Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

A Computational System for the Calculation of the Pressure Dependence of Ionic Polarizabilities in Crystals

Maria Carolina Monard/ILTC
Alicia Batana
Jorge Bruno

Nº 113

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Maio/2000

A Computational System for the Calculation of the Pressure Dependence of Ionic Polarizabilities in Crystals

Maria Carolina Monard*
Universidade de São Paulo/ILTC
Instituto de Ciências Matemáticas de São Carlos
Departamento de Ciências de Computação e Estatística
Caixa Postal 668, 13560-970 - São Carlos, SP, Brasil
e-mail: mcmonard@icmsc.sc.usp.br

Alicia Batana†
Jorge Bruno†
Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Química Inorgánica Analítica y Química Física
Ciudad Universitaria - Pabellón 2
Buenos Aires - Argentina
e-mail: {batana,jbruno}@q1.fcen.uba.ar

Abstract

A computational system is developed that allows for the calculation of polarizabilities of both anions and cations on the basis of a new method and of the extension of the one developed previously. Two alternative ways of evaluation of $\frac{\partial r_i}{\partial p}$ are given. Values of $\frac{\partial \alpha_{total}}{\partial p}$ can be evaluated for a given salt even when experimental data at high pressures are not available.

*Work partially supported by National Research Councils CAPES and FINEP, Brasil
†Work partially supported by CONICET and ANPCyT, Argentina

Contents

List of Tables

1 Introduction

In a previous work [4, 6] a method for the evaluation of ionic polarizabilities and their pressure dependence is discussed and implemented. It is an empirical method that allows for its extension to the study of more complex materials subject to the independent evaluation of the ionic radii with its limitation [1].

On the other hand, as was discussed previously [8], the ab initio calculations give a good microscopical physical insight but present limitations when extended to more complex systems and also due to the assumption α_+ constant.

We here present an alternative way of evaluation of the pressure dependence of ionic polarizabilities both for anions and cations. A computational system, implemented in Maple V, has been developed so as to allow for calculations with the different methods and a comparison of results.

This work is organized as follows: in Section 2 the methods for evaluating the pressure dependence of ionic polarizabilities both for anions and cations are presented. In Section 3 the procedures implemented in the PDIPC Maple module are described in detail and we conclude in Section 5.

2 Methods

In this section we are giving two methods for the evaluation of polarizabilities of anions and cations.

1. The empirical relation between ionic polarizabilities and ionic radii

$$\frac{\alpha_+}{\alpha_-} = \frac{r_+^3}{r_-^3}$$

is assumed to be valid at different pressures; this, together with

$$\alpha_{total} = \alpha_+ + \alpha_-$$

allows for the differentiation with respect to pressure, giving:

$$\frac{\partial \ln \alpha_+}{\partial p} - \frac{\partial \ln \alpha_-}{\partial p} = 3\left[\frac{\partial \ln r_+}{\partial p} - \frac{\partial \ln r_-}{\partial p}\right] \tag{1}$$

and

$$\frac{\partial \alpha_{total}}{\partial p} = \frac{\partial \alpha_+}{\partial p} + \frac{\partial \alpha_-}{\partial p} \tag{2}$$

where α_+ and α_- are cation and anion polarizabilities and r_+ and r_- the ionic radii respectively.

With the values of $\frac{\partial r_+}{\partial p}$, $\frac{\partial r_-}{\partial p}$ previously evaluated and $\frac{\partial \alpha_{total}}{\partial p}$ (from experimental data using Clausius Mossotti) one can evaluate $\frac{\partial \alpha_+}{\partial p}$ and $\frac{\partial \alpha_-}{\partial p}$

$\frac{\partial \alpha_{total}}{\partial p}$ is calculated with the experimental values of $-V\frac{\partial \epsilon_\infty}{\partial V}$ when available; otherwhise one may introduce in equation (1) $\frac{\partial \alpha_-}{\partial p}$ (or $\frac{\partial \alpha_+}{\partial p}$) calculated independently in order to obtain an evaluation of $\frac{\partial \alpha_+}{\partial p}$ (or $\frac{\partial \alpha_-}{\partial p}$)

2. Our previous method [4, 6] can be extended for the evaluation of $\frac{\partial \alpha_+}{\partial p}$ using similar assumptions as those introduced for $\frac{\partial \alpha_-}{\partial p}$, i.e. the existence of a function $\alpha_+(r_+, r_-)$ that can be differentiated:

$$\frac{\partial \alpha_+}{\partial p} = \left.\frac{\partial \alpha_+}{\partial r_+}\right)_- \frac{\partial r_+}{\partial p} + \left.\frac{\partial \alpha_+}{\partial r_-}\right)_+ \frac{\partial r_-}{\partial p} \tag{3}$$

where

$$\left.\frac{\partial \alpha_+}{\partial r_+}\right)_- \approx \left.\frac{\partial \alpha_+}{\partial R}\right)_- \tag{4}$$

and

$$\left.\frac{\partial \alpha_+}{\partial r_-}\right)_+ \approx \left.\frac{\partial \alpha_+}{\partial R}\right)_+ \tag{5}$$

Now $\left.\frac{\partial \alpha_+}{\partial R}\right)_+$ should be considered zero because in the calculation of $\frac{\partial \alpha_-}{\partial p}$ we are assuming that for a given cation α_+ is constant for different salts, and therefore its value is “transferable” from one salt to another

There is also an alternative evaluation of $\frac{\partial r_i}{\partial p}$. In a previous work [4, 5, 6] we have described the method of evaluation of $\frac{\partial r_i}{\partial p}$ where we propose its evaluation starting from experimental data of the pressure dependence of the compressibility and assuming

$$\chi = -\frac{1}{V}\frac{\partial V}{\partial p} = -\frac{3}{R}\frac{\partial R}{\partial p} \approx -\frac{3}{R}\left(\frac{\partial r_+}{\partial p} + \frac{\partial r_-}{\partial p}\right)$$

By a maximization procedure similar to the one used previously [4, 6] one may evaluate $\frac{\partial r_+}{\partial p}$ and $\frac{\partial r_-}{\partial p}$ from the experimental data of χ and R, for different salts [7]. This method allows for a corroboration of the values obtained previously.

3 Module PDIPC Description

In this section the PDIPC Module implemented in Maple V [2], responsible for the calculation described in Section 2 is described in detail.

There are four main procedures — see Section 3.1 — each of them implementing one of the equations described. Other procedures, which use the main procedures, have been implemented such that calculations for all salts known by the system can be done at once.

Furthermore, the system offers a debug option to verify for errors or missing values in the input data.

After defining the input for the PDIPC module, the user can simply call interactively the procedures already implemented or he/she can combine them into new procedures.

3.1 Main Procedures

This section describes, at a very high level, the equation implemented by each of the four main procedures. It should be observed that those equation are refered in the comments of the Maple V implementation as equation 0, 1, 2 and 3 respectively — see Table 1.

