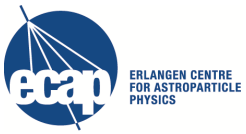
Program for the combinatoric assignment of LS- and jj-coupling term symbols

Bachelorarbeit aus der Physik

Vorgelegt von
Alexander Laska
am 23.10.2012



Remeis-Sternwarte und ECAP
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Prof. Dr. Jörn Wilms

Abstract

There are two important limiting cases of angular momentum coupling that are very important for atomic spectra: LS- and jj-coupling and there are several atomic structure calculating codes like FAC, HULLAC, AUTOSTRUCTURE. I wrote a program that produces all LS- and jj-coupling term symbols, sorts them by Hund's rules as good as possible and so delivers a tool for the comparison of those codes amongst each other and thus for an enhanced comparison of spectroscopic data with numerical results that are of theoretical nature. The program keeps track of the parental history of a term symbol, so that there are no multiple occurrences of a term symbol. The parental histories of the LS-coupling term symbols are chosen to be in agreement with the data of Palmeri et al. (2008) and those of the jj-coupling term symbols are chosen to be in agreement with the energy structure output of FAC.

Acknowledgements

I would like to thank all Prof. Dr. Jörn Wilms and Natalie Hell for their valuable help in writing this thesis and for assisting me with my plans to work at the LLNL in continuation to the work on this thesis in Erlangen and Bamberg.
Additionally, I want to thank my loving parents Annerose and Wolfgang Laska for their love, advice, assistance and finally financial support.

Contents

1. Introduction: The need of mapping LS- and jj-coupling term symbols

An important task in astrophysics is the interpretation of spectra of celestial objects. Spectra are a treasure of information about elemental abundances, processes in the stellar interior and exterior and indirectly about nearly every aspect of the universe on large or small scales. In order to take advantage of this treasure one has to know what the spectrum is expected to look like. The theoretical approach of reproducing a spectrum is dominated by one fact: an atom is a many-body system.
The problem now is that even the three-body problem is not solvable analytically and even solving the non-relativistic Schrödinger equation numerically is difficult. For this reason the appropriate handling of the problem using the relativistic Dirac equation (Dirac, 1928) for a many-body problem in order to yield the energy levels is a very difficult task. But luckily there are many relativistic corrections possible such as spin-orbit coupling or the treatment of the jitter movement as a small perturbation to the Schrödinger equation and therefore one could obtain qualitative "term symbols" describing the energy levels (Reinhold, 2006). Transitions between those levels lead to absorption and emission lines in the spectra (Fließbach, 1991). To generate labels giving a somehow unique description for those levels, the way the angular momenta and spins of the electrons couple is very important. The levels in angular momentum coupling are called term symbols and there are two important limiting cases for light and heavy atoms: LS- and jj-coupling (Demtröder, 2007).
This is not the whole story. For a proper description of a measured spectrum more ingredients are necessary: the broadening, probability amplitudes for the transition between the levels, modeling of the ionisation and (precise) quantitative (not just qualitative as delivered by the mere term symbol approach) information about the energy levels labeled by the term symbols (Bransden and Joachain, 2003). Therefore, the Dirac equation must be solved after all. This is the point when and the reason why several "atomic codes" were developed. Examples are HULLAC (Bar-Shalom et al., 2001), FAC (Gu, 2003, 2004) or AUTOSTRUCTURE (Badnell, 2011).
At this point the problem this thesis tries to solve rises. These codes produce and assign energies to term symbols in the LS-, jj- or some intermediate coupling scheme. But if one now wants to compare results of one code – mind the numerical character of the codes – with results of another code and with a measured spectrum, there is the problem of how to compare LS- and jj-coupling levels. It is not possible to pursue the direct assignment because the states in one scheme are linear combinations of states in the other (Condon and Shortley, 1970). But the idea for providing a simple mapping is to produce all terms in both coupling schemes, then sort them for energy by Hund's rules

and assign the xth row in one list to something close to the xth row in the other list. This is what is done by the program written as part of this thesis. It delivers a dictionary for the translation (in means of comparability, not of unique identification) of the LS-coupling term symbols to the jj-coupling term symbols language and hence delivers a small contribution to the whole process of making the best out of the treasure of spectra. Inter alia this goal was attempted because of a positive evidence for the possibility to assign the languages unambiguously found in a book of Gerhard Herzberg (1937). But the main reason is to be able to compare FAC results with AUTOSTRUCTURE in future examinations.

In the following sections the combinatoric theory behind the program, the usage, and the output of the program are discussed and presented assuming a basic knowledge of quantum mechanics. Nevertheless I tried to go through the important thoery behind angular momentum coupling and prepended a recapitulation of the basic ideas and the basic formalism used in physics to describe an atom.

2. Theoretical basics of LS- and jj-coupling

> "If ... only one sentence passed on to the next generations of creatures, what statement would contain the most information ...? I believe it is the *atomic hypothesis* ... that all things are made of atoms - little particles that move around in perpetual motion, attracting each other ..., but repelling upon being squeezed into one another."
>
> Richard Feynman

2.1. Basic quantum mechanics

Feynman's subsequent sentence to the sentences in the quote above was: "In that one sentence, you will see, there is an enormous amount of information about the world, if just a little imagination and thinking are applied..." (Feynman et al., 1963). This chapter has the goal to apply a little bit of that imagination and thinking to the *atomic hypothesis* and hence explain the necessary theory behind the combinatorial investigation of the presented work. Electrical charges were those "little" things repelling and attracting each other. But what are charges, how do they interact and form an emergent structure called *atom*? How is it realized that there are *differ*ent states? How do we label those states? What are those *differ*ences and how could they be analyzed quantitatively? The following approach tries to answer these questions *starting* (nearly) *from scratch*, but without getting lost (too much) in details. It is just a recapitulation of the relevant theory to get a more precise idea about the justification for and the formalism used in the subsequent work. The description starts more dilute and is meant to get more dense in proportion as it comes closer to the LS- and jj-coupling.

2.1.1. Schrödinger and Dirac equation

The basic tool to describe our physical world mathematically is a tensor *field* $\psi(x^\mu)$, where x^μ are the three space dimensions ($\mu = 1, 2, 3$) and the time dimension ($\mu = 0$).

A field is a multilinear and multidimensional array (Rebhan, 2010). Therefore, a *field* is the assignment of a scalar, a vector or in general a tensor *locally* to a point in spacetime, where $\mathscr{W}$ ($\ni x^\mu$) is the spacetime or the *world* set and $\mathscr{F}$ describes the properties of our physical reality as we see it - the *field* set (Peskin and Schroeder, 1995):

$$\psi : \mathscr{W} \rightarrow \mathscr{F}. \tag{2.1}$$

The spacetime assumed here is the four dimensional Minkowski spacetime ($\mathrm{sgn}(1,3)$). Hence the examination is *relativistic* and each resulting constitutive equation has to be Lorentz invariant. Their solutions have to be a representation of at least the Lorentz group $O(1,3)$ (the group of all isometries of the spacetime $\mathscr{W}$ and thus an *external* symmetry) and additionally of other symmetry groups - either those that are correlated to the *field* set or those that arise from the symmetries of the examined problem. Both symmetries are gauge symmetries acting on $\mathscr{F}$ and thus they are *inner* symmetries (Rebhan, 2010).
Due to the Noether theorem there is a constant of motion correlated to each symmetry. E.g., translational symmetry $\leftrightarrow$ four-vector momentum p^μ (energy E and momentum $\vec{p}$), rotational symmetry $\leftrightarrow$ angular momentum $\vec{L}$ (Altland and Simons, 2006).
There are discrete phenomena at a sufficiently small scale and we have to take account of them. Discrete values arise in mathematics as a spectrum of eigenvalues of an operator. So a natural and successful way to describe those discrete phenomena is to assign operators to the observable (field) we look at. This is done by expanding the expected solution as a sum of plane waves $e^{\vec{x}\cdot\vec{k}-\omega t}$, which leads to the following quantization rules[1] (Fließbach, 1991):

$$\begin{array}{rcrcl} E & \rightarrow & \hat{E} & = & -\hbar\,\partial_t \\ \vec{p} & \rightarrow & \hat{\vec{p}} & = & \hbar\,\vec{\nabla} \\ \vec{x} & \rightarrow & \hat{\vec{x}} & = & \vec{x} \\ \vec{L} & \rightarrow & \hat{\vec{L}} & = \hat{\vec{x}}\times\hat{\vec{p}} = & \hbar\,\vec{x}\times\vec{\nabla}, \end{array} \tag{2.2}$$

and the field itself:

$$\psi \rightarrow \hat{\psi}. \tag{2.3}$$

One gets an *operator field* $\hat{\psi}(x^\mu)$. The first assignment is called *first quantization*, while the quantization of the field itself is called *second quantization*. The hats that mark operator fields are suppressed in this thesis, because the effects that arise from the field also being an operator are not important for the subsequent theory. Nevertheless it is an important fact and should be mentioned for the sake of completeness.
To describe the dynamic and static behavior of the system, an equation of motion has to be derived. This is done by formulating a Lagrange density $\mathscr{L}(\psi, \partial\psi)$ and obtaining the so called Euler-Lagrange equations by extremalizing the action (Rebhan, 2010)

$$S = \int d^4x\ \mathscr{L}, \tag{2.4}$$

[1]The Planck constant $\hbar$ and the speed of light c are set to one and are therefore supressed in this thesis. No value in any unit system has to be calculated absolutely.

that is kept by the Hamiltonian principle

$$\delta S \stackrel{!}{=} 0, \tag{2.5}$$

and then evaluating the Euler-Lagrange equations[2] for each field:

$$\partial_\mu \frac{\partial \mathscr{L}}{\partial(\partial_\mu \psi)} - \frac{\partial \mathscr{L}}{\partial \psi} = 0. \tag{2.6}$$

Due to the need to be a representation of $O(1,3)$, the electron field has to be a *spinor*, a four component array with a special transform behavior. That implies a spin of $\frac{1}{2}$[3], and due to gauge invariance, where the gauge field is the electro-magnetic field A_μ, the easiest free Lagrange density (Feynman, 1986) for an electron field is

$$\mathscr{L} = \bar{\psi}(\not{D} - m)\psi - \frac{1}{4} F_{\mu\nu} F^{\mu\nu}, \tag{2.7}$$

where m is the mass of the fermion, $D_\mu = \partial_\mu + ieA_\mu$ the covariant derivation and $F_{\mu\nu} = -\frac{i}{e}[D_\mu, D_\nu]$ the field strength tensor (Rebhan, 2010), using Feynman's slash notation (Feynman, 1986) $\not{a} = a_\mu \gamma^\mu$ with γ^μ being the Dirac matrices[4].
The Euler-Lagrange equation with respect to the field ψ is the same as the Dirac equation with an additional term on the right side describing the electrodynamic effects, of which the description is based upon the field strength tensor $F_{\mu\nu}$,

$$(i\gamma^\mu \partial_\mu - m)\psi = e\gamma_\mu A^\mu \psi \tag{2.8}$$

and the Euler-Lagrange equations resulting out of the variations with respect to the gauge fields A_μ are

$$\partial_\mu F^{\mu\nu} = e\bar{\psi}\gamma^\nu \psi. \tag{2.9}$$

Those four equations are basically Maxwell's equation involving the field ψ in the minimal way that still facilitates Lorentz-invariance of the whole theory. This set of equations is also referred to as the basic equations of quantum electro dynamics (QED) (Feynman, 1986).
The next step would be to formulate a set of *many-body* Maxwell-Dirac equations for atoms that are such many-body systems. But because even the simplest coupled three-body problem cannot be solved analytically (Bransden and Joachain, 2003), except for some physically meager special solutions, a clever method is necessary to be able to deal with a fruitful theoretical description of the atom. Nevertheless these equations are very important and those systems of relativistically correct equations of motions are solved by some atomic codes such as FAC (Gu, 2003, 2004).
The idea to get more insights is now to separate a non-relativistic equivalent of the Dirac equation, the Schrödinger equation, and add relativistic and quantum field theory (QFT)

[2]The Euler-Lagrange equations are the equations of motion for every field that is varied via 2.5.
[3]So particles described by spinors are fermions in contrast to bosons that have an integer spin (Rebhan, 2010).
[4]The Dirac matrices are the tensor representation of a Clifford algebra (Jagannathan, 2010).

effects as small perturbations (Bransden and Joachain, 2003; Fließbach, 1991). The Schrödinger equation was called "a non-relativistic equivalent" to the Dirac equation, because it is not just the limiting case obtained by the limit speed of light $c \to \infty$. The Dirac equation acts on algebraically distinguished objects, the spinors, and the field ψ in the Schrödinger equation is only a scalar Rebhan (2010).
Another way to get the Dirac equation is to quantize[5] the relativistic dispersion relation

$$E = \sqrt{p^2 + m_0^2}, \tag{2.10}$$

where $p^2 = \sum_{i=1}^{3} p_{x^i}^2$ is the square of the total momentum and m_0 is the rest mass (Rebhan, 2010). The square root in the expression leads to the algebraic structure of a Clifford Algebra and therefore to the Dirac matrices and the spinors instead of simple scalar field functions.
Instead now the non-relativistic counterpart

$$E = \frac{p^2}{2m} \tag{2.11}$$

can be quantized easily yielding the free Schrödinger equation (Fließbach, 1991)

$$\hat{E}\psi = i\partial_t \psi = \frac{\hat{p}^2}{2m} = -\frac{1}{2m}\vec{\nabla}^2\psi. \tag{2.12}$$

The operator that has the energy as its eigenvalue is commonly denoted by $\hat{H}$ and is called *Hamilton operator*. Adding the potential energy operator $\hat{V}$ on the right side[6] yields the time-dependent Schrödinger equation (Fließbach, 1991):

$$\hat{H}\psi = i\partial_t \psi = \left(-\frac{1}{2m}\vec{\nabla}^2 + \hat{V}\right)\psi. \tag{2.13}$$

This is the non-relativistic equation of motion for a particle with mass m in a potential V. The basic interpretation of the Schrödinger equation is that the solutions for the field ψ_i, which are eigenvalues in a vector space $\mathcal{H}$, called Hilbert space, describe the probability $\rho(x^\mu) = |\psi|^2$, that the system is in that state. Those functions have complex values and form an orthonormal base taking the normalization

$$\int d^4x \rho \stackrel{!}{=} 1, \tag{2.14}$$

that is common for the interpretation as a probability distribution (Feynman et al., 1963). The Hilbert space $\mathcal{H}$ possesses a dual space $\mathcal{H}^*$. Vectors in this space can act on vectors in the original $\mathcal{H}$ by multiplication and integration over the whole $\mathscr{W}$. Vectors

[5]That means to use the quantization rules.
[6]Here are the contributions *to* the energy.

in $\mathcal{H}$ are called *bra* and vectors in $\mathcal{H}^*$ *ket*[7] (Rae, 2002). The former are expressed by $\langle\psi|$ and the latter by $|\psi\rangle$. So the scalar product is written in the form

$$\langle\psi_i|\ \psi_j\rangle \equiv \psi_i \cdot \psi_j = \int_{\mathbb{W}} d^4x\ \psi_i^* \psi_j. \tag{2.15}$$

The indices i and j express different solutions, hence different states. Their orthonormality is now easily expressible as

$$\langle\psi_i|\ \psi_j\rangle = \delta_{ij}. \tag{2.16}$$

The static Schrödinger equation is the equation of "motion" that describes stationary states[8]. It has the simple form

$$E_n \psi_{n\alpha} = \hat{H} \psi_{n\alpha}, \tag{2.17}$$

where α denotes the possibility of degeneracy: different energies are possible (n), but as well the same energy eigenvalue E_n is possible for more states. So these states form a degenerated subspace of the whole Hilbert space. Those different states belonging to the same E_n are numbered by α.
An operators $\hat{O}$ belonging to an observable has to be self adjoint (hermitian) to ensure reality of their eigenvalues, formal:

$$\hat{O} = \hat{O}^\dagger. \tag{2.18}$$

The matrix elements of an operator are

$$O_{ij} = \langle\psi_i|\, O\, |\psi_j\rangle\,. \tag{2.19}$$

As in every vector space the dyadic of two vectors ($|\psi_i\rangle$, $|\psi_j\rangle$) is an operator P_{ij} projecting into the subspace that is spanned by these vectors:

$$P_{ij} = |\psi_i\rangle \otimes \langle\psi_j| = |\psi_i\rangle\, \langle\psi_j| \tag{2.20}$$

and summing over the projection into all subspaces spanned by the single state vectors has to yield the identity

$$\mathbb{1} = \sum_i |\psi_i\rangle\, \langle\psi_i|\,, \tag{2.21}$$

in the case of a discrete base, or

$$\mathbb{1} = \int |\psi\rangle\, \langle\psi|\,, \tag{2.22}$$

in the case of a continuous base (Fließbach, 1991).

[7] This nomenclature is called Dirac's bracket (bra-ket) notation (Fließbach, 1991).

[8] This is a little bit confusing, but in stationary states ψ is still time dependent. Just $\hat{H}$ is meant to be time independent and with $\bar{H}$ also its eigenvalues E_n, where n numbers different stationary solutions of the time independent Schrödinger equation. ψ can be multiplied by a time dependent phase. But the physically important observable ρ is time independent as well.

The next step is to describe many bodies simultaneously using a many-body wave function Ψ. For the Fermionic electrons the antisymmetry (with respect to the exchange of single electron states) has to be ensured, leading to the so called Slater determinant (Bransden and Joachain, 2003)

$$\Psi(x_1^\mu, \ldots, x_N^\mu) = \begin{vmatrix} \psi_1(x_1^\mu) & \psi_2(x_1^\mu) & \ldots & \psi_N(x_1^\mu) \\ \psi_1(x_2^\mu) & \psi_2(x_2^\mu) & \ldots & \psi_N(x_2^\mu) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(x_N^\mu) & \psi_2(x_N^\mu) & \ldots & \psi_N(x_N^\mu) \end{vmatrix}, \tag{2.23}$$

where the products between the single particle states is the tensor product $\otimes$ and the lower right index now denotes not different single particle eigenfunctions as before but different particles' coordinates and wave functions. The antisymmetry

$$\Psi(\psi_1, \ldots, \psi_i, \ldots, \psi_j, \ldots, \psi_N) = -\Psi(\psi_1, \ldots, \psi_j, \ldots, \psi_i, \ldots, \psi_N) \tag{2.24}$$

ensures the Pauli exclusion principle

$$\Psi(\psi_1, \ldots, \psi_i, \ldots, \psi_i, \ldots, \psi_N) = -\Psi(\psi_1, \ldots, \psi_i, \ldots, \psi_i, \ldots, \psi_N) = 0. \tag{2.25}$$

It says that a many-body wave function must vanish if there are two particles in the same state; or short in the case of electrons: there cannot be two electrons with the same quantum numbers (Feynman et al., 1963).
The last very important relation that has to be introduced is the Heisenberg uncertainty principle. If the mean quadratic deviation of the observable that is linked to an operator $\hat{O}$ is defined as

$$\left(\Delta\hat{O}\right)^2 = \left\langle \left(\hat{O} - \left\langle \hat{O} \right\rangle\right)^2 \right\rangle, \tag{2.26}$$

using the common notation for an operator's expectation value $\left\langle \hat{O} \right\rangle$,

$$\left\langle \hat{O} \right\rangle \equiv \langle\psi| \hat{O} |\psi\rangle \tag{2.27}$$

the following relation emerges by straight forward calculation:

$$\left(\Delta\hat{O}_1\right)^2 \left(\Delta\hat{O}_2\right)^2 \geq \frac{\left\langle i[\hat{O}_1, \hat{O}_2] \right\rangle}{4}, \tag{2.28}$$

where $[A, B] \equiv AB - BA$ is the commutator. This is the *uncertainty principle* (Fließbach, 1991). It is not a direct consequence of the Schrödinger equation, rather it is related to the algebraic properties of operators and their ability not to commute[9].

[9] "Uncertainty" is also a property that is deeply linked to the fact that ψ behaves like a damped wave and the natural link between the frequency components of a function delivered by a Fourier transform and the untransformed function (Ashcroft and Mermin, 1976).

After the many-particle wave function is constructed, a whole system Hamiltonian $\hat{H}_{\text{tot}}$ has to be written down[10], probably involving relativistic effects as perturbative extra terms. Finally the Schrödinger equation of the whole system has to be solved.
But before the complex many-body problem describing arbitrary atoms is encountered I will perform a closer look at the analytically solvable one-body problem. This one-body problem of an electron and a nucleus is the hydrogen atom described in the next section[11].

