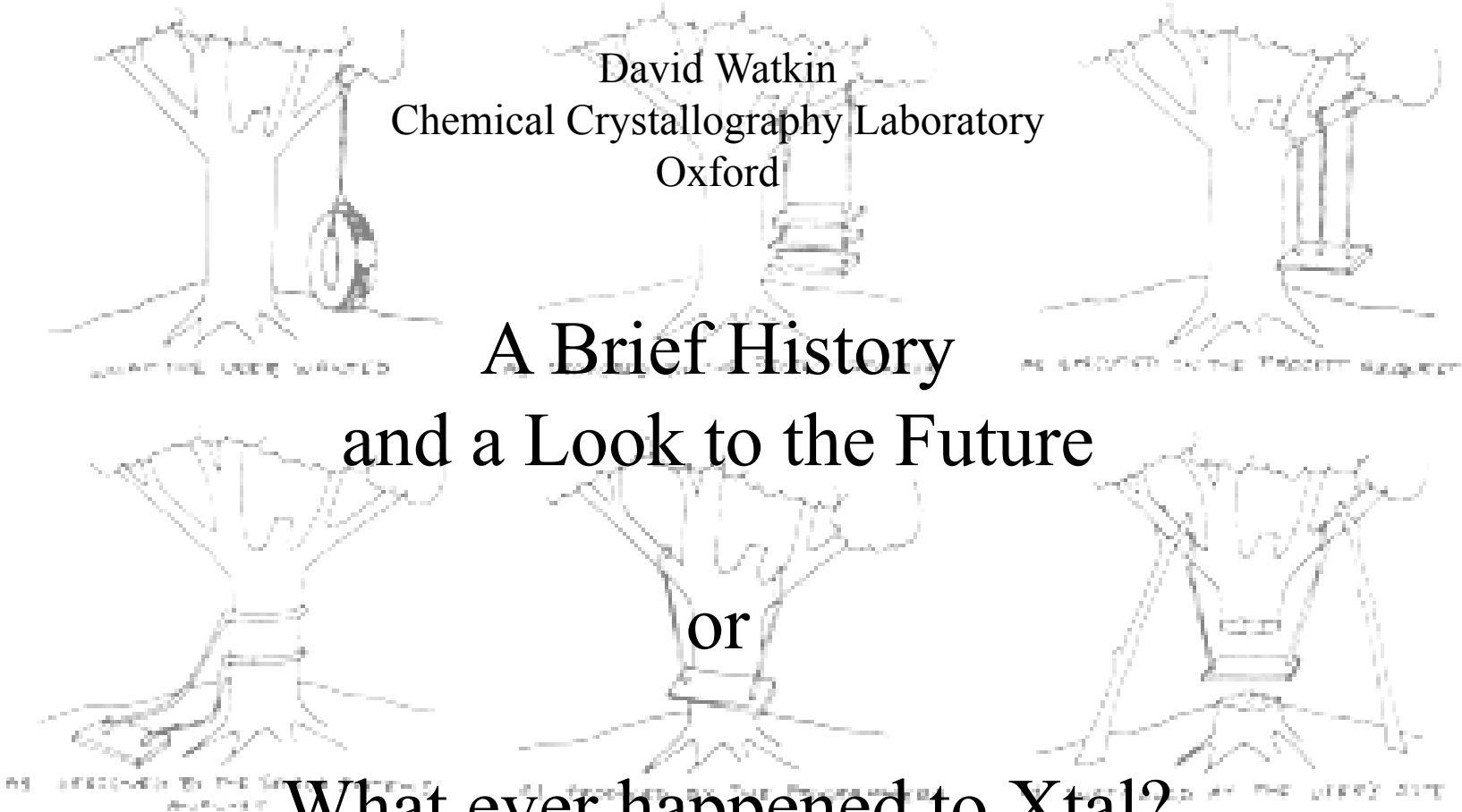
# Small Molecule Computing

David Watkin
Chemical Crystallography Laboratory
Oxford



A Brief History
and a Look to the Future

or

What ever happened to Xtal?

# Early Computing
# The 1950's

Most of the crystallographic 'computers' at this period were electro-mechanical analogue machines, using technology originally developed for military purposes.

Data were input via dials or other mechanical adjustments.

# Help with the Calculations

A.D. Booth, 1948

Mechanical, electrical, electro-mechanical and optical devices were built to help with computation of trigonometric functions and the summation of series.

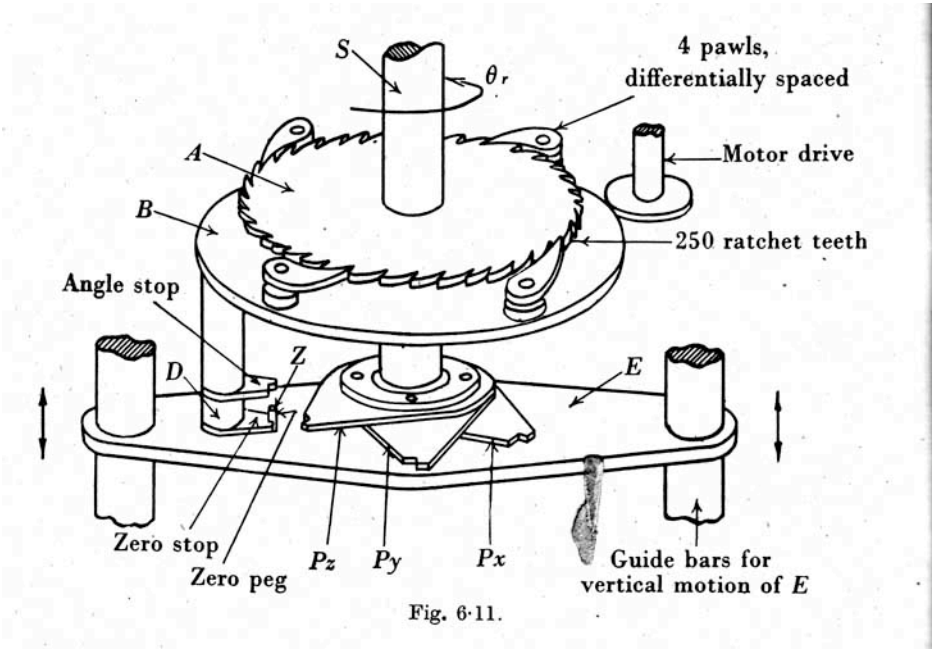


Fig. 6·11.

with $A$ whereas $C$ can rotate inside $A$. The induced current in the system $BC$ is thus

$$(B_n + C_n \cos\theta) \cos 2\pi\varpi t,$$

where $\theta$ is the angle through which the axis of $C$ has been turned with respect to that of $A$; $B_n$, $C_n$, coefficients of induction and $\varpi$ the frequency of the supply to $A$. By having a greater number of turns on $B$ than on $C$, $B_n > C_n$ so that no phase reversal can occur.

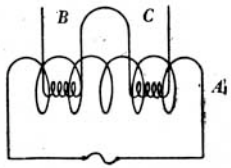
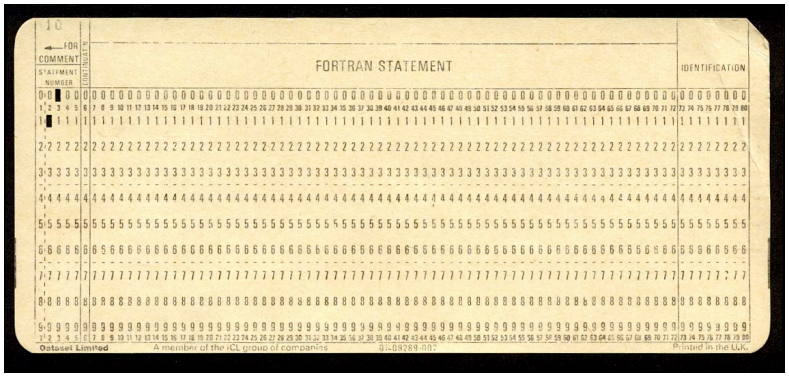


Fig. 6·15. Electromagnetic cosine resolver.

# Early Computing

Later, punched card accounting machines were co-opted for the computation of Fourier maps.

Data were input on Hollerith punched cards. The 80-column legacy is only slowly dying. Until recently, lines in a CIF were limited to 80 characters, including trailing blanks.



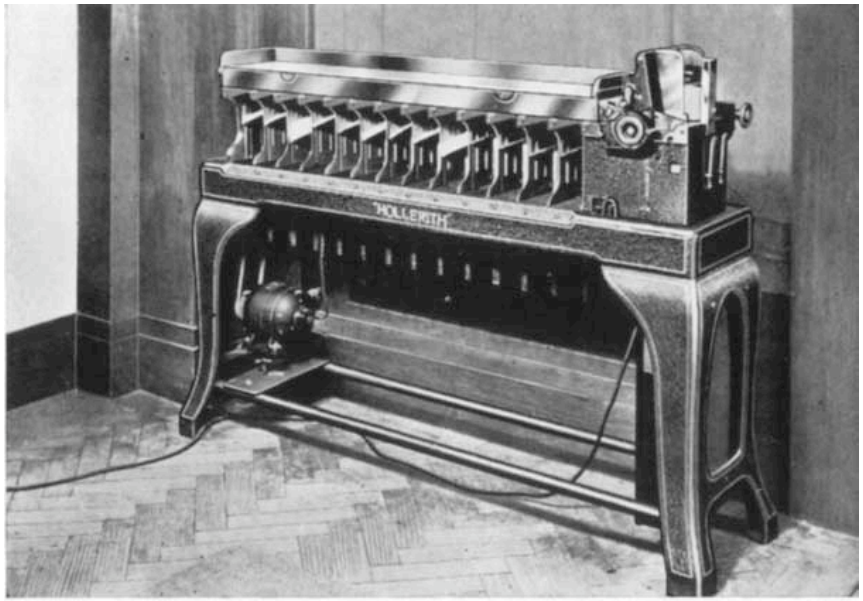
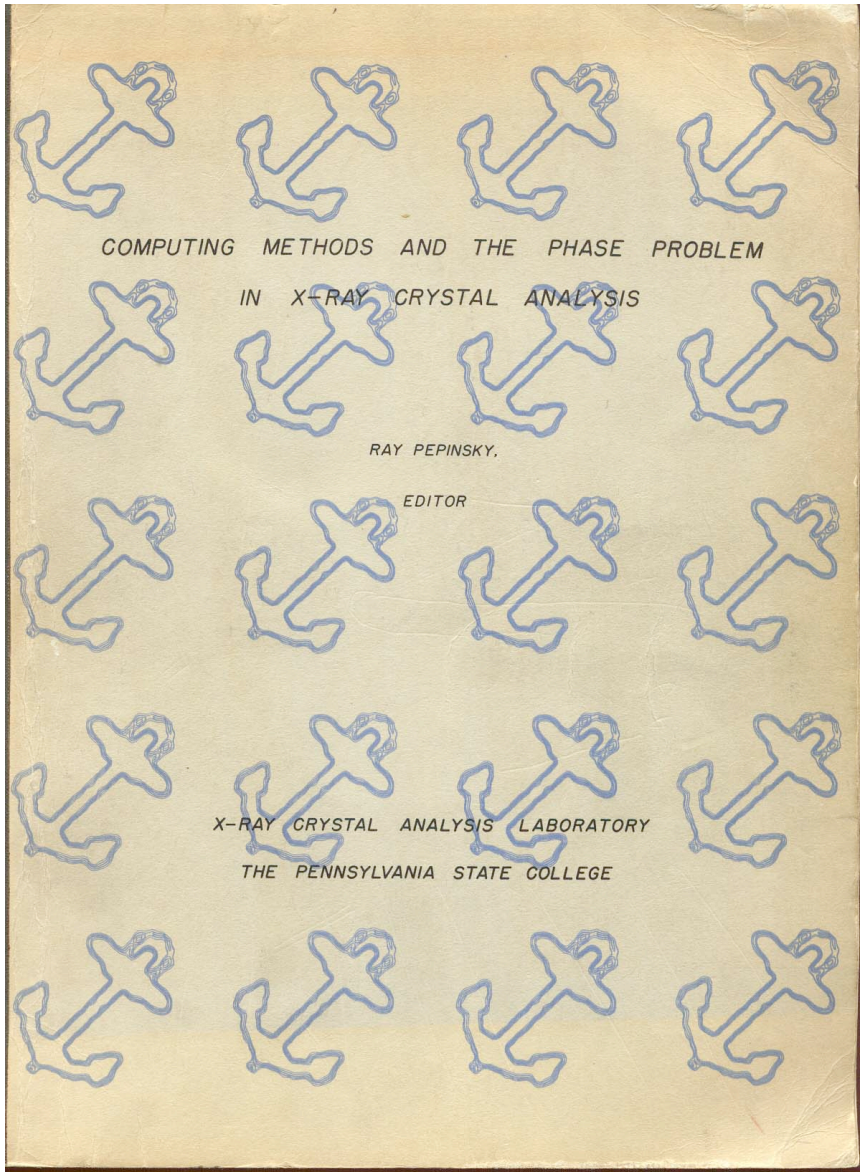



