Rapport BIPM-90/5

BUREAU INTERNATIONAL DES POIDS ET MESURES

On the Hyperfine Structure of Iodine:

1.

How to Calculate

Hyperfine Transition Energies

by

Susanne Fredin-Picard

PAVILLON DE BRETEUIL
F-92312 SÈVRES CEDEX
France

Rapport BIPM-90/5

Bureau International des Poids et Mesures

On the Hyperfine Structure of Iodine:

1.

How to Calculate Hyperfine Transition Energies

by Susanne Fredin-Picard

Abstract

A detailed description how to calculate hyperfine line positions in iodine, using known constants and already developed algebraic expressions for the interaction terms is given. A computer program in Turbo-Pascal has been written for this purpose. Its accuracy has been compared to earlier calculations found in the litterature.

Table of Contents

1 Introduction 5

2 Molecular Iodine 8

3 Origin of Saturated-Absorption Hyperfine Spectra 12
  3.1 Iso-spin in Iodine and Hyperfine Structure 12
  3.2 Saturated Absorption Main Lines 14
  3.3 Cross-over Lines 14

4 Interaction Terms of Hyperfine Splitting in Homonuclear Diatomic Molecules 18

5 An Example of Hyperfine Spectra: the R(98) 58-1 B-X Transition 20
  5.1 How to Find the Hyperfine States 20
  5.2 How to Create the Hamiltonian Matrix and to Find the Energy Eigenvalues 21
  5.3 Numerical Procedure 24
  5.4 Comparison between BIPM Programme and Earlier Calculations 25

6 Conclusion 26

7 References 27

Appendix I-IV

1 Introduction

The observation of hyperfine structure at high resolution, hidden by the Doppler effect, was considerably improved by the use of laser, a powerful light source which offers coherence, polarization and a good definition in wavelength. Numerous studies on the hyperfine structure in molecular iodine have been reported in the scientific literature since Ezekiel and Weiss [1] first observed hyperfine components in I_2 using an Ar^+ laser in a molecular beam experiment[1]. The technique of saturated-absorption [3,4] is frequently used in order to resolve hyperfine spectra in atoms and molecules, but other methods of Doppler-free spectroscopy, like polarization spectroscopy [5], have also been successfully applied.

In connection with the new definition of the metre 1983 at the 17th CGPM[2] [6], five recommended frequencies were given to be used for its realization [7]. These correspond to precisely measured hyperfine transitions in iodine and methane that coinicide with commonly used laser lines. High precision in mechanics, optics and electronics is consequently requested to stabilize the laser frequencies to the transitions. The most common means of frequency stabilization is probably the third-derivative locking technique [8]; by modulating the laser frequency, the third harmonic of the saturated-absorption signal from the reference-gas is detected. Often this sort of laser stabilization is made in one step, directly on to the saturated component. This gives a typical frequency reproducibility of the order of 1 part in 10^{11} [9]. Methods which stabilize a frequency modulated laser in two steps, correcting for short-term and long-term variations, were introduced in the second half of the seventies [10,11] and offer a frequency reproducibility of about 1 part of 10^{12}. Using high finesse gas-cavities, frequency reproducibilities can nowadays be extended to 1 part in 10^{13} or even 10^{14} [12-14]. Using a fast external electro-optic modulator in the laser system, Hall et al. [15] and Bjorklund [16] obtained an improved signal-to-noise ratio by their optical phase modulation/rf sideband technique in the beginning of the eighties. This technique is becoming more common. Recently, Salomon at al. [17] obtained a relative frequency stability in the

1 A brief review of milestones in the research of hyperfine structure of I_2 can be found in Ref. [2].

2 Conférence Générale des Poids et Mesures

millihertz region by experiments in which two 633 nm He-Ne lasers were stabilized to the same high-finesse Fabry-Perot cavity. The latest possibilities for a precise frequency source concern the work being done on cooled and trapped atoms and ions [18,19].

Whatever the method chosen for frequency stabilization, one must also control the influence of experimental parameters on the measured hyperfine frequencies. Investigations of hyperfine frequency as a function of factors like the pressure in the reference gas [20], the laser frequency modulation amplitude [21], the laser power [21], the wave front geometry [22], direction of laser power output [23], gas-lens effects [24], and pollution of the reference gas [25-28] have already been carried out.

Beside experimental progress in the study of hyperfine structure, a theory that describes the dynamics and interactions involved has simultaneously been developed for molecules [29-33]. One can nowadays calculate hyperfine spectra (line positions, line widths and line strengths) with high accuracy. In this way one can carry forward the understanding of the hyperfine structure and its purturbations.

In this report we concentrate on the hyperfine structure of iodine, which is used for frequency stabilization at the BIPM[3]. Two reports on calculated hyperfine spectra, presented by our former co-worker Michael Gläser, have already appeared in this series [34,35]. For this purpose, Gläser used computer programmes put at his disposition by Foth at the PTB[4] [36,37]. These programmes are no longer available so it seemed convenient, once and for all, to have these programmes on call at the BIPM. Calculated data are nescessary, for example, for the identification of new hyperfine transitions. The aim of this report is to give a detailed description *how* to calculate hyperfine line positions with departure from known constants and already developed expressions for the interaction terms. This report is also a contribution to maintain a continuity of the earlier studies performed at BIPM. A modified version of the programme presented here is intended for the fitting of a set of hyperfine constants to a set of hyperfine components. This will be reported later. For those who does not every day confront problems on hyperfine structure and other subjects toutching molecular spectroscopy, a brief

3 At BIPM, stabilized lasers, corresponding to the following wavelengths, are at the moment available: 3.39 um He-Ne(CH_4), 633 nm He-Ne(I_2), 612 nm He-Ne(I_2), 543 nm He-Ne(I_2) and 515 nm $Ar^+(I_2)$.

4 Physikalisch-Technische Bundesanstalt

review of the most currently concepts in this field has earlier been presented in this report series [38]. Additionally, assembled data on hyperfine constants of molecular iodine are listed in Ref.[39]

When chosing programming language, Turbo-Pascal seemed to be an attractive possibility. Turbo-Pascal is well adapted to micro-computers, offers a short compilation time, and one can find a large number of useful subroutines already written for the commercial market. The programme that has been written is listed in Appendix II.

In the next section we describe aspects of the structure of molecular iodine that are important for our particular use. Section 3 reviews the origin of hyperfine structure as it relates to the saturated-absorption technique. Algebraic expressions for the principal energy terms contributing to the hyperfine splitting are listed in Sec.4, and in Sec. 5 we apply the algebraic expressions for the terms to a transition in $^{127}I_2$.

2 Molecular Iodine

Iodine was discovered in 1811 by Courtois, the French chemist, who also discovered the morphine. There are 23 known iodine isotopes of which the only naturally abundant isotope is ^{127}I. Iodine is particularely useful in the field of medecine and is especially important for the functionning of the tyroid gland and the production of thyroxine. In metrology, in work on length standards, $^{127}I_2$ is used for frequency stabilization. Sometimes $^{129}I_2$, which has a very long half-time, is also used for this purpose.

An extensive review of the spectroscopy of I_2 has been given by Mulliken [40]. The most important publications and spectroscopic data up to 1977 are listed by Huber and Herzberg [41]. More recently, Gerstenkorn and Luc observed a large number of lines in the visible wavelength region [42]. They were detected by Fourier transform spectroscopy and originated from the $B^3\Pi_{0_u^+}$- $X^1\Sigma_g^+$ system in $^{127}I_2$. Because of the extensive number of data and high resolution, the B-X system could be unambigously analyzed [43] so precise rotational and vibrational molecular constants were calculated [44]. As this spectrum is very dense, there is a high probability of finding a ro-vibrational transition that is resonant with a laser line. This possibility has been realized on numerous occations: parts of the hyperfine structure emanating from a wide range of ro-vibrational transitions that coincides with commonly used He-Ne, Ar^+ ion or Kr^+ laser lines have been observed in the B-X system in I_2, see for example [2,45,46].

Using well established nomenclature [47], the energy term G(v) of a vibrational level can be written as

$$G(v) = \omega_e\rho\left(v + \frac{1}{2}\right) - \omega_e x_e \rho^2 \left(v + \frac{1}{2}\right)^2 + \ldots \qquad (1)$$

where the vibrational quantum number v=0,1,2... and ω_e and $\omega_e x_e$ are vibrational constants. The isotopic correction factor ρ is defined as the square-root of the fraction between the reduced masses [47], that is, ρ=1 for $^{127}I_2$ and ρ is slightly less than 1 for $^{129}I_2$.

The energy term $F_v(J)$ of a rotational level can be written as

$$F_v(J) = B_v J(J+1) - D_v J^2(J+1)^2 + \dots \tag{2}$$

where B_v and D_v are rotational constants. The total energy $T_v(J)$ can be written as

$$T_v(J) = T_e + G(v) + F_v(J) \tag{3}$$

where T_e represents the vertical electronic energy. The energies T_e, $G(v)$ and $F_v(J)$ are represented in Fig.1-a.

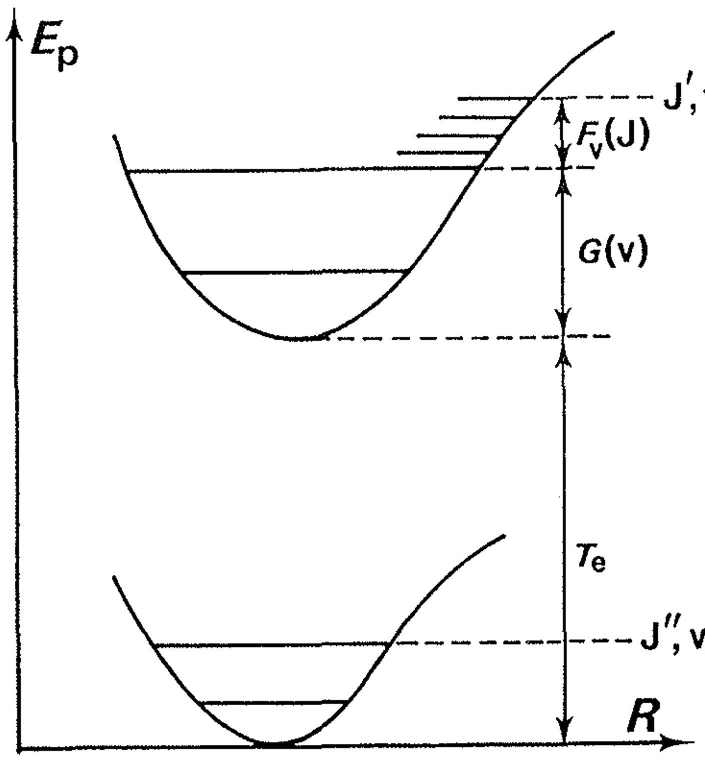

Fig.1-a. Representation of the potential energy curves of the B and X states, where the electronic (T_e), vibrational (G(v)) and rotational (F_v(J)) energy contributions are represented. R is the internuclear distance.

The selection rule for J in the B-X system is

$$\Delta J = J' - J'' = \pm 1 \tag{4}$$

where the (') and (") indicate upper and lower state respectively. The condition for Δv is not restricted (except that the intensity of a Δv transition depends on the vibrational wavefunction overlap; the Franck-Condon factors).

When saturated-absorption is obtained in an iodine-glass-cell within the laser cavity, ground state vibrational numbers v"=5 or v"=6 can be achieved due to the strong internal laser intensity. However, when the iodine-glass-cell is placed outside the laser cavity, only v"=0 or v"=1 can be excited as a consequence of the Boltzmann distribution at room-temperature. An example of some ro-vibrational transitions is schematically represented in Fig.1-b.

