CrysFML: A crystallographic library in modern Fortran

Juan Rodriguez-Carvajal
Laboratoire Léon Brillouin, (CEA-CNRS), CEA/ Saclay
FRANCE

Content of the talk

 ➤ Scientific Computing: Why Fortran?
 ➤ Crystallographic computing: CrysFML

Programming paradigms for scientific applications (I)

 ➤ Procedural, imperative structured programming (PP)
   Pascal, C, Fortran 77, ...
 ➤ Module-Oriented Programming (MOP)
   Fortran 95, ADA95, Modula-2, ... Fortran 2003
 ➤ Object oriented programming (OOP)
   C++, Java, Smalltalk, Eiffel, ADA95, ... Fortran 2003

Programming paradigms for scientific applications (II). Why Fortran?

Some reasons for developing in modern Fortran

 ➤ Simplicity and clarity of the syntax and the new facilities for global array manipulation. This is important for the common scientist that may write programs occasionally. This makes programming in Fortran more natural and problem solving oriented.
 ➤ Availability of many OOP features in modern Fortran: user-defined types, encapsulation, overload of procedures and functions. The lacking features (e.g. direct inheritance and class methods) are of less importance for scientific computing than those already available (all of them are available in Fortran 2003).

Programming paradigms for scientific applications (III). Why Fortran?

Some reasons for developing in modern Fortran

 ➤ The powerful implicit interface provided by encapsulating all functions and subroutines in modules, allowing to catch many errors at compile time, if one uses the intent attribute for procedure arguments. We may consider that Module Oriented Programming as an alternative/complement to OOP.
 ➤ Efficiency of the generated executable codes compared to C/C++ programs of similar complexity.
 ➤ Compatibility with legacy code and availability of a huge amount of free mathematical subroutines and functions. Re-usability of procedures written in Fortran 77 was already a reality.

Programming paradigms for scientific applications (IV). Why Fortran?

Some reasons for developing in modern Fortran

 ➤ The new standard (published in November 2004): Fortran 2003 contains all necessary features to perform pure OOP
 ➤ John Reid, WG5 Convener: The new features of Fortran 2003, PDF document available directly from the Internet:
 ➤ To our knowledge Fortran 2003 exist partially in NagF95, G95, in the new Lahey compiler for .NET, ...

Some reasons for developing in modern Fortran

 ➤ The powerful implicit interface provided by encapsulating all functions and subroutines in modules, allowing to catch many errors at compile time, if one uses the intent attribute for procedure arguments. We may consider that Module Oriented Programming as an alternative/complement to OOP.
 ➤ Efficiency of the generated executable codes compared to C/C++ programs of similar complexity.
 ➤ Compatibility with legacy code and availability of a huge amount of free mathematical subroutines and functions. Re-usability of procedures written in Fortran 77 was already a reality.
Existing Crystallographic Libraries (CCSL)

CCSL (Crystallographic Cambridge Subroutines Library)
J. Brown, J.C. Matthewmann
(W.I.F. David for powder diffraction)
⇒ The most complete set of procedures for crystallographic calculations. Well documented.
⇒ Written in Fortran 77 and with single crystal work in mind. Profuse use of commons. Difficult to adapt to modern programming techniques.

Existing Crystallographic Libraries (cctbx, Clipper)

Computational Crystallography Toolbox (cctbx)
R.W. Grosse-Kunstleve, P.D. Adams,…
Clipper
Kevin Cowtan
⇒ Written in C++ and handled using Python scripts.

CrysFML: a collection of F-modules for crystallography

Crystallographic Fortran Modules Library (CrysFML)
A simple toolbox for crystallographic computing programs
Commission on Crystallographic Computing, IUCr
There are many other modules that are not ready for distribution:
Magnetism, Cost_functions Instrument descriptions (Four circles + large PSD)
Refinement codes for molecular crystals

CrysFML info

In some cases the information about a particular procedure doesn’t appear, then goto the source code!
The reason: I didn’t obey my own rules for documentation!

