Fitting Notes

IFittingAlgorithm.h
↳ base class for main fitting algs
↳ take ws and function then creates domain - handles properties
This creates separation to types of workspaces

Domains:
fitting alg creates 'Domain creators' which creates the domain once they have all the correct info
↳ this is because we need to know ws + function types to know properties to add but should not create domain (ex values) until later
→ created when cost func initalised

Most common domain is FitMW.h
↳ used for multi ws + standard function

4 methods in domain need override
→ declare dataset properties:
declares additional props specific to dataset e.g. ws index, table or columns

→ create domain : creates domain + values for the given workspace
↳ for fitMW handles weights + normalisation here
weights gererally 1/error unless set elsewhere
(note: if error is 0 or ~~negative~~ infinite, or nan etc ~~or~~ then weight=0)
if excluding data, weight of those y values is 0.

→ create Output Worksapce : creates all the ~~of~~ output workspace for domain

→ initFunction : initalizes function w. workespace, should call setWorkspace / setMatrix workspace on func

Minimizer then does main fitting
(note: ~~I~~FittingAlgorithm just calls exec Concrete which is overriden in Fit.cpp)

Minimizers:
Levenberg Marquardt is main one
, 2 variants → one ours, one GSL
MD version

, Most minimizers from GSL
↳ google GSL multidimensional minimizer for details

- → In MD case, uses normal equation, reduces jacobin to smaller
- works for any cost func (non MD version can't be used w. Poisson)
- used for v. large datasets

library optimizers - (functions integrate differently
- require cost func as a vector
library will square sum
pass 1 Cost Func a library of functions
- cost Func with minimizer don separately - should be combined in future

GSLFunction.h / .cpp :
require vector x, params and function
- file has docs to say what ↑ are.
→ only calculate values + derivs of params

Minimizer only knows about costFunc not function.

Cost Function :
- function of parameters
- usually least square (chi squared)

- least square unweighed
↳ not using errors
↳ errors could be in data
↳ used for standard data

- RWP, if exp data poisson
- tries to simulate poisson

(branch unmerged for poisson fitting
↑ now merged + docs added
(ask Anthony for details)

ICostFunction.h
interface for cost function
calculates values and derivatives
+ second derivatives (Hessian)
↳ inherited by CostFuncFitting which implements val, deriv
In principle can do multidimensional fit but no one does

cost func → handles constraints
- adds penalty if outside constraint
(not lagrange multipliers)
- add quadratic penalty to costfunc for boundary constraints
handled differently in GSL to ICostFunc

In CostFuncFitting.cpp when calc val Deriv Hessian it gets constraint from function,
check() returns value if outside constraints or 0
problems - could not be accurate, can be outside boundary (but should be within error)
- fixed penalty factor if outside region

(lagrange multipliers
constraint might not be quadratic
→ not necessarily same minimum)

GSL get constraints:

add penalty to every 10th point
as GSL does not give cost function
- can't add to every point as
that would just shift data

ROOT Minimizers
inherited property so can have properties like accuracy, stop condition
↳ so many as uncertain or best

LM more accurate + stable
(would be useful if combined)
use noreLM if can work

TrustRegion - ~~Cobra~~ scientific computing made it, fit successfully in more cases
↳ fortran code

Anders compares them can give which is best

LM - timing, iterations

Damped Gauss Newton - fastest

Deas minimizer - gradient ones

Simplex - more well known

FUMILI basian minimizer
builds probability distribution

FABADA → better error to true
lower cost func → high distribution
basiaian distribution don't
need to try all values
doesn't need whole space
knows distribion of solution
explicitly, (dismibution with multi
calculate exact shape of dismibution) (variant)
LM give dismibution ←

FABADA → indirect
→ sangamehra understands
→ unknown how it works
→ has its own output ws
→ more complecated.

To create new
initalize
finalize → mainly used by FABADA
iterate → Minimum you have
to do

~~

mimize → so minimizer can be
used outside fit

IFunction
large interface
→ given domain will calc values

attributes - file name

file starts with attributes
↳ non-fitting parameter
wrapper around boostvariant
↳ types can be string, int, double, bool, vector
done so if someone as new type it won't compile
To access values in variant need visitor overloads function to find which type the attribute is
Templated setAttributeValue there so should need changing
(special case for string)

Complexity so that it is all consistent

(could have used algorithm property type but no GUI implementation
anyone implementate but no interface)
attributes
types are limited

IFunction

setworkspace will set inital values (discussed before)

Initialize() so users don't need to construct definitions etc (not used much as users don't make own very much)

progress reporter - if it takes long time

function - calculates values

funcDeriv - calcs derivative over func parameters

parameters - accessing by index quicker than name

`isExplicitlySet` - flag whether set by user or default

Fixing parameters - may fix resonable values if not enough in ws fit basically ignores

