dials

Diffraction geometry parameterisation and refinement

David Waterman
CCP4

DIALS-3 @ LBNL

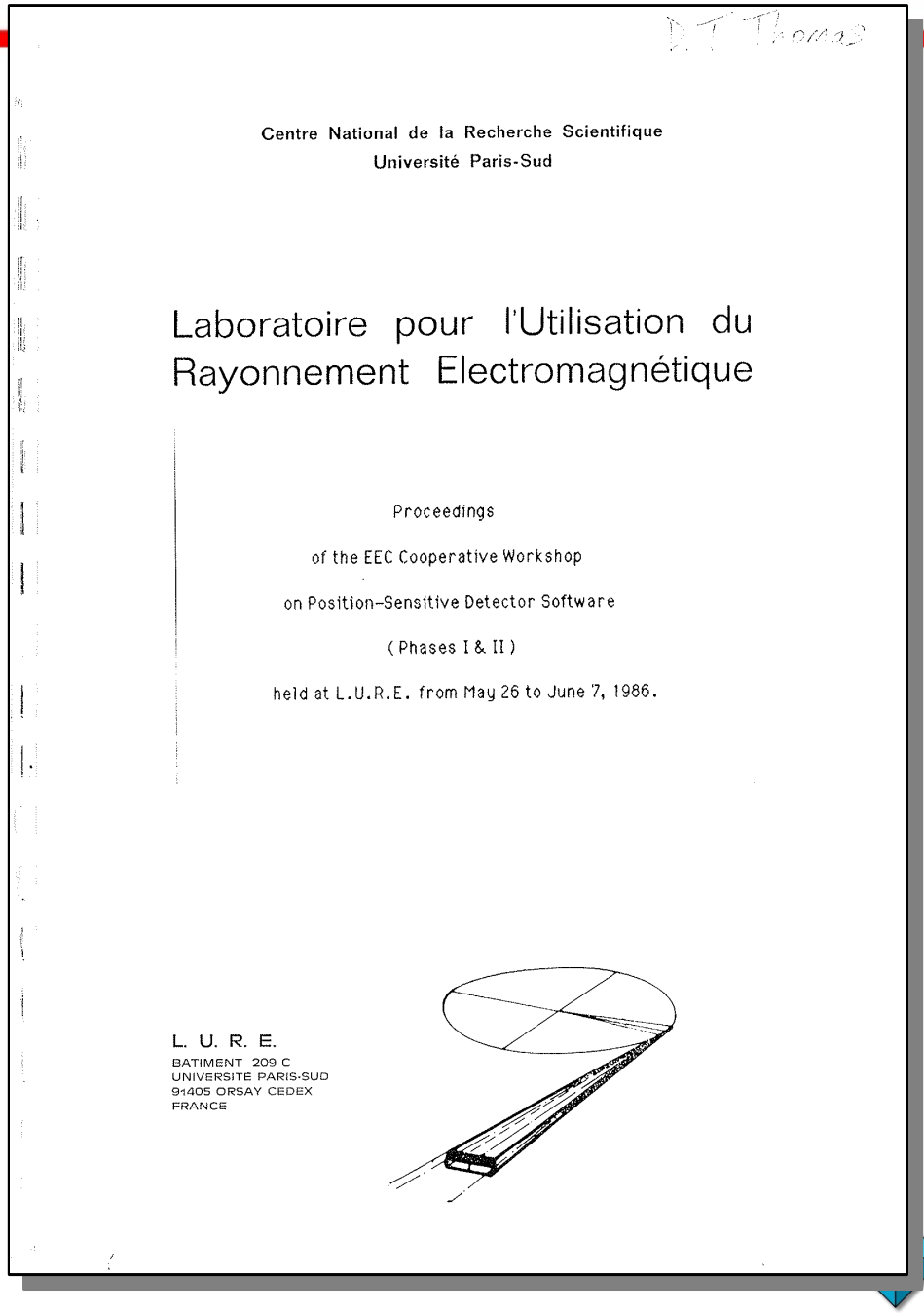
11 February 2013

CCP4 diamond

Motivation for global refinement

- See Graeme's slides for overview
- Input diffraction spot indices, their centroids and estimated uncertainties (h, k, l; X, Y, ϕ; σ_X, σ_Y, σ_ϕ)
- Use all (useful) data available to refine a model to reduce rmsd of predicted centroids
- Global refinement helps to recover from poorly defined parameters in local ϕ window
- More physically meaningful: avoids mopping up of effects by correlated parameters and therefore obtains realistic parameter values
- Refine profile parameters separately
- Potential second round with improved centroid observations

Lessons from LURE



D.T. Thomas

Centre National de la Recherche Scientifique
Université Paris-Sud

Laboratoire pour l'Utilisation du Rayonnement Electromagnétique

Proceedings

of the EEC Cooperative Workshop

on Position-Sensitive Detector Software

(Phases I & II)

held at L.U.R.E. from May 26 to June 7, 1986.

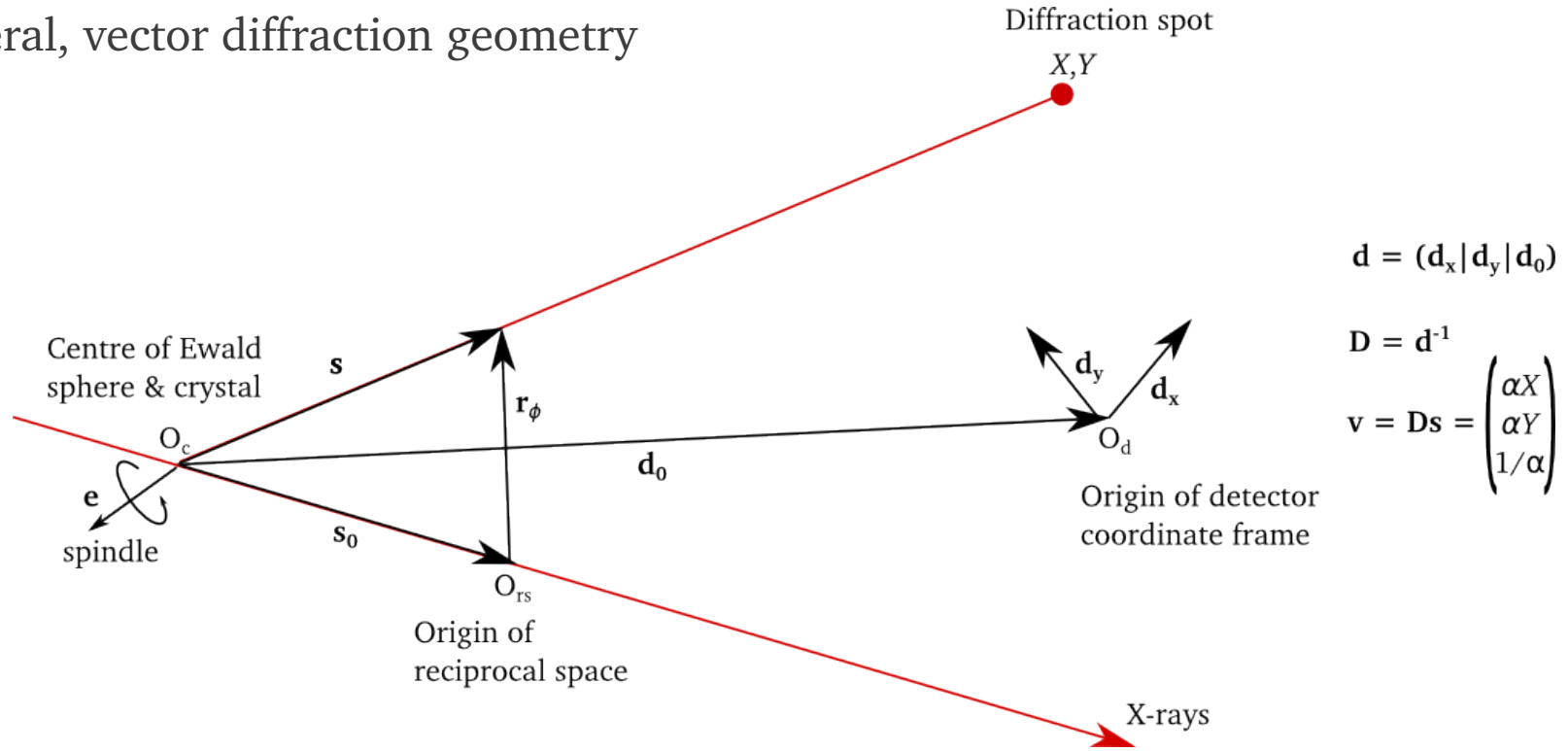
L. U. R. E.
BATIMENT 209 C
UNIVERSITE PARIS-SUD
91405 ORSAY CEDEX
FRANCE

Lessons from LURE

- The description of diffraction geometry should be generalised and abstracted
- Do not depend on arbitrary reference frame conventions or hardware-specific features
- Parameterisation should be modular and extensible
- This is all 'obvious', but is easier to do from the ground up, with the toolbox approach

Vectorial reflection prediction

- General, vector diffraction geometry



- The detector abstract frame is a hardware-independent adapter
- Positional corrections can be accounted for in the mm-to-px mapping function