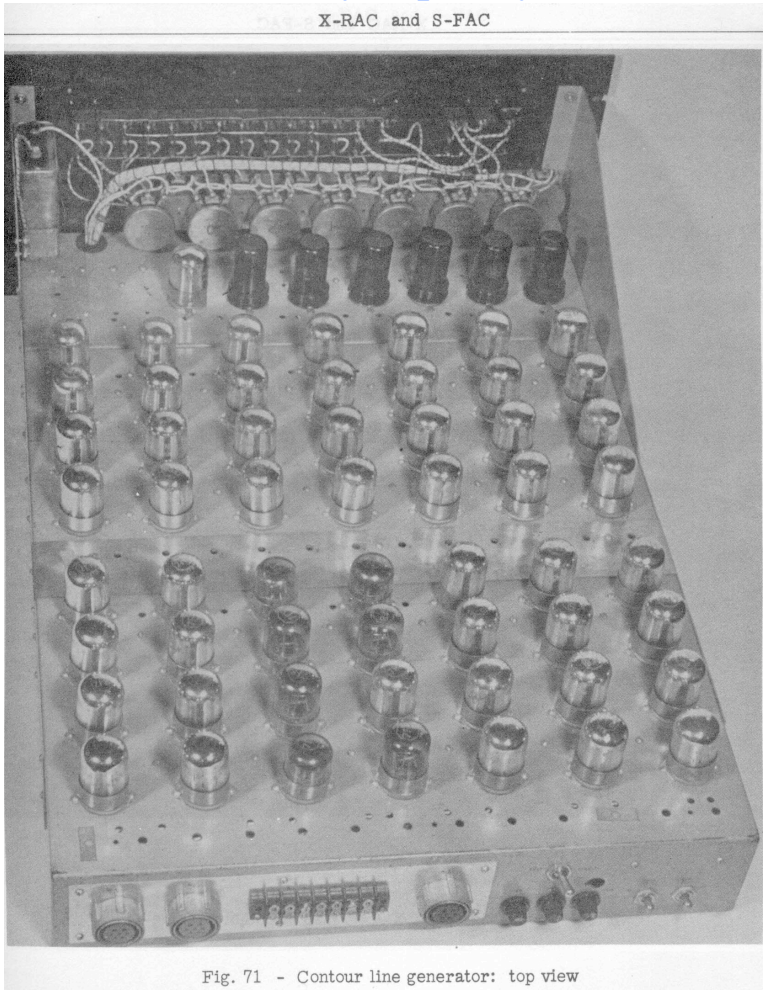

Fig. 6·21. Hollerith sorting machine.

# Early Computing – 1952




**Computing Methods and the Phase Problem in X-ray Crystal Analysis, 1952**

**Ed Ray Pepinsky**



# Early Computing

Electron density map displayed on X-RAC, David Sayre making some adjustments

X-RAC and S-FAC

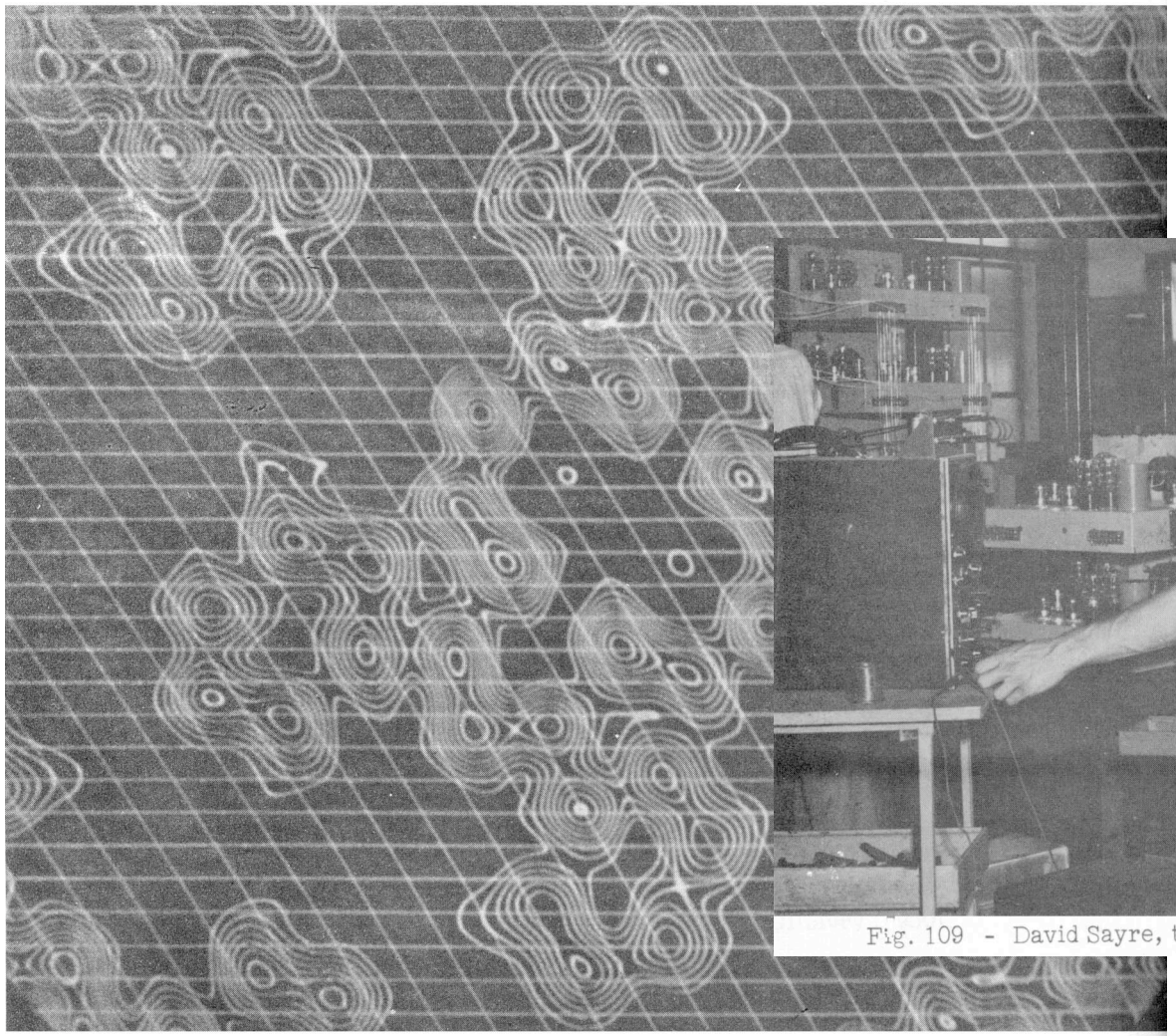


Fig. 72 - X-RAC map of phthalocyanine

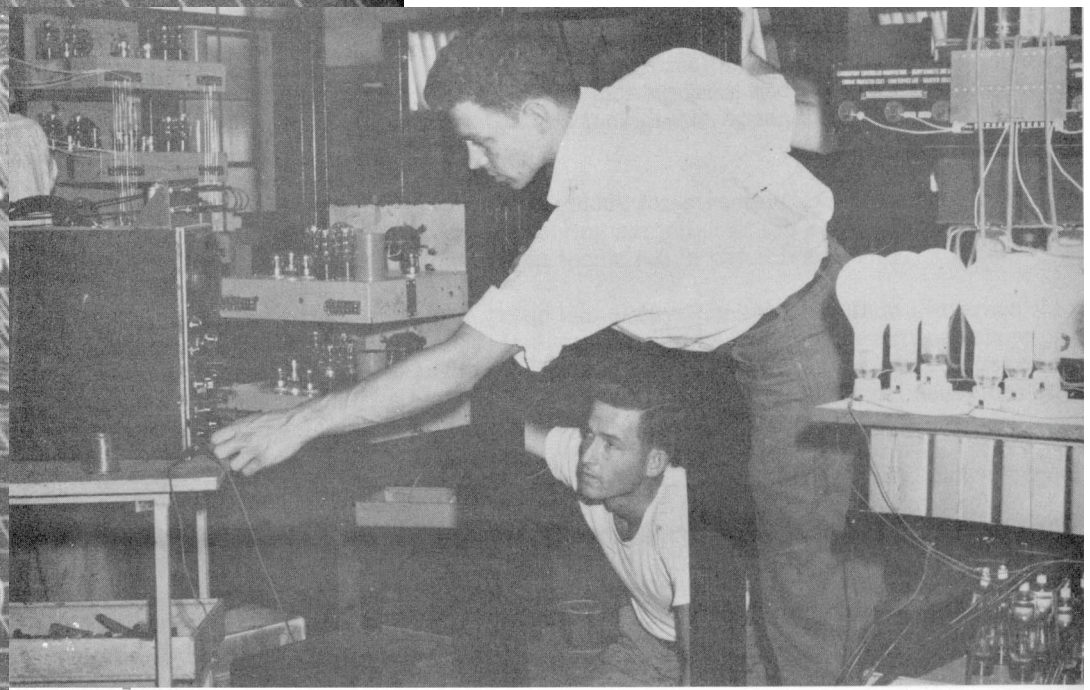


Fig. 109 - David Sayre, testing first X-RAC power regulators in Auburn.

