))
  /r1r12r13r2r23r3, {r1 = x1,
  r12^2 = (x1-x2)^2+y2^2, r13^2 = (x1-x3)^2
  +y3^2+z3^2,r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+
  (y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal16 := simplify((1/2)*r1*r12*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2, r2))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
```

```
> eqnormal17 := simplify((1/2)*r1*r12*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3, r3))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2,r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2,r3^2 = x3^2+y3^2+z3^2}):
> eqnormal18 := simplify((1/2)/r1r12r13r2r23r3
  *(2*r1*r12*r13*r2*r23*r3*(diff(phi(
  x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r12, r12))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2, r13^2 =
  (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal19 := simplify((1/2)/r1r12r13r2r23r3
  *(2*r1*r12*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r13, r13))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2, r13^2 =
  (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal20 := simplify((1/2)/r1r12r13r2r23r3
  *(2*r1*r12*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r23, r23))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal21 := simplify((1/2)*r1*r12*r13*r2*r3^2
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3, r23))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2
  +y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal22 := simplify((1/2)*r1*r12*r13*r23^2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2, r23))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal23 := simplify((1/2)/r1r12r13r2r23r3
  *(-r1*r12*r13*r3^3*(diff(phi(
  x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2, r23))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2, r13^2 =
  (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal24 := simplify((1/2)*r1*r12*r2*r23^2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r13, r23))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2,r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal25 := simplify((1/2)*r1*r12*r2*r23*r3^2
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3, r13))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal26 := simplify((1/2)/r1r12r13r2r23r3
  *(-r1*r13^3*r2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r12, r23))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal27 := simplify((1/2)*r1*r13^2*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)),r12, r13))
  /r1r12r13r2r23r3,{r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal28 := simplify((1/2)*r1*r13*r2^2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2, r12))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal29 := simplify((1/2)*r1*r13*r2*r23^2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)),
  r12, r23))/r1r12r13r2r23r3, {r1 = x1,
  r12^2 = (x1-x2)^2+y2^2, r13^2= (x1-x3)^2
  +y3^2+z3^2, r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal30 := simplify((1/2)/r1r12r13r2r23r3
  *(-r1*r2*r23^3*r3*(diff(phi(
  x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r12, r13))),
  {r1 = x1, r12^2 =(x1-x2)^2+y2^2, r13^2 = (x1-x3)^2
  +y3^2+z3^2, r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal31 := simplify((1/2)*r12^2*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r1, r12))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 =(x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal32 := simplify((1/2)*r12*r13^2*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r1, r13))
  /r1r12r13r2r23r3, {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 = (x1-x3)^2 +y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal33 := simplify((1/2)/r1r12r13r2r23r3
  *(-r12*r2*r23*r3^3*(diff(phi(
  x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r1, r13))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2, r13^2
  = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+
  (y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal34 := simplify((1/2)/r1r12r13r2r23r3
  *(-r13*r2^3*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)),r1, r12))),