Vectorial reflection prediction

For refinement we want at least the first derivatives of predicted centroids

$$\frac{\partial \phi}{\partial p} = -\frac{\frac{\partial \mathbf{r}_\phi}{\partial p} \cdot \mathbf{s} + \mathbf{r}_\phi \cdot \frac{\partial \mathbf{s_0}}{\partial p}}{(\mathbf{e} \times \mathbf{r}_\phi) \cdot \mathbf{s_0}}$$

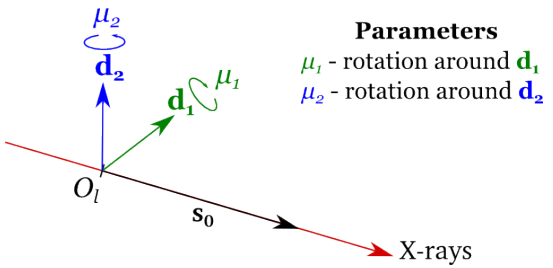
$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}p} = -\mathbf{D}\frac{\partial \mathbf{d}}{\partial p}\mathbf{v} + \mathbf{D}\left[\frac{\partial \mathbf{r}_\phi}{\partial p} + (\mathbf{e} \times \mathbf{r}_\phi)\frac{\partial \phi}{\partial p} + \frac{\partial \mathbf{s_0}}{\partial p}\right]$$

Neatly, these are factored into independent models

$\frac{\partial \mathbf{d}}{\partial p}$	$\frac{\partial \mathbf{r}_\phi}{\partial p}$	$\frac{\partial \mathbf{s_0}}{\partial p}$
detector	crystal	beam direction

Parameterisation

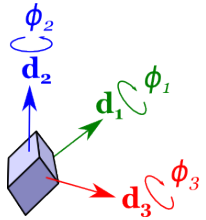
Each model parameterisation provides ∂[state]/∂p



Parameters
μ_1 - rotation around $\mathbf{d}_1$
μ_2 - rotation around $\mathbf{d}_2$

$\frac{\partial \mathbf{s_0}}{\partial p}$

beam direction



Parameters
ϕ_1 - rotation around $\mathbf{d}_1$
ϕ_2 - rotation around $\mathbf{d}_2$
ϕ_3 - rotation around $\mathbf{d}_3$
$a, b, c, \alpha, \beta, \gamma$ - unit cell

$\frac{\partial \mathbf{U}}{\partial p}$

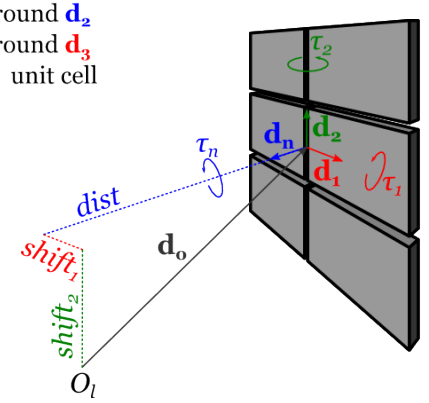
crystal orientation

$\frac{\partial \mathbf{B}}{\partial p}$

crystal unit cell

$\frac{\partial \mathbf{d}}{\partial p}$

detector position and orientation

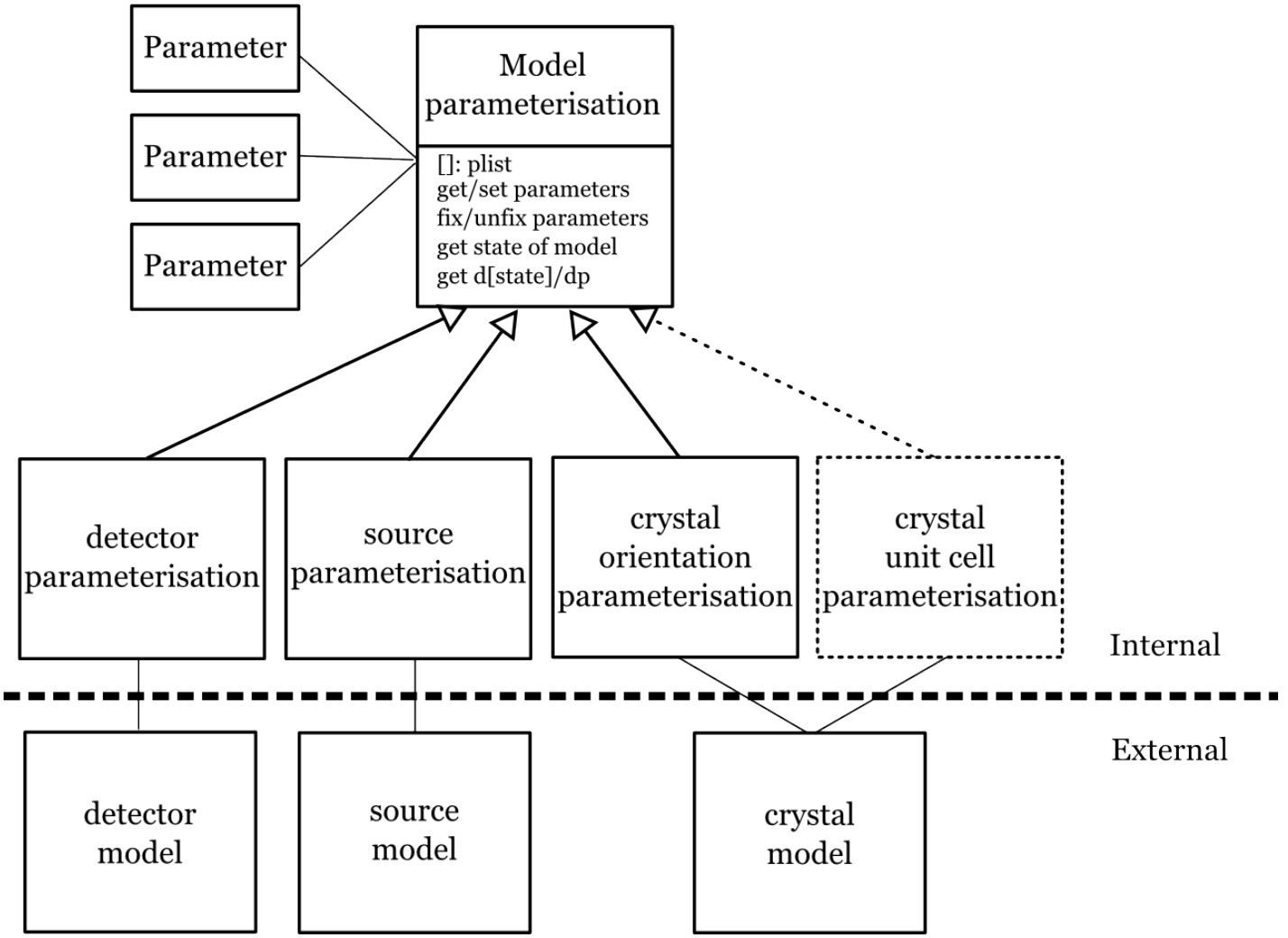


Parameters
dist - detector distance along $\mathbf{d}_n$
$shift_1$ - translation in direction $\mathbf{d}_1$
$shift_2$ - translation in direction $\mathbf{d}_2$
τ_n - rotation around $\mathbf{d}_n$ ≅ "roll"
τ_1 - rotation around $\mathbf{d}_1$ ≅"tilt"
τ_2 - rotation around $\mathbf{d}_2$ ≅"twist"

- Separate 'parameterisation of prediction equation' object takes ∂[state]/∂p for each model and converts to derivatives of *X*, *Y*, ϕ for each reflection
- Individual model parameterisations are encapsulated

Design

Parameterisation of experimental models



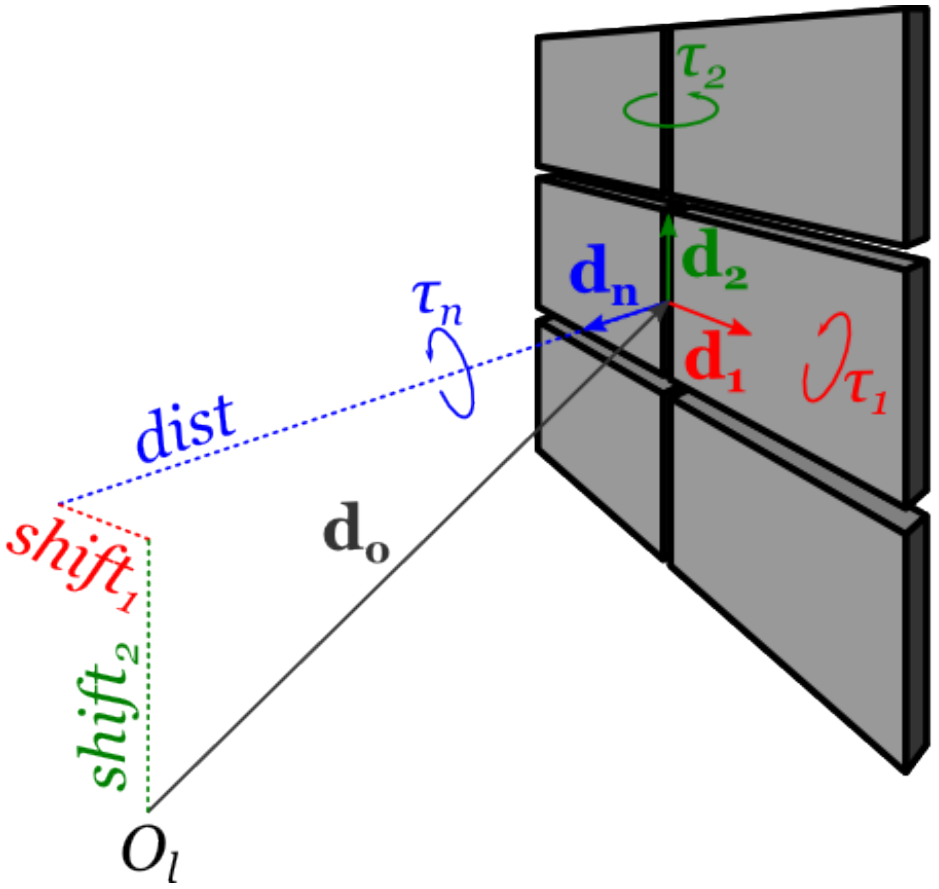
Parameterisation

The abstract interface specifies that:

- Model parameterisations are initialised with an initial state of the model
- New states are composed by the action of functions of the parameters on the initial state
- A state and its derivatives are either a vector or a matrix
- The parameters are either distances or angles with associated unit directions

Parameterisation

- A concrete example: detector parameterisation



Parameters

dist - detector distance along $\mathbf{d_n}$

$shift_1$ - translation in direction $\mathbf{d_1}$

$shift_2$ - translation in direction $\mathbf{d_2}$

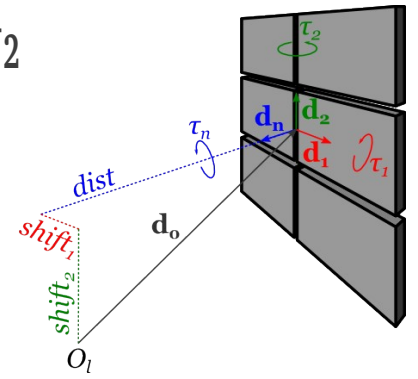
τ_n - rotation around $\mathbf{d_n}$ ≅ "roll"

τ_1 - rotation around $\mathbf{d_1}$ ≅"tilt"

τ_2 - rotation around $\mathbf{d_2}$ ≅"twist"

Parameterisation

- Initial sensor matrix provides $\mathbf{d}_0$, $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_n$
- Translation parameters are immediately *dist* along $\mathbf{d}_n$ and $shift_1$, $shift_2$ along $\mathbf{d}_1$, $\mathbf{d}_2$
- Initial rotation angles all 0.0, around axes $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_n$



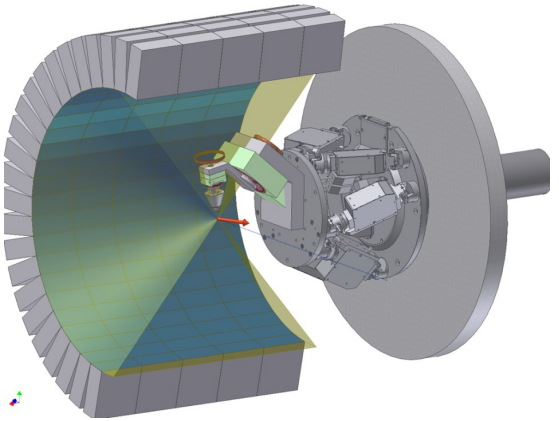
Parameterisation

- Accommodates refinement of multi-tile detectors as one rigid unit
- Each sensor panel k has its own matrix $\mathbf{d}^k = (\mathbf{d}_x^k | \mathbf{d}_y^k | \mathbf{d}_0^k)$
- These vectors are linear combinations of $\mathbf{d}_0$ and the local coordinate system $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_n$ that moves with the detector:

$$\mathbf{d}_x^k = \alpha_1^k \mathbf{d}_1 + \alpha_2^k \mathbf{d}_2 + \alpha_3^k \mathbf{d}_n$$

$$\mathbf{d}_y^k = \beta_1^k \mathbf{d}_1 + \beta_2^k \mathbf{d}_2 + \beta_3^k \mathbf{d}_n$$

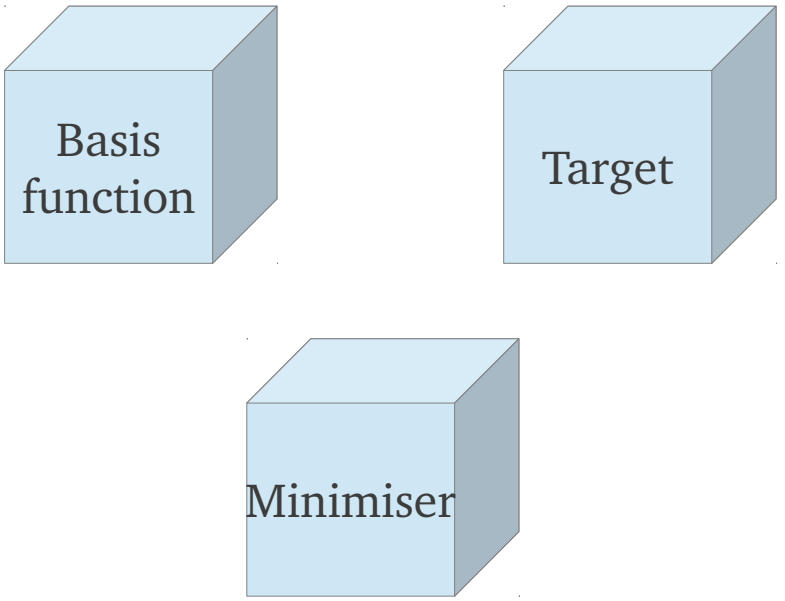
$$\mathbf{d}_0^k = \mathbf{d}_0 + \gamma_1^k \mathbf{d}_1 + \gamma_2^k \mathbf{d}_2 + \gamma_3^k \mathbf{d}_n$$



- Thus the derivatives $\partial \mathbf{d}^k / \partial p$ for each sensor are easily calculated by linear combinations of $\partial \mathbf{d}_o / \partial p$, $\partial \mathbf{d}_1 / \partial p$, $\partial \mathbf{d}_2 / \partial p$ and $\partial \mathbf{d}_n / \partial p$

Design

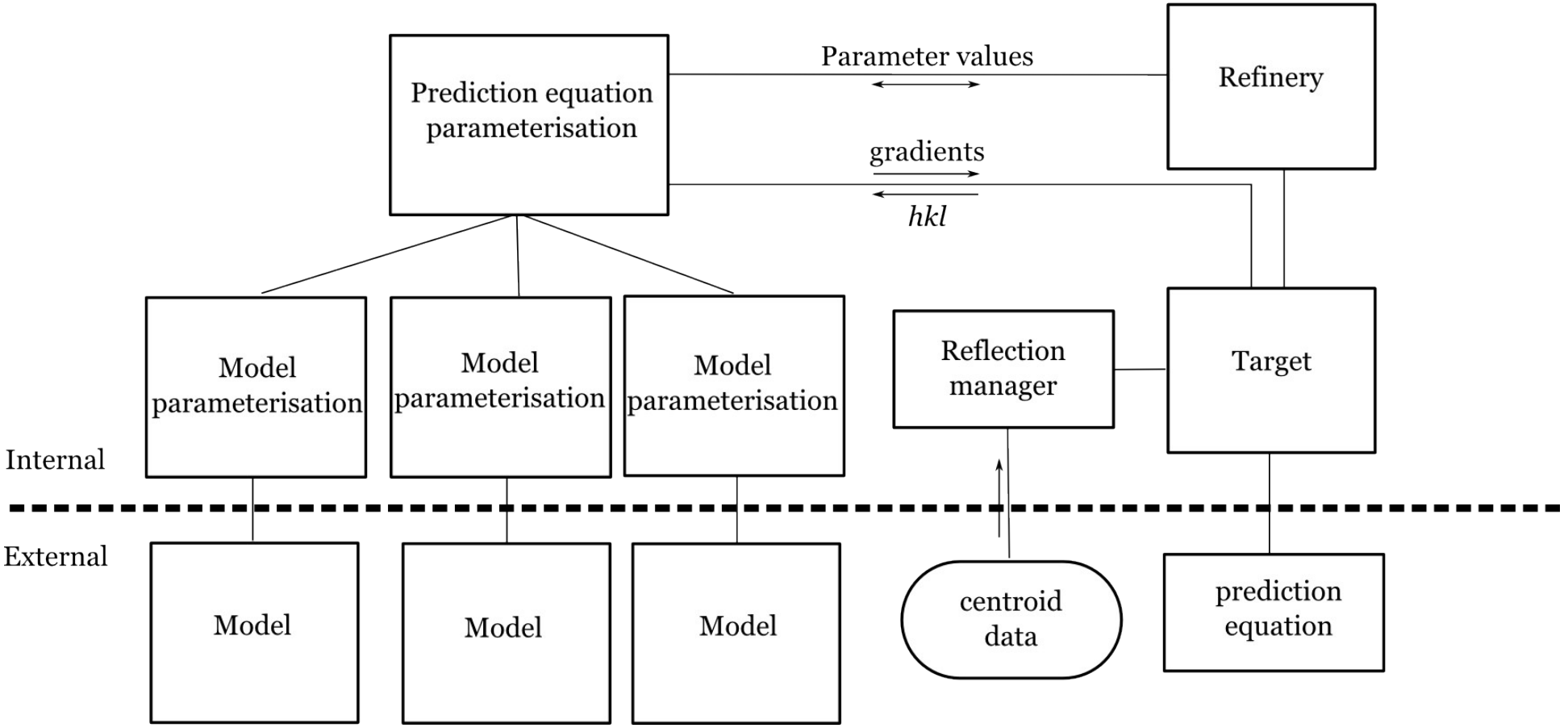
- Further encapsulation within refinement module



