CrysFML: A crystallographic library in modern Fortran

Juan Rodríguez-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:
ftp://ftp.nag.co.uk/sc22wg5/N1551-N1600/N1579.pdf

➔To our knowledge **Fortran 2003** exist partially in NagF95, G95, in the new Lahey compiler for .NET, …

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.

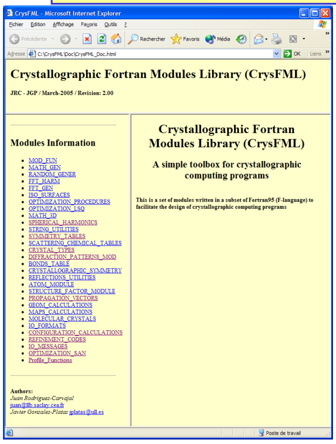
Existing Crystallographic Libraries (cctbx, Clipper)

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


CrysFML - Microsoft Internet Explorer

Adresse C:\CrysFML\Doc\CrysFML_Doc.html

Crystallographic Fortran Modules Library (CrysFML)

JRC - JGP / March-2005 / Revision: 2.00

Modules Information

- MOD_FUN
- MATH_GEN
- RANDOM_GENER
- FFT_HARM
- FFT_GEN
- ISO_SURFACES
- OPTIMIZATION_PROCEDURES
- OPTIMIZATION_LSQ
- MATH_3D
- SPHERICAL_HARMONICS
- STRING_UTILITIES
- SYMMETRY_TABLES
- SCATTERING_CHEMICAL_TABLES
- CRYSTAL_TYPES
- DIFFRACTION_PATTERNS_MOD
- BONDS_TABLE
- CRYSTALLOGRAPHIC_SYMMETRY
- REFLECTIONS_UTILITIES
- ATOM_MODULE
- STRUCTURE_FACTOR_MODULE
- PROPAGATION_VECTORS
- GEOM_CALCULATIONS
- MAPS_CALCULATIONS
- MOLECULAR_CRYSTALS
- IO_FORMATS
- CONFIGURATION_CALCULATIONS
- REFINEMENT_CODES
- IO_MESSAGES
- OPTIMIZATION_SAN
- Profile_Functions

Authors:
Juan Rodriguez-Carvajal juan@llb.saclay.cea.fr
Javier Gonzalez-Platas jplatas@ull.es

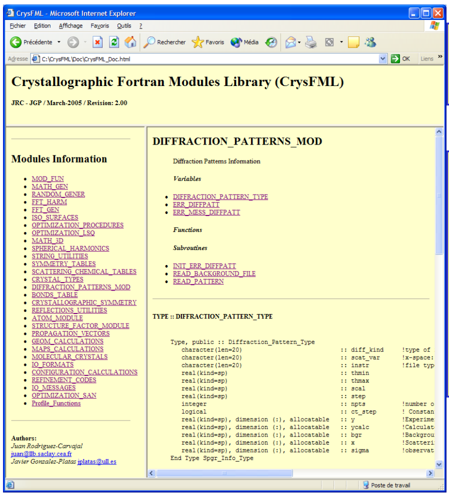
Crystallographic Fortran Modules Library (CrysFML)

A simple toolbox for crystallographic computing programs

This is a set of modules written in a subset of Fortran95 (F-language) to facilitate the design of crystallographic computing programs

"Crystallographic Fortran Modules Library (CrysFML). A simple toolbox for crystallographic computing programs"
Commission on Crystallographic Computing, IUCr Newsletter No.1, pp 50-58, January 2003.

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 - Microsoft Internet Explorer

Adresse C:\CrysFML\Doc\CrysFML_Doc.html

Crystallographic Fortran Modules Library (CrysFML)

JRC - JGP / March-2005 / Revision: 2.00

Modules Information

- MOD_FUN
- MATH_GEN
- RANDOM_GENER
- FFT_HARM
- FFT_GEN
- ISO_SURFACES
- OPTIMIZATION_PROCEDURES
- OPTIMIZATION_LSQ
- MATH_3D
- SPHERICAL_HARMONICS
- STRING_UTILITIES
- SYMMETRY_TABLES
- SCATTERING_CHEMICAL_TABLES
- CRYSTAL_TYPES
- DIFFRACTION_PATTERNS_MOD
- BONDS_TABLE
- CRYSTALLOGRAPHIC_SYMMETRY
- REFLECTIONS_UTILITIES
- ATOM_MODULE
- STRUCTURE_FACTOR_MODULE
- PROPAGATION_VECTORS
- GEOM_CALCULATIONS
- MAPS_CALCULATIONS
- MOLECULAR_CRYSTALS
- IO_FORMATS
- CONFIGURATION_CALCULATIONS
- REFINEMENT_CODES
- IO_MESSAGES
- OPTIMIZATION_SAN
- Profile_Functions

Authors:
Juan Rodriguez-Carvajal juan@llb.saclay.cea.fr
Javier Gonzalez-Platas jplatas@ull.es

DIFFRACTION_PATTERNS_MOD

Diffraction Patterns Information

Variables

- DIFFRACTION_PATTERN_TYPE
- ERR_DIFFPATT
- ERR_MESS_DIFFPATT

Functions

Subroutines

- INIT_ERR_DIFFPATT
- READ_BACKGROUND_FILE
- READ_PATTERN

TYPE :: DIFFRACTION_PATTERN_TYPE

```
Type, public :: Diffraction_Pattern_Type
   character(len=20)                              :: diff_kind     !type of
   character(len=20)                              :: scat_var      !x-space:
   character(len=20)                              :: instr         !file typ
   real(kind=sp)                                  :: thmin
   real(kind=sp)                                  :: thmax
   real(kind=sp)                                  :: scal
   real(kind=sp)                                  :: step
   integer                                        :: npts          !number o
   logical                                        :: ct_step       ! Constan
   real(kind=sp), dimension (:), allocatable      :: y             !Experime
   real(kind=sp), dimension (:), allocatable      :: ycalc         !Calculat
   real(kind=sp), dimension (:), allocatable      :: bgr           !Backgrou
   real(kind=sp), dimension (:), allocatable      :: x             !Scatteri
   real(kind=sp), dimension (:), allocatable      :: sigma         !observat
End Type Spgr_Info_Type
```

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)
WinPLOTR
⇒ Carlos Frontera (ICMAB, Barcelona, Spain)
Polarized neutrons, Flipping ratio data handling
⇒ Marc Janoschek (PSI, Villigen, Switzerland)
Polarized neutrons, 3D-Polarimetry
⇒ Laurent Chapon & Aziz Daoud-Aladine (ISIS, U.K.)
T.O.F. powder diffraction, WCrysFGL, Fp_Studio
Incommensurate crystal structures

Developers of …

We are not professional programmers!

⇒ Juan Rodríguez-Carvajal (LLB, CEA-CNRS, France)
Structural, electronic and magnetic properties of oxides and intermetallics. Modeling of magnetic structures
⇒ Javier González-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

Scope of CrysFML

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.

F-language (strict subset of Fortran 95)

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

ftp://ftp.swcp.com/~walt/pub/F

See also:

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

Free Fortran 95 compiler G95: strong development

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…

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).

Present status of CrysFML

⇒ 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.

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.