Procedure	Equation Implemented	Refered as
`vapopres(Salt)`	6	Equation 0
`der_alphas(Salt)`	7	Equation 1
`der_alphaminus(Salt,DerAlphaPlus)`	8	Equation 2
`der_alphaplus(Salt,DerAlphaMinus)`	9	Equation 3

Table 1: Correspondence between Main Procedure, Equation Number and Number used in Comments

3.1.1 Procedure vapopres(Salt)

This procedure returns the value of $\frac{\partial\alpha_-}{\partial p}$, for a given `Salt`, defined as

$$\frac{\partial\alpha_-}{\partial p} = \left.\frac{\partial\alpha_-}{\partial R}\right)_+ \frac{\partial r_-}{\partial p} + \left.\frac{\partial\alpha_-}{\partial R}\right)_- \frac{\partial r_+}{\partial p} \tag{6}$$

3.1.2 Procedure der_alphas(Salt)

This procedure returns the value of $\frac{\partial\alpha_+}{\partial p}$ and $\frac{\partial\alpha_-}{\partial p}$, for a given `Salt`, which are the solutions of the following system of equations

$$\left\{\begin{array}{ccccc} \frac{1}{\alpha_+}\frac{\partial\alpha_+}{\partial p} & - & \frac{1}{\alpha_-}\frac{\partial\alpha_-}{\partial p} & = & 3\left[\frac{1}{r_+}\frac{\partial r_+}{\partial p} - \frac{1}{r_-}\frac{\partial r_-}{\partial p}\right] \\ \frac{\partial\alpha_+}{\partial p} & + & \frac{\partial\alpha_-}{\partial p} & = & \frac{\partial\alpha_{total}}{\partial p} \end{array}\right. \tag{7}$$

3.1.3 Procedure der_alphaminus(Salt,DerAlphaPlus)

This procedure returns the value of $\frac{\partial\alpha_-}{\partial p}$, for `DerAlphaPlus`= $\frac{\partial\alpha_+}{\partial p}$ for a given `Salt`, defined as

$$\frac{\partial\alpha_-}{\partial p} = \alpha_-\left\{\frac{1}{\alpha_+}\frac{\partial\alpha_+}{\partial p} - 3\left[\frac{1}{r_+}\frac{\partial r_+}{\partial p} - \frac{1}{r_-}\frac{\partial r_-}{\partial p}\right]\right\} \tag{8}$$

3.1.4 Procedure der_alphaplus(Salt,DerAlphaMinus)

This procedure returns the value of $\frac{\partial\alpha_+}{\partial p}$, for `DerAlphaMinus`= $\frac{\partial\alpha_-}{\partial p}$ for a given `Salt`, defined as

$$\frac{\partial\alpha_+}{\partial p} = \alpha_+ \left\{ 3 \left[\frac{1}{r_+}\frac{\partial r_+}{\partial p} - \frac{1}{r_-}\frac{\partial r_-}{\partial p} \right] + \frac{1}{\alpha_-}\frac{\partial\alpha_-}{\partial p} \right\} \tag{9}$$

3.2 Data Structure

We use the Maple table data structure to store the data in Tables 5 and 6 pg. 24. Unlike arrays, where indices must be integers, the indices (or keys) of a table can be any value. This allow us to use the name of the salts, anions and cations, as keys of the different table entries. With this rich data structure, the user can easily (and safetly) add new salts, anions and cations to any table known by the module.

These tables can either be stored in Maple file(s) which should be read in by the module, or can be directly defined in the module worksheet. Table 2 shows the name of each table, as known by the PDIPC module, the corresponding contents as well as the equation (or procedure, see Table 1) that access the information inside that table.

Table Name	Contents	Used by Equation/Procedure
`Dlt_fxca`	$\left.\frac{\partial\alpha_-}{\partial R}\right)_+$	0
`Dlt_fxan`	$\left.\frac{\partial\alpha_-}{\partial R}\right)_-$	0
`Dlt_ranca`	$\frac{\partial r_+}{\partial p}$ and $\frac{\partial r_-}{\partial p}$	0 1 2 3
`Dlt_alphatot`	$\frac{\partial\alpha_{total}}{\partial p}$	1
`Tlt_alpha_plus`	α_+	1 2 3
`Tlt_alpha_minus`	α_-	1 2 3
`Rlt_anca`	r_+ and $r-$	1 2 3

Table 2: Tables Holding Data Values for Salts, Anions and Cations

4 Worksheet Description

Initially the module is restarted and optionally the user can define the number of digits carried in floats (default is 10). At any moment the user can set as `true` or `false` the global variable `Debug` used by the PDIPC module to inform the user about possible erros detected.

```
> restart: Digits:=6: Debug:=true
```

Next the tables described in Table 2 — see Section 6 — should be read in, if stored in a Maple file, or directly defined in the worksheet. For simplicity, the latest option is used here.

4.1 Data Tables Definition

The following Maple regions define each one of the data tables in Table 2.

```
> Dlt_fxca:= table([(Li,F)=1.54448, (Li,Cl)=5.88438,(Li,I)=9.17727,
> (Li,Br) =7.27507,(Na,F)= 2.19991,(Na,Cl)=6.16093,(Na,I) = 9.40833,
> (Na,Br)= 7.48394, (K,F)= 2.53621,(K,Cl)= 6.74405,(K,I)=10.09774,
> (K,Br)= 8.05675,(Rb,F)= 2.58094,(Rb,Cl)= 7.02686,(Rb,I)=10.51739,
> (Rb,Br)= 8.45523,(Cs,F)=1.61768, (Cs,Cl)=6.57648,(Cs,I)=10.17983,
> (Cs,Br)=7.96919]):
```

```
> Dlt_fxan := table( [(Li,F)=.60838, (Li,Cl)=.86975, (Li,I)=1.08272,
> (Li,Br) =.89898, (Na,F)= .40023, (Na,Cl)= .65620, (Na,I) =  .85392,
> (Na,Br)=.69642, (K,F)= .26056,(K,Cl)= .47053,(K,I)= .65512,
> (K,Br)= .51719, (Rb,F)=.22300, (Rb,Cl)=.41127, (Rb,I) = .58267,
>  (Rb,Br) = .45273, (Cs,F)=.18232, (Cs,Cl)=.32780,(Cs,I)=0.46841,
> (Cs,Br)=.36480]):
```

```
> Dlt_ranca:= table([(Li)=- 0.2780, (K)=-2.0636,(Rb)=-2.8707,(Na)=-0.8466,
> (F)=-0.6709,(Cl)=-1.9148,(Br)=-2.6866,(I)=-4.0229,(Cs)=-3.5537]):
```

```
> Dlt_alphatot := table([(Li,F)=-0.9, (Li,Cl)=-7.5, (Li,I)=-29.7,(Li,Br)=-14.2,
> (Na,F)= -1.5, (Na,Cl)= -7.3, (Na,I)= -26.2,(Na,Br)=-12.5, (K,F)= -3.3,
> (K,Cl)= -9.7,(K,I)= -25.4,(K,Br)= -14.9, (Rb,F)= -5.1, (Rb,Cl)=-11.7,
>  (Rb,I)= -31.8, (Rb,Br)= -17.5,(Cs,Cl)= -19.63,(Cs,I)= -46.48, (Cs,Br)=-27.78]):
```

```
> Tlt_alpha_plus:= table([(Li,F)=0.030, (Li,Cl)=0.030,(Li,I)=0.030,
> (Li,Br) =0.030,(Na,F)=0. 147,(Na,Cl)=0. 147 ,(Na,I) = 0. 147,
> (Na,Br)= 0. 147, (K,F)= 0.81,(K,Cl)=0.81,(K,I)=0.81,
>  (K,Br)= 0.81,(Rb,F) = 1.35,(Rb,Cl)= 1.35,(Rb,I) = 1.35, (Rb,Br)= 1.35,
> (Cs,F)=2.34, (Cs,Cl)=2.34,(Cs,I)=2.34,(Cs,Br)=2.34]):
```

```
> Tlt_alpha_minus:= table([(Li,F)=0.89, (Li,Cl)=2.94,(Li,I)=6.19,
> (Li,Br) =4.13,(Na,F)=1.02,
> (Na,Cl)=3.14 ,(Na,I) = 6.35,(Na,Br)= 4.28, (K,F)= 1.20,(K,Cl)=3.36,
> (K,I)=6.66, (K,Br)= 4.54,(Rb,F) = 1.18,(Rb,Cl)= 3.45,(Rb,I) = 6.80,
>  (Rb,Br)= 4.58,(Cs,F)=1.27, (Cs,Cl)=3.55,(Cs,I)=6.81,(Cs,Br)=4.66]):
```

```
> Rlt_anca:= table([(Li)=0.60, (K)=1.33,(Rb)=1.48,(Na)=0.95,(F)=1.36,(Cl)=1.81,
> (Br)=1.95,(I)=2.16,(Cs)=1.69]):
```