- Make these independent (where possible)

Design

Overview



Target function

- Least squares target, normalised by the number of observed-predicted pairs currently in play
- The reflections included in refinement are controlled by a reflection manager object

$$L = \frac{1}{2\mathrm{n_h}} \sum_{\mathrm{h}} w_{x,h}(X_c - X_o)^2 + w_{y,h}(Y_c - Y_o)^2 + w_{\phi,h}(\phi_c - \phi_o)^2$$

$$\frac{dL}{dp} = \frac{1}{n_{\mathrm{h}}} \sum_{\mathrm{h}} w_{x,h}(X_c - X_o)\frac{dX_c}{dp} + w_{y,h}(Y_c - Y_o)\frac{dY_c}{dp} + w_{\phi,h}(\phi_c - \phi_o)\frac{d\phi_c}{dp}$$

- First order approximation to the curvatures

$$\frac{d^2L}{dp^2} \approx \frac{1}{n_{\mathrm{h}}} \sum_{\mathrm{h}} w_{x,h}\left(\frac{dX_c}{dp}\right)^2 + w_{y,h}\left(\frac{dY_c}{dp}\right)^2 + w_{\phi,h}\left(\frac{d\phi_c}{dp}\right)^2$$

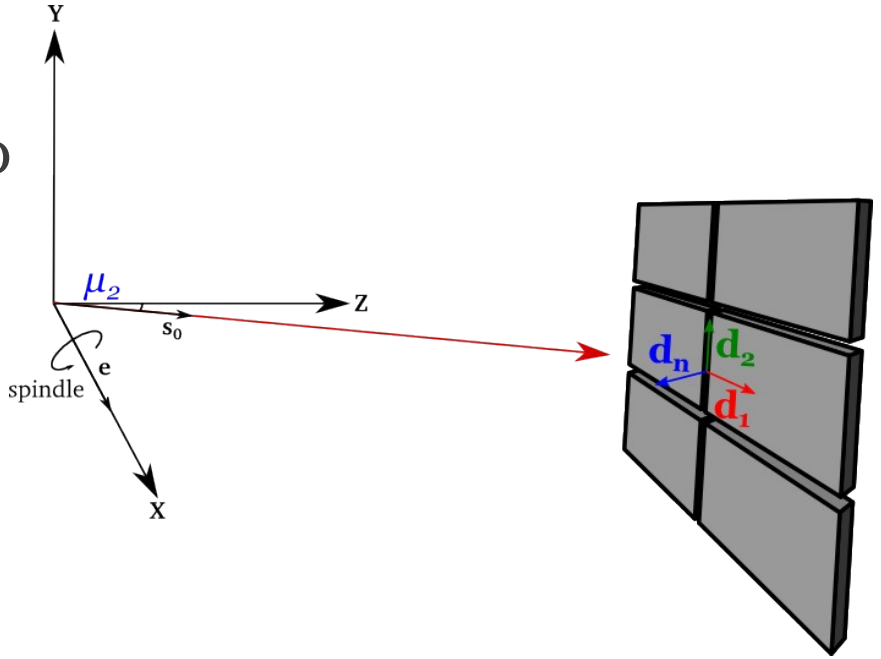
Minimiser

- So far, only tried scitbx.lbfgs treating it as a “black box”
- Two modes: 1. simple. 2. with curvatures
- Strongly sensitive to parameter scales!

Initial test case

Protocol

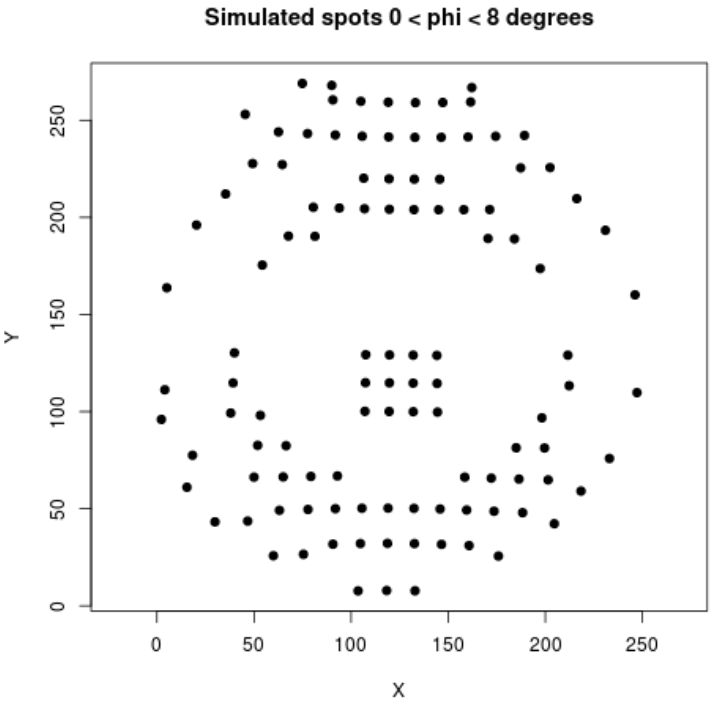
- Generate random geometry, close to an ideal (imgCIF frame)
- Generate a small, random, P1 cell (19.6, 16.7, 15.8, 90.2, 90.2, 89.5)
- Select wavelength (here 1.18 Å)
- Pilatus 2M sized flat panel, ~200 mm from the crystal



- Parameters upset by small amounts from initial values
  - Detector translations shifted 1 mm
  - All rotations (3 detector, 1 beam, 3 crystal) shifted 2 mrad
  - Unit cell fixed and perfect

Initial test case

- For 180° sweep, 2 Å resolution limit, ~2.5K 'observations' generated by reflection prediction code
- Parameters reset to initial values
- Four runs of refinement attempted



Number of steps to convergence		
	simple	*curvatures*
mm, rad scale	148	"Line search failed" after 3
mm, mrad scale	15	11

Issues

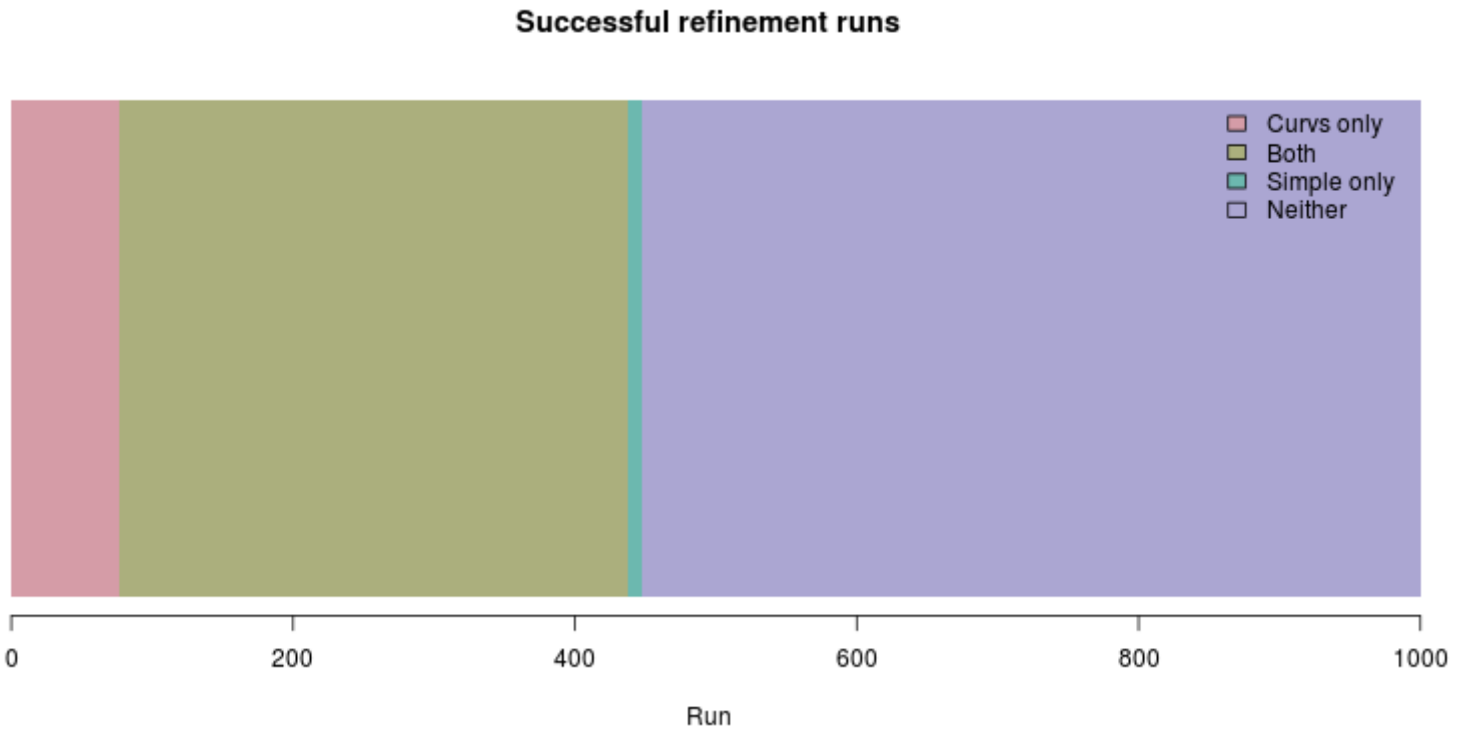
- Initial tests raised a number of problems
  - Original parameter scaling very poor. Problems with line search
  - Tried approximate curvatures to improve performance
  - Likely highly correlated parameters (→ large off-diagonal elts of normal matrix, requires investigation)
  - The problem is inherently bounded. Better to have a minimiser that accepts bounds
  - The number of parameters is small, and the target is least squares. More appropriate algorithms than L-BFGS exist (→ Gauss-Newton, Levenberg-Marquardt)
  - No time-dependent parameterisation attempted yet; left until other problems are fixed first