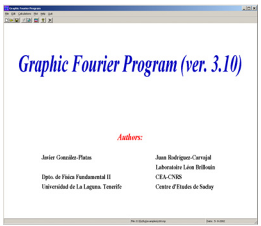
SIMBO: Program for the analysis of the magnetic topology of an arbitrary crystal structure. Generates a formal description of the Fourier transform of the exchange interactions to be used by other programs.

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.

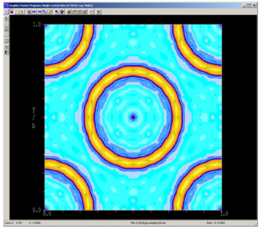


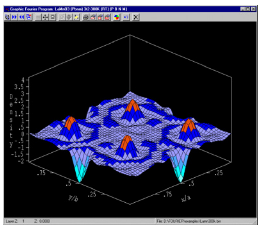
Graphic Fourier Program (ver. 3.10)

Authors:

Javier González-Platas
Dpto. de Fisica Fundamental II
Universidad de La Laguna. Tenerife

Juan Rodríguez-Carvajal
Laboratoire Léon Brillouin
CEA-CNRS
Centre d'Etudes de Saclay

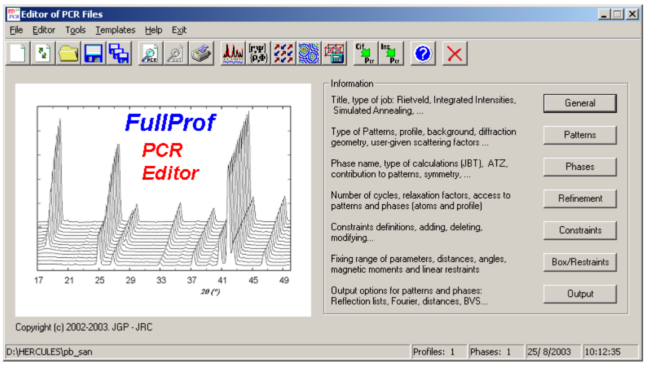




The programs Gfourier and Fourier are based in CrysFML

Graphic utilities: Winteracter
http://www.winteracter.com

A GUI for *FullProf*: EdPCR



Editor of PCR Files

Information

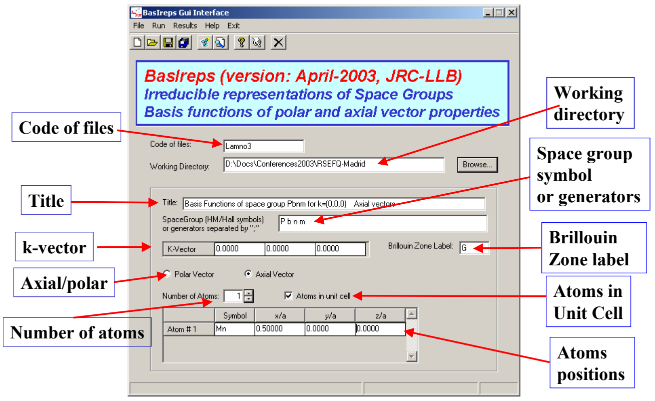
Title, type of job: Rietveld, Integrated Intensities, Simulated Annealing, ...	General
Type of Patterns, profile, background, diffraction geometry, user-given scattering factors ...	Patterns
Phase name, type of calculations (JBT), ATZ, contribution to patterns, symmetry, ...	Phases
Number of cycles, relaxation factors, access to patterns and phases (atoms and profile)	Refinement
Constraints definitions, adding, deleting, modifying...	Constraints
Fixing range of parameters, distances, angles, magnetic moments and linear restraints	Box/Restraints
Output options for patterns and phases: Reflection lists, Fourier, distances, BVS...	Output

Copyright (c) 2002-2003. JGP - JRC

D:\HERCULES\pb_san | Profiles: 1 | Phases: 1 | 25/ 8/2003 | 10:12:35

GUI using Winteracter: http://www.winteracter.com

GUI for *BasIreps*



BasIreps Gui Interface

BasIreps (version: April-2003, JRC-LLB)
Irreducible representations of Space Groups
Basis functions of polar and axial vector properties

Code of files: Lamno3
Working Directory: D:\Docs\Conferences2003\RSEFQ-Madrid
Title: Basis Functions of space group Pbnm for k=(0,0,0) Axial vectors
SpaceGroup (HM/Hall symbols) or generators separated by ";": P b n m
K-Vector: 0.0000 0.0000 0.0000
Brillouin Zone Label: G
○ Polar Vector ● Axial Vector
Number of Atoms: 1
☑ Atoms in unit cell

	Symbol	x/a	y/a	z/a
Atom # 1	Mn	0.50000	0.0000	0.0000

Code of files
Title
k-vector
Axial/polar
Number of atoms
Working directory
Space group symbol or generators
Brillouin Zone label
Atoms in Unit Cell
Atoms positions

ftp://ftp.cea.fr/pub/llb/divers/fullprof.2k

Example of *BasIreps* output: *.bsr

```
PROPAGATION VECTOR GROUP INFORMATION
====================================

=> The input propagation vector is: K=(  0.5000  0.5000  0.5000 )
=> K .. IS NOT .. equivalent to -K
=> The operators following the k-vectors constitute the co-set decomposition G[Gk]
   The list of equivalent k-vectors are also given on the right of operators.
=> The star of K is formed by the following   2 vectors:

       k_1  = (  0.5000  0.5000  0.5000 )     Op: (  1) x,y,z
                                              Op: (  3) x,-y,-z              -> (   0.5000 -0.5000 -0.5000 )
                                              Op: (  4) -x+1/2,-y,z+1/2      -> (  -0.5000 -0.5000  0.5000 )
                                              Op: (  7) -x+1/2,y,-z+1/2      -> (  -0.5000  0.5000 -0.5000 )
                                              Op: ( 10) y+3/4,-x+1/4,-z+3/4  -> (  -0.5000  0.5000 -0.5000 )
                                              Op: ( 13) -y+3/4,-x+1/4,z+3/4  -> (  -0.5000 -0.5000  0.5000 )
                                              Op: ( 14) -y+3/4,x+3/4,-z+1/4  -> (   0.5000 -0.5000 -0.5000 )
                                              Op: ( 16) y+3/4,x+3/4,z+1/4    -> (   0.5000  0.5000  0.5000 )

 Eqv. -K: k_2  = (  0.5000 -0.5000  0.5000 )  Op: (  2) -y+1/4,x+3/4,z+1/4
                                              Op: (  5) y+1/4,x+3/4,-z+1/4   -> (   0.5000  0.5000 -0.5000 )
                                              Op: (  6) y+1/4,-x+1/4,z+3/4   -> (  -0.5000  0.5000  0.5000 )
                                              Op: (  8) -y+1/4,-x+1/4,-z+3/4 -> (  -0.5000 -0.5000 -0.5000 )
                                              Op: (  9) -x,-y,-z             -> (  -0.5000 -0.5000 -0.5000 )
                                              Op: ( 11) -x,y,z               -> (  -0.5000  0.5000  0.5000 )
                                              Op: ( 12) x+1/2,y,-z+1/2       -> (   0.5000  0.5000 -0.5000 )
                                              Op: ( 15) x+1/2,-y,z+1/2       -> (   0.5000 -0.5000  0.5000 )

=> G_k has the following symmetry operators:
     1 SYMM(  1) = x,y,z
     2 SYMM(  3) = x,-y,-z
     3 SYMM(  4) = -x+1/2,-y,z+1/2
     4 SYMM(  7) = -x+1/2,y,-z+1/2
  . . . . . . . . .
```