  {r1 = x1, r12^2 = (x1-x2)^2
  +y2^2, r13^2 =(x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal35 := simplify((1/2)/r1r12r13r2r23r3
  *(4*Z*r1*r12*r13*r2*r23), {r1 = x1,
  r12^2 = (x1-x2)^2+y2^2, r13^2 = (x1-x3)^2
  +y3^2+z3^2, r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
```

```
> eqnormal36 := simplify((1/2)/r1r12r13r2r23r3
  *(4*Z*r1*r12*r13*r23*r3), {r1 = x1,
  r12^2 = (x1-x2)^2+y2^2, r13^2 = (x1-x3)^2
  +y3^2+z3^2, r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal37 := simplify((1/2)/r1r12r13r2r23r3
  *(4*Z*r12*r13*r2*r23*r3), {r1 = x1, r12^2 =
  (x1-x2)^2+y2^2, r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal38 := simplify((1/2)/r1r12r13r2r23r3
  *(2*r1*r12*r13*r2*r23*(diff(phi(
  x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r3))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2
  = x2^2+y2^2,r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal39 := simplify((1/2)/r1r12r13r2r23r3
  *(4*r1*r12*r13*r2*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r23))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2
  +y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal40 := simplify((1/2)/r1r12r13r2r23r3
  *(2*r1*r12*r13*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r2))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2,
  r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal41 := simplify((1/2)
  /r1r12r13r2r23r3*(4*r1*r12*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r13))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2
  +y2^2, r23^2 = (x2-x3)^2+(y2-y3)^2
  +z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal42 := simplify((1/2)
  /r1r12r13r2r23r3*(4*r1*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r12))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2, r13^2 =
  (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal43 := simplify((1/2)/r1r12r13r2r23r3
  *(2*r12*r13*r2*r23*r3
  *(diff(phi(x1(r1, r2, r3, r12, r13, r23),
  x2(r1, r2, r3, r12, r13, r23),
  y2(r1, r2, r3, r12, r13, r23),
  x3(r1, r2, r3, r12, r13, r23),
  y3(r1, r2, r3, r12, r13, r23),
  z3(r1, r2, r3, r12, r13, r23)), r1))),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormal44 := simplify((1/2)/r1r12r13r2r23r3
  *(-4*r1*r12*r13*r2*r3), {r1 = x1,
  r12^2 = (x1-x2)^2+y2^2, r13^2 = (x1-x3)^2
  +y3^2+z3^2, r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal45 := simplify((1/2)/r1r12r13r2r23r3
  *(-4*r1*r12*r2*r23*r3), {r1 = x1,
  r12^2 = (x1-x2)^2+y2^2, r13^2 = (x1-x3)^2
  +y3^2+z3^2, r2^2 = x2^2+y2^2, r23^2 = (x2-x3)^2
  +(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> eqnormal46 := simplify((1/2)
  /r1r12r13r2r23r3*(-4*r1*r13*r2*r23*r3),
  {r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2,
  r3^2 = x3^2+y3^2+z3^2}):
> eqnormalFinal :=simplify(eqnormal1+eqnormal2+eqnormal3
  +eqnormal4+eqnormal5+eqnormal6+eqnormal7+eqnormal8+eqnormal9
  +eqnormal10+eqnormal11+eqnormal12+eqnormal13+eqnormal14
  +eqnormal15+eqnormal16+eqnormal17+eqnormal18+eqnormal19
  +eqnormal20+eqnormal21+eqnormal22+eqnormal23+eqnormal24
  +eqnormal25+eqnormal26+eqnormal27+eqnormal28+eqnormal29
  +eqnormal30+eqnormal31+eqnormal32+eqnormal33+eqnormal34
  +eqnormal35+eqnormal36+eqnormal37+eqnormal38+eqnormal39
  +eqnormal40+eqnormal41+eqnormal42+eqnormal43+eqnormal44
  +eqnormal45+eqnormal46,{r1 = x1, r12^2 = (x1-x2)^2+y2^2,
  r13^2 = (x1-x3)^2+y3^2+z3^2, r2^2 = x2^2+y2^2,
  r23^2 = (x2-x3)^2+(y2-y3)^2+z3^2, r3^2 = x3^2+y3^2+z3^2}):
> #not enough memory for these computations either
```

FrolovCoordinates and FrolovPrefactors

The first parts of these two worksheets are identical. The only difference is that in the worksheet 'FrolovCoordinates' the special choice $(a, b, c, d, e, f) = (1, 1, 1, 1, 1, 1)$ is made. The first part for both worksheets is:

```
> restart:
> #compute the laplacian in spherical coordinates
> apply(psi, a, b, c, d, e, f):
> fr1(x1,x2,x3,x4,x5,x6,x7,x8,x9):= sqrt(x1^2+x2^2+x3^2):
> fr2(x1,x2,x3,x4,x5,x6,x7,x8,x9):=sqrt(x4^2+x5^2+x6^2):
> fr3(x1,x2,x3,x4,x5,x6,x7,x8,x9):=sqrt(x7^2+x8^2+x9^2):