Exploring parameter space

- Investigate convergence properties over 1000 tests
- Similar protocol as before:
  - Near imgCIF ideal geometry, small P1 cell (11.3, 16.5, 18.4, 89.6, 90.3, 90.1), beam 0.89 Å. Pilatus 2M sized flat panel ~200 mm away
  - Parameter shifts by normal deviates; σ=2.0 mm each translation, σ=4 mrad each rotation angle
  - 180° sweep, 2 Å resolution limit, ~1.6K 'observations' generated by reflection prediction code
  - Parameters reset to initial values
  - Refinement attempted to acceptable rmsd

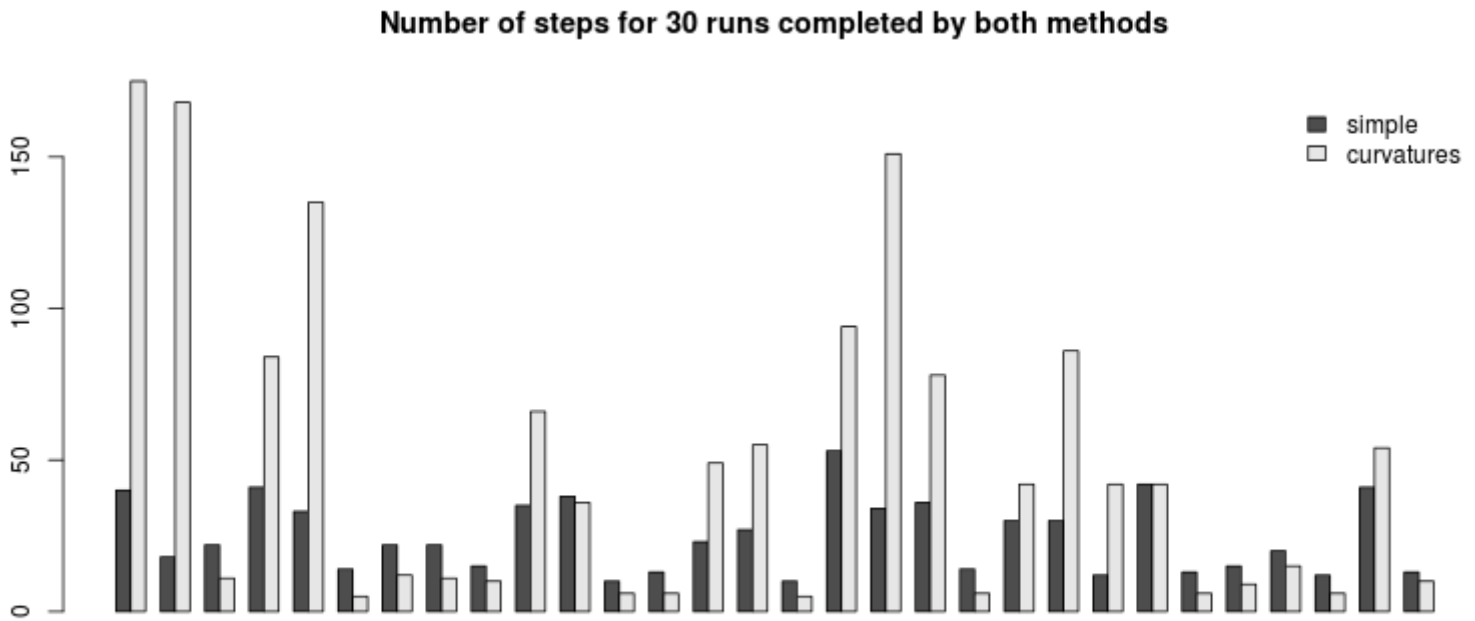
Exploring parameter space

- Results for 1000 attempted runs



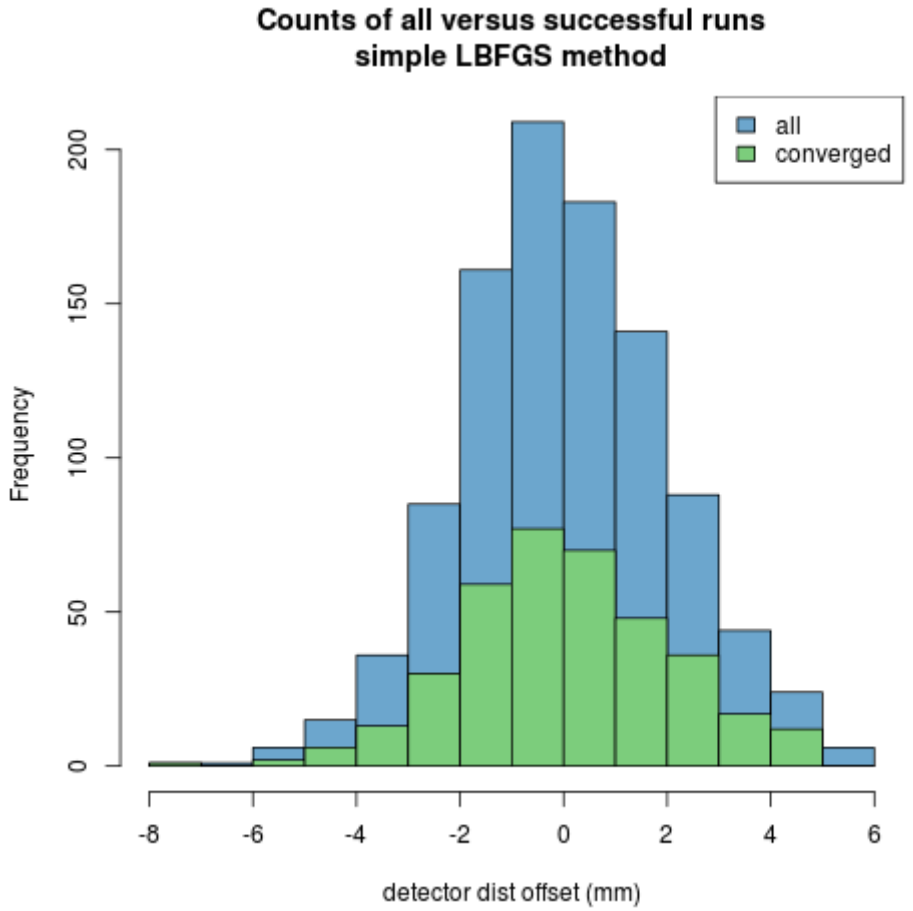
Exploring parameter space

- Using curvatures is surprisingly (?) less efficient for the more difficult runs