# Fourier Modification 2008

The charge flipping algorithm
Gabor Oszlanyi and Andras Suto
Acta Cryst. (2008). A64, 123–134

*"This paper summarizes the current state of charge flipping, a recently developed algorithm of ab initio structure determination. Its operation is based on the perturbation of large plateaus of low electron density but not directly on atomicity. Such a working principle radically differs from that of classical direct methods"*

# Early Computing – Fourier Modification - 1952

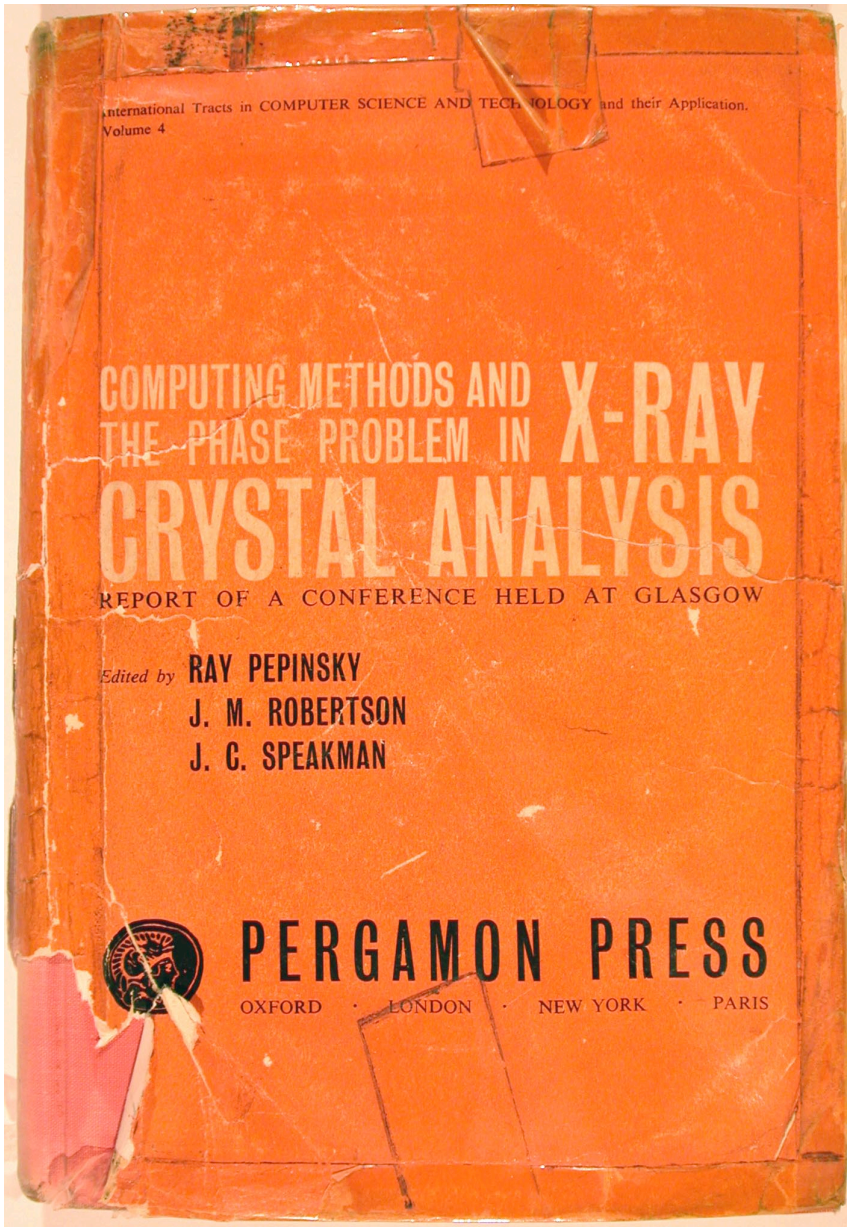
These were adventurous times in which anything might be possible if one was inventive enough.

Fred Ordway, in "Crystallographic Calculations by High-speed Digital Computers" **(1952)** wrote

"***A procedure involving successive Fourier inversions, with elimination of negative excursions of the electron density function at each step, has been coded but not yet tried***"

| | |
|---|---|
| Addition, subtraction | 0.0009 sec |
| Multiplication, Division | 0.003 sec |
| Square Root | 0.05 sec |
| Binary to decimal conversion | 0.1 sec |

# Electronic Computers - 1961




In less than ten years, analogue machines had disappeared, and were replace by digital calculators.

The second '*Computing Methods*' meeting laid out the foundations of almost every computation we do today.

Languages varied from simple autocodes to Algol

# Phase Problem (1961)

28 papers presented at this seminal meeting.

10 were concerned with solving the Phase Problem

- 4 by Patterson Methods
- 3 by Direct Methods
- 1 by Isomorphous Replacement
- 1 by Anomalous Dispersion
- 1 by Monte Carlo Methods

# Monte Carlo 1994
# Shake-n-bake

**Structure Solution by Minimal-Function Phase refinement and Fourier Filtering. II. Implementation and Applications**

By Charles M. Weeks, George T. Detitta And Herbert A. Hauptman

*Acta Cryst.* (1994). AS0, 210-220

A trial structure or model is generated that is comprised of a number of randomly positioned atoms and their symmetry-related mates sufficient to specify the origin and enantiomorph for the space group in question. The starting coordinate sets are

# Monte Carlo Methods

The Use of a Monte Carlo Method in X-ray Structure Analysis

V. Vand and A. Niggli

Computing Methods and the Phase Problem, 1961

"*the direct application to crystal structures, consisting of emitting a large number of random structures and comparing their structure factors with [the observed] ones of the structure to be solved, seems to be inapplicable owing to the low probability of a sufficiently close hit.*"

The random structures and a subset of the complete data were partially refined by the "Optimal Shift Method". Structures giving a fair agreement were used to phase the full data set.

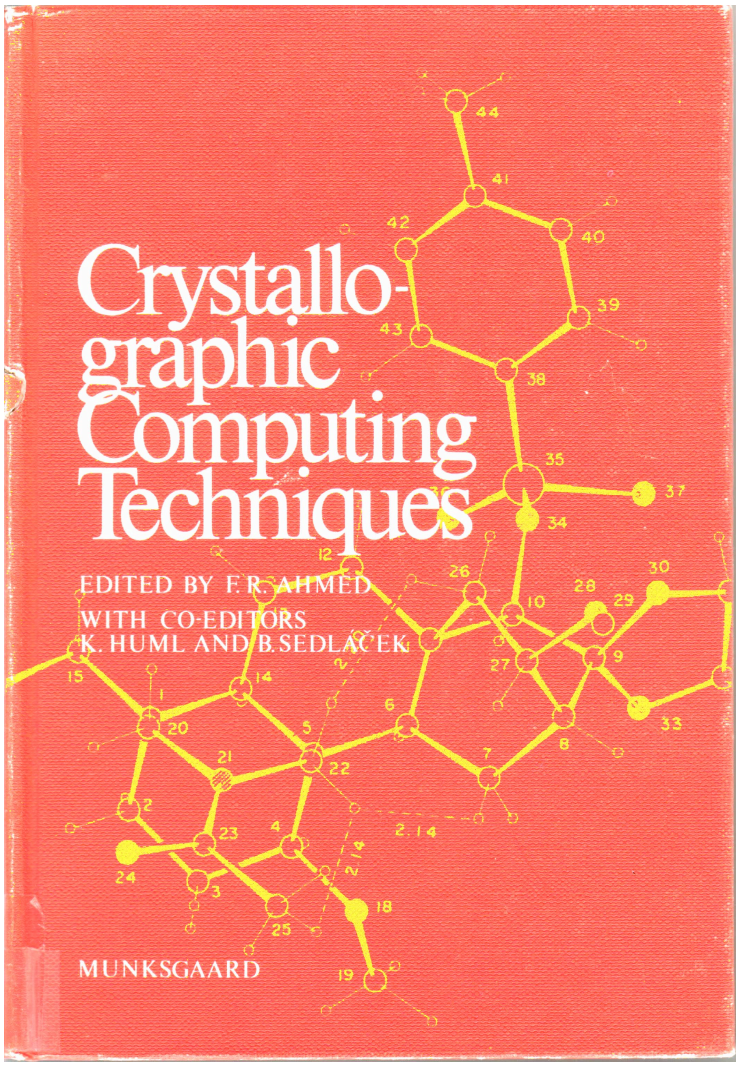
# MULTAN
# 1970

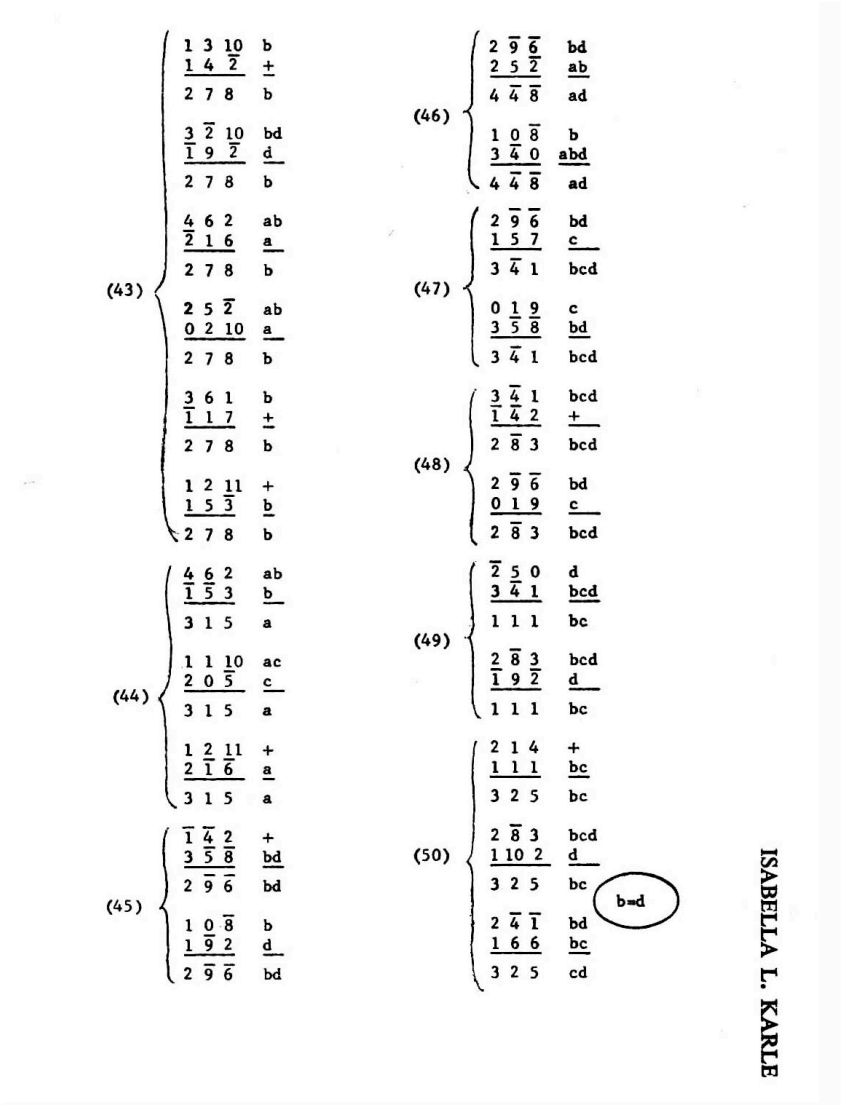
The development of multi-solution tangent formula programs for solving the phase problem quickly displaced most other strategies for small molecular structures.