Example of *BasIreps* output: *.bsr

```
 => Number of elements of G_k:   8
 => Number of irreducible representations of G_k:   2
 => Dimensions:  2 2

 => Symmetry elements of G_k and ireps:
    Symmetry elements reduced to the standard form (positive translations < 1)
    The matrices of IRreps have been multiplied by the appropriate phase factor
   . . . . . . . . . . . . . . . . . . . .
-> SYMM_K( 2):  -x+1/2,-y,z+1/2          :   2 ( 0, 0, z) ->  h4
    Phase factor for correcting input data:  0.0000
    Matrix of IRrep( 1):
                     i   0
                     0  -i
    Matrix of IRrep( 2):
                     i   0
                     0  -i
    . . . . . . . . . . . . . .
 -> SYMM_K( 8):  y+3/4,x+3/4,z+1/4        :   m ( x, x, z) -> h37
    Phase factor for correcting input data:  1.5000
    Matrix of IRrep( 1):
                     0   i
                    -1   0
    Matrix of IRrep( 2):
                     0  -i
                     1   0
```

Example of *BasIreps* output: *.bsr

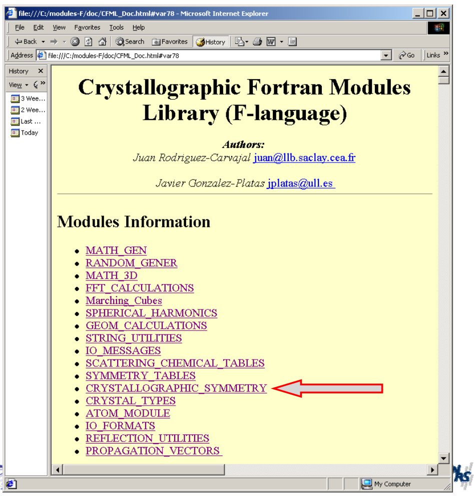
```
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
=> Basis functions of Representation IRrep( 1) of dimension  2 contained 3 times in GAMMA
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SYMM  x,y,z   -x+1/2,-y,z-1/2   y+3/4,-x+1/4,-z+3/4   -y+1/4,x+1/4,-z+3/4
Atoms:      Cu_1              Cu_2              Cu_3              Cu_4
 1:Re (    1    0    0) (    0    0    0) (    0   -1    0) (    0   -1    0)
   Im (    0    0    0) (   -1    0    0) (    0    0    0) (    0    0    0)
 2:Re (    0    1    0) (    0    0    0) (    1    0    0) (    1    0    0)
   Im (    0    0    0) (    0   -1    0) (    0    0    0) (    0    0    0)
 3:Re (    0    0    1) (    0    0    0) (    0    0   -1) (    0    0    1)
   Im (    0    0    0) (    0    0    1) (    0    0    0) (    0    0    0)
 4:Re (   -1    0    0) (    0    0    0) (    0    0    0) (    0    0    0)
   Im (    0    0    0) (   -1    0    0) (    0    1    0) (    0   -1    0)
 5:Re (    0    1    0) (    0    0    0) (    0    0    0) (    0    0    0)
   Im (    0    0    0) (    0    1    0) (    1    0    0) (   -1    0    0)
 6:Re (    0    0    1) (    0    0    0) (    0    0    0) (    0    0    0)
   Im (    0    0    0) (    0    0   -1) (    0    0   -1) (    0    0   -1)

----- LINEAR COMBINATIONS of Basis Functions: coefficients u,v,w,p,q ....
      General expressions of the Fourier coefficients Sk(i) i=1,2,...nat

      SYMM x,y,z                                      Atom: Cu_1      0.0000  0.0000  0.5000
      Sk(1):  (u-p,v+q,w+r)

      SYMM -x+1/2,-y,z-1/2                            Atom: Cu_2      0.5000  0.0000  0.0000
      Sk(2): i.(-u-p,-v+q,w-r)

      SYMM y+3/4,-x+1/4,-z+3/4                        Atom: Cu_3      0.7500  0.2500  0.2500
      Sk(3):  (v,-u,-w)+i.(q,p,-r)

      SYMM -y+1/4,x+1/4,-z+3/4                        Atom: Cu_4      0.2500  0.2500  0.2500
      Sk(4):  (v,-u,w)+i.(-q,-p,-r)
```

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


file:///C:/modules-F/doc/CFML_Doc.html#var78 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address file:///C:/modules-F/doc/CFML_Doc.html#var78
Go Links
History
View
3 Wee...
2 Wee...
Last ...
Today
My Computer

Crystallographic Fortran Modules Library (F-language)

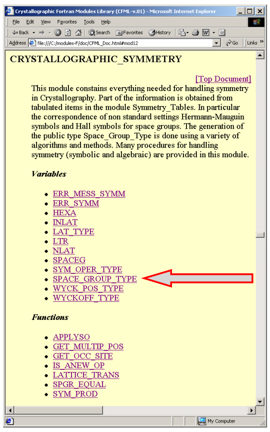
Authors:

Juan Rodriguez-Carvajal juan@llb.saclay.cea.fr

Javier Gonzalez-Platas jplatas@ull.es

Modules Information

- MATH_GEN
- RANDOM_GENER
- MATH_3D
- FFT_CALCULATIONS
- Marching_Cubes
- SPHERICAL_HARMONICS
- GEOM_CALCULATIONS
- STRING_UTILITIES
- IO_MESSAGES
- SCATTERING_CHEMICAL_TABLES
- SYMMETRY_TABLES
- CRYSTALLOGRAPHIC_SYMMETRY
- CRYSTAL_TYPES
- ATOM_MODULE
- IO_FORMATS
- REFLECTION_UTILITIES
- PROPAGATION_VECTORS


Crystallographic Fortran Modules Library (CFML-v.01) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address file:///C:/modules-F/doc/CFML_Doc.html#mod12
Go Links
My Computer

CRYSTALLOGRAPHIC_SYMMETRY

[Top Document]

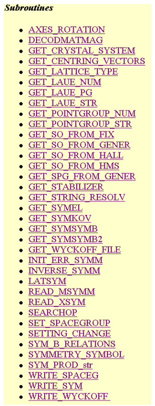
This module constains everything needed for handling symmetry in Crystallography. Part of the information is obtained from tabulated items in the module Symmetry_Tables. In particular the correspondence of non standard settings Hermann-Mauguin symbols and Hall symbols for space groups. The generation of the public type Space_Group_Type is done using a variety of algorithms and methods. Many procedures for handling symmetry (symbolic and algebraic) are provided in this module.

