Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2569

A COMPUTATIONAL SYSTEM FOR THE CALCULATION OF POLARIZABILITIES AND THEIR PRESSURE DEPENDENCE

MARIA CAROLINA MONARD
JORGE BRUNO
ALICIA BATANA

Nº 51

RELATÓRIOS TÉCNICOS DO ICMSC

São Carlos
Jan./1997

A Computational System for the Calculation of Polarizabilities and their Pressure Dependence

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

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

Abstract

In this work we present and discuss a new way of evaluating the r dependence of the ionic polarizabilities particularly for anions from experimental dielectric data. The Clausius Mossotti treatment is here applied to alkali halides, for which high pressure experimental data are available for comparison. This work requires computational support to analyze experimental data and give the ionic information we propose. To accomplish this task, a computational system which has been implemented in Maple V — a system for mathematical computation — is presented, together with an analysis of the results.

*Work partially supported by National and State Research Council — CNPq and FAPESP, Brasil
†Work partially supported by CONICET, Argentina

Contents

List of Figures

List of Tables

1 Introduction

Polarizabilities and dispersion coefficients for ions are important parameters in the theoretical description of the optical and mechanical properties of ionic solids. Polarizabilities of ions occur in the theories of refractive index and dielectric constant of ionic crystals and in the most widely used approximate theories of the dispersion contribution to the lattice energy. They also provide a starting point for the description of collision-induced spectra of ionic melts. It is therefore important to obtain reliable values for these quantities.

Refractive index can be a useful guide to the electronic structure of a crystalline solid, since its volume depends on the distribution of polarizable matter within the lattice. In an ionic solid there are strong interactions between component ions, but the polarizability density may still be assigned to a sum of contributions from individual ions. The effective polarizabilities of ions in crystals have been studied by ab initio methods[2, 3, 4] and there is now a good understanding of the calculation of these quantities and the physical factors that determine their values[5]. Anions in crystals have polarizabilities that are significantly smaller than those of the isolated ions, and show a strong variation with crystal; cations of rare gas electronic configurations are insensitive to environment and have near-constant polarizabilities. Anions in crystals are smaller, less polarizable and more strongly bound than their free counterparts; in particular, calculations of in-crystal polarizabilities of small ions such as F^- , O^{--} and Cl^- show dramatic reductions arising from electrostatic and overlap compression of the anion by its cationic neighbours[2, 3, 4], which must be taken into account when modelling refractive indices or lattice energies.

As we have mentioned much progress have been made in the study of ion polarizabilities of relatively simple crystals[6, 7, 8, 9, 11, 12, 13]. Future challenges are now

1. the study of crystals with polyatomic ions
2. the study of the volume and pressure dependence of ionic polarizabilities, related to the volume and pressure dependence of dielectric constants, which are useful for critical tests of the theory of dielectric polarization and interionic forces operative in crystals

These two issues are treated in the present work, starting from the Clausius-Mossotti treatment we propose different ways of obtaining information of ionic polarizabilities and their volume and pressure dependence from experimental data, starting from relatively simple crystals and extending the treatment when more complex ions are involved. This work requires a strong computational support to analize experimental data and give the ionic information we propose. To accomplish this task, a computational system, described in some details, has been implemented in Maple V[1] — a system for mathematical computation.

This work is organized as follows: in Section 2 the Clausius-Mossoti treatment is described. Section 3 presents the experimental data used to obtain information about ionic polarizabilities and their dependence. Section 4 provides some basic knowledge for using the

[1] Release 3 for MS Windows

Maple code that implements the computational system which is described in Section 5. Section 6 shows the experimental results obtained and Section 7 presents our conclusions.

2 Clausius-Mossoti

Ionic crystals: for these ions Na^+ | F^- Cl^- Br^- I^- where Na^+Cl^- for example is the salt NaCl and

- α_+ refers for the cation polarizability of Na and
- α_- refers to the anion polarizability of F, Cl, Br and I

Clausius-Mossoti equation establishes the following

$$\frac{\varepsilon - 1}{\varepsilon + 2} = \frac{4\pi}{3}\frac{\alpha}{V} \tag{1}$$

where α is the sum of the cation and anion polarizabilities, i.e.

$$\alpha = \alpha_+ + \alpha_-$$

and ε is the dielectric constant. The values

- ε_∞ the dielectric constant at frequency ∞ and
- ε_0 static dielectric constant

are taken from experiments. For a given salt, we consider

$$V = Cr_0^3$$

where

- r_0 interionic distance.
- C constant with different values, for each of the four following structures

C	Structure
2	fcc (face-centered cubic)
$\frac{8}{3\sqrt{3}}$	bcc
$\frac{16}{3\sqrt{3}}$	fluorite
$\frac{16}{3\sqrt{3}}$	Zn blende

In this work the two first structures — fcc and bcc — will be considered.

Starting from the model included in the Clausius-Mossotti treatment, a symbolic computational program has been developed to study α_- and its dependence with interionic distance r, assuming α_+ constant for a given ion in different salts and using experimental data of dielectric constants. Values of $\partial\alpha_-/\partial r$ are obtained and compared with $\partial\alpha/\partial r$ obtained from experimental data of the volume dependence of dielectric constants. Values for α_- and $\partial\alpha_-/\partial r$ are calculated for

1. fixed cation
2. fixed anion

It follows a description of the experimental data used in this work.

3 Experimental Data

Data on Tables 1 and 2 have been taken from the experimental data at Low Temperature – LT – and Room Temperature – RT – described in [13] in the following way

- r_0 (RT and LT) from Table 10.1 – pg. 115.
- ε_0 and ε_∞ (RT and LT) from Table 10.3 – pg. 117
- $\mathcal{X}_T$ from Table 10.2 – pg. 116
- D_1 from Table 10.4 – pg.118
- D_2 from Table 10.5 – pg.119
- In Table 1 some data was not available at LT. In some cases — data in *italic* — that missing data has been substituted by data at RT