Active parameters:
- the ones cost func will use
- when not fixed or when different to declared
- used in Gaussian: $1/\sigma^2$

$$\frac{1}{2}\frac{(y-c)^2}{\sigma^2} = \frac{1}{2}s(x-c)^2$$

↑ works better

- polynomial, far from 0, fit ~~as~~ unstable (could replace $a+bx+cx^2+\ldots$ to smaller values for a, b, c would use active parameters)

Ties - similar to fix
- ties parameters to value of others
e.g. tie height = $1-A_0$
height is inactive, calculated using $1-A_0$
- if $H=2$ it is fixed

Sort Ties → checks they are applied in correct order - should apply automatically

crystalFit - getValuesSize
MD function - create EquivalentFun

Child function - moved from composite function to make life easier - no casting needed

getNumberDomains - moved from multidomenfunction
↳ fit only uses to determine if MD function

calNumericalDeriv - if function contains ties or cannot be done directly it is done this way

setChiSquared - value of cost fune

setHandler used in fitFunctionBrowser each function gets handler

ParameterStatus - describes parameter

storeReadonly - crystal fit

Algorithm property must be able to write ToString
- compliacated when adding new stuff

No parameters in IFunction

IFunction combines 2 interfaces
- one that calculates stuff
- one that deals with parameters (example IFunctionLocation)
- no issues with this yet

↳ IFunctionID, IFunctionMD
↙
FunctionID - makes domain simpler
→ uses raw pointers
↳ slightly dangerous but quicker
- not called by users

ParamFunction - has its own parameter
CompositeFunction - don't own params (may come from member function)

CompositeFunction - add them together

others - product function, Convolution, Multidomain, immutable

immutablecomposite - way to define to look as as a simple functio

when outputted as string params have prefix fo - , f1 - , etc (some rename them to be different children of immutable

- Will work for any domain

Jacobians
- number of data by parameter
- derivate of a value with respect
- cannot use matrix class
- used in constraint value calculation
- Partial Jacobian ~~[illegible]~~ is partial part of jacobian
  uses pointer of original jacobian

Composite:: funcDeriv
getAttr("NumDeriv")
if all functions are analytics with ties it needs this

Doesn't in theory need to be there but causes errors if not.
Often in warning messages
May now be possible to remove attribute "NumDeriv"

Check Function - takes apart + adds back together
- make sure its in correct

When functions are added or removes & tie names change
- removed then added
- could be improved

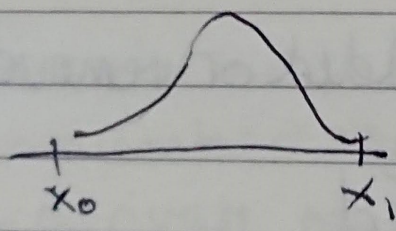
Covolution:
composite but must be 1D
2 members - resolution fuction
so third call creates additional composite in 2nd slot

- res
- Gauss
- Gauss etc

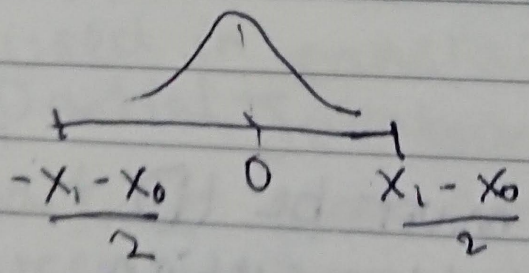
Convolution
Checks if regular can use FFT
if not do directly
FFT in GSL
When delta function add resolution
only in 2nd convolution

Cant have convolution of convolution
but might work

"Fix Resolution"
Resolution usually fixed from file but
Sometimes not, by default fix
resolution set to true



model calculated in this region



resolution in this range
peak should be around 0

Resolution.h ← most commonly used for resolution
uses TabulatedFunction
↳ tabulates between predefined point
iterates
→ can set X and Y or gets from workspace
→ doesnt have fitting parameters

ProductFunction:
Similar to convolution but with multiply

MultiDomainFunction:
type of composite

In principle all can apply to multiple domains but composite only so far

create
add member functions
assign which domain is used by which function → using set domainIndex/ indices

To use with factory

for string:
"composite = CompositeFunction, Attribute",
name = ...,

(look in MultiDomFuncTest.)
↳ lots of examples of strings
domain has $ prefix
means attribute of parent
used to set local attribute

domain:i mean equal to index
ties can be written f1.A = f2.A = f3.A

IFunction ID:

IFunctions With location splits background and peakfunctions

↓

centre X
height Y
It is needed so that if these params are not known it calcs it.
(Prob not used in background)

Peaks from IPeakfunction
adds width + intensity
Fwhm↑

Peakradius - number of widths from centre
(avoids setting outside local range for things like Gaussian)
↑goes to 0 quickly on wings