Variables

- ERR_MESS_SYMM
- ERR_SYMM
- HEXA
- INLAT
- LAT_TYPE
- LTR
- NLAT
- SPACEG
- SYM_OPER_TYPE
- SPACE_GROUP_TYPE
- WYCK_POS_TYPE
- WYCKOFF_TYPE

Functions

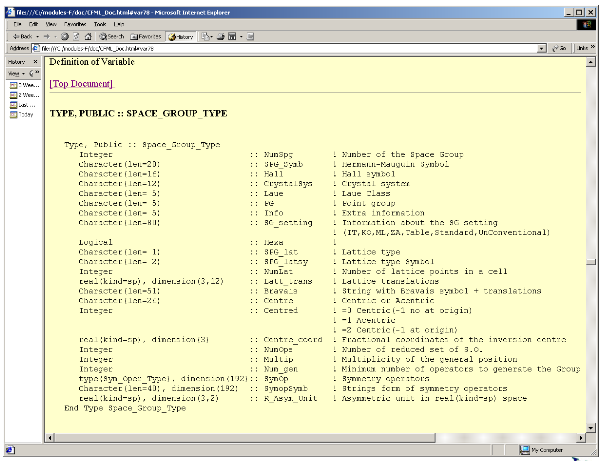
- APPLYSO
- GET_MULTIP_POS
- GET_OCC_SITE
- IS_ANEW_OP
- LATTICE_TRANS
- SPGR_EQUAL
- SYM_PROD



Subroutines

- AXES_ROTATION
- DECODMATMAG
- GET_CRYSTAL_SYSTEM
- GET_CENTRING_VECTORS
- GET_LATTICE_TYPE
- GET_LAUE_NUM
- GET_LAUE_PG
- GET_LAUE_STR
- GET_POINTGROUP_NUM
- GET_POINTGROUP_STR
- GET_SO_FROM_FIX
- GET_SO_FROM_GENER
- GET_SO_FROM_HALL
- GET_SO_FROM_HMS
- GET_SPG_FROM_GENER
- GET_STABILIZER
- GET_STRING_RESOLV
- GET_SYMEL
- GET_SYMKOV
- GET_SYMSYMB
- GET_SYMSYMB2
- GET_WYCKOFF_FILE
- INIT_ERR_SYMM
- INVERSE_SYMM
- LATSYM
- READ_MSYMM
- READ_XSYM
- SEARCHOP
- SET_SPACEGROUP
- SETTING_CHANGE
- SYM_B_RELATIONS
- SYMMETRY_SYMBOL
- SYM_PROD_str
- WRITE_SPACEG
- WRITE_SYM
- WRITE_WYCKOFF

Example using the HTML automatically generated documentation


file:///C:/modules-F/doc/CFML_Doc.html#var78 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address file:///C:/modules-F/doc/CFML_Doc.html#var78
Go Links
History
View
3 Wee...
2 Wee...
Last ...
Today
My Computer

Definition of Variable

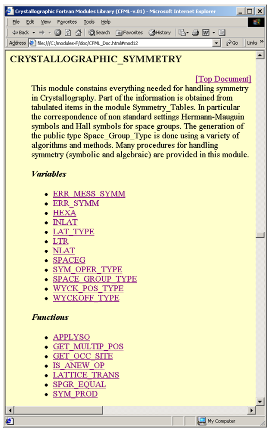
[Top Document]

TYPE, PUBLIC :: SPACE_GROUP_TYPE

```
   Type, Public :: Space_Group_Type
      Integer                                :: NumSpg        ! Number of the Space Group
      Character(len=20)                      :: SPG_Symb      ! Hermann-Mauguin Symbol
      Character(len=16)                      :: Hall          ! Hall symbol
      Character(len=12)                      :: CrystalSys    ! Crystal system
      Character(len= 5)                      :: Laue          ! Laue Class
      Character(len= 5)                      :: PG            ! Point group
      Character(len= 5)                      :: Info          ! Extra information
      Character(len=80)                      :: SG_setting    ! Information about the SG setting
                                                              ! (IT,KO,ML,ZA,Table,Standard,UnConventional)
      Logical                                :: Hexa          !
      Character(len= 1)                      :: SPG_lat       ! Lattice type
      Character(len= 2)                      :: SPG_latsy     ! Lattice type Symbol
      Integer                                :: NumLat        ! Number of lattice points in a cell
      real(kind=sp), dimension(3,12)         :: Latt_trans    ! Lattice translations
      Character(len=51)                      :: Bravais       ! String with Bravais symbol + translations
      Character(len=26)                      :: Centre        ! Centric or Acentric
      Integer                                :: Centred       ! =0 Centric(-1 no at origin)
                                                              ! =1 Acentric
                                                              ! =2 Centric(-1 at origin)
      real(kind=sp), dimension(3)            :: Centre_coord  ! Fractional coordinates of the inversion centre
      Integer                                :: NumOps        ! Number of reduced set of S.O.
      Integer                                :: Multip        ! Multiplicity of the general position
      Integer                                :: Num_gen       ! Minimum number of operators to generate the Group
      type(Sym_Oper_Type), dimension(192):: SymOp         ! Symmetry operators
      Character(len=40), dimension(192)      :: SymopSymb     ! Strings form of symmetry operators
      real(kind=sp), dimension(3,2)          :: R_Asym_Unit   ! Asymmetric unit in real(kind=sp) space
   End Type Space_Group_Type
```

Space Group Type in CrysFML

```
Type :: Space_Group_Type
     Integer                              :: NumSpg       ! Number of the Space Group
     Character(len=20)                    :: SPG_Symb     ! Hermann-Mauguin Symbol
     Character(len=16)                    :: Hall         ! Hall symbol
     Character(len=12)                    :: CrystalSys   ! Crystal system
     Character(len= 5)                    :: Laue         ! Laue Class
     Character(len= 5)                    :: PG           ! Point group
     Character(len= 5)                    :: Info         ! Extra information
     Character(len=80)                    :: SG_setting   ! Information about the SG setting
                                                          ! (IT,KO,ML,ZA,Table,Standard,UnConventional)
     Logical                              :: Hexa         !
     Character(len= 1)                    :: SPG_lat      ! Lattice type
     Character(len= 2)                    :: SPG_latsy    ! Lattice type Symbol
     Integer                              :: NumLat       ! Number of lattice points in a cell
     real(kind=sp), dimension(3,12)       :: Latt_trans   ! Lattice translations
     Character(len=51)                    :: Bravais      ! String with Bravais symbol + translations
     Character(len=26)                    :: Centre       ! Centric or Acentric
     Integer                              :: Centred      ! =0 Centric(-1 no at origin)
                                                          ! =1 Acentric
                                                          ! =2 Centric(-1 at origin)
     real(kind=sp), dimension(3)          :: Centre_coord ! Fractional coordinates of the inversion centre
     Integer                              :: NumOps       ! Number of reduced set of S.O.
     Integer                              :: Multip       ! Multiplicity of the general position
     Integer                              :: Num_gen      ! Minimum numb. of oper. to generate the group
     type(Sym_Oper_Type), dimension(192):: SymOp        ! Symmetry operators
     Character(len=40), dimension(192)    :: SymopSymb    ! Strings form of symmetry operators
     type(wyckoff_type)                   :: Wyckoff      ! Wyckoff Information
     real(kind=sp), dimension(3,2)        :: R_Asym_Unit  ! Asymmetric unit in real space
   End Type Space_Group_Type
```



Crystallographic Fortran Modules Library (CFML-v.01) - Microsoft Internet Explorer

file:///C:/modules-F/doc/CFML_Doc.html#mod12

CRYSTALLOGRAPHIC_SYMMETRY

[Top Document]

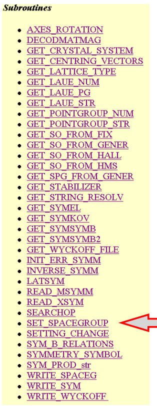
This module constains everything needed for handling symmetry in Crystallography. Part of the information is obtained from tabulated items in the module Symmetry_Tables. In particular the correspondence of non standard settings Hermann-Mauguin symbols and Hall symbols for space groups. The generation of the public type Space_Group_Type is done using a variety of algorithms and methods. Many procedures for handling symmetry (symbolic and algebraic) are provided in this module.