3.1 Low Temperature

Struct.	Salt	α_+ $10^{-24}cm^3$	α_- $10^{-24}cm^3$	r_0 $10^{-8}cm$	ε_0	ε_∞	$\mathcal{X}_T$ $10^{-12}cm^2dyn^{-1}$	D_1	D_2
fcc	Li	0.030							
C=2	LiF		0.89	1.996	8.50	1.93	1.43	-	-
	LiCl		2.94	2.539	10.83	2.79	3.20	-	-
	LiBr		4.13	2.713	11.95	3.22	*4.20*	-	-
	LiI		6.19	2.951	15.28	3.89	*5.7803*	-	-
	Na	0.147							
	NaF		1.02	2.295	4.73	1.75	1.99	-	-
	NaCl		3.14	2.789	5.43	2.35	3.98	-	-
	NaBr		4.28	2.954	5.78	2.64	4.70	-	-
	NaI		6.35	3.194	6.62	3.08	6.45	-	-
	K	0.81							
	KF		1.20	2.648	5.11	1.86	3.10	-	-
	KCl		3.36	3.116	4.49	2.20	5.26	-	-
	KBr		4.54	3.262	4.52	2.39	6.17	-	-
	KI		6.66	3.489	4.66	2.68	7.75	-	-
	Rb	1.35							
	RbF		1.18	2.789	5.99	1.94	*3.81*	-	-
	RbCl		3.45	3.259	4.53	2.20	*6.40*	-	-
	RbBr		4.58	3.410	4.51	2.36	*7.69*	-	-
	RbI		6.80	3.628	4.55	2.61	*9.48*	-	-
	Cs	2.34							
	CsF		1.27	2.9827	7.27	2.17	*4.25*	-	-
bcc	Cs	2.34							
C= $\frac{8}{3\sqrt{3}}$	CsCl		3.55	3.5150	6.68	2.67	4.5387	-	-
	CsBr		4.66	3.6645	6.38	2.83	5.4476	-	-
	CsI		6.81	3.9018	6.29	3.09	6.9396	-	-

Table 1: Experimental Data at Low Temperatures

3.2 Room Temperature

Struct.	Salt	α_+ $10^{-24}cm^3$	α_- $10^{-24}cm^3$	r_0 $10^{-8}cm$	ϵ_0	ϵ_∞	χ_T $10^{-12}cm^2dyn^{-1}$	D_1	D_2
fcc	Li	0.030							
C=2	LiF		0.89	2.0132	9.032	1.93	1.54	23.0	3.82
	LiCl		2.94	2.5698	11.989	2.75	3.41	30.4	8.62
	LiBr		4.13	2.7498	13.295	3.16	4.20	40.4	13.0
	LiI		6.19	3.0003	16.904	3.80	5.7803	-	-
	Na	0.147							
	NaF		1.02	2.3173	5.094	1.74	2.11	12.5	5.37
	NaCl		3.14	2.8203	5.9165	2.33	4.26	13.0	9.25
	NaBr		4.28	2.9871	6.298	2.60	5.03	14.9	11.9
	NaI		6.35	3.2372	7.304	3.01	7.07	16.6	15.1
	K	0.81							
	KF		1.20	2.6735	5.514	1.85	3.30	12.8	7.63
	KCl		3.36	3.1467	4.853	2.17	5.63	8.45	10.2
	KBr		4.54	3.2999	4.914	2.36	6.70	8.35	11.6
	KI		6.66	3.5331	5.115	2.65	8.54	8.16	13.8
	Rb	1.35							
	RbF		1.18	2.8203	6.497	1.93	3.81	14.5	8.47
	RbCl		3.45	3.2986	4.934	2.18	6.40	7.79	10.2
	RbBr		4.58	3.4482	4.884	2.34	7.69	7.46	11.7
	RbI		6.80	3.6711	4.955	2.58	9.48	6.94	13.4
	Cs	2.34							
	CsF		1.27	3.0400	8.108	2.16	4.25	-	-
bcc	Cs	2.34							
C= $\frac{8}{3\sqrt{3}}$	CsCl		3.55	3.559	7.000	2.62	5.4745	15.8	-
	CsBr		4.66	3.712	6.680	2.78	6.4212	14.7	-
	CsI		6.81	3.861	6.600	2.96	7.9979	11.4	-

Table 2: Experimental Data at Room Temperatures

4 Introduction to Maple V

Maple V is a *Symbolic Computational System* or *Computer Algebra System* which first appeared commercially in 1995. Today, Maple V is one of the most powerfull and reliable systems currently available for doing mathematics by computer. Maple V is accurate, expandable and *interactive*; this interaction makes it easier to test and correct programs. Maple V is available for nearly every operating system available, including Macintosh, MS Windows, MS DOS, UNIX, VMS, NeXT, Ultrix, and UNICOS.

Maple V can perform symbolic and numeric mathematics, generate two- and three-dimentional graphics and animations, display output using standard mathematical notation, and provide a high-level programming language similar to Pascal. Unlike traditional programming languages, the Maple language contains many powerful commands which allow to perform complicated tasks with a single command. It comes with a large library of prewritten routines, including graphical display primitives. Maple has a wide range of commands for generating both two- and three- dimensional plots. In this work two- dimensional plots are used based on the many specialized graphics routines found in the `plot` package. The input to these commands is typically one or more Maple formulae,

operations or functions, along with information about domains and possibly ranges. In all cases, the graphics commands allow for the setting of options, specifying such attributes as coloring, shading, or axes style.

Maple V's extensive mathematical funcionality is easily accessed through its advanced worksheet-based graphical interface. The worksheet is an integrated environment in which the user interactively solves problems and documents his/her work. Worksheets contain not only text and conventional document-processing information, but live mathematical commands and their automatically generated results. They also provide easy ways to change the sequence of commands and to re-execute them. Since Maple V worksheets share a common format, they can be taken from one platform to another. In this work Maple V-Release 3 for MS Windows is used.

The aim of this section is to provide basic knowledge for proficiently using the Maple code that implements the computational system described in this work. The user interface for this system is based on the concept of worksheets. The four types of regions that can be included in a worksheet are *input*, *output*, *text*, and *graphics*. The user cannot physically manipulate output or graphics regions but can edit text and input regions.

In addition to having individual regions, Maple V for Windows worksheets also have *region groups*. A region group may consist of input regions, any related output regions, and any adjacent text region. The system implemented makes extensive use of region groups to execute logical units. To see where the Region Groups are, the user can turn on Separators Lines in the View menu. Placing the cursor anywhere within an input region of a region group and pressing Enter or double-click on the left mouse button, the user automatically re-executes all the input lines in that region group. This is useful when recalculating a worksheet.

The worksheet-based user interface also allows to recalculate an entire Maple worksheet by selecting Execute Worksheet from the Format menu. All Maple input will be executed, and the output areas updated. The prompt character > indicates that Maple is waiting for input. A command is followed immediately by its result. If a command is completed with a ";" then Maple echoes the result. If completed with a ":" the result is not echoed.

Plots are created, named and manipulated much like any other algebraic object in Maple V, but are displayed in a separate window. These windows can be moved about the screen, resized, and closed in standard Windows fashion. Resizing a plot window redraws the plot to fill the new window size. Multiple plot windows can be opened concurrently or closed at any time. If a plot window has been closed, the plot can be redisplayed either by referring to the plot object by name or by re-entering the appropriate plot command from the worksheet.

5 Implementation

The computational system implemented to obtain information of ionic polarizabilities and their volume and pressure dependence from experimental data, based in the Clausius-Mossoti treatment, is here described in some details. First of all the data structure used

to represent the experimental data is discussed.

5.1 Data Structure

Maple has many available data structures, such as expression sequences, lists, sets, arrays and tables. A table is an extension of the concept of the array data structure. The difference between an array and a table is that a table can have anything for indices, not just integers.