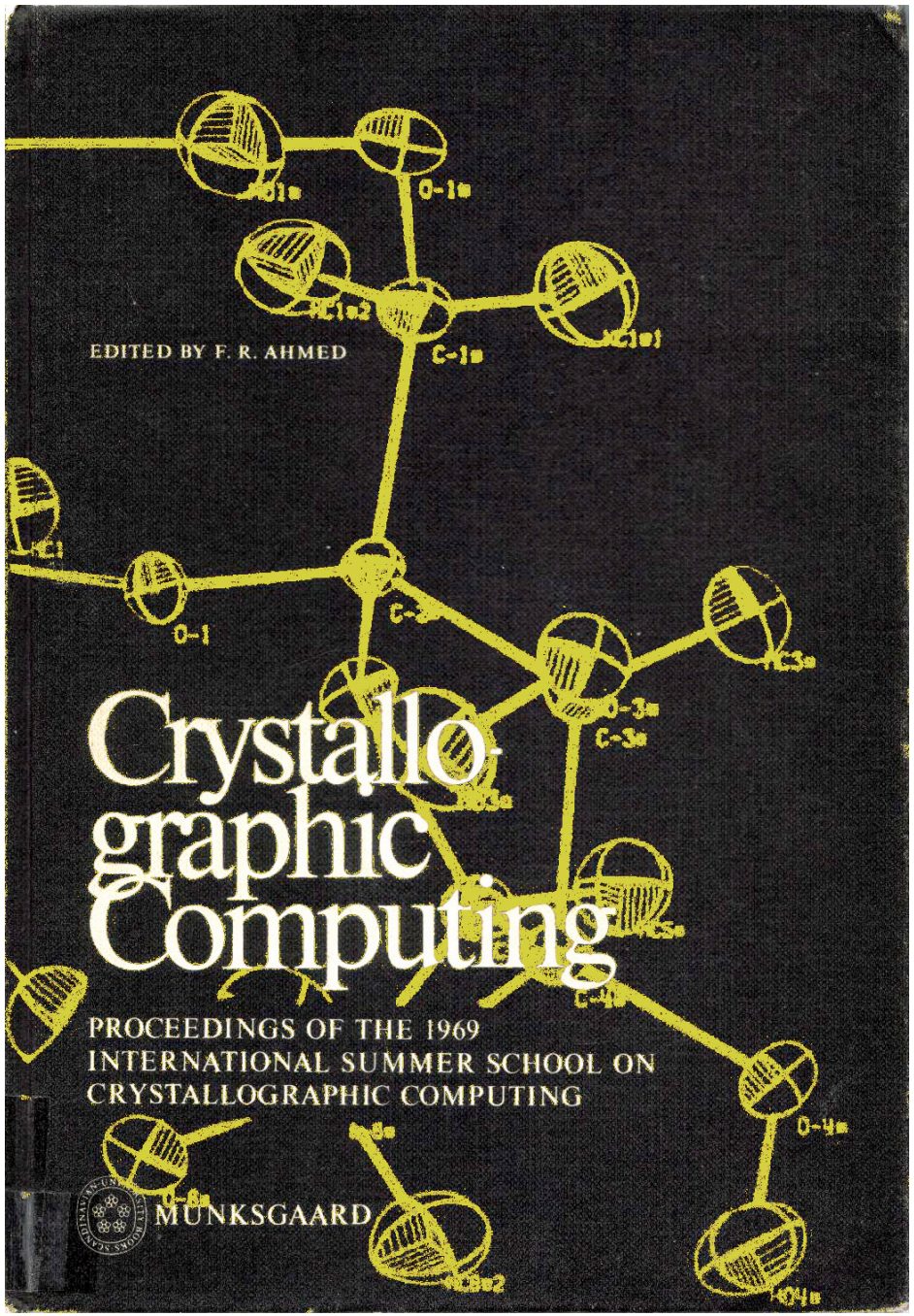
The speed of computers, random-start methods and powerful discriminators against false solutions made much of the theoretical development of the previous decades redundant.

# Direct Methods (1976)

In 1976 structure solution by hand was co-existing with the new multi-solution programs.






# Automation

## 1970

*Automatic solution of crystal structures*

Methods have been developed by Ford, Hodgson, Rollett & Stonebridge (unpublished) for automatic solution of crystal structures. A Fourier map can frequently be produced, by Patterson or direct methods, which reveals the sites of some of the atoms of a structure. If a heavy atom is located with certainty, it can be put into the structure factor calculation with full, fixed weight. Other atoms can be put in with occupation numbers (individual scale factors) of zero, and the correctness of the sites tested by refinement of the occupation numbers by least squares. After three cycles the shifts have usually become small and the improved model can be used

# Automatic Structure Analysis, 1991

Until the mid-1990's, we had been trying to build fully automated systems, e.g.

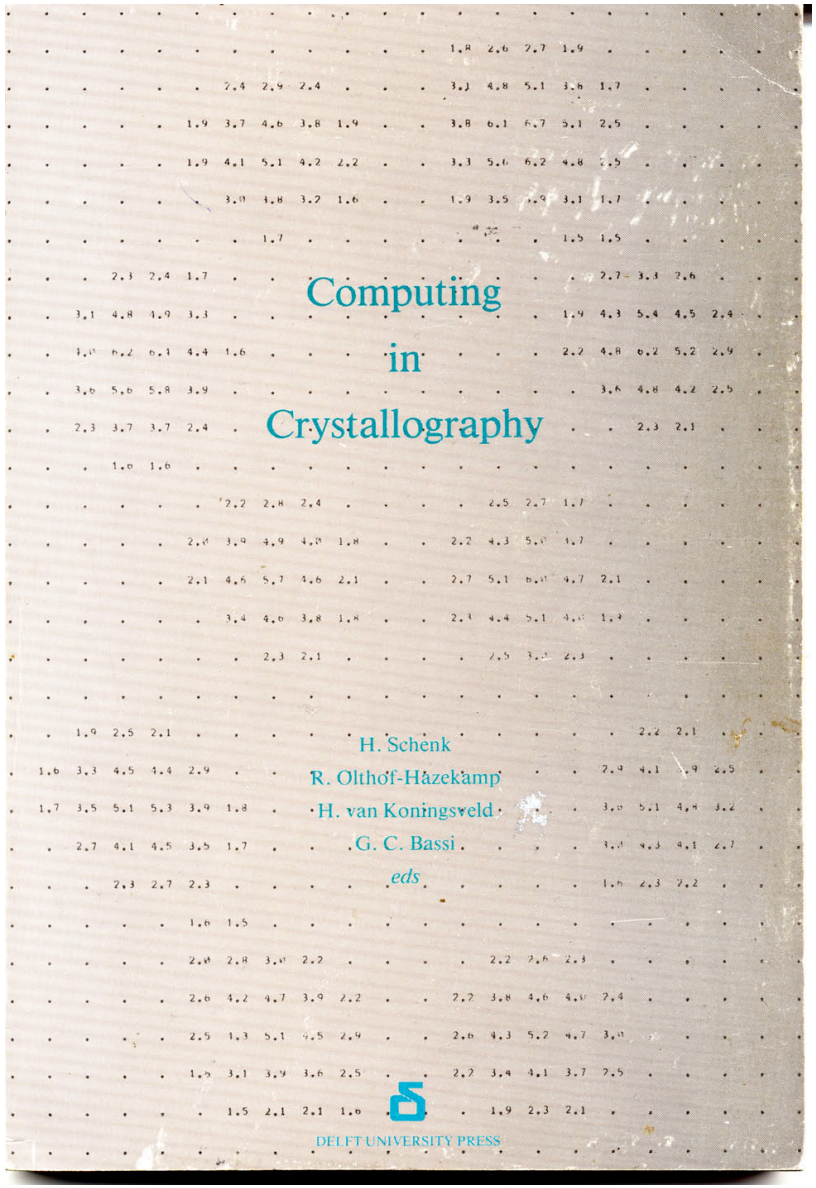
**Automatic Solution and Refinement of Crystal Structures by Means of the Package *UNIQUE***

Cascarano, Giacovazzo, Camalli , Spagna and Watkin. *Acta Cryst.* (1991). A47, 373-381

**"An automatic procedure for crystal structure solution and refinement has been devised. It is able to take decisions at each critical point of the analysis by taking careful account of all information available at that point. The procedure has been implemented into the package *UNIQUE* (*CRYSTALS*+SIR88)"**

**We eventually realised that it was easier to teach chemists some basic crystallography than to teach programs sophisticated chemistry.**

# A Turning Point




The computing school in Twente (Netherlands) in 1978, perhaps marked the high point of computing schools.

Speakers discussed both their programming philosophies, and also the detailed algorithms used in their programs.

Mini-computers made their first serious appearance.

# The Period of Great Diversity

During the 1960's -1970's software for structure analysis was being developed in almost every X-ray laboratory.

Even if one imported an established program such as ORFLS, it was often necessary to prepare small subroutines to deal with special cases. Users needed some programming skills.

# Widely Distributed Early Programs

| | |
|---|---|
| FORDAP | 1962, Zalkin |
| ORFLS | 1962, Busing, Martin & Levy |
| ORFFE | 1964, Busing, Martin & Levy |
| ORTEP | 1965, Johnson |
| MULTAN | 1970, Germain, Main & Woolfson |

Written in FORTRAN

See also IUCr World List of Crystallographic Computer Programs, Acta Cryst. (1962). 15, 1190.

"The present World List contains entries for 577 programs, many of which also appear in the ACA lists. Nearly all programs listed were in existence prior to 1 January 1962."

# Systems

## Improved Productivity Through Crystallographic Packages

Bert Frenz (1991[†]) argued convincingly that the productivity of a structure analyst can be increased if there is a smooth data-flow between the various utilities needed for the analysis.

The convenience for the user is obtained at the expense of complexity for the designer.