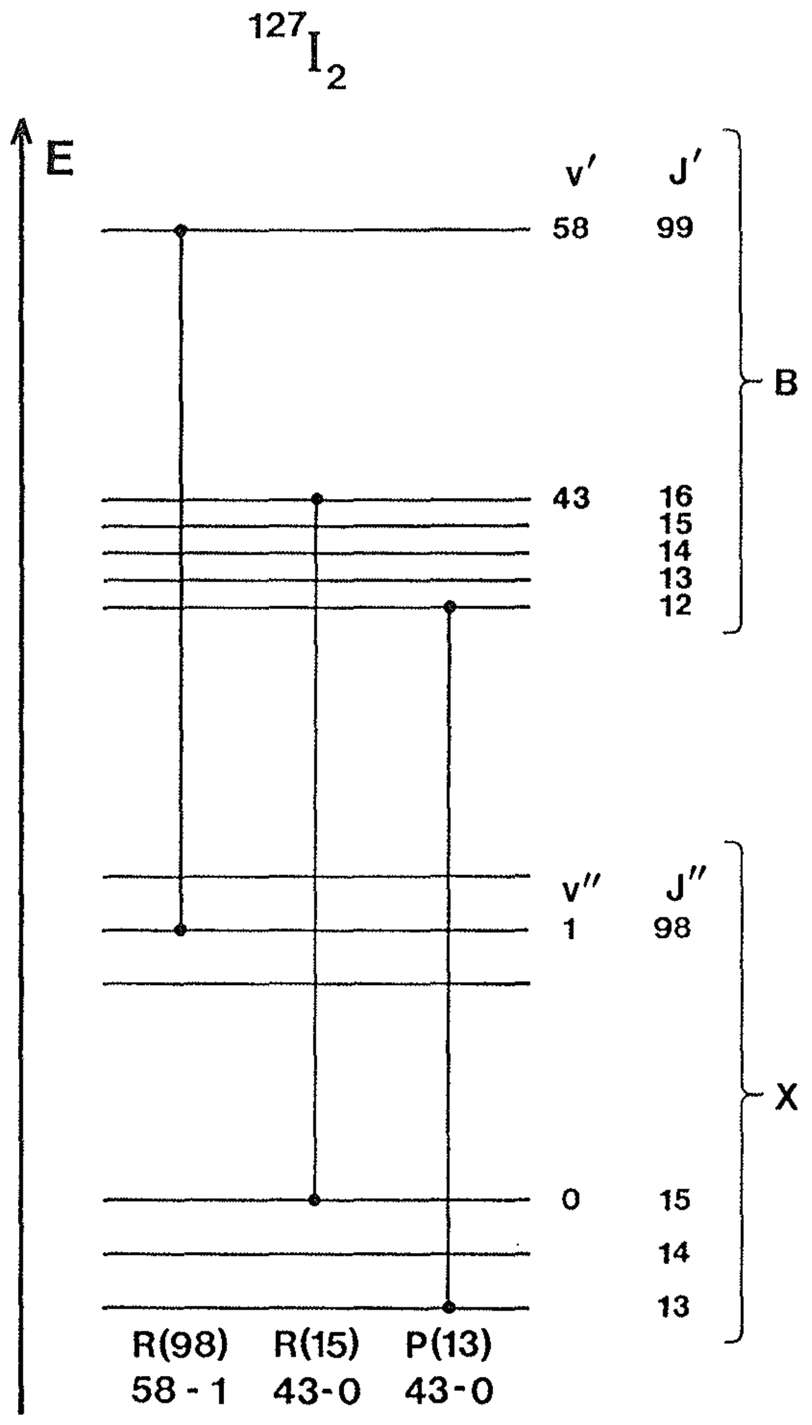

Fig.1-b. Three examples of ro-vibrational transitions of the B-X system of I_2: R(98) 58-1, P(13) 43-0 and R(15) 43-0. These lines can be excited by the Ar^+ laser line at 515 nm.

3 Origin of Saturated-Absorption Hyperfine Spectra

3.1 Iso-spin in Iodine and Hyperfine Structure

Let I_1 represent the intrinsic spin of each nucleus in a diatomic molecule (like iodine). In ^{127}I the intrinsic spin is I_1=5/2, and in the isotope ^{129}I, I_1=7/2. Using the sum-rules for angular momenta, the total nuclear spin I becomes I = 0, 1, 2 ... 5 for $^{127}I_2$. Due to the interaction between the total molecular angular momentum and the iso-spin, **J** and **I** couple and form the angular momentum **F**. If J>I we get

$$F = |J-I|, |J-I+1| \ldots |J+I-1|, |J+I| \tag{5}$$

The levels labelled by F and I are generally not degenerate (see Sec.4); they constitute a set of (2I+1) sublevels to J. The hyperfine structure is illustrated in Fig.2 for one level J.

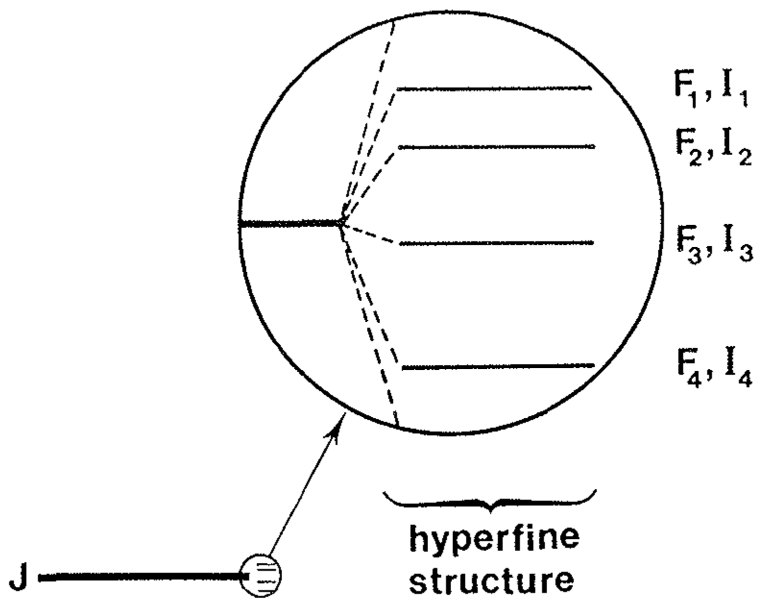

Fig.2. Hypefine structure, labelled by F and I, originating from one rotational level, labelled J.

Because of the symmetries of the involved wavefunctions, states are named ortho and para when J is odd or even. Ortho states are defined by I=1,3,5, while para states have I=0,2,4 in $^{127}I_2$. This is summerized for the B and the X states in Tab.1, taken from the thesis by Camy [48]. With a good approximation $\Delta I=0$

Table 1.

electronic states	ortho-states I=odd	para-states I=even
$B^3\Pi_{0_u^+}$	J even	J odd
$X^1\Sigma_g^+$	J odd	J even

For hyperfine transitions, that is, transitions between hyperfine sub-levels belonging to different electronic states, the strongest transitions are observed when $\Delta F=\Delta J$ if $J>>I$. These transitions are called 'main lines'. In iodine we then have $\Delta F=\Delta J=\pm 1$ (see Eq.(4)). Less probable are the $\Delta F=0$ transitions, which are about $1/2J^2$ less intense that $\Delta F=\Delta J$. One can also have the combination $\Delta F=-\Delta J$, which describes transitions more or less forbidden, with an intensity $1/10J^4$ less than that for $\Delta F=\Delta J$. Using saturated-absorption, so called 'cross-over lines' can also be observed in which the intensity is about 1/2J of that for the $\Delta F=\Delta J$ transitions. These features are described in Sec.3.3. We expect only transitions obeying $\Delta F=\Delta J$ for high values of J.

The main lines are sometimes named a, b, c, d... in decreasing frequency order. It has also been suggested that the components should be named in increasing frequency order [49], a_1, a_2, a_3... This we have chosen to do here.

3.2 Saturated Absorption Main Lines

For a detailed description of the saturated-absorption process, Ref.[8] is recommended. Here we give a short description emphasising only its principal features.

Let a gas containing molecules (or atoms) be excited by two counter propagating laser beams originating from the same laser with frequency f_0. As the molecules have thermal motion, the frequency will appear as $f=f_0(1\pm v_z/c)$ to a molecule, where v_z is the velocity component of the molecule in the laser beam direction, and c is the velocity of light. This means that different molecules, moving in different directions, will interact differently with the incoming and outgoing laser photon frequency. The frequency will appear the same for the incoming and outgoing laser beam if $v_z=0$ ($f=f_0$). In this case the same molecule can interact with incoming and outgoing laser photons of the same frequency, which also means that such molecules are exposed to a stronger laser field than are molecules with $v_z\neq0$. If the laser intensity is sufficiently strong, the lower state will be depleted (or the excited state will saturate), so the absorption coefficient is reduced. The excess photons that do not participate in the excitation process pass through the gas without being absorbed. The line profile of this transmitted light conforms with the natural linewidth of the absorbing transition in the ideal case. The decrease in absorption coefficient is sometimes called the '*Bennet hole*' (for any v_z) or the '*Lamb dip*' (for $v_z=0$) and the increase in transmitted light is sometimes called the '*inverse Lamb-dip*'. All these processes are referred to as '*hole burning*'.

3.3 Cross-over Lines

Cross-over lines are lines related to ordinary saturated-absorption lines in which molecules with $v_z\neq0$ contribute to saturated-absorption signals. This is illustrated by Fig.3-a which shows transitions with a common lower level. As the resonant frequency f of the molecule appears as $f=f_0(1\pm v_z/c)$ relative to the laser photon frequency, two hyperfine levels can be simultaneously excited if they are separated by $2f_0v_z/c$ and the excited molecules moves with the velocity $+v_z$ or $-v_z$: for the incoming direction one transition can be excited; for the outgoing direction the other transition in the same molecule can be excited. As the both transitions originate from the same state, this state will be depleted; the absorption coefficient is thus reduced. The excess laser light is not absorbed, and a signal can be observed at a frequency which corresponds to half the sum-frequency of the two transitions[5]. The strongest transitions that can contribute to cross-over lines are those for $\Delta F=\Delta J$ and $\Delta F=0$. In terms of intensity factors given in Sec. 3.1 we note that cross-over lines are expected only for rather low J-values. The number of possible cross-over lines is twice as great as the number of $\Delta F=0$ lines. In Fig.3-b an energy level scheme is shown where the transitions have a common upper level.

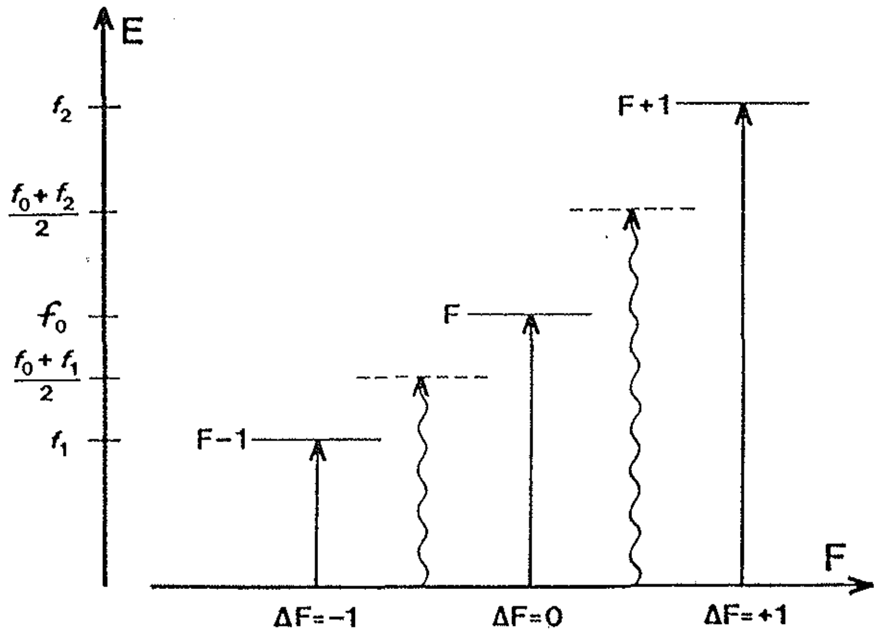

Fig.3-a. Scheme of energy levels involved to obtain a cross-over resonance: The straight arrows represent the laser photon energy in the molecule frame on its way towards/away from the laser source. The wavy arrow represent the laser photon energy in the laboratory frame. These photons passes the sample when saturation is obtained.

5 The recoil effect has not been taken into account here. Bordé et al. [50] have studied this aspect in iodine excited by a 515 nm Ar^+ laser.

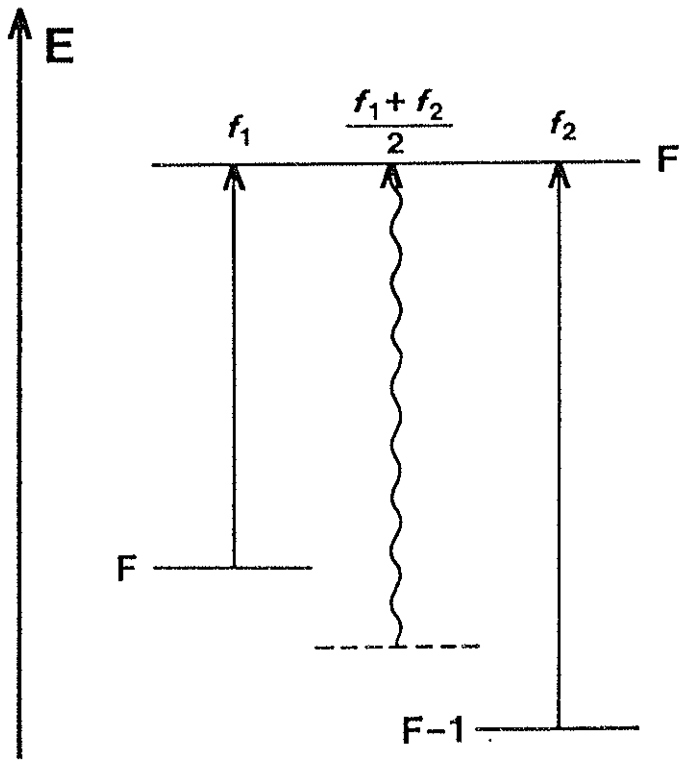

Fig.3-b. Similar scheme to that of Fig.3-a but where the common level is the upper one.