Variables

- ERR_MESS_SYMM
- ERR_SYMM
- HEXA
- INLAT
- LAT_TYPE
- LTR
- NLAT
- SPACEG
- SYM_OPER_TYPE
- SPACE_GROUP_TYPE
- WYCK_POS_TYPE
- WYCKOFF_TYPE

Functions

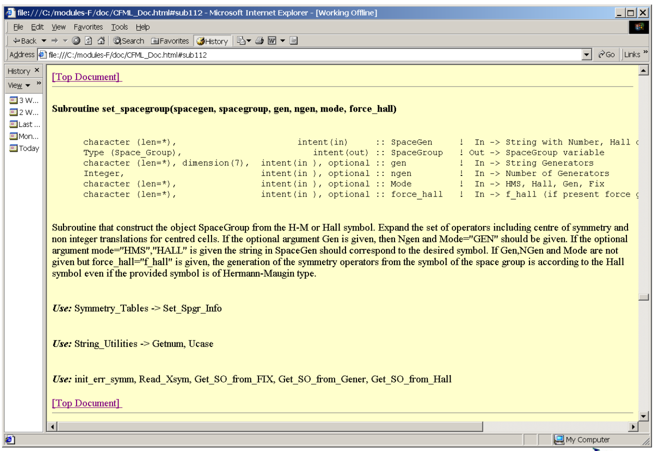
- APPLYSO
- GET_MULTIP_POS
- GET_OCC_SITE
- IS_ANEW_OP
- LATTICE_TRANS
- SPGR_EQUAL
- SYM_PROD



Subroutines

- AXES_ROTATION
- DECODMATMAG
- GET_CRYSTAL_SYSTEM
- GET_CENTRING_VECTORS
- GET_LATTICE_TYPE
- GET_LAUE_NUM
- GET_LAUE_PG
- GET_LAUE_STR
- GET_POINTGROUP_NUM
- GET_POINTGROUP_STR
- GET_SO_FROM_FIX
- GET_SO_FROM_GENER
- GET_SO_FROM_HALL
- GET_SO_FROM_HMS
- GET_SPG_FROM_GENER
- GET_STABILIZER
- GET_STRING_RESOLV
- GET_SYMEL
- GET_SYMKOV
- GET_SYMSYMB
- GET_SYMSYMB2
- GET_WYCKOFF_FILE
- INIT_ERR_SYMM
- INVERSE_SYMM
- LATSYM
- READ_MSYMM
- READ_XSYM
- SEARCHOP
- SET_SPACEGROUP
- SETTING_CHANGE
- SYM_B_RELATIONS
- SYMMETRY_SYMBOL
- SYM_PROD_str
- WRITE_SPACEG
- WRITE_SYM
- WRITE_WYCKOFF

Example using the HTML automatically generated documentation



file:///C:/modules-F/doc/CFML_Doc.html#sub112 - Microsoft Internet Explorer - [Working Offline]

file:///C:/modules-F/doc/CFML_Doc.html#sub112

[Top Document]

Subroutine set_spacegroup(spacegen, spacegroup, gen, ngen, mode, force_hall)

```
character (len=*),                      intent(in)     :: SpaceGen     !  In -> String with Number, Hall c
Type (Space_Group),                       intent(out) :: SpaceGroup   ! Out -> SpaceGroup variable
character (len=*), dimension(7),  intent(in ), optional :: gen          !  In -> String Generators
Integer,                          intent(in ), optional :: ngen         !  In -> Number of Generators
character (len=*),                intent(in ), optional :: Mode         !  In -> HMS, Hall, Gen, Fix
character (len=*),                intent(in ), optional :: force_hall   !  In -> f_hall (if present force g
```

Subroutine that construct the object SpaceGroup from the H-M or Hall symbol. Expand the set of operators including centre of symmetry and non integer translations for centred cells. If the optional argument Gen is given, then Ngen and Mode="GEN" should be given. If the optional argument mode="HMS","HALL" is given the string in SpaceGen should correspond to the desired symbol. If Gen,NGen and Mode are not given but force_hall="f_hall" is given, the generation of the symmetry operators from the symbol of the space group is according to the Hall symbol even if the provided symbol is of Hermann-Maugin type.

Use: Symmetry_Tables -> Set_Spgr_Info

Use: String_Utilities -> Getnum, Ucase

Use: init_err_symm, Read_Xsym, Get_SO_from_FIX, Get_SO_from_Gener, Get_SO_from_Hall

[Top Document]

The procedure *Set_SpaceGroup*

```
Subroutine Set_Spacegroup(Spacegen,Spacegroup,Gen,Ngen,Mode,Force_Hall)
   !----Arguments ----!
   character (len=*),              intent(in )           :: SpaceGen
   Type (Space_Group_Type),        intent(out)           :: SpaceGroup
   character (len=*), dimension(:), 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 *Set_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.

The procedure *Write_SpaceGroup*

```
!------------------------------------------
!  Example of simple program using CrysFML
!------------------------------------------
  Program Get_SPG_info
    use crystallographic_symmetry, only:  &
    space_group_type,set_spacegroup, write_spacegroup
    character(len=20)        :: spg_symb
    type(space_group_type)  :: SPG

    do
      write(unit=*,fmt="(a)",advance="no")  &
      " => Please enter a space group (H-M/Hall/number): "
      read(unit=*,fmt="(a)") spg_symb
      if(len_trim(spg_symb) == 0) exit
      call set_spacegroup(spg_symb,SPG)
      call write_spacegroup(SPG,full=.true.)
    end do
    stop
  End Program Get_SPG_info
```

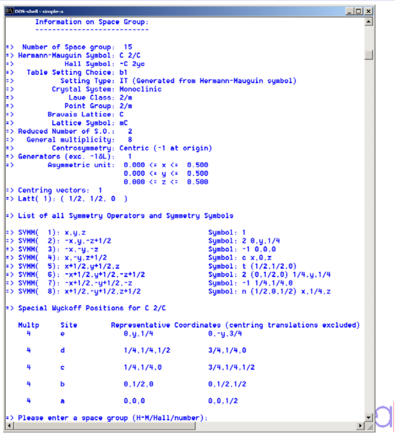
The procedure *Write_SpaceGroup*

The argument `full` in procedure `Write_SpaceGroup` means that all detailed information in asked to be output in the screen. One may change the instruction to write directly to an already opened file. For instance writing:

```
Call Write_SpaceGroup(SPG,iunit=3,full=.true.)
```

directs the output to the file connected with logical unit 3

Output of the small program: Get_SPG_info



```
       Information on Space Group:
       ---------------------------

=>  Number of Space group:  15
=> Hermann-Mauguin Symbol: C 2/C
=>            Hall Symbol: -C 2yc
=>   Table Setting Choice: b1
=>           Setting Type: IT (Generated from Hermann-Mauguin symbol)
=>         Crystal System: Monoclinic
=>             Laue Class: 2/m
=>            Point Group: 2/m
=>         Bravais Lattice: C
=>         Lattice Symbol: mC
=> Reduced Number of S.O.:   2
=>   General multiplicity:   8
=>         Centrosymmetry: Centric (-1 at origin)
=> Generators (exc. -1&L):   1
=>        Asymmetric unit:  0.000 <= x <=  0.500
                            0.000 <= y <=  0.500
                            0.000 <= z <=  0.500
=> Centring vectors:  1
=> Latt( 1): ( 1/2, 1/2, 0  )