Developers of
CrysFML/WinPLOTR/FullProf
Juan Rodríguez-Carvajal (LLB, France)
CrysFML, FullProf, Basireps, Simbo, Enermag, Polar3D,…
Javier González-Platas (ULL, Tenerife, Spain)
CrysFML, GUIs, GFourier, EdPCR
⇒ Contributors:
⇒ Thierry Roisnel (LCSIM, Rennes, France)
⇒ Carlos Frontera (ICMAB, Barcelona, Spain)
Polarized neutrons, Flipping ratio data handling
⇒ Marc Janoschek (PSI, Villigen, Switzerland)
⇒ Javier Gonzalez-Platas (ULL, Tenerife, Spain)
⇒ Carlos Frontera (ICMAB, Barcelona, Spain)
⇒ Magnetic properties of oxides
⇒ Aziz Daoud-Aladine (ISIS, UK)
⇒ Charge, spin and orbital ordering in manganites (Co-resp. SXD)
⇒ Laurent Chapon (ISIS, UK)
⇒ Thermo-electrics, multi-ferroics, … (Co-resp. GEM)
⇒ Marc Janoschek (PSI, Villigen, Switzerland)
Polarized neutrons instrumentation, Mu-PAD

Developers of …
We are not professional programmers!

⇒ Juan Rodriguez-Carvajal (LLB, CEA-CNRS, France)
Structural, electronic and magnetic properties of oxides and intermetallics. Modeling of magnetic structures
⇒ Javier Gonzalez-Platas (ULL, Tenerife, Spain)
Crystal structure determination of organic natural compounds. Teaching in Physics.
⇒ Thierry Roisnel (LCSIM, Rennes, France)
Crystal structure determination of cluster compounds
Single Crystal X-ray diffraction service (U. of Rennes)
⇒ Carlos Frontera (ICMAB, Barcelona, Spain)
⇒ Magnetic properties of oxides
⇒ Aziz Daoud-Aladine (ISIS, UK)
⇒ Charge, spin and orbital ordering in manganites (Co-resp. SXD)
⇒ Laurent Chapon (ISIS, UK)
⇒ Thermo-electrics, multi-ferroics, … (Co-resp. GEM)
⇒ Marc Janoschek (PSI, Villigen, Switzerland)
Polarized neutrons instrumentation, Mu-PAD
We have developed a set of Fortran 95 modules, Crystallographic Fortran Modules Library (CrysFML), that may be used (in the Fortran 95 sense) in crystallographic and diffraction computing programs.

- Modern array syntax and new features of Fortran 95 are used through the modules. In fact the whole library is written in F-language, a strict subset of Fortran 95 for which free compilers are available.
- We take advantage of all object oriented programming (OOP) techniques already available in Fortran: user-defined types, encapsulation, overload (polymorphism) of procedures and functions. The lacking features (e.g. inheritance and class methods) will be easily implemented as soon as Fortran 2003 compilers become available.
- Main programs using the adequate modules may perform more or less complicated calculations with only few lines of code.

**Scope of CrysFML**

All free F-compilers can be downloaded from the site:


See also:

http://www.fortran.com/fortran/Imagine1

**F-language (strict subset of Fortran 95)**

All implementations of the G95-compiler (based in gcc) can be downloaded from the G95 home page:

http://www.g95.org

Platforms: Linux, Windows, Mac OS, Solaris, OpenBSD, etc...

**Free Fortran 95 compiler G95: strong development**

At present there is no formal way of distributing CrysFML. I can send copies (of the most stable modules) by e-mail to everyone wishing to use it.

There are parts of the library that are not completely developed so be patient and comprehensive.

The library is distributed with a set of working examples so that the user can mimic in order to create his (her) own programs.

**Present status of CrysFML**

- The present CrysFML contains general and specific Mathematical modules (FFTs, geometrical calculations, optimizers, matrix operations). Procedures for reading files of many different formats, string utilities for handling free format, generation and reading of CIF files.
- Modules for generating space groups from their Hermann-Mauguin or Hall symbols. Generic space groups with non-conventional lattice centring vectors can also be built using user-defined generators.
- Reflection handling modules, including propagation vectors, may be used for generating reflections in selected regions of reciprocal space and for calculating structure factors.
- The documentation is written within the source code using special comment symbols. A document, in HTML format, containing the description of all modules and procedures can be generated using a Fortran program (get_doc).

**Programs using CrysFML (I)**

- **FullProf**: Crystal and magnetic structure refinement, powder/single crystals, polarised neutrons, constant wavelength, TOF, energy dispersive, multiple patterns.
- **FOURIER, GFOURIER** and **EdPCR**. These programs work on Windows and Linux and are already distributed from the LLB Web site.
- **BasIREPS**: Program for calculating basis functions of irreducible representations of space groups. This program is useful for determining magnetic structures and phonon symmetry analysis.
- **SIMBO**: Program for the analysis of the magnetic topology of an arbitrary crystal structure. Generates a formal description of the Fourier transform of the exchange interactions to be used by other programs.
**Programs using CrysFML (II)**

**ENERMAG:** Program to analyse the classical magnetic energy as a function of the exchange interactions and the point in the Brillouin Zone. This program can be used to generate theoretical magnetic phase diagrams in the J-space in order to get insight into the experimentally determined magnetic structures.

