LEGACY CODE

FORTRAN, I AM LOOKING AT YOU!

AGE CONCERN

- Talk at Mieres Computing School 2001
- UK grant 2005-10 to addressing aging of small molecule software
  - All FORTRAN 77-ish
  - Extract the best algorithm
  - Make them available in an open and durable framework
  - CCP4 (and Phenix) as a model
- This was a different world! C.f. Simon's talk: vindication!

LEGACY CODE

- language agnostic
- practices make legacy
- e.g. FORTRAN does not sin, FORTRAN programmers do

FORTRAN + SCRIPING LANGUAGE

- FORTRAN purely for number crunching

```
SUBROUTINE SF(N, F, X, Y, Z, U11, U22, ...)
REAL X(N), Y(N), Z(N), U11(N), U22(N), ...
COMPLEX F(N)
DO I=1,N
! Here goes your structure factor computation
ENDO
```

- Then bridge with Python and business logic there

FORTRAN + SCRIPING LANGUAGE

- or at least **simple** file interface

```
PROGRAM SF
READ(*,'(A)') FN
OPEN(666, FILE=FN, ...)
REAL X(N), Y(N), Z(N), U11(N), U22(N), ...
READ(666, *) (X(I), Y(I), Z(I), U11(I), ..., I=1,N)
CALL SF(N, F, X, Y, Z, U11, U22, ...)
```

- and create that file with Python, Perl, etc.

BUT NO CRYSTALLOGRAPHER PROGRAM LIKE THAT

- COMMON /MEMORY/

```
COMMON /MEMORY/ A
...
SUBROUTINE F(I)
COMMON /MEMORY/ A
...
! Access A(I) to A(I+11) for something
...
SUBROUTINE G(I)
COMMON /MEMORY/ A
...
! Access A(I) to A(I+17) for something else
```

THE LIST GOES ON...

- common variant

```
COMMON /STORE/ data(N), params(N)
...
COMMON /STORE/ params(N), data(N)
```

BUT NO CRYSTALLOGRAPHER PROGRAM LIKE THAT

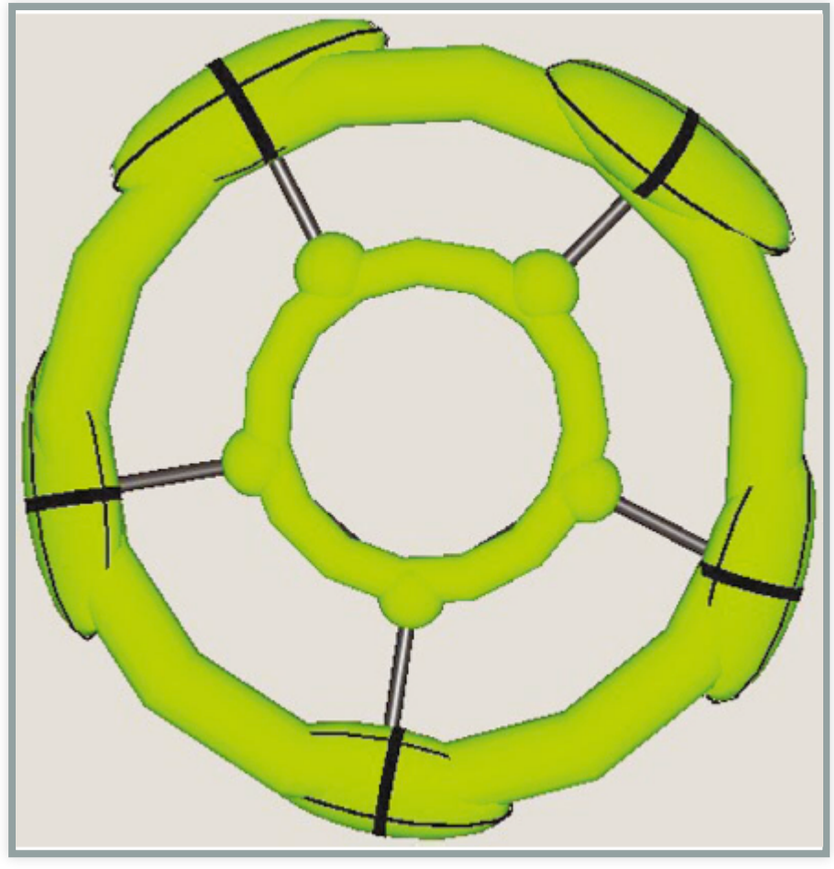
- Array of records

```
COMMON /STORE/ STRUCTURE
REAL STRUCTURE(absurdly big number)
! STRUCTURE
!
X, Y, Z, U11, ..., U23 (1st atom)
!
X, Y, Z, U11, ..., U23 (2nd atom)
!
etc
X = STRUCTURE(I)
Y = STRUCTURE(I+1)
...
U23 = STRUCTURE(I+8)
XX = STRUCTURE(I+9) ! next atom
```

- What if you need more parameters per atom?

EXAMPLE: ANHARMONIC TEMPERATURE FACTORS

- Go beyond U_{ij} with T_{ijk}: $e^{\sum_{ijk} T_{ijk} h_i h_j h_k}$
- Or atom delocalised over a torus!
  - implemented in Crystals
  - was a major struggle
- But is it better in modern framework?



CCTBX

- an atom:

```
class scatterer {
public:
fractional site;
symm_mat<3> u_star;
...
```

- Object-oriented to the rescue!

```
class enhanced_scatterer : scatterer {
public:
tensor<3> t;
...
```

CCTBX

- not so easy
- Dozens and dozens of functions use `scatterer`!
  1. make them template with the scatterer type generic
     - ouch!
  2. make `scatterer` a virtual class
     - `scatterer` object not passed by value: OK
- But arrays of `scatterer`'s (not of pointers to that)!

PRACTICAL ADVICES (AT LAST!)

- what shall I do with an old FORTRAN code?
- I could have told you about mining
  from my experience with SHELXL and Crystals
- But first try FABLE: Grosse-Kunstleve RW, Terwilliger TC, Sauter NK, Adams PD:
  Automatic Fortran to C++ conversion with FABLE
  Source Code for Biology and Medicine 2012, 7:5.

FABLE: BATTLE TESTED

- Used to convert Solve, Resolve (Tom Terwilinger)
  - Automated structure solution for MIR, SAD, and MAD
  - density modification, model building, etc
- Converted while the FORTRAN was still worked upon!
- Available in a CCTBX distro near you

FABLE: SOME FEATURES

- A runtime library for support, named FEM
- COMMON's are translated to C++ struct: so multiple instances of FORTRAN program in C++!
- FORTRAN multidimension array: `A(I, J)` remains `a(i,j)` using a special array-like class in FEM
- Handle *comman variants*

```
COMMON /STORE/ data(N), params(N)
...
COMMON /STORE/ params(N), data(N)
```