In this work, the experimental data in Tables 1 and 2 pg 4, are defined using Maple V table data structure. A table named `fcc_LT` represents the low temperature data shown in Table 1 while data from Table 2 is represented by a similar table named `fcc_RT`. To ilustrate, table `fcc_LT` is shown bellow.

```
> # Experimental Data at Low Temperatures --- fcc  C = 2
> # Structure defined is such that the first index refers to cations'
> # alfa_plus, the 2nd index refers to a list of 7 elements related with
> # the salt (-1 indicates that this information does not exist) where:
> # 1 -> alfa_minus,2 -> r,3 -> Ep_0,4 -> Ep_inf,5 ->Xt,6 -> D1,7 -> D2
```

```
> fcc_LT :=table([Li=0.030,
>                          LiF=[0.89, 1.996, 8.50, 1.93, 1.43, -1, -1],
>                          LiCl=[2.94,2.539, 10.83, 2.79, 3.20, -1, -1],
>                          LiBr=[4.13, 2.713, 11.95, 3.22, 4.20, -1, -1],
>                          LiI= [6.19, 2.951, 15.28, 3.89, 5.7803, -1, -1],
>                          Na=0.147,
>                          NaF=[1.02, 2.295, 4.73, 1.75, 1.99, -1, -1],
>                          NaCl=[3.14, 2.789,5.43, 2.35, 3.98, -1, -1],
>                          NaBr=[4.28, 2.954, 5.78, 2.64, 4.70, -1, -1],
>                          NaI=[6.35, 3.194, 6.62, 3.08, 6.45, -1, -1],
>                          K=0.81,
>                          KF=[1.20, 2.648, 5.11, 1.86, 3.10, -1, -1],
>                          KCl=[3.36, 3.116, 4.49, 2.20, 5.26, -1, -1 ],
>                          KBr=[4.54, 3.262, 4.52, 2.39, 6.17, -1, -1],
>                          KI=[6.66, 3.489, 4.66, 2.68, 7.75, -1, -1],
>                          Rb=1.35,
>                          RbF=[1.18, 2.789, 5.99, 1.94, 3.81, -1, -1 ],
>                          RbCl=[3.45,3.259, 4.53, 2.20, 6.40, -1, -1],
>                          RbBr=[4.58, 3.410, 4.51, 2.36, 7.69, -1, -1],
>                          RbI=[6.80, 3.628, 4.55, 2.61, 9.48, -1, -1] ,
>                          Cs=2.34,
>                          CsF=[1.27, 2.9827, 7.27, 2.17, 4.25, -1, -1],
> #  C=8/(3*sqrt(3)) for three cases bellow
>                          CsCl=[3.55, 3.5150, 6.68, 2.67, 4.5387, -1, -1],
>                          CsBr=[4.66, 3.6645, 6.38, 2.83, 5.4476, -1, -1],
>                          CsI=[6.81, 3.9018, 6.29, 3.09, 6.9396, -1, -1]]):
```

It can be observed that each index is a name. For index `Li`, `Na`, `K`, `Rb` and `Cs`, each entry is a number representing the value of the respective α_+. For example, `fcc_LT[Li]` is 0.030. For index related to salt's name each entry is a list representing the row of the corresponding table —Table 1 in this case. For example, `fcc_LT[LiF]` is the list `[0.89, 1.996, 8.50, 1.93, 1.43, -1, -1]`. Particular elements of a list can be extracted. In this example `fcc_LT[LiF][1]` refers to 0.89 `fcc_LT[LiF][2]` to 1.996 and so on. We consider that this data structure may make the system more efficient, and easier to understand and increment when having more data available.

5.2 Clausius-Mossoti Procedure

Clausius-Mossoti — equation 1 pg. 2 — is implemented by the following Maple V procedure

```
> # Clausius Mossoti equation and variables to be calculated
> # it does not evaluate, for that should use the function evalf
> # It verifies the number of arguments n and that all arguments are given
> # It verifies C (second argument)  have a float value
> # if n=4 then returns the equivalent to Ep=....
> # if n=5 then returns the equivalent to alfa_plus=....
> # if n=6 then returns the equivalent to alfa_minus=....
```

```
> Clau_Moss := proc(n,C,r,Ep,alfa_plus,alfa_minus)
> local Clausius_Mossoti;
> if nargs <> 6
>  then ERROR( 'Wrong number of arguments...there are', nargs,
>  'arguments instead of 6') fi:
> if n < 0 or n < 4 or n > 6 or not type(n,integer)
>         then ERROR('Argument to be evaluated', n, 'does not exist') fi:
> if not type(evalf(C),float)
>    then ERROR('Argument', C, 'should have a value') fi:
> Clausius_Mossoti := (Ep-1)/(Ep-2) = 4*Pi*(alfa_plus + alfa_minus)/
> (3*C*r^3):
> if n=4 then simplify(solve(Clausius_Mossoti, Ep))
>      elif n=5 then simplify(solve(Clausius_Mossoti,alfa_plus))
>          else simplify(solve(Clausius_Mossoti,alfa_minus)) fi:
> end:
```

See bellow how this procedure can be activated to express argument `n` as a function of the other arguments. The following call gives Ep=f(C,r,alfa_plus,alfa_minus) for C=2

```
> Clau_Moss(4,2,r,Ep,alfa_plus,alfa_minus);
```

$$\frac{3\,r^3 - 4\,\pi\ alfa\_plus - 4\,\pi\ alfa\_minus}{3\,r^3 - 2\,\pi\ alfa\_plus - 2\,\pi\ alfa\_minus}$$

now for `alfa_plus`

```
> Clau_Moss(5,2,r,Ep,alfa_plus,alfa_minus);
```

$$\frac{1}{2}\,\frac{3\,Ep\,r^3 - 3\,r^3 - 2\,\pi\,alfa\_minus\,Ep + 4\,\pi\,alfa\_minus}{(\,Ep - 2\,)\,\pi}$$

and `alfa_minus`

```
> Clau_Moss(6,2,r,Ep,alfa_plus,alfa_minus);
```

$$\frac{1}{2}\,\frac{3\,Ep\,r^3 - 3\,r^3 - 2\,\pi\,alfa\_plus\,Ep + 4\,\pi\,alfa\_plus}{(\,Ep - 2\,)\,\pi}$$

other calls are not legal. Legal calls can be evaluated. For example the last one can be evaluated as shown bellow

```
> evalf(Clau_Moss(6,2,r,Ep,alfa_plus,alfa_minus));
```

$$.1591549431(\,3.\,Ep\,r^3 - 3.\,r^3 - 6.283185308\,alfa\_plus\,Ep + 12.56637062\,alfa\_plus\,)\,/(\,Ep - 2.\,)$$

```
> evalf(Clau_Moss(6,2,r,Ep,3.55,alfa_minus));
```

$$.3819718633\,10^{-8}(.125000000\,10^{9}\,Ep\,r^3 - .125000000\,10^{9}\,r^3 - .929387827\,10^{9}\,Ep + .185877565\,4\,10^{10})\,/(\,Ep - 2.\,)$$

```
> evalf(Clau_Moss(6,2,r,2.75,3.55,alfa_minus));
```

$$1.114084601\,r^3 - 3.550000000$$

```
> evalf(Clau_Moss(6,2,6.44,2.75,3.55,alfa_minus));
```

$$294.0108384$$

5.3 Worksheets Description

The computational system implemented is interactive and consists of two independent worksheets

1. for fixed anion
2. for fixed cation

In both cases, a common table and variable named `fcc_TA` and `temperature` are used. This table is instanciated with table `fcc_LT` and `temperature` with `LT` if LT data is to be used or with `fcc_RT` and `RT` respectively if RT is to be used. Both worksheets use the Clausius-Mossoti procedure described in section 5.2.