> fr12(x1,x2,x3,x4,x5,x6,x7,x8,x9):=sqrt((x4-x1)^2
  +(x5-x2)^2+(x6-x3)^2):
> fr13(x1,x2,x3,x4,x5,x6,x7,x8,x9):=sqrt((x7-x1)^2
  +(x8-x2)^2+(x9-x3)^2):
> fr23(x1,x2,x3,x4,x5,x6,x7,x8,x9):=sqrt((x7-x4)^2
  +(x8-x5)^2+(x9-x6)^2):
> with(LinearAlgebra): with(VectorCalculus):
> eq := simplify((1/2)*Laplacian(psi(
  fr1(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr2(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr3(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr12(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr13(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr23(x1, x2, x3, x4, x5, x6, x7, x8, x9)),
  [x1, x2, x3, x4, x5, x6, x7, x8, x9])
  +(2*(E+Z/r1+Z/r2+Z/r3-1/r12-1/r13-1/r23))
  *psi(fr1(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr2(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr3(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr12(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr13(x1, x2, x3, x4, x5, x6, x7, x8, x9),
  fr23(x1, x2, x3, x4, x5, x6, x7, x8, x9)),
  {x1^2+x2^2+x3^2 = r1^2, x4^2+x5^2+x6^2 = r2^2,
  x7^2+x8^2+x9^2 = r3^2, (x4-x1)^2+(x5-x2)^2
  +(x6-x3)^2 = r12^2, (x7-x1)^2+(x8-x2)^2
  +(x9-x3)^2 = r13^2, (x7-x4)^2+(x8-x5)^2
  +(x9-x6)^2 = r23^2}):
> eqsimple := expand(sort(collect(simplify(eq,
  assume = positive), D))):
> #introduce frolov coordinates
> #a,b,c,d,e,f can be substituted by any positive numbers
> a1 := a: a2 := b: a3 := c: a4 := d: a5 := e: a6 := f:
> x1(r1,r2,r3,r12,r13,r23):=a11/(2)(r1+r2-r12):
> x2(r1,r2,r3,r12,r13,r23):=a21/(2)(r1+r12-r2):
> x3(r1,r2,r3,r12,r13,r23):=a31/(2)(r12+r2-r1):
> x4(r1,r2,r3,r12,r13,r23):=a41/(2)(r3+r13-r1):
> x5(r1,r2,r3,r12,r13,r23):=a51/(2)(r23+r12-r13):
> x6(r1,r2,r3,r12,r13,r23):=a61/(2)(r23+r2-r3):
> #substitute the Ansatz into the differential equation
> #this is done via string mainpulation
> phi(a,b,c,d,e,f):=exp(-1/2*(a+b+c+d+e+f))
  *L(a)*L(b)*L(c)*L(d)*L(e)*L(f):
> eqStr := convert(eqsimple, string):
> with(StringTools):
> for i to 6 do
  if i < 4 then hi := i
  elif i = 4 then hi := 12
  elif i = 5 then hi := 13
  elif i = 6 then hi := 23
  end if:
  eqStr := SubstituteAll(eqStr, Join(["D[",
  convert(i, string), "]psi)(r1,r2,r3,r12,r13,r23)"],
  ""),Join(["diff(phi(x1(r1,r2,r3,r12,r13,r23),
  x2(r1,r2,r3,r12,r13,r23),x3(r1,r2,r3,r12,r13,r23),
  x4(r1,r2,r3,r12,r13,r23),x5(r1,r2,r3,r12,r13,r23),
  x6(r1,r2,r3,r12,r13,r23),r", convert(hi, string),
  ")"], "")):
  for j to 6 do
  if j < 4 then hj := j
  elif j = 4 then hj := 12
  elif j = 5 then hj := 13
  elif j = 6 then hj := 23
  end if:
  eqStr := SubstituteAll(eqStr, Join(["D[",
  convert(i, string),",", convert(j, string), "]
  (psi)(r1,r2,r3,r12,r13,r23)"], ""),Join
  (["diff(phi(x1(r1,r2,r3,r12,r13,r23),
  x2(r1,r2,r3,r12,r13,r23),x3(r1,r2,r3,r12,r13,r23),
  x4(r1,r2,r3,r12,r13,r23),x5(r1,r2,r3,r12,r13,r23),
  x6(r1,r2,r3,r12,r13,r23)),r", convert(hi, string),
  ",r", convert(hj, string), ")"], "")):
  end do
  end do:
> eqStr := SubstituteAll(eqStr, "psi(r1,r2,r3,r12,r13,r23)",
  "phi(x1(r1,r2,r3,r12,r13,r23),x2(r1,r2,r3,r12,r13,r23),
  x3(r1,r2,r3,r12,r13,r23),x4(r1,r2,r3,r12,r13,r23),
  x5(r1,r2,r3,r12,r13,r23),x6(r1,r2,r3,r12,r13,r23))"):
> eqparsed := parse(eqStr):
```

```
> #invert the coordinates to simplify
> r1sim := (a2*x1+a1*x2)/(a1*a2)
> r2sim := (a3*x1+a1*x3)/(a1*a3):
> r3sim := (a4*a5*a6*x1+a1*a5*a6*x4+a1*a4*a6*x5-a1*a4*a5*x6)
  /(a1*a4*a5*a6):
> r12sim := (a3*x2+a2*x3)/(a2*a3):
> r13sim := (a4*a5*a6*x2+a2*a5*a6*x4+a2*a4*a5*x6-a2*a4*a6*x5)
  /(a2*a4*a5*a6):
> r23sim := (a3*a5*a6*x4+a3*a4*a6*x5+a3*a4*a5*x6-a4*a5*a6*x3)
  /(a3*a4*a5*a6):
> eqfrolov := subs(r1 = r1sim, r2 = r2sim, r3 = r3sim,
  r12 = r12sim,r13 = r13sim, r23 = r23sim, eqparsed):
> #further simplifications
> eqfrolov := numer(eqfrolov):
> eqfrolov := simplify(expand(eqfrolov
  /(exp(-(1/2)*x2-(1/2)*x3)*exp(-(1/2)*x3-(1/2)*x1)
  *exp(-(1/2)*x6-(1/2)*x4-(1/2)*x5+(1/2)*x3)))):
```

At this point the worksheets continue differently. The worksheet 'FrolovCoordinates' continues to apply Pekeris' method on the expression 'eqfrolov' containing of 4884 terms.