=> List of all Symmetry Operators and Symmetry Symbols

=> SYMM(  1): x,y,z                           Symbol: 1
=> SYMM(  2): -x,y,-z+1/2                     Symbol: 2 0,y,1/4
=> SYMM(  3): -x,-y,-z                        Symbol: -1 0,0,0
=> SYMM(  4): x,-y,z+1/2                      Symbol: c x,0,z
=> SYMM(  5): x+1/2,y+1/2,z                   Symbol: t (1/2,1/2,0)
=> SYMM(  6): -x+1/2,y+1/2,-z+1/2             Symbol: 2 (0,1/2,0) 1/4,y,1/4
=> SYMM(  7): -x+1/2,-y+1/2,-z                Symbol: -1 1/4,1/4,0
=> SYMM(  8): x+1/2,-y+1/2,z+1/2              Symbol: n (1/2,0,1/2) x,1/4,z

=> Special Wyckoff Positions for C 2/C

   Multp    Site       Representative Coordinates (centring translations excluded)
     4       e            0,y,1/4              0,-y,3/4

     4       d            1/4,1/4,1/2          3/4,1/4,0

     4       c            1/4,1/4,0            3/4,1/4,1/2

     4       b            0,1/2,0              0,1/2,1/2

     4       a            0,0,0                0,0,1/2

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

Another small program: test_subgroup

```
Program test_subgroups
  use crystallographic_symmetry, only:  get_T_SubGroups,space_group_type, &
              set_spacegroup, write_spaceg, Lattice_trans,similar_transf_SG
  use math_3d, only : determ_a

  character(len=20)      :: spg_symb
  type(space_group_type) :: SpG,SpGn
  type(space_group_type), dimension(48) :: Subgroup
  real, dimension (3,3)  :: trans
  real, dimension (3  )  :: orig
  integer :: nsg, i, j, ng, l
  real    :: det

 do
   write(unit=*,fmt="(a)",advance="no")  " => Please enter a space group (H-M/Hall/number): "
   read(unit=*,fmt="(a)") spg_symb
   if(len_trim(spg_symb) == 0) exit
   call set_spacegroup(spg_symb,SPG)       !Constructing the space group SPG
   write(unit=*,fmt="(a)",advance="no")  " => Please enter a transformation matrix: "
   read(unit=*,fmt=*) (trans(i,:),i=1,3)
   det=determ_a(trans)
   det=abs(det)
   write(unit=*,fmt="(a)",advance="no")  " => Please enter the new origin: "
   read(unit=*,fmt=*) orig

   call similar_transf_SG(trans,orig,SpG,SpGn)  !Construct the subgroup of SPG that is compatible
   call write_spaceg(SPGn,full=.true.)          !with the transformation matrix and change of origin
                                                !give above

  !Determine all subgroups of the new space group
   call get_T_SubGroups(SPGn,SubGroup,nsg)

   write(unit=*,fmt="(/,a,/)") " => LIST of Translationengleische Subgroups: "
   do i=1,nsg
     j=SPGn%Multip/SubGroup(i)%multip
     ng=SubGroup(i)%numops
     write(unit=*,fmt="(4a,i2,30a)") " => ", SubGroup(i)%Spg_Symb, SubGroup(i)%hall,&
       " Index: [",j,"]   ->  { ", (trim(SubGroup(i)%SymopSymb(l))//" : ",l=1,ng-1),&
       trim(SubGroup(i)%SymopSymb(ng)),"  }     ", trim(SubGroup(i)%centre)
   end do
 end do
 stop
End Program test_subgroups
```

Output of the small program: test_subgroup

```
        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

```
Program Check_Group
  use crystallographic_symmetry, only: Space_Group_Type, set_spacegroup
  use reflections_utilities, only: Hkl_Absent
  use Symmetry_Tables, only: spgr_info, Set_Spgr_Info

  ....... ! Read reflections, apply criterion of "goodness" for checking,
          ! set indices i1,i2 for search in space group tables ...
  ....... ! omitted for simplicity
  call Set_Spgr_Info()
  m=0
  do_group: do i=i1,i2
    hms=adjustl(spgr_info(i)%HM)
    hall=spgr_info(i)%hall
    if( hms(1:1) /= "P" .and. .not. check_cent ) cycle do_group ! Skip centred groups
    call set_spacegroup(hall,Spacegroup,Force_Hall="y")
    do j=1,nhkl
       if(good(j) == 0) cycle  !Skip reflections that are not good (overlap) for checking
       absent=Hkl_Absent(hkl(:,j), Spacegroup)
       if(absent .and. intensity(j) > threshold) cycle do_group  !Group not allowed
    end do
    ! Passing here means that all reflections are allowed in the group -> Possible group!
    m=m+1
    num_group(m)=i
  end do  do_group
  write(unit=*,fmt=*) " => LIST OF POSSIBLE SPACE GROUPS, a total of ",m," groups are possible"
  write(unit=*,fmt=*) "     -----------------------------------------------------"
  write(unit=*,fmt=*) "     Number(IT)      Hermann-Mauguin Symbol     Hall Symbol"
  write(unit=*,fmt=*) "     -----------------------------------------------------"
  do i=1,m
    j=num_group(i)
    hms=adjustl(spgr_info(j)%HM)
    hall=spgr_info(j)%hall
    numg=spgr_info(j)%N
    write(unit=*,fmt="(i10,4a)")  numg,"                  ",hms,"          ",hall
  end do
  ...............
```

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:

      h   k   l   Intensity      Sigma      2theta      FWHM  Good?
      1   0   1      0.0000     0.0000      3.2518    0.0093    1
      1   1   0      3.4230     0.4030      3.6146    0.0113    1
      0   0   2      0.5280     0.2050      4.0212    0.0091    1
      1   1   1      1.8570     0.3130      4.1363    0.0111    1
      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      2   2   2   3562.2319    38.4840      8.2781    0.0138    1
      2   1   3      5.4550     0.3910      8.3152    0.0118    0
      3   1   1     23.2680     0.1620      8.3347    0.0124    0
      1   0   4      0.0000     0.0000      8.4448    0.0102    1
```

Check_Group output (2)

```
 => 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
    -----------------------------------------------------
    Number(IT)      Hermann-Mauguin Symbol     Hall Symbol
    -----------------------------------------------------
       75                  P 4                    P 4
       76                  P 41                   P 4w
       77                  P 42                   P 4c
       78                  P 43                   P 4cw
       81                  P -4                   P -4
       83                  P 4/M                 -P 4
       84                  P 42/M                -P 4c
       89                  P 4 2 2                P 4 2
       90                  P 4 21 2               P 4ab 2ab
       91                  P 41 2 2               P 4w 2c
       92                  P 41 21 2              P 4abw 2nw
   . . . . . . . . . . . . . . . . . . . . . . . . . . .
      113                  P -4 21 M              P -4 2ab
      114                  P -4 21 C              P -4 2n
      115                  P -4 M 2               P -4 -2
   . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