Each worksheet consists of several regions. For each anion (or cation) treated a comment of the form

```
# BEGIN <element>
```

⋮

```
# END <element>
```

identifies the element treated. Inside BEGIN and END regions should be executed sequentially since they refer to the same anion (or cation). Care has been taken such that there is no computational side effect for different anions (or cations). Thus the program can easily be extended to treat other elements not considered here. Sections 5.3.1 and 5.3.2 shows respectively the code and execution for LT data of the first worksheet for fixed anion Cl and the second worksheet for fixed cation Li.

5.3.1 Program to Calculate α_- and $\partial\alpha_-/\partial r$ for a Fixed Anion in Different Salts

This program can be used for both RT and LT data. In what follows, the results of running the program at LT for anion Cl are shown. α_- is approximated by $a\frac{1}{r^2}+b$

```
> # BEGIN Cl
```

```
> # Calculation CM_alfa_minus at  temperature:= RT or LT for Ep_inf
> # for LiCl, NaCl, KCl, RbCl,CsCl
> # LiCl->calc[1], NaCl->calc[2], KCl->calc[3], RbCl->calc[4],CsCl->calc[5]
> CM_alfa_minus:=Clau_Moss(6,2,r0,Ep_inf,alfa_plus,alfa_minus):
> calc[1]:=evalf(subs(r0=fcc_TA[LiCl][2],Ep_inf=fcc_TA[LiCl][4],
> alfa_plus=fcc_TA[Li],CM_alfa_minus )):
> calc[2]:=evalf(subs(r0=fcc_TA[NaCl][2],Ep_inf=fcc_TA[NaCl][4],
> alfa_plus=fcc_TA[Na],CM_alfa_minus )):
> calc[3]:=evalf(subs(r0=fcc_TA[KCl][2],Ep_inf=fcc_TA[KCl][4],
> alfa_plus=fcc_TA[K],CM_alfa_minus )):
> calc[4]:=evalf(subs(r0=fcc_TA[RbCl][2],Ep_inf=fcc_TA[RbCl][4],
> alfa_plus=fcc_TA[Rb],CM_alfa_minus )):
> # CsCl has a different constant
> CM_alfa_minus:=Clau_Moss(6,(8/(3*sqrt(3))),r0,Ep_inf,alfa_plus,alfa_minus):
> calc[5]:=evalf(subs(r0=fcc_TA[CsCl][2],Ep_inf=fcc_TA[CsCl][4],
> alfa_plus=fcc_TA[Cs],CM_alfa_minus )):
```

```
> # Plot LiCl->calc[1],  NaCl->calc[2],  KCl->calc[3], RbCl->calc[4],
> # CsCl->cal[5]  calculated as f(r0)
> CM_alfa_minus_calc_Cl:=[[fcc_TA[LiCl][2],calc[1]], [fcc_TA[NaCl][2],
> calc[2]],[fcc_TA[KCl][2],calc[3]],[fcc_TA[RbCl][2],calc[4]],
> [fcc_TA[CsCl][2],calc[5]]]:
```

```
> Plot_calc_Cl:=plot(CM_alfa_minus_calc_Cl, style=POINT,
> color=red,symbol=CIRCLE):
```

```
> # Plot LiCl,  NaCl,  KCl, RbCl, CsCl  from experimental data as f(r0)
> CM_alfa_minus_data_Cl:= [[fcc_TA[LiCl][2],fcc_TA[LiCl][1]],
> [fcc_TA[NaCl][2],fcc_TA[NaCl][1]],[fcc_TA[KCl][2],fcc_TA[KCl][1]],
> [fcc_TA[RbCl][2],fcc_TA[RbCl][1]],[fcc_TA[CsCl][2],fcc_TA[CsCl][1]]]:
> Plot_data_Cl:=plot(CM_alfa_minus_data_Cl, style=POINT, color=blue,
> symbol=BOX):
> Plot_temp_Cl:=textplot([2.6,3.5,temperature]):
```

```
> display({Plot_calc_Cl,  Plot_data_Cl,Plot_temp_Cl}, labels=['r0','alfa-'],
> axes=BOXED,title='alfa-:= f(r0) for LiCl, NaCl,KCl,RbCl, CsCl from
> Experimental Data');
```

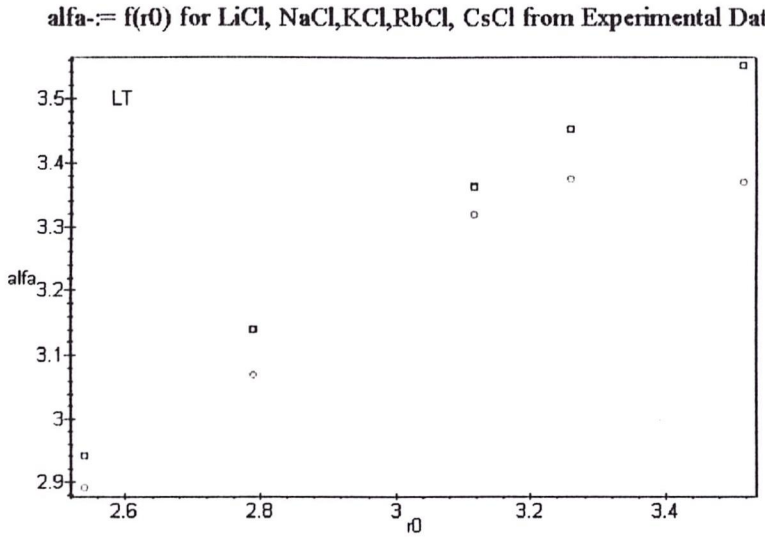

Figure 1: Command display({Plot_calc_Cl,Plot_data_Cl,Plot_temp_Cl})

```
> print('[r,Alfa-] and [r,Alfa- calculated] for LiCl,  NaCl,  KCl, RbCl at',
>  temperature ):
> print(array(CM_alfa_minus_data_Cl),array(CM_alfa_minus_calc_Cl)):
```