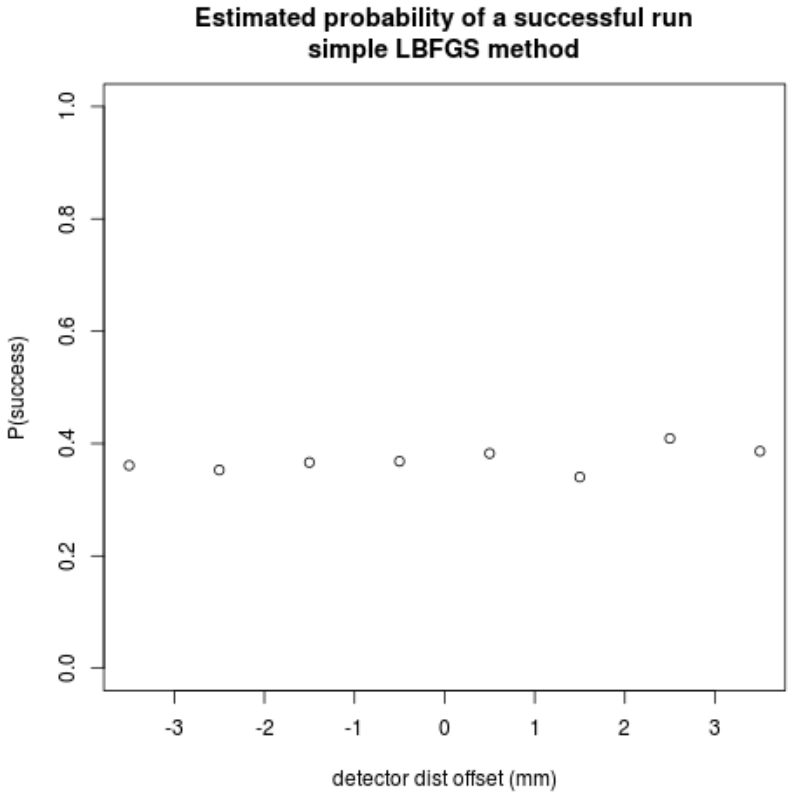
Convergence radius

- Compare all runs with those successful, one parameter at a time
- Only looking at results for the “simple L-BFGS” method



Convergence radius

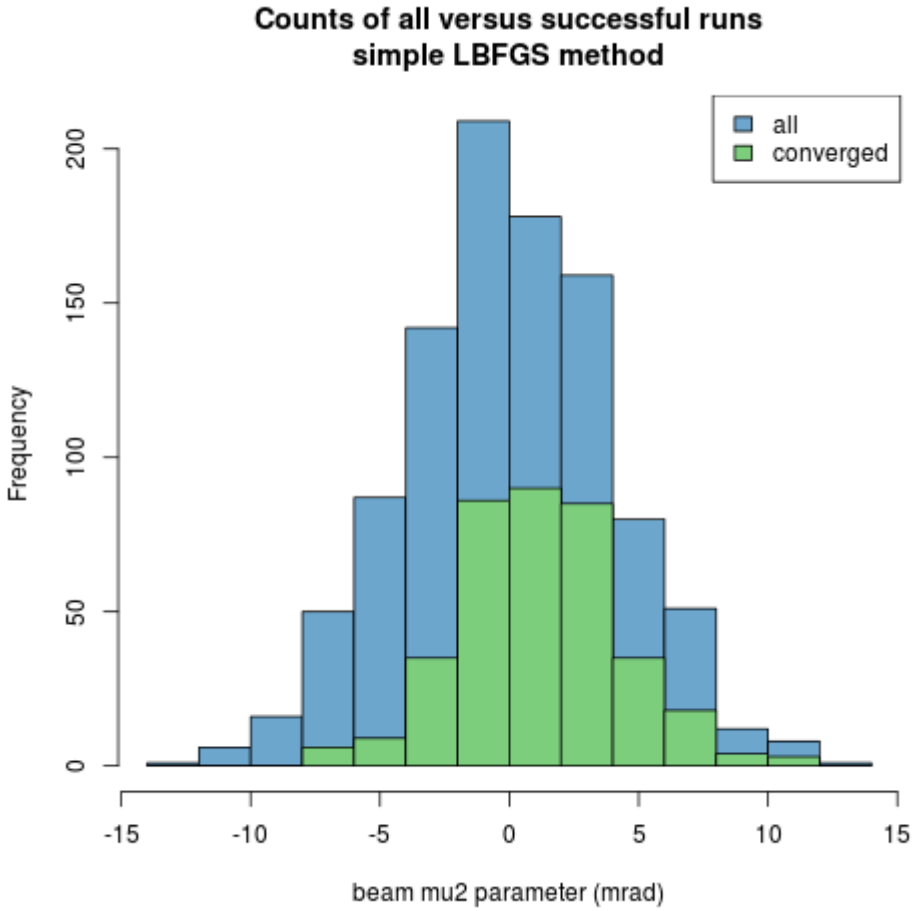
- Compare all runs with those successful, one parameter at a time



- Did not appear to find a failure point for translation parameters

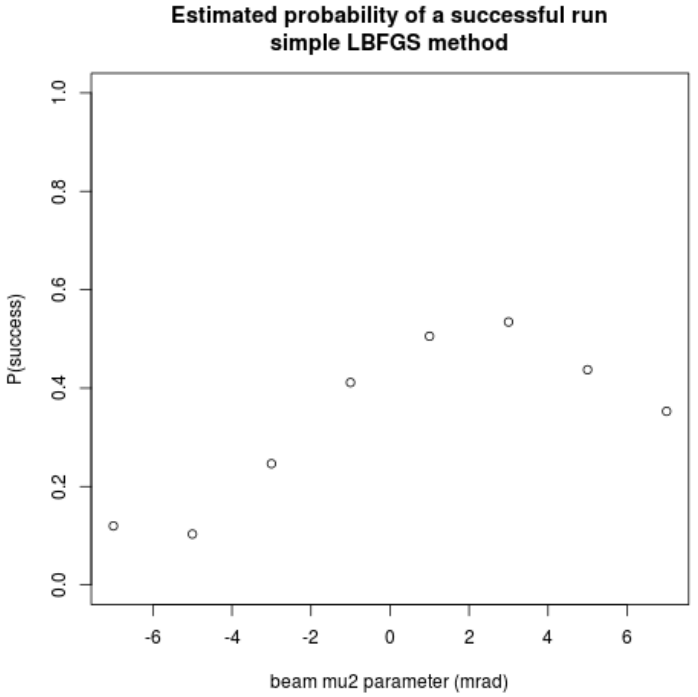
Convergence radius

- Beam inclination showed most sensitivity



Convergence radius

- Beam inclination showed most sensitivity



- Not sure why there is a shift of peak success away from the origin
- This investigation is difficult! Is there an established method?

Next steps

- Include crystal unit cell parameterisation
- Explore different (LS-specific) minimisation engines
- Test against real data
- Time-dependent parameterisation for the crystal; mix global/time-varying refinement
- Multi-sensor detector
- Faster implementation: C++ and parallelism

Interim conclusions

- The basic goal of approaching diffraction geometry refinement in a general and modular way is feasible
- Despite some problems with use of the minimiser, the synthetic test cases show promise
- Global refinement with time-varying crystal parameters seems to be an achievable goal
- The “learning by doing” approach breaks the problem down into smaller achievable goals. We are learning much about the nature of the components and interfaces as we go!

Acknowledgements

- The CCP4 core team for my time
- Invaluable contributions from Andrew Leslie at the start
- Continuing assistance from a panel of experts: Andrew Leslie, Garib Murshudov, Phil Evans (MRC/LMB), Gleb Bourenkov (EMBL)
- DIALS-East and DIALS-West for pooling resources
- The cctbx community for openness and responsiveness (which is even better than documentation)