In Tab.2a-b the number of lines corresponding to ortho and para states are listed [48]. One can observe that for an ortho state transition altogether 90 lines are present!

Table 2-a.

	I=1	I=3	I=5	total
ΔF=ΔJ	3	7	11	21
ΔF=0	2	6	10	18
ΔF=-ΔJ	1	5	9	15
cross-over	4	12	20	36

Table 2-b.

	I=0	I=2	I=4	total
ΔF=ΔJ	1	5	9	15
ΔF=0	0	4	8	12
ΔF=-ΔJ	0	3	7	10
cross-over	0	8	16	24

4 Interaction Terms of Hyperfine Splitting in Homonuclear Diatomic Molecules

To calculate the energies of the hyperfine components we first have to derive the Hamiltonian: the Hamiltonian for hyperfine interaction, H_{hfs}, can be written as [36]

$$H_{hfs} = H_{EQ} + H_{SR} + H_{SSS} + H_{TSS} \tag{6}$$

where H_{EQ} and H_{SR} represent the electric quadrupole and spin-rotation interaction respectively. The terms H_{SSS} and H_{TSS} represent scalar spin-spin and tensor spin-spin interactions respectively. Additional higher order terms can be included [2], but these we neglect. The algebraic expressions for the eigenvalues of H_{EQ}, H_{SR}, H_{SSS} and H_{TSS}[6] are listed below and are taken from Foth and Spieweck [36]. The big parentheses represent 3-j symbols, small and large curls brackets correspond to 6-j and 9-j symbols.

$$\begin{aligned} < F, I, J | H_{EQ} | F, I', J' > &= (-1)^{F+J+2I} \frac{1}{2} eqQ(J', J)\,[(2I+1)(2I'+1)]^{1/2} \times \\ &\times \left[\begin{pmatrix} I_1 & I_1 & 2 \\ I_1 & -I_1 & 0 \end{pmatrix} \begin{pmatrix} J' & J & 2 \\ J & -J & 0 \end{pmatrix} \right]^{-1} \begin{Bmatrix} I_1 & I_1 & I \\ 2 & I' & I_1 \end{Bmatrix} \begin{Bmatrix} F & J & I \\ 2 & I' & J' \end{Bmatrix} \end{aligned} \tag{7}$$

$$< F, I, J | H_{SR} | F, I', J' > = \delta_{JJ'} \delta_{II'} \frac{1}{2} C\,[F(F+1) - I(I+1) - J(J+1)] \tag{8}$$

6 The labels SSS and TSS have been inversed in [36], and a missprint has been copied from Bunker and Hanes [30] in the 6-j of the H_{TSS}. The latter was pointed out by Broyer et al. [32] but has no significance in these calculations.

$$< F,I,J| H_{SSS}| F,I',J' >= \delta_{J'J}\delta_{I'I}\frac{1}{2}A[I(I+1)-2I_1(I_1+1)] \tag{9}$$

$$< F,I,J| H_{TSS}| F,I',J' >= \delta_{J'J}(-1)^{F+I'+1}D(2J+1)[I_1(I_1+1)(2I_1+1)]\,[30(2I+1)(2I'+1)]^{1/2}\times$$
$$\times\begin{pmatrix} J & 2 & J \\ 0 & 0 & 0 \end{pmatrix}\begin{Bmatrix} F & J & I \\ 2 & I' & J \end{Bmatrix}\begin{Bmatrix} I_1 & I_1 & 1 \\ I_1 & I_1 & 1 \\ I & I' & 2 \end{Bmatrix} \tag{10}$$

eqQ(J',J), C, A and D represent the constants for the electric quadrupole, spin-rotation, scalar spin-spin and tensorial spin-spin interactions respectively.

It has been shown that for electric quadrupole interaction, one has to consider the interaction between states separated by ΔJ=±2 [45]. Sometimes it is even necessary to consider states separated by ΔJ=±4 or even ΔJ=±8, but we will here only calculate elements where ΔJ=0 and ±2. As a consequence, one must add the rotational energy contribution, given by Eq.(2), to each diagonal element before treating the Hamiltonian matrix.

5 An Example of Hyperfine Spectra: the R(98) 58-1 B-X Transition

We exemplify the calculation of hyperfine lines by studying the R(98) 58-1 transition in the B-X system of $^{127}I_2$. This line coincides with the 515 nm laser line of Ar^+, and its hyperfine structure has been observed by several groups [11,36]. We here refer mainly to the work reported by Foth and Spieweck [36].

5.1 How to Find the Hyperfine States

For this R line we know that J"=98 and J'=99. We here only consider transitions that obey the selection rule ΔF=ΔJ, that is, the strongest transitions. Let us label the transitions using the lower state quantum numbers. As both the lower and upper states can be defined as para states, we know that I=0, 2 and 4. Using the addition rules given by Eq.(5) we hence obtain

										2xI+1
F(I=0)					98					1
F(I=2)			96	97	98	99	100			5
F(I=4)	94	95	96	97	98	99	100	101	102	9

We obtain 15 components since both the upper and lower state contain 15 hyperfine components, so giving 15 hyperfine lines. However, due to the electric quadrupole interaction, states which are separated with ΔJ=±2 will interfere. The listing of components that have to be considered in our calculation is then:

F(I=0)							98							ΔJ=0
F(I=2)					96	97	98	99	100					(J=98)
F(I=4)			94	95	96	97	98	99	100	101	102			

F(I=0)					96									Δ J=-2
F(I=2)			94	95	96	97	98							(J=96)
F(I=4)	92	93	94	95	96	97	98	99	100					

F(I=0)									100					ΔJ=+2
F(I=2)							98	99	100	101	102			(J=100)
F(I=4)					96	97	98	99	100	101	102	103	104	

N=	1	+1	+3	+3	+6	+5	+7	+5	+6	+3	+3	+1	+1	= <u>45</u>
F=	92	93	94	95	96	97	98	99	100	101	102	103	104	

We thus have to consider interactions between 45 components.

5.2 How to Create the Hamiltonian Matrix and to Find the Energy Eigenvalues

We now need to calculate each matrix element of the Hamiltonian. Using Eqs(6-10) and (2) this is a straight-forward matter. The character of the 3-j, 6-j and 9-j symbols have been described by, for example, Edmonds [51] and Brink and Satchler [52]. The expressions for H_{EQ} have been explicitly written in the appendix of Ref.[45]. Further guidelines on the 9-j symbols are given in [2]. Note that H_{SR} and H_{SSS} are purely diagonal.

We hence treat two 45x45 matrices, one for the upper state and one for the lower one. By labelling the matrix by consecutive F, this matrix becomes block-diagonal; it contains 13 sub-matrices where each one has the dimension N (see Sec.5.1) and is labelled by its specific quantum number F. Each matrix element within a sub-matrix is labelled by its quantum number J and I. To calculate the diagonalized 45x45 matrix we diagonalize each sub-matrix which is, evidently, more convenient. The diagonal elements will then represent the energy of the upper (lower) state. We are only interested in the diagonal matrix elements that have J=J' (or J=J"). Fig. 4 shows schematically a part of the hyperfine splitting and some allowed main hyperfine transitions for the R(98) 58-1 line.

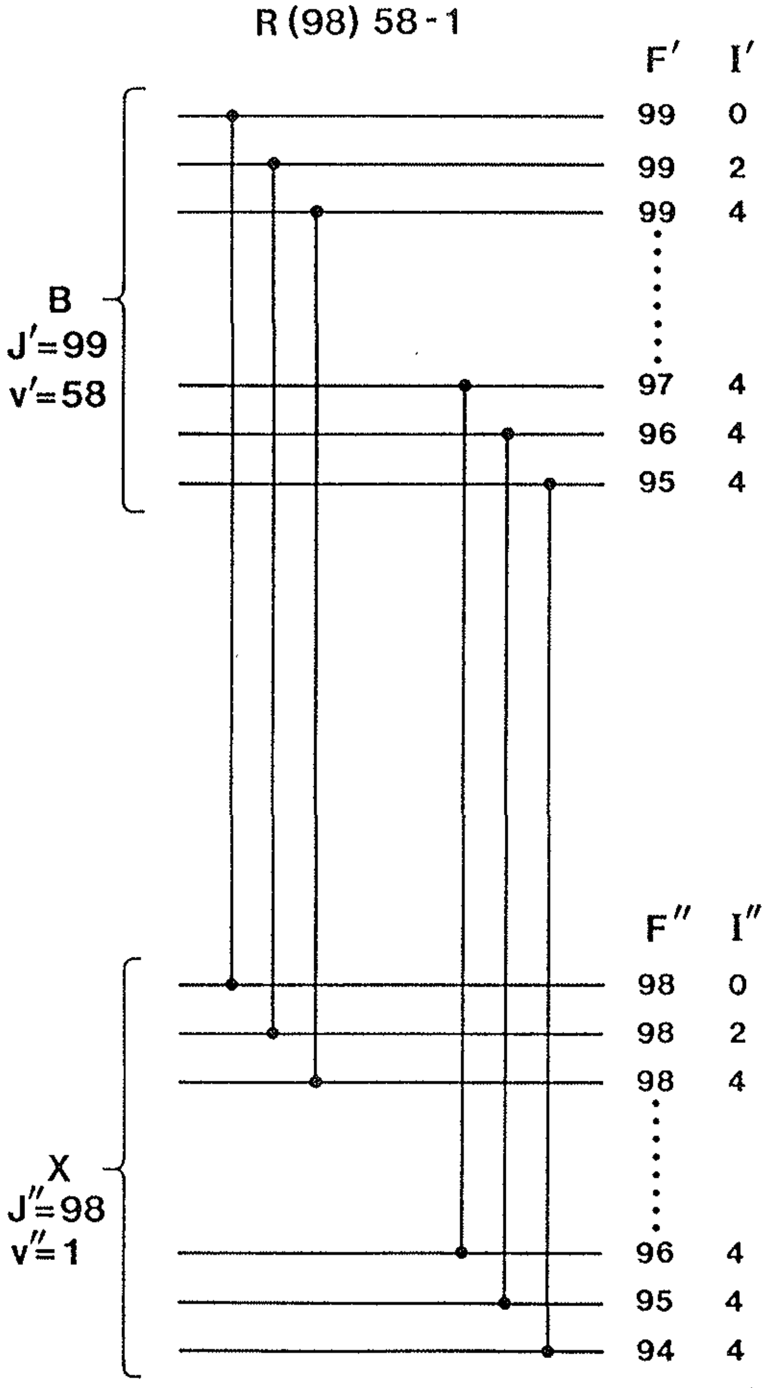

Fig.4. Schematic representation of the transitions between some of the hyperfine components of the upper and lower state respectively. Note that the energy of each level is only symbolic and does not correspond with the energy of the components in reality.

5.3 Numerical Procedure

Appendices I-III show step by step the calculation where the line R(98) 58-1 is used as an example. Below is listed the contents of each appendix.

Appendix I-a	Dimension and labels F, J and I for each submatrix of the upper state hyperfine Hamiltonian for the R(98) 58-1 line.
Appendix I-b	Dimension and contents of each sub-matrix.
Appendix I-c	Dimension and diagonal elements of each diagonalized sub-matrix, corresponding to the labels listed in App.I-a. The energies are given in MHz, and J=J' are marked with an arrow in the right margin.
Appendix II-a	Main programme written in Turbo-Pascal 3.0.
Appendix II-b	Subroutines for calculating the Hamiltonian matrix and deducing the energy differences.
Appendix II-c	Subroutines for calculating the rotational contribution.
Appendix II-d	Subroutines to diagonalize the symmetrical matrix and to calculate the energy eigenvalues.
Appendix III-a	Format of input file.
Appendix III-b	Appearance on screen during a run.
Appendix III-c	Output file where $\Delta F=\Delta J$, $\Delta F=0$ and cross-over lines are listed in MHz. The lines where the frequency is 0.000 do not exist.

5.4 Comparison between BIPM Programme and Earlier Calculations

We have compared the calculations made by our programme with four different calculated structures where three programmes of different origin have been used. These calculations have appeared in works by

a)	Foth and Spieweck [36]
b)	Bordé et al. [2]
c)	Razet [53].

These comparisons are listed in App.IV. In general, the differences between the BIPM calculations and the others differ by less than 1 kHz. This can be explained by round-off errors. However, one discrepancy has been noted: it seems to us that I=0 and I=2 for F=J have been reversed in Tab.1 of Foth and Spieweck [36]. These levels belong to para states. This has no influence on the calculated frequency of the main lines. Unfortunately, we believe that Gläser et al. [54] used the same programme source yet our frequencies differ from theirs for the $\Delta F=0$ and the cross-over lines, when the levels associated to I=0 and I=2 for F=J are involved. We think this may arise from the reversal and, in confirmation, note that compatible values are found for ortho states.

6 Conclusion

A programme that calculates hyperfine lines using already known constants has been written in Turbo-Pascal. It has been tested and reproduce earlier calculations made by other groupes within an error of less than 1 kHz.