$[r, Alfa-]$ *and* $[r, Alfa-$ *calculated*$]$ *for* $LiCl$, $NaCl$, KCl, $RbCl$ *at*, LT

$$\begin{bmatrix} 2.539 & 2.94 \\ 2.789 & 3.14 \\ 3.116 & 3.36 \\ 3.259 & 3.45 \\ 3.5150 & 3.55 \end{bmatrix}, \begin{bmatrix} 2.539 & 2.890431382 \\ 2.789 & 3.067633311 \\ 3.116 & 3.317296688 \\ 3.259 & 3.372004791 \\ 3.5150 & 3.368146018 \end{bmatrix}$$

```
> # Calculate the same but in function of 1/(r0^2), invr2 is a function
> new_Cl:=map(invr2,CM_alfa_minus_calc_Cl):
> Plot_new_Cl:=plot(new_Cl, style=POINT, color=red,symbol=CROSS):
>
> # Now prepare the data for leastsquare, xvalue yvalue are pocedures
> alfa_Cl:=map(yvalue,new_Cl):
> r2_Cl:=map(xvalue,new_Cl):
> eq_fit_Cl:=fit[leastsquare[[r2,alfa],alfa=b*r2+a,{b,a}]]([r2_Cl,alfa_Cl]);
> corr_Cl:=describe[linearcorrelation](r2_Cl,alfa_Cl);
> # transform into a procedure
> eq_function_Cl:=unapply(rhs(eq_fit_Cl),r2):
> Plot_eq_Cl:=plot(eq_function_Cl(r2),r2=(r2_Cl[5]-0.001)..(r2_Cl[1]+0.001),
> color=green):
> Plot_eq_temp_Cl:=textplot([0.085,3.0,temperature]):
> alfa_predicted_Cl:=transform[apply[eq_function_Cl]](r2_Cl):
> # Find residuals
> residuals_Cl:=transform[multiapply[(x,y)->x-y]]([alfa_Cl,
> alfa_predicted_Cl]);
```

$$eq\_fit\_Cl := alfa = -7.117903997\, r2 + 4.002818524$$

$$corr\_Cl := -.9789202335$$

$$residuals\_Cl := [-.008240580, -.020113570, .047568051, .039353833, -.058567741]$$

```
> display({Plot_new_Cl,Plot_eq_Cl,Plot_eq_temp_Cl},labels=[`1/r**2`,`alfa-`],
> axes=BOXED,title=`alfa-=f(1/r**2) for LiCl,NaCl,KCl,RbCl, CsCl`);
```

```
> # Interpolation line  slope for Cl to calculate der(alfa-/r)
> pend_line_Cl:=eq_function_Cl(1)-eq_function_Cl(0):
> # Pick up original r's values of Cl
> r_original_Cl:=map(xvalue,CM_alfa_minus_data_Cl):
> Deriv_alfa_minus_r_Cl:=map(x->-2*pend_line_Cl*(x^(-3)),r_original_Cl);
```

$$Deriv\_alfa\_minus\_r\_Cl := [.8697491624, .6562005332, .4705326614, .4112719028, .3277978748]$$

```
> # Plot derivatives for Cl
> Plot_deriv_Cl:=plot(zip((x,y) ->[x,y],r_original_Cl,Deriv_alfa_minus_r_Cl),
> style=POINT,color=red,symbol=CROSS):
> display({Plot_deriv_Cl,textplot([3.4,0.8,temperature])},
> labels=[`r`,`d alfa-/dr`],axes=BOXED,title=`Derivative of
> alfa with respect to r for LiCl,NaCl,KCl,RbCl, CsCl`);
```

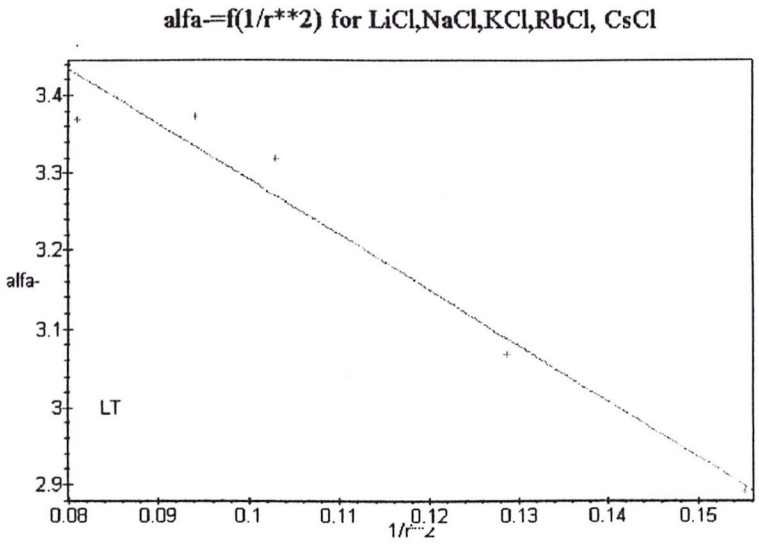

Figure 2: Command display({Plot_new_Cl,Plot_eq_Cl,Plot_eq_temp_Cl})

```
> # END Cl
```

5.3.2 Program to Calculate α_- and $\partial\alpha_-/\partial r$ for a Fixed Cation in Different Salts

This program can be used for both RT and LT data. In what follows, the results of running the program at LT for cation Li are shown. α_- is approximated by $ar^2 + br + c$

```
> # BEGIN Li
```

```
> # Calculation CM_alfa_minus at  temperature:= RT or LT for Ep_inf for
> # LiF, LICl, LiBr, LiI
> #  LiF->calc[1],  LiCl->calc[2],  LiBr->calc[3], LiI->calc[4]
> CM_alfa_minus:=Clau_Moss(6,2,r0,Ep_inf,alfa_plus,alfa_minus):
> calc[1]:=evalf(subs(r0=fcc_TA[LiF][2],Ep_inf=fcc_TA[LiF][4],
> alfa_plus=fcc_TA[Li],CM_alfa_minus )):
> calc[2]:=evalf(subs(r0=fcc_TA[LiCl][2],Ep_inf=fcc_TA[LiCl][4],
> alfa_plus=fcc_TA[Li],CM_alfa_minus )):
> calc[3]:=evalf(subs(r0=fcc_TA[LiBr][2],Ep_inf=fcc_TA[LiBr][4],
> alfa_plus=fcc_TA[Li],CM_alfa_minus )):
> calc[4]:=evalf(subs(r0=fcc_TA[LiI][2],Ep_inf=fcc_TA[LiI][4],
> alfa_plus=fcc_TA[Li],CM_alfa_minus )):
```

```
> # Plot LiF->calc[1],  LiCl->calc[2],  LiBr->calc[3], LiI->calc[4]
> # calculated as f(r0)
```