dials

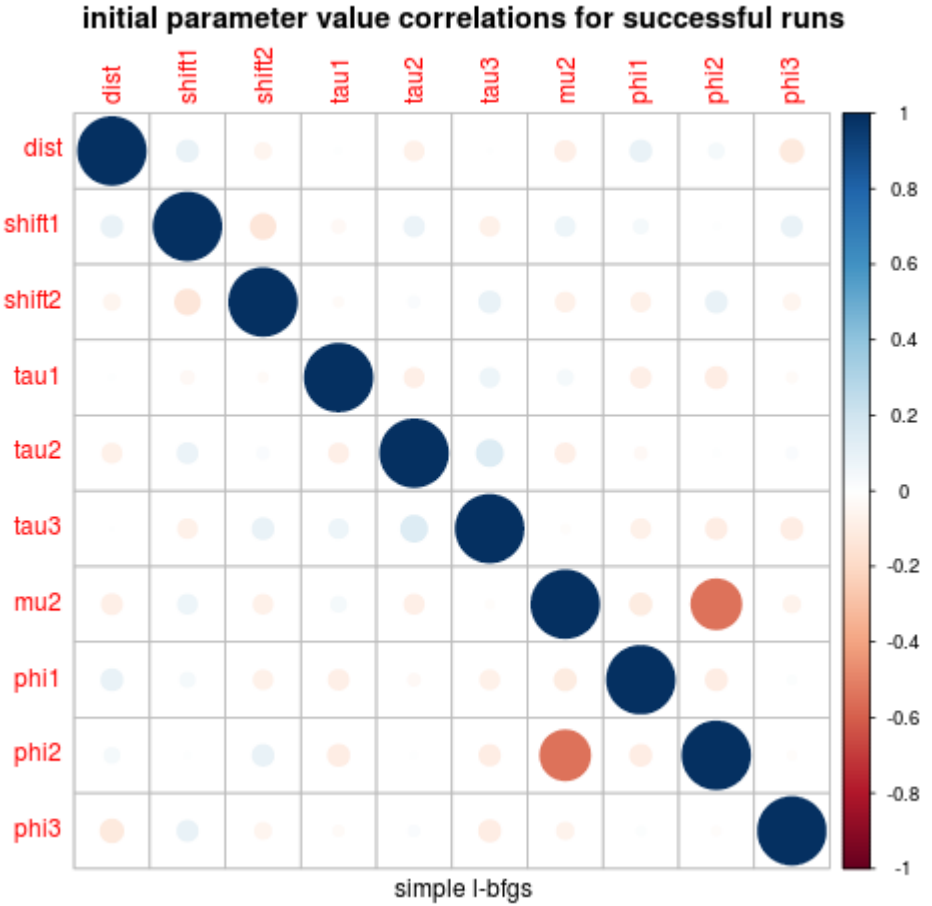
CCP4 diamond

Extra slides

Radius of convergence

simple L-BFGS method

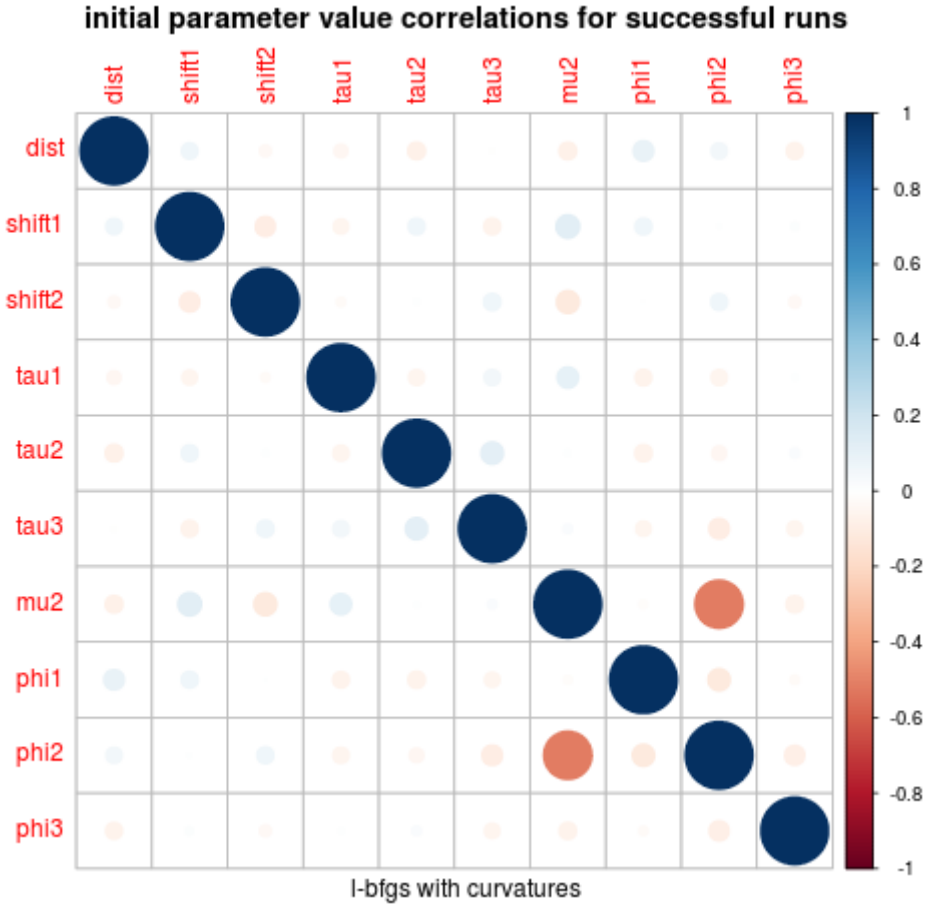
Parameter	σ_{all}	σ_{conv}	$\sigma_{conv}/\sigma_{all}$
dist	0.2	0.2	0.99
shift1	0.2	0.21	1.05
shift2	0.2	0.19	0.97
tau1	3.9	3.59	0.92
tau2	3.8	3.58	0.94
tau3	3.91	3.85	0.99
mu2	4.09	3.25	0.8
phi1	4.01	3.34	0.83
phi2	4.03	3.15	0.78
phi3	4.01	3.74	0.93



Radius of convergence

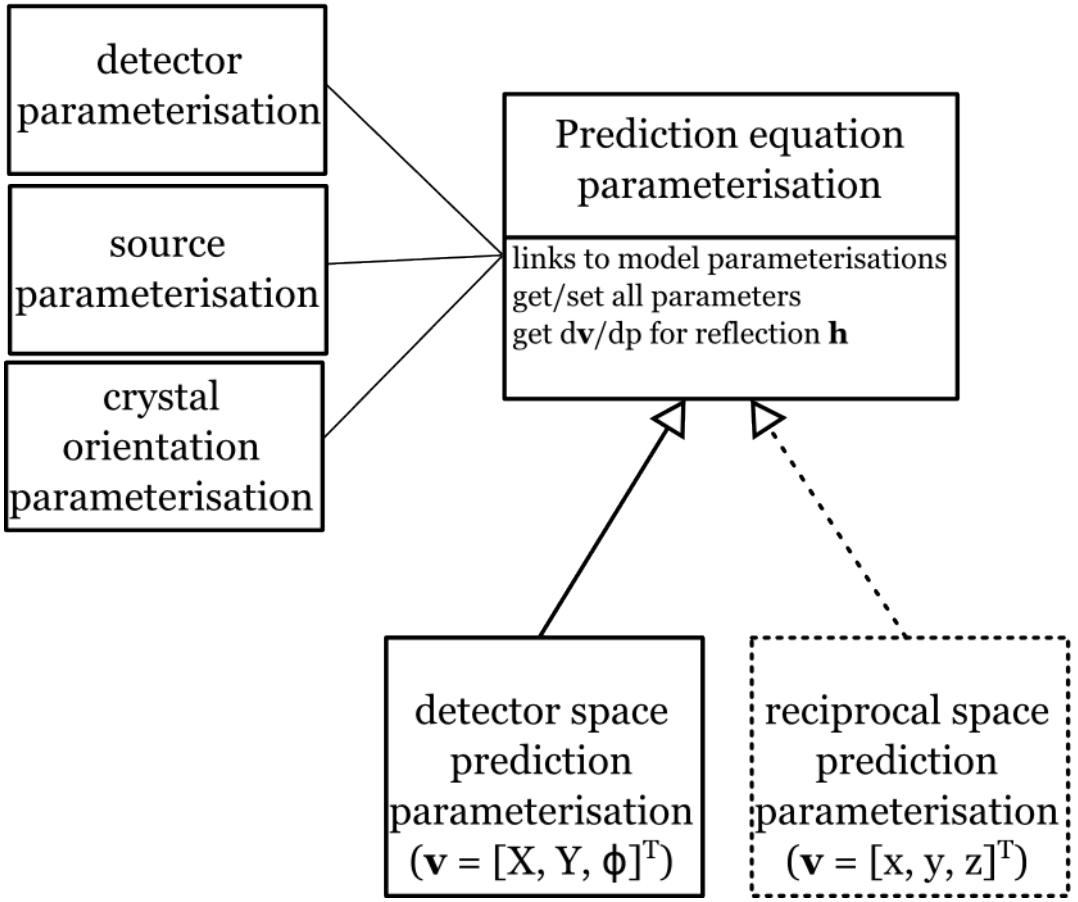
L-BFGS with curvatures

Parameter	σ_{all}	σ_{conv}	$\sigma_{conv}/\sigma_{all}$
dist	0.2	0.2	0.99
shift1	0.2	0.2	1.03
shift2	0.2	0.19	0.97
tau1	3.9	3.84	0.99
tau2	3.8	3.67	0.97
tau3	3.91	3.74	0.96
mu2	4.09	3.44	0.84
phi1	4.01	3.33	0.83
phi2	4.03	3.21	0.8
phi3	4.01	3.96	0.99



