Analytical Representation of Numerical Self Consistent Field (SCF) Hartree–Fock Radial Wave Functions for Ions

A Thesis Presented to The School of Graduate Studies

Addis Ababa University

In Partial Fulfillment of the Requirement

of the Degree of Master of Science in Physics

By

Elias Getachew

June 1996

To Atiye, a beloved mother
and Mohammed Abdo, a friend in deed.

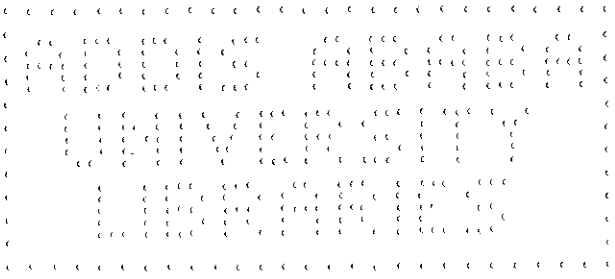
Acknowledgment

I would like to express my gratefulness to my instructor and advisor Dr. S. Kotelnikov for his continual assistance to bring this work to an end.

I extend my warmest gratitude for my relatives Ato Tadeous Getachew, W/o Menbere N. Tibeb and, Ato Andualem N. Tibeb for their financial support throughout my work.

I would also like to take this opportunity to appreciate the friendly staff members of the physics department of Addis Ababa University.

Last but not least, I am quite grateful to my parents Ato Getachew Abebe and W/o Mistre Nekatibeb for their underlying help throughout all the years I have been studying.



Contents

Abstract

Numerical values of wave functions, calculated from Hartree–Fock method, are nowadays available for many atoms and ions . Analytical forms of these functions are important in some cases. By studying Prony's method of representing numerical results analytically into sums of exponentials, it is shown that it is possible to apply the technique for approximating the numerically given radial part of atomic wave functions into sums of exponentials analytically. For each important step in Prony's method computer program is developed. The integrated program of the Prony's method is applied to the numerical radial self consistent field (SCF) wave functions of a closed-shell Cd^{++} ion and the exponents and coefficients for the exponentials of the approximated analytical functions are determined. Comparison between the analytical functions and the numerical functions show good agreement for wave functions with smaller principal quantum number value. A new algorithm for approximating tabulated numerical radial SCF atomic wave functions analytically into sums of exponentials is introduced. A computer program is developed for the proposed new algorithm and the technique is applied to 4d function of cadmium ion, Cd^{++}, and to 2s and 2p functions of Floride ion ,F^{-}. Exponents and coefficients of the exponentials of the approximating functions are determined. The approximated analytical forms of the numerical radial wave functions are used to reproduce numerical values of the function. The analytical and numerical functions are found to be in good agreement. These techniques can also be applied to analytically approximate numerical SCF functions of other ions.

1 Introduction

In the quantum mechanical treatment of many electron atoms, the total anti-symmetric wave functions describing the different atomic states are usually approximated by the sum of one, two or more determinants of one-electron wave functions, each being product of an atomic orbital (AO) and a spin function. The atomic orbitals are determined from the basic Schrodinger equation for the atom by means of variation principle as products of radial parts and spherical harmonics, and the best expressions for the former are obtained numerically by step-by-step integration of the Hartree-Fock-Equations by using the Self Consistent Field (SCF) technique developed by Hartree[HAR2].

Though the best single determinant atomic wave functions are given numerically, analytical forms of these functions become sometimes important in some areas of physics. For example investigations of the properties of crystals such as energy levels of of probabilities of different transitions can not be done numerically starting from the Hartree–Fock calculations for the cell of the crystal. It requires an intermediate step of the analytical representation of the numerical values of the functions of the ions building the cell[PET]. Therefore, for this and other not mentioned different cases it would be desirable to develop a method of approximating numerical functions analytically.

Different algorithms have been developed for analytical approximation of numerically tabulated wave functions sofar, most of which are the works of Slater and Lowdin. Though there are different types of the methods, each of them have their own advantages and shortcomings. For that reason the problem of developing new and efficient algorithms with no or little shortcoming for analytical approximation

of numerical wave functions is always at the spot.

In this thesis paper we have proposed two algorithms for analytical approximation of tabulated Self Consistent Field (SCF) numerical functions. The first is to modify our tabulated numerical wave functions such that we shall be able to use the existing Prony's method of approximating numerical wave functions analytically as a sum of exponentials. The second is to use a new method of successive iteration of a trial wave function until the desired accuracy is achieved.

In the second section the method of solving the Schroedinger equation of a system of electrons and the Hartree-Fock equations are discussed briefly. the third section is devoted to the study of the behaviour of our numerical data. The existing methods of approximating numerical functions analytically and those methods which we propose are discussed thoroughly in the fourth section. Section five deals with the application of our methods to specific ions, Cd^{++} and F^{-}, and interpretation of the results obtained. The numerical functions of Cd^{++} and F^{-} have never been approximated analytically[ATO]. Conclusions drawn from our work and outlook are given in sections 6 and 7 respectively.

2 Hartree–Fock method for the calculation of wave functions

2.1 Introduction

In quantum mechanics as in classical mechanics, there are few systems of physical interest for which the equations of motion can be solved exactly[LIN]. Especially, considering a system of electrons, an exact solution can be found only for a single-electron atom–the hydrogen. For the rest of the atoms, those having more than a single electron, the atomic Hamiltonian becomes complex to get a solution[HAR3]. The Helium atom is perhaps the simplest example of quantum mechanical many body problem which can not be reduced by separation of variables to a set of one body problems[FOCNES]. After carrying out a center of mass transformation, there still remains a non separable Schrodinger equation for two interacting electrons moving in a common central field. One of the most successful approach to integrating this equation has been the method of Hylleraas[HAR1] which consists of the variational calculation with a trial wave function depending explicitly on the separation between the two electrons, r_{12}.

Unfortunately, the method of Hylleraas can not be easily applied to systems with more than a very small number of particles[NES]. This is true primarily because the number of relative coordinates r_{ij} increases quadratically with the number of particles, while the number of independent coordinates r_i increases only linearly. For this reason, either there had to be significant developments in methods for dealing with Hylleraas wave function or it should be necessary to develop a sort of approximation scheme to find the eigen function for the complex Hamiltonian for an atom

consisting more than a few electrons. The development of the Hylleraas method by itself met a number of difficulties and the search for the solution of the complex Hamiltonian highly inclined to developing approximation schemes. A natural approximation in solution to the Hamiltonian of many electron atom is to assume that each electron moves in an average potential due to the nucleus and the other electrons[LIN]. This assumption leads to the independent - particle model, which essentially reduces the many electron model to the problem of solving a number of single particle equations. For an arbitrary potential, the solution of the single electron equations can have quite a complicated form, and they do not serve as a convenient basis for further calculations[LIN]. For this reason, one normally assumes that the potential is spherically symmetrical. The approximate Hamiltonian then describes a number of electrons moving in an average central field[ASC]. In a self consistent field approximation to the problem of the motion of electrons, the Hartree–Fock equations furnish the best set of one electron wave functions[SLA1]. The Hartree–Fock equations are improved versions of the ordinary Hartree method. The ordinary Hartree method follows from the variational principle, by setting up a wave function which is the product of one electron orbitals of various electrons, finding the average value of the Hamiltonian for this function and varying the one electron orbitals to minimize the energy[SLA5]. However such a product does not satisfy Pauli's exclussion princile[SLA4]. The simplest function which satisfies this principle is a determnental function, more complicated examples are linear combinations of determnental wave functions. The Hartree-Fock method arises from varying these orbitals to minimize the energy of such a function[SLA5]. The following three topics focus on describing the wave function for a many electron atom.

2.2 The Self Consistent Field (SCF) method

The Hamiltonian for an N-electron atom, with atomic number Z (by allowing N to be different from Z we give the probability of handling ions as well as atoms), is

$$(\mathrm{H})_{\mathrm{op}} = -\sum_{\mathrm{i}} \nabla_{\mathrm{i}}^2 - \sum_{\mathrm{i}} \frac{2\mathrm{Z}}{\mathrm{r_i}} + \sum_{\mathrm{i.j}} \frac{2}{\mathrm{r_{ij}}} \qquad (1)$$

where the summation over i is over all N-electrons, that over pairs is over each pair counted once, excluding the case $i = j$. The first sum in Eq.(1) is the sum of the kinetic energies for all the electrons, the second is the potential energy in the field of the nucleus, r_i being the distance from the nucleus to the i^{th} electron, and the last is the repulsive Coulomb potential of interaction between the i^{th} and the j^{th} electrons. The Hamiltonian of Eq.(1) is put in atomic units (see appendix C for explanation). The single–electron Schrodinger equation then becomes

$$\left[-\nabla_1^2 - \frac{2\mathrm{Z}}{\mathrm{r_1}} + \frac{2}{\mathrm{r_{12}}} \right] \upsilon_{\mathrm{a}}(1) = \mathrm{E_a} \upsilon_{\mathrm{a}}(1) \qquad (2)$$

The eigen function for the Schrodinger equation is obviously a function of the radius vector and the angles.More specifically we can write the eigen function for the Hamiltonian equation as

$$\upsilon_{nlm}(r, \theta, o) = R(r) Y_{m_l}^{l}(\theta, o) \qquad (3)$$

where

$R(r)$ is the radial wave function and

$Y_{m_l}^{l}(\theta, o)$ is the spherical harmonics.

The angular dependence of the solution of Schrodinger equation is the same as that for the Hydrogen atom[WEY]. The reason is that the separation goes through just the same step when the potential is considered to be spherically symmetrical and

the equations for the functions of θ and o are the same as in Hydrogen. That is we assume that the one-electron wave functions.or orbitals as we call them, have the form

$$\psi_{nlm}(r,\theta,o) = \frac{(-1)^{(m_l+|m_l|)/2}}{\sqrt{4\pi}}\sqrt{\frac{(2l+1)(l-|m_l|)!}{(l+|m_l|)!}}\times R_{nl}(r)P_l^{|m_l|}(cos\theta)exp(im_l o) \tag{4}$$

where

$$R_{nl}(r) = \frac{P_{nl}(r)}{r} \tag{5}$$

n is the principal quantum number

l is the azimuthal quantum number

m is the magnetic quantum number

$P_l^{|m_l|}$ are the associated legender functions

and where R_{nl} or P_{nl} is normalized so that

$$\int_0^\infty [\mathrm{P_{nl}(r)}]^2 \mathrm{dr} = 1 \tag{6}$$

The wave function for the system Ψ can be assumed to be a product of functions of the various electrons.In other words we start by assuming

$$\Psi(\mathrm{r_1\theta_1\phi_1 \ldots r_N\theta_N\phi_N}) = \psi_{\mathrm{n_1 l_1 m_{l_1}}}(\mathrm{r_1\theta_1\phi_1}) \ldots \psi_{\mathrm{n_N l_N m_{l_N}}}(\mathrm{r_N\theta_N\phi_N}) \tag{7}$$

where $\psi's$ on the right side of Eq.(7) are orbitals of the type (4), each one a function of the coordinates of a single electron. We assign the quantum numbers $n_1 l_1 m_{l_1} \ldots n_N l_N m_{l_N}$ of the various electrons according with the configurations in which we expect the atom to be found. In accordance to Pauli's exclusion principle, we may assign no more than two electrons to a given set of (nlm_l) values, of which we assume that one has m_s=1/2, the other has m_s= -1/2.

It is convenient to rewrite the Hamiltonian of Eq.(1) in the form

$$(H)_{op} = \sum_i f_i + \sum_{i,j} g_{ij} \tag{8}$$

where

$$f_i = -(\nabla^2)_i - 2Z/r_i \tag{9}$$

$$g_{ij} = 2/r_{ij} \tag{10}$$

The characteristic of the f_i's is that each operates on the coordinates of only one electron while each g_{ij} operates on the coordinates of two electrons: consequently they are called one-electron and two-electron operators respectively.

We now let the operators of Eq.(1) or Eq.(8) operate on the wave function of Eq.(7), multiply by the conjugate of the wave function and integrate over all values of the coordinate. f_i operates only on the orbitals $= \psi_{n_i l_i m_{l_i}}(r_i\theta_i o_i)$. In integrating over the coordinates of the electrons, we can integrate first over the coordinates of all electron except the i^{th}, each such integral giving unity on account of the normalization of the orbitals, hence the contribution of operator f_i to the average value of the Hamiltonian is

$$\int \psi^*_{\mathrm{n_i l_i m_{l_i}}}(\mathrm{r_i\theta_i\phi_i})\mathrm{f_i}\psi_{\mathrm{n_i l_i m_{l_i}}}(\mathrm{r_i\theta_i o_i})\mathrm{dv_i} = < \mathrm{i/f/i} > . \tag{11}$$

where the expression $< i/f/i >$ is an abbreviation for the integral.

Next we consider one of the two-electron operators g_{ij}. It operates on two orbitals, with indices i and j. We can then integrate first over the coordinates of all the electrons except these two and find the contribution of this operator g_{ij} to the

average Hamiltonian.

$$\int [\upsilon^*_{n_i l_i m_{l_i}}(r_i\theta_i o_i)\upsilon^*_{n_j l_j m_{l_j}}(r_j\theta_j o_j)$$
$$g_{ij}\upsilon_{n_i l_i m_{l_i}}(r_i\theta_i o_i)\psi_{n_j l_j m_{l_j}}(r_j\theta_j o_j)]_{av} dv_i dv_j \qquad (12)$$
$$=< ij/g/ij >_{av}$$

where the subscript av refers to the spherical average which we are to perform. and where $(ij/g/ij)$ is a notation for the integral of Eq.(12).

We may then combine these results and find that

$$(\mathrm{H})_{\mathrm{av}} = \sum_{\mathrm{i}} < \mathrm{i/f/i/} > + \sum_{\mathrm{i,j}} < \mathrm{ij/g/ij} >_{\mathrm{av}} \qquad (13)$$

There are three ingredients going in to the central field method[SLA1]. One of these is merely an approximate method of solving Schrodinger's Equation for the many–body problems of Z electrons. The other two are the postulates of the electron spin and of the Pauli's Exclusion Principle. In the central field approximation we replace the instantaneous action of all the electrons of an atom by a field in which each electron is assumed to be acted on by the averaged charge distribution of each other electron averaged by taking the quantity $\upsilon^*\upsilon$ for the corresponding wave function. Then following Hartree we proceed in the following fashion: we assume that each electron moves in such an averaged potential arising from the other electrons. We solve Schrodinger equation for an electron moving in such a potential. a simple problem on account of spherical symmetry.

2.3 The Hartree Approximation

For the moment we will ignore the anti-symmetry requirement of the wave function and and consider the mathematical problem posed on Eq.(3). As explained

in the previous section the Hamiltonian can be written as the sum of individual Hamiltonians depending only upon the individual coordinates. and also the total wave function could be written as the product of one–electron wave functions. each depending on the coordinates of one electron. Hartree suggested a variational calculation in which the wave function is approximated by such the product and the energy, $< \Psi|H|\Psi > / < \Psi|\Psi >$. is minimized[HAR1]. Even if the wave function is not so accurate we might hope that the energy would be a good approximation.