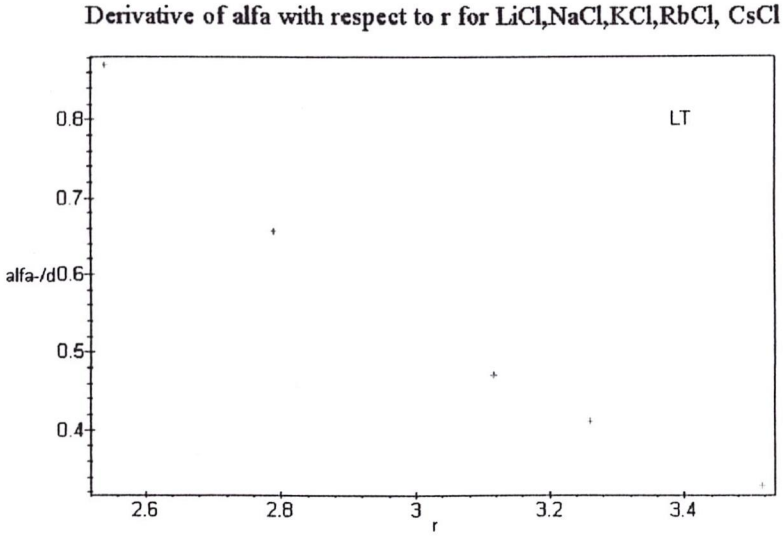

Figure 3: Command display({Plot_deriv_Cl})

```
> CM_alfa_minus_calc_Li:=[[fcc_TA[LiF][2],calc[1]], [fcc_TA[LiCl][2],
> calc[2]],[fcc_TA[LiBr][2],calc[3]],[fcc_TA[LiI][2],calc[4]]]:
> Plot_calc_Li:=plot(CM_alfa_minus_calc_Li, style=POINT, color=red,
> symbol=CIRCLE):
```

```
> # Plot LiF,  LiCl,  LiBr, LiI  from experimental data as f(r0)
> CM_alfa_minus_data_Li:= [[fcc_TA[LiF][2],fcc_TA[LiF][1]],
> [fcc_TA[LiCl][2],fcc_TA[LiCl][1]],[fcc_TA[LiBr][2],fcc_TA[LiBr][1]],
> [fcc_TA[LiI][2],fcc_TA[LiI][1]]]:
> Plot_data_Li:=plot(CM_alfa_minus_data_Li, style=POINT, color=blue,
> symbol=BOX):
> Plot_temp_Li:=textplot([2.03,6,temperature]):
```

```
> display({Plot_calc_Li,  Plot_data_Li,Plot_temp_Li}, labels=['r0','alfa-'],
> axes=BOXED,title='alfa-:= f(r0) for LiF, LiCl, LiBr, LiI  from
> Experimental Data');
```

```
> print('[r,Alfa-] and [r,Alfa- calculated] for LiF,  LiCl,  LiBr, LiI
> at', temperature ):
> print(array(CM_alfa_minus_data_Li),array(CM_alfa_minus_calc_Li)):
```

$[r, Alfa-]$ *and* $[r, Alfa-$ *calculated*$]$ *for* LiF, $LiCl$, $LiBr$, LiI *at*, LT

$$\begin{bmatrix} 1.996 & .89 \\ 2.539 & 2.94 \\ 2.713 & 4.13 \\ 2.951 & 6.19 \end{bmatrix}, \begin{bmatrix} 1.996 & .8684903049 \\ 2.539 & 2.890431382 \\ 2.713 & 4.024835490 \\ 2.951 & 5.990486168 \end{bmatrix}$$

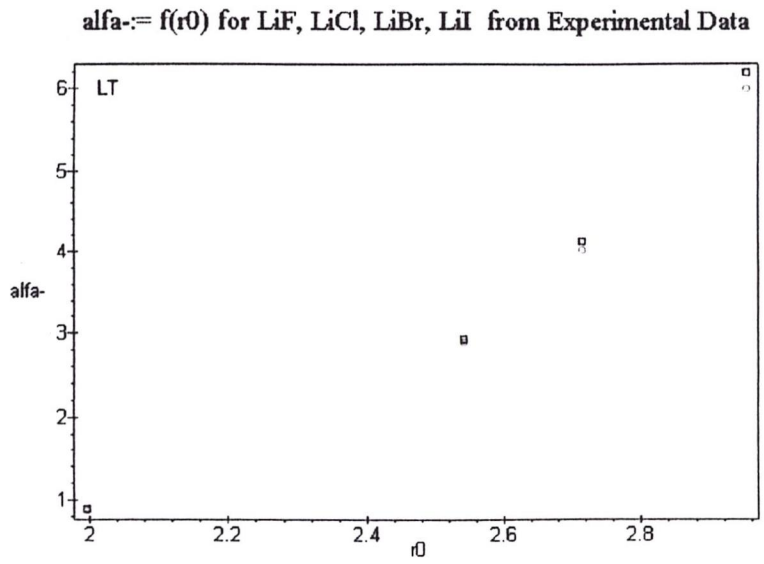

Figure 4: Command display({Plot_calc_Li,Plot_data_Li,Plot_temp_Li})

```
> # Using data in CM_alfa_minus_calc_Li prepare data for leastsquare
> alfa_Li:=map(yvalue,CM_alfa_minus_calc_Li):
> r_Li:=map(xvalue,CM_alfa_minus_calc_Li):
> eq_fit_Li:=fit[leastsquare[[r,alfa],alfa=a*r^2+b*r+c,{a,b,c}]]
> ([r_Li,alfa_Li]):
> # transform into a procedure both
> eq_function_Li:=unapply(rhs(eq_fit_Li),r);
> deriv_eq_function_Li:=unapply(value(Diff(eq_function_Li(r),r)),r):
> values_deriv_Li:=map(deriv_eq_function_Li,r_Li);
> Plot_eq_Li:=plot(eq_function_Li(r),r=r_Li[1]-0.05..r_Li[4]+0.05,
> color=green):
> alfa_predicted_Li:=transform[apply[eq_function_Li]](r_Li):
> # Find residuals
> residuals_Li:=transform[multiapply[(x,y)->x-y]]([alfa_Li,
> alfa_predicted_Li]);
```

$$eq\_function\_Li := r \rightarrow 3.996226163\,r^2 - 14.40845289\,r + 13.70720762$$

$$values\_deriv\_Li := [1.54448195, 5.88438357, 7.27507027, 9.17727392]$$

$$residuals\_Li := [-.0004743151, .004529772, -.005938610, .001883118]$$

```
> display({Plot_calc_Li,Plot_eq_Li,Plot_temp_Li},labels=['r','alfa-'],
> axes=BOXED,title='alfa-:= f(r) for LiF, LiCl, LiBr, LiI');
```