[†]**Crystallographic Computing 5. Ed Moras, Podjarny & Thierry**

# The Rise of the Systems

| | |
|---|---|
| X-ray | 1963, Stewart, Kundell & Baldwin |
| CRYSTALS | 1970, Carruthers and Spagna |
| XTL | 1972, Sparks |
| SHELX | 1972, Sheldrick |
| SDP | 1975, Okaya & Frenz |
| RONTGEN 75 | 1975, Andrianov |
| DIRDIF | 1975, Gould, van den Hark, Beurskens |
| XTAL | 1978, Hall & Stewart |
| CRYSTAN | 1978, Burzlaff, Bohem & Gomm |
| NRC-PDP8 | 1978, Larson & Gabe (NRCVAX) |
| PLATON | 1982, Spek (EUCLID) |
| GSAS | 1986, Larson & von Dreele |
| NRCVAX | 1987, Gabe, Lee & Le Page |
| Molen | 1990, Fair |

# and their Fall

By the mid 1990's systems were dying

**Extinct:**

- X-ray 1963, Stewart, Kundell & Baldwin
- NRC-PDP8 1978, Larson & Gabe (NRCVAX)
- NRCVAX 1987, Gabe, Lee & Le Page
- Molen 1990, Fair

**Endangered:**

- CRYSTALS 1970, Carruthers and Spagna
- DIRDIF 1975, Gould, van den Hark, Beurskens
- SDP 1975, Okaya & Frenz
- XTAL 1978, Hall & Stewart
- PLATON 1982, Spek (EUCLID)
- GSAS 1986, Larson & von Dreele

# Some Commercial Packages

Massive effort has been put into creating comprehensive commercial packages, yet most of them have faded away.

The costs of developing and maintaining software are enormous.

Robert Langridge, in his address at the special session entitled "Crystallographic Computing for the 1990's: What Can We Expect?" stated that ***software maintenance represents 75% of the cost over the lifetime of a computer system***.

**In other words, writing a new program is only a small part of the final cost if it is to remain in use over a long period.**

# Built to Last

It might be thought that ‘program systems’, developed by a group of workers, would be the most durable because understanding of the inner working is distributed across the group.

Paradoxically, it turns out that the most portable and enduring software:

- has naïve input
- has plain-text output
- is focussed onto a narrow range of tasks
- was generally written or maintained by one person
- **The “one person” has a secure research post**

# Built to Last

Paradoxically, the most portable and enduring software has naïve input, plain-text output, is focussed onto a narrow range of tasks, and was generally written by one person.

Instead of a monolithic program, a more maintainable product consists of a range of modules which speak to each other through very simple, well defined, interfaces.

Each module can be developed more or less independently of the others.

# Maintainability & Understandability

**Reciprocal Space Tutorial - George M. Sheldrick**
**Siena, 2005**

symmetry and reflection data all in free format.
Trigonal bovine trypsin P3121 #152
CuKa
54.735 54.735 106.786 90 90 120
6 symops follow, then h,k,l,I and sig(I)
1 0 0 0 1 0 0 0 1 0.0 0.0 0.0
0 -1 0 1 -1 0 0 0 1 0.0 0.0 0.333333
-1 1 0 -1 0 0 0 0 1 0.0 0.0 0.666667
0 1 0 1 0 0 0 0 -1 0.0 0.0 0.0
1 -1 0 0 -1 0 0 0 -1 0.0 0.0 0.666667
-1 0 0 -1 1 0 0 0 -1 0.0 0.0 0.333333
22 -3 -41 21.38 3.27
-2 6 -12 162.92 11.71
-19 4 -32 81.44 6.82
-13 -9 -51 16.44 3.87
etc. 389596 reflections in total,
terminated by the end of the file.

**The task was to sort the reflections into a standard order, eliminate systematic absences, merge equivalent reflections and detect centric reflections.**

**The half-dozen solutions to the problem showed many differing virtues, from being very brief to expansive but self-explanatory.**

# Maintainability & Understandability

**Reciprocal Space Tutorial - George M. Sheldrick**

http://journals.iucr.org/iucr-top/comm/ccom/siena2005/notes.html

| Author and Language | No. of pages | |
|---|---|---|
| George Sheldrick Fortran 77 | 3 | |
| Tim Gruene C++ | | 7 |
| Michel Fodje C++ | 8 | |
| Juan Rodriguez-Carvajal Fortran-95 | 5 | |
| Bradley Smith Java | 12 | |
| Ralf W. Grosse-Kunstleve Python | 1 | |
| CRYSTALS Simple Datafile | 1 | |

# George Sheldrick

```
      PROGRAM SMERG
C
C Fortran-77 sort-merge solution for Siena exercise
C
      PARAMETER(NX=2000000)
      INTEGER IH(NX),IK(NX),IL(NX),IP(NX),IQ(NX)
      REAL FF(NX),SI(NX),SY(12,24)
C
C Read data from standard input
C
      READ(*,'(/)')
      READ(*,*)NS
      READ(*,*)((SY(I,J),I=1,12),J=1,NS)
      NR=0
  1   N=NR+1
      IF(N.GT.NX)STOP '** Too many reflections **'
      READ(*,*,END=5)IH(N),IK(N),IL(N),FF(N),SI(N)
      IP(N)=N
      NR=N
C
C Convert reflection indices to standard setting
C
      U=REAL(IH(N))
      V=REAL(IK(N))
      W=REAL(IL(N))
        DO 4 M=-1,1,2
          DO 3 J=1,NS
          I=M*NINT(SY(1,J)*U+SY(4,J)*V+SY(7,J)*W)
          K=M*NINT(SY(2,J)*U+SY(5,J)*V+SY(8,J)*W)
          L=M*NINT(SY(3,J)*U+SY(6,J)*V+SY(9,J)*W)
          IF(L.LT.IL(N))GOTO 3
          IF(L.GT.IL(N))GOTO 2
          IF(K.LT.IK(N))GOTO 3
          IF(K.GT.IK(N))GOTO 2
          IF(I.LE.IH(N))GOTO 3
  2       IH(N)=I
          IK(N)=K
          IL(N)=L
  3       CONTINUE
  4     CONTINUE
        GOTO 1
etc
```

**The program is self-contained, uses no obscure FORTRAN features, and with a little effort even the DO 4 ….4 CONTINUE loop can be understood**

# Ralph Grosse-Kunstleve

```
# sort_merge_initial.py was written in exactly 30 minutes while
# sitting in the audience as others explained their solutions.
#
# It doesn't solve the exercise exactly, but demonstrates how to
# work with the high-level cctbx facilities to solve most of the
# exercise. Note that sort_merge.py produces significantly more
# information than was requested, e.g. the space group name,
# data completeness, etc.
#
from cctbx.array_family import flex
from cctbx import crystal
from cctbx import uctbx
from cctbx import sgtbx
from cctbx import miller
import sys
def run(args):
  assert len(args) == 1
  lines = open(args[0]).read().splitlines()
  title = lines[0]
  unit_cell = uctbx.unit_cell(lines[1])
  n_symops = int(lines[2].split()[0])
  space_group = sgtbx.space_group()
  for line in lines[3:3+n_symops]:
    coeffs = [float(field) for field in line.split()]
    space_group.expand_smx(sgtbx.rt_mx(coeffs[:9], coeffs[9:]))
  crystal_symmetry = crystal.symmetry(
    unit_cell=unit_cell,
    space_group=space_group)
  miller_indices = flex.miller_index()
  data = flex.double()
  sigmas = flex.double()
  for i_line in xrange(3+n_symops,len(lines)):
    fields = lines[i_line].split()
    assert len(fields) == 5
```

**This was the tersest solution, and *almost* solved the problem as set.**

**On a computer with the full toolbox installed, the development environment enables the reader to backtrack to discover the functionality of the modules.**

```
if (__name__ == "__main__"):
  run(sys.argv[1:])
etc. 389596 reflections in total
```

# Program or User-Commands?

George's solution is obviously a program in the classical sense.

Ralph's solution is less clearly defined. It is evidently a program in that it instructs the computer, but it is more like a set of user commands in that the maths is hidden from view.

*The advent of scripting languages blurs the line between programs and data.*

# CRYSTALS

**This data file causes CRYSTALS to perform much the same calculations as Ralph's program, in much the same time as George's.**

File "job.dat"

```
#use start.dat
#OPEN HKLI reflections.hkl
#LIST 6
READ TYPE=FREE
END
#CLOSE HKLI
#SYSTEMATIC
END
#SORT
END
#MERGE
END
```

File "start.dat"

```
#TITLE Trigonal bovine trypsin P3121
# 152 CuKa
LIST 1
REAL 54.735 54.735 106.786 90 90 120
END
#SPACEGROUP
SYMBOL P 31 2 1
END
```

File "reflections.hkl"

```
22 -3 -41 21.38 3.27
-2 6 -12 162.92 11.71
-19 4 -32 81.44 6.82
-13 -9 -51 16.44 3.87
etc. 389596 reflections in total,
```

# Topas - Alan Coelho

```
' DCOND2 + Pb + silica wool 25tns/RT, run # 1 (Overall sum)
                  r_exp  2.858
                  r_exp_dash  18.720
                  r_wp  3.702
                  r_wp_dash  24.245
                  r_p  4.724
                  r_p_dash  30.647
                  weighted_Durbin_Watson  1.333
                  gof  1.295
'do_errors
xdd prl52076_87_25tns_tof_xye.dat xye_format
                  neutron_data
                  x_calculation_step = Yobs_dx_at(Xo);
      weighting = If(SigmaYobs < .01, 1, 1/SigmaYobs^2);
'#################################################################

 TOF_LAM(0.001)
                                TOF_x_axis_calibration(!t0, 3.88810, !difc, 4677.19027, !difa, 0.13451)

   prm !exp1  7.15326 min 1 max 10

     TOF_Exponential(, 47.03716,, 109.24076, exp1, difc, +)
   push_peak
     TOF_Exponential(, 17.27854,,  3.43204, exp1, difc, +)
   bring_2nd_peak_to_top
     TOF_Exponential(, 1431.31523,,  33.75847_LIMIT_MIN_-13.5347376, exp1, difc, -)
     scale_top_peak    88.4790102 min .001 del = Val .05 + 1;
   add_pop_1st_2nd_peak

'#################################################################
```

**Coelho has developed a "crystallographic" language which enables the user to tailor a computation to the individual task.**

**This includes declaring new variables, and defining new calculations.**

# Case Study:

## Xtal and SHELXx

**You could run an X-ray laboratory using either of these packages.**

**Why has Xtal almost disappeared?**

# Xray and Xtal

The first reference to Xray was about 1963, (Stewart, Kundell & Baldwin). This grew into a large system written in ‘pidgin Fortran’

# XTAL: New Concepts in Program System Design

S. R. Hall, James M. Stewart & Robert J. Munn

*Acta Cryst.* (1980). A36, 979-989

To try to get round implementation difficulties on different systems, the FORTRAN program was re-cast into RATMAC, a language that could be pre-processed into RATFOR, and then FORTRAN

# XTAL Modules


**System Editor**
Syd Hall
**Co-Editors**
James Stewart
Howard Flack
Geoff King
Doug du Boulay
Roeli Olthof-Hazekamp

| | | | | | |
|---|---|---|---|---|---|
| ABSORB | ADDATM | ADDREF | ATABLE | BAYEST | BONDAT |
| BUNYIP | CBAZA | CHARGE | CIFENT | CONTRS | CREDUC |
| CRILSQ | CRISP | CRYLSQ | DIFDAT | FC | FOURR |
| GENEV | GENSIN | GIP | LATCON | LISTFC | LSABS |
| LSLS | LSQPL | LSRES | MAPLST | MODEL | MODHKL |
| NEWCEL | NEWMAN | ORTEP | PATSEE | PIG | PLOTX |
| PREABS | PREVUE | REFCAL | REFM90 | REGFE | REGWT |
| RFOURR | RIGBOD | RMAP | RSCAN | SHAPE | SHELIN |
| SLANT | SORTRF | STARTX | VUBDF | XTINCT | |

# Formal Design

The X-ray system, and later the Xtal system, was a carefully managed project. It was designed to be maintainable and extensible. Documentation was an important feature.