2.1.2. H-Atom

The hydrogen atom in its neutral state consists of a nucleus (proton and possibly neutrons) and an electron. The nucleus of all atoms can be seen as very massive[12]. The description takes place in the rest frame of the nucleus. Then the Hamiltonian of the electron in the electro-static potential is

$$\hat{H} = \hat{H}_{\text{kin}} + \hat{H}_{\text{pot}} = -\frac{1}{2m_e}\vec{\nabla}^2 - e^2\frac{1}{r}, \tag{2.29}$$

where r is the distance in the three space dimensions, leading to the static Schrödinger equation

$$E_n\psi_\alpha(x^\mu) = \left(-\frac{1}{2m_\mu}\vec{\nabla}^2 - e^2\frac{1}{r}\right)\psi_\alpha\left(x^\mu\right). \tag{2.30}$$

m_μ is the reduced mass that arises from the change in the center of mass rest frame that is nearly the rest frame of the proton and so $m_\mu \approx m_{\text{e}}$ the electron mass. To solve the equation the wave function is separated into a radial part $R_{n\alpha}(r)$ (only dependent on r) and an angle dependent part $Y_{n\beta}(\theta,\phi)$, where θ and ϕ are the polar and azimuthal angle of spherical coordinates (Bransden and Joachain, 2003). Additionally, the whole equation is expressed in spherical coordinates involving the angular momentum operator that is hidden in the Laplace operator $\Delta = \vec{\nabla}^2$ expressed in spherical coordinates (Fließbach, 1991):

$$E_nR_{n\alpha}(r)Y_{n\beta}(\theta,\phi) = \left(-\frac{1}{2\mu}\frac{1}{r^2}\frac{\partial}{\partial r}r^2\frac{\partial R_{n\alpha}(r)}{\partial r} - \hat{L}^2R_{n\alpha}(r) - \frac{e^2}{r}R_{n\alpha}(r)\right)Y_{n\beta}(\theta,\phi) \tag{2.31}$$

The angular momentum operator has some very important eigenvalues and eigenfunctions. The eigenvalues of its square $\hat{L}^2$ are

$$\hat{L}^2Y_{lm_l} = l(l+1)Y_{lm_l} \tag{2.32}$$

[10] In the following the "tot" in the index of the total system Hamiltonian $\hat{H}_{\text{tot}}$ is skipped, because always the total system is examined; even in the case of just the one electron in hydrogen: if there is only one, only one is the total system.

[11] It is conceptionally a two-body problem, but it can be translated into a one-body problem.

[12] This fact becomes very important for ions with more than one electron, because due to this relatively high mass in comparison to the electron mass the motion of the core can be neglected and this is a big step in the process of reducing the complexity of the problem. In the case of the hydrogen atom this assumption is not necessary.

and the eigenvalues of its z-component

$$\hat{L}_z Y_{lm_l} = m_l Y_{lm_l}, \tag{2.33}$$

where l is limited to integer values

$$l = 0, 1, 2, 3, \ldots \tag{2.34}$$

and the associated m_l to

$$m_l = 0, \pm 1, \pm 2, \pm 3, \ldots, \pm l. \tag{2.35}$$

Y_{lm_l}, the so called *spherical harmonics* (Semendjajew, 2008), are the common eigenfunctions of $\hat{L}^2$ and its z-component $\hat{L}_z$:

$$Y_{lm_z}(\theta, \phi) = \sqrt{\frac{2l+1}{4\pi}} \sqrt{\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi}, \tag{2.36}$$

including the associated Legendre polynomials

$$P_l^m = (-1)^m (1-x^2)^{\frac{m}{2}} \frac{d^m}{dx^m} \left(P_l(x) \right), \tag{2.37}$$

themselves again including the "normal" Legendre polynomials

$$P_l(x) = \frac{1}{2^l l!} \frac{d^l}{dx^l} \left[(x^2 - 1)^l \right]. \tag{2.38}$$

The last equation is called Rodrigues' formula (Strang et al., 2010).
That those two operators share the same eigenfunctions but different eigenvalues is due to the fact that the square of the angular momentum operator $\hat{L}^2$ and its z-component commute (Fließbach, 1991):

$$\left[\hat{L}^2, \hat{L}_z \right] = 0, \tag{2.39}$$

while the components among each other do not commute:

$$\left[\hat{L}_i, \hat{L}_j \right] = i \epsilon_{ijk} \hat{L}_k, \tag{2.40}$$

where ϵ_{ijk} is the total anti symmetric pseudo tensor (epsilon tensor). This implies that due to the uncertainty principle 2.28 the square of the angular momentum operator $\hat{L}^2$ can be measured exactly simultaneously with its z-component $\hat{L}_z$ in spite of its x- and y-component. This smearing in the xy-plane and fixing of the length ($\sqrt{\hat{L}^2}$) and z-component forces the angular momentum vector to lie on a cone (see picture 2.2).
Now the suggestive nomenclature $Y_{n\beta}$ in the H-atom static Schrödinger equation 2.31 becomes clear. The $Y_{n\beta}$ can be chosen as the spherical harmonics $Y_{n\beta}$ and so β is m_l. Evaluating their action on $\hat{L}^2$ in 2.31 yields

$$Y_{n\beta}(\theta, \phi) \left(E_n R_{n\alpha} - \left(-\frac{1}{2\mu} \frac{1}{r^2} \frac{\partial}{\partial r} r^2 \frac{\partial R_{n\alpha}(r)}{\partial r} - l(l+1) R_{n\alpha}(r) - \frac{e^2}{r} R_{n\alpha}(r) \right) \right) = 0, \tag{2.41}$$

where $Y_{l\beta}$ can be canceled out[13] This is leading to the equation for the radial part $R_{n\alpha}$ (Fließbach, 1991)

$$\frac{d^2u(r)}{dr^2} + 2m_\mu \left(E - V_{\text{eff}}\right) u(r) = 0, \tag{2.42}$$

having inserted the substitution $u(r) \equiv rR_{n\alpha}(r)$ and having combined the electric potential $-V(r) = \frac{e^2}{r}$ and $-\frac{l(l+1)}{2m_\mu r^2}$ to V_{eff}. Solving and resubstituting the solution yields

$$R_{n\alpha}(r) = R_{nl}(r) = \frac{2}{n^2 a_0^{\frac{2}{3}}} \sqrt{\frac{(n-l-1)!}{(n+l)!}} \left(\frac{2r}{na_0}\right)^l e^{-\frac{r}{na_0}} L_{n-l-1}^{2l+1}\left(\frac{2r}{na_0}\right), \tag{2.43}$$

this time involving the Laguerre polynomials $L_b^a(x)$ defined as

$$L_b^a(x) = \sum_{i=0}^{k} \frac{(a+b)!(-x)^i}{(b-i)!(a+i)!i!}. \tag{2.44}$$

The number $a_0 = a_0(ml) = \frac{4\pi}{m_\mu l^2} \approx 0.53$ Å is called *Bohr radius*. It is the characteristic length scale of the quantum mechanical Coulomb problem (Fließbach, 1991). It corresponds to the radius $r_{\min}$ of the minimum of the effective potential V_{eff}, where $r_{\min} = l(l+1)a_0$ (Reinhold, 2006).
n has to be an integer except zero:

$$n = 1, 2, 3, \ldots \tag{2.45}$$

and l is constrained to be smaller than n:

$$l = 0, 1, 2, \ldots, n-2, n-1, \tag{2.46}$$

where n is called *main quantum number*, l is called *angular momentum quantum number*, m_l is called *magnetic quantum number*.
Quantum numbers that commute with the Hamilton operator $\hat{H}$ of the system are called *good quantum numbers* because they are time independent (Fließbach, 1991). Since $\hat{L}^2$ commutes with $\hat{H}$,

$$\left[\hat{L}^2, \hat{H}\right] = 0, \tag{2.47}$$

it follows that the Hamilton operator $\hat{H}$ as well as $\hat{L}^2$ and $\hat{L}_z$ can be diagonalized simultaneously and they can have the same eigenfunctions Y_{lm_l}.
Concluding the space related part of the eigenfunction of the Hydrogen problem is

$$\psi_{nlm_l}(r, \theta, \phi) = R_{nl}(r) Y_{lm_l}(\theta, \rho), \tag{2.48}$$

[13]The separation approach is now justified, because the remaining equation for $R_{n\alpha}$ is free of any angular dependence. Canceling means that the product on the left side has to yield zero and because $Y_{nm_l} \neq 0$ the large expression in parentheses has to be equal to zero.

belonging to the eigenvalues of the Hamilton operator $\hat{H}$, hence the energies E_n:

$$E_n = \langle \psi_{nlm_l} | \, \hat{H} \, | \psi_{nlm_l} \rangle = -\frac{R_y}{n^2}, \tag{2.49}$$

where $R_y = \frac{m_\mu e^4}{2(4\pi)^2} \approx 13.6\,\text{eV}$ is the *Rydberg constant*.
Using the non-relativistic Schrödinger equation has led to a degeneracy into states with the same energy E_n, but many different angular momentum or magnetic quantum numbers l and m_l. The degeneracy belonging to the different values of l is lifted when solving the relativistic Dirac equation or if additional terms correcting the non-relativistic nature of the Schrödinger equation towards relativistic correctness are included (Bransden and Joachain, 2003). The degeneracy due to m_l is based on the spherical symmetry of the Hydrogen atom an therefore of the Hydrogen problem. This degeneracy can be lifted if an external magnetic field is applied, selecting a favorite direction and hence making the projection in the arbitrarily chosen z-direction important[14] (Feynman et al., 1963).
Another consequence of the usage of the non-relativistic Schrödinger equation is the neglecting of the spin, which is a relativistic phenomenon. It has to be introduced manually. One has to add

$$\hat{S} = \frac{1}{2} \begin{pmatrix} \sigma^1 \\ \sigma^2 \\ \sigma^3 \end{pmatrix}, \tag{2.50}$$

the spin operator, to the set of operators describing observables, where $\sigma^i (i = 1, 2, 3)$ are the Pauli matrices. $\hat{S}_i$ measures the spin in i-direction (now $i = x, y, z$). A very important behavior of the spin is, that the eigenvalues and eigenfunctions of and the spin operator $\hat{S}$ itself behaves like the angular momentum operator $\hat{L}$. Thus all the relations found to be true for $\hat{L}$ are also true for $\hat{S}$. They can be written as if there is a $\hat{S}$ instead of $\hat{L}$ and s instead of l. The spin wave function is again a vector in a vector space. This vector space is called $\mathcal{H}_s$ (Fließbach, 1991). States that are space and spin describing states are composed by multiplying the space wave function tensorially with the spin wave function ψ_{sm_s}:

$$|nlm_l sm_s\rangle \equiv |nlm_l m_s\rangle \equiv |\psi_{nlm_l m_s}\rangle \equiv \psi_{nlm_l m_s} \equiv \psi_{nlm} \otimes \psi_{sm_s} \equiv |\psi_{nlm}\rangle \otimes |\psi_{sm_s}\rangle \,. \tag{2.51}$$

The denotation of the *spin quantum number* s is unimportant, because all electrons have $s = \frac{1}{2}$. Since all electrons have the same s, states differ from other states in no case by means of s. Thus $s = \frac{1}{2}$ delivers no helpful information it is omitted when referring to the electron quantum numbers. The important quantum number here – to distinguish states – is $m_s = \pm\frac{1}{2}$, the *magnetic spin quantum number* and so this quantum number is given when describing a state of an electron. This issue is expressed by the first equivalence in 2.51.
The one-electron stationary states ψ_{nlm_l} are called *orbitals* and due to the analytically unseparability of the many-body problem they are an important tool in the approximation of the many-body problem (Reinhold, 2006).

[14] Most likely this caused the name *magnetic* quantum number m.

2.1.3. Multi-electron systems

Atoms that are the business of this thesis are many-body systems. The goal is to describe a system that consists of a positively charged core surrounded by negatively charged electrons. Neglecting spin, other relativistic and QFT effects, the kinetic energies, electromagnetic interactions of all electrons with the core have to be considered (first sum in $\hat{H}$ below). The mutual electromagnetic interactions of all electrons with each other, with their distances $r_{ij} = |r_i - r_j|$, also need to be taken into account (second sum). The full Hamiltonian becomes

$$\left[\sum_{i=1}^{N}\left(-\frac{1}{2}\vec{\nabla}^2_{x_i^\mu} - \frac{Ze}{r_i}\right) + \sum_{\substack{i,j=1\\ i<j}}^{N}\frac{e^2}{r_{ij}}\right]\Psi(x_1^\mu, \dots x_N^\mu) = E\Psi(x_1^\mu, \dots x_N^\mu), \tag{2.52}$$

where Ψ is the antisymmetric many-particle wave function constructed in 2.23.
In order to be able to perform perturbation theory, the Hamiltonian of a many-electron atom is partitioned in an unperturbed part containing the experience of a mean field potential V_{mean} instead of the complex coupling producing mutual coupling. This part reads $\hat{H}_{\text{mean}}$ and the second part contains the rest of the terms of the full Hamiltonian (Bransden and Joachain, 2003) $\hat{H}_{\text{rest}}$. Thus

$$\hat{H} = \hat{H}_{\text{mean}} + \hat{H}_{\text{rest}}, \tag{2.53}$$

where

$$\hat{H}_{\text{mean}} = \sum_{i=1}^{N}\hat{h}_i, \tag{2.54}$$

with the single electron Hamilton operator

$$\hat{h}_i = -\frac{1}{2}\vec{\nabla}^2_{x_i^\mu} + V_{\text{mean}}(x_i^\mu), \tag{2.55}$$

and

$$\hat{H}_{\text{rest}} = \sum_{\substack{i,j=1\\ i<j}}^{N}\frac{e^2}{r_{ij}} - \sum_{i=1}^{N}\left[\frac{Ze}{r_i} + V(r_i)\right], \tag{2.56}$$

is the Hamiltonian in the mean field approximation (Reinhold, 2006). The advantage of this splitting is that now the first part $\hat{H}_{\text{mean}}$ is solvable in the same way the hydrogen atom was solved in the last section and the second sum (and the term leading to angular momentum coupling in the next section) can be treated as perturbations. Whether they are small enough for this approach to be a successful description of the atom depends on the size of the atom and will determine which coupling scheme is the better choice (Bransden and Joachain, 2003).
Now i is the index of the electrons. The mean field related part

$$\hat{H}_{\text{mean}}\Psi_{\text{mean}} = E_{n,mean}\Psi_{\text{mean}} \tag{2.57}$$

can be solved using a separation ansatz

$$\Psi_{\text{mean}} = \psi_1 \psi_2 \dots \psi_N, \tag{2.58}$$

possessing the solutions

$$\psi_i = \psi_{n_i l_i m_{l_i}}. \tag{2.59}$$

This approach leads to the incorrect, since depending on the mean field approximation and the neglect of relativistic and QFT effect, picture that electrons in a many-electron system are having the same quantum numbers like the one electron in the case of hydrogen and that those electrons "fill" the possible orbitals that hydrogen offers his electron to be in[15]. Adding an electron to the atom changes the whole system and not just fills the structure that the atom has before the electron is added.
The term $\hat{H}_{\text{rest}}$ is neglected here, because the perturbation theoretical treatment leads to a correction that is unimportant. But one important difference in the case of the mean field potential V_{mean} is that the degeneracy due to l_i is omitted while the absence of the distinction of one direction still leaves the degeneracy with respect to m_{l_i} (Bransden and Joachain, 2003).

2.2. Angular momentum coupling

The last step in order to describe angular momentum coupling is turning on the so called *spin-orbit interactions*

$$\hat{H}_{\text{s-o}} = \sum_{i=1}^{N} \frac{1}{2m_\mu} \frac{1}{r_{ij}} \frac{dV_{\text{mean}}(r_i)}{dr_i} \hat{\vec{l}}_i \cdot \hat{\vec{s}}_i, \tag{2.60}$$

where $\hat{\vec{l}}_i$ is the angular momentum operator of the electron number i and $\hat{\vec{s}}_i$ the spin operator of the same electron.
The Hamiltonian including angular momentum therefore reads (Bransden and Joachain, 2003)

$$\hat{H} = \hat{H}_{\text{mean}} + \hat{H}_{\text{rest}} + \hat{H}_{\text{s-o}}. \tag{2.61}$$

A to a certain extent misleading comparison is the picture of the moon orbiting the earth while having synchronized its angular momentum and its own rotation to the rotation of the earth about its axis (Carroll and Ostlie, 2007). More precisely the term "coupling" refers to the fact that the angular momentum and spin quantum number of a single electron are not "good quantum numbers" (their assigned operators do not commute with the Hamilton operator of the whole system) while the joint and allowed (obeying the Pauli principle) vectorial sums to L, S, J (LS-coupling) or to j_i ($\forall i = 1, \dots, N$, where N is the number of electrons) and J are good quantum numbers (Landau and Lifshitz, 1987).

[15] Or as in the case of one particle in any other central potential. The arising of this hydrogen-like behavior is related to the spherical $O(3)$ symmetry (central potential) and not to the special case of a Coulomb potential (Fließbach, 1991).

The two limiting cases for coupling arise from the different length scales of distances between preferred regions of the electrons around the nucleus in differently sized atoms. This goes along with the relative size of the two perturbative terms. The spin-orbit coupling is weak in relation to global coupling if the distances are small, yielding LS-coupling and coming along with $\hat{H}_{\text{rest}} \gg \hat{H}_{\text{s-o}}$. In the case of a heavy system the distances are large; so the effect of the spin-orbit coupling is no more negligible in relation to electrostatic effects ($\hat{H}_{\text{s-o}} \gg \hat{H}_{\text{rest}}$) and the single electron wave function "sees" its own spin and angular momentum before it is "thinking to care" about the angular momentum states of the other electrons and therefore jj-coupling is the scheme of choice (Cowan, 1981; Bransden and Joachain, 2003; Friedrich, 1990; Condon and Shortley, 1970) (see figure 2.2).
If both spin-orbit effects and the residual electrostatic effects are important and no term can be left out of the examination *intermediate coupling* takes places. To handle this case is an extensive undertaking and in this thesis just the two extreme cases of LS- and jj-coupling will be examined (Bransden and Joachain, 2003).

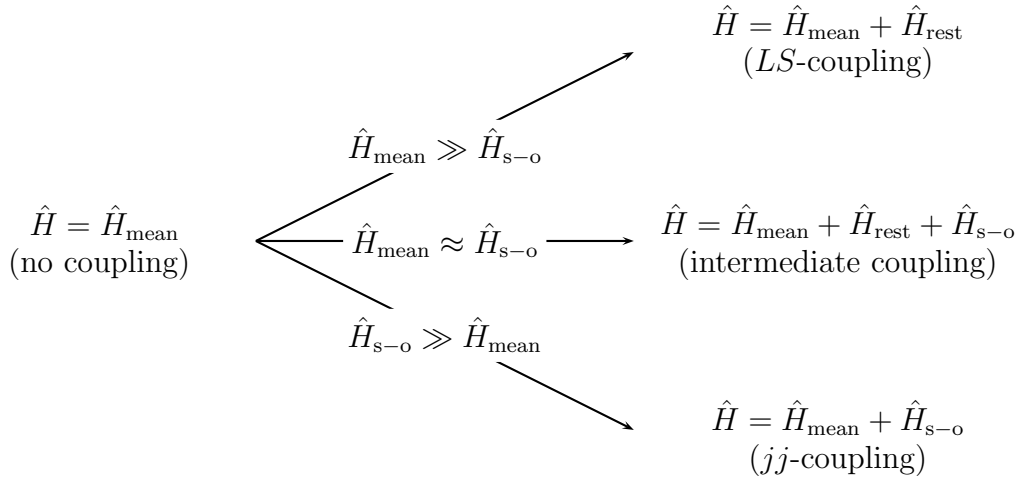


Figure 2.1.: For the three cases of the relative strength of the two terms $\hat{H}_{\text{rest}}$ and $\hat{H}_{\text{s-o}}$ different coupling schemes are valid.