4.2 Auxiliar Procedures

In this section several auxiliar procedures implemented are described in detail. These procedures have access to different salt, anion and cation data tables. Table 3 shows the data tables accessed by each one of them. It should be observed that the auxiliar procedures `all_saltsInTables` and `all_commonsaltsInTables` have access to a variable number of data tables (minimum 2).

Data Table	Auxiliar Procedures		
	`is_in3tables`	`is_in4tables`	`all_saltsInTables` `all_commonsaltsInTables`
`Dlt_fxca`	×		⊕
`Dlt_fxan`	×		⊕
`Dlt_ranca`	×	×	
`Dlt_alphatot`			⊕
`Tlt_alpha_plus`		×	⊕
`Tlt_alpha_minus`		×	⊕
`Rlt_anca`		×	

Table 3: Data Tables accessed by Auxiliar Procedures: × all of them; ⊕ combination of at least two of these Data Tables

4.2.1 Procedure is_in3tables(Salt)

This procedure verifies that the elements in the two elements list `x` representing a salt name given as a list `[Anion,Cation]` is a defined entry in tables `Dlt_fxan`, and `Dlt_fxca`. It also verifies that the first element `Anion` and second element `Cation` of the list `x` are defined in table `Dlt_ranca`.

If the global variable `Debug` is `true` the procedure also prints whenever the salt, cation and/or anion for this salt are not defined in the common three tables `Dlt_fxan`, `Dlt_fxca` and `Dlt_ranca`

```
> is_in3tables:=proc(x:list)
> local fxan,fxca, ranca1,ranca2;
> global Dlt_fxan,Dlt_fxca,Dlt_ranca, Debug;
> fxan:=member(x,[indices(Dlt_fxan)]);
> fxca:=member(x,[indices(Dlt_fxca)]);
> ranca1:=member([op(1,x)],[indices(Dlt_ranca)]);
> ranca2:=member([op(2,x)],[indices(Dlt_ranca)]);
>  if (fxan and fxca and ranca1 and ranca2) then true else
```

```
>     if Debug then
>         if not fxan then print(`Salt`,x,` not defined in Table Dlt_fxan`) fi;
>         if not fxca then print(`Salt`,x,` not defined in Table Dlt_fxca`) fi;
>         if not ranca1 then print(`Anion`, op(1,x),
>                            ` not defined in Table Dlt_ranca`) fi;
>         if not ranca2 then print(`Cation`,op(2,x),
>                            ` not defined in Table Dlt_ranca`) fi;
>     fi;
> false fi;
> end:
```

The Maple function `indices(Table)` returns sequences of the keys entries of the table `Table` corresponding to the entries that are explicitly stored, in an arbitrary order. However, there is a one-to-one correspondence between the result of indices and entries. For example, the indices of data table `Dlt_fxan` are returned by the following call

```
> indices(Dlt_fxan);
```

$$[Cs, Br], [Li, F], [Li, Cl], [Li, Br], [Na, I], [Li, I], [K, Br], [Na, Cl], [Rb, F], [Rb, Cl], [Rb, Br], [Na, F], [Cs, F], [Cs, Cl], [K, I], [Rb, I], [Cs, I], [Na, Br], [K, F], [K, Cl]$$

The Maple function `member(Elem,List)` determines if `Elem` is a member of the set or list `List`. It returns `true` if so, and `false` otherwise. For example, the following call returns `true` since the element `[Li,F]` is a member of the list of indices of data table `Dlt_fxan`

```
> member([Li,F],[indices(Dlt_fxan)]);
```

true

while the following call returns `false` since the element `[X,F]` is not a member of the list of indices of data table `Dlt_fxan`

```
> member([X,F],[indices(Dlt_fxan)]);
```

false

The `op(i,expr)` Maple function extracts the components of an expression `expr`. If the first argument is a nonnegative integer `i` then the result is the ith operand of `expr`. For example

```
> op(1,[Li,F]);
```

$$Li$$

```
> op(2,[Li,F]);
```

$$F$$

This means that the following calls return `true` since the elements `Li` and `F` are both members of the list of indices of data table `Dlt_ranca`

```
> member([op(1,[Li,F])],[indices(Dlt_ranca)]);
```

true

```
> member([op(2,[Li,F])],[indices(Dlt_ranca)]);
```

true

while the following call returns `false` since the element `X` is not a member of the list of indices of data table `Dlt_ranca`

```
> member([op(1,[X,F])],[indices(Dlt_ranca)]);
```

false

Finally, executing the procedure for salt `[Li,F]` returns `true` since that salt is defined in data tables `Dlt_fxan` and `Dlt_fxca`, and cation `Li` and anion `F` are both defined in data table `Dlt_ranca`

```
> is_in3tables([Li,F]);
```

true

while executing the procedure for salt `[X,F]` returns `false` since that salt is not defined in data tables `Dlt_fxan` and `Dlt_fxca`, and cation `X` is not defined in data table `Dlt_ranca`. As the global variable `Debug` is set to `true`, these facts are informed to the user

```
> is_in3tables([X,F]);
```

Salt, $[X, F]$, *not defined in Table Dlt_fxan*

Salt, $[X, F]$, *not defined in Table Dlt_fxca*

Anion, X, *not defined in Table Dlt_ranca*

false

4.2.2 Procedure is_in4tables(Salt)

This procedure verifies if the elements in the list of two elements `x:= [Anion,Cation]` representing a salt's name is a defined entry in tables `Tlt_alpha_plus` and `Tlt_alpha_minus`. It also verifies that the first element `Anion` and second element `Cation` of `x` are defined in tables `Dlt_ranca` and `Rlt_anca`.