**SIMILAR:** Program to make conversion of settings for describing crystallographic structures. It determines automatically the splitting of Wyckoff positions on going from a space group to one of their subgroups. Calculate all the translationengleiche subgroups of a space group, co-set decompositions, etc.

**DATARED:** Program for data reduction of single crystal data. It handles twinning and incommensurate magnetic and crystal structures. Prepares files to be read by FullProf when using single crystals.

---

**Example of BasIreps output: *.bsr**

**PROPERATION VECTOR GROUP INFORMATION**

\[
\begin{align*}
\text{G}_k & \text{ has the following symmetry operators:} \\
1 \text{ SYMM}(1) &= x,y,z \\
2 \text{ SYMM}(3) &= x,-y,-z \\
3 \text{ SYMM}(4) &= -x+1/2,-y,z+1/2 \\
4 \text{ SYMM}(7) &= -x+1/2,y,-z+1/2 \\
& \quad \vdots
\end{align*}
\]

**Example of BasIreps output: *.bsr**

**Number of elements of \( G_k \):** 8

**Number of irreducible representations of \( G_k \):** 2

**Dimensions:** 2

**Symmetry elements of \( G_k \) and irreps:**

- **SYMm K(2):** -x+1/2,-y,z+1/2 : \( m ( x, x, z) \rightarrow h37 \)
- **Matrix of Irrep(2):**
  \[
  \begin{pmatrix}
  0 & i \\
  i & -1
  \end{pmatrix}
  \]

**GUI for FullProf: EdPCR**

GUI using Winteracter: [http://www.winteracter.com](http://www.winteracter.com)

**Example of BasIreps output: *.bsr**

**Number of elements of \( G_k \):** 8

**Number of irreducible representations of \( G_k \):** 2

**Dimensions:** 2

**Symmetry elements of \( G_k \) and irreps:**

- **SYMm K(2):** -x+1/2,-y,z+1/2 : \( m ( x, x, z) \rightarrow h37 \)
- **Matrix of Irrep(2):**
  \[
  \begin{pmatrix}
  0 & i \\
  i & -1
  \end{pmatrix}
  \]

**GUI for BasIreps**


Example of BasIreps output: *.bsr

**Number of elements of \( G_k \):** 8

**Number of irreducible representations of \( G_k \):** 2

**Dimensions:** 2

**Symmetry elements of \( G_k \) and irreps:**

- **SYMm K(2):** -x+1/2,-y,z+1/2 : \( m ( x, x, z) \rightarrow h37 \)
- **Matrix of Irrep(2):**
  \[
  \begin{pmatrix}
  0 & i \\
  i & -1
  \end{pmatrix}
  \]
Example of BasIreps output: *.bsr

Example of BasIreps output: *.bsr

Example using the HTML automatically generated documentation

Programming with CrysFML using it nearly as a black-box

A Fortran 95/2003 compiler is needed (G95 is free!)

Learn the main structure types and procedures existing in the modules of the library by reading the documentation

Write a main program, using the modules of the library, for a particular purpose

Example using the HTML automatically generated documentation

Example of BasIreps output: *.bsr

Example of BasIreps output: *.bsr

Example using the HTML automatically generated documentation

Programming with CrysFML using it nearly as a black-box

A Fortran 95/2003 compiler is needed (G95 is free!)

Learn the main structure types and procedures existing in the modules of the library by reading the documentation

Write a main program, using the modules of the library, for a particular purpose

Example using the HTML automatically generated documentation

Programming with CrysFML using it nearly as a black-box

A Fortran 95/2003 compiler is needed (G95 is free!)

Learn the main structure types and procedures existing in the modules of the library by reading the documentation

Write a main program, using the modules of the library, for a particular purpose

Example using the HTML automatically generated documentation
The procedure **Set_SpaceGroup**

Subroutine Set_SpaceGroup(Spacegen, Spacegroup, Gen, Ngen, Mode, Force_Hall)

!!!! Arguments !--!

character (len=k), intent(in) :: SpaceGen
Type (Space_Group_Type), intent(out) :: SpaceGroup
character (len*, dimension(n)), intent(in), optional :: gen
Integer, intent(in), optional :: ngen
character (len*), intent(in), optional :: Mode
character (len*), intent(in), optional :: Force_Hall

Header of the subroutine Set_SpaceGroup. Only two arguments are needed in the most simple cases.