This variational procedure leads directly to the Hartree Equation from which the one-electron functions that minimize the energy may be determined. Hence we have

$$\left(\frac{-d^2}{dr_i^2} + \frac{l_i(l_i+1)}{r_i^2} - \frac{2Z}{r_i} + \sum_{j\neq i} \frac{2}{r_i} Y_0(n_j l_j . n_j l_j / r_i)\right) P_{n_i l_i}(r_i) = E_{n_i l_i} P_{n_i l_i}(r_i) \tag{14}$$

where

$$\mathrm{Y_0(n_j l_j . n_j l_j / r_j)} = \int_0^{\mathrm{r_i}} \mathrm{P^2_{n_i l_i}(r_i) dr_j} + \int_{\mathrm{r_.}}^{\infty} \mathrm{P^2_{n_. l_.}} \left(\frac{\mathrm{r_i}}{\mathrm{r_j}}\right) \mathrm{dr_j} \tag{15}$$

is used in finding the potential arising from the distribution. The Hartree equation is interpreted as the radial wave function for an electron moving in a spherical potential produced by the nuclear charge of Z units and by the spherically average charge distribution of all other electrons .

It is often convenient to rewrite the potential in the form

$$\mathrm{Z_{p_i}} = \mathrm{Z} - \sum_{\mathrm{j\neq i}} \mathrm{Y_0(n_j l_j . n_j l_j / r_i)} \tag{16}$$

so that the Hartree equation becomes

$$\left[\frac{-\mathrm{d}^2}{\mathrm{dr_i^2}} + \frac{\mathrm{l_i(l_i+1)}}{\mathrm{r_i^2}} - \frac{2\mathrm{Z_{p_i}}}{\mathrm{r_i}}\right] \mathrm{P_{n_i l_i}(r_i)} = \mathrm{E_{n_. l_.} P_{n_i l_i}(r_i)} \tag{17}$$

This is just the radial part of of the general Schrodinger equation for the orbital $\psi_{n_i l_i m_{l_i}}(r_i\theta_i\phi_i)$ in the self consistent field, which is

$$\begin{aligned}\left(-\nabla_i^2 - \frac{2Z_{p_i}}{r_i}\right)\psi_{n_i l_i m_{l_i}}(r_i\theta_i o_i) &= (H_i)_{op}\psi_{n_i l_i m_{l_i}}(r_i\theta_i\phi_i) \\ &= E_{n_i l_i m_{l_i}}\psi_{n_i l_i m_{l_i}}(r_i\theta_i\phi_i)\end{aligned} \tag{18}$$

where in Eq.(18) we have defined our own one electron operator as $(H_i)_{op}$. This is the one–electron Hamiltonian of an electron moving in the field of the nucleus, and of all electrons other than the i^{th}, spherically averaged.

Hartree's equations are of a form called integro–differential equations : The unknown function $P_{n_i l_i}(r_i)$ appears both in the ordinary way found in a differential equation and also in terms of certain integrals, the Y_0's. The only practical method of solving the set of simultaneous equations for the occupied orbitals is the method of iteration, or successive approximations. We take the P's determined from one stage of approximation and use them to calculate Y_0's. We then insert these values in to Eq(14) and solve these equations in the usual way. From these solutions we determine new P's. We then use these as the starting point of the next stage of approximation, calculating the Y_0's from them, and so on.

2.4 The Hartree–Fock approximation

The product wave function which we assumed in the Hartree method does not satisfy the Pauli's exclussion principle[GRE1,SLA5]. We must take linear combinations of each product wave function such that the total wave function changes sign under the interchange of any two electrons[SLA2]. The corresponding wave function can

be written as a Slater determinant.

$$\Psi(r_1, r_2, \ldots, r_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(r_1) & \psi_1(r_2) & \cdots & \psi_1(r_N) \\ \psi_2(r_1) & \psi_2(r_2) & \cdots & \psi_2(r_N) \\ \cdots & \cdots & \cdots & \cdots \\ \psi_N(r_1) & \psi_N(r_2) & \cdots & \psi_N(r_N) \end{vmatrix} \tag{19}$$

where $\frac{1}{\sqrt{N!}}$ is the normalizing constant.

Each of the spin-orbitals is assumed to be a product of a function of coordinate and a function of α or β of spin; that is, each spin orbital corresponds to a definite value of m_s.

The Hartree–Fock equations are obtained by varying the Ψ_i's so as to make the energy

$$\frac{1}{N!} \int \begin{vmatrix} \psi_1^*(r_1) & \cdots & \psi_1^*(r_N) \\ \cdots & \cdots & \cdots \\ \psi_N^*(r_1) & \cdots & \psi_N^*(r_N) \end{vmatrix} H \begin{vmatrix} \psi_1(r_1) & \cdots & \psi_1(r_N) \\ \cdots & \cdots & \cdots \\ \psi_N(r_1) & \cdots & \psi_N(r_N) \end{vmatrix} dr_1 \ldots dr_N \tag{20}$$

an extremum[SLA3] where the integration over dr's are interpreted to include summing over spins. The Hartree–Fock equations can then be written in the form

$$\begin{aligned} H_1\psi_i(r_1) + \left[\sum_{k=1}^{N} \int \psi_k^*(r_2)\psi_k(r_2)(2/r_{12})dr_2\right] \psi_i(r_1) \\ - \sum_{k=1}^{N} \left[\int \psi_k^*(r_2)\psi_i(r_2)(2/r_{12})dr_2\right] \psi_i(r_1) \\ = E_i\psi_i(r_1) \end{aligned} \tag{21}$$

Here H_1 is the kinetic energy operator for the electron of coordinate r_1, plus its potential energy in the field of all nuclei. $2/r_{12}$ is the Coulomb potential energy of interaction between electrons 1 and 2 expressed in atomic units. The extra term which has appeared in the Hartree–Fock equation is called the exchange interaction

as distinguished from the direct interaction which was present also in the Hartree equation. It is purely Coulombic in origin and arises from the correlated motion of the electrons due to the antisymmetry of the wave function[SLA1,SLA2]. The ψ's as before are assumed to depend on spin and coordinate, and the integrations over dr_2 include summation over spin, so that the exchange terms, the last ones on the left side of Eq(21) automatically vanish unless the functions ψ_i and ψ_k correspond to spins in the same direction.

3 The Hartree–Fock Numerical Radial Wave Functions of Cadmium ion, Cd^{++}, and Floride ion, F^-

The new algorithms for analytical approximation of numerical SCF wave functions, which we intend to develop, are to be applied on tabulated numerical radial Self Consistent Field (SCF) functions of Cd^{++} and F^- ions.

Analytical representation of these SCF numerical functions in some cases, particularly in our case, requires the values of these functions and their corresponding r values at the extreme points, at the nodes and also at the the flexibility points. More over the behaviour each numerical SCF functions near the center of the atom and far from the nucleus are merely important. In fact one can study the behaviour of the numerical SCF functions in comparison to the analytical approximate solution of the Hartree–Fock equations.

3.1 The assymptotic behaviour of the numerical SCF functions of Cd^{++} and F^-

The analytical behaviour of the function for small and large r, outside the sinusoidal region, can be determined by general arguments[FIN]. For every small r, the term $1/r^2$ in Eq(17) is large compared to the terms $E_{n_i l_i}$ and $2Z/r$. If we disregard the two latter terms, so that we have

$$\frac{d^2 P_{n_i l_i}(r)}{dr^2} - \frac{l_i(l_i+1)}{r_i^2} = 0 \qquad (22)$$

we find that we have two independent solutions

$$P(r) = r^{l+1}, r^{-l} \tag{23}$$

Since we do not allow a wave function which becomes infinite at $r = 0$, we have $P(r)$ which must behave like r^{l+1} at small r. Next we consider the case for very large values of r, for which the terms $2Z/r$ and $-l(l+1)/r^2$ can be neglected and the differential equation can be approximated by

$$\frac{d^2 P_{n_i l_i}(r)}{dr^2} + E_{n_i l_i} P_{n_i l_i}(r) = 0 \tag{24}$$

whose solution are

$$P_{n_i l_i}(r) = exp(\pm\sqrt{-E_{n_i l_i}}r) \tag{25}$$

If $E_{n_i l_i}$ is negative, the positive exponential will lead to a function becoming infinity which we must discard , so that our wave function must behave like a negative exponential (this is the case leading to negative exponential). Between the two limits which we have discussed, the wave function behaves in a generally sinusoidal fashion[LAN].

The numerical radial SCF functions of Cd^{++} and F^- whose electronic configurations are

$$\begin{aligned} & 1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^2 4p^6 4d^{10} \\ and \quad & 1s^2 2s^2 2p^6 \end{aligned} \tag{26}$$

respectively behave similar to the above discussion.

4 Analytical Representation of Wave Functions

4.1 Existing Methods for analytical approximation of numerical wave functions

The single determinant atomic wave functions obtained by numerical solution of Hartree-Fock equations are, of course, the best solutions for Schrodinger equation of system of electrons in the atom. However, it would be compulsory to use analytical forms of the wave functions, as discussed in the introductory part, for some physical problems. The required analytic forms of the wave functions can be derived in two ways[HYL], either directly by fixing parameters in a given analytic function and use the means of the variational principle or indirectly by approximating the numerically given Hartree–Fock functions in some way analytically, as was proposed by Slater.

Different attempts have been made, using both the variational[BRE] and approximation techniques, to develop efficient and least time consuming algorithms to approximate the given numerical SCF functions analytically. Zener and Morse-Young-Haurwitz[STE] determined variationally the values of parameters in determinants of one electron wave functions for various configurations involving $1s, 2s$ and $2p$ electrons.

Green, Mulder, Levis and Woll.Jr have also used the variation technique to determine analytical wave functions of the type[HYL]

$$R^N(r) = Nre^{-Zr}(1 + CZ^2r^2) = Nre^{-Zr}(1 + kr^2) \qquad (27)$$

where $k = CZ^2$, which is a two parameter function for $1s^2$ configuration of Helium type atom.The analytical wave functions were discussed for elements from H^- to Carbon.

Green. Mulder. Lewis and Woll,Jr substituted the normalized ground state wave function of Eq.(27)

$$\Psi^N = \frac{1}{4\pi}\frac{1}{r_1}R^N(r_1)\frac{1}{r_2}R^N(r_2) \tag{28}$$

into the energy integral and determined the values of the parameters, C and Z, giving the lowest energy. In determining the values of the parameters, value of k were chosen, and the energy expression for each k was then minimized with respect to Z. From the table of k–versus the energy the particular k–which gives the lowest energy to five decimal places was selected[HYL]. Green et al. had actually found excellent approximations for their analytic functions since energies computed from the varied wave functions and from the corresponding Hartree–Fock quantities had become comparable. However, their result deviates considerably in case of H^- from the corresponding energy calculated from the numerically given Hartree–Fock wave function.

Additional investigation of the accuracy of varied analytic orbitals shows that the Zener, and Morse–Young–Haurwitz function containing only a few exponents represents rather poor approximations of the self-consistent -fields and hence the deviations are appreciable, particularly at large distances.

Slater showed that it was possible to obtain a good result by approximating the numerically given Hartree-Fock tables by using a sum of the form $C_k r^n \epsilon^{-a_K r}$[LOW1]. For the lowest functions, slightly modified by Lowdin this gives the following expanssions:

$$P_{1s}(r) = r\Sigma_k A_k exp(-a_k r) \tag{29}$$

$$P_{2s}(r) = r\Sigma_k A_k exp(-a_k r) - r^2\Sigma_k B_k exp(-b_k r) \tag{30}$$

$$P_{2p}(r) = r^2\Sigma_k B_k exp(-b_k r) \tag{31}$$

	Maximum error in units of 10^{-3}		
r-interval	Rb^+_{2s}	Rb^+_{3s}	Rb^+_{3p}
0.00	14	-41	-9
0.04	-30	$\pm$17	-46
0.20	-15	$\pm$28	14
0.50	2	25	-31
1.00		14	5

Table 1: Survey of the maximum errors in different intervals in the analytic SCF- functions of Rb^+ (without exchange) given by Slater[LOW2] as an example of the accuracy of his graphical method

$$P_{3s}(r) = r\Sigma_k A_k exp(-a_k r) - r^2\Sigma_k B_k exp(-b_k r) + r^3\Sigma_k C_k exp(-c_k r). \quad (32)$$

$$P_{3p}(r) = r^2\Sigma_k B_k exp(-b_k r) - r^3\Sigma_k C_k exp(-c_k r) \ldots etc. \quad (33)$$

where the exponents $a_k, b_k \ldots$ and the coefficients $A_k, B_k \ldots$ may be different for each orbital. The value of of these parameters is determined by numerical method, which is the development of Slater's graphical method. The accuracy of the analytic Self Consistent Field functions obtained by Slater's graphical method may be illustrated by his own example for Rb^+ in table I. Even if the maximum error is of the order of 45 x 10^{-3}, the approximation is certainly good for many applications. The analytic SCF functions, calculated from Slater's exponents for other atoms of the periodic system, have also errors of the order of 20 X 10^{-3}.

Lowdin has also fitted a number of Hartree–Fock numerical functions very closely by using slightly more generalized sums of functions of the Slater type than those originally employed by Slater himself[LOW1]. In his investigation of the alkali

AO	k=	1	2	3	k=	1	2	3
$Cl^-(3p)$	b_k	4.2435	8.4758	22.314	c_k	0.92426	1.6658	2.9859
	B_k	9.0441	26.493	2.49	C_k	0.07099	1.3955	8.4230

Table 2: Exponents and coefficients in an analytical SCF functions of the Slater type approximation for $Cl^-(3p)$ with exchange[LOW1].

chlorides. Lowdin used tabulated $3p$ function of Cl^- with exchange and by using six exponentials he obtained a fit as good as 1.5×10^{-3}[7], that is , the analytic SCF-function had the same accuracy as the numerical function itself. Lowdin had also produced analytic SCF-functions for two other closed-shell ions in the same way, namely, F^- without exchange and Na^+ with exchange. His results fitted with slightly less accuracy with the numerical ones.

It is worth to consider a point here. By the generalized Slater method described above, it is possible to approximate SCF functions analytically from the numerically given tables with any desired accuracy, but, even if the technique is simple, the computations are still time consuming and rather tedious[LOW1]. It was found important to reexamine the basic methods for further improvement since the periodic table had to be investigated on a large scale basis in order to obtain analytic SCF-functions having errors of the order of magnitude (0.001-0.002) [LOW2].

It was this time that Lowdin tried to develop the idea of constructing an analytical fit for the given numerical SCF function to a higher degree of simplicity and accuracy.

The basic idea proposed by Lowdin was to divide the numerical SCF function $P(r)$ by a polynomial having the same nodes as the function it self and then to

express the resulting monotonously decreasing quotient function say $g(r)$ as a sum of exponentials[LOW2].