Acknowledgement

I am grateful to Dr L. Robertsson for kindly having aquainted me with Turbo-Pascal, and Mrs D. Le Coz for faithfully having assisted in the literature research. I also thank Dr. L. Sturesson (Lund, Sweden) for stimulating discussions.

7 References

[1] S. Ezekiel and R. Weiss, Phys. Rev. Lett. 20 (1968) 91-93.

[2] Ch.J. Bordé, G. Camy, B. Decomps, J.-P. Descoubes and J. Vigué, J. Physique 42 (1981) 1393-1411.

[3] P.H. Lee and M.L. Skolnick, Appl. Phys. Lett. 10 (1967) 303-305.

[4] R.L. Barger and J.L Hall, Phys. Rev. Lett. 22 (1969) 4-8.

[5] C. Wieman and T.W. Hansch, Phys. Rev. Lett. 36 (1976) 1170-1173.

[6] Comptes Rendus 17^e CGPM, (1983) 45-49.

[7] Documents Concerning the New Definition of the Metre, Metrologia 19 (1984) 163-177.

[8] W. Demtröder: Laser Spectroscopy (Springer-Verlag, Berlin, Heidelberg, New York 1982), 2nd corrected edition.

[9] J.-M. Chartier, IEEE Trans. Inst. Meas. IM-32 (1983) 81-83.

[10] L.A. Hackel, R.P. Hackel and S. Ezekiel, Metrologia 13 (1977) 141-143.

[11] G. Camy, B. Decomps, J.-L. Gardissat and C.J. Bordé, Metrologia 13 (1977) 145-148.

[12] P. Cerez, A. Brillet, C.N. Man-Pichot and R. Felder, IEEE Trans. Inst. Meas. IM-29 (1980) 352-354.

[13] A. Clairon, B. Dahmani, A. Filimon and J. Rutman, IEEE Trans. Inst. Meas. IM-34 (1985) 265-268.

[14] O. Acef, Thèse: Améliorisations et comparaisons d'etalons de fréquences optiques. Developpement d'un spectromètre à très haute résolution autour de 28 THz et application à la spectroscopie de SF_6. (Paris XI - 1989).

[15] J.L. Hall, L. Hollberg, T. Baer and H.G. Robinson, Appl. Phys. Lett. 39 (1981) 680-682.

[16] G.C. Bjorklund, Opt. Lett. 5 (1980) 15-17.

[17] Ch. Salomon, D. Hils and J.L. Hall, J. Opt. Soc. Am. B 5 (1988) 1576-1587.

[18] G. Werth, Metrologia 22 (1986) 190-194 and references therein.

[19] W.D. Philips: "Breaking the Laser Cooling Limit", Physics Today Jan. (1989) S.13 and references therein.

[20] M. Gläser, Metrologia 18 (1982) 53-58.

[21] J.-M. Chartier, J. Helmcke and A.J. Wallard, IEEE Trans. Inst. Meas. IM-25 (1976) 450-453.

[22] J.L. Hall and C.J. Bordé, Appl. Phys. Lett. 29 (1976) 788-790.

[23] R. Felder, Metrologia 23 (1986/87) 101-109.

[24] P. Cerez and R. Felder, Appl. Opt. 22 (1983) 1251-1256.

[25] W.R.C. Rowley and B.R. Marx, Metrologia 17 (1981) 65-66.

[26] F. Spieweck, IEEE. Trans. Inst. Meas. IM-34 (1985) 246-248.

[27] P. Gill and R.C. Thompson, Metrologia 23 (1986/87) 161-166.

[28] S. Fredin-Picard, Metrologia 26 (1989) 235-244.

[29] M. Kroll, Phys. Rev. Lett. 23 (1969) 631-633.

[30] P.R. Bunker and G.R. Hanes, Chem. Phys. Lett. 28 (1974) 377-379.

[31] L.A. Hackel, K.H. Casleton, S.G. Kukolich and S. Ezekiel, Phys. Rev. Lett. 35 (1975) 568-571.

[32] M. Broyer, J. Vigué and J.C. Lehmann 39 (1978) 591-609.

[33] J. Vigué, M. Broyer and J.C. Lehmann, J. Physique 42 (1981) 937-947 and references therein.

[34] M. Gläser, Rapport BIPM -82/12.

[35] M. Gläser, Rapport BIPM -83/5.

[36] H.J. Foth and F. Spieweck, Chem. Phys. Lett. 65 (1979) 347-352.

[37] M. Gläser, K. Dschao and H.J. Foth, Opt. Comm. 38 (1981) 119-123.

[38] S. Fredin-Picard, Rapport BIPM-89/6.

[39] M. Gläser, PTB Bericht Opt-25 (1987) 173.

[40] R.S. Mulliken, J. Chem. Phys. 55 (1971) 288-309.

[41] K.P. Huber and G. Herzberg, Constants of Diatomic Molecules, (Van Nostrand Reinhold Company, New York 1979).

[42] S. Gerstenkorn and P. Luc, Atlas du spectre d'absorption de la molécule d'iode 14800-20000 cm^{-1}, (CNRS 1978).

[43] S. Gerstenkorn and P. Luc, Atlas du spectre d'absorption de la molécule d'iode 14800-20000 cm^{-1}. Complement: Identification des transitions du système (B-X), (CNRS 1985).

[44] S. Gerstenkorn and P. Luc, J. Physique 46 (1985) 867-881.

[45] G.R. Hanes, J. Lapierre, P.R. Bunker and K.C. Shotton, J. Mol. Spec. 39 506-515.

[46] M.D. Levenson and A.L. Schawlow, Phys. Rev. A6 (1972) 10-20.

[47] G. Herzberg, Spectra of Diatomic Molecules (Van Nostrand Reinhold Company, New York 1950).

[48] G. Camy, Thèse d'état: Sources laser ultrastables en spectroscopie de saturation. Realisation d'étalons optiques de fréquence et bcaracterisation de leurs qualités (Paris XIII - 1985).

[49] M. Gläser, CCDM/82-32.

[50] C.J. Bordé, G. Camy and B. Decomps, Phys. Rev. A. 20 (1979) 254-268.

[51] A.R. Edmonds, Angular Momentum in Quantum Mechanics (Princeton University Press, Princeton, New Jersey 1960).

[52] D.M. Brink and G.R. Satchler, Angular Momentum (Oxford University Press 1971).

[53] A. Razet, Thèse: Transitions hyperfines de l'iode moléculaire au service de la métroligie des fréquences optiques (Paris XI - 1988).

[54] M. Gläser, Opt. Comm. 54 (1985) 335-342.

APPENDIX I

		F	J	I
1				
	1	93	97	4
1				
	2	94	97	4
3				
	3	95	97	2
	3	95	97	4
	3	95	99	4
3				
	4	96	97	2
	4	96	97	4
	4	96	99	4
6				
	5	97	97	0
	5	97	97	2
	5	97	97	4
	5	97	99	2
	5	97	99	4
	5	97	101	4
5				
	6	98	97	2
	6	98	97	4
	6	98	99	2
	6	98	99	4
	6	98	101	4
7				
	7	99	97	2
	7	99	97	4
	7	99	99	0
	7	99	99	2
	7	99	99	4
	7	99	101	2
	7	99	101	4
5				
	8	100	97	4
	8	100	99	2
	8	100	99	4
	8	100	101	2
	8	100	101	4
6				
	9	101	97	4
	9	101	99	2
	9	101	99	4
	9	101	101	0
	9	101	101	2
	9	101	101	4
3				
	10	102	99	4
	10	102	101	2
	10	102	101	4
3				
	11	103	99	4
	11	103	101	2
	11	103	101	4
1				
	12	104	101	4
1				
	13	105	101	4

```
   13
    1
-13.812723

    1
-13.808353

    3
-13.805212    0.003547   -0.009439
  0.003547  -13.802989   -0.002714
 -0.009439   -0.002714   -0.028364

    3
-13.791770    0.005068   -0.006742
  0.005068  -13.796598   -0.004113
 -0.006742   -0.004113   -0.023830

    6
-13.785336    0.009426    0.000000   -0.011373    0.000000    0.000000
  0.009426  -13.781805    0.005609    0.004276   -0.004458    0.000000
  0.000000    0.005609  -13.789152   -0.001072   -0.004966    0.000000
 -0.011373    0.004276   -0.001072   -0.020519    0.003548   -0.009438
  0.000000   -0.004458   -0.004966    0.003548   -0.018300   -0.002716
  0.000000    0.000000    0.000000   -0.009438   -0.002716   13.967716

    5
-13.775431    0.005173    0.005291   -0.002600    0.000000
  0.005173  -13.780620   -0.002421   -0.005289    0.000000
  0.005291   -0.002421   -0.006910    0.005069   -0.006740
 -0.002600   -0.005289    0.005069   -0.011743   -0.004115
  0.000000    0.000000   -0.006740   -0.004115   13.972413

    7
-13.772760    0.003695   -0.011258    0.004364   -0.001174    0.000000    0.00000
  0.003695  -13.770969    0.000000   -0.004238   -0.005068    0.000000    0.00000
 -0.011258    0.000000   -0.000312    0.009426    0.000000   -0.011372    0.00000
  0.004364   -0.004238    0.009426    0.003219    0.005609    0.004277   -0.00445
 -0.001174   -0.005068    0.000000    0.005609   -0.004129   -0.001073   -0.00496
  0.000000    0.000000   -0.011372    0.004277   -0.001073   13.975895    0.00355
  0.000000    0.000000    0.000000   -0.004456   -0.004967    0.003550   13.97810

    5
-13.760170   -0.006540   -0.004283    0.000000    0.000000
 -0.006540    0.009757    0.005172    0.005291   -0.002598
 -0.004283    0.005172    0.004572   -0.002423   -0.005289
  0.000000    0.005291   -0.002423   13.989670    0.005070
  0.000000   -0.002598   -0.005289    0.005070   13.984834

    6
-13.748189   -0.009343   -0.002884    0.000000    0.000000    0.000000
 -0.009343    0.012594    0.003694   -0.011259    0.004363   -0.001173
 -0.002884    0.003694    0.014389    0.000000   -0.004240   -0.005067
  0.000000   -0.011259    0.000000   13.996432    0.009426    0.000000
  0.000000    0.004363   -0.004240    0.009426   13.999963    0.005609
  0.000000   -0.001173   -0.005067    0.000000    0.005609   13.992615

    3
  0.025355   -0.006542   -0.004282
 -0.006542   14.006666    0.005171
 -0.004282    0.005171   14.001483

    3
  0.037499   -0.009344   -0.002883
 -0.009344   14.009669    0.003692
 -0.002883    0.003692   14.011468

    1
 14.022600

    1
 14.034908
```

```
   13
 1
  -1.38127230000000E+005
 1
  -1.38083530000000E+005
 3
  -1.38078201871944E+005
  -1.38003878154855E+005
  -2.83569973201285E+002 ←
 3
  -1.37885749733650E+005
  -1.37997975578720E+005
  -2.38254687630513E+002 ←
 6
  -1.37955078302640E+005
  -1.37727986939517E+005
  -1.37880005161877E+005
  -2.31203970840877E+002 ←
  -1.56914587816620E+002 ←
   1.39677228962692E+005
 5
  -1.37722408970694E+005
  -1.37838150846074E+005
  -3.71301332427789E+001 ←
  -1.49394669508544E+002 ←
   1.39724174619520E+005
 7
  -1.37756751657725E+005
  -1.37680676924358E+005
  -1.04771131532650E+002 ←
   1.22276901108068E+002 ←
  -2.97254542089946E+001 ←
   1.39732923641354E+005
   1.39807254625363E+005
 5
  -1.37601744373173E+005
   1.29517872974233E+002 ←
   1.37674372475754E+001 ←
   1.39928714610798E+005
   1.39816374452154E+005
 6
  -1.37481959468077E+005
   9.68371323008827E+001 ←
   1.72925836129723E+002 ←
   1.39862735128497E+005
   1.40089741850066E+005
   1.39937759521083E+005
 3
   2.53506284968139E+002 ←
   1.40098629079713E+005
   1.39982904635318E+005
 3
   3.74921573492518E+002 ←
   1.40067707216351E+005
   1.40143731210157E+005
 1
   1.40226000000000E+005
 1
   1.40349080000000E+005
```