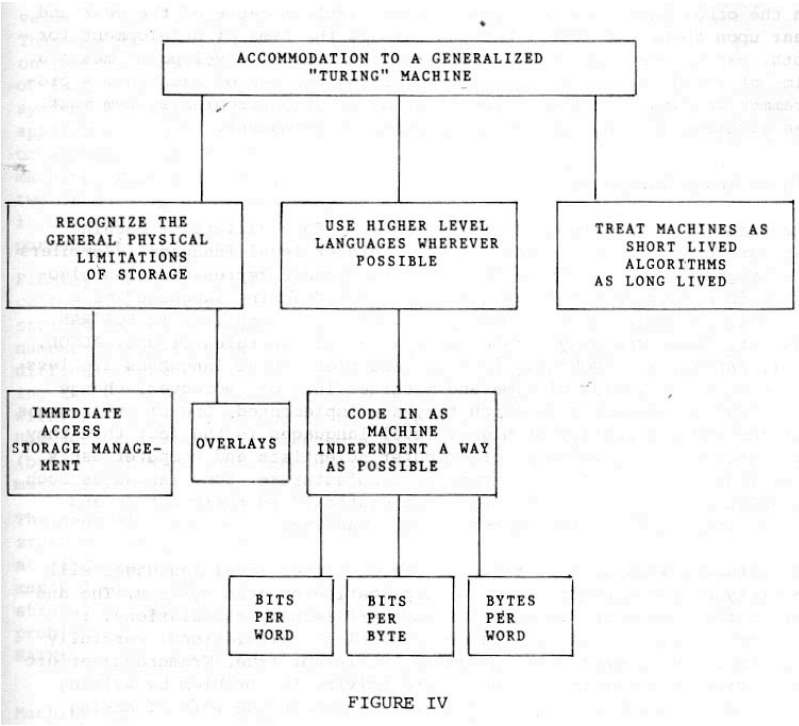



FIGURE IV

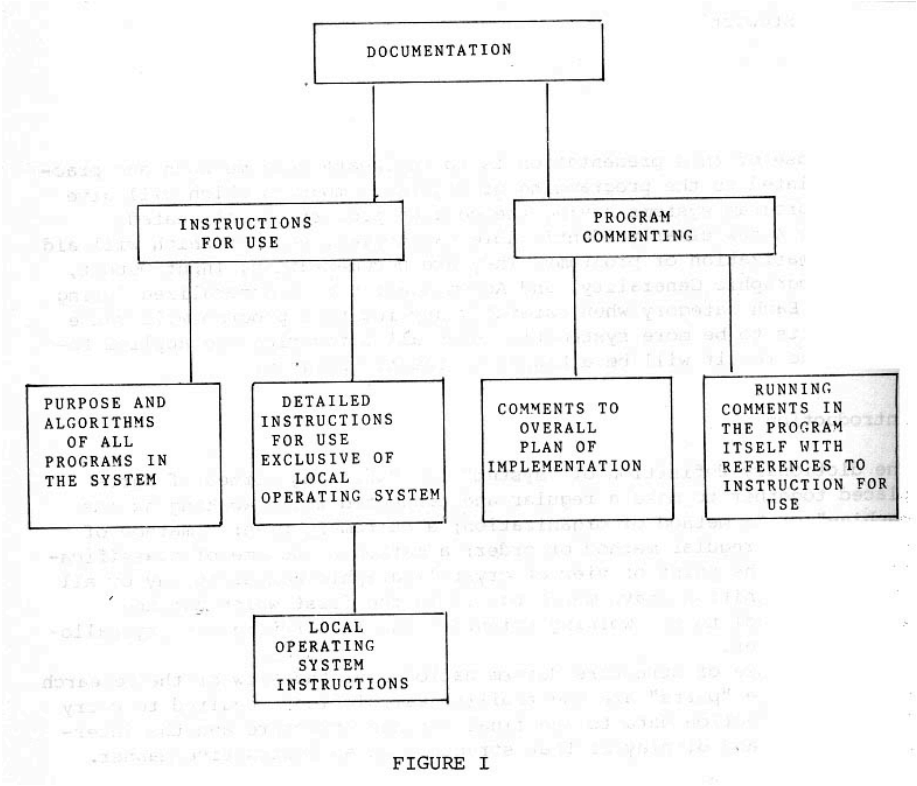
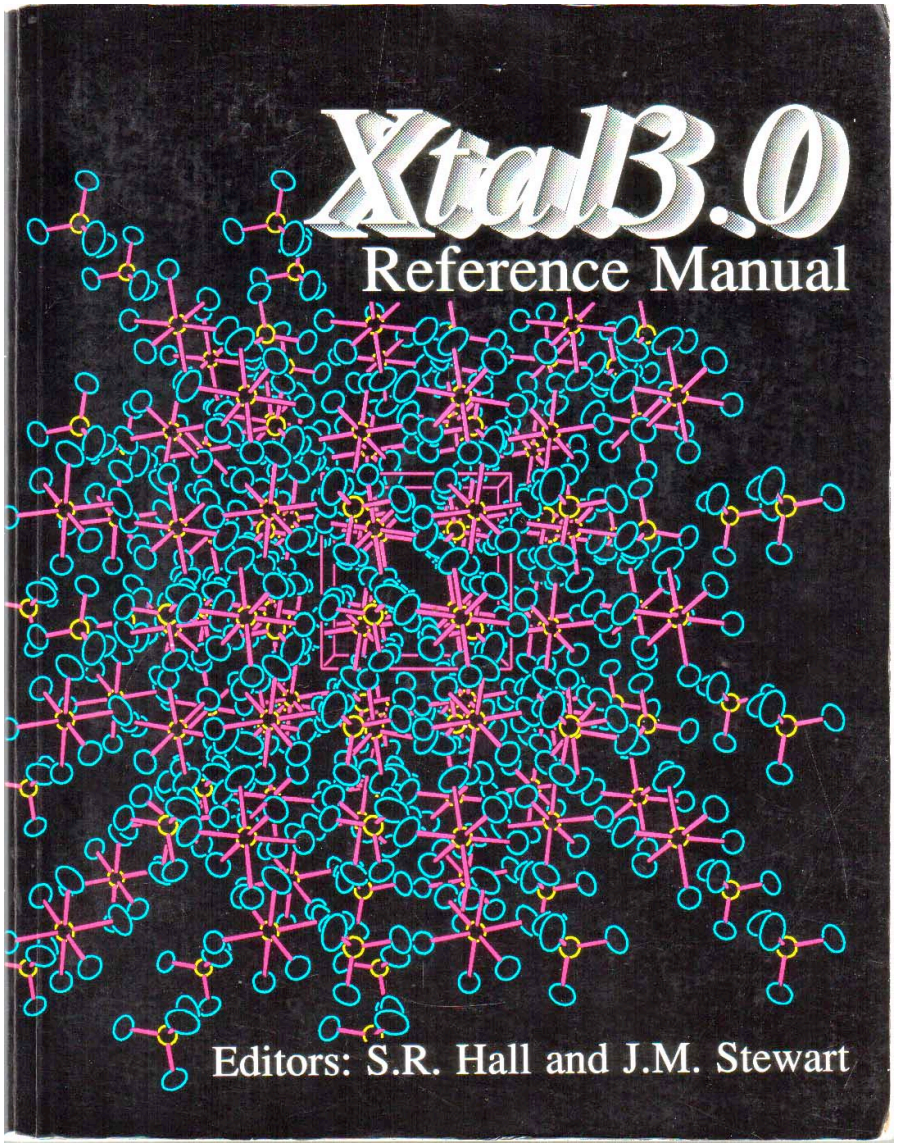



FIGURE I

**Even the documentation was documented**

# Formal Design




In addition to a modest user guide, the authors also produced a very detailed description of the algorithms used.

In the 400-page Reference Manual, users could find out more or less exactly what each computation was doing.

Published 1990

# Informal Design

At about the same time that X-ray was morphing into Xtal, George Sheldrick was creating SHELX.

Note that there was no version number – GMS had no idea how important his creation was to become.

Documentation was minimal and the algorithms were not described in detail.

A definitive version was released in 1976 as SHELX76.

The original program, which can still easily be re-dimensioned to handle modestly large structures, was a complete system including data processing, structure solution, refinement and table generation.

# Informal Design

The ‘single executable’ program has inevitable maintenance problems when ever new features need to be added.

A more manageable model is to divide the overall problem into pieces that are more-or-less independent, and focus effort onto each problem in turn.

Over the years, GMS has released updated modules, e.g.

SHELXS
SHELXL
SADABS
SHELXD
XP

The SHELX76 ‘system’ has been divided into separate parts with a consistent data and instruction input style.