```
> nops(eqfrolov):
> #again use string manipulation
> a := convert(eqfrolov, string):
> a := SubstituteAll(a, "L(x1)", "Lx1"):
> a := SubstituteAll(a, "L(x2)", "Lx2"):
> a := SubstituteAll(a, "L(x3)", "Lx3"):
> a := SubstituteAll(a, "L(x4)", "Lx4"):
> a := SubstituteAll(a, "L(x5)", "Lx5"):
```

```
> a := SubstituteAll(a, "L(x6)", "Lx6"):
> a := SubstituteAll(a, "D(L)(x1)", "DLx1"):
> a := SubstituteAll(a, "D(L)(x2)", "DLx2"):
> a := SubstituteAll(a, "D(L)(x3)", "DLx3"):
> a := SubstituteAll(a, "D(L)(x4)", "DLx4"):
> a := SubstituteAll(a, "D(L)(x5)", "DLx5"):
> a := SubstituteAll(a, "D(L)(x6)", "DLx6"):
> a := SubstituteAll(a, "`@@`(D,2)(L)(x1)", "DDLx1"):
> a := SubstituteAll(a, "`@@`(D,2)(L)(x2)", "DDLx2"):
> a := SubstituteAll(a, "`@@`(D,2)(L)(x3)", "DDLx3"):
> a := SubstituteAll(a, "`@@`(D,2)(L)(x4)", "DDLx4"):
> a := SubstituteAll(a, "`@@`(D,2)(L)(x5)", "DDLx5"):
> a := SubstituteAll(a, "`@@`(D,2)(L)(x6)", "DDLx6"):
> eq := parse(a):
> #collect terms that can be simplified
> eq := simplify(eq, {x1^3*Lx1 = x3Lx1}):
> eq := simplify(eq, {x2^3*Lx2 = x3Lx2}):
> eq := simplify(eq, {x3^3*Lx3 = x3Lx3}):
> eq := simplify(eq, {x4^3*Lx4 = x3Lx4}):
> eq := simplify(eq, {x5^3*Lx5 = x3Lx5}):
> eq := simplify(eq, {x6^3*Lx6 = x3Lx6}):
> eq := simplify(eq, {x1^3*DLx1 = x3DLx1}):
> eq := simplify(eq, {x2^3*DLx2 = x3DLx2}):
> eq := simplify(eq, {x3^3*DLx3 = x3DLx3}):
> eq := simplify(eq, {x4^3*DLx4 = x3DLx4}):
> eq := simplify(eq, {x5^3*DLx5 = x3DLx5}):
> eq := simplify(eq, {x6^3*DLx6 = x3DLx6}):
> eq := simplify(eq, {x1^3*DDLx1 = x3DDLx1}):
> eq := simplify(eq, {x2^3*DDLx2 = x3DDLx2}):
> eq := simplify(eq, {x3^3*DDLx3 = x3DDLx3}):
```

```
> eq := simplify(eq, {x4^3*DDLx4 = x3DDLx4}):
> eq := simplify(eq, {x5^3*DDLx5 = x3DDLx5}):
> eq := simplify(eq, {x6^3*DDLx6 = x3DDLx6}):
> eq := simplify(eq, {x1^2*Lx1 = x2Lx1}):
> eq := simplify(eq, {x2^2*Lx2 = x2Lx2}):
> eq := simplify(eq, {x3^2*Lx3 = x2Lx3}):
> eq := simplify(eq, {x4^2*Lx4 = x2Lx4}):
> eq := simplify(eq, {x5^2*Lx5 = x2Lx5}):
> eq := simplify(eq, {x6^2*Lx6 = x2Lx6}):
> eq := simplify(eq, {x1^2*DLx1 = x2DLx1}):
> eq := simplify(eq, {x2^2*DLx2 = x2DLx2}):
> eq := simplify(eq, {x3^2*DLx3 = x2DLx3}):
> eq := simplify(eq, {x4^2*DLx4 = x2DLx4}):
> eq := simplify(eq, {x5^2*DLx5 = x2DLx5}):
> eq := simplify(eq, {x6^2*DLx6 = x2DLx6}):
> eq := simplify(eq, {x1^2*DDLx1 = x2DDLx1}):
> eq := simplify(eq, {x2^2*DDLx2 = x2DDLx2}):
> eq := simplify(eq, {x3^2*DDLx3 = x2DDLx3}):
> eq := simplify(eq, {x4^2*DDLx4 = x2DDLx4}):
> eq := simplify(eq, {x5^2*DDLx5 = x2DDLx5}):
> eq := simplify(eq, {x6^2*DDLx6 = x2DDLx6}):