APPENDIX II

```
Program Hyper;

(* ************************************************ *)
(*     Calculate frequencies for delta-F=delta-J     *)
(*         and delta-F=0 hyperfine transitions.      *)
(*             Calculate cross-over lines.           *)
(*           Give line, J", v' and v", then          *)
(*         upper and lower hyperfine constants       *)
(*               in an external file.                *)
(* ************************************************ *)

LABEL  20, 30, 40, 50;

CONST  convert = 29979.2458;

       IOerr : boolean = false;
       TNArraySize = 8;

TYPE   pp      = RECORD
                   F, JP, IP, JB, IB    : integer;
                 END;

       mattyp7 = array [1..15,1..8,1..8] of real;
       atyp    = array [1..63] of pp;
       btyp    = array [1..63] of real;
       ctyp    = array [1..15] of integer;
       dtyp    = array [1..15,1..8,1..8] of pp;
       htyp    = array [1..8] of real;

       TNvector    = array[1..TNArraySize] of real;
       TNmatrix    = array[1..TNArraySize] of TNvector;
       TNIntVector = array[1..TNArraySize] of integer;

VAR    M                                : mattyp7;
       till, fran,
       EQQ, CKON, AKON, DKON            : real;
       k1, m1, n1, jinit, iinit,
       antal, antmat, aja,
       Jbis, vprim, vbis, NofLines,
       Dimen, x1                        : integer;
       mille                            : TNmatrix;
       slask, egen                      : atyp;
       eget, egetu, egetl               : btyp;
       nostat                           : ctyp;
       punkt                            : dtyp;
       hkon                             : htyp;
       bok, state                       : char;

{$I a:S1.PAS }
{$I a:ROTER.PAS }
{$I a:ANROP.PAS }

(* ************************************************ *)

BEGIN

(* ************************************************ *)
(*            Determine initial conditions.          *)
(*         Read hyperfine constants from file.       *)
(* ************************************************ *)
```

```
20: SetNumber(vprim, vbis, bok, Jbis, aja, hkon);
    IF aja=1 THEN GOTO 30;
    state:='L';

(* ************************************************* *)
(*  Calculate energy differences between rotational  *)
(*  levels with delta-J=+-2. First lower, the upper. *)
(* ************************************************* *)

40: jinit:=Jbis-2;

    IF state='U' THEN
    BEGIN
      IF bok = 'R' THEN jinit:=jinit+1
      ELSE jinit:=jinit-1;
      EQQ:=hkon[1];
      CKON:=hkon[3];
      AKON:=hkon[5];
      DKON:=hkon[7];
      till:=Diffup(vprim, jinit+4)
           -Diffup(vprim, jinit+2);
      fran:=Diffup(vprim, jinit+2)
           -Diffup(vprim, jinit);
    END
    ELSE
    BEGIN
      EQQ:=hkon[2];
      CKON:=hkon[4];
      AKON:=hkon[6];
      DKON:=hkon[8];
      till:=Difflow(vbis, Jbis+2)-Difflow(vbis, Jbis);
      fran:=Difflow(vbis, Jbis)-Difflow(vbis, Jbis-2);
    END;

(* ************************************************* *)
(*              Find eigen vectors.                  *)
(*                F=(J-I)...(J+I)                    *)
(* ************************************************* *)

    iinit:=0;
    IF Odd(Jbis) THEN iinit:=1;

    FindEigVec(jinit, iinit, slask, antal);

(* ************************************************* *)
(*  Arrange eigenvectors F,J,I in increasing order.  *)
(* ************************************************* *)

    RangeEig(slask, egen, antal, jinit, iinit,
             state, Noflines);

(* ************************************************* *)
(*         Count number of eigen vectors             *)
(*      and find the number of sub matrices.         *)
(* ************************************************* *)

    CalcEig(egen, nostat, antal, antmat);

(* ************************************************* *)
(*      Make matrix containing the eigen states.      *)
(* ************************************************ *)

    PunktMat(punkt, egen, nostat, antmat);

(* ************************************************ *)
(*                 Calculate matrix.                  *)
(* ************************************************ *)

    MatElement(antmat, nostat, punkt, M, till, fran,
               state, EQQ, CKON, AKON, DKON);

(* ************************************************ *)
(*        Diagonalize and find eigenvalues.           *)
(* ************************************************ *)

    x1:=0;

    FOR n1:=1 TO antmat DO
    BEGIN
      Dimen:=nostat[n1];
      FOR m1:=1 TO Dimen DO
      FOR k1:=1 TO Dimen DO
      mille[m1, k1]:=0.0001*M[n1, m1, k1];
      diag(eget, Dimen, x1, mille);
    END;

    IF state<>'L' THEN egetu := eget
    ELSE egetl:=eget;
    WRITELN(' Program is running ! ');

    IF state<>'L' THEN GOTO 50;
    state:='U';
    GOTO 40;

(* ************************************************ *)
(*         Calculate frequency differences.           *)
(* ************************************************ *)

50: differ(egen, egetu, egetl, nostat, jinit,antmat,
           NofLines, vprim, vbis, Jbis, bok, hkon);

30: END.
```

```
(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function di0dj0(a,b,c:integer):real;

(* 3j: Delta-I=0 Delta-J=0 *)

CONST    I1 = 2.5;

VAR      F, I, J, d, frad, g, h   : real;

BEGIN
    F:=a;
    J:=b;
    I:=c;
    frad:=F*(F+1)-I*(I+1)-J*(J+1);
    d:=-3/(16*(2*J+3)*(2*J-1)*I1*(2*I1-1));
    g:=(F+I-J+1)*(F+J-I)*(F+I+J+2)*(I+J-F+1);
    g:=g*((2*I1+1)*(2*I1+1)-(I+1)*(I+1))/
         ((2*I+1)*(2*I+3));
    h:=(F+I-J)*(F+J-I+1)*(F+I+J+1)*(I+J-F);
    h:=h*((2*I1+1)*(2*I1+1)-I*I)/(4*I*I-1);
    di0dj0:=d*(frad*frad+2*frad+g+h-
               16*I1*(I1+1)*J*(J+1)/3);
END; (* Function di0dj0 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function dim2dj0(a,b,c:integer):real;

(* 3j: Delta-I=-2 Delta-J=0 *)

CONST    I1 = 2.5;

VAR      F, I, J, p, q, x, y, z   : real;

BEGIN
    F:=a;
    J:=b;
    I:=c;
    x:=3*SQRT((2*I1+1)*(2*I1+1)-I*I)
        *SQRT((2*I1+1)*(2*I1+1)-(I-1)*(I-1));
    y:=-16*(2*J+3)*(2*J-1)*I1*(2*I1-1)*SQRT((4*I*I-1)
          *(2*I-3)*(2*I-1));
    z:=SQRT((F+I-J-1)*(F+I-J)*(F+J-I+1));
    p:=SQRT((F+J-I+2)*(F+I+J)*(F+I+J+1));
    q:=SQRT((I+J-F-1)*(I+J-F));
    dim2dj0:=(x/y)*z*p*q;
END; (* Function dim2dj0 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function di0dj2(a,b,c:integer):real;

(* 3j: Delta-I=0 Delta-J=2 *)

CONST    I1 = 2.5;

VAR      x, y, z, p, q, F, I, J    : real;

BEGIN
```

```
    F:=a;
    J:=b;
    I:=c;
    x:=3*(3*I*(I+1)-3-4*I1*(I1+1));
    y:=16*(2*J+3)*(2*I+3)*(2*I1-1)*(2*I-1)*I1
         *SQRT((2*J+1)*(2*J+5));
    z:=SQRT((F+I-J-1)*(F+I-J)*(F+J-I+1));
    p:=SQRT((F+J-I+2)*(F+I+J+2)*(F+I+J+3));
    q:=SQRT((I+J-F+1)*(I+J-F+2));
    di0dj2:=(x/y)*z*p*q;
END; (* Function di0dj2 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function di2dj2(a,b,c:integer):real;

(* 3j: Delta-I=2 Delta-J=2 *)

CONST    I1 = 2.5;

VAR      F, I, J, x, y, p, q      : real;

BEGIN
    F:=a;
    J:=b;
    I:=c;
    x:=3*SQRT((2*I1+I+2)*(2*I1+I+3)
        *(2*I1-I-1)*(2*I1-I));
    y:=32*(2*J+3)*(2*I+3)*(2*I1-1)*I1
         *SQRT((2*J+1)*(2*J+5)*(2*I+1)*(2*I+5));
    p:=SQRT((F+I+J+5)*(F+J+I+4)*(F+I+J+2)*(F+I+J+3));
    q:=SQRT((I+J-F+1)*(I+J-F+2)*(I+J-F+3)*(I+J-F+4));
    di2dj2:=(x/y)*p*q;
END; (* Function di2dj2 *)

(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function dim2dj2(a,b,c:integer):real;

(* 3j: Delta-I=-2 Delta-J=2 *)

CONST    I1 = 2.5;

VAR      x, y, z, p, q, F, I, J         : real;

BEGIN
    F:=a;
    J:=b;
    I:=c;
    x:=3*SQRT((2*I1-I+2)*(2*I1-I+1)
        *(2*I1+I+1)*(2*I1+I));
    y:=32*(2*J+3)*(2*I-1)*(2*I1-1)*I1
         *SQRT((2*J+1)*(2*J+5)*(2*I+1)*(2*I-3));
    p:=SQRT((F+I-J)*(F+I-J-1)*(F+I-J-2)*(F+I-J-3));
    q:=SQRT((F+J-I+1)*(F+J-I+2)*(F+J-I+3)*(F+J-I+4));
    dim2dj2:=(x/y)*p*q;
END; (* Function dim2dj2 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)
```

```
Function NioJ0(a:integer):real;

(* 9j: Delta-I=0 *)

CONST    I1 = 2.5;

VAR      I, S    : real;

BEGIN
    I:=a;
    S:=(4*I1*(I1+1)+I*(I+1))/
         (I1*(I1+1)*(2*I1+1)*SQRT(120));
    NioJ0:=S*SQRT((I*(I+1))/((2*I-1)*(2*I+1)*(2*I+3)));
END; (* Function NioJ0 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function NioJ2(a:integer):real;

(* 9j: Delta-I=2 *)

CONST    I1 = 2.5;

VAR      I, S    : real;

BEGIN
    I:=a;
    S:=SQRT(((2*I1+1)*(2*I1+1)-(I-1)*(I-1))*(I-1)*I/5);
    s:=S*SQRT(((2*I1+1)*(2*I1+1)-I*I)/
         ((2*I-3)*(2*I-1)*(2*I+1)));
    NioJ2:=-S/(2*I1*(2*I1+1)*(2*I1+2));
END; (* Function NioJ2 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function Cfunk(f1,j1,i1:integer):real;

(* HFS-SR *)

VAR      C, X, F, I, J          : real;

BEGIN
    F:=f1;
    J:=j1;
    I:=i1;
    X:=F*(F+1)-I*(I+1)-J*(J+1);
    Cfunk:=X/2;
END; (* Function Cfunk *)

(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function Afunk(a:integer):real;

(* HFS-SSS *)

CONST    I1 = 2.5;

VAR      X, I          : real;

BEGIN
```

```
    I:=a;
    X:=I*(I+1)-2*I1*(I1+1);
    Afunk:=X/2;
END; (* Function Afunk *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function Dfunk(f1,j1,a,b:integer):real;

(* HFS-TSS *)

CONST     I1 = 2.5;

VAR       C, X, F, IP, IB, J    : real;
          fas, s                : integer;

BEGIN
    fas:=1;
    F:=f1;
    J:=j1;
    IP:=a;
    IB:=b;
    s:=TRUNC(0.02+J+IP+F);
    IF Odd(s) THEN fas:=-1;
    X:=SQRT(30*(2*IB+1)*(2*IP+1))*(2*J+1)
                             *I1*(I1+1)*(2*I1+1);
    X:=X*SQRT((J*(J+1))/((2*J+3)*(2*J+1)*(2*J-1)));
    Dfunk:=X*fas;
END; (* Function Dfunk *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Function sexj0(A1,B1,C1:integer):real;

(* 6j:  a b c  *)
(*      2 c b  *)

LABEL   10;

VAR     fas, s                          : integer;
        a, b, c, x1, y1, z1, X, se      : real;

BEGIN
    se:=0.0;
    a:=A1;
    b:=B1;
    c:=C1;
    s:=TRUNC(a+b+c+0.02);
    X:=b*(b+1)+c*(c+1)-a*(a+1);
    fas:=1;
    x1:=3*X*(X-1)-4*b*(b+1)*c*(c+1);
    IF x1=0.0 THEN GOTO 10;
    y1:=(2*b+3)*(2*b+2)*(2*b+1)*2*b*(2*b-1);
    z1:=(2*c+3)*(2*c+2)*(2*c+1)*2*c*(2*c-1);
    IF Odd(s) THEN fas:=-1;
    se:=x1/SQRT(y1*z1);
10: sexj0:=2*fas*se;
END; (* Function sexj0 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)
```

```
Function sexj2(A1,B1,C1:integer):real;

(* 6j:  a  b  c   *)
(*      2 c-2 b   *)

VAR     fas, s1                          : integer;
        a, b, c, x1, y1, z1, X, s        : real;

BEGIN
    a:=A1;
    b:=B1;
    c:=C1;
    s1:=TRUNC(a+b+c+0.1);
    s:=a+b+c;
    fas:=1;
    x1:=6*s*(s+1)*(s-2*a-1)*(s-2*a);
    y1:=(s-2*b-1)*(s-2*b)/((2*b-1)
        *2*b*(2*b+1)*(2*b+2)*(2*b+3));
    z1:=(s-2*c+1)*(s-2*c+2)/((2*c-3)
        *(2*c-2)*(2*c-1)*2*c*(2*c+1));
    IF Odd(s1) THEN fas:=-1;
    sexj2:=fas*SQRT(x1*y1*z1);
END; (* Function sexj2 *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Procedure SetNumber(VAR vprim, vbis:integer;
                    VAR bok         :char;
                    VAR Jbis, aja   :integer;
                    VAR hkon        :htyp);

(* Read initial data from file *)

LABEL 2;

VAR     FileName        : string[255];
        F1              : text;
        n1              : integer;

BEGIN
    WRITE('File name of input hyperfine constants ? ');
    READLN(FileName);
    ASSIGN(F1, FileName);
    RESET(F1);

    aja:=0;

(* ---------- Read file on form R 98 58 1 ---------- *)

    READ(F1, bok);
    READ(F1, Jbis);
    READLN(F1, vprim, vbis);

    bok:=UpCase(bok);
    IF (bok<>'R') AND (bok<>'P') AND (bok<>'Q') THEN
    BEGIN
      WRITELN('Forbidden line !! ');
      aja:=1;
      GOTO 2;
    END;

    IF bok='Q' THEN
    BEGIN
      WRITELN('Forbidden transition !!!');
      aja:=1;
      GOTO 2;
    END;

(* ---------- Read 8 hyperfine constants; ---------- *)
(* ------ upper and lower for each constant. ------- *)

    FOR n1:=1 TO 8 DO
    READLN(F1, hkon[n1]);

    CLOSE(F1);
2:END; (* Procedure SetNumber *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Procedure FindEigVec(jinit, iinit  :integer;
                 VAR slask         :atyp;
                 VAR antal:        integer);

(* Find appropriate eigen-vectors *)

VAR     sum, n1, JC, m1, IC, ftal   : integer;

BEGIN
    sum:=0;
    FOR n1:=0 TO 2 DO
    BEGIN
      JC:=jinit+2*n1;
      FOR m1:=0 TO 2 DO
      BEGIN
        IC:=iinit+2*m1;
        FOR ftal:=(JC-IC) TO (JC+IC) DO
        BEGIN
          sum:=sum+1;
          WITH slask[sum] DO
          BEGIN
            F:=ftal;
            JP:=JC;
            IP:=IC;
          END;
        END;
      END;
    END;

(* ---------- antal = dimension of matrix ---------- *)

    antal:=sum;
END; (* Procedure FindEigVec *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Procedure RangeEig(VAR slask, egen            :atyp;
                       antal, jinit, iinit  :integer;
                       state                :char;
                   VAR NofLines             :integer);
```

```
(* Arrange eigen-vectors by increasing F, I and J *)

VAR    sum, n1, m1, k1     : integer;

BEGIN
    sum:=0;
    k1:=0;
    FOR m1:=(jinit-(iinit+4)) TO
              (jinit+4+(iinit+4)) DO
    FOR n1:=1 TO antal DO
    IF slask[n1].F=m1 THEN
    BEGIN
      sum:=sum+1;
      egen[sum]:=slask[n1];
      IF state='U' THEN
      IF egen[sum].JP=jinit+2 THEN
      k1:=k1+1;
    END;

(* ----- NofLines = number of main transitions ----- *)

    NofLines:=k1;
END; (* Procedure RangeEig *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Procedure CalcEig(egen             :atyp;
              VAR nostat           :ctyp;
              VAR antal, antmat    :integer);

(* Find number of sub-matrices and dimensions *)

LABEL 1;

VAR   m1, sum, k1, n1  : integer;

BEGIN
    m1:=egen[1].F;
    sum:=0;
    k1:=1;

    FOR n1:=2 TO antal DO
    WITH egen[n1] DO
    BEGIN
      sum:=sum+1;

      IF F>m1 THEN
      BEGIN
        nostat[k1]:=sum;
        antmat:=k1;
        k1:=k1+1;
        sum:=0;
        m1:=F;
      END;

      IF n1=antal THEN
      BEGIN
        nostat[k1]:=sum+1;
        antmat:=k1;
        GOTO 1;
      END;
    END;

(* -------- antmat = number of sub-matrices -------- *)
(* ------- nostat = dimension of each matrix ------- *)

1:  antmat:=k1;
END; (* Procedure CalcEig *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Procedure PunktMat(VAR punkt   :dtyp;
                       egen    :atyp;
                       nostat  :ctyp;
                       antmat  :integer);

(* Create matrix of eigen-vectors *)

VAR     n1, m1, k1, sum     : integer;

BEGIN
    sum:=0;
    FOR n1:=1 TO antmat DO
    BEGIN
      FOR m1:=1 TO nostat[n1] DO
      FOR k1:=1 TO nostat[n1] DO
      BEGIN
        punkt[n1, m1, k1].F:=egen[sum+1].F;
        punkt[n1, m1, k1].JP:=egen[sum+k1].JP;
        punkt[n1, m1, k1].IP:=egen[sum+k1].IP;
        punkt[n1, m1, k1].JB:=egen[sum+m1].JP;
        punkt[n1, m1, k1].IB:=egen[sum+m1].IP;
      END;
      sum:=sum+nostat[n1];
    END;
END; (* Procedure PunktMat *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

Procedure MatElement(antmat                     :integer;
                     nostat                     :ctyp;
                     punkt                      :dtyp;
                 VAR M                          :mattyp7;
                     till, fran                 :real;
                     state                      :char;
                     EQQ, CKON, AKON, DKON      :real);

(* Calculate raw matrix elements including constants *)
(*     Calculate the hyperfine correction matrix      *)

VAR     n1, m1, k1, ms, ns             : integer;
        C, A, D, p1, p2, p3, Delta     : real;

BEGIN


(*  Calculate half the matrix and symmetrize later.  *)

    FOR n1:=1 TO antmat DO
```

```
BEGIN
  ns:=1;
  ms:=1;
  FOR m1:=ms TO nostat[n1] DO
  BEGIN
    FOR k1:=ns TO nostat[n1] DO
    WITH punkt[n1, m1, k1] DO
  . BEGIN
      C:=0.0;  A:=0.0;  D:=0.0;
      p1:=0.0; p2:=0.0; p3:=0.0;
      Delta:=0.0;

      IF JP-JB=0 THEN
      BEGIN
        p2:=0.0;
        p3:=0.0;
        IF IP-IB=0 THEN
        BEGIN
          C:=Cfunk(F, JP, IP);
          A:=Afunk(IP);
          p1:=di0dj0(F, JB, IB);
          p2:=sexj0(F, JB, IB);
          p3:=NioJ0(IP);
(* ----- Delta = rotational energy difference ------ *)
          IF JP=jinit+2 THEN Delta:=0
          ELSE IF JP=jinit+4 THEN Delta:=till
          ELSE IF JP=jinit THEN Delta:=-fran;
          Delta:=Delta*convert;
        END

        ELSE IF IP-IB=-2 THEN
        BEGIN
          p1:=dim2dj0(F, JB, IB);
          p2:=sexj2(F, JB, IB);
          p3:=NioJ2(IB);
        END

        ELSE IF IP-IB=2 THEN
        BEGIN
          p1:=dim2dj0(F, JP, IP);
          p2:=sexj2(F, JP, IP);
          p3:=NioJ2(IP);
        END
        ELSE IF ABS(IP-IB)>2.05 THEN
        p1:=0;

        D:=Dfunk(F, JP, IP, IB)*p2*p3;
      END

      ELSE IF JP-JB=2 THEN
      BEGIN
```

```
              IF IP-IB=0 THEN
              p1:=di0dj2(F, JB, IB)
              ELSE IF IP-IB=2 THEN
              p1:=di2dj2(F, JB, IB)
              ELSE IF IP-IB=-2 THEN
              p1:=dim2dj2(F, JB, IB)
              ELSE IF ABS(IP-IB)>2.05 THEN
              p1:=0;
            END

            ELSE p1:=0;

(*  p1 set to 0 because matrix is symmetriced later. *)

            M[n1, m1, k1]:=p1*EQQ+C*CKON+Delta
                          +A*AKON+D*DKON;

(* ---------------- Ssymmetrize ! ------------------ *)

            M[n1, k1, m1]:=M[n1, m1, k1];
          END;
          ms:=ms+1;
          ns:=ns+1;
        END;
      END;
END; (* Procedure MatElement *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)

procedure differ(egen                         :atyp;
                 egetu, egetl                 :btyp;
                 nostat                       :ctyp;
                 jinit, antmat, NofLines,
                 vprim, vbis, Jbis            :integer;
                 bok                          : char;
                 hkon                         :htyp);

(* Calculate differences between states *)
(*        for Delta-F=Delta-J,          *)
(*    Delta-F=0 and cross over lines    *)

LABEL   33, 44, 69;

VAR     FileName              : string[255];
        UtFil                 : text;
        n1, x1, y1, m1,
        Fbis                  : integer;
        diff                  : real;
        p,b                   : char;
        a, etikett            : atyp;
        linje, lu, ll,
        DD, crossu, crossl    : btyp;

(* lu - upper level *)
(* ll - lower level *)
(* DD line with delta-F=0 *)
(* crossu - cross-over with common upper level *)
(* crossl - cross-over with common lower level *)

BEGIN
```

```
    WRITE('File name output frequency differences ? ');
    READLN(FileName);
    ASSIGN(UtFil, FileName);
    REWRITE(UtFil);

(* ----------------- Reset to 0. ------------------- *)

    y1:=0;
    x1:=0;

    FOR n1:=1 TO NofLines DO
    BEGIN
      lu[n1]:=0;
      ll[n1]:=0;
      DD[n1]:=0;
      crossu[n1]:=0;
      crossl[n1]:=0;
    END;

(* ---------- Set prime and double prime. ---------- *)

    p:=chr(39);
    b:=chr(34);

(* ---------- Find upper and lower levels. --------- *)

    FOR n1:=1 TO antmat DO
    BEGIN
      FOR m1:=1 TO nostat[n1] DO
      BEGIN
        x1:=x1+1;
        IF egen[x1].JP=jinit+2 THEN
        BEGIN
          y1:=y1+1;
          etikett[y1]:=egen[x1];
          lu[y1]:=egetu[x1];
          ll[y1]:=egetl[x1];
        END;
      END;
    END;

(* --- Find Delta-F = Delta-J and Delta-0 lines. --- *)

    FOR n1:=1 TO NofLines DO
    linje[n1]:=lu[n1]-ll[n1];

    IF bok='R' THEN
    FOR n1:=1 TO (NofLines-1) DO
    FOR m1:=0 TO 2 DO
    IF (m1+n1)<NofLines THEN
    IF etikett[n1].F = (etikett[n1+1+m1].F-1) THEN
    IF etikett[n1].IP = etikett[n1+1+m1].IP THEN
    DD[n1]:=lu[n1]-ll[n1+1+m1];

    IF bok='P' THEN
    FOR n1:=2 TO NofLines DO
    FOR m1:=0 TO 2 DO
    IF (n1-m1)>1 THEN
    IF etikett[n1].F = (etikett[n1-1-m1].F+1) THEN
    IF etikett[n1].IP = etikett[n1-1-m1].IP THEN
    DD[n1]:=lu[n1]-ll[n1-1-m1];

(* ------------ Find cross-over lines. ------------- *)

    FOR n1:=1 TO NofLines DO
    BEGIN
      IF DD[n1]=0 THEN GOTO 69 ELSE
      crossu[n1]:=(linje[n1]+DD[n1])/2;
69: END;

    IF bok='R' THEN
    FOR n1:=2 TO NofLines DO
    FOR m1:=0 TO 2 DO
    IF (n1-m1)>1 THEN
    IF etikett[n1].F = (etikett[n1-1-m1].F+1) THEN
    IF etikett[n1].IP = etikett[n1-1-m1].IP THEN
    BEGIN
      IF DD[n1-1-m1]=0 THEN GOTO 33 ELSE
      crossl[n1]:=(linje[n1]+DD[n1-1-m1])/2;
33: END;

    IF bok='P' THEN
    FOR n1:=1 TO NofLines-1 DO
    FOR m1:=0 TO 2 DO
    IF (n1+m1)<NofLines THEN
    IF etikett[n1].F = (etikett[n1+1+m1].F-1) THEN
    IF etikett[n1].IP = etikett[n1+1+m1].IP THEN
    BEGIN
      IF DD[n1+1+m1]=0 THEN GOTO 44 ELSE
      crossl[n1]:=(linje[n1]+DD[n1+1+m1])/2;
44: END;

(*    Arrange lines in increasing frequency order.    *)

    REPEAT
      diff:=linje[1];
      FOR n1:=1 TO NofLines-1 DO
      IF linje[n1+1]<linje[n1] THEN
      BEGIN
        diff:=linje[n1+1];
        linje[n1+1]:=linje[n1];
        linje[n1]:=diff;

        diff:=DD[n1+1];
        DD[n1+1]:=DD[n1];
        DD[n1]:=diff;

        diff:=crossu[n1+1];
        crossu[n1+1]:=crossu[n1];
        crossu[n1]:=diff;

        diff:=crossl[n1+1];
        crossl[n1+1]:=crossl[n1];
        crossl[n1]:=diff;

        a[n1]:=etikett[n1+1];
        etikett[n1+1]:=etikett[n1];
        etikett[n1]:=a[n1];
      END;
    UNTIL diff = linje[1];
```

```
(* --- Print data on file. --- *)

    WRITE(UtFil,' TRANSITION IN IODINE :  ');
    WRITELN(UtFil,bok:2,' (',Jbis:3,') ',
                  vprim:3,'-',vbis:2);
    WRITELN(UtFil);

    WRITELN(UtFil,'        Constants: eqQ',p:1,
                  ' eqQ',b:1,' C',p:1,' C',b:1,
                  ' A',p:1,' A',b:1,' D',p:1,' D',b:1);
    WRITELN(UtFil);
    FOR n1:=1 TO 8 DO
    WRITELN(UtFil, n1:5, hkon[n1]:13:6);
    WRITELN(UtFil);
    WRITELN(UtFil);
    WRITELN(UtFil,' no',' F',p:1,'  F',b:1,'  I ',
                  '  D-F=D-J ','F',p:1,'=F',b:1,
                  '   D-F=0  ','  F',p:1,'   cros up ',
                               '  F',b:1,'   cros lo ');
    WRITELN(UtFil);

    FOR n1:=1 TO NofLines DO
    WITH etikett[n1] DO
    BEGIN
      IF bok='R' THEN Fbis:=F-1 ELSE Fbis:=F+1;
      WRITELN(UtFil, n1:3, F:4, Fbis:4, IP:3,
                               linje[n1]:10:3,
                             F:5, DD[n1]:10:3,
                             F:5, crossu[n1]:10:3,
                          Fbis:5, crossl[n1]:10:3);
    END;

    WRITELN(UtFil);
    CLOSE(UtFil);
END; (* Procedure differ *)

(*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*)
```

```
(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)
(*          Diaginalization of square matrix            *)
(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)

Procedure Jacobi(Dimen          : integer;
                 Mat            : TNmatrix;
                 MaxIter        : integer;
                 Tolerance      : real;
             VAR Eigenvalues    : TNvector;
             VAR Eigenvectors   : TNmatrix;
             VAR Iter           : integer;
             VAR Error          : byte);

CONST    TNNearlyZero = 1E-15;

VAR   Row, Column, Diag  : integer;
      SinTheta, CosTheta : real;
      SumSquareDiag      : real;
      Done               : boolean;

(* --------------------------------------------------- *)

Procedure TestData(Dimen      : integer;
               VAR Mat        : TNmatrix;
                   MaxIter    : integer;
                   Tolerance  : real;
               VAR Error      : byte);

VAR   Row, Column : integer;

BEGIN
  Error:=0;
  IF Dimen<1 THEN Error:=1;
  IF Tolerance<=TNNearlyZero THEN Error:=2;
  IF MaxIter<1 THEN Error:=3;
  IF Error=0 THEN
  FOR Row:=1 TO Dimen-1 DO
  FOR Column:=Row+1 TO Dimen DO
  IF ABS(Mat[Row, Column]-Mat[Column, Row])>
  TNNearlyZero THEN Error:=4;
END; (* Procedure TestData *)

(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)

Procedure Initialize(Dimen            : integer;
                 VAR Iter             : integer;
                 VAR Eigenvectors     : TNmatrix);

VAR   Diag : integer;

BEGIN
  Iter:=0;
  FillChar(Eigenvectors, SizeOf(Eigenvectors), 0);
  FOR Diag:=1 TO Dimen DO
  Eigenvectors[Diag, Diag]:=1;
END; (* Procedure Initialize *)

(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)

Procedure CalculateRotation(RowRow    : real;
                                       RowCol    : real;
                                       ColCol    : real;
                                   VAR SinTheta : real;
                                   VAR CosTheta : real);

VAR   TangentTwoTheta, TangentTheta, Dummy : real;

BEGIN
  IF ABS(RowRow-ColCol)>TNNearlyZero THEN
  BEGIN
    TangentTwoTheta:=(RowRow-ColCol)/(2*RowCol);
    Dummy:=SQRT(SQR(TangentTwoTheta)+1);

    IF TangentTwoTheta<0 THEN
    TangentTheta:=-TangentTwoTheta-Dummy
    ELSE
    TangentTheta:=-TangentTwoTheta+Dummy;

    CosTheta:=1/SQRT(1+SQR(TangentTheta));
    SinTheta:=CosTheta*TangentTheta;
  END
  ELSE
  BEGIN
    CosTheta:=SQRT(1/2);
    IF RowCol<0 THEN SinTheta:=-SQRT(1/2)
    ELSE
    SinTheta:=SQRT(1/2);
  END;
END; (* Procedure CalculateRotation *)

(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)

Procedure RotateMatrix(Dimen    : integer;
                       SinTheta : real;
                       CosTheta : real;
                       Row      : integer;
                       Col      : integer;
                   VAR Mat      : TNmatrix);

VAR   CosSqr, SinSqr, SinCos             : real;
      MatRowRow, MatColCol, MatRowCol,
      MatRowIndex, MatColIndex           : real;
      Index                              : integer;

BEGIN
  CosSqr:=Sqr(CosTheta);
  SinSqr:=Sqr(SinTheta);
  SinCos:=SinTheta*CosTheta;
  MatRowRow:=Mat[Row, Row]*CosSqr+2*Mat[Row, Col]*SinCos
            +Mat[Col, Col]*SinSqr;
  MatColCol:=Mat[Row, Row]*SinSqr-2*Mat[Row, Col]*SinCos
            +Mat[Col, Col]*CosSqr;
  MatRowCol:=(Mat[Col, Col]-Mat[Row, Row])*SinCos
            +Mat[Row, Col]*(CosSqr-SinSqr);

  FOR Index:=1 TO Dimen DO
  IF NOT(Index in [Row, Col]) THEN
  BEGIN
    MatRowIndex:=Mat[Row, Index]*CosTheta
                +Mat[Col, Index]*SinTheta;
    MatColIndex:=-Mat[Row, Index]*SinTheta
                 +Mat[Col, Index]*CosTheta;
    Mat[Row, Index]:=MatRowIndex;
    Mat[Index, Row]:=MatRowIndex;
    Mat[Col, Index]:=MatColIndex;
    Mat[Index, Col]:=MatColIndex;
  END;
  Mat[Row, Row]:=MatRowRow;
  Mat[Col, Col]:=MatColCol;
  Mat[Row, Col]:=MatRowCol;
  Mat[Col, Row]:=MatRowCol;
END; (* Procedure RotateMatrix *)

(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)

Procedure RotateEigenvectors(Dimen        : integer;
                             SinTheta     : real;
                             CosTheta     : real;
                             Row          : integer;
                             Col          : integer;
                         VAR Eigenvectors : TNmatrix);

VAR   EigenvectorsRowIndex, EigenvectorsColIndex : real;
      Index : integer;

BEGIN
  (* Transform eigenvector matrix *)
  FOR Index:=1 TO  Dimen DO
  BEGIN
    EigenvectorsRowIndex:=
                CosTheta*Eigenvectors[Row, Index]
               +SinTheta*Eigenvectors[Col, Index];
    EigenvectorsColIndex:=
               -SinTheta*Eigenvectors[Row, Index]
               +CosTheta*Eigenvectors[Col, Index];
    Eigenvectors[Row, Index]:=EigenvectorsRowIndex;
    Eigenvectors[Col, Index]:=EigenvectorsColIndex;
  END;
END; (* Procedure RotateEigenvectors *)

(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)

Procedure NormalizeEigenvectors(Dimen        : integer;
                            VAR Eigenvectors : TNmatrix);

VAR   Row     : integer;
      Largest : real;

(* -------------------------------------------------- *)

Procedure FindLargest(Dimen       : integer;
                  VAR Eigenvector : TNvector;
                  VAR Largest     : real);

VAR   Term : integer;

BEGIN
  Largest:=Eigenvector[1];
  FOR Term:=2 TO Dimen DO
  IF ABS(Eigenvector[Term])>ABS(Largest) THEN
  Largest:=Eigenvector[Term];
END; (* Procedure FindLargest *)

(* -------------------------------------------------- *)

Procedure DivVecConst(Dimen          : integer;
                  VAR ChangingRow : TNvector;
                      Divisor     : real);

VAR   Term : integer;

BEGIN
  FOR Term:=1 TO Dimen DO
  ChangingRow[Term]:=ChangingRow[Term]/Divisor;
END; (* Procedure DivVecConst *)

(* -------------------------------------------------- *)

BEGIN (* Procedure NormalizeEigenvectors *)
  FOR Row:=1 TO Dimen DO
  BEGIN
    FindLargest(Dimen, Eigenvectors[Row], Largest);
    DivVecConst(Dimen, Eigenvectors[Row], Largest);
  END;
END; (* Procedure NormalizeEigenvectors *)

(* -------------------------------------------------- *)

BEGIN (* Procedure Jacobi *)
  TestData(Dimen, Mat, MaxIter, Tolerance, Error);
  IF Error=0 THEN
  BEGIN
    Initialize(Dimen, Iter, Eigenvectors);
    REPEAT
      Iter:=Succ(Iter);
      SumSquareDiag:=0;
      FOR Diag:=1 TO Dimen DO
      SumSquareDiag:=SumSquareDiag+SQR(Mat[Diag, Diag]);
      Done:=TRUE;
      FOR Row:=1 TO Dimen-1  DO
      FOR Column:=Row+1 TO Dimen DO
      IF ABS(Mat[Row, Column])>
      Tolerance*SumSquareDiag THEN
      BEGIN
        Done:=false;
        CalculateRotation(Mat[Row, Row], Mat[Row, Column],
                          Mat[Column, Column],
                          SinTheta, CosTheta);
        RotateMatrix(Dimen, SinTheta, CosTheta,
                     Row, Column, Mat);
        RotateEigenvectors(Dimen, SinTheta, CosTheta,
                           Row, Column, Eigenvectors);
      END;
    UNTIL Done OR (Iter>MaxIter);
    FOR Diag:=1 TO Dimen DO
    Eigenvalues[Diag]:=Mat[Diag, Diag];
    NormalizeEigenvectors(Dimen, Eigenvectors);
    IF Iter>MaxIter THEN Error:=5
  END;
END; (* Procedure Jacobi *)
```

```
(* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *)

Procedure diag(VAR eget   :btyp;
                   Dimen  :integer;
               VAR k1     :integer;
                   Mat    :TNmatrix);

VAR      MaxIter,Iter    : integer;
         Tolerance       : real;
         Eigenvectors    : TNmatrix;
         Eigenvalues     : TNvector;
         Error           : byte;
         m1              : integer;

(* -------------------------------------------------- *)

BEGIN
    Tolerance:=1E-8;
    MaxIter:=200;

    Jacobi(Dimen, Mat, MaxIter, Tolerance,
           Eigenvalues, Eigenvectors, Iter, Error);

    FOR m1:=1 TO Dimen DO
    BEGIN
      k1:=k1+1;
      eget[k1]:=10000*Eigenvalues[m1];
    END;
END; (* Procedure Diag *)
```