The string `Spacegen` may contain the Hermann-Mauguin (H-M) symbol, the Hall symbol or, simply, the number of the space group.

The object `Spacegroup` is provided by a call to the subroutine.

The procedure **Write_SpaceGroup**

One can make a call to the subroutine as follows:

! Declarations omitted

Ngen=3
Gen(1)="y,-x, z"
Gen(2)="-x,-y,-z"
Gen(3)="x+1/2, y+1/2, -z"
Call Set_Spacegroup(Spacegen, Spacegroup, Gen, Ngen, "GEN")

On output the object `Spacegroup` of type `Space_Group_type` is filled with all possible information obtained from the list of the given generators.

Example using the HTML automatically generated documentation
Output of the small program: Get_SPG_info

Information on Space Group:
----------------------------------
=> Number of Space Group: 176
=> Hermann-Mauguin Symbol: P 63/M
=> Hall Symbol: -P 6c
=> Table Setting Choice:
  Setting Type: a'=a, b'=b, c'=c  -> Origin: (0,0,0)

=> LIST of Translationengleiche Subgroups:
  P 63                 P 6c            Index: [2]   ->  { x,y,z : …},  Acentric
  P 63/M              -P 6c            Index: [1]   ->  { x,y,z : …},  Centric
  P 3                  P 3             Index: [4]   ->  { x,y,z : …},  Acentric
  P -3                -P 3             Index: [2]   ->  { x,y,z },     Centric
  P 1 1 21             P 2c            Index: [6]   ->  { x,y,z : …},  Acentric
  P 1 1 21/M          -P 2c            Index: [3]   ->  { x,y,z : …},  Centric
  P -1                -P 1             Index: [6]   ->  { x,y,z },     Centric
  unknown              P -6c           Index: [2]   ->  { x,y,z : …},  Acentric
  unknown              P -2c           Index: [6]   ->  { x,y,z : …},  Acentric

Please enter a space group (H-M/Hall/number):

Another Example: Check_Group

=> Number of good reflections  :  94
=> Maximum intensity           :    3562.2319
=> Minimum (for observed)      :       3.5622
=> Number of Space Group tested:  85

=> LIST OF POSSIBLE SPACE GROUPS, a total of  24 groups are possible

<table>
<thead>
<tr>
<th>Number(IT)</th>
<th>Hermann-Mauguin Symbol</th>
<th>Hall Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>P 4</td>
<td>P 4</td>
</tr>
<tr>
<td>76</td>
<td>P 41</td>
<td>P 4w</td>
</tr>
<tr>
<td>77</td>
<td>P 42</td>
<td>P 4c</td>
</tr>
<tr>
<td>78</td>
<td>P 43</td>
<td>P 4cw</td>
</tr>
<tr>
<td>81</td>
<td>P -4</td>
<td>P -4</td>
</tr>
<tr>
<td>83</td>
<td>P 4/M</td>
<td>P 4</td>
</tr>
<tr>
<td>84</td>
<td>P 42/M</td>
<td>P 4c</td>
</tr>
<tr>
<td>89</td>
<td>P 4 2 2</td>
<td>P 4 2ab</td>
</tr>
<tr>
<td>90</td>
<td>P 4 21 2</td>
<td>P 4 2nw</td>
</tr>
<tr>
<td>91</td>
<td>P 41 2 2</td>
<td>P 41 2nw</td>
</tr>
<tr>
<td>92</td>
<td>P 41 2 1 2</td>
<td>P 41 2nw</td>
</tr>
<tr>
<td>93</td>
<td>P 4 2 1</td>
<td>P 4 2nw</td>
</tr>
<tr>
<td>94</td>
<td>P 4 1 2</td>
<td>P 4 1 nw</td>
</tr>
<tr>
<td>95</td>
<td>P 4 1 2 1</td>
<td>P 4 1 2nw</td>
</tr>
<tr>
<td>96</td>
<td>P 4 1 2 1 2</td>
<td>P 4 1 2nw</td>
</tr>
<tr>
<td>97</td>
<td>P 4 1 2 1 2 1</td>
<td>P 4 1 2nw</td>
</tr>
<tr>
<td>98</td>
<td>P 4 1 2 1 2 2</td>
<td>P 4 1 2nw</td>
</tr>
<tr>
<td>99</td>
<td>P 4 1 2 1 2 3 1</td>
<td>P 4 1 2nw</td>
</tr>
<tr>
<td>100</td>
<td>P 4 1 2 1 2 3 2</td>
<td>P 4 1 2nw</td>
</tr>
</tbody>
</table>