ComptonProfile - have real life parameters e.g. mass

underlying fit calcs from these

parseIntensityConstraintMatrix
(in ComptonScatteringCountRate

↳ uses special form of constraint passed as a string, unusual format ask Maryln

Style of MATLAB done for scientist, could be done for all ties

DiffRotDiscreteCircle
addfunction then ~~add~~ set Alias

done so you can set as string

UserFunction (ID dupricated)
- has formula attribute will get attributes then define + set them

Function
mv:: Parser :
create it
define where it gets any variable from

Parameter Ties
m_parser
m_varMap params used in Tie

Originally ties were kept in function
Now if possible they go as deep as they can

Ties can be at different levels
must be at least at level of functions involved
tie might be held by a child if it can

Param Tie based on ParamRef
creates perminant reference
To help with changes (Ties might not be removed)
help with name change
holds 2 pointers to 2 functions
1 sets ref to function
1 that owns reference

m owner holds ref to func

thinking about
seperate out properties, would simplify references + minimizers

ParameterTie hold expression (exp)
replaces param names with '#'
to help with renaming
hashes are replaced with new name

→ class does too much!
↳ changing function would've been better at a UI level

parameterReference are used elsewhere

CrystalField: different parameter
use quantum mechanics
calc eigenvalues
- don't know how many peaks from initial params
↳ some are small, some overlap

- number of params depends on other params
- depends on number of peaks

(in scripts)

- want fit multidomain
- want multi spectra
  no. of params change
  want to fit to physical properties as well

quantum - only use energy from potential

crystal inside crystal
4 ions
2 sets of field parameters
- multiple ion sites
spectra overlap for both

over time replaced complex with more complex but old ones remain

field can have symmetry defines ties etc
temperatures are attributed

cannot get width from quantum calc so can:
- can provide width function → not free param
- vary length - free individually fixed

field params: B 4 0
two ions

peak shape can change,
don't know names of peak params

CrystalFieldFunction
originally used original naming but varied on whether single or multi site

Single site - field param B11

multi site - " " con 0.sp0 pk0

peak

spectra

could be bg for background

- when it get position + intensity it decide what to do with them
- gets the width from source
- gets description of bkg
- builds function for peaks + bkg constructs
- evaluates spectra function on x values

Extra classes for additional real life parameters

function generator creates additional function

reason for source + target.

constrol function
recieves ions + symmetries
decides structor

source fun does quantam
↳ crystalField peaks
↳ " " " BaseImpl
has crystalFieldPeaksbase
↓
calculare Eigensystem

crystalElectricFilld from fortran
↳ use comple FortranMatrix
wrapper round fitting Matrix
Way to deal with indexing starting
in negative

2 tolerance
peak to small
to close together

Scientists want hamiltonian avaliable

Multi site must have it own composite
function with peaks
background belongs to parent peaksplit

~~want to ~~ able

IFuncGeneral
- don't know specific about domain - will know size

- getNumberValuesPerArgument
- getValuesSize - how many values per domain entry

~~F~~unction Domain General uses m_columns.

& Ask Doc Le about any extra Crystal Fit stuff

Fit.cpp

"FitSizeWarning"
reinitalize minimizer, starts again (but not no. of iterations)

IFunction1Dspectrum
when you have ws with properties inside like Q so need spectrum number
May be different way to do it
(use FunctionQDepends)
(Indirect want a function on Q)

FunctionFactory

* createFunction
* createInitialized: - takes string
  - biggest problem brackets
    cant use simple methods

Expression: Parses a string
keeps structure
creates tree with name
and terms
look at test for example

Factory creates expression
; lowest precedent
↑ makes it composite
examines string to get functionalities, constraints etc

Use to build constants etc

getFunctionName GUI
property files have excludes.
curvefitting.gui.Exclude
treat all as 1D function

CalculateChiSquared.
outputs info to log
DOF - degrees of freedom
_errors:
first column values from fit
then tries to find min in one parameter
then estimates error of slice

pdf - probability distribution function
errors using s.d in curves

Chi2Min

To do slices have chebfun base
If given a func it will try to
estimate

Could be used for some analysis

Some generic width calcs for peaks

EstimateFitParameters

searches for best inital values used in crystal field as no way to find them well

Poisson Fitting

~~Gaussian fitting~~

Chi-squared often wrong due to assumption of errors in a Gaussian curve
↳often under/over estimate area under peak

n_i = no. of events in ith bin
$\vec{n} = (n_1, n_2, \ldots, n_K)$
$N = \sum_i n_i$

J parameters, indexed j
$\vec{a} = (a_1, a_2, \ldots, a_j, \ldots a_J)$ = set of parameters

y_i = no. of events predicted to be in i-th bin
$\vec{y} = (y_1, y_2, \ldots, y_K)$