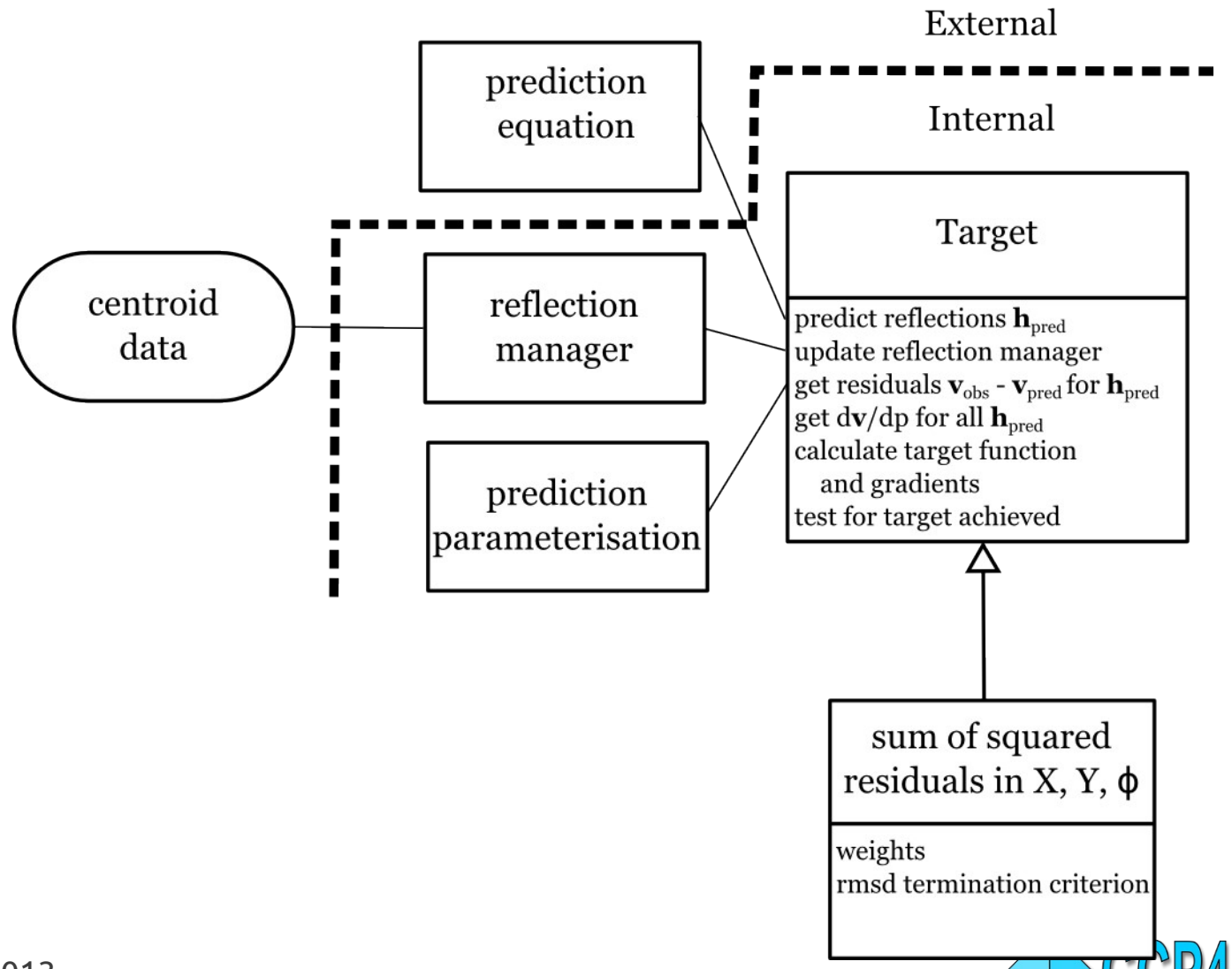
Design

Parameterisation of the prediction equation



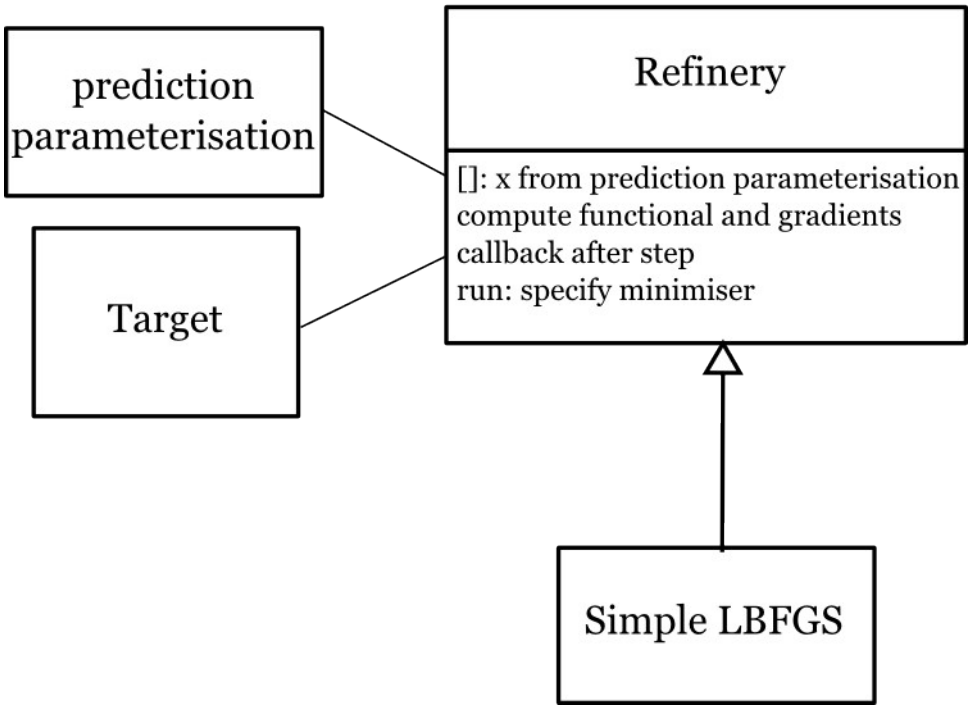
Design

Data handling and target function



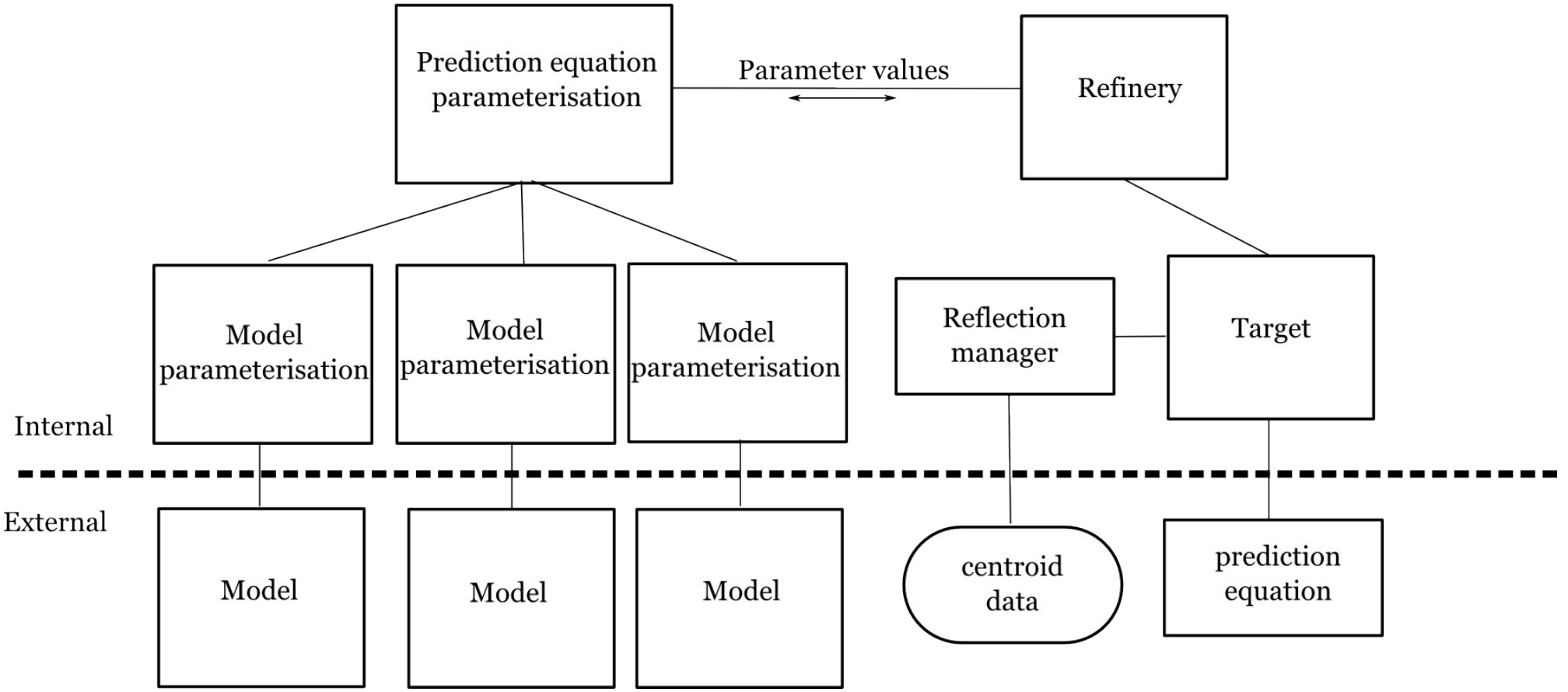
Design

Refinement engine



Design

Overview



Problems

- “Line search failed” error.

```
lbfgs minimizer step
lbfgs minimizer.run(): f=0.511684, |g|=9.81135, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0467172, |g|=42.5567, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0464431, |g|=42.0279, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.514731, |g|=20.3204, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0463982, |g|=41.9407, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0463332, |g|=41.8142, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.518362, |g|=30.9968, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0463223, |g|=41.793, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0463064, |g|=41.762, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.520692, |g|=36.469, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0463037, |g|=41.7568, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0462998, |g|=41.7491, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.521985, |g|=39.2191, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0462991, |g|=41.7478, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0462981, |g|=41.7459, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0456016, |g|=40.3669, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0451537, |g|=39.458, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0450036, |g|=39.1493, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.522011, |g|=39.2732, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer.run(): f=0.0450036, |g|=39.1493, x_min=-18.0174, x_mean=31.6191, x_max=199.999
lbfgs minimizer exception: lbfgs error: Line search failed: Rounding errors prevent further progress.
There may not be a step which satisfies the sufficient decrease and curvature conditions. Tolerances
may be too small.
lbfgs minimizer done.
```

Problems

- “Line search failed”