2.2.1. LS-coupling

As mentioned above the LS- (or Russel-Saunders-) coupling is an appropriate way of describing small electronic configurations (atoms with $Z \leq 10$ (Haigh, 1995)). The reason for this is that the spin-orbit coupling is weak in comparison to the electrostatic effects. This means that in the central field approximation the ith electron's term in the Hamiltonian $\hat{H}$ proportional to $\vec{l}_i \cdot \vec{s}_i$ (the so called spin-orbit effects – or in this common order better: orbit-spin effects) is small enough to be negligible Bransden and Joachain (2003). In terms of perturbation theory this leads to first solving the static Schrödinger

equation

$$E_{n_i,mean}\psi_{i,\text{mean}} = \hat{H}_{\text{mean}}\psi_{\text{i,mean}}, \tag{2.62}$$

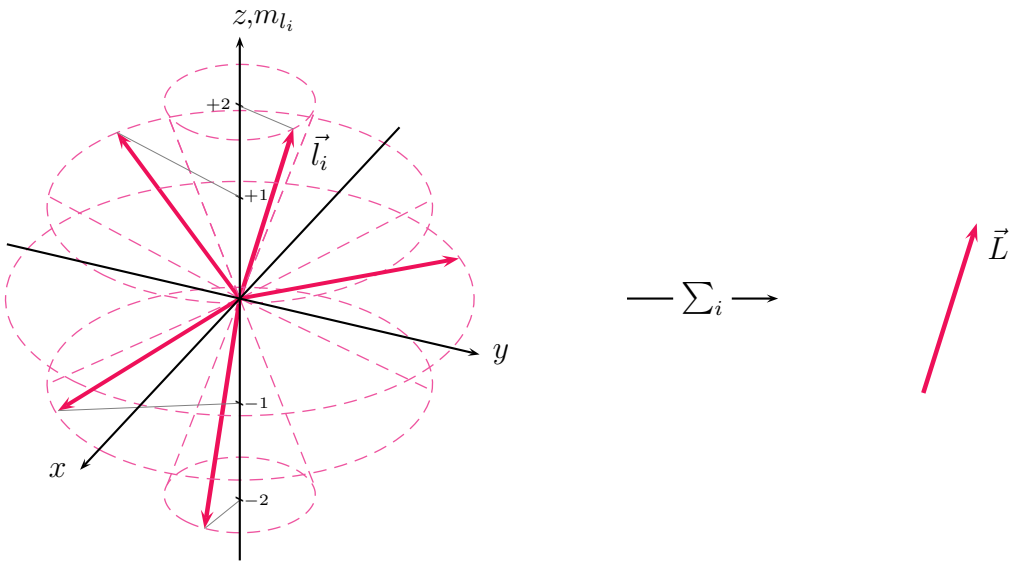
yielding unperturbed single particle wave functions $\psi_{n_i l_i m_{l_i} m_{s_i}}$. Their angular momenta couple to a common $\vec{L}$ and $\vec{S}$ and "then" a finer structure emerges from the coupling of $\vec{L}$ and $\vec{S}$ to a whole system total angular momentum $\vec{J}$. It is important that the "coupling" here can be expressed in the vectorial sums (Fließbach, 1991):

$$\vec{L} = \sum_{i=1}^{N} \vec{l}_i, \tag{2.63}$$

$$\vec{S} = \sum_{i=1}^{N} \vec{s}_i, \tag{2.64}$$

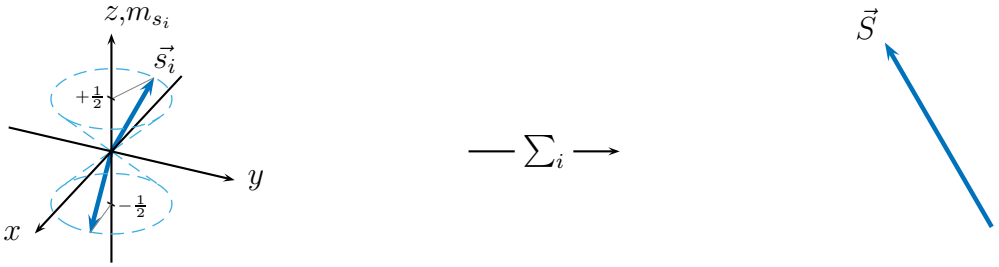
$$\vec{J} = \vec{L} + \vec{S}. \tag{2.65}$$

The impact of the *vectorial sum* together with the fact that only $\vec{L}^2$, $\vec{S}^2$, $\vec{J}^2$ and L_z, S_z, J_z are commutating with $\hat{H}$ (and hence just L^2, S^2, J^2, m_L, m_S and m_J are good quantum numbers) is hardly to be overrated.
These facts are the reason for the combinatorics that necessitate the program described in this thesis. Mind that the conservation of the length and the projection to one axis of the vectors are leading to circles of points (intersection points of sphere and a plane). Thus the set of possible vectors having the same length and z-axis projection lie on a cone as mentioned above (Reinhold, 2006). There are cones with heights in z-direction from $m_L = -L, -L+1, \ldots$ up to $m_L = \ldots, +L-1, +L$ possible for a single value of L^2 (Landau and Lifshitz, 1987). This is also true for S^2 and J^2; but more importantly (for LS-coupling) for all the l_i^2 and s_i^2 (See Figure 2.2, 2.3 and 2.4).
The smearing in x- and y-direction is referred to as Heisenberg's uncertainty principle 2.28. First all $\vec{l}_i$ and all $\vec{s}_i$ add up vectorially being adjusted to a possible value of m_{l_i} and m_{s_i} to $\vec{L}$ and $\vec{S}$. This leads to a lot of permutations for the single electron quantum numbers ($|n_1, l_1, m_{l_1}, m_{s_1}\rangle \otimes \cdots \otimes |n_N, l_N, m_{l_N}, m_{s_N}\rangle$) and because of those to a set of collective quantum numbers ($|L, S, J, m_J\rangle$). Due to the Pauli exclusion principle only some of those states are possible, because some L or S cannot be reached without having electrons in the same states which is forbidden.
LS-coupling is based on the assumption that the contribution of a single term such as $\hat{\vec{l}}_i \cdot \hat{\vec{s}}_i$ is small, but after adding up to $\vec{L}$ and $\vec{S}$ those whole system spin and orbit related values "talk" to each other and now contribute to $\hat{H}$. To obtain the energy contribution the bra and ket vector of an LS-coupling state have to act on the term in the Hamilton operator of the system that is proportional to $\hat{\vec{L}} \cdot \hat{\vec{S}}$ (Fließbach, 1991). This approach is necessary in order to get the first order energy correction from the term $\hat{H}_{\text{s-o}}$ to obtain a finer structure depending on J using perturbation theory. That a term proportional to $\hat{\vec{L}} \cdot \hat{\vec{S}}$ can be diagonalized yielding the same subspace as $\hat{H}_{\text{s-o}}$ can be shown by the



Single electron's $\vec{l}_i$ add up vectorially to $\vec{L}$.

Figure 2.2.: Illustration of a single electrons angular momentum vector $\vec{l}_i$ for the case of $l_i = 2$. This enables the vector to have a m_{l_i} from -2 up to $+2$ in steps of the size 1. For each of those adjustments a representation is drawn in the graphic. All those vectors are on cones due to the uncertainty principle and their length is equal to $\sqrt{l_i(l_i+1)}$. All physically possible (Pauli exclusion principle) orientated groups of single electron angular momenta for the whole atom "then" vectorially add up to the total system angular momentum $\vec{L}$ (see equation 2.63).



Single electron's $\vec{s}_i$ add up vectorially to $\vec{S}_i$.

Figure 2.3.: There are two possible adjustments for the spin vector of each electron: Up $\uparrow$ and down $\downarrow$ coming along with the two possible values $\pm\frac{1}{2}$ for m_{s_i}. In those cases of all combinations of all single electron spins they add up vectorially to a whole atom total spin vector $\vec{S}$, in which the whole system obeys the Pauli principle (see equation 2.64).

Wigner-Eckart theorem (Cornwell, 1997). So we have

$$\begin{aligned}\langle E_{LS}\rangle &= \langle L,S,J,m_J|\,\hat{H}_{LS}\,|L,S,J,m_J\rangle \propto \langle \Psi_{LSJm_J}|\,\hat{\vec{L}}.\hat{\vec{S}}\,|\Psi_{LSJm_J}\rangle \\ &= \frac{1}{2}(\langle \Psi_{LSJm_J}|\,\hat{\vec{J}}^2 - \hat{\vec{L}}^2 - \hat{\vec{S}}^2\,|\Psi_{LSJm_J}\rangle) \\ &= \frac{1}{2}(\langle \Psi_{LSJm_J}|\,(J(J+1) - L(L+1) - S(S+1))\,|\Psi_{LSJm_J}\rangle) \\ &= \frac{1}{2}(J(J+1) - L(L+1) - S(S+1)),\end{aligned}$$

assuming $|\Psi_{LSJm_J}\rangle = |L,S,J,m_J\rangle$ to be an orthonormal eigenbase of the system Hamiltonian $\hat{H}$ and hence of $\hat{H}_{\mathrm{s-o}}$ (Landau and Lifshitz, 1987).
For a given set of L and S there are several values possible for J if the anti-symmetry of the many-body system is guaranteed: J ranges from $|L-S|$ in steps of size 1 up to $L+S$. This is known as the triangular condition (Reinhold, 2006)

$$J = |L-S|, |L-S|+1, \ldots, L+S-1, L+S. \tag{2.66}$$

An example for the practical understanding of this important relation is given in the next section because it is a result of the mechanism how angular momenta add and this mechanism governs everything while determining the level structure of neutral atoms and as well as ions. But a formal proof is possible by counting the degeneracy of linear subspaces of $\hat{H}$ belonging to values of m_J and then deducing the possible values of J (Wormer, 2012). Informally this is the same as one sometimes refers to as producing all m_J and afterwards crossing out all values from the current maximal $(m_J)_{max}$ in steps of size one to $-(m_J)_{max}$ until nothing is left in a general version.
The resulting physically possible states (characterized by S, L, J, m_J) are expressed by a common nomenclature that is called the *term symbols*:

$${}^{2S+1}L_J^{(\circ)}. \tag{2.67}$$

The "$\circ$" in the top right corner is only used of a state that has o̲dd parity, P, where

$$P = (-1)^{\sum_{i=1}^{N} l_i}. \tag{2.68}$$

A term symbol expresses a *level* (L, S, J), the whole set of quantum numbers including m_J is called a *state* and the term symbol without giving J is called a *term* (Haigh, 1995). Conventionally in this notation L is not described by numbers expressing the angular momentum operator eigenvalues. Instead letters are used (see table 2.1). This is called "spectroscopic notation" due to the relation between line characteristics ("sharp", "principal", "diffuse" and "fundamental" or "fine") and the theoretically revealed eigenvalues of the angular momentum operator that were discovered later (Bransden and Joachain, 2003). The splitting into $2S+1$ different m_S values leads to the denotation of this "multiplicity" in the top left corner. The value of m_J is not expressed in this notation. Nevertheless a state in LS-coupling is characterized by this value and there

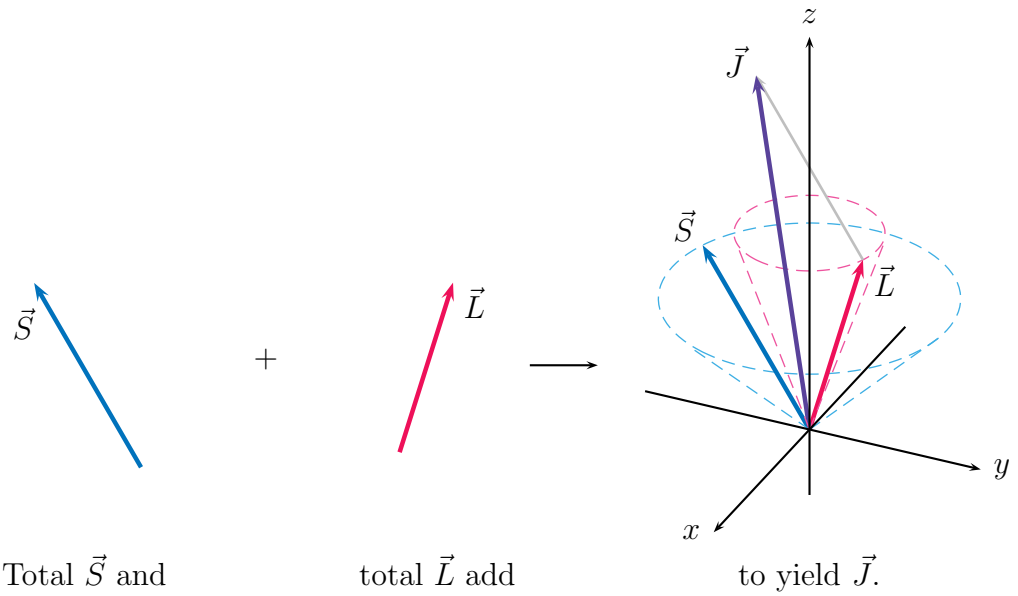


Figure 2.4.: The total Spin $\vec{S}$ and the whole system angular momentum add up vectorially to $\vec{J}$, the whole system total angular momentum. Again there are $2S+1$ possible m_S values and $2L+1$ values for m_L. Being adjusted to a combination of them the $\vec{L}$ and $\vec{S}$ add to yield a set of possible m_J values and therefore to a set of $\vec{J}$ (equation 2.65) ranging from $|L-S|$ to $L+S$ in steps of size one (see triangular condition 2.66).

Table 2.1.: The common assignment of letters instead of numbers to values of angular momenta L in term symbols and l_i in orbital configurations.

Value of angular momentum (L, l) in [$\hbar$]:	0	1	2	3	4	5	6	7	...
Letter assigned to L in term symbols:	S	P	D	F	G	H	I	K	...
Letter assigned to l_i in orbital configurations:	s	p	d	f	g	h	i	k	...

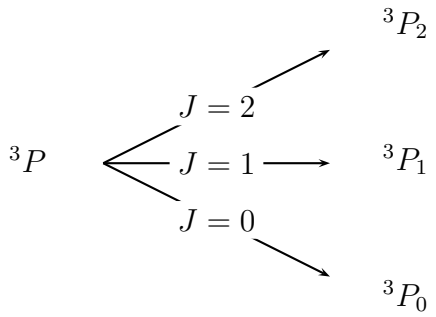


Figure 2.5.: Splitting of a term into a multiplet. Each of those *levels* in the multiplet is expressed by a term symbol.

is a degeneracy of $2J + 1$ states per value of J. They split into these values in the case of a measurement that emphasizes a special direction so that the projection of J that is m_J becomes important. Applying a magnetic field in some special direction during the transition can experimentally reveal this splitting property. The notation also does not stress the fact that there are $2L + 1$ possible values for m_L which leads to a degeneracy of $(2L + 1)(2S + 1)$ per set pair of values of L and S. But the multiplicity is still a wise choice because it counts how many values of J are at most possible (without taking the Pauli exclusion principle into account) to build a multiplet (see triangular condition 2.66 and figure 2.2.1).
For the orbital configuration the common notation is used:

$$n_1 l_1^{x_1} n_2 l^{x_2} \dots n_m l_m^{x_m}, \tag{2.69}$$

where n_i denotes the principal quantum numbers, l_i the angular momentum quantum numbers and x_m the number of electrons in the subshell defined by both n_i and l_i. The total number of electrons is implicitly given by:

$$\sum_{i=1}^{m} x_i = N \tag{2.70}$$

The l_i are expressed by the same letters as L, but written in lower case (see table 2.1). Sometimes in the physics literature the orbital (or electronic) configuration is given in a more explicit form that uses parentheses but means the same. For example, one uses $(3d)^2$ instead of $3d^2$.
The notation introduced above (equation 2.67) is simple and a good choice for a quick general insight, but it is not unique since the same level can result from different linear combinations of subsets of quantum numbers in the same electron configuration. A more precise version can be produced due to the distinguishability of terms having the same L, S, J by taking the "parental history" (Condon and Shortley, 1970; Condon and Odabasi, 1980) of the final term (and therefore level) into account. This history is

the way the successive coupling of shells takes place. One can think of the x_k electrons in a shell with n_k and l_k. They couple (LS-coupling) to a term ${}^{2S_k+1}L_k$ and this term couples with the joint term of all shell with an index $a < k$. "Then" the x_{k+1} electrons in the next outer shell (n_{k+1} and l_{k+1}) couple to a term ${}^{2S_{k+1}+1}L_{k+1}$. This term again couples with the joint term of all shells with an index $a < k+1$. This leads to the used notation[16]:

$$\left(\ldots\left(\left(n_1 l_1^{x_1}\left({}^{2S_1+1}L_1\right)\ n_2 l_2^{x_2}\left({}^{S_2+1}L_2\right)\right)\ {}^{2S_{1\&2}+1}L_{1\&2}\right)\ldots n_m l_m^{x_m}\left({}^{S_m+1}L_m\right)\right)\ {}^{2S+1}L_J^{(\circ)} \tag{2.71}$$

One has to bear in mind that the simple version of jj-coupling introduced in equation 2.67 appears in the more sophisticated version in the end (red) and the "parental history" (blue) of the term symbol appears in the outermost parentheses (black):

$$\left(\ldots\left(\left(n_1 l_1^{x_1}\left({}^{2S_1+1}L_1\right)\ n_2 l_2^{x_2}\left({}^{S_2+1}L_2\right)\right)\ {}^{2S_{1\&2}+1}L_{1\&2}\right)\ldots n_m l_m^{x_m}\left({}^{S_m+1}L_m\right)\right)\ {}^{2S+1}L_J^{(\circ)} \tag{2.72}$$

The partial orbital configuration $n_i l_i^{x_i}$ is to be interpreted as all partial orbital configurations with the same $n = n_i$:

$$n_i l_i^{x_i} \equiv n_i l_{i1}^{x_{i1}} n_i l_{i2}^{x_{i2}} \ldots n_i l_{ir}^{x_{ir}}, \tag{2.73}$$

if there are r possibly different values for l for the value n_i. This is simply a short notation that expresses a whole shell with the symbol of one subshell. For example $n_2 l_2^{x_2}$ expresses the same as $2s^2 2p^1$ in the configuration $1s^2 2s^2 2p^1 3s^1$. This configuration has the structure $n_1 l_1^{x_1} n_2 l_2^{x_2} n_3 l_3^{x_3}$ that is semantically equivalent to $n_1 l_{11}^{x_{11}} n_2 l_{21}^{x_{21}} n_2 l_{22}^{x_{22}} n_3 l_{31}^{x_{31}}$. So the expanded notation introduced in equation 2.71 is given with a non expanded orbital configuration notation.

2.2.2. jj-coupling

In the case of atoms with a large number of electrons the terms proportional to $\vec{l_i} \cdot \vec{s_i}$ are more important than the electrostatic behavior in the mean field. So the latter are considered as a small perturbation (Bransden and Joachain, 2003). This fact leads to the behavior of the atoms that their electrons first couple their $\vec{l_i}$ and $\vec{s_i}$ to single electron total angular momenta $\vec{j_i}$ and those add up (vectorially) to a whole system total angular momentum $\vec{J}$:

$$\vec{j_i} = \vec{l_i} + \vec{s_i} \tag{2.74}$$

[16] One has to keep in mind that this is just one way to get a unique set of term symbols. In fact a lot of histories are possible and it is possible to show that all those lead to the same physically possible levels with the same J. This is done for example in (Condon and Odabasi, 1980). This way is chosen because it couples first those electrons together that are "nearer" and then those that are "more far" away from each other because of them being farther away from the nucleus and more importantly because Palmeri chose this coupling order (AUTOSTRUCTURE) and the program written as part of this thesis is intended to compare this results with results of others like FAC and HULLAC.

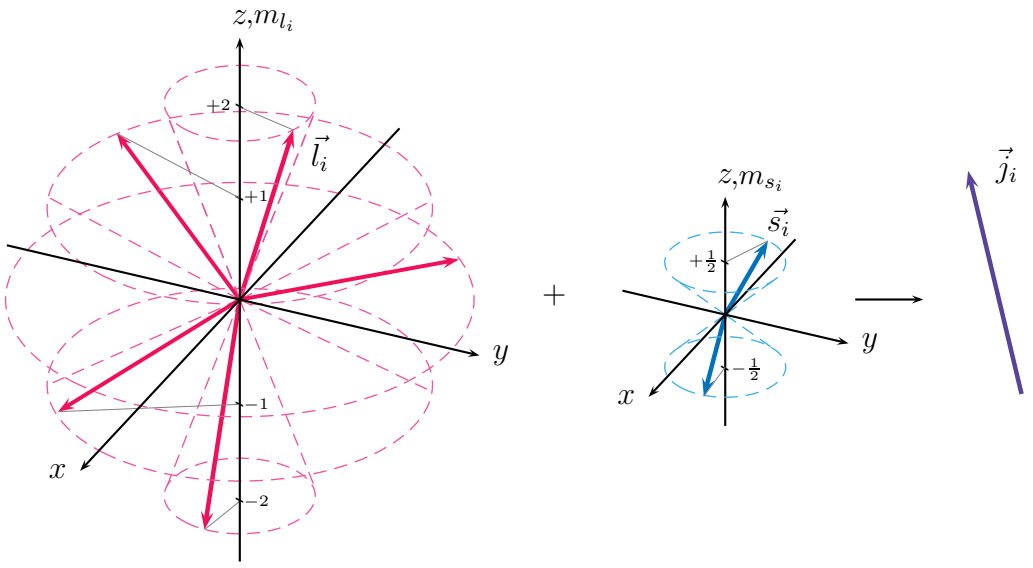


Figure 2.6.: Illustration of the angular momentum vector $\vec{l}_i$ of a single electron adding up with the single electron's spin $\vec{s}_i$ yielding the single electron's total angular momentum $\vec{j}_i$ (equation 2.74). That this happens "before" those $\vec{j}_i$ are adding up to the whole atom's total angular momentum distinguishes the jj-coupling from the LS-coupling. Again all those vectors are on cones due to the uncertainty principle and their lengths are equal to $\sqrt{l_i(l_i+1)}$, $\sqrt{s_i(s_i+1)}$ and $\sqrt{j_i(j_i+1)}$. All physically possible (Pauli exclusion principle) orientated single electron variable vectors "then" vectorially add up to the whole system total angular momentum $\vec{J}$ (see figure 2.2.2).

$$\vec{J} = \sum_{i=1}^{N} \vec{j}_i \tag{2.75}$$

The same combinatoric machinery as in the LS-coupling takes place in the jj-case; but the other way around. The single electron vector quantities have some possible m_{l_i} and m_{s_i}. "Then" they could add up to lots of possible $\vec{j}_i$. Finally those – holding values of m_{j_i} – add up to J. But again only those combinations of the single electron quantities that obey the Pauli principle are physically possible. In the case of LS-coupling the equivalent electrons are those with a common n and l, but in the case of jj-coupling the electrons that have a common n_i, l_i, j_i. Also the set of quantum numbers that are important for the coupling are n, l, j, m_j[17].

[17]But for the filling of the electron orbitals the "normal" quantum numbers n, l, m_l, m_s are important. Additionally in both cases s is an important quantum number, but it is suppressed in every notation and never stressed to be important, because it is the same ($s = \frac{1}{2}$) for all electrons as mentioned in the theory section.