$N_0 = \sum_i y_i$ = total no. of events predicted by model

$\vec{y}$ is a function of $\vec{a}$

weights

$$S = \sum_i w_i (n_i - y_i)^2$$

$$\sum_i y_i = \sum_i n_i \quad \text{(extra constraint)}$$

principle of max likelihood with
$w_i = 1/\sigma_i^2$ (σ_i^2 variances)

likelihood func for poisson hists

$$L_P(\vec{y}; \vec{n}) = \prod_i \exp(-y_i)\, y_i^{n_i} \frac{1}{n_i!}$$

if $\vec{m}$ is true values of $\vec{n}$ without error then likelihood ratio:

$$\lambda = \frac{L(\vec{y}; \vec{n})}{L(\vec{m}; \vec{n})}$$

likelihood ratio test:

$$x_\lambda^2 = -2 \ln \lambda = -2 \ln L(\vec{y}; \vec{n}) + 2 \ln L(\vec{n}; \vec{n})$$

replace $\vec{m}$ with bin by bin model-indep max likelihood ($\vec{n}$)

$$x^2_{\lambda,P} = 2 \sum_i y_i - n_i + n_i \ln(n_i/y_i)$$

$$= 2 \sum_i y_i - n_i + n_i (\ln n_i - \ln y_i)$$

preservation of area
preserve no. of events: $N_0 \equiv \Sigma y_i$
$= \Sigma n_i$
$\equiv N$

$$\frac{\partial L_P}{\partial a_j} = 0$$

$$\Rightarrow \frac{\partial L_P}{\partial y_i} \frac{\partial y_i}{\partial a_j} = 0$$

$$\Rightarrow \sum_i (n_i - y_i) \frac{1}{y_i} \left(\frac{\partial y_i}{\partial a_j} \right) \quad j = 1, \ldots, J$$

Suppose can find params
$\vec{a}' = (a_1', \ldots a_J')$ which are funcs of old params with non-vanishing Jacobians $\frac{\partial (a_1', \ldots a_J')}{\partial (a_1, \ldots a_J)} = 0$

such that
$L(\vec{n}|\vec{m})$ $y_i = a_1' \times g_i (a_2', a_3', \ldots a_J')$

minimise L_P with respect to $\vec{a}_j'$

$$\frac{\partial L_P}{\partial a_j'} = 0 \Rightarrow \frac{\partial \ln L_P}{\partial a_j'} = 0$$

$$\Rightarrow \sum_i (n_i - y_i) \left(\frac{1}{y_i} \right) \frac{\partial y_i}{\partial a_j'} = 0$$

Cost Functions

All inherit from CostFuncFitting.h
(not necessarily directly)

- Least Squares: CostFuncLeastSquares.h

$$\sum_i \left(\frac{y_i^{obs} - y_i}{\sigma_i} \right)^2$$

$y_i = y(x_i)$ = calculated value at x_i
y_i^{obs} = observed value at x_i e.g. counts

σ_i = error estimate at x_i
↳ 1/weights

- Unweighted least squares:
CostFuncUnweightedLeastSquares.h

$$\sum_i (y_i^{obs} - y_i)^2$$

(note: inherits from CostFuncLeastSquares, overrides getWeights so for valid data weight = 1

- RWP : CostFuncRwp.h

$$\sum_i \left(\frac{y_i^{obs} - y_i}{\sigma_i} \right)^2 \Big/ \sum_j \left(\frac{y_j^{obs}}{\sigma_j} \right)^2$$

Note : inherits from Least squares and overrides getWeights

- Poisson : CostFuncPoisson.h

$$2 \sum_i \left\{ y_i^{obs} \log(y_i^{obs}) - y_i^{obs} \log(y_i) - (y_i^{obs} - y_i) \right\}$$

There is now documentation for this one.

Minimizers

Levenberg-Marquardt:

For each iteration we want to replace the parameter vector $\underline{\beta}$ with a new estimate $\underline{\beta} + \underline{\delta}$

approximate the function as
$f(x_i, \underline{\beta} + \underline{\delta}) \approx f(x_i, \underline{\beta}) + \underline{J}_i \underline{\delta}$

$J_i = \frac{\partial f_i}{\partial \beta}$ (gradient of f w.r.t β)

After some derivation (I like wiki version) we get

$$(\underline{J}^T \underline{J} + \lambda \underline{I}) \underline{\delta} = \underline{J}^T [y - f(\underline{\beta})]$$

J = jacobian, J^T = transpose of Jacobian
$(JJ^T = J^T J = I)$

I = identity matrix, e.g. $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

λ is damping parameter

Solve above for $\underline{\delta}$

if the iteration is good (i.e params are better than previous)

divide λ by at most 3
then do next iteration with new β and new λ

if iteration is bad
multiply λ by 2 and do next iteration with original params and new λ.