The problem of developing a function $g(r)$ given in the form of a series of entries $g_0, g_1, g_2, \ldots g_n \ldots$ for equidistant points $r = r_0, r_0 + h, r_0 + 2h, \ldots r_0 + nh, \ldots$ into sum of exponentials

$$g(x) = \sum_{\nu}^{p} A_\nu \exp(-a_\nu r) \tag{34}$$

is in principle exactly possible. Introducing the notation $k_\nu = exp(-a_\nu)$ and $\alpha_\nu = A_\nu exp(-a_\nu r_0)$.one can get

$$g_n = \alpha_1 k_1^n + \alpha_2 k_2^n + \ldots + \alpha_p k_p^n. \tag{35}$$

which implies that the series $g_0, g_1, g_2, \ldots g_n \ldots$should be expressed as the sum of p geometrical series with the quotients $k_1, k_2, \ldots k_p$. Assuming that the quotients satisfy the algebraic equation

$$c_0 + c_1 k + c_2 k^2 + \ldots + c_p k^p = 0. \tag{36}$$

and by using Eq.(34) and Eq.(35).one obtains further

$$\begin{array}{lllll} c_0 g_n & + c_1 g_{n+1} & + \ldots & c_p g_{n+p} & = 0 \\ c_0 g_{n+1} & + c_1 g_{n+2} & + \ldots & c_p g_{n+p+1} & = 0 \\ \ldots\ldots & \ldots\ldots & \ldots & \ldots\ldots & \ldots\ldots \\ c_0 g_{n+p} & + c_1 g_{n+p+1} & + \ldots & c_p g_{n+2p} & = 0. \end{array} \tag{37}$$

These equations form together a linear system of order $(p + 1)$ in the unknown coefficients $c_0, c_1, c_2, \ldots c_p$, and this system has a non trivial solution only if its

determinant vanishes:

$$\begin{vmatrix} g_n & g_{n+1} & \cdots & g_{n+p} \\ g_{n+1} & g_{n+2} & \cdots & g_{n+p+1} \\ \cdots & \cdots & \cdots & \cdots \\ g_{n+p} & g_{n+p+1} & \cdots & g_{n+2p} \end{vmatrix} = 0 \qquad (38)$$

Generally, Lowdin used the idea of Hankel Determinant to solve the above determinant and to find the values of k_ν (ν runs from 1 to p) from which he calculated the coefficients A_ν and a_ν (ν again runs from 1 to p). The method used above is called **exact method.**

Lowdin used his exact method to express the numerical SCF functions for $Na^+(2p)$ analytically by three exponentials. He determined the six parameters involved from the six consecutive entries of the function $g_{2p} = P_{2p}/r^2$, where P_{2p} represents the numerical SCF function of $Na^+(2p)$. His analytic SCF function can then be written as

$$g^*_{2p}(r) = 3.6164exp(-2.1880r + 15.66exp(-3.7288r) + 18.729exp(-6.8864r). \quad (39)$$

The accuracy of Lowdin's SCF analytic function when compared to the numerical SCF function can be judged from his own words[LOW2]: ..."even if the corresponding analytic function passes nicely through the six given points, it deviates considerably in a large number of other points...". for this reason Lowdin himself devised another method which he called, **method of successive approximation.** to giving a single analytic approximation of almost perfect accuracy, that is , one which deviates from the given SCF function by quantities smaller than 0.001 – 0.002. by utilizing only some of the simplest formulas of the exact theory.

In the Lowdin's approximation method one works inwards from the tail $r = \infty$

towards the origin and in each step two exponentials are determined. The method of **successive approximation** differs basically from the **exact theory** in that in the former case nodeless quotients

$$g_{nl} = P_{nl}(r)/r^{l+1}(r'_0 - r)(r''_0 - r)\ldots \qquad (40)$$

where

$P_{nl}(r)$ is the numerically tabulated SCF function

$r'_0, r''_0 \ldots$ are the zero points of the numerical SCF function

and n and l are the usual quantum numbers

are approximated to analytic functions.

The advantage of this method over the **exact theory** is that it removes cancellation of significant figures from the numerically given SCF function, which was the basic difficulty in the exact theory. Moreover excellent results with high accuracy are found with this method: best example is the analytic SCF function approximated for Argon using this method. Unfortunately in applications to molecular and crystal theory, analytic SCF functions containing several exponentials will lead to rather cumbersome calculations[LOW2]. So it would be desirable to have less accurate analytic approximations containing only one, two or three exponentials.

4.2 Looking for new methods of representing numerical SCF functions analytically

The problem of finding an analytical fit to the numerically given SCF functions is complicated by the fact that at least from the practical point of view , as discussed in the previous section, there does not seem to exist a uniquely defined "best approximation" but instead a whole family of analytic functions of about the same accuracy.

The objectives of this paper, as mentioned in the introduction, is to develop algorithms to approximate a numerically given radial wave function analytically. For that reason two methods are proposed in this paper. The first is to use the known Prony's method of representing numerical solutions as a sum of exponentials and the other is to use a method of successive iterations keeping the maxima and minima of the numerical wave function fixed. Both the methods are discussed thoroughly in the coming sections.

4.2.1 A method of analytical approximation relying on Prony's method

Prony's method is an algorithm which can be used to approximate tabulated equidistant numerical functions analytically as a sum of exponentials. Consider a system whose solution can be written in the form

$$g(r) = \Sigma_{\nu=1}^{n} c_\nu e^{a_\nu r} \tag{41}$$

or simply

$$g(r) = c_1 e^{a_1 r} + c_2 e^{a_2 r} + \ldots + c_n e^{a_n r} \tag{42}$$

If we make the substitution,

$$p_k = e^{a_k} \tag{43}$$

then we can write,

$$g(r) = c_1 p_1^r + c_2 p_2^r + \ldots + c_n p_n^r \tag{44}$$

We suppose that a linear change of variables has been introduced, in advance, in such a way that values of $g(r)$ are specified at N equally spaced points $r = 0, 1, ..., N-1$.

If the set of points are to fit the expression (44), then it must be true that.

$$\begin{aligned}
c_1 \quad &+ \quad c_2 \quad + \quad \ldots \quad + \quad c_n \quad = \quad g(0), \\
c_1 p_1 \quad &+ \quad c_2 p_2 \quad + \quad \ldots \quad + \quad c_n p_n \quad = \quad g(1), \\
c_1 p_1^2 \quad &+ \quad c_2 p_2^2 \quad + \quad \ldots \quad + \quad c_n p_n^2 \quad = \quad g(2), \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
c_1 p_1^{N-1} \quad &+ \quad c_2 p_2^{N-1} \quad + \quad \ldots \quad + \quad c_n p_n^{N-1} \quad = \quad g(N-1)
\end{aligned} \tag{45}$$

If the constants $p_1, \ldots, p_n$ are known, then this set comprises n linear equations in n unknowns and can be solved exactly. If $N > n$, then the unknown constants are found approximately by the least squares technique.

If, however, the p_k must also be determined, at least $2n$ equations are needed which are now nonlinear in the p's. This difficulty can be minimized by the method which we will describe next.

Let $p_1, \ldots, p_n$ be the roots of algebraic equation

$$\mathrm{p^n - d_1 p^{n-1} - d_2 p^{n-2} - \ldots - d_n = 0.} \tag{46}$$

so that the left-hand member of (46) is identified with the product $(p - p_1)(p - p_2) \ldots (p - p_n)$. In order to determine the coefficients $d_1, \ldots, d_n$, we multiply the first equation of(45) by d_n, the second by $d_{n-1}, \ldots$, the n-th equation by d_1 and the $(n+1)th$ equation by -1 and add the results. If we make use of the fact that each p satisfies (46), the result is seen to be of the form,

$$g(n) - d_1 g(n-1) - d_2 g(n-2) - \ldots = 0 \tag{47}$$

A set of N-n-1 equations can be set up in the same way by starting instead with the second, third...., (n-1) equations. In this way we find that (45) and (46) imply the

N-n linear equations:

$$\begin{array}{lllllll} g(n-1)d_1 & + & g(n-2)d_2 & + & \ldots & + & g(0)d_n & = & g(n), \\ g(n)d_1 & + & g(n-1)d_2 & + & \ldots & + & g(1)d_n & = & g(n+1), \\ \ldots & & & & & & & & \\ g(N-2)d_1 & + & g(N-3)d_2 & + & \ldots & + & g(N-n-1)d_n & = & g(N-1) \end{array} \quad (48)$$

Since the ordinates g(k) are known, this set can be solved directly for the n ds of $N = 2n$ or solved approximately by the method of least squares if $N > 2n$.

After the d's are determined, the p's are found as the roots of equation(46). They may be real or complex. The equations (45) then become a set of linear equations in the coefficients c_i. The c's can be determined from any set of n equations or, preferably, by applying the least squares technique to the entire set.

Thus the nonlinearity of the system is concentrated in the single algebraic equation(46). The technique described is known as **Prony's method**[BEL]. Most parts of the calculations in Prony's method are tedious for manual calculations. Therefore computer programs are developed for the important parts of the method.

The radial SCF functions of cadmium ion Cd^{++} are approximated as a sum of exponentials using Prony's method and the exponents and coefficients of the exponents are determined.

4.2.2 Method of Successive Iterations With Fixed Extremum Points (MSIFEP)

The second type of algorithm we propose for analytical approximation of SCF numerical wave functions is a method of successive iteration of an assumed trial function until the required tolerance between coefficients and exponents of exponentials

is achieved.

We will start from the assumption that the self consistent field (SCF) is of the form[8]

$$f(r) = constr^{\alpha}\epsilon^{-\beta r} \tag{49}$$

where

α is an integer

β is a function of r. In the actual computations. β is found rather slowly varying so that we can consider it as a constant function.

For the investigation of of properties of crystals. wave functions of of the ions must be given as products of polynomial function and exponential functions.

If we have N-extremum points, we will approximate our function as a sum of N-exponentials

$$f(r) = \sum_{i=1}^{N} A_i r^{\alpha}\epsilon^{-\beta_i r} \tag{50}$$

where

A_i and β_i are constants to be determined. For a first step approximation we will express our function by only one of the extremum points. This simply means that our function can be expressed from the first extremum point as

$$f(r) = A_1 r^{\alpha}\epsilon^{-\beta_1 r} \tag{51}$$

or from the second extremum as

$$f(r) = A_2 r^{\alpha}\epsilon^{-\beta_2 r} \tag{52}$$

and from the Nth extremum as

$$f(r) = A_N r^{\alpha}\epsilon^{-\beta_N r} \tag{53}$$

Considering Eq.(3) the first extremum value, f_{m1} is situated at a point $r = r_{m1}$. The derivative of (5) at $r = r_{m1}$ is obviously zero. Therefore we will have

$$0 = A_1 \alpha r_{m_1}^{\alpha-1} e^{-\beta_N r_{m_1}} - \beta_1 r_{m_1}^{\alpha} A_N e^{-\beta_N r_{m_1}} \tag{54}$$

from which we get

$$\beta_1 = \frac{\alpha}{r_{m_1}} \tag{55}$$

substituting Eq.(7) in to Eq.(5) at the extremum point we get the relation

$$f_{m_1} = A_1 r_{m_1}^{\alpha} e^{-\beta_1 r_{m_1}} \tag{56}$$

and this equation yields

$$A_1 = \frac{f_{m_1}}{r_{m_1}^{\alpha} e^{-\beta_1 r_{m_1}}} \tag{57}$$

β_1 and A_1 are the first step approximate values and let's represent them by $\beta_1^{(1)}$ and $A_1^{(1)}$. Since we have N such type of constants, the first approximate values for As and βs in the general case is given by

$$\begin{aligned} \beta_i^{(1)} &= \frac{\alpha}{r_{m_i}} \\ A_i^{(n)} &= \frac{f_{m_i}}{r_{m_i}^{\alpha} e^{-\beta_i r_{m_i}}} \end{aligned} \tag{58}$$

If we express our approximating function as a sum of all functions obtained from each extremum points we see that this function is different from the numerical function at the extremum points. Suppose that the difference between the numerical function $f(r)$ and the approximating function constructed from all the extremum points except the first extremum point be equal to the approximated function at the first extremum point. This simply means that

$$A_1^{(1)} r^{\alpha} e^{-\beta_1^{(1)} r} = f(r) - \sum_{i \neq 1}^{N} A_i^{(1)} r^{\alpha} e^{-\beta_i^{(1)} r} \tag{59}$$

The derivative of Eq.(12) at the first maximum point is zero.

$$A_1^{(1)}\alpha r_{m_1}^{\alpha-1}e^{-\beta_1^{(1)}r_{m_1}} - \beta_1^{(1)}A_1^{(1)}r_{m_1}^{\alpha}e^{-\beta_1^{(1)}r_{m_1}} = 0 - \sum_{i\neq j}^{N} \tag{60}$$

where $\sum_{i\neq j}^{N}$ represents the derivative of the Sum at the extremum point. Let us label the expression $A_1^{(1)}r_{m_1}^{\alpha}e^{-\beta_1^{(1)}r_{m_1}}$ by a letter T_1. Then from Eq.(13) we get the relation

$$\beta_1^{(2)} = \frac{\alpha}{r_{m_1}} + \frac{\sum_{i\neq j}^{N}}{T_1} \tag{61}$$

$\beta_1^{(2)}$ is the second step approximate value of β_1. Replacing $\beta_1^{(1)}$ by $\beta_1^{(2)}$ in Eq.(12) and solving the equation at the first maximum point we find the second approximate value of A_1.

$$A_1^{(2)} = \frac{f_{m_1} - \sum_{i\neq 1}^{N} A_i^{(1)}r_{m1}^{\alpha}e^{-\beta_i^{(1)}r_{m_1}}}{r_{m1}^{\alpha}e^{-\beta_i^{(1)}r_{m_1}}} \tag{62}$$

We improve the constants As and βs not only for the A_1s andβ_1s but for all A_is andβ_is where i runs from 1 to N. Therefore, in the general case we have

$$\beta_j^{(2)} = \frac{\alpha}{r_{m_j}} + \frac{\sum_{i\neq j}^{N}}{T_j} \tag{63}$$

$$A_j^{(2)} = \frac{f_{m_j} - \sum_{i\neq j}^{N} A_i^{(1)}r_{mj}^{\alpha}e^{-\beta_i^{(j)}r_{m_j}}}{r_{m_j}^{\alpha}e^{-\beta_j^{(2)}r_{m_j}}} \tag{64}$$

Now we have improved the values of A_js andβ_js by one step. From these values we then calculate other improved values of A_js andβ_js for a third approximation. This iteration continues until the difference between consecutive A_js and β_js is very very small. We call this method as **a method of successive iteration with fixed extremum points (MSIFEP).**

5 Discussion of Results

5.1 Prony Method

Prony's method of approximating numerical functions as a sum of exponentials analytically starts from using equally spaced and monotonously decreasing numerical functions[BEL]. The wave function of atoms, as can be seen from the graphs of Cd^{++}, are, however, oscillating by nature. (See Fig.1 and Fig.2). Moreover, numerical

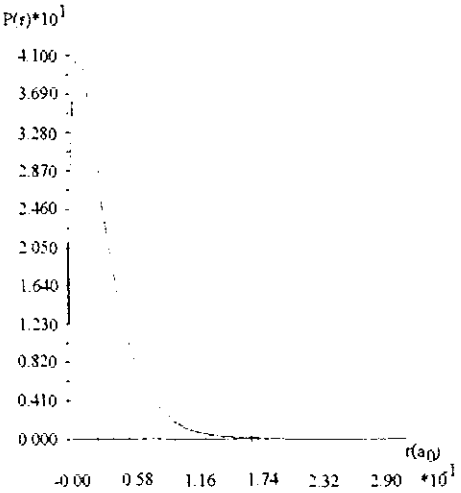


Figure 1: *The 1s atomic wave function of Cd^{++}*

wave functions of cadmium are not tabulated at equally spaced points. Therefore, to use Prony's method of approximating numerical functions analytically as a sum of exponentials, equally spaced, monotonously decreasing and nodeless functions had to be prepared from the tabulated oscillating numerical SCF functions. Preparation of equally spaced functions as a form of entries $P_0, P_1, \ldots P_n$ for equidistant points $r = r_0, r_0 + h, r_0 + 2h \ldots, r_0 + nh$, where the $P_i s$ are the sought equally spaced numerical wave functions and h is the step or gap between the functions, from the given numerical oscillating functions was the most difficult task we encountered. The first attempt made was to linearly interpolate equally spaced functions from the given numerical functions. This is, in principle, exactly possible. However, to get

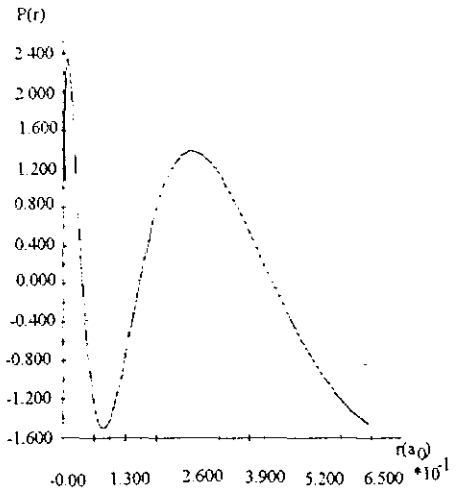


Figure 2: *The 4s atomic wave function of Cd^{++}*

accurate linearly interpolated equally spaced functions which are in agreement with the numerical functions we were supposed to take as many close points as possible. The consideration of close points, on the other hand, led to the accumulation of large size data which became a constraint for least computation time and use of limited computer memory in the progressive computations. For that reason the linear interpolation technique[RIV] had to be abandoned and the spline interpolation technique was used in effect.