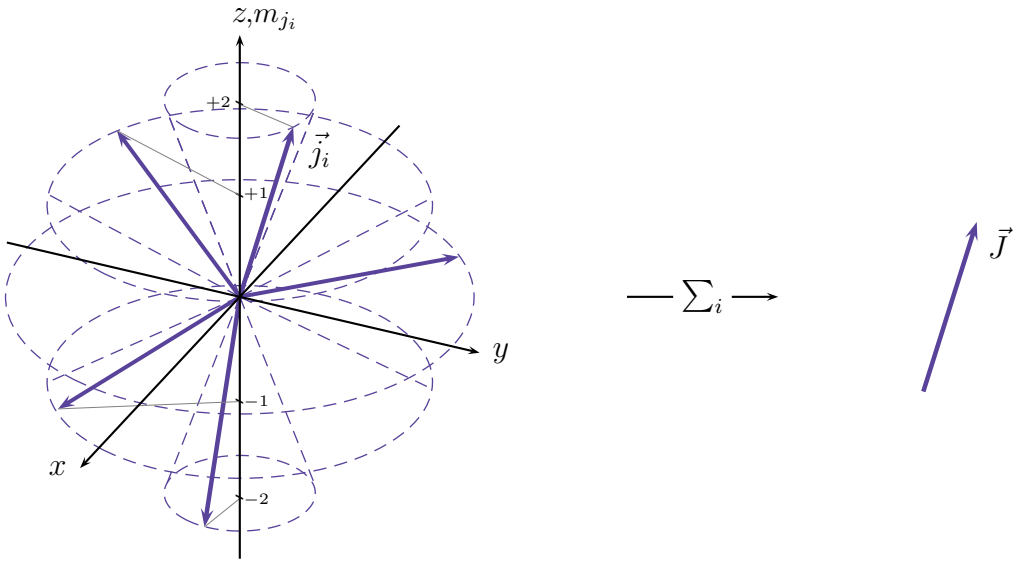


Figure 2.7.: Illustration of the single electrons angular momentum vector $\vec{j}_i$ for the case of $j_i = 2$. This enables the vector to have a m_{j_i} from -2 up to $+2$ in steps of the size 1. All those vectors are on cones due to the uncertainty principle and their length is equal to $\sqrt{j_i(j_i+1)}$. All physically possible (Pauli exclusion principle) orientated groups of single electron angular momenta for the whole atom "then" add vectorially up to the total system angular momentum $\vec{J}$ (equation 2.75).

For this coupling scheme there are several conventions to denote the possible states. The following general form was chosen to enable the simple identification of possible transitions between the term symbols (Herzberg, 1937):

$$(j_1, j_2, \ldots, j_N)_J^{(\circ)} \tag{2.76}$$

The "∘" in the top right index denotes the odd parity if present just as used in LS-coupling nomenclature. To provide a more precise indication of possible states by these terms it is possible to couple electrons in the same shell of equivalent electrons (same n_i, l_i, j_i) first and then couple those shell total angular momenta successively, similarly to the parental history in LS-coupling. This could be denoted by grouping all j belonging to one shell in parentheses and adding a common j_{group} in the lower right index:

$$[n_i l_i \pm]_{j_i}^{x_i}, \tag{2.77}$$

where + denotes spin up ($m_s = \frac{1}{2}$) and − denotes spin down ($m_s = -\frac{1}{2}$). For example, $d+$ is an equivalent expression for $j = \frac{5}{2}$. This shell can then couple its whole shell's total angular momentum to the "next" outer shell. An important point about this nomenclature is that there could be parts of the orbital configuration with the same n_i and l_i, because they have a different m_s leading to a different j_i. This leads to the following form of term symbols in jj-coupling:

$$\left(\ldots \left([n_1 l_1 \pm]_{j_1}^{x_1} \, [n_2 l_2 \pm]_{j_2}^{x_2} \right)_{j_{1\&2}} \ldots [n_k l_k \pm]_{j_k}^{x_k} \right)_J^{(\circ)} \tag{2.78}$$

Hence the indices in 2.78 are not the same as in 2.69, because now they count the portions of equivalent electrons in jj-coupling. One has to bear in mind that the simple version introduced in equation 2.76 now appears distributed over whole term symbol (red) and the "parental history" (blue) is distributed over the whole term symbol as well:

$$\left(\ldots \left([n_1 l_1 \pm]_{j_1}^{x_1} \, [n_2 l_2 \pm]_{j_2}^{x_2} \right)_{j_{1\&2}} \ldots [n_k l_k \pm]_{j_k}^{x_k} \right)_J^{(\circ)} \tag{2.79}$$

In this equation the "last" j_{group} (to be exact $j_{1\&\ldots\&m}$) is the result of successively coupling all subshells and therefore all electrons. This is the same as the whole atom's total angular momentum J and finally just "J" is denoted in the lower right index of the jj-term symbol. This notation is modeled on the history stressing notation introduced for LS-coupling and the output of the already mentioned atomic code FAC.

2.2.3. A problem of a simple assignment of LS- to jj-coupling term symbols: base transform

States in the one coupling scheme are not directly related to states in the other (Landau and Lifshitz, 1987; Condon and Shortley, 1970; Condon and Odabasi, 1980). Since term symbols in one scheme are linear combinations of the states in the other. To illustrate this the case of two entities having angular momenta and the base transform from the

product state Hilbert space to the common angular momentum Hilbert space is examined (Eisberg, 1985):

$$\mathcal{H}_{j_1} \otimes \mathcal{H}_{j_2} \to \mathcal{H}_J. \tag{2.80}$$

The angular momenta j_1 and j_2 and their total angular momentum J can be exchanged with all the "angular-momentum-like" values such as:

- l_i and s_i and their common angular momentum value j_i,
- $j_{1\&\ldots\&k}$ and j_l and their added $j_{1\&\ldots\&k\&l}$ or
- L and S with their total angular momentum J.

In other words, there are "angular-momentum like" quantities at all levels that can couple. But there is a problem with the assignment of states in one scheme to those in the other: Both pictures are equivalent but states in the one description are sums of states in the other description weighted by the Clebsch-Gordan coefficients. These coefficients are the unitary base transform from the product-state base ($|j_1, m_{j_1}, j_2, m_{j_2}\rangle = |j_1, m_{j_1}\rangle \otimes |j_2, m_{j_2}\rangle$) to the eigenbase of the coupled states ($|j_1, j_2, J, m_J\rangle$):

$$|j_1, j_2, J, m_J\rangle = \sum_{m_{j_1}, m_{j_2} = -j_1, -j_2}^{j_1, j_2} |j_1, m_{j_1}, j_2, m_{j_2}\rangle \langle j_1, m_{j_1}, j_2, m_{j_2}|\ j_1, j_2, J, m_J\rangle\,, \tag{2.81}$$

where the orthonormality of the product-state base was used by multiplying with the identity from the left. The complex numbers $\langle j_1, m_{j_1}, j_2, m_{j_2}|\ j_1, j_2, J, m_J\rangle$ are the Clebsch-Gordan coefficients (Fließbach, 1991). Such a base transform is necessary in every coupling step to have the total angular momentum operator diagonalized simultaneously with the operator of its z-component (Condon and Shortley, 1970; Condon and Odabasi, 1980). The common states of single electrons l_i and s_i are therefore linear combinations of the single angular momentum states. The coupling states of these states are also linear combinations of the j_i and so the resulting jj-coupling terms are linear combinations of the whole set of single angular momentum states l_i and s_i ($\forall i = 1, \ldots, N$). Similarly in the case of LS-coupling the successive coupling of the l_i to the L and of the s_i to the S leads – finally with the coupling of L and S – to the fact that also the LS-coupling terms are linear combinations of the single angular momentum states belonging to all the l_i and s_i. Finally all the terms (and hence the term symbols) are linear combinations of the fundamental angular momentum states $|l_i, m_{l_i}\rangle$ and $|s_i, m_{s_i}\rangle$ but – and this is the important fact – *in different ways*. Without exceptions no state in the one coupling corresponds to the other. They are all formed out of many other states in the other coupling scheme weighted with base transform coefficients (like the Clebsch-Gordan coefficients). This means that the way how the basal angular momenta s_i and l_i combine linearly is different for the two coupling cases but in both cases the final states build a base in which the $\hat{J}_{jj}$ and $\hat{J}_{LS}$ are diagonal and yield the same eigenvalues J (Condon and Shortley, 1970; Condon and Odabasi, 1980).
The idea of bypassing this issue when trying to obtain corresponding levels is to produce all term symbols, sort them energetically (as good as possible) and then assign the xth

term symbol in the LS-coupling list to a term symbol close to xth entry of the jj-coupling list.
Herzberg (1937) states that an "unambiguous correlation is possible", because in both schemes are equally many term symbols with the same values of J. But two sentences later he weakens this statement and says that it "has only a very limited value".

2.2.4. Hund's rules

Another problem of this assignment method is that **Hund's rules** (Hund, 1925; Herzberg, 1937; Engel, 2006):

1. The term, and thus the term symbol, with maximal S has the lowest energy,
2. If there is more than one term, the term with maximal L has the lowest energy,
3. If the open subshell is less than half filled the term symbol that has the lowest J has the lowest energy. If the open subshell is more than half filled, the term symbol that has the highest J has the lowest energy.

are only valid if the Coulomb interaction of the outer electrons is stronger than the spin-orbit interaction. This is exactly the validity regime for LS-coupling ($\hat{H}_{\mathrm{rest}} \gg \hat{H}_{\mathrm{s-o}}$) and not for jj-coupling ($\hat{H}_{\mathrm{s-o}} \gg \hat{H}_{\mathrm{rest}}$). A mapping by sorting is not very effective if just one list is sorted by means of energy and the other is not.
If there is more than one open subshell, Hund's rules do not tell how to treat this case. I chose the outermost open subshell for the application of the third rule.
There is even a bigger problem. Hund's rules are itself not very good to *sort* even LS-coupling terms, because they are intended to just find the level of lowest energy, the ground state. They are phrased in a way that only allows to abuse them by picking out two terms and finding the "ground state" of those two terms and to get this way a comparing tool for the energy of terms. But this is not what Hund's rules are able to. They just inform the user of them about what term is the ground state out of *all* terms belonging to *one* electron configuration.
Nevertheless this mapping approach was chosen, because misusing Hund's rules is the best of all bad alternatives. Most likely the mapping is nonsense in a strict sense. But using Hund's rules was the only possibility and it may at least be not too wrong in some cases. Slightly exaggerated one could say, doing any sorting is better than a random order.

3. Internal principles of operation of the program

Based on the discussion of the previous chapter I have written a program as the main part of this thesis. The program is written in the language Python. It takes the orbital configuration as the input and produces as the output a list containing lists with the quantum numbers for the states in both jj- and L-S-coupling. It sorts them by Hund's rules and writes this output to a latex table contained in a .tex file. Optionally a FAC notation output can also be written to a file. This option is turned on by default. Python allows fast handling of out- and input, because of an easier handling of data types (nearsighted thought) and because of the nice iteration module (itertools) that is available for Python and makes tasks like combinations, permutations, or cartesian products a simple step. These operations are important as examined below.

3.1. LS-coupling terms producing algorithm

The algorithm used to produce LS term symbols is the one proposed by Doggett and Sutcliffe (1998). The algorithm is able to produce all LS-coupling term symbols for a given electron configuration of arbitrary complexity, but in the program the electron configuration is partitioned into all shells that contain electrons and then this algorithm is just used for producing all terms that arise from this shell. Afterwards those shells are coupled successively just using the triangular condition, Eq. 2.66.

In general the input information is the quantum numbers of all electrons $n_1 l_1^{x_1} n_2 l^{x_2} \dots n_m l_m^{x_m}$ and the **algorithm** is:

1. - The total number of electrons N is calculated by $N = \sum_{i=1}^{m} x_i$.
   - The total number of l-degenerated orbitals n is calculated by summing up all l-multiplicities $n = \sum_{i=1}^{m} (2l_i + 1)$.
2. All those orbitals get assigned with an unique integer. Explicitly this means that all those m_{l_i} are sorted arbitrarily and then are mapped onto the set $\{1, 2, 3, \dots, n\}$.
3. All Weyl diagrams are produced that belong to N. A Weyl diagram is a two dimensional array consisting of two columns. The sum of the two lengths of the columns has to be N: $\lambda_1 + \lambda_2 = N$, where λ_i is the length of column number i. Additionally, to ensure that there are no duplicates the second column must not

be longer than the first: $\lambda_1 \geq \lambda_2$. The fields in the first column represent electrons with spin up and the field in the second column electrons with spin down. The values $S = \frac{\lambda_1 - \lambda_2}{2}$ are calculated and saved for each diagram.

4. A Weyl diagram is picked out and filled with the values numbering the orbitals in every way it is possible to fulfill the following conditions (for the implementation of these steps the itertools module is helpful):
   - The numbers are not allowed to decrease in a row from small indices to high indices of the fields in that row.
   - The numbers in a column must increase.
   - No more values assigned to m_{l_i} that belong to l_i are allowed to be filled into one Weyl tableaux than electrons are in the l_i shell. This basically means that a magnetic quantum number m_{l_i} is assigned to every electron in this shell, obeying the Pauli exclusion principle.

   Those filled Weyl diagrams that are possible are called *Weyl tableaux*.

5. For each type of Weyl tableaux (unique set of λ_1 and λ_2) the following steps are to be executed:
   - Sum all m_{l_i} in one tableaux to get m_L for all the tableaux: $m_L = \sum_{\mathrm{i}\ \substack{in\ one \\ tableux}} m_{l_i}$.
   - Count how often the maximum m_{l_i} occurs and strike out that many occurrences of all tableaux having a m_L from $-m_{L,\max}$ to $+m_{L,\max}$ in steps of size one. All of these belong to a term that has $L = m_{L,\max}$ as its angular momentum quantum number and $S = \frac{\lambda_1 - \lambda_2}{2}$ as its spin. Therefore to the final list of resulting terms the term ${}^{2S+1}L$ as many times as the maximum $m_{L,\max}$ was counted.
   - Again in the remaining list the maximum element is identified, counted and striked off that often with all the associated tableaux with $m_L = -m_{L,\max} + 1, \ldots, +m_{L,\max}$ and that many terms with that $m_{L,\max} = L$ are added to the list of resulting terms as often the maximum $m_{L,\max}$ appeared.
   - This process is iterated until the list of Weyl tableaux is empty.

6. The last two steps (4. and 5.) are executed for all Weyl diagrams and the resulting terms are added to the previous list of terms.

7. After all Weyl diagrams are processed the resulting list should contain all LS-coupling terms without their parental history, but with the number of occurrences.

This is just the algorithm that is used for one shell. The parental history turns up by applying this algorithm to every shell and afterwards coupling those shells successively using the triangular condition, Eq. 2.66.

As an example the configuration $2d^2$ ($n_1 = 1, l_1 = 2, x_1 = 2$) is carried out in the following:

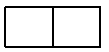
1.
   - The number of electrons is $N = \sum_{i=1}^{1} x_i = x_1 = 2$.
   - Number of all possible magnetic quantum numbers m_{l_i} $n = \sum_{i=1}^{1}(2l_i + 1) = 2l_1 + 1 = 2 \cdot 2 + 1 = 5$.

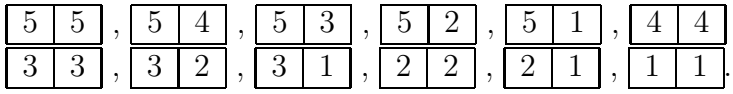
2. All those orbitals get assigned with an unique integer:

m_{l_1}	-2	-1	0	$+1$	$+2$
labeled by	1	2	3	4	5

3. All possible Weyl diagrams are:

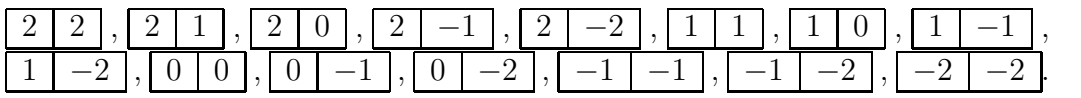
	$1 = \lambda_1 \leq \lambda_2 = 1$	$\lambda_1 + \lambda_2 = 1 + 1 = 2 = N$	$S = \frac{\lambda_1 - \lambda_2}{2} = \frac{1-1}{2} = 0$
	$2 = \lambda_1 \leq \lambda_2 = 0$	$\lambda_1 + \lambda_2 = 2 + 0 = 2 = N$	$S = \frac{\lambda_1 - \lambda_2}{2} = \frac{2-0}{2} = 1$

4. Picking out the Weyl diagram  and filling with the labels of the m_{l_i} yields:



5. Producing LS-terms by the following steps:

   - The Weyl tableaux are translated back to the real m_{l_i} values:

     This leads to the following list of $m_L \in \{4,3,2,2,1,1,0,0,0, -1,-1,-2,-2,-3,-4\}$.
   - The maximum element $m_{L,\max} = 4$ and it occurs once. Thus the elements 4,3,2,1,0,$-$1,$-$2,$-$3,$-$4 are striked off the list and the remaining $m_L \in \{2,1,0,0,-1,-2\}$. The spin for this type of Weyl tableaux is $S = 0$ that has the multiplicity $2S + 1 = 1$. The angular momentum of the term is $L = m_{L,\max} = 4 \equiv$ G. Hence the term ${}^1G$ is added to the list of terms.
   - The new maximum element $m_{L,\max} = 2$ and it occurs once. Thus the elements 2,1,0,$-$1,$-$2 are striked off the list and the remaining list is $\{0\}$. The spin for this type of Weyl tableaux is again $S = 0$ and the angular momentum of the term is $L = m_{L,\max} = 2 \equiv$ D. Hence the term ${}^1D$ is added to the list of terms $\{{}^1G, {}^1D\}$.
   - The new maximum element $m_{L,\max} = 0$ and it occurs once. Thus the element 0 is striked off the list and the remaining list is empty forcing the program to go to the next unprocessed Weyl tableaux. The spin $S = 0$ and so the new term is $L = m_{L,\max} = 0 \equiv$ S. Hence the term ${}^1S$ is added to the list of terms $\{{}^1G, {}^1D, {}^1S\}$.

6. The next and last Weyl diagram to be picked out is $\begin{array}{|c|}\hline \\ \hline \\ \hline \end{array}$. This yields the following Weyl tableaux:

$$\begin{array}{|c|}\hline 4 \\ \hline 5 \\ \hline \end{array}, \begin{array}{|c|}\hline 3 \\ \hline 5 \\ \hline \end{array}, \begin{array}{|c|}\hline 2 \\ \hline 5 \\ \hline \end{array}, \begin{array}{|c|}\hline 1 \\ \hline 5 \\ \hline \end{array}, \begin{array}{|c|}\hline 3 \\ \hline 4 \\ \hline \end{array}, \begin{array}{|c|}\hline 2 \\ \hline 4 \\ \hline \end{array}, \begin{array}{|c|}\hline 1 \\ \hline 4 \\ \hline \end{array}, \begin{array}{|c|}\hline 2 \\ \hline 3 \\ \hline \end{array}, \begin{array}{|c|}\hline 1 \\ \hline 3 \\ \hline \end{array}, \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline \end{array}$$

7. - This list results in the following real m_{l_i} values:

$$\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline \end{array}, \begin{array}{|c|}\hline 0 \\ \hline 2 \\ \hline \end{array}, \begin{array}{|c|}\hline -1 \\ \hline 2 \\ \hline \end{array}, \begin{array}{|c|}\hline -2 \\ \hline 2 \\ \hline \end{array}, \begin{array}{|c|}\hline 0 \\ \hline 1 \\ \hline \end{array}, \begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline \end{array}, \begin{array}{|c|}\hline -2 \\ \hline 1 \\ \hline \end{array}, \begin{array}{|c|}\hline -1 \\ \hline 0 \\ \hline \end{array}, \begin{array}{|c|}\hline -2 \\ \hline 0 \\ \hline \end{array}, \begin{array}{|c|}\hline -1 \\ \hline -2 \\ \hline \end{array}$$

   - Adding those values in each diagram yields $m_J \in \{3, 2, 1, 1, 0, 0, -1, -1, -2, -3\}$. The maximum element $m_{J,\max} = 3$ and thus $3, 2, 1, 0, -1, -2, -3$ are striked off leaving $m_{l_i} \in \{1, 0, -1\}$. The spin of this type of Weyl diagram is $S = 1$, hence the multiplicity $2S+1 = 3$. The angular momentum $L = m_{J,\max} = 3 \equiv$ F. The term ${}^3F$ is added to the list of terms $\{{}^1G, {}^1D, {}^S, {}^3F\}$.
   - The new maximum $m_{J,\max} = 1$, yielding an empty list after striking out $1, 0, -1$ forcing the complete algorithm to stop, because no type of diagram and for this type no tableaux is left. The final term added has again $S = 1$ and $L = m_{J,\max} = 1 \equiv$ P: ${}^3P$.

The algorithm produced the terms ${}^1G$,${}^1D$,${}^1S$,${}^3F$,${}^3P$.

3.2. jj-coupling term symbols producing algorithm

The jj-coupling term symbols are produced by a straightforward approach. After all possible partitions into $n_i l_i-$ and $n_i l_i+$ had been calculated, the following algorithm is used to couple those jj-coupling-equivalent electrons to get jj-coupling terms for each $n_i l_i \pm$. Those terms then are coupled together successively as in the case of LS-coupling using the triangular condition 2.66. So the input is $[n_i l_i]^{x_{i-}}$, $[n_i l_i+]^{x_{i+}}$ for all subshells i.

The **algorithm** is as follows:

1. For each electron every value of m_{j_i} is generated: $m_{j_i} = -j_i, -j_i + 1, \ldots, +j_i$ and all the quantum numbers n_i, l_i, j_i, m_{j_i} are saved. The cartesian product of all those single electron values is produced.

2. All those products are scanned for duplicates of quantum numbers. Those products are deleted, because they disobey the Pauli exclusion principle. Additionally, all items in the list are checked for items that are equal under permutation of the electrons and just one of those is kept due to the interchangeability of electrons.

3. Then all the m_{j_i} are added to yield a value m_J. Now there is a list of all physically possible products and their assigned m_J and all the information about the single electrons including there $j_i \equiv l_i \pm$.

4. In this list the maximum $m_{J,\max}$ is identified and counted[1]. As in the case of LS-coupling that many occurrences of products with $m_J = -m_{J,\max}, -m_{J,\max} + 1, \ldots, +m_{J,\max}$ are deleted as often $m_{J,\max}$ was counted. A term with $J = m_{J,\max}$ is added to the list of resulting terms as often as $m_{J,\max}$ was counted.