# System Design

| | Pro | Con |
|---|---|---|
| Stand-alone dedicated programs | Low development cost<br>Low maintenance cost<br>Easily *linked* to other systems<br>Easy to add a low-sophistication GUI | Difficult to *integrate* closely into other systems.<br>Difficulties with sharing intermediate results |
| Closely Integrated system | Big increase in functionality for complex problems | High development cost<br>Very high maintenance cost |

# Why do Programs Die Out?

Inappropriate language

Too machine specific

Over-complex internal organisation

Outdated algorithms

Poorly Documented

Unacceptable user-interface

Over-proliferation of unwanted goodies

Small user-base

Licence Costs

**No-one to support it**

# Future-proofing

This can be aimed at by:

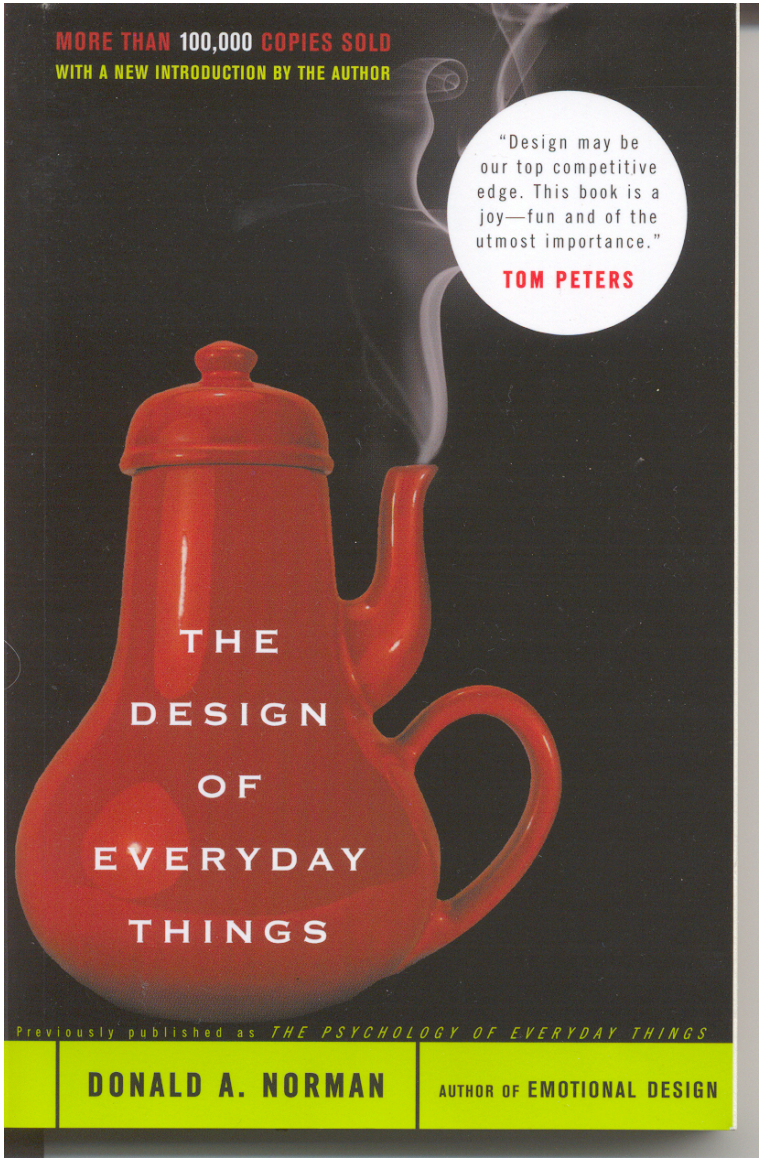
| | SHELX | Xtal |
|---|---|---|
| **Using a slowly-evolving language that is well characterised** | **X** | |
| **Not using proprietary extensions to the language** | **X** | |
| **Not using hardware specific facilities** | **X** | **X** |
| **Easy to install/compile** | **X** | |
| **Making minimal or no use of external libraries** | **X** | **X** |
| **Providing definitive user-documentation** | **X** | **X** |
| **Providing worked examples** | **X** | **X** |
| **Having a well defined data structure that can be extended** | | **X** |
| **Providing definitive programmer documentation** | | **X** |
| **Commenting the code, especially the little tweaks that have evolved to improve stability or functionality** | | **X** |

# Case Study:

## Xtal and SHELXx

| Xtal | SHELXx |
|---|---|
| Large (at the time) user-base | Very large user-base |
| Continuously updated | Definitive editions |
| Exhaustive selection of options | Small, focussed, selection of options |
| Verbose input | Terse instruction set |
| Complicatd to install | Single, easily compiled, file |
| Continuously updated structural data base | Each job totally self contained |

# User Interface




**Bad Ideas**

- Complex interface for a simple task
- Over-simple interface for a complex task

**Good Ideas**

- The user can easily get started on simple tasks
- The user can form a mental image of the activity.

**GPS example.**

If you come across unexpected road works, you may need a map in order to suggest alternative routes to the instrument.

# CRYSTALS

40 years of accumulated experience and development

A complete package from hkli files through to publication cifs *with the exception of structure solution*.

Collaboration with Sheldrick, Giacovazo and Palatinus provides seamless structure solution.

Internal code and seamless links to CIFcheck and PLATON provide structure validation.

# Richness and Complexity

CRYSTALS is a very ‘rich’ program – there are many hundreds of user-adjustable parameters.

*This provides the new user with a bewildering range of choices – it is too complex*

“Scripts” enable an experienced crystallographer to produce pre-prepared, simplified, strategies for routine tasks.

# Home-made Interpreters

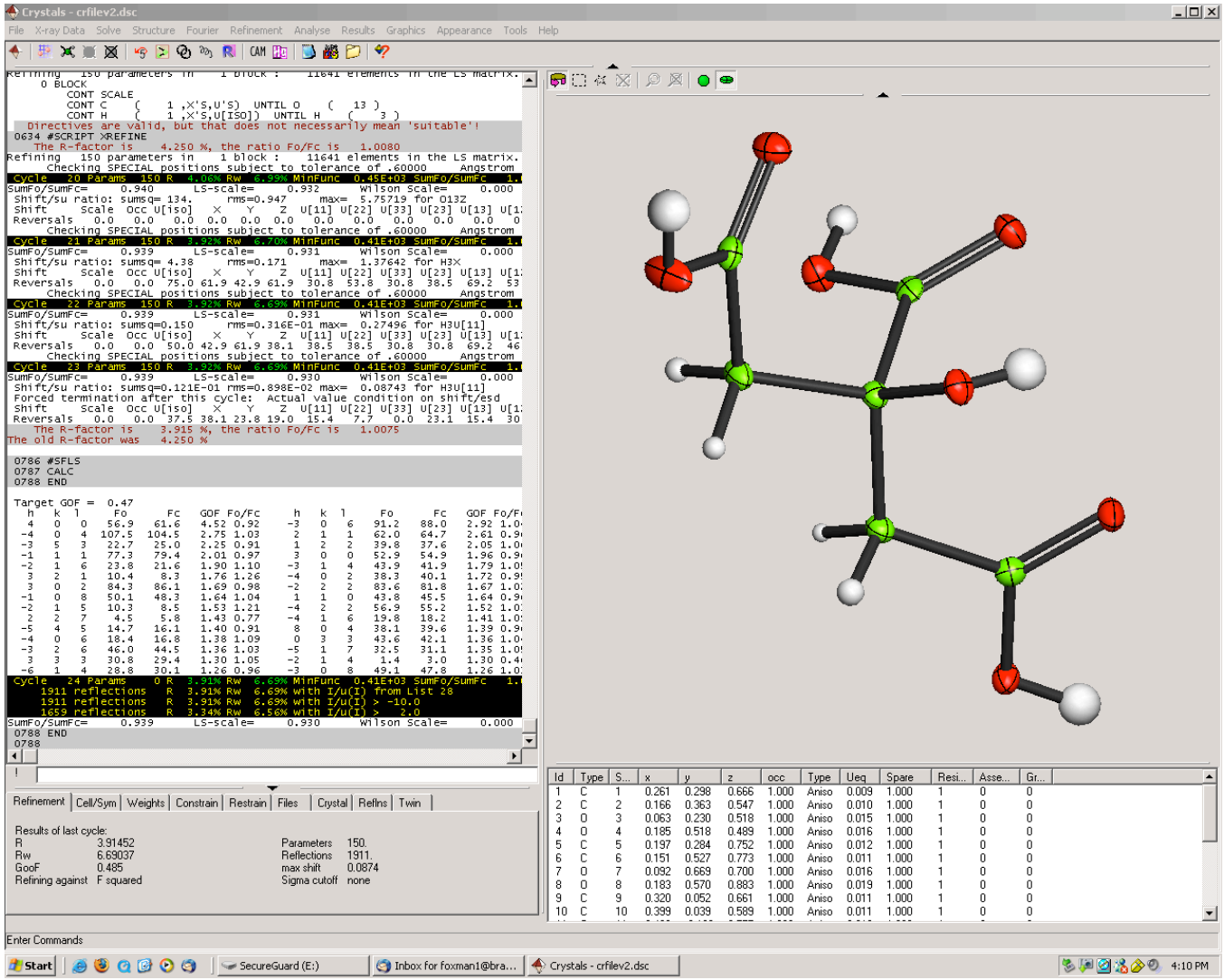
```
%SCRIPT COLLECT
A SCRIPT to help collect atoms into a molecule, or bring new peaks close to existing
atoms.
%% check we have an atom list
%  IF EXISTS 5 .NE. 1 THEN
  You do not have an atom list.
%  FINISH
%  END IF
%  BLOCK
%    ON ERROR REPEAT
%    QUEUE REWIND
%    CLEAR
%      VERIFY ALL NEW NONE
%      GET NOSTORE FINAL ABBREVIATED -
  'Collect all the atoms, or just the new peaks' 'NEW'
%      IF VALUE .EQ. 1 THEN
%        INSERT 'SELECT TYPE = ALL'
%        QUEUE SEND
%        CLEAR
%      ELSE IF VALUE .EQ. 2 THEN
%        INSERT 'SELECT TYPE = PEAKS'
%        QUEUE SEND
%        CLEAR
%      ELSE
%        FINISH
%      END IF
%  END BLOCK
%  COPY '#COLLECT'
%  QUEUE PROCESS
%  COPY 'END'
%END SCRIPT
```

**'CRYSTALS' Issue 7 (Betteridge, Prout and Watkin, Oxford, 1983)**

**The program can enter a dialogue with the user.**

# 1995. Richard Cooper added a GUI.

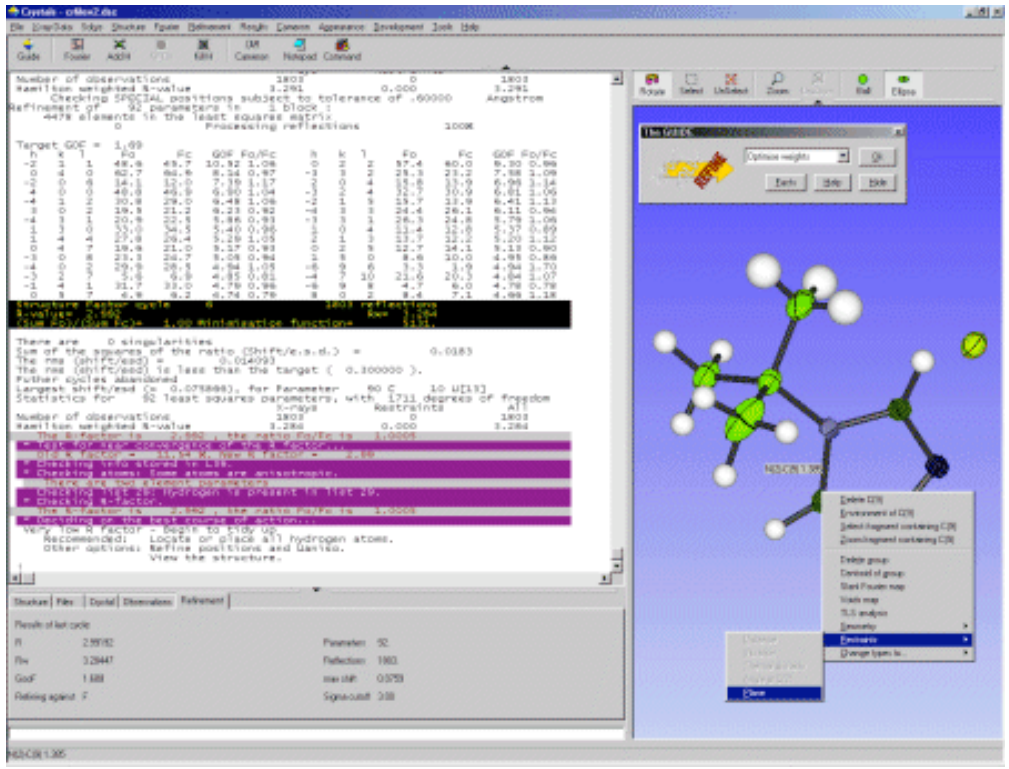
## The Guide enables the user to apply chemical knowledge.