The resulting equally spaced numerical functions calculated from the cubic spline interpolation technique (see computer program Prony-1) are in good agreement with the not-equally spaced numerical functions. As an evidence for this the spline interpolated and equally spaced functions and their primary numerical functions are plotted together for $1s$ and $4d$ orbitals of cadmium ion Cd^{++}. From Fig.(3) and Fig.(4) we see that the two lines on the graph are running together. Numerical radial wave functions can be made monotonously decreasing and nodeless functions using the previously stated relation

$$g_{nl}(r) = \frac{P_{nl}(r)}{r^{l+1}(r_0' - r)(r_0'' - r)} \cdots \tag{65}$$

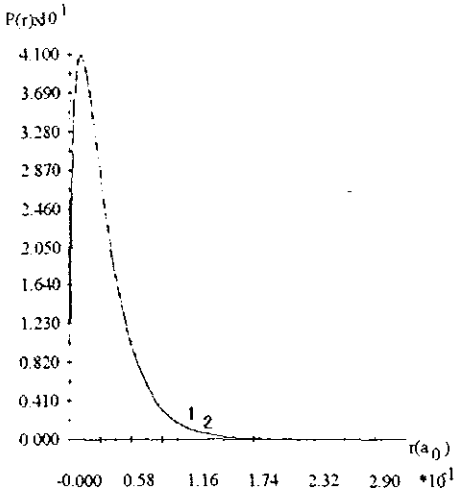


Figure 3: *The spline interpolated equally spaced (1) and the not equally spaced (2) functions of 1s orbital. The two functions are running together at all points.*

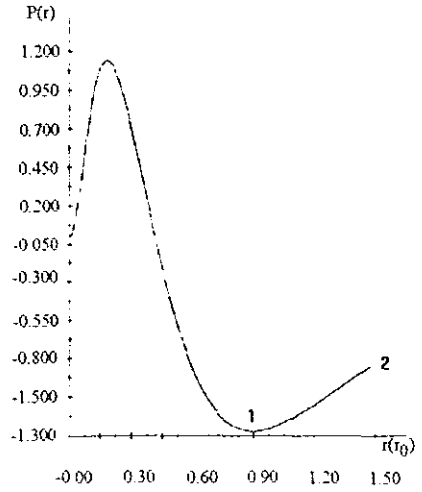


Figure 4: *The spline interpolated equally spaced (1) and the not equally spaced functions (2) of 4d orbital The two functions are running together at all points.*

where $P_{nl}(r)$ is the given oscillating numerical radial wave function. $r_0', r_0'', \ldots$ are the zero points of the radial numerical wave function and n and l are the usual quantum numbers. All the nine one electron orbital functions of Cd^{++} are made nodeless and monotonously decreasing. Monotonously decreasing nodeless functions of 1s and 4d orbitals produced using Eq.(65)(see also program Prony-4) are plotted on Fig.(5) and Fig.(6) respectively for observation. So now we are successful in preparing equally spaced and monotonously decreasing functions from the given oscillating and not equally spaced numerical functions. From this point on we can use Prony's method to approximate the equally spaced monotonously decreasing

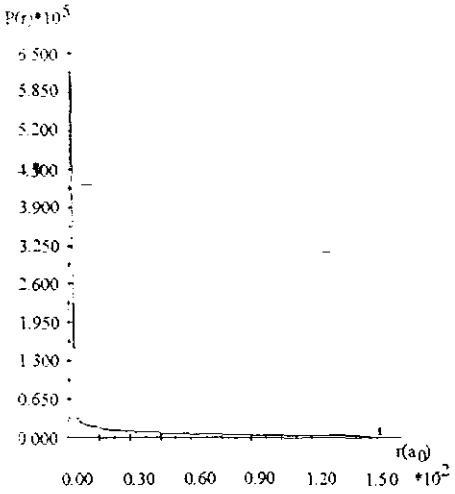


Figure 5: *Monotonously decreasing function of orbital 1s*

functions analytically as a sum of exponentials. The steps of Prony's method of approximation starting from the end are:

1. Calculation of **d-values** of Eq(50).

For this step we have developed 2 programs. In Eq.(50) $N > 2n$. and these types of equations are solved by the least square approximation technique. Using the least square approximation **normal equations**[VAN,PUZ] are constructed. The solution for the normal equations is found using the Gaussian ellimination method. The programs for lest square approximation and Gauss elimination are given in appendix A. Prony-5 and Prony-6 respectively.

2. Calculation of roots of the polynomial

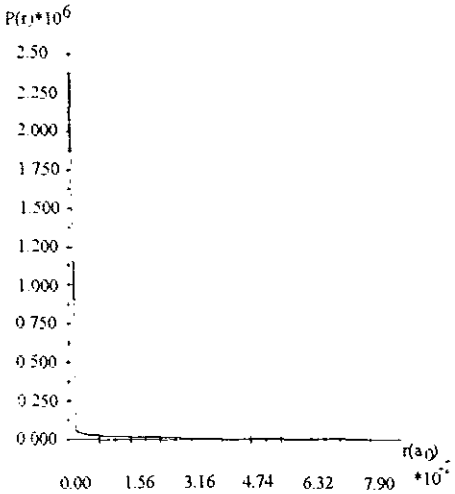


Figure 6: *Monotonously decreasing function of orbital 4d*

function or **p-points** of Eq(48) After the calculation of **d values** the polynomials of Eq.(48) are constructed by using them. The solution for the constructed polynomials is found by the secant method. The program developed for the method. Prony-7, is given in appendix A. Computer program Prony-6 found in appendix A gives regions of the solutions of the polynomial or the primary guesses which are used by the secant method[GRE2].

3. Calculation of exponents

From the zero points of the polynomials calculated using the secant method, the exponents of the exponentials or **a-values** are calculated using Eq.(45). The program for this equation, Prony-8, is given in appendix A.

4. Calculation of coefficients or **c-values** of Eq.(47)

Similar to step one least square approximation and Gauss elimination are used to find **c-values**. The programs for these calculations, Prony-9 and Prony-10 respectively , are given in appendix A.

All the nine equally spaced and monotonously decreasing numerical SCF wave functions of cadmium ion (Cd^{++}) are approximated analytically as a sum of exponentials using Prony's method and the results of the coefficients and exponents of the exponentials are given on table(6).

It seems routine to discuss the procedure and problem tackling techniques encountered while finding the coefficients and exponents for all the nine wave functions of Cd^{++}. For that reason only those two functions which are approximated analytically with minimum error (1s) and the other approximated with larger error (4d) are discussed thoroughly.

AO	i	a_i	c_i
$Cd^{++}(1s)$	1	-108.25	12593.93
	2	-947.36	25838.90
	3	-4183.50	31292.23
	4	-24929.69	600427.41

Table 3: Coefficients and exponents of exponentials for the 1s analytic function of Cd^{++} ion approximated using Prony's method

5.1.1 Analytical approximation of SCF numerical 1s and 4d functions of Cd^{++} using Prony's method

The 1s SCF numerical functions of cadmium ion (Cd^{++}) has no nodes (see Fig.(1)) and also program zero-point calculation). So the spline interpolated equally distant function is made monotonously decreasing by just dividing the oscillating function $P_{nl}(r)$ by the corresponding radius r. In this case $l = 0$. The prepared monotonously decreasing function is then approximated analytically in four exponents. The coefficients, c_is, or c-values and exponents, a_is, or a-values of the exponentials registered after the analytical approximation using Prony's method are given on table(3).

This simply means that the 1s numerical functions of Cd^{++} can be given analytically as

$$\begin{aligned} P^{*}_{1s}(r) = \; & r[12599.4\exp(-108.3r) + 25838.\exp(-947.4r) + \\ & 31292.2\exp(-4183.5r) + 600427.4\exp(-24929.7r)] \end{aligned} \tag{66}$$

where $P^{*}_{nl}(r)$ represents analytical form of the numerical function $P_{nl}(r)$.

To check the accuracy of the resulting analytic function, Eq(66), numerical functions were reproduced from the approximated analytical functions. Comparison

between the analytical and numerical functions shows that their differences are appreciable at the tail region and the functions are in good agreement at the inner regions of the graph. The error (by the error we mean abs($P_{nl} - P^{*}_{nl}$)) at the maximum values of the wave functions is 1.9 which is very small compared to the values of the analytical function (40.9) and the numerical function (42.8). We could not actually expect an exact agreement because, for one thing we do not exactly know in how many exponents should we approximate analytically the given monotonously decreasing function and for the other we can not tell exactly whether our monotonously decreasing function is an exactly exponentially decreasing function or not. In fact the computer programs developed for the Prony's method algorithms were applied to approximate exponentially decaying numerical functions with known number of exponents. The numerical functions reproduced from the approximated analytic functions were, then, found to be in excellent agreement with the initial data showing that the method and the programs work correctly. The big difference observed

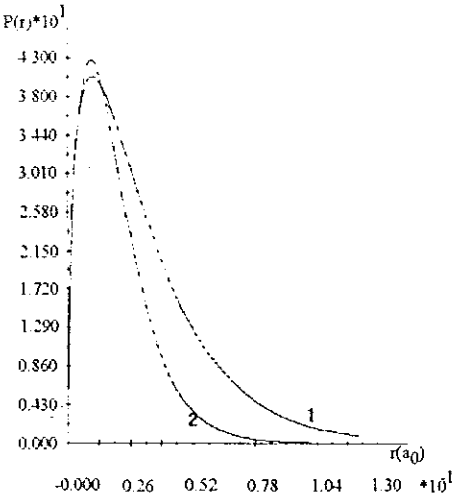


Figure 7: *Numerical (2) and analytical (1) functions of 1s orbital. The difference between the two functions is appreciable at most points.*

between the two functions in our case at the tail region is in fact due to the reason that less points are considered from the tail regions in the calculation of d-values and c-values (see program c-least and d-least) using the least square technique method.

AO	i	a_i	c_i
Cd^{++}(1s)	1	-60.69	5890.60
	2	-278.57	12622.84
	3	-1920.25	31950.78
	4	-1777.51	413418.63

Table 4: Coefficients and exponents of exponentials for the 1s analytic function of Cd^{++} ion approximated using Prony's method with weights

Therefore, the difference between the functions can be narrowed by improving the resulting analytical functions. This can be done by giving more weight to the tail region rather than the inner region at the calculation of d-values and c-values of the least square technique. The weights imposed on the equations at the spot in the least square programs can be varied until the desired accuracy is achieved.

Whence, for the 1s function of cadmium ion (Cd^{++}) more weight was given to the tail region in the least square programs. The approximating equation in the d-least program was multiplied by a cubicaly increasing function, i^3 and the approximating equation in the c-least program was given a linear increase in weight by multiplying it by i, where i is an integer ranging from 1 to the number of data points (N).

The newly calculated coefficients and exponentials registered while approximating the numerical function of the 1s orbital using the weights imposed are tabulated in table(4). This means that the 1s orbital numerical function of Cd^{++} can be written analytically in the new coefficients and exponents as

$$\begin{aligned} P^{*}_{1s}(r) = \; & r[5890.6\exp(-61r) + 12622.8\exp(-278.6r) + \\ & 31950.8\exp(-1920.3.r) + 413411.8\exp(-1777.51r)] \end{aligned} \tag{67}$$

where $P^{*}_{nl}(r)$ represents analytical form of the numerical function $P_{nl}(r)$. Comparison between the numerical and analytical functions shows that the two functions are in good agreement at most points. Especially around the maximum value of the

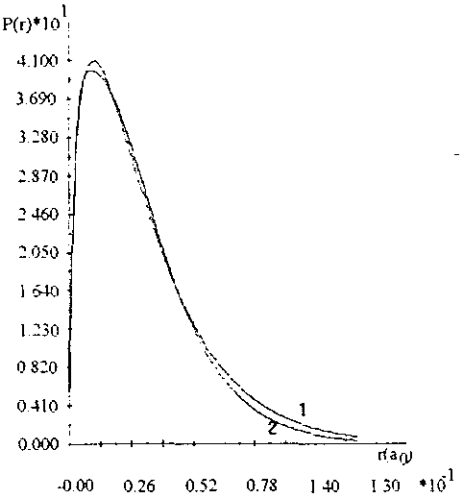


Figure 8: *Numerical (1) and analytical (2) functions of 1s orbital with weights. Difference between functions reduced appreciably.*

wave functions–which are our regions of interest– the maximum error is 1.0 which is very small compared to the values of the analytical and numerical functions which are 40.9 and 39.9 respectively. The errors around the tail region have also reduced appreciably. The 4d orbital numerical function of cadmium ion (Cd^{++}) has one node or zero point at $r_0 = 0.451$ in atomic units . Since l=2 for the 4d orbital the oscillating function can be made monotonously decreasing by dividing $P_{nl}(r)$ by the product of r^3 and the difference between r and the zero points. The not-equally spaced function is made equally spaced function using the spline interpolation technique. Then the equally spaced monotonously decreasing functions are approximated in four exponents using Prony's method. The values of the coefficients and exponentials are tabulated on table (5).