5. In the remaining list the new maximum $m_{J,\max}$ is identified and counted and step 4. is executed. This procedure is repeated until the list of products is empty. The resulting terms of step 4 are appended to the list of resulting terms and this list is in the end the list of jj-coupling terms.

As in the case of LS-coupling an example shall clarify the algorithm. The examined configuration is $[2s+]^2$.

1. The m_{l_i} for each electron are:

electron 1	$m_{l_i} = +\frac{1}{2}, -\frac{1}{2}$
electron 2	$m_{l_i} = +\frac{1}{2}, -\frac{1}{2}$

   The cartesian product is: $(+\frac{1}{2},+\frac{1}{2})$, $(+\frac{1}{2},-\frac{1}{2})$, $(-\frac{1}{2}, +\frac{1}{2})$, $(-\frac{1}{2}, -\frac{1}{2})$.

2. Just $(+\frac{1}{2},-\frac{1}{2})$, $(-\frac{1}{2}, +\frac{1}{2})$ stay in the list after deleting those that harm the Pauli exclusion principle and because they are equal if one interchanges entry one and two of one of these two tuples, just $(+\frac{1}{2},-\frac{1}{2})$ is in the list.

3. Summing both entries of $(+\frac{1}{2},-\frac{1}{2})$ yields 0, hence the list of all $m_J = 0$.

4. The maximum $m_{J,\max} = 0$. After removing 0 from the list, the list is empty and the algorithm stops, yielding one term symbol with $J = m_{J,\max} = 0$

So the algorithm returns the term symbol, namely $[2s+]^2_0$.

3.3. Overall structure of the program

The first step the program executes is to read in the orbital configuration in the FAC input notation (see next section for a description). This notation can be ambiguous. Thus the program evaluates all possibilities for the shell and then produces a list of all the cartesian products of the shell configurations. Now there is a list containing all possible electron configurations that are meant by the FAC input. The following steps are executed for all those configurations:

1. The electron configuration is split into all shells separately and stored in a list A.

[1] Which should be one, because the algorithm is used in a way, that every jj-symbol is unique and therefore no term symbol should occur more often than exactly one time.

2. For all the items in A the algorithm, described in section 3.1, is applied and the results are appended to a list B.

3. The cartesian product of all those coupling results of the single shells is built. All those products are appended to a list C.

4. Every list in C is sorted by means of the shell from small n to larger n, where n is the principal quantum number.

5. Every item of list C is possible way the single shells could have coupled and for all those possibilities successively those shells get coupled by:

   a) Producing the values $L_{i\&j}$ and $S_{i\&j}$ by using the triangular condition 2.66 for L_i along with L_j for $L_{i\&j}$ and S_i together with S_j to get $S_{i\&j}$. First i and j are 1 and 2. The results are saved.

   b) i is set to 1&2 and j to 3. Step one is executed again.

   c) In general i is set to $1\&2\&\ldots\&k-1$ and j to k. This is done until k equals to the cardinal number of list B (See the example in the previous section). So the coupling can be exemplified this way using a formal expression

   $$(\ldots(((1\&2)\&3)\&4)\ldots\&k_{\max}), \tag{3.1}$$

   if there are $k_{\max}$ shells and & denotes the coupling.

   d) This is leading to a final term for which the whole history is saved.

   e) In the last step the final L and S are coupled to yield the total J.

6. All those term symbols belonging to one configuration are appended to a list D.

7. Finally they get sorted by the third of Hund's rules, because the first two are given implicitly in the previous algorithm. In some steps it is necessary to eliminate duplicates and therefore the lists got sorted by means of "something" (this is necessary for an implemented function that eliminates duplicates). And this "something" was chosen to be one time L and the other time S.

8. Now the jj-coupling terms are produced by splitting A in a different way than for LS-coupling term symbols. Now it is split into sub shells (same n and l), yielding a list E.

9. For all items in E all possible single electrons j_i are calculated and stored grouped in jj-coupling subshells ($[n_i l_i\pm]^{x_{i\pm}}$). The cartesian product of all those physically possible partitionings of $n_i l_i$ into $[n_i l_i]^{x_{i-}}[n_i l_i+]^{x_{i+}}$ is saved in list F, where $x_{i\pm}$ is thenumber of electrons in $[n_i l_i\pm]$.

10. The same successive coupling as in the case of LS-coupling takes place for every possible product of shell partitions. This is again done by using the triangular condition 2.66.

11. Finally all those terms get assigned with J using the triangular condition 2.66.

12. Those term symbols get sorted by means of the sum of the involved j_i and not by J yielding a final jj-coupling term symbols containing list G for this particular configuration.

Afterwards all term symbols are handled to a function that takes a term symbol and produces the FAC output and a LaTeXversion. All those are the output for one configuration. But there are many configurations if the FAC input allowed more than one. The first list A containing all those electron configurations is sorted by the following rule: A configuration is higher in energy if more electrons are in sub shell that are situated further away from the nucleus[2].
Finally all the term symbols in both coupling schemes, sorted by Hund' rules as good as possible inside an electron configuration and sorted by the number of electrons in more outer subshells, are written to the two files hclsvsjj.tex and hclsvsjj.txt. The first one is ready to get composed by LaTeXand the second one contains the machine readable output that is also produced by FAC so that the user could compare FAC to other codes more easily. 3.1 visualizes the overall structure of the program in a simplified manner.
"hclsvsjj" is a acronym for highly charged LS versus jj, because the program is inter alia intended to examine highly charged ions. The program itself is a python script and is therefore named hclsvsjj.py. The lines above and below the latex table in the hclsvsjj.tex file is contained in the two files header.txt and foot.txt.

[2]This is not really true. "Situated further away from the nucleus" means here that n is larger and for the same n l has to be larger. Whether a bigger n or l is correlated with being further outside the atom or not is a difficult question to answer. In general it is wrong or just a statement without much meaning, because there are many maxima of $|\psi_{n_i l_i}|^2$ in general. The mean location of most likely measuring the electron maybe sortable, but again this is not very meaningful, because the electron could have a minimum of its ρ there. In spite of this being the mean value, it is very unlikely to find the electron there.

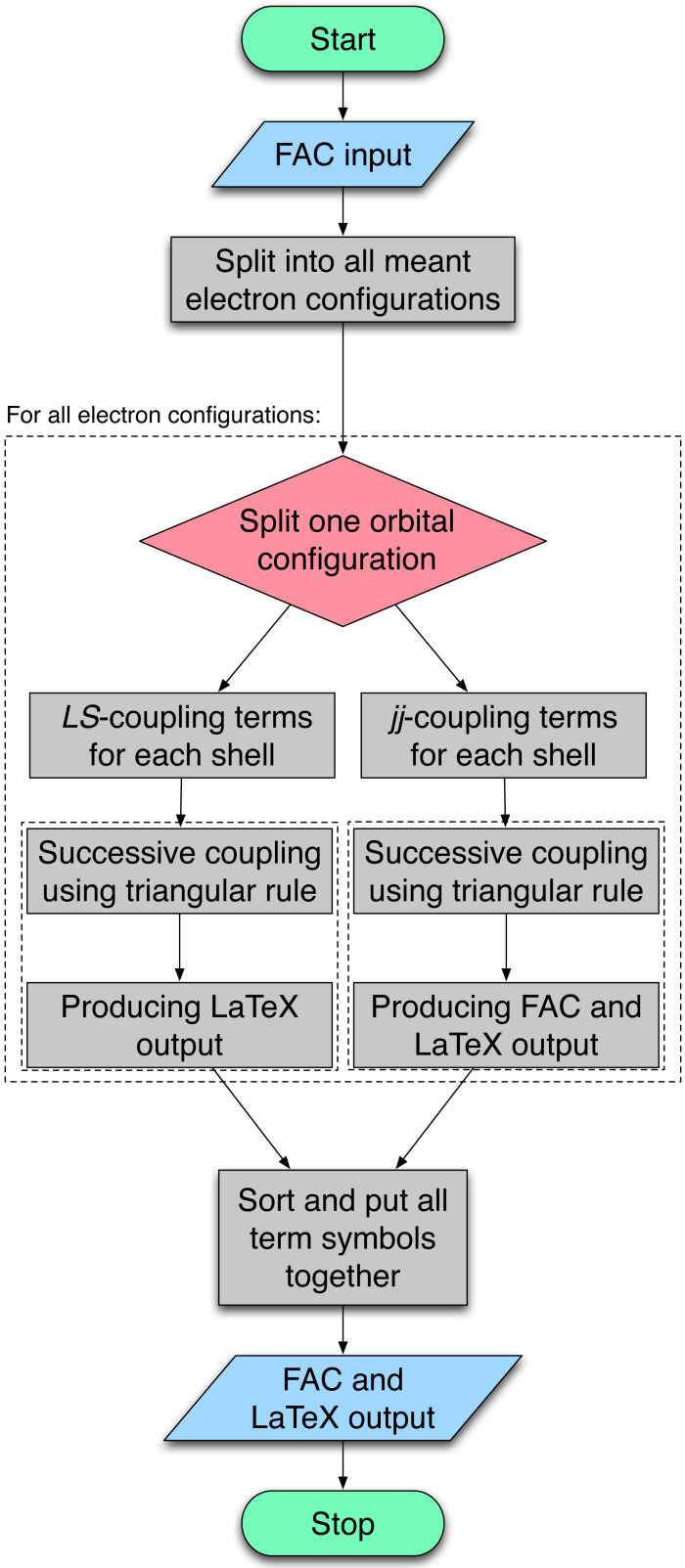


Figure 3.1.: Simplified flowchart of the overall structure of the script.

4. On the usage of the program

In the following section the input and output of the program is described. It is to be seen as a short manual enabling the user of the Python script to obtain LS- and jj-coupling term symbols listed together in a .pdf file and additionally obtaining the FAC output of the jj-coupling term symbols in the output formatting as FAC uses in its energy level tables.

4.1. Input and output formatting

The information that has to be handed over to the function, is just the electron configuration $n_1 l_1^{x_1} n_2 l^{x_2} \dots n_m l_m^{x_m}$ and the output is a list of all term symbols. But the formatting of both is important.

4.1.1. FAC input

If the program is executed the user is asked to write the orbital configuration in the style of the input of an electron configuration in the atomic code FAC.
This input style has the following rules:

- Sub shells are separated by a space.
- Full sub shells can and empty sub shells must be omitted.
- The value of n_i is typed in first as an integer: $1, 2, 3, 4, \dots$.
- The value of l_i is typed in lower case following the spectroscopic notation: $s, p, d, f, \dots$.
- The number x_i of electrons in that sub shell as an integer: $1, 2, 3, 4, \dots, 2l_i + 1$.
- There are two options to group more than one sub shell together by distributing the number of electrons given to all of them over them in all possible ways:
  1. The electrons are distributed to all the sub shells with angular momenta inside square brackets $[l_a, \dots, l_b]$ separated by a comma.
  2. The electrons are distributed over all possible l_{ji} sub shells in all possible ways for the particular n_i, if there is a $*$ between n_i and the number of electrons M: $n_i * M$.
  3. Those ambiguous expressions are separated from other sub shells by a space, too.

For example

```
Enter the configuration (FAC notation): 1s1 2*2 3[s,p]1
```

leads to all the following electron configurations as an input for the rest of the program:

$$
\begin{array}{ccc}
1s1 & 2s2 & 3s1, \\
1s1 & 2s2 & 3p1, \\
1s1 & 2s12p1 & 3s1, \\
1s1 & 2s12p1 & 3p1, \\
1s1 & 2p2 & 3s1, \\
1s1 & 2p2 & 3p1.
\end{array}
$$

Then all those electron configurations are processed by the program.

4.2. Sample procedure

For the ambiguous expression $1s1\ 2*3\ 3s1$ the results are shown below and in this sample procedure a closer look is given to configuration $1s^2 2s^2 2p^1 3s^1$. The whole output is given in the appendix.
The program shows the actual configuration and the term symbols that belong to this configuration as an indicator of progress on the standard system output:

```
Enter the configuration (FAC notation): 1s2 2*3 3s1


------------

Obtaining term symbols for the configuration:  [[1, 0, 2], [2, 0, 2], [2, 1, 1], [3, 0,

2p-1(1)1 3s+1(1)0
2p-1(1)1 3s+1(1)2
2p+1(3)3 3s+1(1)2
2p+1(3)3 3s+1(1)4
------------

Obtaining term symbols for the configuration:  [[1, 0, 2], [2, 0, 1], [2, 1, 2], [3, 0,

2s+1(1)1 3s+1(1)0
2s+1(1)1 3s+1(1)2
2s+1(1)1 2p-1(1)0 2p+1(3)3 3s+1(1)2
2s+1(1)1 2p-1(1)0 2p+1(3)3 3s+1(1)4
2s+1(1)1 2p-1(1)2 2p+1(3)1 3s+1(1)0
2s+1(1)1 2p-1(1)2 2p+1(3)1 3s+1(1)2
2s+1(1)1 2p-1(1)2 2p+1(3)3 3s+1(1)2
2s+1(1)1 2p-1(1)2 2p+1(3)3 3s+1(1)4
2s+1(1)1 2p-1(1)2 2p+1(3)5 3s+1(1)4
2s+1(1)1 2p-1(1)2 2p+1(3)5 3s+1(1)6
2s+1(1)1 2p+2(0)1 3s+1(1)0
2s+1(1)1 2p+2(0)1 3s+1(1)2
2s+1(1)1 2p+2(4)3 3s+1(1)2
2s+1(1)1 2p+2(4)3 3s+1(1)4
2s+1(1)1 2p+2(4)5 3s+1(1)4
2s+1(1)1 2p+2(4)5 3s+1(1)6
------------

Obtaining term symbols for the configuration:  [[1, 0, 2], [2, 1, 3], [3, 0, 1]]
```

```
2p+1(3)3  3s+1(1)2
2p+1(3)3  3s+1(1)4
2p-1(1)1  2p+2(0)1  3s+1(1)0
2p-1(1)1  2p+2(0)1  3s+1(1)2
2p-1(1)1  2p+2(4)3  3s+1(1)2
2p-1(1)1  2p+2(4)3  3s+1(1)4
2p-1(1)1  2p+2(4)5  3s+1(1)4
2p-1(1)1  2p+2(4)5  3s+1(1)6
2p+3(3)3  3s+1(1)2
2p+3(3)3  3s+1(1)4


This is pdfTeX, Version 3.1415926-2.4-1.40.13 (TeX Live 2012)
 restricted \write18 enabled.
 entering extended mode
```

As an example, the LS-coupling term symbols for the orbital configuration $1s^2 2s^2 2p^1 3s^1$ are given below. They are sorted by means of Hund's rules in order of descending energy from bottom to top:

$$\begin{array}{l}
(((1s^2\,(^1S)\;\; 2s^2 2p^1\,(^2P))\;\; ^2P)\;\; 3s^1\,(^2S))\;\; ^3P_2 \\
(((1s^2\,(^1S)\;\; 2s^2 2p^1\,(^2P))\;\; ^2P)\;\; 3s^1\,(^2S))\;\; ^3P_1 \\
(((1s^2\,(^1S)\;\; 2s^2 2p^1\,(^2P))\;\; ^2P)\;\; 3s^1\,(^2S))\;\; ^3P_0 \\
(((1s^2\,(^1S)\;\; 2s^2 2p^1\,(^2P))\;\; ^2P)\;\; 3s^1\,(^2S))\;\; ^1P_1
\end{array}$$

The first term symbol has the following structure: The electrons in the shell with n_1 couple to a term ^1S and the electrons in the shell with $n_2 = 2$ couple to yield a term ^2P. Those two shell related terms couple together resulting in a term ^2P. "Then" all electrons (here one) in the shell with $n_3 = 3$ have no other choice (in this case) but to couple to yield the term ^2S. This term couples with the term that includes all inner shells (^2P) and so the final term describing the whole system arises: ^3P. As the 3 in the top left index suggests, this term splits into three term symbols (triplet) and the final term symbol with the maximum value of $J = 2 = L + S$ (see triangular condition 2.66 and figure 2.2.1) is: $^3P_2$. This final term symbol equals the simple version of the term symbol notation and can easily be read as the term symbol on the right:

$$(((1s^2\,(^1S)\;\; 2s^2 2p^1\,(^2P))\;\; ^2P)\;\; 3s^1\,(^2S))\;\; ^3P_2 \quad \rightarrow \quad ^3P_2$$

The three other terms are structured similarly. This is maybe a boring example, because all the terms have the same history. But this one was chosen, because the important point was to delight the process how one of those level histories comes to be[1]. The difference here in the resulting term symbols is based on the last two coupling steps: on the one hand the coupling of L and S to J and on the other on the two last total spins of ^2P and ^2S coupling to the final S.

The resulting jj-coupling term symbols for the same example as for LS-coupling ($1s^2 2s^2 2p^1 3s^1$) are:

[1] Obviously the parental history becomes more important for different histories of multiple occurrences of the same simple term symbol.

$$\left(\left(\left([1s+]_0^2\,[2s+]_0^2\right)_0\,[2p-]_{\frac{1}{2}}^1\right)_{\frac{1}{2}}\,[3s+]_{\frac{1}{2}}^1\right)_0$$
$$\left(\left(\left([1s+]_0^2\,[2s+]_0^2\right)_0\,[2p-]_{\frac{1}{2}}^1\right)_{\frac{1}{2}}\,[3s+]_{\frac{1}{2}}^1\right)_1$$
$$\left(\left(\left([1s+]_0^2\,[2s+]_0^2\right)_0\,[2p+]_{\frac{3}{2}}^1\right)_{\frac{3}{2}}\,[3s+]_{\frac{1}{2}}^1\right)_1$$
$$\left(\left(\left([1s+]_0^2\,[2s+]_0^2\right)_0\,[2p+]_{\frac{3}{2}}^1\right)_{\frac{3}{2}}\,[3s+]_{\frac{1}{2}}^1\right)_2$$

To clarify this somehow complicated notation the generation of a term symbol is examined. The last term symbol $\left(\left(\left([1s+]_0^2\,[2s+]_0^2\right)_0\,[2p+]_{\frac{3}{2}}^1\right)_{\frac{3}{2}}\,[3s+]_{\frac{1}{2}}^1\right)_2$ is of the following structure: The electrons in the subshell with $n_1 = 1$ and $l_1 = 0$ couple to the single electron total angular momenta $j = \frac{1}{2}$. Those are grouped together, because they are equivalent electrons: $[1s+]_0^2$. These two couple to the total angular momentum $j_1 = 0$, as the 0 in the lower right index denotes. Those in the second subshell with $n_2 = 2$ and $l_2 = 0$ also couple each to $j = \frac{1}{2}$. Again the only possible way to couple to a total angular momentum is to yield $j_2 = 0$, because as in the case of the first subshell the Pauli exclusion principle forces the one electron to have $m_j = +\frac{1}{2}$ and the other electron to have $m_j = -\frac{1}{2}$. Those two parts of a set of equivalent electrons couple and have a total subshell angular momentum of 0. The electron in the subshell with $n_3 = 2$ and $l_3 = 1$ has a j of $\frac{3}{2}$ and all the electrons in this subshell (here just one) couple to yield $j_3 = \frac{3}{2}$, because of the spin down denoted by the plus sign $+$. This angular momentum and the $j_{1\&2} = 0$ couple together to form a $j_{1\&2\&3} = \frac{3}{2}$. The electron in the subshell with $n_4 = 3$ and $l_4 = 0$ again has no other chance than to have a $j = \frac{1}{2} \equiv s+$ and all the electrons in this subshell (again one) couple to a $j_4 = \frac{1}{2}$. This group angular momentum couples with the $j_{1\&2\&3}$ to the whole system total angular momentum $J = j_{1\&2\&3\&4} = 2$. There is no "$\circ$" in the top left index, because the parity is even. The three other term symbols are built similarly. The simple version of this jj-term symbol is obtained by picking out the single electron angular momenta j_i and J:

$$\left(\left(\left(\left(1s^2_{\frac{1}{2},\frac{1}{2}}\right)_0\left(2s^2_{\frac{1}{2},\frac{1}{2}}\right)_0\right)_0\left(2p^1_{\frac{3}{2}}\right)_{\frac{3}{2}}\right)_{\frac{3}{2}}\left(3s^1_{\frac{1}{2}}\right)_{\frac{1}{2}}\right)_2 \quad \Rightarrow \quad \left(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{3}{2},\tfrac{1}{2}\right)_2$$

5. Conclusion

> The purpose of computing is insight, not numbers.
>
> Richard Hamming

The purpose of this work was to write a program that enables the user to get an assigned jj-coupling term symbol for each LS-term symbol and vice versa. This should be a tool to compare the energetic structure delivered by atomic codes for astrophysics. The original idea of and need for this assignment arose in the master thesis of Nathalie Hell (2012). There the idea was to sort and assign the term symbols by means of J manually, but it turned up that this is a hard task, especially for more complex configurations. Hence it was self-evident trying to do things automatically using a program and I started to write a program.
While writing a simple version of the program, a problem arose with several terms independent of the sorting issues that were mentioned in the theory section. Some terms occurred more than once and mapping two different sized lists with elements that have multiple occurrences are not good candidates for finding a bijection between them. The solution was to take their parental history into account and this was done in the style of the results that are planned to be compared in the near future: FAC and AUTOSTRUCTURE. Now those lists have – as they should, otherwise something would be fairly wrong – the same number of entries. So one could at least check whether there are all possible term symbols due to angular momentum coupling or not. This is yet another advantage of having done this work. Those term symbols are sorted and despite of this sorting being not aware of objection, the result delivers a tool to specify approximately which LS-coupling term symbol belongs energetically to which jj-coupling term symbol.
I tested the program by comparing the results with results in Bransden and Joachain (2003) and several papers (Tuttle, 1967; Doggett and Sutcliffe, 1998; Rubio and Perez, 1986; Orofino and Faria, 2010; Gorman, 1973; Gauerke and Campbell, 1994). Although these papers do not deal with the parental histories, a comparison with them was useful, because in all the cases the same values for J occurred. Those values occurred as many times as my program produces another parental history. Thus the results the program produces cannot be too wrong. Additionally both coupling schemes in program yield the same amount of terms for each J in all examined cases. The comparison in means of J was chosen, because it is the only good quantum number of the whole system that for both coupling schemes.
As mentioned in the theory section the LS-coupling term symbols are sorted by means

of Hund's rules. To be able to assign the term symbols unambiguously one has to sort the jj-coupling term symbols with a parallel set of rules. It seems that there are no rules for an qualitative sorting of jj-coupling term symbols. Thus one has to calculate numerically their energies and sort them quantitatively. In an extensive review of the literature I was not able to find such rules. This is probably a result of the tendency of nearly every author to mention only LS-coupling and if mentioning jj-coupling they reduced their explanations to the bare minimum – with the exception of Bransden and Joachain (2003), Condon and Shortley (1970), Condon and Odabasi (1980), but there were no sorting patterns. Additionally, the results of the program showed that even if there were such rules it would be very difficult or even impossible to assign by taking J into account except for some easy configurations that also could be carried out by hand.
There is another problem that has not been mentioned yet. The two parental histories of two term symbols in the two different schemes cannot be consistent in this work, because in LS-coupling the program couples everything in a shell (same n) and then all those shells successively to $\vec{L}$ and $\vec{S}$, but for the jj-coupling term symbols the program couples everything in a jj-subshell (same n_i,l_i,j_i). So the electrons do not just couple differently (LS- versus jj-coupling), they couple differently partitioned in different groups (shell versus jj-subshells). Thus those both histories are not consistent. This inconsistency was accepted, because FAC (**??**) andPalmeri et al. (2008) have the same problem of inconsistency in relation to each other and if one wants to compare their results, one has to adopt their way of coupling. Thus my program has to speak their language to fulfil its intention, namely a comparison of those codes.
The whole theory of Racah, Wigner and Weyl using constructs as Wigner-d-functions or Racah's 3-j-symbols was not examined because it would have exceeded the available time for a Bachelor thesis, but maybe there an answer or hint could be found for further theoretical investigations (Condon and Shortley, 1970; Condon and Odabasi, 1980; Biedenharn, 1981).
Finally I hope – and further investigations will show or not – that my work obeys the essence of the quote by Richard Hamming.