> eq := simplify(eq, {x1*Lx1 = xLx1}):
> eq := simplify(eq, {x2*Lx2 = xLx2}):
> eq := simplify(eq, {x3*Lx3 = xLx3}):
> eq := simplify(eq, {x4*Lx4 = xLx4}):
> eq := simplify(eq, {x5*Lx5 = xLx5}):
> eq := simplify(eq, {x6*Lx6 = xLx6}):
> eq := simplify(eq, {x1*DLx1 = xDLx1}):
> eq := simplify(eq, {x2*DLx2 = xDLx2}):
> eq := simplify(eq, {x3*DLx3 = xDLx3}):
```

```
> eq := simplify(eq, {x4*DLx4 = xDLx4}):
> eq := simplify(eq, {x5*DLx5 = xDLx5}):
> eq := simplify(eq, {x6*DLx6 = xDLx6}):
> eq := simplify(eq, {x1*DDLx1 = xDDLx1}):
> eq := simplify(eq, {x2*DDLx2 = xDDLx2}):
> eq := simplify(eq, {x3*DDLx3 = xDDLx3}):
> eq := simplify(eq, {x4*DDLx4 = xDDLx4}):
> eq := simplify(eq, {x5*DDLx5 = xDDLx5}):
> eq := simplify(eq, {x6*DDLx6 = xDDLx6}):
> #Laguerre relations
> #placeholders for the Laguerre polynomials
> Lx1 := u^k:
> Lx2 := v^l:
> Lx3 := w^m:
> Lx4 := x^n:
> Lx5 := y^o:
> Lx6 := z^p:
> #compute the relations
> xDDLx1 := -k*subs(k = k-1, Lx1)-DLx1:
> xDDLx2 := -l*subs(l = l-1, Lx2)-DLx2:
> xDDLx3 := -m*subs(m = m-1, Lx3)-DLx3:
> xDDLx4 := -n*subs(n = n-1, Lx4)-DLx4:
> xDDLx5 := -o*subs(o = o-1, Lx5)-DLx5:
> xDDLx6 := -p*subs(p = p-1, Lx6)-DLx6:
> xLx1 := -(k+1)*subs(k = k+1, Lx1)+(2*k+1)
  *Lx1-k*subs(k = k-1, Lx1):
> x2Lx1 := -(k+1)*subs(k = k+1, xLx1)+(2*k+1)
  *xLx1-k*subs(k = k-1, xLx1):
> x3Lx1 := -(k+1)*subs(k = k+1, x2Lx1)+(2*k+1)
  *x2Lx1-k*subs(k = k-1, x2Lx1):
```

```
> xDLx1 := k*Lx1-k*subs(k = k-1, Lx1):
> x2DLx1 := k*xLx1-k*subs(k = k-1, xLx1):
> x3DLx1 := k*x2Lx1-k*subs(k = k-1, x2Lx1):
> x2DDLx1 := -k*subs(k = k-1, xLx1)-xDLx1:
> x3DDLx1 := -k*subs(k = k-1, x2Lx1)-x2DLx1:
> xLx2 := subs(k = l, u = v, xLx1):
> x2Lx2 := subs(k = l, u = v, x2Lx1):
> x3Lx2 := subs(k = l, u = v, x3Lx1):
> xDLx2 := subs(k = l, u = v, xDLx1):
> x2DLx2 := subs(k = l, u = v, x2DLx1):
> x3DLx2 := subs(k = l, u = v, x3DLx1):
> x2DDLx2 := subs(k = l, u = v, x2DDLx1):
> x3DDLx2 := subs(k = l, u = v, x3DDLx1):
> xLx3 := subs(k = m, u = w, xLx1):
> x2Lx3 := subs(k = m, u = w, x2Lx1):
> x3Lx3 := subs(k = m, u = w, x3Lx1):
> xDLx3 := subs(k = m, u = w, xDLx1):
> x2DLx3 := subs(k = m, u = w, x2DLx1):
> x3DLx3 := subs(k = m, u = w, x3DLx1):
> x2DDLx3 := subs(k = m, u = w, x2DDLx1):
> x3DDLx3 := subs(k = m, u = w, x3DDLx1):
> xLx4 := subs(k = n, u = x, xLx1):
> x2Lx4 := subs(k = n, u = x, x2Lx1):
> x3Lx4 := subs(k = n, u = x, x3Lx1):
> xDLx4 := subs(k = n, u = x, xDLx1):
> x2DLx4 := subs(k = n, u = x, x2DLx1):
> x3DLx4 := subs(k = n, u = x, x3DLx1):
> x2DDLx4 := subs(k = n, u = x, x2DDLx1):
> x3DDLx4 := subs(k = n, u = x, x3DDLx1):