Hence, the numerical wave function of Cd^{++} 4d orbital can be written approxi-

AO	i	a_i	c_i
Cd^{++}(4d)	1	-26.93	124257.11
	2	-185.15	191473.89
	3	-768.21	299592.47
	4	-4625.80	12801105.74

Table 5: Coefficients and exponents of exponentials for the 4d analytic function of Cd^{++} ion approximated using Prony's method.

mately as

$$P^{*}_{4d}(r) = r^3(0.451 - r)[12425.7exp(-26.9r) + 191473.9\epsilon xp(-185.2r) + 299592.5\epsilon xp(-768.21r) + 12801105.7\epsilon xp(-4625.8r)] \quad (68)$$

Comparisons between analytical and numerical functions show that the two functions are not in agreement. The error at the maximum point of the function is 114

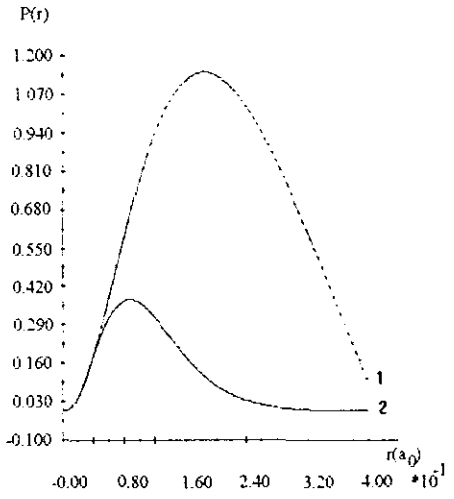


Figure 9: *Numerical (1) and Analytical (2) functions of 4d orbital without weight*

which is about 12% in relative forms (see Fig.(9)). Attempts were made to narrow the difference between the two functions by improving the results of calculations of the approximated wave function giving different weights to the tail and inner regions

the previous mentioned least square programs. However the improvements in the values of the functions were insignificant. Even a heavy weight of $1/(i+1)^2$, which allows to take more points from the inner region of the graph than the tail region improved the error encountered before by only 5% (see Fig.(10)).

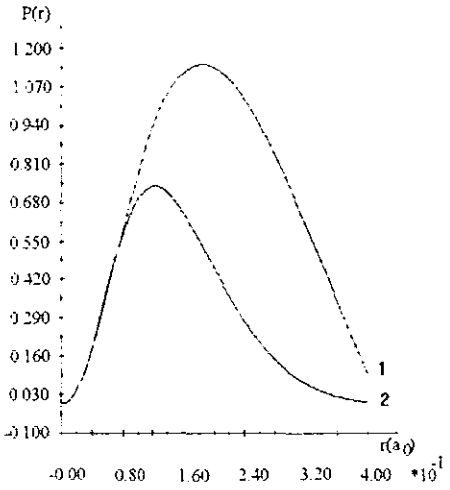


Figure 10: *Numerical (1) and Analytical (2) functions of 4d orbital with weight. The difference between the two functions is is reduced after using weight.*

5.2 Method of successive iterations with fixed extremum points (MSIFEP)

The method of analytical representation by successive iterations with fixed extremum points was applied to the numerical 4d orbital function of cadmium ion. Cd^{++}. The value of α in Eq.(24) is chosen to be 3. The numerical functions are then approximated analytically in two exponentials by the latter method. The value of the coefficients and exponents of the exponentials found are given in table (7). Therefore we can write the $4d$ numerical function analytically as

$$P_{4d} = r^3[3193.14e^{-15.26r} - 110.3e^{4.9r}] \tag{69}$$

Numerical functions reproduced from these analytic functions are in excellent agreement at the extremum points with the data. The analytic and numerical functions

AO	i	a_i	c_i	AO	i	a_i	c_i
2s	1	-38.98	47721.06	3d	1	-15.2	10187.12
	2	-273.32	91155.42		2	-57.15	28128.49
	3	-1220.58	103158.08		3	-339.58	46906.34
	4	-7820.16	9058161.50		4	-3443.50	5404517.04
2p	1	-28.71	16542.27	4s	1	-31.24	136713.26
	2	-129.41	35546.24		2	-217.84	165477.77
	3	-842.37	85797.71		3	-816.96	292524.20
	4	-8191.48	1994278.80		4	-4458.67	12891152.18
3s	1	-31.18	124257.65	4p	1	-19.08	49379.55
	2	-214.37	191479.63		2	-96.70	68995.87
	3	-889.47	299629.69		3	-346.0	120546.08
	4	-5355.49	12810685.47		4	-926.13	7378.32
3p	1	-21.41	50625.07		5	-4438.32	49270841.76
	2	-134.87	131963.57	Cd^{++}(4d)	1	-26.93	124257.11
	3	-790.73	133617.92		2	-185.15	191473.89
	4	-6393.96	22591878.8		3	-768.21	299592.47
					4	-4625.80	12801105.74

Table 6: Coefficients and exponents of exponentials for the analytic function of Cd^{++} ion orbitals approximated using Prony's method

AO	i	A_i	β_i
$Cd^{++}(4d)$	1	3193.14	15.26
	2	-110.03	4.9

Table 7: Coefficients and exponents of exponentials for the 4d analytic function of Cd^{++} ion approximated using method of successive iteration.

of Cd^{++} 4d orbital are plotted on Fig.(11). The third line appearing between the two graphs is the error or difference between the two functions. We can see from the graph that the agreement is excellent at the extremum points.

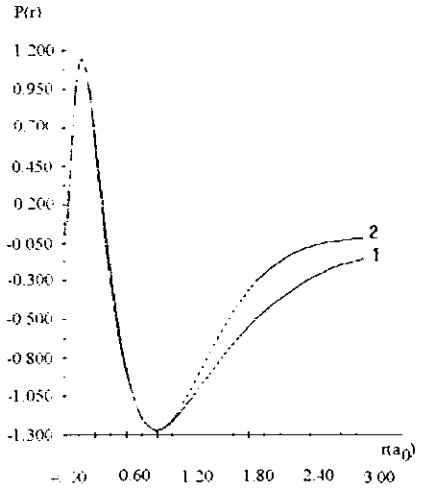


Figure 11: *Numerical (1) and analytic (2) functions of the 4d orbital of Cd^{++} approximated using the MSIFEP*

The method of successive approximation (see appendix B) was also applied to the 2s and 2p wave functions of floride ion. F^-. However the results obtained were not satisfactorily in agreement with the numerical data. For that reason the successive iteration with fixed extremum points method was improved a little. In this case the bifurcation points of the function were made fixed in addition to the maximum points. For the 2s orbital numerical SCF function the first zero point of the function was removed by dividing the function by $(r0' - r)$ where $r0'$ is the first

AO	i	A_i	β_i
Cd^{++}(4d)	1	3193.14	15.26
	2	-110.03	4.9

Table 7: Coefficients and exponents of exponentials for the 4d analytic function of Cd^{++} ion approximated using method of successive iteration.

of Cd^{++} 4d orbital are plotted on Fig.(11). The third line appearing between the two graphs is the error or difference between the two functions. We can see from the graph that the agreement is excellent at the extremum points.

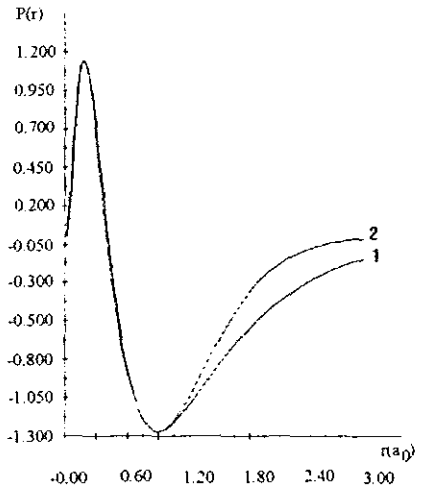


Figure 11: *Numerical (1) and analytic (2) functions of the 4d orbital of Cd^{++} approximated using the MSIFEP*

The method of successive approximation (see appendix B) was also applied to the 2s and 2p wave functions of floride ion, F^-. However the results obtained were not satisfactorily in agreement with the numerical data. For that reason the successive iteration with fixed extremum points method was improved a little. In this case the bifurcation points of the function were made fixed in addition to the maximum points. For the 2s orbital numerical SCF function the first zero point of the function was removed by dividing the function by $(r0' - r)$ where $r0'$ is the first

zero point of the numerical wave function and is equal to 0.2441 in atomic units (au).α was chosen to be 1. Then the resulting function was approximated by four exponents using the method under discussion and the values for the coefficients and exponentials obtained are tabulated on table(8). Thus the 2s numerical functions floride ion. F^- van be expressed exponentially as

$$\begin{aligned} P^*_{2s}(r) &= r(0.24-r)[1.13e^{-1.64r}+7.73e^{-5.41r} \\ &\quad 28.04e^{-4.64r}+10.87e^{-2.68r})] \end{aligned} \tag{70}$$

Comparison between the numerical functions and the analytical functions show that

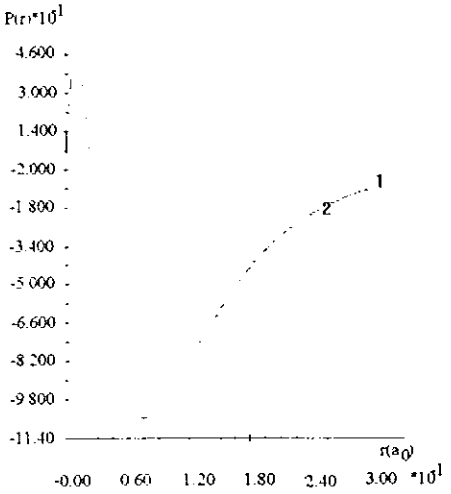


Figure 12: *Numerical (1) and analytical (2) functions for the 2s orbital of F^-. The two graphs are running together at all points showing excellent precision of the method.*

the two functions are in excellent agreement. The maximum error found is in the order of 10^{-5}.From Fig.(2), we can also see that the analytical and the numerical functions are going together in all the points.

The numerical SCF function of the 2p orbital is approximated by the SIFEP method in four exponentials. α was chosen to be 4 in this case. The values of the coefficients and exponentials obtained are tabulated in table(8). This means that the 2p orbital function of F^- can be given analytically as

$$\begin{aligned} P^*_{2p}(r) &= r^4(0.21e^{-0.92r}+10.9e^{-3.52r} \\ &\quad 2.54e^{-1.81r}+5.61e^{-6.81r}) \end{aligned} \tag{71}$$

AO	i=	A_i	β_i
2s	1	1.1302	1.6381
	2	7.7265	5.4098
	3	28.0407	4.6410
	4	10.8711	2.6844
2p	1	0.2181	0.9245
	2	10.935	3.5236
	3	2.5431	1.8133
	4	5.6121	6.8198

Table 8: Coefficients and exponents of exponentials for 2s and 2p orbital analytic functions of floride, F^- approximated using the SIFEP method

Comparisons between analytical and numerical functions approve the excellent accuracy of the 2p orbital analytic function approximated using the latest method. The maximum error is found to be in the order of 10^{-5} (See also Fig.(12)).

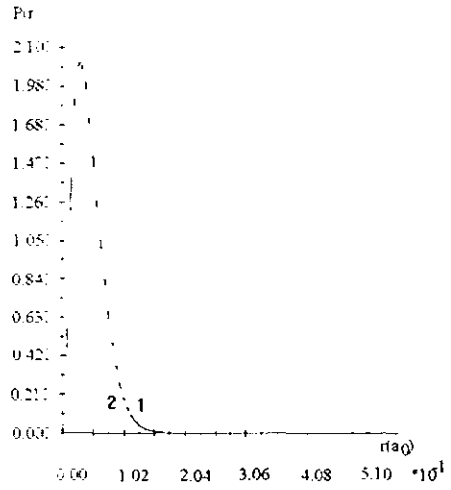


Figure 13: *Analytical (2) and numerical (1) functions for the 2p orbital of F^-. The two graphs are running together at all points showing excellent precision of the method.*

5.3 Advantages and disadvantages of Prony method and the MSIFEP

Prony's method has the following advantages:

1. In prony's method the number of exponentials used to approximate the numerical functions is not restricted. This means that one can approximate the tabulated numerical functions by any number of exponentials desired.

2. There are no fixed extremum points to be set, nor do we have fixed nodes.

3. There is not any iteration in Prony's method such that the computations using this method are not expensive in view of computation time.

Prony's method has the following disadvantages:

1. Prony's method can only be applied for functions tabulated at equispaced points. During preparation of equispaced function from the tabulated functions one might face truncation or rounding errors.

2. In the Prony's method at the construction of normal equations using the least square approximation, important functions might be cancelled. Though it is possible to give the required weights to the difference between the real function and the approximating functions at the important functions, selection of weights until the required accuracy is achieved is tiresome.

3. The approximated analytical function exhibit large errors at some points though they pass nicely at most points. Especially analytic functions of orbitals (functions) with large principal quantum number vary considerably from the numerical functions.

The method of successive iterations with extremum points has the following advantage:

1. The results of approximations of the method are very accurate at the extremum points of the functions.

Disadvantages of the MSIFEP

1. The number of exponents for the approximating functions is always fixed and must be equal to the number of extremum points.

2. The iterations are expensive with respect to computation time.

6 Conclusions

Two conclusions can be drawn from this work. (1) Prony's method can be applied to approximate numerical SCF wave functions of ions analytically in general. This method is effective for smaller principal quantum number orbitals. The application of the method for higher quantum numbers involves a considerable additional labor. (2) The method of successive iterations successfully works to analytical approximation of 4d orbital numerical SCF function of Cd^{++}. With some modification the method works to the analytical approximation of 2s and 2p orbitals of floride ion. F^-.