If global variable `Debug` is true then the procedure also prints whenever the salt, cation and/or anion for this salt are not defined in the common four tables `Tlt_alpha_plus`, `Tlt_alpha_minus`, `Dlt_ranca` and `Rlt_anca`. This procedure is used by the procedures that implement equations 1, 2 and 3.

```
> is_in4tables:=proc(x:list)
> local alpha_plus,alpha_minus, ranca1,ranca2,anca1,anca2;
> global Tlt_alpha_plus, Tlt_alpha_minus, Dlt_ranca, Rlt_anca, Debug;
> alpha_plus:=member(x,[indices(Tlt_alpha_plus)]);
> alpha_minus:=member(x,[indices(Tlt_alpha_minus)]);
> ranca1:=member([op(1,x)],[indices(Dlt_ranca)]);
> ranca2:=member([op(2,x)],[indices(Dlt_ranca)]);
> anca1:=member([op(1,x)],[indices(Rlt_anca)]);
> anca2:=member([op(2,x)],[indices(Rlt_anca)]);
> if (alpha_plus and alpha_minus and ranca1 and ranca2 and anca1 and anca2)
>  then true else
>     if Debug then
>         if not  alpha_plus then print(`Salt`,x,` not defined in Table
> Tlt_alpha_plus`) fi;
>         if not  alpha_minus then print(`Salt`,x,` not defined in Table
> Tlt_alpha_minus`) fi;
>         if not ranca1 then print(`Anion`, op(1,x),` not defined in Table
> Dlt_ranca`) fi;
>         if not ranca2 then print(`Cation`,op(2,x),` not defined in Table
> Dlt_ranca`) fi;
>         if not anca1 then print(`Anion`,op(1,x),` not defined in Table
> Rlt_anca`) fi;
>         if not anca2 then print(`Cation`,op(2,x),` not defined in Table
> Rlt_anca`) fi;
>     fi;
> false fi;
> end:
```

For example, executing this procedure for salt `[Li,F]` returns `true` since that salt is defined in data tables `Tlt_alpha_plus` and `Tlt_alpha_minus`, and cation `Li` and anion `F` are both defined in data tables `Dlt_ranca` and `Rlt_anca`

```
> is_in4tables([Li,F]);
```

true

while executing the procedure for salt `[X,F]` returns `false` since that salt is not defined in data tables `Tlt_alpha_plus` and `Tlt_alpha_minus`, and cation `X` is not defined in data tables `Dlt_ranca` and `Rlt_anca` As the global variable `Debug` is set to `true`, these facts are informed to the user

```
> is_in4tables([X,F]);
```

$$Salt, [X, F], \textit{ not defined in Table Tlt\_alpha\_plus}$$

$$Salt, [X, F], \textit{ not defined in Table Tlt\_alpha\_minus}$$

$$Anion, X, \textit{ not defined in Table Dlt\_ranca}$$

$$Anion, X, \textit{ not defined in Table Rlt\_anca}$$

$$\text{false}$$

4.2.3 Procedure all_saltsInTables(Salt1,Salt2,...)

This procedure returns all diferent salts defined in a variable number of salt tables (minimum 2)

```
> all_saltsInTables:=proc()
> local i,Ltemp;
> if nargs < 2 then ERROR(`less than two arguments`) fi;
> for i from 1 to nargs do if not type(args[i],table) then ERROR(` argument`,
> i, ` is not of type table`) fi od;
> Ltemp:=convert([indices(args[1])],set) union convert([indices(args[2])],set);
> for i from 3 to nargs do Ltemp:=Ltemp union convert([indices(args[i])],set) od;
> convert(Ltemp,list);
> end:
```

Within a Maple procedure, the special name `nargs` has as its value the number of elements in the expression sequence args. Using the selection operation on this expression sequence, it follows that `args[i]` is the i'th actual argument. First of all this procedure verifies if the number of arguments is greater than 2. Afterwards, through the Maple boolean type-checking function `type(expr,type)` which returns `true` if `expr` is of type `type`, `false` otherwise, it checks through the call `type(args[i],table)` that the type of every procedure argument is `table`. If this is not the case, a call to the Maple `ERROR` function causes an immediate return to the point where the current procedure was invoked and the following message is printed: `Error, <in procname>` followed by the evaluated expression sequence specified in the call to `ERROR`. Only after having succeed this tests the Maple `convert` function is called to convert the list of indices of every input table to a `set` through the call `convert([indices(args[i])],set)`, the Maple keywords `union` is used for the `set` union and the final result is converted back to a `list`. For example, the following call

```
> all_saltsInTables(Dlt_alphatot,Tlt_alpha_plus,Tlt_alpha_minus);
```

$$[[\,Li, F\,], [\,Na, Br\,], [\,Li, Cl\,], [\,Na, F\,], [\,Cs, F\,], [\,Na, Cl\,], [\,Li, Br\,], [\,Na, I\,],$$
$$[\,Cs, Cl\,], [\,K, F\,], [\,K, Cl\,], [\,K, Br\,], [\,Rb, F\,], [\,Rb, Cl\,], [\,Rb, Br\,], [\,Li, I\,],$$
$$[\,K, I\,], [\,Cs, Br\,], [\,Rb, I\,], [\,Cs, I\,]]$$

returns all the 20 different salts which are defined in these three data tables.

4.2.4 Procedure all_commonsaltsInTables(Salt1,Salt2,...)

This procedure returns all common diferent salts defined in a variable number of data salt tables (minimum 2)

```
> all_commonsaltsInTables:=proc()
> local i,Ltemp;
> if nargs < 2 then ERROR(`less than two arguments`) fi;
> for i from 1 to nargs do if not type(args[i],table) then ERROR(` argument`,
> i, ` is not of type table`) fi od;
> Ltemp:=convert([indices(args[1])],set) intersect convert([indices(args[2])],
> set);
> for i from 3 to nargs do Ltemp:=Ltemp intersect convert([indices(args[i])],
> set) od;
> convert(Ltemp,list);
> end:
```

This procedure is similar to the previous one, the only difference is that set intersect is used instead of set union. For example, the call

```
> all_commonsaltsInTables(Dlt_alphatot,Tlt_alpha_plus,Tlt_alpha_minus);
```

$$[[\,Li, F\,], [\,Na, Br\,], [\,Li, Cl\,], [\,Na, F\,], [\,Na, Cl\,], [\,Li, Br\,], [\,Na, I\,], [\,Cs, Cl\,],$$
$$[\,K, F\,], [\,K, Cl\,], [\,K, Br\,], [\,Rb, F\,], [\,Rb, Cl\,], [\,Rb, Br\,], [\,Li, I\,], [\,K, I\,],$$
$$[\,Cs, Br\,], [\,Rb, I\,], [\,Cs, I\,]]$$

returns all 19 different which are common to these three data salt tables. Observe that it returns one salt less than the previous procedure, since salt `[Cs,F]` is not defined in all these three tables, specifically it is not defined in data table `Dlt_alphatot`.

These auxiliar procedures are used by the main procedure to verify that the needed data to do the calculation has already been defined by the user.

4.3 Main Procedures

In this section the main procedures that implement Equation 0 through Equation 3 — see Table 1, pg. 3 — are explained in detail.