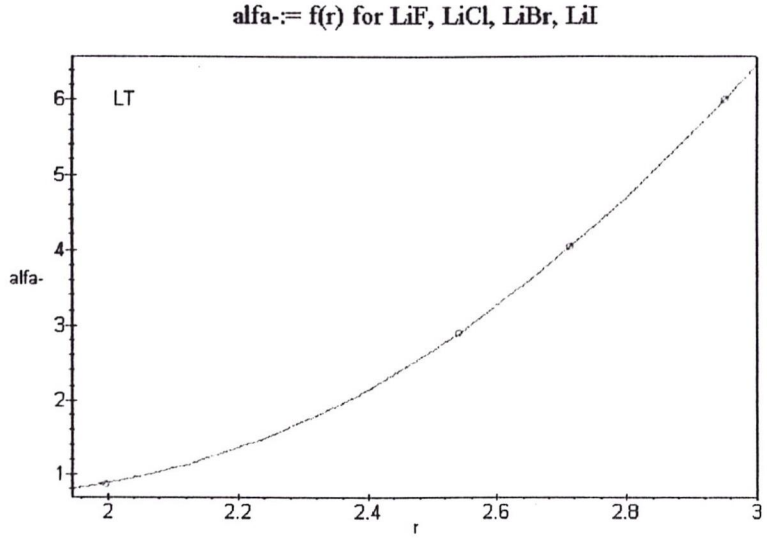

Figure 5: Command display({Plot_calc_Li,Plot_eq_Li,Plot_temp_Li})

```
> # Plot points Deriv Li
> Plot_deriv_Li:=plot(zip((x,y) -> [x,y], r_Li,values_deriv_Li),style=POINT,
> color=red,symbol=CROSS):
> display({Plot_deriv_Li,Plot_temp_Li},labels=['r','d alfa-/dr'],axes=BOXED,
> title='Derivative of alfa with respect to r for LiF, LiCl, LiBr, LiI');
```

```
> # END Li
```

6 Results

In the following tables are given: the experimental results obtained by executing the complete first worksheet, fixed anion, at LT — section 6.1 — and RT — section 6.2 — as well as the second worksheet, fixed cation, at LT — section 6.3 — and RT — section 6.4.

After executing the complete first worksheet, results will allow for a study of not only the anion polarizability for different salts but of the effect on the anion polarizability going from one salt to another by exchanging cations keeping the anion fixed. The corresponding results are shown in Table 3 – pg. 18 – and Table 6 – pg. 19 – for low and room temperature input data, respectively.

It is also possible to study a "generic" anion polarizability, in which case the anion is exchanged while keeping the cation fixed in order to obtain different compounds. Results are shown in Table 9 – pg. 20 – and Table 12 – pg. 21 – for low and room temperature input data, respectively.

Input data at room temperature were also used to show results, to allow for the case in which experimental data at low temperatures are not available.

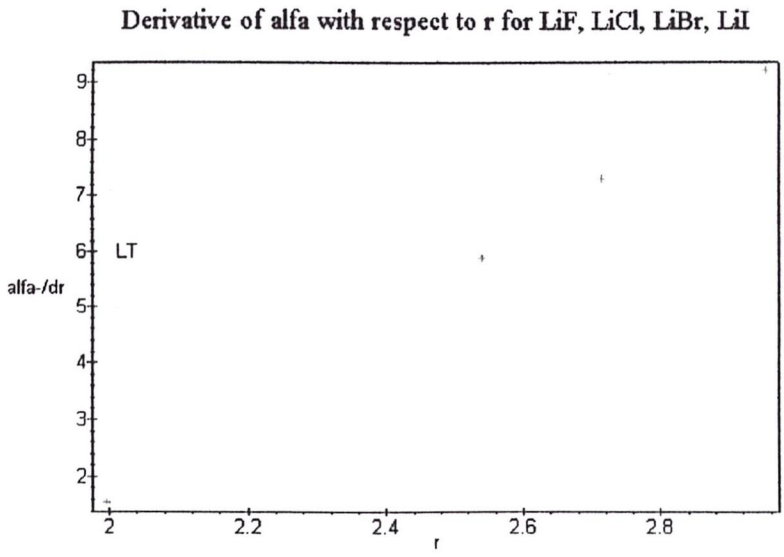

Figure 6: Command display({Plot_deriv_Li,Plot_temp_Li})

Functions shown in Table 4 – pg. 18 – and Table 10 – pg. 20 – (for low temperature input data) or Table 7 – pg. 19 – and Table 10 – pg. 20 – (for room temperature input data) are required to be able to evaluate the variation of α_- with respect to the interionic distance (Tables 5, 8, 11, 14) from which it is possible to know the material response under high pressures, as far as ionic polarizabilities are concerned.

6.1 Calculated α_- and $\partial\alpha_-/\partial r$ with LT Data for a Fixed Anion in Different Salts

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	.8684903049	1.007302010	1.165179072	1.121260689	1.214838834
Cl^-	2.890431382	3.067633311	3.317296688	3.372004791	3.368146019
Br^-	4.024835490	4.203094852	4.437392337	4.555503654	4.512793307
I^-	5.990486168	6.223092582	6.469603472	6.612843356	6.624849691

Table 3: α_- calculated – Units: $10^{-24}cm^3$

		Correlation
F^-	$-2.418947614\,r2 + 1.474270115$	-.9785435553
Cl^-	$-7.117903997\,r2 + 4.002818524$	-.9789202341
Br^-	$-8.975751010\,r2 + 5.253108315$	-.9700866766
I^-	$-13.91209056\,r2 + 7.599154067$	-.9846329387

Table 4: Interpolating Functions $r2 = \frac{1}{r^2}$

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	.6083798872	.4002288482	.2605568628	.2230031080	.1823172394
Cl^-	.8697491624	.6562005332	.4705326614	.4112719028	.3277978748
Br^-	.8989828588	.6964169286	.5171885746	.4527283128	.3648014990
I^-	1.082716520	.8539209258	.6551179262	.5826671804	.4684113162

Table 5: $\frac{\partial\alpha_-}{\partial r}$ calculated – Units: $10^{-16}cm^2$

6.2 Calculated α_- and $\partial\alpha_-/\partial r$ with RT Data for a Fixed Anion in Different Salts

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	.8919185414	1.028569443	1.204374187	1.184648302	1.400475338
Cl^-	2.955271260	3.142963299	3.364045831	3.487668175	3.470004897
Br^-	4.125743732	4.279422074	4.541754010	4.694130014	4.660603873
I^-	6.195374324	6.351428020	6.662056839	6.799325027	6.019750108

Table 6: α_- calculated – Units: $10^{-24}cm^3$

		Correlation
F^-	$-3.271250689\,r2 + 1.669840409$	-.9494902396
Cl^-	$-7.799331893\,r2 + 4.140348231$	-.9819660780
Br^-	$-10.16796728\,r2 + 5.462556037$	-.9716576330
I^-	$-3.715258075\,r2 + 6.723542842$	-.2033987631