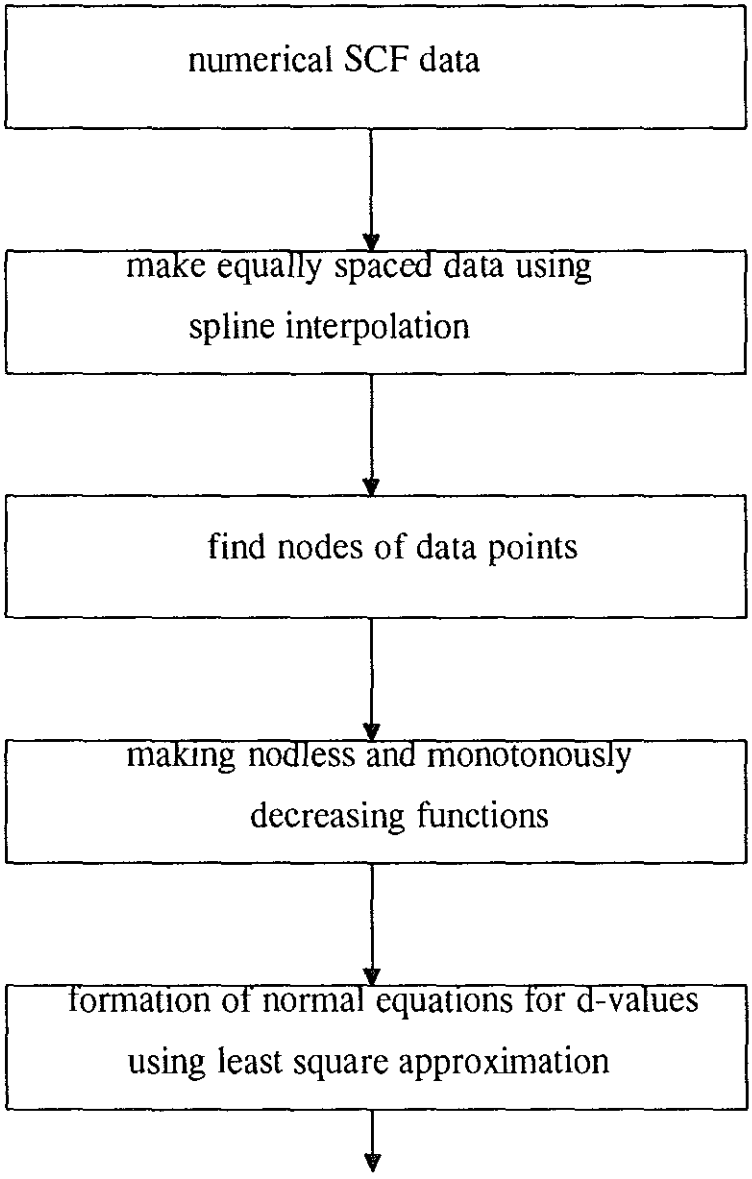
7 outlook

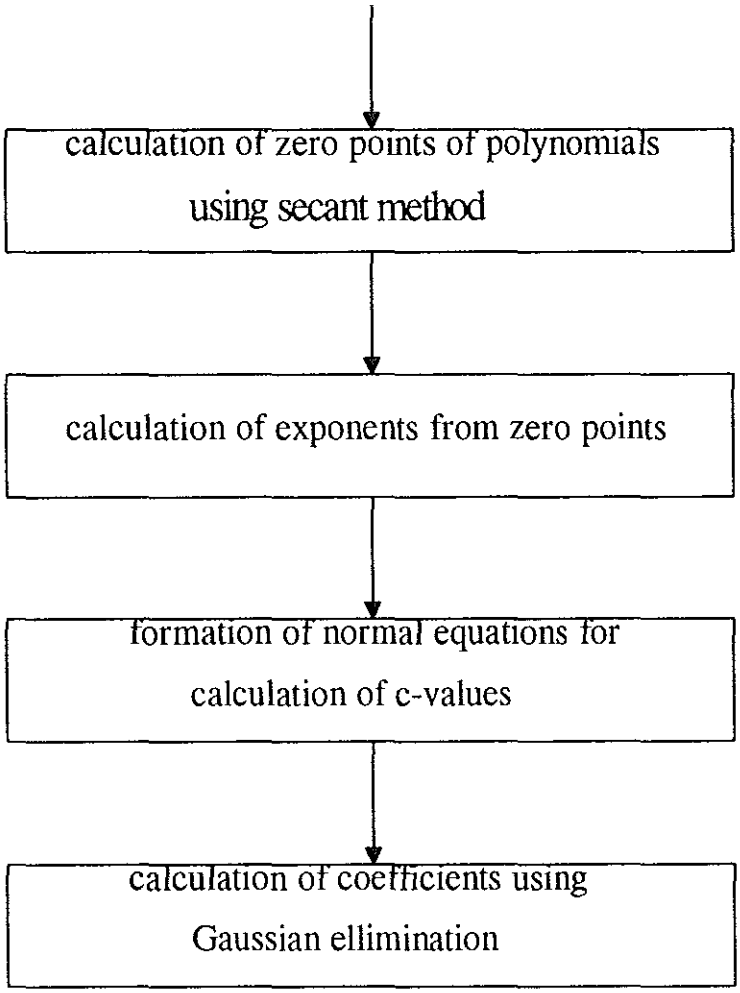
Inspection of the accuracy of analytic wave functions approximated using the methods show that excellent agreement between analytical and numerical functions are found at the extremum values of the functions for the MSIFEP and Prony method shows very good precision at points slightly far from the extremum points. Therefore it is the writer's belief that a combination of the methods might yield analytic functions with better precision.

Appendix

A Computer Programs for Prony Method

A.1 Flow Sheet of Prony Method




calculation of zero points of polynomials using secant method
calculation of exponents from zero points
formation of normal equations for calculation of c-values
calculation of coefficients using Gaussian ellimination

A.2 Prony-1

{This is a Cubic Spline technique to approximate numerical not-
equally spaced functions at eqidistant points.}

```
program Eliasspl;
uses myspline, crt;
const NumPoints=100;
var
Error:byte;
outfile,infile:text;
step:real;
i:integer;
Rad:real;
Y,X:real;
function WaveFunc(X:float):float;
var
XData:TNvector;
YData:Tnvector;
Coef0,Coef1,Coef2,Coef3:Tnvector;
Xinter:float;
Yinter:float;
Error:byte; infile:text;
Y:real;
i:integer;
begin assign(infile,'C:\tp\kotel\elias\values.dat');
reset(infile);
for i:=1 to Numpoints do
begin
readLn(infile,XData[i],YData[i]);
readLn(infile);
end;
close(infile);
{this is the procedure for Cubic Spline interpolation} {reffer Borland Turbo Pascal
6.0 for the Procedure of Cubic SplineFree} CubicSplineFree(Numpoints,XData,YData,X
,Coef0,Coef1,Coef2,Coef3, Y,Error);
if Error¡¿0 then begin writeLn('there is an error in spline. press Enter');
readLn; Halt(1) end;
WaveFunc:=Y; end; {- function WaveFunc -}


begin { Main }
assign(outfile, 'C:\tp\kotel\elias\speqvalues.dat');
rewrite(outfile);
step:=2/500;
```

```
Rad:= 1.024606767533345E-0006;
for i:=1 to 500 do
begin
X:=Rad +(i-1)*step; Y:=WaveFunc(Rad +(i-1)*step);
writeLn(outfile,X,' ',Y);
writeLn(X,' ',Y);
end;
close(outfile);
end.
```

A.3 Prony-2

```
{* This is program finds zero point of numerical wave functions*}
Program Prony-2:
var a.b.c.d.x1.x2.y1.y2 .ro.s: real:
rr: array[1..1000] of real:
psi: array[1..1000] of real:
n.i.m.j.l.o : integer :
si.zeo : text:
{The following procedure calculates accurate zero point.}
procedure findroot(var g :real: x1.x2.y1.y2 : real):
var e.f.h.l1.l2.l :real:
begin
e :=x1:
f :=x2:
l1:=y1:
l2:=y2:
g:=(e-f)*l2/(l2-l1) + f:
end:
begin {main}
writeLn(' How many data points (spline) do you have?'):
writeLn('Data shold be less than 1000'):
readLn(n):
assign(si,'C:\tp\kotel\elias\eqspval.dat'):
reset (si):
for i:=1 to n do readLn(si.rr[i],psi[i]):
close(si):
assign(zeo,'C:\tp\kotel\elias\fileofzo.dat'):
rewrite(zeo):
for J:=1 to n do begin
x1 :=rr[j]: x2:=rr[j+1]:
y1:=psi[j]: y2:=psi[j+1]:
if psi[j]*psi[j+1] ¡ 0.0 then begin
findroot(ro.x1.x2.y1.y2):
writeln('the zero point is '.ro);
writeln(zeo.ro);
end:
end:
close(zeo):
writeln('we have finished. Please RETURN'):
readln:
end. {main}
```

A.4 Prony-3

```
{* This program makes an oscillating wave function monotonously
decreasing. }
Program Prony-3;
uses crt;
{Here we put the values of the zero points of the functions}
const       r0 = 4.40245436780060E-0001;
type
      MyArray=array[1..1000] of real;
      MydoubleArray=array[1..10,1..10] of real;
var
         R: MydoubleArray : {real_matrix: coefficients}
         P,S,Psi,Psip,rr: MyArray : {real_n_vector: right-hand-side of equations}
         i,j,l,m,n,o : integer :
         y,z : real;
         si,sip:text;
begin{main}
ClrScr;
writeln(' please check if you have taken the correct constant for the ');
writeln(' zero point');
write(' Enter number of spline data available ');
readln(n);
assign(si,'C:\tp\kotel\elias\eqspval.dat');
reset (si);
for i:=1 to n do readLn(si,rr[i],psi[i]);
close(si);

            assign(sip,'C:\tp\kotel\elias\exponent.dat');
             rewrite(sip) ;
              for i:= 1 to n do
            begin
              psip[i]:= abs(psi[i]/(rr[i]*(rr[i]-r01)));
              writeln(sip,rr[i],' ',psip[i]);
            end;
              close(sip);
writeln('WE HAVE FINISHED');
readln;
end. {main}
```

A.5 Prony-4

```
{* This is a least square approximation program
      to find the normal equations for the calculation
      of d-values. The program is
      written for Eq.(48) in Prony method *}
Program Prony-4;
uses crt;
const M=4: { number of exponentials to represent the function}
           N=1000; { number of available data}
type vector=array[1..M] of real;
        matrix=array[1..M,1..M] of real:


var     i,k,j,l,cond:integer;
        A :matrix; {real matrix; coefficients}
        Y :vector; { real_n_ vector: right hand side of equation}
        rr : array[0..N] of real; { equally spaced distances from file}
        x: array[0..N] of real; {data points from file}
        input1: text:
        input2: text ;
        out1: text ;
        out2: text;
        data: text;

begin { main program}

writeln(' Choose one of the following numbers for weight'):
writeln(' 1 weight not necessary');
writeln(' 2 weight = i');
writeln(' 3 weight = sqr(i)'):
writeln(' 4 weight = i cubed');
writeln(' 5 weight = exp(-i)');
writeln(' 6 weight = 1/(1+i)');
writeln(' 7 weight = 1/X[i-1]');
write(' Write your choice and RETURN :- '):
readln(cond);
ClrScr;
writeln(' Please wait ');
        assign(data,'C:\tp\kotel\ elias\exponent.dat'):
         reset (data);
          for i:=0 to N-1 do
           begin
            readLn(data,rr[i],x[i]);
           end;
```

```
        close(data);

{* Following is a program to evaluate data according to
the algorithms of least square approximation*}

for j:=1 to M do
    for l:=1 to M do
    begin A[j,l]:=0; for i:=1 to N-M do begin
     if cond = 1
       then A[j,l]:= A[j,l]+X[M+i-1-l]*X[M+i-1-j];
     if cond = 2
       then A[j,l]:= A[j,l]+X[M+i-1-l]*X[M+i-1-j]*i;
     if cond = 3
       then A[j,l]:= A[j,l]+X[M+i-1-l]*X[M+i-1-j]*sqr(i);
     if cond =4
       then A[j,l]:= A[j,l]+X[M+i-1-l]*X[M+i-1-j]*sqr(i)*i;
     if cond = 5
       then A[j,l]:= A[j,l]+X[M+i-1-l]*X[M+i-1-j]*exp(-i);
     if cond = 6
       then A[j,l]:= A[j,l]+X[M+i-1-l]*X[M+i-1-j]*(1/(i+1));
     if cond = 7
       then A[j,l]:= A[j,l]+X[M+i-1-l]*X[M+i-1-j]*abs(1/X[i-1]);
     end;
    end;
    for j:=1 to M do
    begin Y[j]:=0; for i:=1 to N-M do begin
     if cond = 1 then
       Y[j]:=Y[j] + X[M+i-1-j]*X[M+i-1];
     if cond = 2 then
       Y[j]:=Y[j] + X[M+i-1-j]*X[M+i-1]*i;
     if cond = 3 then
       Y[j]:=Y[j] + X[M+i-1-j]*X[M+i-1]*sqr(i);
     if cond = 4 then
       Y[j]:=Y[j] + X[M+i-1-j]*X[M+i-1]*sqr(i)*i;
     if cond = 5 then
       Y[j]:=Y[j] + X[M+i-1-j]*X[M+i-1]*exp(-i);
     if cond = 6 then
       Y[j]:=Y[j] + X[M+i-1-j]*X[M+i-1]*(1/i+1);
     if cond = 7 then
       Y[j]:=Y[j] + X[M+i-1-j]*X[M+i-1]*abs(1/X[i-1]);
     end;
    end;
       assign(out1,'C:\tp\kotel\ elias\cof2.dat');
       rewrite(out1);
        for i:= 1 to M do
```

```
        writeln(out1, A[i,1],' ',A[i,2],' ',A[i,3],' ',A[i,4],' ',Y[i]);
    close(out1);
    writeln(''); { empty full line on screen}

    for j:=1 to M do
    writeLn(' new arguments ',Y[j]);
    writeln('');
    for l:=1 to M do for j:=1 to M do
    writeLn(' new coefficients ',A[l,j]);
    readLn;
end. {main program}
```

A.6 Prony-5

```
    {* This program is to calculate d-values of the prony's
    technique (see Eq.(48)) starting from the NxN matrix
    constructed using least square technique. The method
    used here is the Gausian ellimination method. *}
PROGRAM Gauss; type
   Myarray=array[1..20] of real;
   Mydoublearray=array[1..20,1..20] of real;
VAR
   n, k : INTEGER;
   U : MYARRAY; {*real-n-vector; results of calculation*}
   R : MydoubleARRAY ;{real-matrix; coefficients}
   P : MyaRRAY ; {real-n-vector; right-hand-side of equations}
   S : MyARRAY ;
   i,m,j,l,o : integer ;
   y,z : real;
   f : text;
   data : text;
   input1 : text;
   input2 : text;
   x : array[0..30] of real; { data points from file}
   rr : array[0..30] of real; { equally spaced distance from file}
     {* Following is the procedure for Gausian elimination Ref[ ] *}
       Procedure Gauss_elim
          (var G : MyARRAY ; R : Mydoublearray; P : Myarray );
       var
        A : Mydoublearray ;
        b : MyARRAY ;
        x : Myarray; real-n-vector;
        i,j,k : integer;
        tol : real;
        amult : real;
        begin { gause_elim}
        A := R ;
        b := P ;          {* Triangularize * }
       for i:=1 to n-1 do
        for j := i+1 to n do
         begin
          {* compute multiplier for j^th row *}
          amult := A[j,i]/A[i,i];
          {*compute non zero elements for j^th row*}
           for k:=i+1 to n do
             A[j,k] := A[j,k]-amult*A[i,k];
             {* compute new b[j] *}
```

```
            b[j] := b[j] - amult*b[i];
          end;
            {* Back substitution *}
            for i:=n downto 1 do
             begin
               x[i] := b[i];
            {Subtract terms in already computed x[j]}
            for j:=i+1 to n do
             x[i] := x[i] -A[i,j]*x[j];
              {solve for x[i]}
             x[i]:= x[i]/A[i,i];
             G[i] :=x[i];
             writeln('x[',i,'] =',G[i]);
            U[i] :=G[i];
             end;
        end; {gauss_elim}
begin {main program}
writeln(' please enter the number of coefficients you want ?');
Readln(n);
      {* the following reads data from file *}
      assign(input1,'c:\ elias \ cof2.dat');
      reset(input1);
        for i:= 1 to n do
          readln(input1,R[i,1],R[i,2],R[i,3],R[i,4],P[i]);
      close(input1);
          {* call procedure gauss_elim *}
           Gauss_elim( S , R , P);
           {* This part will write result on file *}
           assign(f,'c:\ elias \ fileofd.dat');
           rewrite(f) ;
            for i:= 1 to n do
            writeln(f, U[i]);
           close(f);
readln;
end. {of main }
```