4.3.1 Equation 0

The following procedure calculates the variation of polarizability with the pressure from data in tables `Dlt_fxan`, `Dlt_fxca` and `Dlt_ranca`, for a given salt `x:=[Anion,Cation]` as defined by equation 0 — *i.e.* equation 6 pg. 3.

```
> vapopres := proc(x:list)
> global Dlt_fxan,Dlt_fxca,Dlt_ranca;
> if is_in3tables(x) then Dlt_fxca[op(1,x),op(2,x)]*Dlt_ranca[op(2,x)] +
> Dlt_fxan[op(1,x),op(2,x)]*Dlt_ranca[op(1,x)] else false fi end:
```

First of all the procedure verifies through the call `is_in3tables(x)` that salt `x` is defined in the three salt data tables `Dlt_fxan`, `Dlt_fxca` and `Dlt_ranca`. If so, the value of $\frac{\partial\alpha_-}{\partial p}$ defined by equation 6 is returned. For example, the call

```
> vapopres([Li,F]);
```

$$-1.20532$$

returns $\frac{\partial\alpha_-}{\partial p} = -1.20532$ for salt LiF. On the other hand, executing the procedure for a salt such as `[X,F]` which is not defined, returns `false`. As the global variable `Debug` is set to `true`, the user is informed about the reasons of failure.

```
> vapopres([X,F]);
```

$$Salt, [\,X, F\,],\ \textit{not defined in Table Dlt\_fxan}$$

$$Salt, [\,X, F\,],\ \textit{not defined in Table Dlt\_fxca}$$

$$Anion, X,\ \textit{not defined in Table Dlt\_ranca}$$

$$\text{false}$$

Procedure `vapopres(Salt)` calculates the value of $\frac{\partial\alpha_-}{\partial p}$ defined by equation 6 for a given salt. The following procedure does the same executing `vapopres(Salt)` for all the different salts `Salt` defined in tables `Dlt_fxca` and `Dlt_fxan`.

```
> all_vapopres := proc()
> global Dlt_fxan,Dlt_fxca,Dlt_ranca;
> local allsalts,salt;
```

```
> print(`Salt  `, `Equation 0`);
> allsalts:=all_saltsInTables(Dlt_fxan,Dlt_fxca);
> for salt in allsalts do print(salt, vapopres(salt)) od;
> end:
```

If the user wants to know the result of equation 0 for all salts known by the system, this procedure can be ativated by the following call

```
> all_vapopres():
```

$$Salt\ \ , Equation\ 0$$

$$[Cs, Cl], -13.7575$$

$$[Cs, Br], -22.7064$$

$$[Cs, I], -42.6169$$

$$[K, Br], -22.7126$$

$$[Rb, F], -2.37172$$

$$[Rb, Cl], -14.6356$$

$$[Rb, Br], -24.0155$$

$$[Cs, F], -1.73321$$

$$[Na, Cl], -12.3524$$

$$[Na, Br], -20.6960$$

$$[K, F], -2.23923$$

$$[K, Cl], -13.8845$$

$$[Rb, I], -43.9831$$

$$[Li, Cl], -11.5092$$

$$[Li, F], -1.20532$$

$$[Na, I], -38.5717$$

$$[Li, I], -37.2202$$

$$[Li, Br], -19.7951$$

$$[K, I], -41.9739$$

$$[Na, F], -1.81476$$

This ended the necessary code for implementing equation 0 — *i.e.* equation 6 pg. 3.

4.3.2 Equation 1

The following procedure calculates $\frac{\partial \alpha_+}{\partial p}$ and $\frac{\partial \alpha_-}{\partial p}$ for a given salt `x:=[Anion,Cation]` from the system of equations defined by equation 1 — *i.e.* equation 7 pg. 3.

```
> der_alphas:=proc(x:list)
> global  Dlt_alphatot, Tlt_alpha_plus, Tlt_alpha_minus, Dlt_ranca, Rlt_anca;
> if not is_in4tables(x) then false else
> if not member(x,[indices(Dlt_alphatot)]) then print(`Salt`,x,` not defined in
> Table Dlt_alphatot`); false else
> solve({ (1/Tlt_alpha_plus[op(1,x),op(2,x)])*Dalpha_plus - (1/Tlt_alpha_minus[o
> p(1,x),op(2,x)])*Dalpha_minus=3*((1/Rlt_anca[op(1,x)])*Dlt_ranca[op(1,x)]-(1/Rl
> t_anca[op(2,x)])*Dlt_ranca[op(2,x)]), Dalpha_plus + Dalpha_minus=Dlt_alphat
> ot[op(1,x),op(2,x)]}) fi end:
```

First of all the procedure verifies through the call `is_in4tables(x)` that salt `x` is defined in the four data tables `Tlt_alpha_plus`, `Tlt_alpha_minus`, `Dlt_ranca`, and `Rlt_anca`. If so, the Maple function `solve(eqns)` is called for the system of two equations in terms of the two unknown variables `Dalpha_plus` and `Dalpha_minus`. For example, the call

```
> der_alphas([Li,F]);
```

$$\{ Dalpha\_plus = -.0267379, Dalpha\_minus = -.873260 \}$$

returns $\frac{\partial \alpha_+}{\partial p} = -.0267379$ and $\frac{\partial \alpha_-}{\partial p} = -.873260$. On the other hand, executing the procedure for a salt such as `[X,F]` which is not defined, returns `false`. As the global variable `Debug` is set to `true`, the user is informed about the reasons of failure.

```
> der_alphas([X,F]);
```

$$Salt, [\, X, F\,],\ \textit{not defined in Table Tlt\_alpha\_plus}$$

$$Salt, [\, X, F\,],\ \textit{not defined in Table Tlt\_alpha\_minus}$$

$$Anion, X,\ \textit{not defined in Table Dlt\_ranca}$$

$$Anion, X,\ \textit{not defined in Table Rlt\_anca}$$

$$\text{false}$$

The following procedure simply executes procedure `der_alphas(Salt)` for all salts `Salt` defined in tables `Tlt_alpha_plus`, `Tlt_alpha_minus` and `Dlt_alphatot`. If the necessary data to solve the system of equations for a specific salt is not found, the procedure informs that fact and continues.

```
> all_der_alphas:=proc()
> global   Tlt_alpha_plus, Tlt_alpha_minus, Dlt_alphatot;
> local allsalts,salt;
> allsalts:=all_saltsInTables(Tlt_alpha_plus,Dlt_alphatot,Dlt_alphatot);
> print(`Salt `, `Equation 1`);
> for salt in allsalts do print(salt,der_alphas(salt)) od;
> end:
```