```
Program Calc_structure_factors
 use crystallographic_symmetry,only: space_group_type, Write_SpaceGroup
 use Atom_Module,               only: Atoms_List_Type, Write_Atoms_List
 use crystal_types,             only: Crystal_Cell_Type, Write_Crystal_Cell
 use Reflections_Utilities,     only: Reflection_Type, Hkl_Uni, get_maxnumref
 use IO_Formats,                only: Readn_set_Xtal_Structure,err_mess_form,err_form
 use Structure_Factor_Module,   only: Structure_Factors, Write_Structure_Factors
 type (space_group_type)   :: SpG
 type (Atoms_list_Type)    :: A
 type (Crystal_Cell_Type)  :: Cell
 type (Reflection_Type),allocatable, dimension(:) :: hkl
 character(len=256)        :: filcod  !Name of the input file
 real                      :: stlmax  !Maximum Sin(Theta)/Lambda
 integer                   :: MaxNumRef, Num, lun=1
    do
     write(unit=*,fmt="(a)") " => Code of the file xx.cif (give xx): "
     read(unit=*,fmt="(a)") filcod
     if(len_trim(filcod) == 0) exit
     write(unit=*,fmt="(a)") " => Maximum sinTheta/Lambda: "
     read(unit=*,fmt=*) stlmax
     open(unit=lun,file=trim(filcod)//".sfa", status="replace",action="write")

     call Readn_set_Xtal_Structure(trim(filcod)//".cif",Cell,SpG,A,Mode="CIF")

     If(err_form) then
       write(unit=*,fmt="(a)") trim(err_mess_form)
       exit
     else
       call Write_Crystal_Cell(Cell,lun)
       call Write_SpaceGroup(SpG,lun)
       call Write_Atoms_List(A,lun=lun)
       MaxNumRef = get_maxnumref(stlmax,Cell%CellVol,mult=SpG%Multip)
       if(allocated(hkl)) deallocate(hkl); allocate (hkl(MaxNumRef))

       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")
     end if
     close(unit=lun)
    end do
End Program Calc_structure_factors
```

```
Program Calc_structure_factors
 . . . . . . . . . .

call Readn_set_Xtal_Structure(trim(filcod)//".cif",&
                              Cell,SpG,A,Mode="CIF")

 . . . . . . . . . . .

End Program Calc_structure_factors
```

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

```
Program Calc_structure_factors
 use crystallographic_symmetry,only: space_group_type, Write_SpaceGroup
 use Atom_Module,               only: Atoms_List_Type, Write_Atoms_List
 use crystal_types,             only: Crystal_Cell_Type, Write_Crystal_Cell
 use Reflections_Utilities,     only: Reflection_Type, Hkl_Uni, get_maxnumref
 use IO_Formats,                only: Readn_set_Xtal_Structure,err_mess_form,err_form
 use Structure_Factor_Module,   only: Structure_Factors, Write_Structure_Factors
 type (space_group_type)   :: SpG
 type (Atoms_list_Type)    :: A
 type (Crystal_Cell_Type)  :: Cell
 type (Reflection_Type),allocatable, dimension(:) :: hkl
 character(len=256)        :: filcod  !Name of the input file
 real                      :: stlmax  !Maximum Sin(Theta)/Lambda
 integer                   :: MaxNumRef, Num, lun=1
    do
     write(unit=*,fmt="(a)") " => Code of the file xx.cif (give xx): "
     read(unit=*,fmt="(a)") filcod
     if(len_trim(filcod) == 0) exit
     write(unit=*,fmt="(a)") " => Maximum sinTheta/Lambda: "
     read(unit=*,fmt=*) stlmax
     open(unit=lun,file=trim(filcod)//".sfa", status="replace",action="write")

     call Readn_set_Xtal_Structure(trim(filcod)//".cif",Cell,SpG,A,Mode="CIF")

     If(err_form) then
       write(unit=*,fmt="(a)") trim(err_mess_form)
       exit
     else
       call Write_Crystal_Cell(Cell,lun)
       call Write_SpaceGroup(SpG,lun)
       call Write_Atoms_List(A,lun=lun)
       MaxNumRef = get_maxnumref(stlmax,Cell%CellVol,mult=SpG%Multip)
       if(allocated(hkl)) deallocate(hkl); allocate (hkl(MaxNumRef))

       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")
     end if
     close(unit=lun)
    end do
End Program Calc_structure_factors
```

```
Program Calc_structure_factors
 use crystallographic_symmetry,only: space_group_type, Write_SpaceG
 use Atom_Module,               only: Atoms_List_Type, Write_Atoms_List
 use crystal_types,             only: Crystal_Cell_Type, Write_Crystal_Cell
 use Reflections_Utilities,     only: Reflection_Type, Hkl_Uni, get_maxnumref
 use IO_Formats,                only: Readn_set_Xtal_Structure,err_mess_form,err_form
 use Structure_Factor_Module,   only: Structure_Factors, Write_Structure_Factors
 type (space_group_type)   :: SpG
 type (Atoms_list_Type)    :: A
 type (Crystal_Cell_Type)  :: Cell
 type (Reflection_Type),allocatable, dimension(:) :: hkl
 character(len=256)        :: filcod  !Name of the input file
 real                      :: stlmax  !Maximum Sin(Theta)/Lambda
 integer                   :: MaxNumRef, Num, lun=1
    do
     write(unit=*,fmt="(a)") " => Code of the file xx.cif (give xx): "
     read(unit=*,fmt="(a)") filcod
     if(len_trim(filcod) == 0) exit
     write(unit=*,fmt="(a)") " => Maximum sinTheta/Lambda: "
     read(unit=*,fmt=*) stlmax
     open(unit=lun,file=trim(filcod)//".sfa", status="replace",action="write")

     call Readn_set_Xtal_Structure(trim(filcod)//".cif",Cell,SpG,A,Mode="CIF")

If(err_form) then
       write(unit=*,fmt="(a)") trim(err_mess_form)
       exit
     else
       call Write_Crystal_Cell(Cell,lun)
       call Write_SpaceG(SpG,lun)
       call Write_Atoms_List(A,lun=lun)
       MaxNumRef = get_maxnumref(stlmax,Cell%CellVol,mult=SpG%Multip)
       if(allocated(hkl)) deallocate(hkl); allocate (hkl(MaxNumRef))

       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")
     end if
     close(unit=lun)
    end do
End Program Calc_structure_factors
```

```
Program Calc_structure_factors
 . . . . . . . . . . .

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")
```

`Hkl_Uni`: 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_Structure_Factors` : Writes the results in a file

```
 . . . . . . . . . . .
End Program Calc_structure_factors
```

```
Program Calc_structure_factors
 use crystallographic_symmetry,only: space_group_type, Write_SpaceG
 use Atom_Module,               only: Atoms_List_Type, Write_Atoms_List
 use crystal_types,             only: Crystal_Cell_Type, Write_Crystal_Cell
 use Reflections_Utilities,     only: Reflection_Type, Hkl_Uni, get_maxnumref
 use IO_Formats,                only: Readn_set_Xtal_Structure,err_mess_form,err_form
 use Structure_Factor_Module,   only: Structure_Factors, Write_Structure_Factors
 type (space_group_type)   :: SpG
 type (Atoms_list_Type)    :: A
 type (Crystal_Cell_Type)  :: Cell
 type (Reflection_Type),allocatable, dimension(:) :: hkl
 character(len=256)        :: filcod  !Name of the input file
 real                      :: stlmax  !Maximum Sin(Theta)/Lambda
 integer                   :: MaxNumRef, Num, lun=1
    do
     write(unit=*,fmt="(a)") " => Code of the file xx.cif (give xx): "
     read(unit=*,fmt="(a)") filcod
     if(len_trim(filcod) == 0) exit
     write(unit=*,fmt="(a)") " => Maximum sinTheta/Lambda: "
     read(unit=*,fmt=*) stlmax
     open(unit=lun,file=trim(filcod)//".sfa", status="replace",action="write")

     call Readn_set_Xtal_Structure(trim(filcod)//".cif",Cell,SpG,A,Mode="CIF")