Check_Group output (1)

 PROGRAM CHECK_GROUP: attempt to select the possible space groups from an experimental Powder Diffraction Pattern

Author: J.Rodriguez-Carvajal (version 0.01, based on CrysFML)

Conditions:
Input hkl-file : testga1.hkl
Crystal System : Tetragonal
Check centred cells?: Y
Maximum angle   :  20.0000
Number of FWHMs :   2.0000
Threshold in %  :   0.1000

=> list of read reflections:

Another small program: test_subgroup

Output of the small program: test_subgroup

=> Another small program: test_subgroup

Output of the small program: Get_SPG_info

Another small program: test_subgroup

Output of the small program: Get_SPG_info

Another Example: Check_Group

=> Another small program: test_subgroup

Output of the small program: Get_SPG_info

Another Example: Check_Group

=> Another small program: test_subgroup

Output of the small program: Get_SPG_info

Another Example: Check_Group

=> Another small program: test_subgroup

Output of the small program: Get_SPG_info

Another Example: Check_Group

=> Another small program: test_subgroup
Program Calc_structure_factors

use crystallographic_symmetry, only: space_group_type, Write_SpaceG
use Atom_Module, only: Atoms_List_Type, Write_Atoms_List
use crystallographic_symmetry, only: space_group_type, Write_SpaceG
use Structure_Factor_Module, only: Structure_Factors, Write_Structure_Factors
use Atom_Module, only: Atoms_List_Type, Write_Atoms_List
use Structure_Factor_Module, only: Structure_Factors, Write_Structure_Factors

real :: stlmax  !Maximum Sin(Theta)/Lambda
integer :: MaxNumRef, Num, lun=1

type (Atoms_list_Type)   :: A

type (Reflection_Type),allocatable, dimension(:) :: hkl

type (Crystal_Cell_Type) :: Cell

type (space_group_type)  :: SpG

use Structure_Factor_Module,  only: Structure_Factors, Write_Structure_Factors
use IO_Formats,               only: Readn_set_Xtal_Structure,err_mess_form,err_form
use Reflections_Utilities,    only: Reflection_Type, Hkl_Uni, get_maxnumref
use crystal_types,            only: Crystal_Cell_Type, Write_Crystal_Cell
use crystallographic_symmetry,only: space_group_type, Write_SpaceG

Hkl_Uni:
call Hkl_Uni(Cell,SpG, true, 0.0,stlmax, &
        's',Num,hkl)
call Structure_Factors(A,SpG,Num,hkl,mode="NUC")
call Write_Structure_Factors(lun,Num,hkl,mode="NUC")

Generates unique reflections in a sinθ/λ range
(constructs, partially, the array of hkl objects)

Structure_Factors: Completes the construction of the array of hkl objects

Write_Struct_Factors : Writes the results in a file

Reads a CIF file and sets up the objects:
Cell : contains everything related to metrics
SpG : contains everything related to symmetry
A : contains everything concerned with atoms in the asymmetric unit
Installing and compiling CrysFML using G95 in Windows

G95 in Windows using MinGW
Copy the file g95-MinGW.exe in a temporary directory and double-click on it: select the installation folder and say “yes” to all questions! (e.g. c\G95, … warning! do not use “Program Files”)

- Create a directory called “CrysFML” (e.g. c:\CrysFML)
- Copy the file CrysFML_G95.zip in and extract all files respecting the directory structure
- Compile and build the library running the file “crysfml_g95.bat”

The content of the CrysFML folder and sub-folders

All CrysFML files start with the prefix “CFML_” and have extension .f95

Content of the “SimplePrograms” folder

Four main programs and make*.bat files, one *.hkl file coming from FullProf, a *.inf file and a sub-folder called “Sfac”

Content of the “Sfac” folder

Source code files:
- There are two modules: “observed_reflections” in file “observ.f90”
- “cost_functions” in file “cost_functions.f90”

Three main programs:
- “Calc_structure_factors” in “sfac_test.f90”
- “Optimizing_structure” in “Opt_Sfac.f90”
- “Optimizing_structure” in “Opt_restraints.f90”

Input files for CrysFML (CIF and CFL)

- Title: NiFePO5
  - Numbers: 3
  - alpha beta gamma
  - Space Group
  - Atom
  - Codes for refinement
  - Codes for refinement

- Title: Piket
  - Numbers: 3
  - alpha beta gamma
  - Space Group
  - Atom
  - Codes for refinement
  - Codes for refinement

Code:

- Name of the cost function
- Treatment of initial configuration
- Treatment of initial configuration