An example of execution is given bellow

```
> all_der_alphas();
```

$$\textit{Salt , Equation 1}$$

$$[Rb, Cl], \{Dalpha\_minus = -5.84264, Dalpha\_plus = -5.85738\}$$

$$[Li, Br], \{Dalpha\_minus = -14.1794, Dalpha\_plus = -.0207014\}$$

$$[K, Cl], \{Dalpha\_minus = -6.84922, Dalpha\_plus = -2.85079\}$$

$$[Rb, F], \{Dalpha\_plus = -5.45342, Dalpha\_minus = .35339\}$$

$$[Rb, Br], \{Dalpha\_minus = -11.7583, Dalpha\_plus = -5.74166\}$$

$$[Cs, Cl], \{Dalpha\_minus = -7.41032, Dalpha\_plus = -12.2196\}$$

$$[Li, Cl], \{Dalpha\_plus = -.0227871, Dalpha\_minus = -7.47721\}$$

$$[Cs, Br], \{Dalpha\_plus = -12.6747, Dalpha\_minus = -15.1051\}$$

$$[Li, F], \{Dalpha\_minus = -.873262, Dalpha\_plus = -.0267381\}$$

$$[K, F], \{Dalpha\_minus = -.434872, Dalpha\_plus = -2.86513\}$$

$$[Na, Br], \{Dalpha\_minus = -12.2924, Dalpha\_plus = -.207609\}$$

$$[Li, I], \{Dalpha\_minus = -29.6819, Dalpha\_plus = -.017933\}$$

$$[Na, I], \{Dalpha\_minus = -26.0258, Dalpha\_plus = -.174147\}$$

$$[K, I], \{Dalpha\_plus = -2.08071, Dalpha\_minus = -23.3193\}$$

$$[Rb, I], \{Dalpha\_minus = -26.2716, Dalpha\_plus = -5.52840\}$$

$$[Cs, I], \{Dalpha\_minus = -33.3377, Dalpha\_plus = -13.1424\}$$

$$[Na, F], \{Dalpha\_plus = -.342295, Dalpha\_minus = -1.15771\}$$

$$\textit{Salt}, [Cs, F], \textit{ not defined in Table Dlt\_alphatot}$$

$$[Cs, F], \text{false}$$

$$[K, Br], \{Dalpha\_minus = -12.2857, Dalpha\_plus = -2.61437\}$$

$$[Na, Cl], \{Dalpha\_minus = -7.04374, Dalpha\_plus = -.256220\}$$

4.3.3 Equation 2

The following procedure calculates $\frac{\partial \alpha_-}{\partial p}$ for a given salt `x:=[Anion,Cation]` given `y:=`$\frac{\partial \alpha_+}{\partial p}$, as defined by equation 2 — *i.e.* equation 8 pg. 3.

```
> der_alphaminus:=proc(x:list,y:float)
> global Tlt_alpha_plus, Tlt_alpha_minus, Dlt_ranca, Rlt_anca;
> if not is_in4tables(x) then false else
> Tlt_alpha_minus[op(1,x),op(2,x)]*((1/Tlt_alpha_plus[op(1,x),op(2,x)])*y -
> 3*((1/Rlt_anca[op(1,x)])*Dlt_ranca[op(1,x)]-(1/Rlt_anca[op(2,x)])*
> Dlt_ranca[op(2,x)])) fi end:
```

First of all the procedure verifies through the call `is_in4tables(x)` that salt `x` is defined in the four data tables `Tlt_alpha_plus`, `Tlt_alpha_minus`, `Dlt_ranca`, and `Rlt_anca`. If so, $\frac{\partial \alpha_-}{\partial p}$ for that salt in function of the given value `y:=`$\frac{\partial \alpha_+}{\partial p}$ is returned. For example, the call

```
> der_alphaminus([Li,F],-0.0863);
```

$$-2.64027$$

returns $\frac{\partial \alpha_-}{\partial p} = -2.64027$ for salt LiF given as input that $\frac{\partial \alpha_+}{\partial p} = -0.0863$, value already calculated by Batana — see Table 7, pg 25. On the other hand, executing the procedure for a salt such as `[X,F]` which is not defined, returns `false`. As the global variable `Debug` is set to `true`, the user is informed about the reasons of failure.

```
> der_alphaminus([X,F],-0.8866);
```

Salt, [*X*, *F*], *not defined in Table Tlt_alpha_plus*

Salt, [*X*, *F*], *not defined in Table Tlt_alpha_minus*

Anion, *X*, *not defined in Table Dlt_ranca*

Anion, *X*, *not defined in Table Rlt_anca*

false

The following procedure simply executes procedure `der_alphaminus(x,y)` for all salts `x` defined in tables `Tlt_alpha_plus`, `Tlt_alpha_minus` , and table `x1`. If the necessary data for a salt is not found, the procedure informs that fact and continues.

```
> all_der_alphaminus:=proc(x1:table)
> global  Tlt_alpha_plus, Tlt_alpha_minus;
> local commonsalts,salt;
```

```
> commonsalts:= all_commonsaltsInTables(Tlt_alpha_plus,Tlt_alpha_minus,x1);
> print(`Salt `, `Equation 2`);
> for salt in commonsalts do print(salt,der_alphaminus(salt,x1[op(1,salt),
> op(2,salt)])) od;
> end:
```

For example, if the user wants to execute this procedure using the values of $\frac{\partial \alpha_+}{\partial p}$ found by Batana — see Table 7 — then he/she can define a table holding those values, for example

```
> Dlt_alpha_plus_calculatedbyBatana:=
>  table( [(Li,F)=-0.0863, (Li,Cl)=-0.00168, (Li,I)=-0.0889, (Li,Br) =-0.0365,
> (Na,F)= -0.8866, (Na,Cl)= -1.0440, (Na,I) = -1.2131,(Na,Br)=-1.1147,
>  (K,F)= -6.8022,(K,Cl)=-5.8603,(K,I)= -5.7480,(K,Br)= -5.8458,
> (Rb,F)= -12.0415, (Rb,Cl)=-10.1647, (Rb,I)= -9.8213, (Rb,Br)= -10.2190]):
```

afterwards the procedure can be called with that table as follows

```
> all_der_alphaminus(Dlt_alpha_plus_calculatedbyBatana):
```

$$\mathit{Salt}\ ,\ \mathit{Equation\ 2}$$

$$[Rb, Cl], -16.8502$$

$$[Li, Br], -16.3543$$

$$[K, Cl], -19.3331$$

$$[Rb, F], -5.40508$$

$$[Rb, Br], -26.9481$$

$$[Li, Cl], -5.40872$$

$$[Li, F], -2.64027$$

$$[K, F], -6.26758$$

$$[Na, Br], -38.7029$$

$$[Li, I], -44.3246$$

$$[Na, I], -70.9060$$

$$[K, I], -53.4726$$

$$[Rb, I], -47.8952$$

$$[Na, F], -4.93451$$

$$[K, Br], -30.3977$$

$$[Na, Cl], -23.8711$$

4.3.4 Equation 3

The following procedure calculates $\frac{\partial\alpha_+}{\partial p}$ for a given salt `x:=[Anion,Cation]` given `y:=`$\frac{\partial\alpha_-}{\partial p}$, as defined by equation 3 — *i.e.* equation 9 pg. 4. Its implementation is similar to the previous one but for equation 9.

```
> der_alphaplus:=proc(x:list,y:float)
> global  Tlt_alpha_plus, Tlt_alpha_minus, Dlt_ranca, Rlt_anca;
> if not is_in4tables(x) then false else
> Tlt_alpha_plus[op(1,x),op(2,x)]*(3*((1/Rlt_anca[op(1,x)])*Dlt_ranca[op(1,x)]-
> (1/Rlt_anca[op(2,x)])*Dlt_ranca[op(2,x)])+ (1/Tlt_alpha_minus[op(1,x),
> op(2,x)])*y )
>  fi end:
```

For example, the call

```
> der_alphaplus([Li,F],-1.2053);
```

$$-.0379305$$

returns $\frac{\partial\alpha_+}{\partial p} = -.0379305$ for salt LiF given as input that $\frac{\partial\alpha_-}{\partial p} = -1.2053$, value already calculated by Batana — see Table 7, pg 25. On the other hand, executing the procedure for a salt such as `[X,F]` which is not defined, returns `false`. As the global variable `Debug` is set to `true`, the user is informed about the reasons of failure.

```
> der_alphaplus([X,F],-1.8147);
```