A.7 Prony-6

```
{This program is used to find the range of solutions of
a given polynomial function. More over it records the approximate
regions of the solution from which we calculate the most probable
solution of the polynomial using the secant method}

Program Prony-6; uses crt, dr2d;
type Darray=array[1..30] of real;
var i,n:integer;
    infile:text;
    infileP:text;
    outfile:text;
    d :Darray;
    step,pp,p :real;
    G :Matrix;
    ArrayNumPoints:Mynumpoints;
function Polynom(p:real;d:Darray):real;
var
  i:integer;
  sum:real;
function Power(p:real;i:integer):real;
  var
   j:integer; product:real;
   begin
   if i=0 then Power:=1 else
    begin
    Product:=p;
    for j:=1 to i-1 do product:=product*p; power:=product end;
    end;
      begin
      sum:=Power(p,n);
      for i:=1 to n do sum:=sum - power(p,n-i)*d[i];
      Polynom:=sum;
end; {-function Polynom -}

begin{main}
ClrScr;
writeln('please enter the number of coefficients you want ?');
Readln(n);
       assign(infile,'C:\tp\kotel\elias\fileofd.dat');
       reset( infile);
       for i:=1 to n do readLn(infile,d[i]);
         assign(outfile,'C:\tp\kotel\elias\gues.dat');
         rewrite(outfile);
```

```
        step:=0.1;
        p:=-10;
        for i:=1 to 500 do
        begin
          G[i,0]:= p +i*step;
          G[i,1]:=Polynom(p +i*step,d);
          G[i+1,1]:=Polynom (p +(i+1)*step,d);
          if G[i,1]*G[i+1,1] ¡ 0.0 then
          begin
          Writeln('zero point here', p+i*step, p+(i+1)*step);
          Writeln(outfile,p+i*step,' ',p+(i+1)*step);
          end;
        end;
        writeln(' Please RTURN'); readln;
        close(outfile);
      close(infile);
for i:=1 to 500 do begin G[i,0]:= p +i*step;
G[i,1]:=Polynom (p +i*step,d);end;
ArrayNumPoints[1]:=500;
{Draw the graph of the polynomial on the screen}
Plot2D(1,ArrayNumPoints,1,1,G);
writeln;
assign(infileP,'C:\tp\kotel\elias\fileofp.dat');
reset( infileP);
WriteLn(' check if they are correct roots');
writeln('');
for i:=1 to n do begin readLn(infileP,PP);
writeln(pp, ' ',polynom(pp,d)); end;
close(infileP); readln;
end.{main}
```

A.8 Prony-7

```
{This is a secant program to find zero points of a polynomial
      function. We use it to find p-values,Eq(46).of the
Prony method. (See also App.(B.3)}

Program Prony-7; uses crt;
const
      NumberExp=4;
type
      Coeff=array[1..NumberExp] of real;
var
      D:Coeff;
      g1,g2,z:Coeff;
      g2:Coeff;
      a,b,c,x1 : real;
      x2,t1,t2,q,s: real;
      i,j,k,expe : integer;
      f : text;
      input: text;
      value : text;

      Here is a function to calculate pow of a number to exponent n
      function pow(base : real; exponent : integer) :real :
      var
      count : integer;
      product : real;
      begin
        product := 1;
        for count :=1 to exponent do
      product := product*base;
      pow := product;
      end;


{The foollowing procedure constructs the polynomial}
procedure ff( var y:real; P:real; D:coeff);
var i:integer;
      sum:real;
      begin
      sum:=pow(p, NumberExp);
      for i:=1 to NumberExp do
      sum:=sum - D[i]*pow(p,NumberExp-i); y:=sum;
      end; { procedure }
```

```
        {Below is the procedure to find the accurate zero point.}
    procedure secant_root (var g :real; x1,x2 : real);
    var
    newval,oldval :real;
    xnew : real;
    slope : real;
    newguess,oldguess :real;
    tol,no,denom,numi : real;
    begin
    oldguess := x1;
    newguess := x2;
    ff(newval,newguess,d);
    ff(oldval,oldguess,d);
    repeat
    ff(newval,newguess,d);
    no := newval/oldval;
    denomi := 1-no;
    numi := (oldguess -newguess)*no;
    xnew :=newguess-numi/denomi;
    oldguess := newguess;
    newguess := xnew;
    oldval := newval;
    if oldguess = 0.0
    then exit
    until(abs((newguess - oldguess)/oldguess)≤ 1.0e-7);
    g := xnew;
    z[k] :=g;
    k:=K+1;
    end;

begin main
k:=1;
    assign(value,'C:\tp\kotel\elias\fileofd.dat');
    reset(value);
    for i:=1 to NumberExp do
    begin
    readLn(value,q);
    D[i] := q;
    close(value);
    assign(input,'C:\tp\kotel\elias\gues.dat');
    reset(input);
    for i:= 1 to NumberExp do
    begin
    readln(input,g1[i],g2[i]);
    begin
```

```
        {Below is the procedure to find the accurate zero point.}
    procedure secant_root (var g :real: x1,x2 : real);
    var
    newval,oldval :real;
    xnew : real:
    slope : real:
    newguess,oldguess :real;
    tol,no,denom,numi : real;
    begin
    oldguess := x1;
    newguess := x2:
    ff(newval,newguess,d);
    ff(oldval,oldguess,d);
    repeat
    ff(newval,newguess,d);
    no := newval/oldval;
    denomi := 1-no;
    numi := (oldguess -newguess)*no:
    xnew :=newguess-numi/denomi:
    oldguess := newguess;
    newguess := xnew:
    oldval := newval;
    if oldguess = 0.0
    then exit
    until(abs((newguess - oldguess)/oldguess)≤ 1.0e-7);
    g := xnew;
    z[k] :=g;
    k:=K+1:
    end;

begin main
k:=1;
    assign(value,'C:\tp\kotel\elias\fileofd.dat'):
    reset(value):
    for i:=1 to NumberExp do
    begin
    readLn(value,q);
    D[i] := q;
    close(value);
    assign(input,'C:\tp\kotel\elias\gues.dat');
    reset(input);
    for i:= 1 to NumberExp do
    begin
    readln(input,g1[i],g2[i]);
    begin
```

```
        secant_root(s,g1[i],g2[i]);
        writeln('the zero point is ',s);
        end;
        end;
        close(input);
        assign(f,'C:\tp\kotel\elias\fileofp.dat');
        rewrite(f) ;
        for j := (k-1) downto 1 do
        writeln(f,z[j]);
        close(f);
        writeln('we have finished') ;
readln;
end. main
```

A.9 Prony-8

```
{*This program calculates value of the exponents
or a values of the Prony's method from the calculated
zero points (p-values) by just taking the logarithm
of each zero point to get the corresponding a-value. See
also Eq.(43) of the Prony's method *}

Program Prony-8; const step:= 1e-4: var
   n,i:integer;
   e : real;
   aa :text;
   f:text;
   x:array[1..10] of real;   { zero points}
   a:array[1..10] of real;    { a-values}
begin
writeln('enter the number of coefficients you want ?');
Readln(n);
ClrScr;
            assign(f,'C:\tp\kotel\elias\fileofp.dat');
            reset(f);
             for i:=1 to n do
             readLn(f,x[i]);
            close(f);
              for i:=1 to n do
              begin
               if x[i] ≤ 0.0 then
                 begin
                 writeln(' p value less than zero.');
                 exit;
                 a[i]:=ln(x[i])/step:                    end;
              end;
            { Here We write the result to the file mentioned below}
             assign(aa,'C:\tp\kotel\elias\fileofa.dat');
             rewrite(aa) ;
             for i:= 1 to n do
              writeln(aa, a[i]): { On the file }
             close(aa);
readln;
end.
```

A.10 Prony-9

```
{* This is a program of least square technique that
forms the normal equations, Eq.(45) of Prony method
It is a partial program for calculation of C-values *}


Program Prony-9;
uses crt;
const
      M=4; {* number row of matrix *}
      N=1000; {* number of data available or number of columns }

type
      vector=array[1..M] of real;
      matrix=array[1..M,1..M] of real;

var
      i,k,j,l,cond : integer;
      A      : matrix; {real_matrix: coefficients}
      Y      : vector; {real_n_vector: right hand side of equation}
      P      : array[1..10] of real: {real_n_vector p-values from file}
      x      : array[0..N] of real: {data points from file}
      im     : array[0..N] of real; {distance from file}
      input1  : text;
      input2  : text ;
      out1    : text ;
      out2    : text;
      data    : text;
      data2   : text;

      {* following is a function that calculates power *}
       function Pow(p:real;i:integer):real;
       var
         j: integer;
         product: real;
            begin
              if i=0 then Pow:=1 else
                begin
                Product:=p;
                for j:=1 to i-1 do product:=product*p;
                pow:=product;
                end;
            end;
```

```
begin{ main }
ClrScr; writeln(' Choose one of the following numbers for C-least weight'):
writeln(' 1 weight not necessary');
writeln(' 2 weight = i');
writeln(' 3 weight = sqr(i)');
writeln(' 4 weight = i cubed');
writeln(' 5 weight = exp(-i)');
writeln(' 6 weight = 1/(1+i)');
writeln(' 7 weight = abs(1/X[i-1]');
write(' Write your choice and RETURN :- ');
readln(cond);
ClrScr;
Writeln(' Please wait ');
      {* The data in this case are the p-values
      calculated using secant program. following
      is to read data from file *}
       assign(data,'C:\tp\kotel\elias\fileofp.dat');
       reset (data);
       for i:=1 to M do
        begin
        readLn(data,p[i]);
        writeln('power points ', i , p[i]); {on the screen}
        end;
       close(data);
writeln('');
readln;
clrscr;
       assign(data,'C:\tp\kotel\elias\exponent.dat');
       reset (data2);
         for i:=0 to N-1 do
         begin
          readLn(data2,im[i],x[i]);
          writeln('eq. spaced points ', i , x[i]); {on the screen}
         end;
       close(data2);
      { calculate new coefficients }
   for j:=1 to M do
   begin
    Y[j]:=0;
    for i:=0 to N-1 do begin
     if cond = 1 then
       Y[j]:=Y[j] +X[i]*pow(P[j],i);
     if cond = 2 then
       Y[j]:=Y[j] +X[i]*pow(P[j],i)*i;
     if cond = 3 then
```

```
      Y[j]:=Y[j] +X[i]*pow(P[j,i)*sqr(i):
    if cond = 4 then
      Y[j]:=Y[j] +X[i]*pow(P[j,i)*sqr(i)*i:
    if cond = 5 then
      Y[j]:=Y[j] +X[i]*pow(P[j,i)*exp(-i):
    if cond = 6 then
      Y[j]:=Y[j] +X[i]*pow(P[j,i)*(1/(1+i)):
    if cond = 7 then
      Y[j]:=Y[j] +X[i]*pow(P[j,i)*abs(1/X[i]):
    end:
     end:
        writeln(' New Y-values');
        for j:=1 to M do writeLn( Y[j]);
        writeln(' Please wait');
        Writeln(' ');
  for k:=1 to M do
   begin
    for j:=1 to M do
    begin A[k,j]:=0; for i:=0 to N-1 do begin
    if cond = 1 then
      A[k,j]:=A[k,j] +Pow(P[k]*P[j],i);
    if cond = 2 then
      A[k,j]:=A[k,j] +Pow(P[k]*P[j],i)*i;
    if cond = 3 then
      A[k,j]:=A[k,j] +Pow(P[k]*P[j],i)*sqr(i):
    if cond = 4 then
      A[k,j]:=A[k,j] +Pow(P[k]*P[j],i)*sqr(i)*i;
    if cond = 5 then
      A[k,j]:=A[k,j] +Pow(P[k]*P[j],i)*exp(-i):
    if cond = 6 then
      A[k,j]:=A[k,j] +Pow(P[k]*P[j],i)*(1/(1-i)):
    if cond = 7 then
      A[k,j]:=A[k,j] +Pow(P[k]*P[j],i)*abs(1 X[i-1]):
    end:
      end:
          { write new coefficients and new arguments on file }
       assign(data,'C:\tp\kotel\elias\cof4.dat');
          rewrite(out1);
          for i:= 1 to M do
          writeln(out1, A[i.1],' ',A[i,2],' ',A[i.3],' ',A[i,4],' ',Y[i]);
          close(out1);
writeln('');
       writeLn(' new coefficients ');
       for j:=1 to M do begin write( A [1,j]): end: writeLn;
       for j:=1 to M do begin write( A [2,j]): end: writeLn;
```

```
        for j:=1 to M do begin write( A [3,j]); end; writeLn;
        for j:=1 to M do begin write( A [4,j]); end; writeLn;
        writeLn('please, press Enter key');
readLn;
end. {main}
```