> xLx5 := subs(k = o, u = y, xLx1):
```

```
> x2Lx5 := subs(k = o, u = y, x2Lx1):
> x3Lx5 := subs(k = o, u = y, x3Lx1):
> xDLx5 := subs(k = o, u = y, xDLx1):
> x2DLx5 := subs(k = o, u = y, x2DLx1):
> x3DLx5 := subs(k = o, u = y, x3DLx1):
> x2DDLx5 := subs(k = o, u = y, x2DDLx1):
> x3DDLx5 := subs(k = o, u = y, x3DDLx1):
> xLx6 := subs(k = p, u = z, xLx1):
> x2Lx6 := subs(k = p, u = z, x2Lx1):
> x3Lx6 := subs(k = p, u = z, x3Lx1):
> xDLx6 := subs(k = p, u = z, xDLx1):
> x2DLx6 := subs(k = p, u = z, x2DLx1):
> x3DLx6 := subs(k = p, u = z, x3DLx1):
> x2DDLx6 := subs(k = p, u = z, x2DDLx1):
> x3DDLx6 := subs(k = p, u = z, x3DDLx1):
> #unfortunately, not all derivatives vanished
```

The second part of the worksheet 'FrolovPrefactors' computes and compares the prefactors.

```
> #find the coefficients for x1
> coeffDLx1 := coeff(eqfrolov, (D(L))(x1)):
> coeffDLx1 := simplify(coeffDLx1, {x1 = 0, L(x2) = 1, L(x3) = 1,
  L(x4) = 1, L(x5) = 1, L(x6) = 1, (D(L))(x2) = 1,
  (D(L))(x3) = 1, (D(L))(x4) = 1, (D(L))(x5) = 1, (D(L))(x6) = 1}):
> coeffDDLx1 := coeff(eqfrolov, ((D@@2)(L))(x1)):
> coeffDDLx1 := simplify(coeffDDLx1, {x1^2 = 0, L(x2) = 1,
  L(x3) = 1, L(x4) = 1, L(x5) = 1, L(x6) = 1}):
> coeffDDLx1 := coeffDDLx1/x1:
> #x2
> coeffDLx2 := coeff(eqfrolov, (D(L))(x2)):
> coeffDLx2 := simplify(coeffDLx2, {x2 = 0, L(x1) = 1, L(x3) = 1,
  L(x4) = 1, L(x5) = 1, L(x6) = 1, (D(L))(x1) = 1,
  (D(L))(x3) = 1, (D(L))(x4) = 1, (D(L))(x5) = 1, (D(L))(x6) = 1}):
> coeffDDLx2 := coeff(eqfrolov, ((D@@2)(L))(x2)):
> coeffDDLx2 := simplify(coeffDDLx2, {x2^2 = 0, L(x1) = 1,
  L(x3) = 1, L(x4) = 1, L(x5) = 1, L(x6) = 1}):
> coeffDDLx2 := coeffDDLx2/x2:
> #x3
> coeffDLx3 := coeff(eqfrolov, (D(L))(x3)):
> coeffDLx3 := simplify(coeffDLx3, {x3 = 0, L(x1) = 1, L(x2) = 1,
  L(x4) = 1, L(x5) = 1, L(x6) = 1, (D(L))(x1) = 1,
  (D(L))(x2) = 1, (D(L))(x4) = 1, (D(L))(x5) = 1, (D(L))(x6) = 1}):
> coeffDDLx3 := coeff(eqfrolov, ((D@@2)(L))(x3)):
> coeffDDLx3 := simplify(coeffDDLx3, {x3^2 = 0, L(x1) = 1,
  L(x2) = 1,L(x4) = 1, L(x5) = 1, L(x6) = 1}):
> coeffDDLx3 := coeffDDLx3/x3:
> #x4
> coeffDLx4 := coeff(eqfrolov, (D(L))(x4)):
> coeffDLx4 := simplify(coeffDLx4, {x4 = 0, L(x1) = 1, L(x2) = 1,
  L(x3) = 1, L(x5) = 1, L(x6) = 1, (D(L))(x1) = 1,
  (D(L))(x2) = 1, (D(L))(x3) = 1, (D(L))(x5) = 1, (D(L))(x6) = 1}):
> coeffDDLx4 := coeff(eqfrolov, ((D@@2)(L))(x4)):
> coeffDDLx4 := simplify(coeffDDLx4, {x4^2 = 0, L(x1) = 1,
  L(x2) = 1, L(x3) = 1, L(x5) = 1, L(x6) = 1}):
> coeffDDLx4 := coeffDDLx4/x4:
> #x5
> coeffDLx5 := coeff(eqfrolov, (D(L))(x5)):