Salt, [X, F], *not defined in Table Tlt_alpha_plus*

Salt, [X, F], *not defined in Table Tlt_alpha_minus*

Anion, X, *not defined in Table Dlt_ranca*

Anion, X, *not defined in Table Rlt_anca*

false

The following procedure simply executes procedure `der_alphaplus(x,y)` for all salts `x` defined in tables `Tlt_alpha_plus`, `Tlt_alpha_minus` and table `x1`. If the necessary data for a salt is not found, the procedure informs that fact and continues.

```
> all_der_alphaplus:=proc(x1:table)
> global  Tlt_alpha_plus, Tlt_alpha_minus;
> local commonsalts,salt;
```

```
> commonsalts:= all_commonsaltsInTables(Tlt_alpha_plus,Tlt_alpha_minus,x1);
> print(`Salt `, `Equation 3`);
> for salt in commonsalts do print(salt,der_alphaplus(salt,x1[op(1,salt),op(2,salt
> )])) od;
> end:
```

For example, if the user wants to execute this procedure using the values of $\frac{\partial \alpha_-}{\partial p}$ found by Batana — see Table 7 — then he/she can define a table holding those values, for example

```
> Dlt_alpha_minus_calculatedbyBatana:=
>  table( [(Li,F)=-1.2053, (Li,Cl)=-11.4092, (Li,I)=-37.2202, (Li,Br) =-19.7951,
> (Na,F)= -1.8147, (Na,Cl)= -12.3525, (Na,I) = -38.5717,(Na,Br)=-20.6960,
> (K,F)= -2.2392,(K,Cl)= -13.8845,(K,I)= -41.9741,(K,Br)= -22.7125,
> (Rb,F) = -2.3717, (Rb,Cl)=-14.6357, (Rb,I) =  -43.9831, (Rb,Br) =  -24.0155,
> (Cs,Cl)=-13.7575,(Cs,F)=-1.7332,(Cs,I)=-42.6170,(Cs,Br)=-22.7064]):
```

afterwards the procedure can be called as follows

```
> all_der_alphaplus(Dlt_alpha_minus_calculatedbyBatana):
```

$$\mathit{Salt}\;, \mathit{Equation\ 3}$$

$$[\,Rb, Cl\,], -9.29814$$

$$[\,Li, Br\,], -.0614937$$

$$[\,K, Cl\,], -4.54680$$

$$[\,Rb, F\,], -8.57111$$

$$[\,Rb, Br\,], -9.35460$$

$$[\,Cs, Cl\,], -16.4034$$

$$[\,Li, Cl\,], -.0629094$$

$$[\,Cs, Br\,], -16.4917$$

$$[\,Li, F\,], -.0379305$$

$$[\,K, F\,], -4.08306$$

$$[\,Na, Br\,], -.496238$$

$$[\,Li, I\,], -.0544683$$

$$[\,Na, I\,], -.464577$$

$$[\,K, I\,], -4.34954$$

$$[\,Rb, I\,], -9.04465$$

$$[\,Cs, I\,], -16.3309$$

$$[\,Na, F\,], -.436981$$

$$[\,Cs, F\,], -14.4919$$

$$[\,K, Br\,], -4.47466$$

$$[\,Na, Cl\,], -.504751$$

If the values of $\frac{\partial\alpha_-}{\partial p}$ found by Fowler are to be used — see Table 7 — then he/she can define a table holding those values, for example

---

```
> Dlt_alpha_minus_calculatedbyFowler:=
>  table( [(Li,F)=-0.858, (Li,Cl)=-4.89824, (Li,I)=-16.02, (Li,Br) =-8.56206,
> (Na,F)= -1.368, (Na,Cl)= -6.269, (Na,I) = -18.39,(Na,Br)=-8.994, (K,F)= -2.605,
> (K,Cl)= -9.424,(K,I)= -21.48,(K,Br)= -13.714,(Rb,F) = -3.372, (Rb,Cl)=-11.779,
>  (Rb,I) =  -25.75, (Rb,Br) =  -17.134, (Cs,Cl)=-8.89843]):
```

---

and call the procedure using Fowler data table as follows

```
> all_der_alphaplus(Dlt_alpha_minus_calculatedbyFowler):
```

$$Salt\,,\,Equation\ 3$$

$$[\,Rb, Cl\,], -8.18030$$

$$[\,Li, Br\,], .0201024$$

$$[\,K, Cl\,], -3.47150$$

$$[\,Rb, F\,], -9.71552$$

$$[\,Rb, Br\,], -7.32619$$

$$[\,Cs, Cl\,], -13.2005$$

$$[\,Li, Cl\,], .00352890$$

$$[\,Li, F\,], -.0262236$$

$$[\,K, F\,], -4.32997$$

$$[\,Na, Br\,], -.0943226$$

$$[\,Li, I\,], .0482790$$

$$[\,Na, I\,], .00261954$$

$$[\,K, I\,], -1.85701$$

$$[\,Rb, I\,], -5.42484$$

$$[\,Na, F\,], -.372604$$

$$[\,K, Br\,], -2.86920$$

$$[\,Na, Cl\,], -.219950$$

4.4 Summary of Results

The following table shows all the results obtained using the system using data tables from Apendix 6.

Salt	Eq. 0 $\frac{\partial\alpha_-}{\partial p}$	Eq. 1 $\frac{\partial\alpha_-}{\partial p}$	Eq. 2 $\frac{\partial\alpha_-}{\partial p}(Batana)$	Eq. 1 $\frac{\partial\alpha_+}{\partial p}$	Eq. 3 $\frac{\partial\alpha_+}{\partial p}(Batana)$	Eq. 3 $\frac{\partial\alpha_+}{\partial p}(Fowler)$
LiF	-1.20532	-.873262	-2.64027	-.0267381	-.0379305	-.0262236
LiCl	-11.5092	-7.47721	-5.40872	-.0227871	-.0629094	.00352890
LiBr	-19.7951	-14.1794	-16.3543	-.0207014	-.0614937	.0201024
LiI	-37.2202	-29.6819	-44.3246	-.017933	-.0544683	.0482790
NaF	-1.81476	-1.15771	-4.93451	-.342295	-.436981	-.372604
NaCl	-12.3524	-7.0437	-23.8711	-.256220	-.504751	-.219950
NaBr	-20.6960	-12.2924	-38.7029	-.207609	-.496238	-.0943226
NaI	-38.5717	-26.0258	-70.9060	-.174147	-.464577	.00261954
KF	-2.23923	-.434872	-6.26758	-2.86513	-4.08306	-4.32997
KCl	-13.8845	-6.84922	-19.3331	-2.85079	-4.54680	-3.47150
KBr	-22.7126	-12.2857	-30.3977	-2.61437	-4.47466	-2.86920
KI	-41.9739	-23.3193	-53.4726	-2.08071	-4.34954	-1.85701
RbF	-2.37172	.35339	-5.40508	-5.45342	-8.57111	-9.71552
RbCl	-14.6356	-5.84264	-16.8502	-5.85738	-9.29814	-8.18030
RbBr	-24.0155	-11.7583	-26.9481	-5.74166	-9.35460	-7.32619
RbI	-43.9831	-26.2716	-47.8952	-5.52840	-9.04465	-5.42484
CsF	-1.73321				-14.4919	
CsCl	-13.7575	-7.41032		-12.2196	-16.4034	-13.2005
CsBr	-22.7064	-15.1051		-12.6747	-16.4917	
CsI	-42.6169	-33.3377		-13.1424	-16.3309	

Table 4: Results

5 Conclusions

The computational system implemented allows for:

1. Different ways of evaluating anion and cation polarizabilities and their pressure dependence:

   (a) starting from data $p = 0$;

   (b) when high pressure data are available.