Erklärung

Mit meiner Unterschrift versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäßaus anderen Quellen übernommen wurden, so gut als möglich als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Universität oder keinem anderen Institut vorgelegt wurde.

Erlangen, den 23.10.2012

Bibliography

A. Altland and B. Simons. *Condensed Matter Field Theory*. Cambridge University Press, 2006. ISBN 9780521845083.

N.W. Ashcroft and N.D. Mermin. *Solid state physics*. Science: Physics. Saunders College, 1976. ISBN 9780030493461.

N. R. Badnell. A Breit-Pauli distorted wave implementation for AUTOSTRUCTURE. 182:1528–1535, July 2011. doi: 10.1016/j.cpc.2011.03.023.

A. Bar-Shalom, M. Klapisch, and J. Oreg. HULLAC, an integrated computer package for atomic processes in plasmas. 71:169–188, October 2001. doi: 10.1016/S0022-4073(01)00066-8.

L Biedenharn. *Angular momentum in quantum physics : theory and application*. Addison-Wesley Pub. Co., Advanced Book Program, Reading, Mass, 1981. ISBN 0201135078.

B. H. Bransden and C. J. Joachain. *Physics of Atoms and Molecules*. Prentice Hall, Inc., 2003.

B.W. Carroll and D.A. Ostlie. *An introduction to modern astrophysics*. Pearson Addison-Wesley, 2007. ISBN 9780805304022.

Edward Condon and Halis Odabasi. *Atomic Structure*. 1980.

Edward Condon and G. H. Shortley. *The Theory of Atomic Spectra*. 1970.

J.F. Cornwell. *Group Theory in Physics: An Introduction*. Techniques of Physics. Elsevier Science, 1997. ISBN 9780121898007.

Robert D. Cowan. *The Theory of Atomic Structure and Spectra*. University of California Press Berkeley Los Angelos London, 1981.

W. Demtröder. *Experimentalphysik: Atome, Moleküle, Festkörper*. Number Bd. 3 in Springer-Lehrbuch. Springer, 2007. ISBN 9783540337959.

P. A. M. Dirac. The Quantum Theory of the Electron. *Proc. R. Soc. Lond. A*, 117:610, 1928. doi: 10.1098/rspa.1928.0023.

Graham Doggett and Brian Sutcliffe. A modern approach to l-s coupling in the theory of atomic spectra. *J Chem Ed*, 75(1):110, 1998.

Robert Eisberg. *Quantum physics of atoms, molecules, solids, nuclei, and particles*. Wiley, New York, 1985. ISBN 9780471873730.

Thomas Engel. *Physical chemistry*. Pearson Benjamin Cummings, San Francisco, 2006. ISBN 080533842X.

Richard Feynman, Robert B. Leighton, and Matthew L. Sands. *The Feynman Lectures on Physics*. Addison-Wesley, Munich, 1963. 3 volumes.

R.P. Feynman. *QED: The Strange Theory of Light and Matter. (Alix G. Mautner Memorial Lectures)*. Alix G. Mautner Memorial Lectures. Princeton University Press, 1986. ISBN 9780691083889.

T. Fließbach. *Quantenmechanik*. BI-Wiss.-Verlag, 1991. ISBN 9783411149711.

Harald Friedrich. *Theoretical Atomic Physics*. Springer-Verlag Berlin Heidelberg New York, 1990.

Ensign Steven J. Gauerke and Mark L. Campbell. A simple, systmatic method for determining j levels for jj coupling. *Journal of Chemical Education*, 71(6):457, 1994. doi: 10.1021/ed071p457. URL `http://pubs.acs.org/doi/abs/10.1021/ed071p457`.

Mel Gorman. Rules for writing ground state russell-saunders symbols. *Journal of Chemical Education*, 50(3):189, 1973. doi: 10.1021/ed050p189. URL `http://pubs.acs.org/doi/abs/10.1021/ed050p189`.

M. F. Gu. Indirect X-Ray Line-Formation Processes in Iron L-Shell Ions. 582:1241–1250, January 2003. doi: 10.1086/344745.

M. F. Gu. Dielectronic Recombination Rate Coefficients of Na-like Ions from Mg II to Zn XX Forming Mg-like Systems. 153:389–393, July 2004. doi: 10.1086/421328.

C. W. Haigh. The theory of atomic spectroscopy: jj coupling, intermediate coupling, and configuration interaction. *Journal of Chemical Education*, 72(3):206, 1995. doi: 10.1021/ed072p206. URL `http://pubs.acs.org/doi/abs/10.1021/ed072p206`.

Natalie Hell. Laboratory astrophysics: Investigating the mystery of low charge states of si and s in the hmxb cyg x-1. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2012.

Gerhard Herzberg. *Atomic spectra and atomic structure*. Prentice Hall, Inc., 1937.

Friedrich Hund. Zur Deutung verwickelter Spektren, insbesondere der Elemente Scandium bis Nickel. June 1925. URL `http://www.springerlink.com/content/v64t7r610k67k731/`.

R. Jagannathan. On generalized Clifford algebras and their physical applications. *ArXiv e-prints*, May 2010.

L. D. Landau and E. M. Lifshitz. *Quantum mechanics, Vol. 3*. Course of theoretical physics / by L. D. Landau and E. M. Lifshitz, in 7 Vol. Butterworth-Heinemann, 2 edition, January 1987. ISBN 0750627670.

Hugo Orofino and Roberto B. Faria. Obtaining the electron angular momentum coupling spectroscopic terms, jj. *Journal of Chemical Education*, 87(12):1451–1454, 2010. doi: 10.1021/ed1004245. URL `http://pubs.acs.org/doi/abs/10.1021/ed1004245`.

P. Palmeri, P. Quinet, C. Mendoza, M. A. Bautista, J. García, and T. R. Kallman. Radiative and Auger Decay of K-Vacancy Levels in the Ne, Mg, Si, S, Ar, and Ca Isonuclear Sequences. 177:408–416, July 2008. doi: 10.1086/587804.

M.E. Peskin and D.V. Schroeder. *An Introduction To Quantum Field Theory*. Advanced Book Program. Westview Press, 1995. ISBN 9780201503975.

A.I.M. Rae. *Quantum Mechanics*. Institute of Physics Pub., 2002. ISBN 9780750308397.

E. Rebhan. *Theoretische Physik: Relativistische Quantenmechanik, Quantenfeldtheorie und Elementarteilchentheorie*. Number Bd. 4. Spektrum Akademischer Verlag, 2010. ISBN 9783827426024.

J. Reinhold. *Quantentheorie der Moleküle: Eine Einführung*. Studienb Cher Chemie. Teubner, 2006. ISBN 9783835100374.

J. Rubio and J. J. Perez. Energy levels in the jj coupling scheme. *Journal of Chemical Education*, 63(6):476, 1986. doi: 10.1021/ed063p476. URL `http://pubs.acs.org/doi/abs/10.1021/ed063p476`.

Konrad Rudolph. The 'minted' package: Highlighted source code in latex. `http://www.test.org/doe/`, 2011.

K.A. Semendjajew. *Taschenbuch der Mathematik*. Deutsch, 2008. ISBN 9783817120079.

G. Strang, M. Krieger, and K. Lippert. *Wissenschaftliches Rechnen*. Springer-Lehrbuch Masterclass. Springer, 2010. ISBN 9783540784944.

E. R. Tuttle. Terms Obtained from Configurations of Equivalent Electrons. *American Journal of Physics*, 35:26–29, January 1967. doi: 10.1119/1.1973853.

Paul E. S. Wormer. Angular momentum theory. `http://www.theochem.ru.nl/~pwormer/teachmat/angmom.pdf`, 2012.

A. Example output

This is the .txt FAC output of the program for the example in the usage section:

2p-1(1)1 3s+1(1)0
2p-1(1)1 3s+1(1)2
2p+1(3)3 3s+1(1)2
2p+1(3)3 3s+1(1)4
2s+1(1)1 3s+1(1)0
2s+1(1)1 3s+1(1)2
2s+1(1)1 2p-1(1)0 2p+1(3)3 3s+1(1)2
2s+1(1)1 2p-1(1)0 2p+1(3)3 3s+1(1)4
2s+1(1)1 2p-1(1)2 2p+1(3)1 3s+1(1)0
2s+1(1)1 2p-1(1)2 2p+1(3)1 3s+1(1)2
2s+1(1)1 2p-1(1)2 2p+1(3)3 3s+1(1)2
2s+1(1)1 2p-1(1)2 2p+1(3)3 3s+1(1)4
2s+1(1)1 2p-1(1)2 2p+1(3)5 3s+1(1)4
2s+1(1)1 2p-1(1)2 2p+1(3)5 3s+1(1)6
2s+1(1)1 2p+2(0)1 3s+1(1)0
2s+1(1)1 2p+2(0)1 3s+1(1)2
2s+1(1)1 2p+2(4)3 3s+1(1)2
2s+1(1)1 2p+2(4)3 3s+1(1)4
2s+1(1)1 2p+2(4)5 3s+1(1)4
2s+1(1)1 2p+2(4)5 3s+1(1)6
2p+1(3)3 3s+1(1)2
2p+1(3)3 3s+1(1)4
2p-1(1)1 2p+2(0)1 3s+1(1)0
2p-1(1)1 2p+2(0)1 3s+1(1)2
2p-1(1)1 2p+2(4)3 3s+1(1)2
2p-1(1)1 2p+2(4)3 3s+1(1)4
2p-1(1)1 2p+2(4)5 3s+1(1)4
2p-1(1)1 2p+2(4)5 3s+1(1)6
2p+3(3)3 3s+1(1)2
2p+3(3)3 3s+1(1)4

On the next page the .pdf output of the program for the same example is given.

L-S- and jj-coupling term symbols:

Electron orbital configurations($n_1 l_1^{x_1} \ldots n_m l_m^{x_m}$): $1s^2 2s^2 2p^1 3s^1$, $1s^2 2s^1 2p^2 3s^1$, $1s^2 2p^3 3s^1$

L-S-coupling	jj-coupling
$(((1s^2(^1S)\ 2s^2 2p^1(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_2$	$((([1s+]^2_0[2s+]^2_0)_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_0$
$(((1s^2(^1S)\ 2s^2 2p^1(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_1$	$((([1s+]^2_0[2s+]^2_0)_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^2 2p^1(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_0$	$((([1s+]^2_0[2s+]^2_0)_0[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^2 2p^1(^2P))\ ^2P)\ 3s^1(^2S))\ ^1P_1$	$((([1s+]^2_0[2s+]^2_0)_0[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2s^1 2p^2(^4P))\ ^4P)\ 3s^1(^2S))\ ^5P_3$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^2_0)_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_0$
$(((1s^2(^1S)\ 2s^1 2p^2(^4P))\ ^4P)\ 3s^1(^2S))\ ^5P_2$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^2_0)_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^1 2p^2(^4P))\ ^4P)\ 3s^1(^2S))\ ^5P_1$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_0[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^1 2p^2(^2D))\ ^2D)\ 3s^1(^2S))\ ^3D_3$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_0[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2s^1 2p^2(^2D))\ ^2D)\ 3s^1(^2S))\ ^3D_2$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_1[2p+]^1_{\frac{3}{2}})_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_0$
$(((1s^2(^1S)\ 2s^1 2p^2(^2D))\ ^2D)\ 3s^1(^2S))\ ^3D_1$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_1[2p+]^1_{\frac{3}{2}})_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^1 2p^2(^4P))\ ^4P)\ 3s^1(^2S))\ ^3P_2$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_1[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^1 2p^2(^4P))\ ^4P)\ 3s^1(^2S))\ ^3P_1$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_1[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2s^1 2p^2(^4P))\ ^4P)\ 3s^1(^2S))\ ^3P_0$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_1[2p+]^1_{\frac{3}{2}})_{\frac{5}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2s^1 2p^2(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_2$	$(((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p-]^1_{\frac{1}{2}})_1[2p+]^1_{\frac{3}{2}})_{\frac{5}{2}}[3s+]^1_{\frac{1}{2}})_3$
$(((1s^2(^1S)\ 2s^1 2p^2(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_1$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_0)_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_0$
$(((1s^2(^1S)\ 2s^1 2p^2(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_0$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_0)_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^1 2p^2(^2S))\ ^2S)\ 3s^1(^2S))\ ^3S_1$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2s^1 2p^2(^2D))\ ^2D)\ 3s^1(^2S))\ ^1D_2$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2s^1 2p^2(^2P))\ ^2P)\ 3s^1(^2S))\ ^1P_1$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{5}{2}}[3s+]^1_{\frac{1}{2}})_2$

$(((1s^2(^1S)\ 2s^1 2p^2(^2S))\ ^2S)\ 3s^1(^2S))\ ^1S_0$	$((([1s+]^2_0[2s+]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{5}{2}}[3s+]^1_{\frac{1}{2}})_3$
$(((1s^2(^1S)\ 2p^3(^4S))\ ^4S)\ 3s^1(^2S))\ ^5S_2$	$((([1s+]^2_0[2p-]^2_0)_0[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2p^3(^2D))\ ^2D)\ 3s^1(^2S))\ ^3D_3$	$((([1s+]^2_0[2p-]^2_0)_0[2p+]^1_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2p^3(^2D))\ ^2D)\ 3s^1(^2S))\ ^3D_2$	$((([1s+]^2_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_0)_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_0$
$(((1s^2(^1S)\ 2p^3(^2D))\ ^2D)\ 3s^1(^2S))\ ^3D_1$	$((([1s+]^2_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_0)_{\frac{1}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2p^3(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_2$	$((([1s+]^2_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2p^3(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_1$	$((([1s+]^2_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2p^3(^2P))\ ^2P)\ 3s^1(^2S))\ ^3P_0$	$((([1s+]^2_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{5}{2}}[3s+]^1_{\frac{1}{2}})_2$
$(((1s^2(^1S)\ 2p^3(^4S))\ ^4S)\ 3s^1(^2S))\ ^3S_1$	$((([1s+]^2_0[2p-]^1_{\frac{1}{2}})_{\frac{1}{2}}[2p+]^2_2)_{\frac{5}{2}}[3s+]^1_{\frac{1}{2}})_3$
$(((1s^2(^1S)\ 2p^3(^2D))\ ^2D)\ 3s^1(^2S))\ ^1D_2$	$(([1s+]^2_0[2p+]^3_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_1$
$(((1s^2(^1S)\ 2p^3(^2P))\ ^2P)\ 3s^1(^2S))\ ^1P_1$	$(([1s+]^2_0[2p+]^3_{\frac{3}{2}})_{\frac{3}{2}}[3s+]^1_{\frac{1}{2}})_2$

Sorting and nomenclature: In the L-S-table the term symbols are sorted by means of the empiric Hund' rules in increasing order. Hence the ground state is on top. The jj term symbols are sorted with respect to the maximal vectorial sum of the total angular momenta of the electrons, not with respect to the value of J assigned in the lower right index.
L and S are denoting the total angular momentum and spin, while j_i and J are refering to the total angular momentum of the ith electron and of the whole configuration. There are x_i electron in the orbital with main quantum number n_i and angular momentum quantum number l_i that is given in the conventional spectroscopic form: $s, p, d, f, g, h, \ldots$ for $0, 1, 2, 3, 4, 5, \ldots$.
More precisely the L-S-coupling term symbols are in the form:

$$(\cdots((n_1 l_1^{x_1}(^{2S_1+1}L_1)\ n_2 l_2^{x_2}(^{S_2+1}L_2))\ ^{2S_{1\&2}+1}L_{1\&2})\cdots n_m l_m^{x_m}(^{S_m+1}L_m))\ ^{2S+1}L_J^{(\circ)}$$

and the jj-coupling term symbols are in the following form:

$$(\cdots((n_1 l_1^{x_1}{}_{j_1,\cdots,j_{x_1}})_{j_{s1}}(n_2 l_2^{x_2}{}_{j_{x_1+1},\cdots,j_{x_1+x_2}})_{j_{s2}})_{j_{1\&2}}\cdots(n_m l_m^{x_m}{}_{j_{N-x_m+1},\cdots,j_N})_{j_{sm}})_J^{(\circ)}.$$