APPENDIX III

```
R 98 58 1
-493.839
-2374.037
0.8292
0.0
0.3568
0.0
-0.5004
0.0015
```

```
A:\|h1
File name of input hyperfine constants ? a:BXR98.ABC
 Program is running !
 Program is running !
File name output frequency differences ? a:UT.DAT

A:\|
TRANSITION IN IODINE :   R ( 98)  58- 1

       Constants: eqQ' eqQ" C' C" A' A" D' D"

   1   -493.839000
   2  -2374.037000
   3      0.829200
   4      0.000000
   5      0.356800
   6      0.000000
   7     -0.500400
   8      0.001500

no  F'   F"   I   D-F=D-J F'=F"   D-F=0    F'   cros up   F"    cros lo

 1  95   94   4  -525.145   95  -351.471   95  -438.308   94      0.000
 2  99   98   2  -471.189   99  -118.879   99  -295.034   98   -550.894
 3  96   95   4  -306.162   96  -298.737   96  -302.449   95   -328.816
 4  98   97   2  -270.771   98  -630.599   98  -450.685   97   -367.808
 5  97   96   4  -217.391   97   134.968   97   -41.211   96   -258.064
 6 100   99   2  -111.641  100   431.291  100   159.825   99   -115.260
 7  97   96   2    60.596   97  -464.846   97  -202.125   96      0.000
 8  99   98   4    88.847   99   271.554   99   180.200   98     29.015
 9 101  100   4   114.335  101   121.943  101   118.139  100     34.757
10 103  102   4   141.590  103     0.000  103     0.000  102     80.880
11  98   97   4   142.491   98   -30.818   98    55.837   97    138.729
12 102  101   4   202.523  102    20.171  102   111.347  101    162.233
13 100   99   4   315.047  100   -44.822  100   135.113   99    293.300
14  99   98   0   369.941   99     0.000   99     0.000   98      0.000
15 101  100   2   398.612  101     0.000  101     0.000  100    414.951
```