2. The evaluation of $\frac{\partial r_i}{\partial p}$ is used in one of the methods and we here give an alternative way of obtaining these quantities.

3. When high pressure data are not available we can give an estimation of $\frac{\partial \alpha}{\partial p}$ (and $-V\frac{\partial \epsilon_\infty}{\partial V}$) in two different ways:

   (a) based on the new method propossed using $\frac{\partial \alpha_-}{\partial p}$ from ab initio calculations [3]

   (b) i. when $\frac{\partial \alpha_-}{\partial p}$ is calculated starting from data at $p = 0$ and $\frac{\partial \alpha_+}{\partial p}$ is evaluated with the new method

      ii. when both $\frac{\partial \alpha_-}{\partial p}$ and $\frac{\partial \alpha_+}{\partial p}$ are evaluated from data at $p = 0$ with the method discussed previously [4, 6]

4. The programs are written taking into account maximum flexibility, with respect to alternatives. As the system runs interactively, using the main procedures the user can easily program different outputs allowing for a comparison of results that is important when new values are proposed, particularly knowing the intrinsic limitations of each method.

5. Similarly, the user can easily program any sort of analysis of errors to help his decisions.

References

[1] Batana, A.; Bruno, J.; Munn, R.W. Anion Polarizability in Alkali Halide Crystals. *Molecular Physics.* **1997**, 92(6), 1029–1033.

[2] Char, B.W.; Geddes, K.O.; Gonnet, G.O.; Leong, B.L.; Monagan, M.B.; Watt, S.M. *Maple V Language Reference Manual.* Springer-Verlag, **1991**

[3] Jemmer, P.; Fowler, P.W.; Wilson, M.; Madden, P.A. Envirinmental Effects on Anion Polarizability: Variation with Lattice Parameter and Coordination Number. *J. Phys. Chem.* **1998**.

[4] Monard, M.C.; Bruno, J.; Batana, A. A Computational System for the Calculation of Polarizabilities and their Pressure Dependence. Tech. Rep. ICMSC-USP, No. 51, **1997**. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_51.ps.zip`

[5] Monard, M.C.; Batana, A.; Bruno, J. A Computational System for the Calculation of the Pressure Dependence of Ionic Radii in Crystal. Tech. Rep. ICMSC-USP, No. 82, **1999**. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_82.ps.zip`

[6] Monard, M.C.; Batana, A.; Bruno, J. Calculation of the Pressure Dependence of Anionn Polarizabilities in Crystals. *Computers & Chemistry.* **1999**, 23, 493–501.

[7] Monard, M.C.; Batana, A.; Bruno, J. Calculation of Ionic Radii at High Pressure in Different Crystals. Tech. Rep. ICMSC-USP,No. 112, **2000**. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_112.ps.zip`

[8] Wilson, M.; Madden, P.A.; Jemmer, P.; Fowler, P.W.; Batana, A.; Bruno, J.; Munn, R.W.; Monard, M.C. Models of Anion Polarizability. *Mol. Phys.* **1999**, 96(10), 1457.

6 Appendix - Experimental Data

The following values for different salts — Table 5 — anions and cations— Table 6 — at Low Temperature have been used as data.

Salt	$\left.\frac{\partial\alpha_-}{\partial R}\right)_+$	$\left.\frac{\partial\alpha_-}{\partial R}\right)_-$	$\frac{\partial\alpha_{total}}{\partial p}$	α_+	α_-
LiF	1.54448	0.60838	-0.9	0.030	0.89
LiCl	5.88438	0.86975	-7.5	0.030	2.94
LiBr	7.27507	0.89898	-14.2	0.030	4.13
LiI	9.17727	1.08272	-29.7	0.030	6.19
NaF	2.19991	0.40023	-1.5	0.147	1.02
NaCl	6.16093	0.65620	-7.3	0.147	3.14
NaBr	7.48394	0.69642	-12.5	0.147	4.28
NaI	9.40833	0.85392	-26.2	0.147	6.35
KF	2.53621	0.26056	-3.3	0.81	1.20
KCl	6.74405	0.47053	-9.7	0.81	3.36
KBr	8.05675	0.51719	-25.4	0.81	4.54
KI	10.09774	0.65512	-14.9	0.81	6.66
RbF	2.58094	0.22300	-5.1	1.35	1.18
RbCl	7.02686	0.41127	-11.7	1.35	3.45
RbBr	8.45423	0.45273	-17.5	1.35	4.58
RbI	10.51739	0.58267	-31.8	1.35	6.80
CsF	1.61768	0.18232		2.34	1.27
CsCl	6.57648	0.32780	-19.63	2.34	3.55
CsBr	7.96919	0.36480	-27.78	2.34	4.66
CsI	10.17983	0.46841	-46.48	2.34	6.81

Table 5: Data Values for Salts at Low Temperatures

Anion	$\frac{\partial r_+}{\partial p}$	r_+	Cation	$\frac{\partial r_-}{\partial p}$	r_-
Li	-0.2780	0.60	F	-0.6709	1.36
K	-2.0636	1.33	Cl	-1.9148	1.81
Rb	-2.8707	1.48	Br	-2.6866	1.95
Na	-0.8466	0.95	I	-4.0229	2.16
Cs	-3.5537	1.69			

Table 6: Data Values for Anions and Cations at Low Temperatures

The following values for different salts — Table 7 — calculated by Fowler and Batana, have been used as test data for procedures — see Section 3.1

- `der_alphaminus(Salt,DerAplphaPlus)` and
- `der_alphaplus(Salt,DerAplphaPlus)`

Salt	Fowler	Batana	
	$\frac{\partial\alpha_-}{\partial p}$	$\frac{\partial\alpha_-}{\partial p}$	$\frac{\partial\alpha_+}{\partial p}$
LiF	-0.858	-1.2053	-0.0863
LiCl	-4.89824	-11.4092	-0.00168
LiBr	-8.56206	-19.7951	-0.0365
LiI	-16.02	-37.2202	-0.0889
NaF	-1.368	-1.8147	-0.8866
NaCl	-6.269	-12.3525	-1.0440
NaBr	-8.994	-20.6960	-1.1147
NaI	-18.39	-38.5717	-1.2131
KF	-2.605	-2.2392	-6.8022
KCl	-9.424	-13.8845	-5.8603
KBr	-13.714	-22.7125	-5.8458
KI	-21.48	-41.9741	-5.7480
RbF	-3.372	-2.3717	-12.0415
RbCl	-11.779	-14.6357	-10.1647
RbBr	-17.134	-24.0155	-10.2190
RbI	-25.75	-43.9831	-9.8213
CsF		-1.7332	
CsCl	-8.89843	-13.7575	
CsBr		-22.7064	
CsI		-42.6170	

Table 7: Test Values for Salts at Low Temperatures