     If(err_form) then
       write(unit=*,fmt="(a)") trim(err_mess_form)
       exit
     else
       call Write_Crystal_Cell(Cell,lun)
       call Write_SpaceG(SpG,lun)
       call Write_Atoms_List(A,lun=lun)
       MaxNumRef = get_maxnumref(stlmax,Cell%CellVol,mult=SpG%Multip)
       if(allocated(hkl)) deallocate(hkl); allocate (hkl(MaxNumRef))

       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")
     end if
     close(unit=lun)
    end do
End Program Calc_structure_factors
```

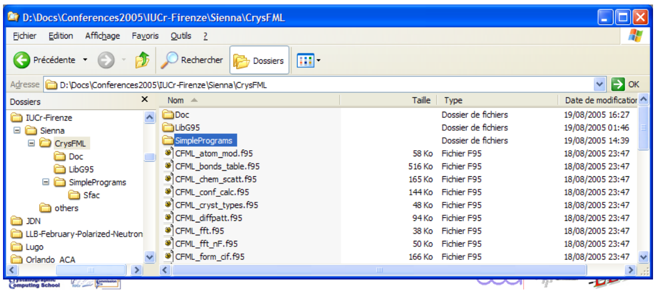
Installing and compiling CrysFML using G95 in Windows

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



D:\Docs\Conferences2005\IUCr-Firenze\Sienna\CrysFML

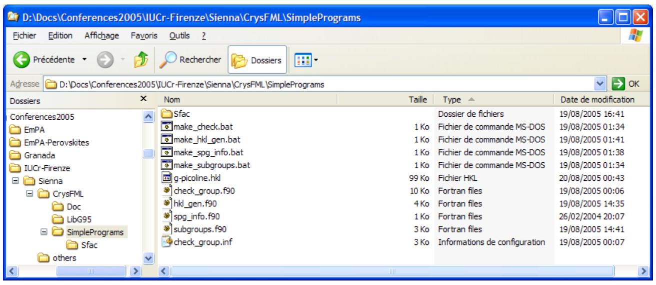
Adresse: D:\Docs\Conferences2005\IUCr-Firenze\Sienna\CrysFML

Dossiers
- IUCr-Firenze
  - Sienna
    - CrysFML
      - Doc
      - LibG95
      - SimplePrograms
        - Sfac
    - others
- JDN
- LLB-February-Polarized-Neutron
- Lugo
- Orlando_ACA

Nom	Taille	Type	Date de modificatior
Doc		Dossier de fichiers	19/08/2005 16:27
LibG95		Dossier de fichiers	19/08/2005 01:46
SimplePrograms		Dossier de fichiers	19/08/2005 14:39
CFML_atom_mod.f95	58 Ko	Fichier F95	18/08/2005 23:47
CFML_bonds_table.f95	516 Ko	Fichier F95	18/08/2005 23:47
CFML_chem_scatt.f95	165 Ko	Fichier F95	18/08/2005 23:47
CFML_conf_calc.f95	144 Ko	Fichier F95	18/08/2005 23:47
CFML_cryst_types.f95	48 Ko	Fichier F95	18/08/2005 23:47
CFML_diffpatt.f95	94 Ko	Fichier F95	18/08/2005 23:47
CFML_fft.f95	38 Ko	Fichier F95	18/08/2005 23:47
CFML_fft_nF.f95	50 Ko	Fichier F95	18/08/2005 23:47
CFML_form_cif.f95	166 Ko	Fichier F95	18/08/2005 23:47

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"



D:\Docs\Conferences2005\IUCr-Firenze\Sienna\CrysFML\SimplePrograms

Adresse: D:\Docs\Conferences2005\IUCr-Firenze\Sienna\CrysFML\SimplePrograms

Dossiers
- Conferences2005
- EmPA
- EmPA-Perovskites
- Granada
- IUCr-Firenze
  - Sienna
    - CrysFML
      - Doc
      - LibG95
      - SimplePrograms
        - Sfac
    - others

Nom	Taille	Type	Date de modification
Sfac		Dossier de fichiers	19/08/2005 16:41
make_check.bat	1 Ko	Fichier de commande MS-DOS	19/08/2005 01:34
make_hkl_gen.bat	1 Ko	Fichier de commande MS-DOS	19/08/2005 01:41
make_spg_info.bat	1 Ko	Fichier de commande MS-DOS	19/08/2005 01:38
make_subgroups.bat	1 Ko	Fichier de commande MS-DOS	19/08/2005 01:34
g-picoline.hkl	99 Ko	Fichier HKL	20/08/2005 00:43
check_group.f90	10 Ko	Fortran files	19/08/2005 00:06
hkl_gen.f90	4 Ko	Fortran files	19/08/2005 14:35
spg_info.f90	1 Ko	Fortran files	26/02/2004 20:07
subgroups.f90	3 Ko	Fortran files	19/08/2005 14:41
check_group.inf	3 Ko	Informations de configuration	19/08/2005 00:07

Content of the "Sfac" folder

Source code files:

There are two modules:
"observed_reflections" in file "observ.f90"
And
"cost_functions" in file "cost_functions.f90"

Three main programs:
"Calc_structure_factors" in "sfac_test.f90"
"Optimizing_structure" in "Optim_Sfac.f90"
"Optimizing_structure" in "Opt_restraints.f90"

Input files for CrysFML (CIF and CFL)

```
Title  NiFePO5
!        a        b        c     alpha    beta   gamma
Cell   7.1882   6.3924   7.4847  90.000  90.000  90.000
!     Space Group
Spgr  P n m a
!                x        y        z      B     occ   Spin Charge
Atom  Ni  NI  0.0000   0.0000  0.0000  0.74  0.5   2.0   2.0
Atom  Fe  FE  0.1443   0.2500  0.7074  0.63  0.5   5.0   3.0
Atom  P   P   0.3718   0.2500  0.1424  0.79  0.5   0.0   5.0
Atom  O1  O   0.3988   0.2500  0.64585 0.71  0.5   0.0  -2.0
Atom  O2  O   0.19415  0.2500  0.0253  0.70  0.5   0.0  -2.0
Atom  O3  O   0.0437   0.2500  0.4728  0.83  0.5   0.0  -2.0
Atom  O4  O   0.3678   0.0566  0.2633  0.77  1.0   0.0  -2.0
! Codes for refinement
Vary xyz 0  1  0  1
!Fix x_Fe y_O4
!Equal y_Fe y_P 1.00
HKL-OBS  mfe.hkl
MIN-DSPACING   1.5
OPTIMIZE  Fobs-Fcal 1.0
SIM_ANN
!       Name of the cost function
CostNam  FobsFcal
!         T_ini     anneal     num_temps
TemParM     8.0        0.95        90
!         Nalgor  Nconf nm_cycl   num_therm   accept
Algor_T     0       1      90          0          0.01
!         Value of Seed (if SeedVAL = 0, random seed)
SeedVAL     0
!         Treatment of initial configuration
InitCON   RAN
```