APPENDIX IV

```
TRANSITION IN IODINE :    R ( 98)  58- 1

       Constants: eqQ' eqQ" C' C" A' A" D' D"

   1  -493.839000
   2 -2374.037000
   3     0.829200
   4     0.000000
   5     0.356800
   6     0.000000
   7    -0.500400
   8     0.001500

no   F'   F"   I    D-F=D-J F'=F"   D-F=0     F'    cros up    F"    cros lo

 1   95   94   4   -525.145   95   -351.471   95   -438.308    94      0.000
 2   99   98   2   -471.189   99   -118.879   99   -295.034    98   -550.894
 3   96   95   4   -306.162   96   -298.737   96   -302.449    95   -328.817
 4   98   97   2   -270.771   98   -630.599   98   -450.685    97   -367.808
 5   97   96   4   -217.391   97    134.968   97    -41.211    96   -258.064
 6  100   99   2   -111.641  100    431.290  100    159.825    99   -115.260
 7   97   96   2     60.596   97   -464.846   97   -202.125    96      0.000
 8   99   98   4     88.847   99    271.554   99    180.200    98     29.015
 9  101  100   4    114.335  101    121.944  101    118.139   100     34.757
10  103  102   4    141.590  103      0.000  103      0.000   102     80.881
11   98   97   4    142.491   98    -30.818   98     55.836    97    138.729
12  102  101   4    202.523  102     20.171  102    111.347   101    162.233
13  100   99   4    315.047  100    -44.821  100    135.113    99    293.300
14   99   98   0    369.942   99      0.000   99      0.000    98      0.000
15  101  100   2    398.613  101      0.000  101      0.000   100    414.952

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

Comparison between BIPM program and calculation by Foth and Spieweck [36].

no   F'   F"   I       f        f+111.631   f[36]

 1   95   94   4   -525.145    -413.514   -413.513
 2   99   98   2   -471.189    -359.558   -359.558
 3   96   95   4   -306.162    -194.531   -194.530
 4   98   97   2   -270.771    -159.140   -159.139
 5   97   96   4   -217.391    -105.760   -105.759
 6  100   99   2   -111.641      -0.010     -0.010
 7   97   96   2     60.596     172.227    172.228
 8   99   98   4     88.847     200.478    200.478
 9  101  100   4    114.335     225.966    225.965
10  103  102   4    141.590     253.221    253.220
11   98   97   4    142.491     254.122    254.122
12  102  101   4    202.523     314.154    314.152
13  100   99   4    315.047     426.678    426.677
14   99   98   0    369.942     481.573    481.573
15  101  100   2    398.613     510.244    510.243
```

```
TRANSITION IN IODINE :    P ( 13)  43- 0

       Constants: eqQ' eqQ" C' C" A' A" D' D"

   1  -558.581000
   2 -2452.551000
   3      0.191488
   4      0.004250
   5     -0.002977
   6      0.000000
   7     -0.101709
   8      0.000000

no  F'   F"   I    D-F=D-J F'=F"    D-F=0     F'    cros up    F"    cros lo

 1   7    8   5   -543.858   7      0.000     7      0.000      8   -572.074
 2  12   13   1   -471.995  12    570.744    12     49.375     13   -577.552
 3  17   18   5   -412.087  17     46.070    17   -183.009     18      0.000
 4   8    9   5   -336.038   8   -600.289     8   -468.164      9   -360.888
 5  13   14   3   -208.859  13    179.136    13    -14.862     14   -232.223
 6  11   12   3   -171.314  11    -42.155    11   -106.734     12   -207.853
 7   9   10   5   -157.081   9   -385.738     9   -271.410     10   -194.835
 8  10   11   3    -73.389  10     55.871    10     -8.759     11    -57.772
 9  16   17   5    -62.373  16    286.512    16    112.070     17     -8.152
10  14   15   3    -43.113  14   -255.586    14   -149.350     15    -45.260
11  15   16   3    -18.124  15    -47.408    15    -32.766     16      0.000
12   9   10   3     23.510   9      0.000     9      0.000     10     39.690
13  12   13   3     87.623  12   -244.391    12    -78.384     13    133.380
14  10   11   5    106.091  10   -232.589    10    -63.249     11     97.084
15  11   12   5    175.309  11     88.076    11    131.693     12    151.084
16  15   16   5    204.665  15    367.550    15    286.108     16    245.589
17  13   14   1    248.844  13   -683.110    13   -217.133     14      0.000
18  14   15   5    328.272  14    429.164    14    378.718     15    347.911
19  11   12   1    330.056  11      0.000    11      0.000     12    450.400
20  12   13   5    345.544  12    126.859    12    236.202     13    337.720
21  13   14   5    405.248  13    329.896    13    367.572     14    417.206
```