A.11 Prony-10

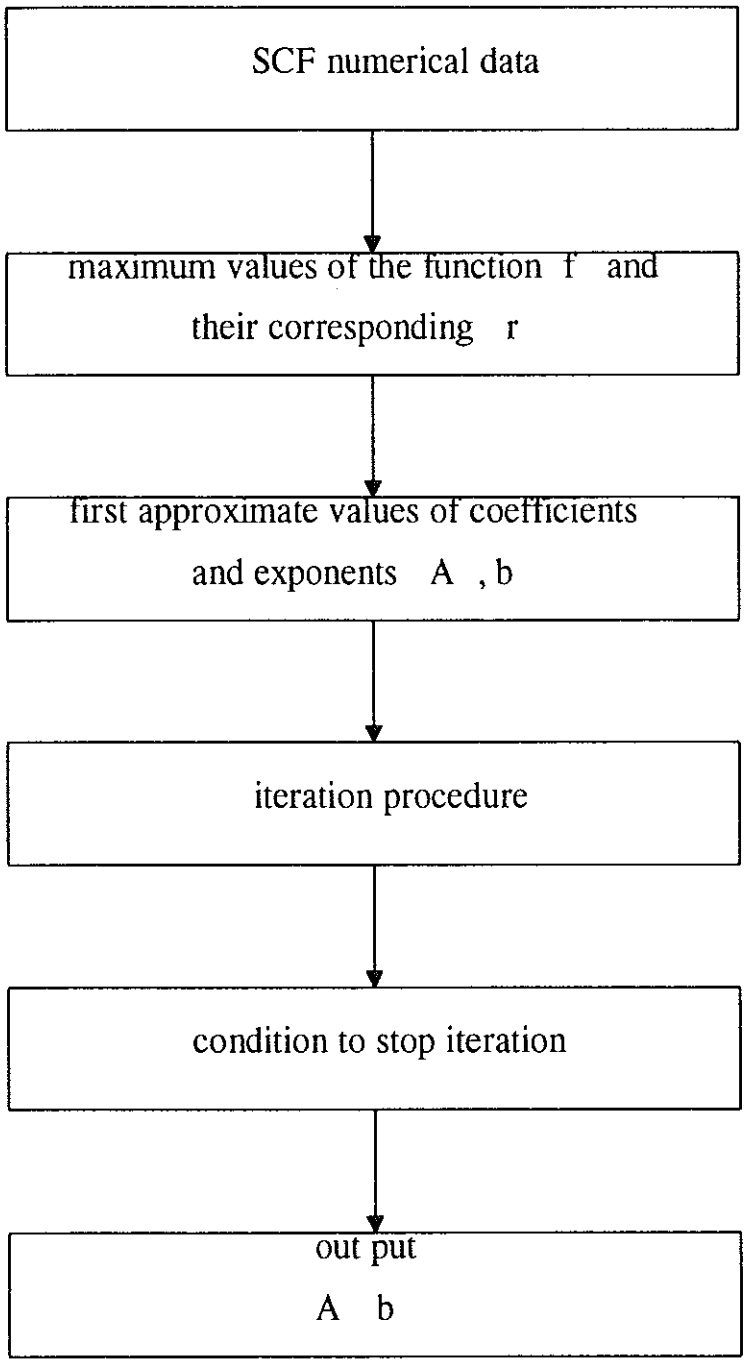
```
{* This program calculate c-values of the Prony's
method (see Eq(45)) starting from the NxN matrix
onstructed using least square approximation. The method
used here is the Gausian ellimination method. *}

Program Prony-10;
type
   MyArray=array[1..20] of real;
   MyDoublearray=array[1..20,1..20] of real;
VAR
   n, k : integer;
   U : MyArray; {*real-n-vector; results of calculation*}
   R : MydoubleArray ;{real-matrix; coefficients}
   P : MyArray ; {real-n-vector; right-hand-side of equations}
   S : MyArray ;
   i,m,j,l,o : integer ;
   y,z : real;
   f : text;
   data : text:
   input1 : text;
   input2 : text;
   x : array[0..30] of real; { data points from file}
   rr : array[0..30] of real; { equally spaced distance from file}
     {* Following is the procedure for Gausian elimination Ref[ ] *}
       Procedure Gauss_elim
          (var G : MyArray : R : MydoubleArray; P : MyArray );
       var
        A : MydoubleArray ;
        b : MyArray ;
        x : MyArray; real-n-vector;
        i,j,k : integer;
        tol : real;
        amult : real;
        begin { Gause_elim}
        A := R ;
        b := P ;          {* Triangularize * }
       for i:=1 to n-1 do
        for j := i+1 to n do
         begin
          {* compute multiplier for j^th row *}
          amult := A[j,i]/A[i,i];
          {*compute non zero elements for j^th row*}
           for k:=i+1 to n do
```

```
            A[j,k] := A[j,k]-amult*A[i,k];
            {* compute new b[j] *}
            b[j] := b[j] - amult*b[i];
          end;
            {* Back substitution *}
            for i:=n downto 1 do
             begin
               x[i] := b[i];
            {Subtract terms in already computed x[j]}
            for j:=i+1 to n do
             x[i] := x[i] -A[i,j]*x[j];
              {solve for x[i]}
             x[i]:= x[i]/A[i,i];
             G[i] :=x[i];
             writeln('x[',i,'] =',G[i]);
            U[i] :=G[i];
             end;
        end; {gauss_elim}
begin {main program}
writeln(' please enter the number of coefficients you want ?');
Readln(n);
      {* the following reads data from file *}
      assign(input1,'c:\tp kotel\ elias \ cof3.dat');
      reset(input1);
       for i:= 1 to n do
        readln(input1,R[i,1],R[i,2],R[i,3],R[i,4],P[i]);
      close(input1);
          {* call procedure gauss_elim *}
           Gauss_elim( S , R , P);
           { Here we devide the calculated c-values by the first
           c-values or c1s because our date points do not start
           from 0-values of r where as in the prony's algorithm the
           data points are put euidistant starting from a zero}
      assign(data1,'c:\tp\kotel\elias\exponent.dat');
           reset (data1);
           for i:=1 to 10 do
            begin
            readLn(data1,rr[i],xx[i]);
            end;
            close(data1);
      assign(data2,'c:\tp\kotel\elias\ fileofa.dat');
           reset (data2);
           for i:=1 to 4 do
           begin
           readLn(data2,aa[i]);
```

```
            end;
            close(data2);
        writeln('Here are the improved results') ;
        assign(f,'c:\tp\kotel\elias\ fileofc.dat');
        rewrite(f) ;
        for i:= 1 to n do
        begin
        cc[i]:= U[i]/(exp(aa[i]*rr[1]));
        writeln(f, cc[i]);
        writeln(cc[i]);
        end;
        close(f);
        readln;
end. main
```

B Computer Program for MSIFEP

B.1 Flow Sheet of MSIFEP



B.2 Programe CdF2

```
Program CdF2int;
uses crt,dos,dr2d;
const
RootD:array[1..1] of double =(4.15558092840456E-01);
NumberExp=2;
type
PointNmatrix=^Matrix;
Vector3=array[1..3] of double;
AmpArray=array[1..NumberExp] of double;
AlphaArray=array[1..NumberExp] of integer;
var
infile :file of double;
Beta,Alpha:double;
MyPoints        :double; {-Number of Points=700 -}
Step        :double; {-Ro2=Step*(NumberPoints-1) -}
Ro1,Ro2,Rot         :double; {- first value of, that connect with R
Ro=Alpha*R + Beta*Ln(R) -}
Z       :double: {-Charge of nucley-}
E       :array[1..9] of double: {-energy of shell-}
N       :Double: {-Main atomic Number-}
L       :Double;{-orbtal atomic Number-}
Q       :Double; {-Number of the electrons on the shell-}
Rmax:Double; {-Last value of radius, we assume that Wavefunction(R)=0 -}
Eps:double; {- the tolerance, if the residuels¡Eps, then is the end of iterations- }
Psi:array[1..9,1..700] of double: {-value of Wavefunction, the norm is int(0,R2 sqr(Psi)=1
-}
Rt      :double: R,Yvalue       :array[1..700] of double: {- value of radius -}
X       :double; i,j,Shift      :integer; ArrayNumPoints        :MyNumPoints; G
      :PointNmatrix; NumPoints      : double; { # of points }
NumTerms        : integer; { # of terms in least squares }
{ at XData points }
StandardDeviation       : Float; { Square root of variance; }
Variance        : Float; { Indicates goodness of fit }
Error       : byte; { Flags if something went wrong }
LastPoint,FirstPoint        :integer;
outfile       :text;
Vect31 :vector3;
Fvect31 :vector3;
Vect32 :vector3;
Fvect32 :vector3;
A :Amparray;
T :Amparray;
B :Amparray;
```

```
Al :Alphaarray;
Rext :Amparray;
Fext :Amparray;
{————————————————————}
function FY(X:double):double;
begin {-function-}
FY:=X*(RootS[1]-X)*(RootS[2]-X)*(RootS[3]-X);
end; {-function-}
{————————————————————}
procedure FindExtremum(x:Vector3;y:vector3; var Rext:double;
var Fext:double);
type Matrix33=array[1..3,1..3] of double;
var B:matrix33;
Delta2,Delta1,Delta0,Delta:double;
i:integer;

    function Determinant(B:matrix33):double;
begin {- Determinant -}
Determinant:=B[1,1]*B[2,2]*B[3,3]+B[1,2]*B[2,3]*B[3,1]+B[2,1]*B[3,2]*B[1,3]
-B[1,3]*B[2,2]*B[3,1]-B[1,2]*B[2,1]*B[3,3]-B[2,3]*B[3,2]*B[1,1];
end; begin for i:=1 to 3 do begin B[i,1]:=sqr(x[i]); B[i,2]:=x[i]; B[i,3]:=1 end;
Delta:=Determinant(B);
for i:=1 to 3 do begin B[i,1]:=y[i]; B[i,2]:=x[i]; B[i,3]:=1 end;
Delta2:=Determinant(B);
for i:=1 to 3 do begin B[i,1]:=sqr(x[i]); B[i,2]:=y[i]; B[i,3]:=1 end;
Delta1:=Determinant(B);
for i:=1 to 3 do begin B[i,1]:=sqr(x[i]); B[i,2]:=x[i]; B[i,3]:=y[i] end;
Delta0:=Determinant(B);

    Rext:=- Delta1/(2*Delta2);
Fext:=Delta2/Delta*sqr(Rext)+Delta1/Delta*Rext +Delta0/Delta;
end;
{————————————————————-}
function SigmaPrime(A:Amparray;Alpha:Alphaarray;
B:Amparray;i:integer;X:double):double;
var j :integer;
Sum:double;
Term:double;


    begin --function SigmaPrime-
Sum:=0;
for j:=1 to NumberExp do
begin
if i=j
```

```
then Term:=0
else Term:=A[j]*exp((Alpha[j]-1)*ln(X)-B[j]*X)*(Alpha[j]-B[j]*X);
Sum:=Sum+Term;
end;
SigmaPrime:=Sum;
end; {—function SigmaPrime–}
{————————————————————————}
function Sigma(A:Amparray;Alpha:Alphaarray;
B:Amparray;i:integer;X:double):double;
var j :integer;
Sum:double;
Term:double;

begin {—function Sigma–}
Sum:=0;
for j:=1 to NumberExp do
begin
if i=j
then Term:=0
else Term:=A[j]*exp(Alpha[j]*ln(X)-B[j]*X);
Sum:=Sum+Term;
end;
Sigma:=Sum;
end; {—function SigmaPrime–}
{————————————————————————}

begin {-main-}
assign (infile,'C:\TP\kotel\cdpl2.dps');
reset(infile);
read(infile,X,Rmax,NumPoints,Step,Beta,Eps,Z,Z,X,Alpha,Ro1,X,X,X,X);
for i:=1 to 9 do
begin
read(infile,E[i],N,L,Q,X);
for j:=1 to 700 do read(infile, Psi[i,j]);
read(infile,x); read(infile,x);
end;

close(infile);

{-find the roots Ro=Alpha*R - Beta*Ln(R) -}
Ro2:=Step*699;
Rt:=(Ro2-Beta*Ln(Ro2))/Alpha;
for i:=1 to 700 do
begin
j:=700-i+1;
```

```
Rot:=Rol+j*Step;

repeat
X:=(Alpha*Rt+Beta*Ln(Rt)-Rot)/(Alpha*Rt +Beta);
Rt:=Rt*(1-X);
until abs(X)¡1E-06;
R[701-i]:=Rt;
end; {-find the roots Ro=Alpha*R - Beta*Ln(R) -}

Al[1]:=3; Al[2]:=4;
i:=0;
repeat
i:=i+1;
until Psi[9,i+1]¡Psi[9,i];
writeLn('i=',i);
writeLn('R[i]=',R[i]);
readLn; {-i=154-}
Vect31[1]:=R[153]; Vect31[2]:=R[154]; Vect31[3]:=R[155];
Fvect31[1]:=Psi[9,152]; Fvect31[2]:=Psi[9,154]; Fvect31[3]:=Psi[9,156];
FindExtremum(Vect31,Fvect31,Rext[1], Fext[1]);
writeLn(Rext[1],' ',Fext[1],' ',Psi[9,154]);
readLn;
B[1]:=Al[1]/(Rext[1]);
A[1]:=Fext[1]*exp(Al[1])*exp(-Al[1]*ln(Rext[1]));

i:=155;
repeat
i:=i+1;
until Psi[9,i+1]¿Psi[9,i];
writeLn('i=',i);
writeLn('R[i]=',R[i]);
readLn; {—i=179—} Vect32[1]:=R[178]; Vect32[2]:=R[179]; Vect32[3]:=R[180];
Fvect32[1]:=Psi[9,178]; Fvect32[2]:=Psi[9,179]; Fvect32[3]:=Psi[9,180];
FindExtremum(Vect32,Fvect32,Rext[2], Fext[2]);
writeLn(Rext[2],' ',Fext[2],' ',Psi[9,179]);
readLn;

B[2]:=Al[2]/(Rext[2]);
A[2]:=Fext[2]*exp(Al[2])*exp(-Al[2]*ln(Rext[2]));

i:=700;
repeat
i:=i-1;
until Psi[9,i]¡¿0;
writeLn('i=',i); {-i=580 for D-function -}
```

C Relation Among Atomic Units

	Rydberg	Hartree
Length r	$r_R = \frac{r}{a_0}$	$r_H = \frac{r}{a_0}$
Energy E	$E_R = \frac{E}{R_\infty}$	$E_H = \frac{E}{R_\infty}$
∇^2	$\nabla_R^2 = a_0^2 \nabla^2$	$\nabla_H^2 = a_0^2 \nabla^2$
$\left(\frac{-\hbar^2}{2m}\nabla^2 - \frac{Ze^2}{r}\right)\Psi = E\Psi$	$\left(-\nabla_R^2 - \frac{2Z}{r_R}\right)\Psi = E_R\Psi$	$\left(-\frac{1}{2}\nabla_H^2 - \frac{Z}{r_H}\right)\Psi = E_H\Psi$

a_0 = Bohr radius = $\frac{h^2}{me^2} = 0.5292\text{A}^0$

R_∞ = Rydberg Constant = $\frac{1}{2}mc^2(e^2/hc)^2 = 13.606\text{ev}$

References

[ASC] N. W. Aschcroft and N.D. Mermis, Solid State Physics, Halt. Reinhardand Winston. Inc., USA(1988).

[ATO] Atomic Data and Nuclear Data Tables,**14**, 177(1974).

[BEL] R. Bellman and R.Roth, Quasiliniarization and the Identification Problem. World Scientific Publishing Company. Singapore(1983).

[BRE] R. G. Breen.Jr.. Phys.Rev.,**111**,1111(1958).

[FIN] W. Finkelburg, Atomic Physics, McGrawhill Book Company. Inc.. USA(1950).

[FOC] V. A. Fock, Fundamentals of Quantum Mechanics, 3^{rd} edition. Mir Publishers. Moscow(1956).

[GRE1] Green. Louis. Mulder. Wyeth, and Woll. Phys.Rev..**93**.273(1958).

[GRE2] Donald Greenspan and Vincenzo Casuli. Numerical Analysis for Applied Mathematics. Science, and Engineering. Edison-Wesley Publishing Company. Inc..UK(1988).

[HAR1] W. A. Harison. Solid State theory. Dover Publishers.Inc.. New York(1985).

[HAR2] H. Hartlet. Phys. Rev.. **88**, 552(1952).

[HAR3] J. F. Hart and Herzberg, Phys. Rev.,**106**,684(1957).

[HYL] E. Hylleraas and J. Midtal, Phys. Rev., **103**,829(1956).

[LAN] L. D. Landau and E. M. Lifshitz, Quantum Mechanics, Non-Relativistic Theory, Vol. 3, Addison Wesley Publishing Company,Inc., Massachussets, USA(1958).

[LIN] I. Lindergen, and J. Morison, Atomic Many Body Theory, 2^{nd} edition, Springer Series, Germany(1956).

[LOW1] P. O. Lowdin, Phys. Rev., **90**,120(1953).

[LOW2] P. O. Lowdin, Phys. Rev.,**94**,1600(1954).

[NES] R. K. Nesbet and R. E. Watson, Phys. Ryv., **110**, 1073(1958).

[PET] G. T. Petrovski, N. V. Starastrin, P.A. Rodnyi, and L.K. Ermakov, Dokl. Akad. Nauk SSSR 322,289-292, 1992.

[RIV] T. J. Rivlin, An Introduction to the Approximation Functions, Dover Publication, Inc., New York,1969.

[PUZ] S. M. Puzer and V. L. Wallace, To Compute Numerically, (Concepts and Strategies), Little Brownand and Company, Canada(1983).

[SLA1] J. C. Slater, Phys. Rev.,**81**,385(1951).

[SLA2] J. C. Slater, Phys. Rev.,**82**,538(1951).

[SLA3] J. C. Slater, Phys. Rev.,**91**,528(1953).

[SLA4] J. C. Slater, Quantum Theory of Atomic Structure, Vol.I, Mc.Grawhill Book Company, Inc., New York(1960).

[SLA5] J.C. Slater. Quantum Theory of Atomic Structure, Vol.II. Mc.Grawhill Book Company, Inc., New York(1960).

[STE] F.Stern. Phys. Rev., **104**.684(1956).

[VAN] S. Vandergraft, Introduction to Numerical Computations, Academic Press. New York(1978):

[WEI] M. Weissbluth. Atoms and Molecules. Academic Press, Inc., New York(1978).

[WIL] A.O. Wiliams Phys. Rev..**90**.803(1953).