B. Code

In this appendix the code – higlighted via *minted* by Konrad Rudolph Rudolph (2011) – of the script that is the main part of this thesis:

```
#! /usr/bin/env python

'''
This is a python script for the production and mapping
(as good as possible with Hund's rules)
of L-S- and jj-coupling term symbols.
This script was written by Alexander Laska,
Friedrich-Alexander-University Nuremberg-Erlangen,
as part of his bachelor thesis.
It was tested using some easy examples and is
in no sense a complete and faultless answer
to the question of mapping coupling term symbols.
Its functions and algorithms are described in the thesis.
'''

import copy
import itertools
import math
import operator


'''
Enable FAC output (if facbool is True the outputs both
in the .tex and the .txt file are like the output
of the atomic code FAC by Ming Gu, if it is False
the output in the .pdf keeps mind
of a subshell-subshell-su... successive coupling
and the also the .txt file, but here with
a FAC based style). Normally the intended use
of the program expects facbool = True.
The input is always in the style of FAC input.
'''

facbool = True
```

```
'''
Convention for the translation between
numbers (indices) and letters
expressing angular momentum eigenvalues
'''

confnom = ['s', 'p', 'd', 'f', 'g', 'h', 'i', 'k',
'l', 'm', 'n', 'o', 'q', 'r', 't', 'u',
'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'e']
lsconv  = ['S', 'P', 'D', 'F', 'G', 'H', 'I', 'K',
'L', 'M', 'N', 'O', 'Q', 'R', 'T', 'U',
'V', 'W', 'X', 'Y', 'Z', 'A', 'B', 'C', 'E']


# Open a text document to write the FAC output into it

txtout = open('hclsvsjj.txt', 'w')


'''
Improved (FAC) version of input interpretation
(2s1, * and [x,y,...] notation)
'''

def read_orbconf_fac():

    # Reading the input and handling it to the orbstr

    orbstr = str(raw_input("Enter the configuration (FAC notation): "))

    # Splitting the input at spaces and the other meta symbols

    orbconfttt = orbstr.split(' ')
    orbconftt = []
    for y in orbconfttt:
        for x in confnom:
            if x in y and '[' not in y:
                orbconftt.append(list(y.rpartition(x)))
        if '*' in y:
            orbconftt.append(list(y.rpartition('*')))
        if '[' in y:
            orbconftt.append(list(y.rpartition('[')))
    orbconft = []
    for y in orbconftt:
        if ']' in y[-1]:
            a = y[-1].rpartition(']')
```

```
            b = list([y[0],a[0],a[2]])
            orbconft.append(b)
        else:
            orbconft.append(y)

# Producing all possible fillings of subshells ([])

orbconf = []
def fillings(n,l,x):
    x = int(x)
    temp = []
    for y in range(len(l)):
        temp.append(list(range(x + 1)))
    temp = list(itertools.product(*temp))
    output = []
    for y in temp:
        if sum(y) == x:
            output.append(list(y))
    for y in range(len(l)):
        z = 0
        while z < len(output):
            if output[z][y] > 2*(2*int(l[y]) + 1):
                output.remove(output[z])
            else:
                z += 1
    tempout = []
    for y in output:
        tempout.append(list(zip(l,y)))
    output = []
    for y in range(len(tempout)):
        temptemp = []
        for z in range(len(tempout[y])):
            if tempout[y][z][1] != 0:
                temp = [int(n),tempout[y][z][0],tempout[y][z][1]]
                temptemp.append(temp)
        output.append(temptemp)

    return output

# Producing all possible fillings of subshells (*)

run = []
for y in orbconft:
    if y[1] == '*':
        l = range(int(y[0]))
        run.append(fillings(y[0],l,y[2]))
```

```
        if len(y[1]) > 1:
            temp = str(y[1])
            l = []
            while len(temp) > 1:
                l.append(confnom.index(temp[0]))
                temp = str(temp[2:])
            l.append(confnom.index(temp[0]))
            run.append(fillings(y[0],l,y[2]))
        if len(y[1]) == 1 and y[1] != '*':
            run.append(fillings(y[0],[confnom.index(y[1][0])],y[2]))

    # Stiching together all combinations for the individual subshells

    runoutput = list(itertools.product(*run))
    output = []
    for y in range(len(runoutput)):
        output.append(list(runoutput[y]))
    put = []
    for y in output:
        put.append(list(flatten(y)))

    return put


# Latex all jj terms

def texalljj(L, orb, facbool):

    output = []
    sortjj = []
    for x in range(len(L)):
        a = texjjt(L[x], orb, facbool)
        output.append(a[0])
        sortjj.append(a[1])
        txtout.write(str(a[2]))
        txtout.write('\n')
    c = zip(output,sortjj)
    c.sort(key=lambda x: x[1],reverse=True)
    b = []
    e = []
    for x in range(len(c)):
        b.append(c[x][0])
        e.append(c[x][1])
    output = b
    return output
```

```
# Producing fac compatible orbital configurations (nl+ and nl-,l != 0)

def orbfac(L):

    output = []
    facoutput = []
    for x in L:
        if x[1] == 0:
            facoutput.append(str(x[0]) + confnom[x[1]] + '+')
            output.append(x)
        else:
            facoutput.append(str(x[0]) + confnom[x[1]] + '-')
            output.append(x)
            facoutput.append(str(x[0]) + confnom[x[1]] + '+')
            output.append(x)
    return facoutput


# Latexable version of one jj coupling term symbol producing function

def texjjt(L, orb, facbool):

    output = []
    sortjj = 0
    fac = ''

    # For the case of just one part. orb. conf.

    if len(orb) == 1:

        '''
        First the orbital configuration is appended
        and then the angular momenta
        '''

        print L

        if facbool is False:
            string = r'$(' + str(orb[0]) + r'_{'
        else:
            string = r'$[' + str(orb[0][0:3]) + r']^{'
        if facbool is False:
            fac += orb[0][0]
            fac += orb[0][1]
        else:
```

```
    fac += orb[0][:3]
factemp = []
factemptemp = []
if facbool is False:
    if int(2*L[0][-1]) % 2 == 0:
        string += str(int(L[0][-1]))
        factemptemp.append(int(2*L[0][-1]))
        sortjj += L[0][-1]
    else:
        string += r'\frac{'
        string += str(int(2*L[0][-1]))
        factemptemp.append(int(2*L[0][-1]))
        string += r'}{2}'
        sortjj += L[0][-1]
    if len(L) > 2:
        for x in range(1, len(L) - 1):
            string += r','
            if int(2*L[x][-1]) % 2 == 0:
                string += str(int(L[x][-1]))
                factemptemp.append(int(2*L[x][-1]))
                sortjj += L[x][-1]
            else:
                string += r'\frac{'
                string += str(int(2*L[x][-1]))
                factemptemp.append(int(2*L[x][-1]))
                string += r'}{2}'
                sortjj += L[x][-1]
    factemp.append([max(factemptemp),'+',
    factemptemp.count(max(factemptemp))])
    factemp.append([min(factemptemp),'-',
    factemptemp.count(min(factemptemp))])
    if factemp[0][2] >= factemp[1][2]:
        fac += str(factemp[0][1])
        fac += str(factemp[0][2])
    else:
        fac += str(factemp[1][1])
        fac += str(factemp[1][2])
    string += r'}'

# Appending the local total J

else:
    fac += str(len(L) - 1)
    string += str(len(L) - 1)
    string += r'}'
if int(2*L[-1]) % 2 == 0:
```

```
            string += str(int(L[-1])) + '}'
            fac += '('
            fac += str(int(2*L[-1]))
            fac += ')'
            fac += str(int(2*L[-1]))
        else:
            string += r'\frac{'
            string += str(int(2*L[-1]))
            string += r'}{2}}'
            fac += '('
            fac += str(int(2*L[-1]))
            fac += ')'
            fac += str(int(2*L[-1]))

    else:

        '''
        Flattening the deeply nested structure
        containing the results of the successive coupling
        '''

        temp = []
        copy = list(L)
        for x in range(len(orb)):
            if x < len(orb) - 1:
                temp.append(copy[-1])
                temp.append(copy[-2][-1])
                copy = list(copy[-2][-2])
            else:
                temp.append(copy)
        temp.reverse()
        L = list(temp)

        '''
        First the orbital configuration is appended
        and then the angular momenta
        '''

        string = '$'
        for x in range(len(orb) - 1):
            string += r'('

        if facbool is False:
            string += str(orb[0]) + r'_{'
        else:
            if L[0][0][2] != 0:
```

```
            string += r'[' + str(orb[0][0:3]) + r']^{'
        else:
            string += r'[' + str(orb[0][0:2]) + r'0]^{'
    if facbool is False:
        fac += orb[0][0]
        fac += orb[0][1]
    else:
        if L[0][0][2] != 0:
            fac += orb[0][:3]
        else:
            fac += orb[0][:2]
            fac += '0'
    factemp = []
    factemptemp = []
    if facbool is False:
        if int(2*L[0][0][-1]) % 2 == 0:
            string += str(int(L[0][0][-1]))
            sortjj += L[0][0][-1]
            factemptemp.append(int(2*L[0][0][-1]))
        else:
            string += r'\frac{'
            string += str(int(2*L[0][0][-1]))
            string += r'}{2}'
            sortjj += L[0][0][-1]
            factemptemp.append(int(2*L[0][0][-1]))
        if len(L[0]) > 2:
            for x in range(1, len(L[0]) - 1):
                string += r','
                if int(2*L[0][x][-1]) % 2 == 0:
                    string += str(int(L[0][x][-1]))
                    sortjj += L[0][x][-1]
                    factemptemp.append(int(2*L[0][x][-1]))
                else:
                    string += r'\frac{'
                    string += str(int(2*L[0][x][-1]))
                    string += r'}{2}'
                    sortjj += L[0][x][-1]
                    factemptemp.append(int(2*L[0][x][-1]))

    # Appending the local total J

        string += r'})_{'
        factemp.append([max(factemptemp),
        '+',factemptemp.count(max(factemptemp))])
        factemp.append([min(factemptemp),
        '-',factemptemp.count(min(factemptemp))])
```

```
    if factemp[0][2] >= factemp[1][2]:
        fac += str(factemp[0][1])
        fac += str(factemp[0][2])
    else:
        fac += str(factemp[1][1])
        fac += str(factemp[1][2])
else:
    fac += str(len(L[0]) - 1)
    string += str(len(L[0]) - 1)
    string += r'}_{'
if int(2*L[0][-1]) % 2 == 0:
    string += str(int(L[0][-1]))
    fac += '('
    fac += str(int(2*L[0][-1]))
    fac += ')'
    fac += str(int(2*L[0][-1]))
else:
    string += r'\frac{'
    string += str(int(2*L[0][-1]))
    string += r'}{2}'
    fac += '('
    fac += str(int(2*L[0][-1]))
    fac += ')'
    fac += str(int(2*L[0][-1]))
if facbool is False:
    string += r'} ('
else:
    string += r'} '
fac += ' '

# Appending the local total J

if facbool is False:
    string += str(orb[1]) + r'_{'
else:
    if L[1][0][2] != 0:
        string += r'[' + str(orb[1][0:3]) + r']^{'
    else:
        string += r'[' + str(orb[1][0:2]) + r'0]^{'
if facbool is False:
    fac += orb[1][0]
    fac += orb[1][1]
else:
    if L[1][0][2] != 0:
        fac += orb[1][:3]
    else:
```

```
            fac += orb[1][:2]
            fac += '0'
    factemp = []
    factemptemp = []
    if facbool is False:
        if int(2*L[1][0][-1]) % 2 == 0:
            string += str(int(L[1][0][-1]))
            sortjj += L[1][0][-1]
            factemptemp.append(int(2*L[1][0][-1]))
        else:
            string += r'\frac{'
            string += str(int(2*L[1][0][-1]))
            string += r'}{2}'
            sortjj += L[1][0][-1]
            factemptemp.append(int(2*L[1][0][-1]))
        if len(L[1]) > 2:
            for x in range(1, len(L[1]) - 1):
                string += r','
                if int(2*L[1][x][-1]) % 2 == 0:
                    string += str(int(L[1][x][-1]))
                    sortjj += L[1][x][-1]
                    factemptemp.append(int(2*L[1][x][-1]))
                else:
                    string += r'\frac{'
                    string += str(int(2*L[1][x][-1]))
                    string += r'}{2}'
                    sortjj += L[1][x][-1]
                    factemptemp.append(int(2*L[1][x][-1]))

    # Appending the local total J

        string += r'})_{'
        factemp.append([max(factemptemp),'+',
        factemptemp.count(max(factemptemp))])
        factemp.append([min(factemptemp),'-',
        factemptemp.count(min(factemptemp))])
        if factemp[0][2] >= factemp[1][2]:
            fac += str(factemp[0][1])
            fac += str(factemp[0][2])
        else:
            fac += str(factemp[1][1])
            fac += str(factemp[1][2])
    else:
        fac += str(len(L[1]) - 1)
        string += str(len(L[1]) - 1)
        string += r'}_{'
```

```
if int(2*L[1][-1]) % 2 == 0:
    string += str(int(L[1][-1]))
    fac += '('
    fac += str(int(2*L[1][-1]))
    fac += ')'
else:
    string += r'\frac{'
    string += str(int(2*L[1][-1]))
    string += r'}{2}'
    fac += '('
    fac += str(int(2*L[1][-1]))
    fac += ')'

string += r'})_{'

if int(2*L[2]) % 2 == 0:
    string += str(int(L[2])) + '}'
    fac += str(int(2*L[2]))
else:
    string += r'\frac{'
    string += str(int(2*L[2]))
    string += r'}{2}}'
    fac += str(int(2*L[2]))
fac += ' '

a = 3
for y in range(2, len(orb)):

    '''
    First the orbital configuration
    is appended and then the angular momenta
    '''

    if facbool is False:
        string += r' (' + str(orb[y]) + r'_{'
    else:
        if L[a][0][2] != 0:
            string += r'[' + str(orb[y][0:3]) + r']^{'
        else:
            string += r'[' + str(orb[y][0:2]) + r'0]^{'
    if facbool is False:
        fac += orb[y][0]
        fac += orb[y][1]
    else:
        if L[a][0][2] != 0:
            fac += orb[y][:3]
```

```
    else:
        fac += orb[y][:2]
        fac += '0'
factemp = []
factemptemp = []
if facbool is False:
    if int(2*L[a][0][-1]) % 2 == 0:
        string += str(int(L[a][0][-1]))
        sortjj += L[a][0][-1]
        factemptemp.append(int(2*L[a][0][-1]))
    else:
        string += r'\frac{'
        string += str(int(2*L[a][0][-1]))
        string += r'}{2}'
        sortjj += L[a][0][-1]
        factemptemp.append(int(2*L[a][0][-1]))
    if len(L[a]) > 2:
        for x in range(1, len(L[a]) - 1):
            string += r','
            if int(2*L[a][x][-1]) % 2 == 0:
                string += str(int(L[a][x][-1]))
                sortjj += L[a][x][-1]
                factemptemp.append(int(2*L[a][x][-1]))
            else:
                string += r'\frac{'
                string += str(int(2*L[a][x][-1]))
                string += r'}{2}'
                sortjj += L[a][x][-1]
                factemptemp.append(int(2*L[a][x][-1]))

    '''
    The total angular momentum for all
    equivalent electrons has to be appended last
    '''

    string += r'})_{'
    factemp.append([max(factemptemp),'+',
    factemptemp.count(max(factemptemp))])
    factemp.append([min(factemptemp),'-',
    factemptemp.count(min(factemptemp))])
    if factemp[0][2] >= factemp[1][2]:
        fac += str(factemp[0][1])
        fac += str(factemp[0][2])
    else:
        fac += str(factemp[1][1])
        fac += str(factemp[1][2])
```

```
            else:
                fac += str(len(L[a]) - 1)
                string += str(len(L[a]) - 1)
                string += r'}_{'
            if int(2*L[a][-1]) % 2 == 0:
                string += str(int(L[a][-1])) + '}'
                fac += '('
                fac += str(int(2*L[a][-1]))
                fac += ')'
            else:
                string += r'\frac{'
                string += str(int(2*L[a][-1]))
                string += r'}{2}}'
                fac += '('
                fac += str(int(2*L[a][-1]))
                fac += ')'

            a += 1

            '''
            The whole system total angular momentum
            for all has to be appended finally
            '''

            string += r')_{'
            if int(2*L[a]) % 2 == 0:
                string += str(int(L[a])) + '}'
                fac += str(int(2*L[a]))
            else:
                string += r'\frac{'
                string += str(int(2*L[a]))
                string += r'}{2}}'
                fac += str(int(2*L[a]))
            fac += ' '
            a += 1

    '''
    Filtering out all full subshells
    of inequivalent electrons
    that are supressed by the FAC notation
    '''

    dec = fac.split()
    c = 0
    d = len(dec)
    while len(dec) > 1 and c < d:
```

```
    if dec[c][2] == '0':
        del dec[c]
        d -= 1
    else:
        c += 1
a = 0
b = len(dec)
while len(dec) > 1 and a < b:
    #if dec[a][5] == '0':
    if dec[a][2] == '+':
        spin = 0.5
    else:
        spin = -0.5
    #print dec[a], 2*(confnom.index(dec[a][1]) + spin) + 1
    if str(int((2*(confnom.index(dec[a][1]) + spin) + 1))) == dec[a][3]:
        del dec[a]
        b -= 1
    else:
        a += 1
tempdec = ''
for x in dec:
    tempdec += x
    tempdec += ' '
dec = str(tempdec)
print dec

stringsplit = string.split('[')
a = 1
b = len(stringsplit)
delcount = 0
while len(stringsplit) > 2 and a < b:
    if stringsplit[a][2] == '0':
        del stringsplit[a]
        delcount += 1
        b -= 1
    else:
        a += 1
stringsplit[0] = stringsplit[0][:len(stringsplit)-delcount]
stringim = ''
for x in range(len(stringsplit) - 1):
    stringim += stringsplit[x]
    stringim += '['
stringim += stringsplit[-1]
output = [stringim, sortjj, dec]

return output
```

```
# Successive coupling of still coupled subshells

def sucjjsh(L):

    output = []

    '''
    For all the cases of one, two or more
    different successive couplings are neccessary:
    non, one or the real successive coupling.
    The values to be coupled are handled to lsrange()
    to produce all values from |j_1 - j_2| to
    j_1 + j_2 in steps of size one.
    If there are more then one subshells the result
    of the last coupling is j_1
    and the next subshells J is j_2.
    '''

    if len(L) == 1:
        output = L[0]
        return output
    if len(L) > 1:
        for y in range(len(L[0])):
            for z in range(len(L[1])):
                a = lsrange(L[0][y][-1], L[1][z][-1])
                for x in range(len(a)):
                    output.append([[L[0][y], L[1][z]], a[x]])
    if len(L) == 2:
        return output
    if len(L) > 2:
        for x in range(2, len(L)):
            temp = []
            for y in range(len(output)):
                for z in range(len(L[x])):
                    a = lsrange(output[y][-1], L[x][z][-1])

                    for k in range(len(a)):
                        temp.append([[output[y], L[x][z]], a[k]])
            output = list(temp)

        return output


'''
```

```
Feeding all subshells of equivalent electrons
in to jjterms() to obtain the term symbols
'''

def sucjjfac(L):

    output = []
    for x in L:
        if x[1] == 0:
            output.append([jjterms([x])])
        else:
            temp = jjtermsfac([x])
            output.append(temp)
    new = list(itertools.product(*output))
    output = []
    for x in new:
        output.append(list(x))
    final = []
    for x in output:
        temp = []
        for y in x:
            temp.extend(list(y))
        a = []
        for y in temp:
            b = []
            b.append(y)
            a.append(b)
        final.append(a)

    return final


# Feeding all subshells into jjterm() to obtain

def sucjj(L):

    output = []
    for x in range(len(L)):
        output.append(jjterms([L[x]]))

    return output


# Producing the  whole LS term symbol list in a latex ready state

def LStex(L, orb):
```

```
    output = []
    for x in range(len(L)):
        output.append(Translatetermsymbol(L[x], orb))

    return output


# Produce a single final term symbol final latex string

def Translatetermsymbol(L, orb):

    output = r'$'
    count = len(orb) - 1
    for x in range(count):
        output += r'('
    if len(L) == 1:
        output += str(orb[0])
        output += r'\ '
        output += str(L[0])
        output += r'$'
        return output
    else:
        output += r'('
        output += str(orb[0])
        output += r'('
        output += str(L[0])
        a = 0
        for x in range(1, count + 1):
            output += r')\ '
            output += str(orb[x])
            output += r'('
            a += 1
            output += str(L[a])
            output += r'))\ '
            a += 1
            output += str(L[a])
        return output


# Translate orbconf in a tex form (for jj terms)

def Translateorballjj(L):

    output = []
    L = list(flatten(L))
```

```
    for x in range(len(L)):
        output.append(Translateorb(L[x]))
    return output


# Translate orbconf in a tex form

def Translateorball(L):

    output = []
    for x in range(len(L)):
        string = ''
        #L[x].reverse()
        for y in range(len(L[x])):
            string += str(Translateorb(L[x][y]))
        output.append(string)
    return output


# Translate whole term symbols list

def Translateall(L):

    output = []
    for x in range(len(L)):
        output.append(Translatelist(L[x]))
    return output


# Translate list

def Translatelist(L):

    output = []
    L.reverse()
    for x in range(len(L) - 1):
        output.append(TranslateLS(L[x]))
    output.append(TranslateLSJ(L[-1]))
    return output


# Three functions to translate terms and term symbols into string

def TranslateLSJ(L):

    output  = r'^{' + str(int(2*float(L[0]) + 1))
```

```
    output += '}' + lsconv[L[1]] + r'_{'
    if int(2*L[2]) % 2 == 0:
        output += str(int(L[2])) + '}'
    else:
        output += r'\frac{'
        output += str(int(2*L[2]))
        output += r'}{2}}'
    return output

def TranslateLS(L):

    output = r'^{' + str(int(2*float(L[0]) + 1)) + '}' + lsconv[L[1]]
    return output

def Translateorb(L):

    output = str(L[0]) + confnom[L[1]] + '^{' + str(L[2]) + '}'
    return output


# LShisttex on whole list

def LShisttexwl(L, orb):

    output = []

    for x in range(len(L)):
        output.append(LShisttex(L[x], orb))

    return output


'''
Function that converts the attached
"parents" of a term into latex code
'''

def LShisttex(L, orb):

    output = []
    count = len(orb) - 1