Comparison between BIPM program and calculation by Bordé at al. [2].

no	F'	F"	I	f	f+543.858	f[2]
1	7	8	5	-543.858	0.000	0.000
2	12	13	1	-471.995	71.863	71.864
3	17	18	5	-412.087	131.771	131.772
4	8	9	5	-336.038	207.820	207.820
5	13	14	3	-208.859	334.999	335.000
6	11	12	3	-171.314	372.544	372.544
7	9	10	5	-157.081	386.777	386.777
8	10	11	3	-73.389	470.469	470.470
9	16	17	5	-62.373	481.485	481.485
10	14	15	3	-43.113	500.745	-
11	15	16	3	-18.124	525.734	525.735
12	9	10	3	23.510	567.368	567.368
13	12	13	3	87.623	631.481	631.482
14	10	11	5	106.091	649.949	-
15	11	12	5	175.309	719.167	719.167
16	15	16	5	204.665	748.523	748.524
17	13	14	1	248.844	792.702	792.703
18	14	15	5	328.272	872.130	-
19	11	12	1	330.056	873.914	-
20	12	13	5	345.544	889.402	889.403
21	13	14	5	405.248	949.106	949.106

no	F'	F"	I	D-F=D-J	cros up	cros lo	a	c-a*	c-a[2]**
1	7	8	5	-543.858	0.000	-572.074	1	-28.216	-28.215
2	12	13	1	-471.995	49.375	-577.552			
3	17	18	5	-412.087	-183.009	0.000			
4	8	9	5	-336.038	-468.164	-360.888	1	75.694	75.693
							4	-24.850	-24.850
							7	-203.807	-203.807
5	13	14	3	-208.859	-14.862	-232.223			
6	11	12	3	-171.314	-106.734	-207.853			
7	9	10	5	-157.081	-271.410	-194.835	4	64.628	64.629
							7	-114.329	-114.328
.									
.									
.									

*The frequency spacing between a main component 'a' and a cross-over line 'c' has been calculated by Bordé et al. and correspond to the underlined data.

**The numerotation of the cross-over lines in [2] is not the same one as recommended in [49].

```
TRANSITION IN IODINE :   R (127)  11- 5

       Constants: eqQ' eqQ" C' C" A' A" D' D"

    1   -559.380000
    2 -2504.048000
    3      0.028405
    4      0.000000
    5     -0.015371
    6      0.000000
    7     -0.020742
    8      0.000000

 no   F'   F"   I    D-F=D-J F'=F"    D-F=0    F'     cros up    F"     cros lo

  1 123  122   5   -510.782  123   -145.689  123   -328.235   122       0.000
  2 128  127   1   -485.996  128    646.756  128     80.380   127    -613.200
  3 133  132   5   -462.225  133      0.000  133      0.000   132    -507.166
  4 124  123   5   -223.686  124    156.734  124    -33.476   123    -184.687
  5 127  126   3   -195.703  127    176.750  127     -9.477   126    -218.139
  6 129  128   3   -193.219  129     -1.685  129    -97.452   128    -237.375
  7 132  131   5   -166.138  132   -552.107  132   -359.123   131    -209.450
  8 125  124   3    -65.737  125    -59.044  125    -62.390   124       0.000
  9 126  125   3    -56.733  126   -240.575  126   -148.654   125     -57.889
 10 130  129   3    -40.907  130    -35.258  130    -38.083   129     -21.296
 11 131  130   3    -32.901  131      0.000  131      0.000   130     -34.080
 12 125  124   5     75.488  125    260.549  125    168.019   124     116.111
 13 128  127   3     97.065  128   -281.531  128    -92.233   127     136.907
 14 131  130   5    119.007  131   -252.761  131    -66.877   130      95.987
 15 126  125   5    222.753  126    234.749  126    228.751   125     241.651
 16 127  126   5    235.958  127    427.453  127    331.705   126     235.353
 17 129  128   5    249.324  129    261.363  129    255.343   128     226.850
 18 130  129   5    262.184  130     72.968  130    167.576   129     261.773
 19 127  126   1    380.087  127   -740.404  127   -180.159   126       0.000
 20 128  127   5    388.164  128    204.375  128    296.270   127     407.808
 21 129  128   1    396.991  129      0.000  129      0.000   128     521.874

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

Comparison between BIPM program and calculation by Razet [53].

 no   F'   F"   I       f        f+510.782    f[53]

  1 123  122   5   -510.782       0.000       0.000
  2 128  127   1   -485.996      24.786      24.786
  3 133  132   5   -462.225      48.557      48.557
  4 124  123   5   -223.686     287.096     287.096
  5 127  126   3   -195.703     315.079     315.081
  6 129  128   3   -193.219     317.563     317.565
  7 132  131   5   -166.138     344.644     344.644
  8 125  124   3    -65.737     445.045     445.046
  9 126  125   3    -56.733     454.049     454.049
 10 130  129   3    -40.907     469.875     469.875
 11 131  130   3    -32.901     477.881     477.882
 12 125  124   5     75.488     586.270     586.271
 13 128  127   3     97.065     607.847     607.847
 14 131  130   5    119.007     629.789     629.789
 15 126  125   5    222.753     733.535     733.536
 16 127  126   5    235.958     746.740     746.739
 17 129  128   5    249.324     760.106     760.105
 18 130  129   5    262.184     772.966     772.966
 19 127  126   1    380.087     890.869     890.869
 20 128  127   5    388.164     898.946     898.946
 21 129  128   1    396.991     907.773     907.774
```

```
TRANSITION IN IODINE :   R ( 47)   9- 2

       Constants: eqQ' eqQ" C' C" A' A" D' D"

    1  -505.114000
    2 -2453.127000
    3      0.025222
    4      0.000000
    5     -0.015813
    6      0.000000
    7     -0.019410
    8      0.000000

no  F'  F"  I   D-F=D-J F'=F"   D-F=0    F'   cros up    F"    cros lo

 1  43  42  5  -509.925  43   -169.937   43  -339.931   42      0.000
 2  48  47  1  -486.730  48    631.671   48    72.470   47   -599.489
 3  53  52  5  -465.235  53      0.000   53     0.000   52   -506.436
 4  44  43  5  -238.928  44    -29.676   44  -134.302   43   -204.432
 5  49  48  3  -200.040  49     -6.687   49  -103.363   48   -239.322
 6  47  46  3  -189.537  47    169.787   47    -9.875   46   -208.076
 7  52  51  5  -152.763  52   -547.638   52  -350.201   51   -190.359
 8  45  44  5   -71.313  45    276.175   45   102.431   44    -50.494
 9  46  45  3   -53.660  46   -226.615   46  -140.138   45    -70.300
10  50  49  3   -45.297  50    -32.345   50   -38.821   49    -25.992
11  51  50  3   -33.715  51      0.000   51     0.000   50    -33.030
12  45  44  3    66.838  45    -86.940   45   -10.051   44      0.000
13  48  47  3    96.837  48   -278.605   48   -90.884   47    133.312
14  51  50  5   131.541  51   -227.954   51   -48.206   50    111.787
15  46  45  5   205.625  46    236.705   46   221.165   45    240.900
16  47  46  5   231.894  47    425.092   47   328.493   46    234.299
17  49  48  5   251.002  49    282.456   49   266.729   48    232.461
18  50  49  5   277.228  50     92.033   50   184.631   49    279.842
19  47  46  1   374.392  47   -712.248   47  -168.928   46      0.000
20  48  47  5   386.461  48    213.920   48   300.191   47    405.777
21  49  48  1   402.332  49      0.000   49     0.000   48    517.002
```

Comparison between BIPM program and calculation by Razet [53].

```
no  F'  F"  I      f        f+509.925    f[53]

 1  43  42  5  -509.925       0.000      0.000
 2  48  47  1  -486.730      23.195     23.195
 3  53  52  5  -465.235      44.690     44.690
 4  44  43  5  -238.928     270.997    270.998
 5  49  48  3  -200.040     309.885    309.885
 6  47  46  3  -189.537     320.388    320.388
 7  52  51  5  -152.763     357.162    357.162
 8  45  44  5   -71.313     438.612    438.612
 9  46  45  3   -53.660     456.265    456.265
10  50  49  3   -45.297     464.628    464.628
11  51  50  3   -33.715     476.210    476.210
12  45  44  3    66.838     576.763    576.763
13  48  47  3    96.837     606.762    606.762
14  51  50  5   131.541     641.466    641.466
15  46  45  5   205.625     715.550    715.550
16  47  46  5   231.894     741.819    741.819
17  49  48  5   251.002     760.927    760.927
18  50  49  5   277.228     787.153    787.153
19  47  46  1   374.392     884.317    884.317
20  48  47  5   386.461     896.386    896.386
21  49  48  1   402.332     962.257    962.257
```