> coeffDLx5 := simplify(coeffDLx5, {x5 = 0, L(x1) = 1, L(x2) = 1,
  L(x3) = 1, L(x4) = 1, L(x6) = 1, (D(L))(x1) = 1, (D(L))(x2) = 1,
  (D(L))(x3) = 1, (D(L))(x4) = 1, (D(L))(x6) = 1}):
> coeffDDLx5 := coeff(eqfrolov, ((D@@2)(L))(x5)):
> coeffDDLx5 := simplify(coeffDDLx5, {x5^2 = 0, L(x1) = 1,
  L(x2) = 1, L(x3) = 1, L(x4) = 1, L(x6) = 1}):
> coeffDDLx5 := coeffDDLx5/x5:
> #x6
> coeffDLx6 := coeff(eqfrolov, (D(L))(x6)):
> coeffDLx6 := simplify(coeffDLx6, {x6 = 0, L(x1) = 1, L(x2) = 1,
  L(x3) = 1, L(x4) = 1, L(x5) = 1, (D(L))(x1) = 1, (D(L))(x2) = 1,
  (D(L))(x3) = 1, (D(L))(x4) = 1, (D(L))(x5) = 1}):
> coeffDDLx6 := coeff(eqfrolov, ((D@@2)(L))(x6)):
> coeffDDLx6 := simplify(coeffDDLx6, {x6^2 = 0, L(x1) = 1,
  L(x2) = 1, L(x3) = 1, L(x4) = 1, L(x5) = 1}):
> coeffDDLx6 := coeffDDLx6/x6:
> #comparing examples
> #x1
> coeff(coeff(coeff(coeff(coeff(coeffDLx1, x2, 3), x3, 1),
x4, 1), x5, 1), x6, 0);
> coeff(coeff(coeff(coeff(coeff(coeffDDLx1, x2, 3), x3, 1),
x4, 1), x5, 1), x6, 0);
> #x5
> factor(coeff(coeff(coeff(coeff(coeff(coeffDLx5, x1, 0),
  x2, 3), x3, 0), x4, 2), x6, 1));
> coeff(coeff(coeff(coeff(coeff(coeffDDLx5, x1, 0), x2,
  3), x3, 0), x4, 2), x6, 1);
> factor(coeff(coeff(coeff(coeff(coeff(coeffDLx5, x1, 3),
  x2, 2), x3, 0), x4, 1), x6, 0));
> coeff(coeff(coeff(coeff(coeff(coeffDDLx5, x1, 3), x2,
  2), x3, 0), x4, 1), x6, 0);
> factor(coeff(coeff(coeff(coeff(coeff(coeffDLx5, x1, 1),
  x2, 1), x3, 2), x4, 1), x6, 1));
> coeff(coeff(coeff(coeff(coeff(coeffDDLx5, x1, 1), x2,
  1), x3, 2), x4, 1), x6, 1);
> factor(coeff(coeff(coeff(coeff(coeff(coeffDLx5, x1, 1),
  x2, 2), x3, 1), x4, 1), x6, 1));
> coeff(coeff(coeff(coeff(coeff(coeffDDLx5, x1, 1), x2, 2),
  x3, 1), x4, 1), x6, 1);
```

The output of this worksheet is

(1) $2a^5c^3bd^3e^3f^5 + 2a^5c^3bd^3e^4f^4$

(2) 0

(3) $-2a^4bc^4d^2e^5f^3(-b + c + f + a)$

(4) 0

(5) $2ab^2c^4d^3e^5f^4(-b + c - d + 2f + a)$

(6) 0

(7) $-2a^3b^3c^2d^3e^5f^3(2b - 2c - 5d + 3f + 2a)$

(8) 0

(9) $-2a^3b^2c^3d^3e^5f^3(-2b + 2c - 3d + 5f + 2a)$

(10) 0

PekerisPrefactors

```
> restart:
> #compute the Laplacian in spherical coordinates
> apply(psi, a, b, c):
> fr1(x1, x2, x3, x4, x5, x6) := sqrt(x1^2+x2^2+x3^2):
> fr2(x1, x2, x3, x4, x5, x6) := sqrt(x4^2+x5^2+x6^2):
> fr12(x1, x2, x3, x4, x5, x6) := sqrt(
  (x4-x1)^2+(x5-x2)^2+(x6-x3)^2):
> with(LinearAlgebra): with(VectorCalculus):
> eq := simplify(Laplacian(psi(fr1(x1, x2, x3, x4, x5, x6),
  fr2(x1, x2, x3, x4, x5, x6), fr12(x1, x2, x3, x4, x5, x6)),
  [x1, x2, x3, x4, x5, x6])+(2*(E+Z/r1+Z/r2-1/r12))
  *psi(fr1(x1, x2, x3, x4, x5, x6), fr2(x1, x2, x3, x4, x5, x6),