# CRYSTALS 2000

Most of the infra-structure needed to enable enhanced functionality to be added was in place for The Millennium.

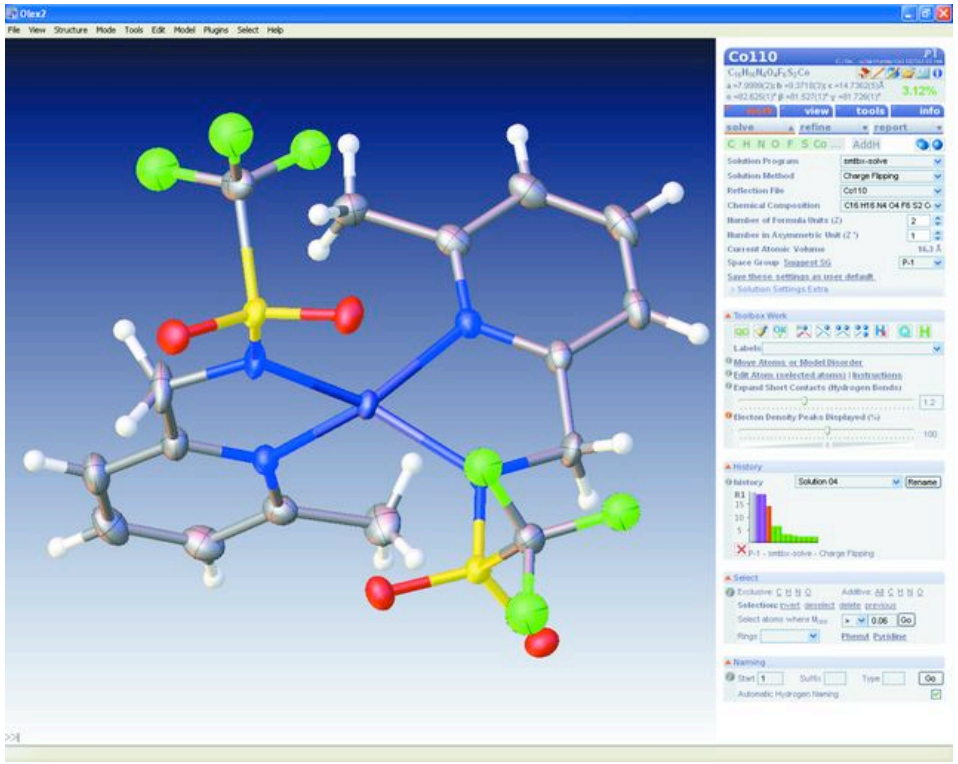
Since then the underlying FORTRAN and user accessible features have continued to be developed in response to the communities needs.



Publicity material, 2003

# Age Concern

As part of the EPSRC “Age Concern” project the group in Durham has shared complete access to CRYSTALS source code and SCRIPTS to help them develop their own user interface.




OLEX2: a complete structure solution, refinement and analysis program
Oleg V. Dolomanov, Luc J. Bourhis, Richard J. Gildea, Judith A. K. Howard and Horst Puschmann

# Age Concern

The project has also given rise to two refinement sub-systems.

**smtbx/cctbx**

"focusing on those key computational details which have been the treasures of the classic programs CRYSTALS or SHELX."

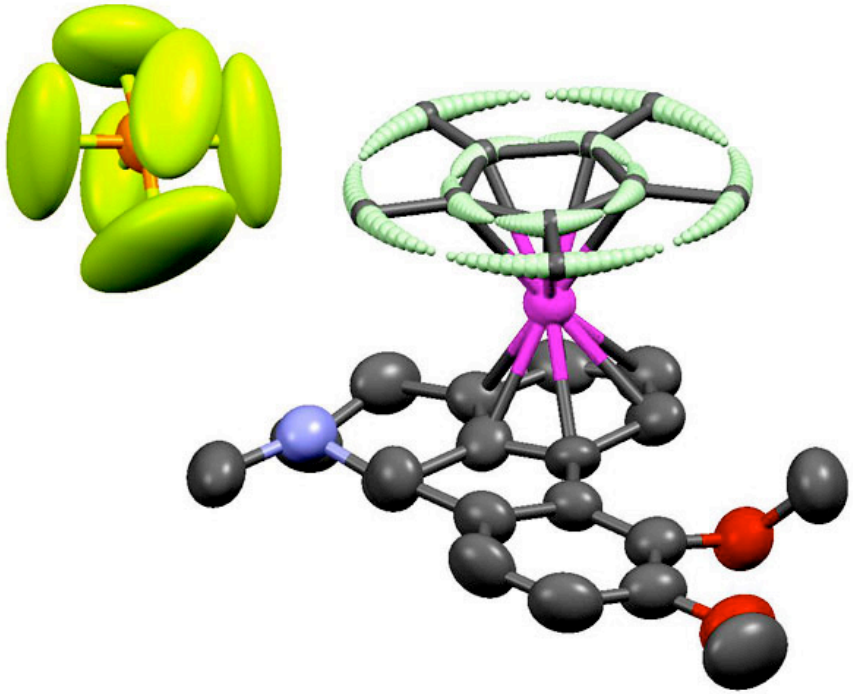
**ACA 2010: 07.26.4 Solution and Refinement with the cctbx and smtbx**

Luc Bourhis, Oleg Dolomanov, Richard Gildea, Judith Howard, Horst Puschmann

**SMTK/cctbx**

"provide a modelling design process, where the model formulation is kept separate from the optimization process."

SMTK – a small-molecule toolkit library for crystallographic modelling and refinement Mustapha Sadki* and David J. Watkin J. Appl. Cryst. (2011). 44, 52–59



# SMTK

**Example. The equation of a 3D plane is:**

**A x +B y + C z + D = 0, divide by C, provided C is not zero.**

**a x + b y + z +d =0**

**The distance of any atom (x,y,z) from the plane is**

**Distance = | ax + by + z + d | / sqrt($a^2 + b^2 + 1$)**

**We must find the values of a,b and d which minimise the sum of the distance of all atoms of interest from the plane.**

```
template<T>
 T Distance2Plane ( Array2D<T> &p , Array2D<> &data)
{
    return abs(p[0]* data[0] +  p[1]* data[1] + data[2]  + p[2]) / sqrt( p[0]*p[0] + p[1]*p[1] +1);
}

     // instantiate the least squares object using the template function with optional
arguments:


    bool  need_covar = true;
    lm_solver<Distance2Plane> lsq(data_points, m, n, need_covar);
    // generates the lsq model f & f(x) and then we call for minimisation
    ret = lsq.minimise (Observation);
```

# Digital Computing

Digital computers have revolutionised crystallography.

Has computing replaced thinking?

Bob Sparks matrix-accumulation benchmark (1976):

| | |
|---|---|
| Microvax 3800 (1989) | 1,824 secs |
| 1.8Ghz Athlon, (2005) | 3secs |
| 3.0Ghz Intel Duo (2010) | 1sec |

We can now do in **one day** what would have taken **three days** in 2005, and almost ***five years*** in 1990

# Molecular Recognition

The speed of instruments and computers is such that crystallographers have become the slowest link.

**'The crystallographer uses intuitively the concept of similarity amongst structures in the survey of F-maps .. It appears desirable now to minimise such human effort ..'**

*Formal Aspects of the Interpretation of Fourier Maps, J.C.J. Bart & A. Busetti, Acta Cryst. (1976). A32, 927*

**Many programs can check if a crystal structure corresponds to a proposed structure. If the match is not exact, it becomes difficult to make a machine-hypothesis about the nature of the differences.**

**Often a chemist can just glance at the structure and decide if it is feasible or not.**

# macHine valiDATion

Word Processors have shown us both the *power and* **the risks of** machine val1dation of routine tasks.

spelling & grammar checks, **automated** caπitalisation *etc.* filter out MANY errors, but it an experienced reader is still required to review the final documen T.

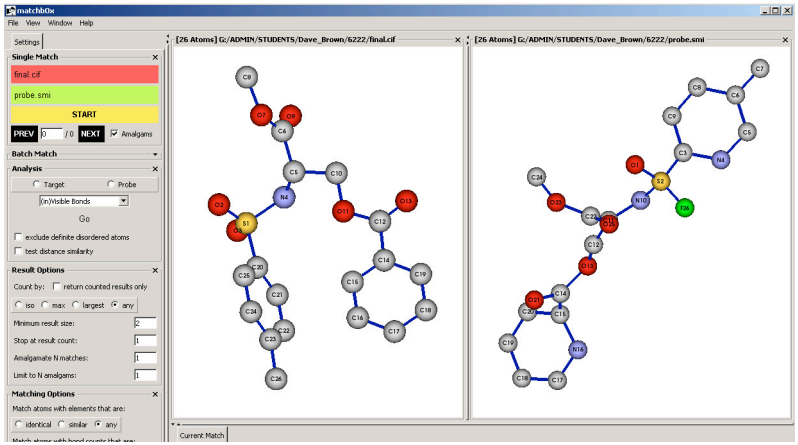
# Age Concern

## Regression to the 1990's

*Once again, we are trying to teach computers some chemistry.*

Spotting that a proposed structure is correct is not difficult. Spotting if it is nearly correct (and in what way it is incorrect) is much more difficult.

O=S(c1Ncc(C)cc1)(NC(COC(C2=NC=CC=C2)=O)C(OC)=O)=F




Dave Brown is working on a project to use chemical information in the form of SMILES strings to help analyse proposed structure solutions and unpick disorder.

# The Future:
# The Microsoft Syndrome

1950

"These were adventurous times in which anything might be possible if one was inventive enough."

2010

"If it's not on a menu, it cannot be done"

**The user-community is no longer excited by computers**

**The GUI must encode the strategy to be applied to the underlying maths.**

# A Test

Crystallography students now lack the skills to test ideas for them selves.

Take an hklf4 file, reject all reflections

where $h+k+l = 5n$

and where I/σ(I) is less than 5.0

and output a new, properly formatted, hklf4 file

# Summary

The early computing schools documented some of the internal workings of programs and details not reported in journals.

Technical details published in journals are dispersed and un-catalogued.

Programs dedicated to a single task are most easily understood by users.

However, productivity can be increased by grouping tasks together into a system.

Documentation for both users and programmers becomes an exponential problem for large systems.

The cost of maintenance usually means that such systems are difficult to support in the long term

In collaborative projects, programming anarchy (sometimes called creativity) can turn the whole enterprise into a house of cards.

Users don’t like change.

Users don’t like choice.