    steps = 0

    temp = list(L)

    if steps == 0 and count > 0:
```

```
        output.append([L[0], L[1], L[2]])
        output.append(L[3][1])
        temp = L[3][0]
        steps += 1
    while steps < count:
        output.append([temp[0], temp[1]])
        output.append(temp[2][1])
        temp = temp[2][0]
        steps += 1
    if steps == count:
        output.append(temp)

    return output


# Function "reading" the depth of a list of lists of litsts of....

depth = lambda L: isinstance(L, list) and max(map(depth, L))+1


'''
Functions deciding whether a shell is more
than half filled (neccessary for fullfilling
the third of Hund's rules)
'''

def halffilled(L):

    if L[-1][2] < 2*(L[-1][1]) + 1:
        return False
    else:
        return True


# Append J to multils using lsrange()

def multiJ(multiin, rev):

    output = []
    for x in range(len(multiin)):
        a = lsrange(multiin[x][0], multiin[x][1])
        if rev == True:
            a.reverse()
        for y in range(len(a)):
            b = [multiin[x][0], multiin[x][1], a[y]]
            if len(multiin[x]) > 2:
```

```
                b.append(multiin[x][2])
            output.append(b)
    return output


# Successive use of mixing

def sucmixing(multif):

    if len(multif) == 1:
        multif = list(flatten(multif))
        multif.sort(key=comparefield1, reverse=True)
        multif.sort(key=comparefield0, reverse=True)
        return multif
    else:
        output = []
        output = mixing(multif[0], multif[1], 0)
        if len(multif) > 2:
            for x in range(2, len(multif)):
                output = mixing(output, multif[x], 0)
        output.sort(key=comparefield1, reverse=True)
        output.sort(key=comparefield0, reverse=True)
        return output


# Mixing function for partially coupled LS terms

def mixing(multi0, multi1, trig):

    multi = [multi0, multi1]

    output = []
    for y in range(len(multi[0])):
        for z in range(len(multi[1])):
            a = lsrange(multi[0][y][0], multi[1][z][0])
            b = lsrange(multi[0][y][1], multi[1][z][1])
            c = []
            d = [multi[0][y], multi[1][z]]
            for k in range(len(a)):
                for l in range(len(b)):
                    c = [a[k], b[l]]
                    c.append(d)
                    output.append(c)
    output.sort(key=comparefield1, reverse=True)
    output.sort(key=comparefield0, reverse=True)
    return output
```

```
# Function for translation of LETTER -> number using lsconv

def translateLN(multi):

    output = []
    for x in range(len(multi)):
        tempoutput = []
        for y in range(len(multi[x])):
            temptempoutput = []
            temptempoutput.append((multi[x][y][0][1] - 1)*0.5)
            for a in range(len(lsconv)):
                if multi[x][y][0][2] == lsconv[a]:
                    temptempoutput.append(a)
            if y == 0:
                tempoutput.append(temptempoutput)
            else:
                if temptempoutput != tempoutput[-1]:
                    tempoutput.append(temptempoutput)
        output.append(tempoutput)
    return output


'''
Paulifilter for a list of tuples of
lists of quantumnumbers in  the form [[n,l,j],mj]
'''

def paulifilter(qnum):
    output = []
    for x in range(len(qnum)):
        a = 1
        for y in range(len(qnum[x])):
            for z in range(y + 1, len(qnum[x])):
                if qnum[x][y] == qnum[x][z]:
                    a = 0
        if a == 1:
            output.append(qnum[x])
    return output


# Removing duplicates in a list of tuples of lists in the outer layer

def unique_items(L):
    output = []
```

```
    for x in range(len(L)):
        L[x] = list(L[x])
        L[x].sort()
    for x in range(len(L)):
        a = 1
        for y in range(x + 1, len(L)):
            if L[x] == L[y]:
                a = 0
        if a == 1:
            output.append(L[x])
    return output


# Function for the coupling values of one set of two j-like values

def lsrange(L, S):

    a = abs(L - S)
    b = L + S
    lsrangeout = []
    while a <= b:
        lsrangeout.append(a)
        a = a + 1
    lsrangeoutput = reversed(lsrangeout)
    return lsrangeout


# Function for reading the orbital configuration

def read_orbconf():

    orbstr = str(raw_input("Enter the configuration: "))
    orblist = orbstr.split(';')

    orbconfttt = []
    for x in orblist:
        orbconfttt.append(x.lstrip('('))

    orbconftt = []
    for x in orbconfttt:
        orbconftt.append(x.replace(')',','))

    orbconft = []
    for x in orbconftt:
        orbconft.append(x.split(','))
```

```
    orbconf = []
    for x in orbconft:
        orbconf.append([int(x[0]),int(x[1]),int(x[2])])

    return orbconf


# Compare two elements

def compareField(field):
    def c(l1,l2):
        return cmp(l1[field], l2[field])
    return c


# Another set of comparing functions

def comparefield0(list):

    return list[0]

def comparefield1(list):

    return list[1]


# Function that generates an n-splittet orbconf list

def imorbconf(orbconf):

    orbconf.sort(compareField(0))
    a = [len(list(subgroup)) for key,
    subgroup in itertools.groupby(orbconf, lambda x: x[0])]
    iorbconf = []
    b = 0
    for x in range(len(a)):
        temp = []
        for y in range(a[x]):
            temp.append(orbconf[b + y])
        temp.sort(key=lambda liste:liste[1])
        iorbconf.append(temp)
        b += a[x]
    return iorbconf


# Function for encoding the Weyl diagrams into mell containng diagrams
```

```
def encode(declist, key):

    enclist = []
    for x in range(len(declist)):
        enclist.append(key[declist[x] - 1])
    return enclist


# Function for adding the m_ell values

def mell_add(weyld):

    Mell = 0
    for x in range(len(weyld[0])):
        Mell = Mell + weyld[0][x]
    for x in range(len(weyld[1])):
        Mell = Mell + weyld[1][x]
    return Mell


# Function that stores two lists into an Latex table

def latex(table, jtable, Parity, iorb):

    header = open('header.txt', 'r')
    foot = open('foot.txt', 'r')
    lstext = open('hclstext.txt', 'r')
    f = open('hclsvsjj.tex', 'w')

    f.write(header.read())

    f.write(str(lstext.read()))

    f.write(r'\textbf{$L$-$S$- and $jj$-coupling term symbols:} \\')
    f.write('\n')
    f.write('\n')
    if len(iorb) == 1:
        f.write(r'Electron orbital configuration')
        f.write(r'$(n_{1}l_{1}^{x_{1}} \dots n_{m}l_{m}^{x_{m}})$: ')
    else:
        f.write(r'Electron orbital configurations')
        f.write(r'$(n_{1}l_{1}^{x_{1}} \dots n_{m}l_{m}^{x_{m}})$: ')

    f.write(r'$')
    for y in range(len(iorb) - 1):
```

```
        for x in range(len(iorb[y])):
            f.write(str(iorb[y][x]))
        f.write(r'$, $')
    for x in range(len(iorb[-1])):
        f.write(str(iorb[-1][x]))
    f.write(r'$\\' + '\n')
    f.write(r'\\' + '\n')
    f.write(r'\begin{longtable}{c|c}' + '\n')
    f.write(r'$L$-$S$-coupling & $jj$-coupling\\' + '\n')
    f.write(r'\hline ')
    f.write(r'\hline')
    f.write(r' \\ ' + '\n')

    for x in range(len(table)):
        for y in range(len(table[x])):
             f.write(table[x][y])
             f.write(r'^{')
             f.write(str(Parity[x][0][0][3]))
             f.write(r'}$ & ')
             f.write(jtable[x][y])
             f.write(r'^{')
             f.write(str(Parity[x][0][0][3]))
             f.write(r'}$\\')
             f.write('\n')

    f.write(foot.read())


'''
Function for filling a column with
every possible filling with numbers from 1 up to n
'''

def fill_column(columnl,n):

    columnlist = []
    prod = list(itertools.product(range(1,n+1), repeat=columnl))
    countl = len(prod) - 1
    while countl >= 0:
        columnlist.append(prod[countl])
        countl = countl - 1
    return columnlist


'''
Function that determines all possible
```

```
total spin values for a configuration like (n,l)x
'''

def spin_list(S):

    t = int(S)*(0.5)
    slist = []
    slist.append(t)
    while t > 0.5:
        t = t - 1
        slist.append(t)
    return slist


# Functions for monotonicity

def strictly_increasing(L):
    return all(x<y for x, y in zip(L, L[1:]))

def strictly_decreasing(L):
    return all(x>y for x, y in zip(L, L[1:]))

def non_increasing(L):
    return all(x>=y for x, y in zip(L, L[1:]))

def non_decreasing(L):
    return all(x<=y for x, y in zip(L, L[1:]))


# L-S-terms belonging to orbconf using the method of Dogget and Sutcliff

def lsterms(table):

    lstext = open('hclstext.txt', 'w')

    # Parity
    lpar = 0
    for x in range(len(table)):
        lpar =+ table[x][2] * table[x][1]
    parity = pow(-1, lpar)
    if parity == -1:
        Parlab = '\circ'
    else:
        Parlab = ' '

    # Step (1)
```

```
tablel = len(table)
count = 0
N = 0
n = 0
while count < tablel:
    N = N + table[count][2]
    n = n + 2* table[count][1] + 1
    count = count + 1

# Step (2)

mell = []
for x in range(len(table)):
    for y in range(2*table[x][1]+1):
        mell.append(y - table[x][1])

# Step (3) & (4)

Ncount = int(N)
protoweyl = []
while Ncount >= N*0.5:
    protoweyl.append(int(Ncount))
    Ncount = Ncount - 1

# Step (5)

# Creates all fillings of Weyl diagrammes

weyl = []
for x in range(len(protoweyl)):
    weyl.append(list(itertools.product(fill_column(protoweyl[x],n),
    fill_column(N - protoweyl[x],n))))

# Sorts out all fillings that are not strictly increasing in a row

tempweyl = []
for x in range(len(weyl)):
    for y in range(len(weyl[x])):
        a = strictly_increasing(list(weyl[x][y][0]))
        b = strictly_increasing(list(weyl[x][y][1]))
        if a and b and len(weyl[x][y][0]) >= len(weyl[x][y][1]):
            tempweyl.append([list(weyl[x][y][0]),list(weyl[x][y][1])])

# Sorts out all fillings that are not nondecreasing
```

```
temptempweyl = []
for x in range(len(tempweyl)):
    det = 1
    if len(tempweyl[x][1]) > 0:
        det = 1
        for y in range(len(tempweyl[x][1])):
            if tempweyl[x][1][y] >= tempweyl[x][0][y]:
                det = det * 1
            else:
                det = det * 0
    if det == 1:
        temptempweyl.append(tempweyl[x])

# Sorts out all Weyl diagram for that the number of electrons is wrong

mell_im = []
count = 1
for x in range(len(table)):
    mell_im.append(range(count, count + 2*table[x][1]+1))
    count = count + 2*table[x][1]+1

weyl = []
for x in range(len(temptempweyl)):
    logic = 1
    for y in range(len(mell_im)):
        count_mell = 0
        for z in range(len(mell_im[y])):
            count_mell = count_mell + temptempweyl[x][0].count(mell_im[y][z])
            count_mell += temptempweyl[x][1].count(mell_im[y][z])
        if count_mell == table[y][2]:
            logic = logic * 1
        else:
            logic = logic * 0
    if logic == 1:
        weyl.append(temptempweyl[x])
    logic = 1

weyl_S = []
weyl_S_enc = []
for x in range(len(protoweyl)):
    weyl_S_temp = []
    weyl_S_enctemp = []
    for y in range(len(weyl)):
        if protoweyl[x] == len(weyl[y][0]):
            weyl_S_temp.append(weyl[y])
            weyl_S_enctemp.append([encode(weyl[y][0], mell),
```

```
            encode(weyl[y][1], mell)])
    if len(weyl_S_temp) != 0:
        weyl_S.append(weyl_S_temp)
        weyl_S_enc.append(weyl_S_enctemp)

# Step (6)

# Add m_ell values

S = []

Mell_for_S = []
for x in range(len(weyl_S_enc)):
    Mell_for_Stemp = []
    S.append((len(weyl_S_enc[x][0][0])-len(weyl_S_enc[x][0][1]))/2.0)
    for y in range(len(weyl_S_enc[x])):
        Mell_for_Stemp.append(mell_add(weyl_S_enc[x][y]))
    Mell_for_S.append(Mell_for_Stemp)

# Find the largest mell and produce the LS terms

Terms = []
Mell_terms = Mell_for_S
for x in range(len(Mell_terms)):
    Mell_terms_temp = Mell_terms[x]
    while Mell_terms_temp != []:
        a = Mell_terms_temp.count(max(Mell_terms_temp))
        b = max(Mell_terms_temp)
        Terms.append([a, b, S[x], parity])
        for y in range(2 * b + 1):
            tempa = 0
            while tempa < a:
                Mell_terms_temp.remove(b - y)
                tempa = tempa + 1

# Identify the ground state using Hund's rule

terms = []
for x in range(len(Terms)):
    terms.append([Terms[x][0], int(2 * Terms[x][2] + 1),
    lsconv[Terms[x][1]], Parlab])

lstext.write('\n')

termsymbols = ls_J_termsymbols(Terms, terms)
```

```
    return termsymbols


# Function that produces the J labeled termsymbols using the terms

def ls_J_termsymbols(Terms, terms):

    termscount = 0
    termsymbolsout = []
    for x in range(len(terms)):
        a = lsrange(Terms[x][1], Terms[x][2])
        termsymbols = terms[x]
        for y in range(len(a)):
            termsymbolsout.append([termsymbols, a[y]])
            termscount += termsymbols[0]

    return termsymbolsout


'''
Function that sorts the LS termsymbols using "Hund's rules",
just the "S maximal" rule, because all other rule are implicite
in the algorthim used producing the terms
'''

def hunds_rules(lslist):

    hundoutput = sorted(lslist, key=lambda S: S[0][1], reverse=True)
    return hundoutput


# Flatten one level of nesting

def flatten(ListOfLists):

    return itertools.chain.from_iterable(ListOfLists)


# jj terms new for fac (works nearly the same as jjterms())

def jjtermsfac(orbconf):

    tempstore = []
    comb = []

    # Brute force generation of all combinations of spin "ups and downs"
```

```
for x in range(len(orbconf)):
    temp = []
    temp.append([orbconf[x][0], orbconf[x][1], + 0.5])
    temp.append([orbconf[x][0], orbconf[x][1], - 0.5])
    comb.append(list(itertools.product(temp, repeat=orbconf[x][2])))

# Generating the cartesian product of all the single electron adjustments

store  = list(itertools.product(*comb))
for x in range(len(store)):
    store[x] = list(flatten(store[x]))

jjtemp = store
jjtemp.sort()

jjtemp = list(jjtemp for jjtemp,_ in itertools.groupby(jjtemp))

# "Blind" production of all coupling results

jredtemp = []
for x in range(len(jjtemp)):
    jredtempap = []
    for y in range(len(jjtemp[x])):
        jredtempap.append([jjtemp[x][y][0], jjtemp[x][y][1],
        abs(jjtemp[x][y][1] + jjtemp[x][y][2])])
    jredtemp.append(jredtempap)
    jredtempap = []

# Eliminating duplicates

for x in range(len(jredtemp)):
    jredtemp[x].sort()
jredtemp.sort()

jredtemp = list(jredtemp for jredtemp,_ in itertools.groupby(jredtemp))

# Splitting the subshell into two jj-subshells (nl- and(!) those with nl+)

a = []
for x in jredtemp:
    cache = [[],[]]
    temp = x[0]
    for y in x:
        if y == temp:
            cache[0].append(y)
```

```
            else:
                cache[1].append(y)
        a.append(cache)

# Function for the seperate coupling of the nl- and nl+ subshell

def splitpm(L):

    # Brute force generation of all possible adjustments of the j_i

    cache = []
    for x in range(len(L)):
        templist = []
        for y in range(len(L[x])):
            temptemplist = []
            a = 0
            while a <= 2*L[x][y][2]:
                temptemplist.append([L[x][y], L[x][y][2] - a])
                a = a + 1
            templist.append(temptemplist)
        cache.append(templist)

    jjtermsout = []
    for x in range(len(cache)):
        jjtermsout.append(list(itertools.product(*cache[x])))

    # Sorting out all sets that do not obtain the Pauli exclusion principle

    jjterms = []
    for x in range(len(jjtermsout)):
        jjterms.append(unique_items(paulifilter(jjtermsout[x])))
    jjterms = list(flatten(jjterms))

    # Calculate J_tot for ones subshell

    jtot = []
    for x in range(len(jjterms)):
        jt = 0
        jtl = []
        for y in range(len(jjterms[x])):
            jt = jt + jjterms[x][y][1]
            jtl.append(jjterms[x][y][0])
        jtl.append(jt)
        jtot.append(jtl)
    jtot.sort(key=lambda x: x[-1])
```

```
    # Arrange output and sort

    jjtermsout = []
    termscount = 0
    copy = list(jtot)
    while copy != []:
        jjtermsout.append(copy[-1])
        a = int(2*copy[-1][-1] + 1)
        aa = float(copy[-1][-1])
        counts = [len(list(subgroup)) for key,
        subgroup in itertools.groupby(jtot,lambda x: x)][-1]
        termscount += counts
        buff = list(copy[-1][:-1])
        for x in range(a):
            b = 0
            buffy = list(buff)
            buffy.append(aa - x)
            while b < counts:
                copy.remove(buffy)
                b = b + 1

    jjtermsout.sort()

    return jjtermsout

'''
Produce the cartesian products of
the different nl- and nl+ subshells
(empty ones are replaced by a dummy to keep
the latex functions working, because they depend
on the number of estimated maximaly possible jj-subshell)
'''

output = []
for x in a:
    if x[1] == []:
        if x[0][0][1] < x[0][0][2]:
            temp = list(itertools.product([[[x[0][0][0],
            x[0][0][1],0],0]],splitpm([x[0]])))
        else:
            temp = list(itertools.product(splitpm([x[0]]),
            [[[x[0][0][0],x[0][0][1],0],0]]))
        output.append(temp)
    else:
        temp = list(itertools.product(splitpm([x[0]]),
        splitpm([x[1]])))
```

```
            output.append(temp)

    new = list(flatten(output))

    return new


# jj terms

def jjterms(orbconf):

    tempstore = []
    comb = []

    # Brute force generation of all combinations of spin "ups and downs"

    for x in range(len(orbconf)):
        temp = []
        temp.append([orbconf[x][0], orbconf[x][1], + 0.5])
        temp.append([orbconf[x][0], orbconf[x][1], - 0.5])
        comb.append(list(itertools.product(temp, repeat=orbconf[x][2])))

    # Generating the cartesian product of all the single electron adjustments

    store  = list(itertools.product(*comb))
    for x in range(len(store)):
        store[x] = list(flatten(store[x]))

    jjtemp = store
    jjtemp.sort()

    jjtemp = list(jjtemp for jjtemp,_ in itertools.groupby(jjtemp))

    # "Blind" production of all coupling results

    jredtemp = []
    for x in range(len(jjtemp)):
        jredtempap = []
        for y in range(len(jjtemp[x])):
            jredtempap.append([jjtemp[x][y][0], jjtemp[x][y][1],
            abs(jjtemp[x][y][1] + jjtemp[x][y][2])])
        jredtemp.append(jredtempap)
        jredtempap = []

    # Eliminating duplicates
```

```
for x in range(len(jredtemp)):
    jredtemp[x].sort()
jredtemp.sort()

jredtemp = list(jredtemp for jredtemp,_ in itertools.groupby(jredtemp))

# Brute force generation of all possible adjustments of the j_i

cache = []
for x in range(len(jredtemp)):
    templist = []
    for y in range(len(jredtemp[x])):
        temptemplist = []
        a = 0
        while a <= 2*jredtemp[x][y][2]:
            temptemplist.append([jredtemp[x][y], jredtemp[x][y][2] - a])
            a = a + 1
        templist.append(temptemplist)
    cache.append(templist)

jjtermsout = []
for x in range(len(cache)):
    jjtermsout.append(list(itertools.product(*cache[x])))

# Sorting out all sets that do not obtain the Pauli exclusion principle

jjterms = []
for x in range(len(jjtermsout)):
    jjterms.append(unique_items(paulifilter(jjtermsout[x])))
jjterms = list(flatten(jjterms))

jtot = []
for x in range(len(jjterms)):
    jt = 0
    jtl = []
    for y in range(len(jjterms[x])):
        jt = jt + jjterms[x][y][1]
        jtl.append(jjterms[x][y][0])
    jtl.append(jt)
    jtot.append(jtl)
jtot.sort(key=lambda x: x[-1])

# Calculate J_tot for ones subshell

jjtermsout = []
termscount = 0
```

```
    copy = list(jtot)
    while copy != []:
        jjtermsout.append(copy[-1])
        a = int(2*copy[-1][-1] + 1)
        aa = float(copy[-1][-1])
        counts = [len(list(subgroup)) for key,
        subgroup in itertools.groupby(jtot,lambda x: x)][-1]
        termscount += counts
        buff = list(copy[-1][:-1])
        for x in range(a):
            b = 0
            buffy = list(buff)
            buffy.append(aa - x)
            while b < counts:
                copy.remove(buffy)
                b = b + 1

    jjtermsout.sort()

    return jjtermsout


# Function for multi-use the LS terms producing function

def multilsf(iorb):

    lsout = []
    for x in range(len(iorb)):
        a = lsterms(iorb[x])
        lsout.append(a)
    lsout = list(lsout)
    return lsout


# Execut the functions

orbconflist = read_orbconf_fac()
print '\n'
def termsforoneconfig(orbconf):

    # A header of each orbital configurations run

    print '____________', '\n'
    print 'Obtaining term symbols for the configuration: ', orbconf, '\n'

    # First all terms are produced in LS and jj coupling
```

```
    terms = lsterms(orbconf)
    lstermsl = hunds_rules(terms)
    jjtermsl = jjterms(orbconf)
    iorb = imorbconf(orbconf)
    multi = multilsf(iorb)
    multin = translateLN(multi)
    multils = sucmixing(multin)
    multilsj = multiJ(multils, halffilled(list(flatten(iorb))))

    # Then the results are interpreted and written into the output files

    multilsjtex = LShisttexwl(multilsj, iorb)
    lsjtex = Translateall(multilsjtex)
    iorbtex = Translateorball(iorb)
    lsjterms = LStex(lsjtex, iorbtex)

    # "Tex" all L-S- and jj-term symbols and the orbital configurations

    if facbool == False:
        iorbtexjj = Translateorballjj(iorb)
        sucjjl = sucjj(list(flatten(iorb)))
        sucjjshl = sucjjsh(sucjjl)
        jjtermsl = texalljj(sucjjshl, iorbtexjj, facbool)
    else:
        facsucjjl = sucjjfac(list(flatten(iorb)))
        facorb = orbfac(list(flatten(iorb)))
        facsucjjshl = []
        for x in facsucjjl:
            facsucjjshl.extend(sucjjsh(x))
        jjtermsl = texalljj(facsucjjshl, facorb, facbool)

    # All the term symbols in a tex-ready form grouped together

    output = [lsjterms, jjtermsl, lstermsl, iorbtex]
    return output


# Rearranging the list for latex()

results = []
orbconflist.reverse()
for x in orbconflist:
    results.append(termsforoneconfig(x))

output = [[],[],[],[]]
```

```
for x in results:
    output[0].append(x[0])
    output[1].append(x[1])
    output[2].append(x[2])
    output[3].append(x[3])

print '\n'

results = list(output)


'''
Executing the latex() function to fill
the .tex and .txt files with results of this script
'''

latex(*results)
```