  fr12(x1, x2, x3, x4, x5, x6)), {x1^2+x2^2+x3^2 = r1^2, x4^2
  +x5^2+x6^2 = r2^2, (x4-x1)^2+(x5-x2)^2+(x6-x3)^2 = r12^2}):
> eqsimple := expand(sort(collect(simplify(eq,
  assume = positive), D))):
> #perimetric coordinates
> #a,b,c can be substituted by positive numbers
> a1 := a:
> a2 := b:
> a3 := c:
> x1(r1, r2, r12) := a1*(r2+r12-r1):
> x2(r1, r2, r12) := a2*(r1+r12-r2):
> x3(r1, r2, r12) := a3*(r1+r2-r12):
> phi(a, b, c) := exp(-(1/2)*a-(1/2)*b-(1/2)*c)*L(a)*L(b)*L(c):
> #substituted the Ansatz into the equation
> #this is done using string manipulation
> eqStr := convert(eqsimple, string):
> with(StringTools):
```

```
> for i to 3 do
  if i < 3 then hi := i
  else hi := 12
  end if:
  eqStr := SubstituteAll(eqStr, Join(["D[", convert(i,
  string), "](psi)(r1,r2,r12)"], ""), Join(["diff(phi
  (x1(r1,r2,r12),x2(r1,r2,r12),x3(r1,r2,r12)),r",
  convert(hi, string), ")"], "")):
  for j to 3 do
  if j < 3 then hj := j
  else hj := 12
  end if:
  eqStr := SubstituteAll(eqStr, Join(["D[", convert(i,
  string),",", convert(j, string), "](psi)(r1,r2,r12)"],
  ""),Join(["diff(phi(x1(r1,r2,r12),x2(r1,r2,r12),
  x3(r1,r2,r12)),r", convert(hi, string), ",r",
  convert(hj, string), ")"], ""))
  end do
  end do:
> eqStr := SubstituteAll(eqStr, "psi(r1,r2,r12)", "phi(
  x1(r1,r2,r12),x2(r1,r2,r12),x3(r1,r2,r12))"):
> eqparsed := parse(eqStr):
> #inverse coordinates to simplify
> r1sim := (a3*x2+a2*x3)/(2*a2*a3):
> r2sim := (a3*x1+a1*x3)/(2*a1*a3):
> r12sim := (a2*x1+a1*x2)/(2*a1*a2):
> eqpekeris := subs(r1 = r1sim, r2 = r2sim,
  r12 = r12sim, eqparsed):
> eqpekeris := simplify(eqpekeris
  *exp(1/2*(x1+x2+x3))):
```

```
> eqpekeris := numer(eqpekeris):
> #compare the prefactors
> x1 := u: x2 := v: x3 := w:
> coeffDLu := coeff(eqpekeris, (D(L))(u)):
> coeffDLu := simplify(coeffDLu, {u = 0, L(v) = 1, L(w) = 1}):
> coeffDDLu := coeff(eqpekeris, ((D@@2)(L))(u)):
> coeffDDLu := simplify(coeffDDLu, {u^2 = 0, L(v) = 1, L(w) = 1}):
> coeffDLv := coeff(eqpekeris, (D(L))(v)):
> coeffDLv := simplify(coeffDLv, {v = 0, L(u) = 1, L(w) = 1}):
> coeffDDLv := coeff(eqpekeris, ((D@@2)(L))(v)):
> coeffDDLv := simplify(coeffDDLv, {v^2 = 0, L(u) = 1, L(w) = 1}):
> coeffDLw := coeff(eqpekeris, (D(L))(w)):
> coeffDLw := simplify(coeffDLw, {w = 0, L(u) = 1, L(v) = 1}):
> coeffDDLw := coeff(eqpekeris, ((D@@2)(L))(w)):
> coeffDDLw := simplify(coeffDDLw, {w^2 = 0, L(u) = 1, L(v) = 1}):
> verify(coeffDLu*u, coeffDDLu);
> verify(coeffDLv*v, coeffDDLv);
> verify(coeffDLw*w, coeffDDLw);
```

The output of this worksheet is

(1) *true*

(2) *true*

(3) *true*