Table 7: Interpolating Functiom

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	.8018313968	.5257714962	.3423759556	.2916472600	.2328751094
Cl^-	.9191564158	.6953468238	.5006351944	.4346087178	.3460217780
Br^-	.9780494490	.7629829800	.5659288030	.4960058436	.3975943764
I^-	.2751217570	.2190334216	.1684811015	.1501863069	.1290980188

Table 8: $\frac{\partial\alpha_-}{\partial r}$ calculated – Units: $10^{-16}cm^2$

6.3 Calculated α_- and $\partial\alpha_-/\partial r$ with LT Data for a Fixed Cation in Different Salts

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	.8684903049	1.007302010	1.165179072	1.121260689	1.214838834
Cl^-	2.890431382	3.067633311	3.317296688	3.372004791	3.368146019
Br^-	4.024835490	4.203094852	4.437392337	4.555503654	4.512793307
I^-	5.990486168	6.223092582	6.469603472	6.612843356	6.624849691

Table 9: α_- calculated – Units:$10^{-24}cm^3$

Li^+	$3.996229884\,r^2 - 14.40847091\,r + 13.70722891$
Na^+	$4.009129669\,r^2 - 16.20199308\,r + 17.07420286$
K^+	$4.495557216\,r^2 - 21.27226045\,r + 25.96984867$
Rb^+	$4.729705887\,r^2 - 23.80135825\,r + 30.71244564$
Cs^+	$4.657896924\,r^2 - 26.16853380\,r + 37.82641810$

Table 10: Interpolating Functions

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	1.54448195	2.19991210	2.53621057	2.58094119	1.61768451
Cl^-	5.88438357	6.16093221	6.74405212	7.02686472	6.57648158
Br^-	7.27507027	7.48394500	8.05675483	8.45523590	7.96919276
I^-	9.17727392	9.40832725	10.09773780	10.51738767	10.17983064

Table 11: $\frac{\partial\alpha_-}{\partial r}$ calculated – Units: $10^{-16}cm^2$

6.4 Calculated α_- and $\partial\alpha_-/\partial r$ with RT Data for a Fixed Cation in Different Salts

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	.8919185414	1.028569443	1.204374187	1.184648302	1.400475338
Cl^-	2.955271260	3.142963299	3.364045831	3.487668175	3.470004897
Br^-	4.125743732	4.279422074	4.541754010	4.694130014	4.660603872
I^-	6.195374324	6.351428020	6.662056839	6.799325027	6.019750108

Table 12: α_- calculated – Units: $10^{-24}cm^3$

Li^+	$3.884415387\,r^2 - 14.10406016\,r + 13.54325119$
Na^+	$3.775651245\,r^2 - 15.18284056\,r + 15.93639376$
K^+	$4.544324868\,r^2 - 21.84680066\,r + 27.12898220$
Rb^+	$4.712349849\,r^2 - 23.98105242\,r + 31.33421598$
Cs^+	$5.391091587\,r^2 - 31.56937873\,r + 47.54832914$

Table 13: Interpolating Functiom

	Li^+	Na^+	K^+	Rb^+	Cs^+
F^-	1.53615016	2.31579270	2.45170441	2.59942814	1.20845811
Cl^-	5.86028113	6.11409785	6.75245346	7.10726200	6.80441117
Br^-	7.25867060	7.37365511	8.14483460	8.51719708	8.45408520
I^-	9.20476260	9.26223586	10.26430772	10.61796264	10.06063049

Table 14: $\frac{\partial\alpha_-}{\partial r}$ calculated – Units: $10^{-16}cm^2$

7 Conclusions

A program is now available that allows for

1. Evaluation of α_-^∞ and $\partial\alpha_-^\infty/\partial r$ for anions in ionic crystals starting from experimental data of ε_∞ and assuming the Clausius-Mossotti treatment
2. The evaluation mentioned in 1. is given for
   (a) different salts with the same anion (i.e. fixed anion)
   (b) different salts with the same cation (i.e. fixed cation)
3. The analytical functions $\alpha_-^\infty(r)$ are given
4. Graphs are available showing
   (a) α_- *vs* r for fixed cation
   (b) α_- *vs* r for fixed anion
   (c) $\partial\alpha_-/\partial r$ *vs* r for fixed cation
   (d) $\partial\alpha_-/\partial r$ *vs* r for fixed anion

The methodology used in this program can now be extended to

- the study of static dielectric data and their polarizabilities at high pressures
- the study of crystals with more complex ions

References

[1] Abell, M.L.; Braselton, J.P. *Maple V by Examples.* AP Professional, Academic Press, 1994

[2] Fowler, P.W.; Madden, P.A. *The In-crystal Polarizability of the Fluoride Ion.* Molecular Physics, Vol. 49, Nº 4, pp. 913-923, 1983.

[3] Fowler, P.W.; Madden, P.A. *In-crystal Polarizabilities of Alkali and Halide Ions.* Physical Review B, Vol. 29, Nº 2, pp. 1035-1042, 1984.

[4] Fowler, P.W.; Madden, P.A. *In-crystal Polarizability of* O^{2-}. J. Phys. Chem. 89, pp 2581-2585, 1985.

[5] Fowler, P.W.; Pyper, N.C. *In Crystal Ionic Polarizabilities Derived by Combining Experimental and* ab initio *Results.* Proc. Roy. Soc. A398 , 377, pp. 377-391, 1985.

[6] Fowler, P.W. *A User's Guide to Polarisabilities and Dispersion Coefficients for Ions in Crystals.* Molecular Simulation, Vol. 4, pp. 313-330, 1990.

[7] Fowler, P.W.; Munn, R.W.; Tole, P. *Polarisability of the Oxide Ion in Crystalline BeO.* Chemical Physics Letters, Vol. 176, Nº 5, pp.439-445, 1991.

[8] Fowler, P.W.; Ding, F.; Munn, R.W. *Polarizabilities of Anions in Anisotropic Environments: the Fluoride Ion in the Perovskite Lattices* $NaMgF_3$, $KMgF_3$ *and* $KCaF_3$. Molecular Physics, Vol. 84, Nº 4, pp. 787-797, 1995.

[9] Mahan, G.D. *Ionic Polarization.* Ferroelectrics, Vol. 136, pp. 57-64, 1982.

[10] Monagan, M.B.; Geddes, K.O.; Heal, K.M.; Labahm, G.; Vorkoetter, S. *Maple V Programming Guide.* Springer-Verlag, NY, 1996.

[11] Monard, M.C.; Bruno, J.; Fracchia, R.M.; Batana, A. *Cálculo de Polarizabilidades y su Variación con la Presión.* Resumos XXI Congreso Argentino de Quimica, 1996.

[12] Shanke, J.; Dixit, S. *Dielectric Constants and their Pressure and Temperature Derivatives for Ionic Crystals.* Phys. Stat. Sol.(a) **123** , 17, (Review), pp. 17-50, 1991.

[13] Fracchia, R.M. *Estudio Teórico de Propiedades Anarmónica de Sólidos.* Tesis Doctor, Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires, Argentina, 1995.