**Functions**

Q\_maker(N::Int64, start::Vector{<:Real}, stop::Vector{<:Real})

Generates a vector of N 3-dimensional points (SVectors) between the start and stop vectors.

Impetus: For a simulation, neutron users are used to specific paths in reciprocal space in this manner, and a calculator will require these q-points.

* **Parameters:**
  + N: The number of points to generate.
  + start: The starting point of the vector, should be a 3-dimensional vector.
  + stop: The ending point of the vector, should be a 3-dimensional vector.
* **Returns:**
  + points: An array of N SVectors of dimension 3.

update\_dispersion\_simulation!(q::Vector{SVector{3, Float64}}, model::AnalyticalModel)

Updates dispersion for each element of a given vector, using an analytical model. It also updates a model's dense model.

Impetus: User may want to update the model dispersion without any experimental data.

* **Parameters:**
  + q: Vector of 3D points (positions)
  + model: An AnalyticalModel instance

update\_dispersion\_and\_residuals!(group::ExpDataGroup, model::AnalyticalModel, update\_dense\_model::Bool=false)

Updates dispersion and residuals for each element of a given experiment data group, using an analytical model. It also updates dense model if specified.

Impetus: Analytical models are sometimes faster and sometimes a phenomenological/analytical description is all that is accessible to the data analyst. The user will need to have access to a way of updating the dispersion observable based upon changes to the model parameters. The residuals calculation is required for optimization.

* **Parameters:**
  + group: An ExpDataGroup instance
  + model: An AnalyticalModel instance
  + update\_dense\_model: Boolean flag to update dense model

update\_dispersion\_and\_residuals!(group::ExpDataGroup, model::SunnyModel, update\_dense\_model::Bool=false)

Updates dispersion and residuals for each element of a given experiment data group, using a Sunny model. It also updates dense model if specified.

Impetus: For well described complicated systems, calculators like Sunny are essential. The user will need to have access to a way of updating the dispersion observable based upon changes to the model parameters. The residuals calculation is required for optimization.=

* **Parameters:**
  + group: An ExpDataGroup instance
  + model: A SunnyModel instance
  + update\_dense\_model: Boolean flag to update dense model

objective\_function(params::Vector{Float64}, param\_names::Vector{String}, model::AbstractModel, group::ExpDataGroup)

Sets parameters to the provided model, updates dispersion and residuals, and returns sum of squares of residuals.

Impetus: In setting up an optimization problem, the user must find and describe the objective function.

* **Parameters:**
  + params: Vector of parameter values
  + param\_names: Vector of parameter names
  + model: An AbstractModel instance
  + group: An ExpDataGroup instance

refine\_model!(model::AbstractModel, group::ExpDataGroup, verbose::Bool = false)

Refines a given model by updating parameters and optimizing them to minimize residuals.

Impetus: Have a simple high-level call for users to perform model refinements.

* **Parameters:**
  + model: An AbstractModel instance
  + group: An ExpDataGroup instance
  + verbose: Boolean flag to control verbosity

empty\_experiment\_group(q\_points\_list::Vector{Vector{SVector{3, Float64}}}, n\_dense::Int64=100)

Returns an empty experiment data group for given q-points.

Impetus: In the current coding scheme, this empty experiment is required to simulate data, since part of the model information is actually tied into the ExpDataGroup.

* **Parameters:**
  + q\_points\_list: Vector of vectors of 3D points
  + n\_dense: Integer specifying density

set\_initial\_spin\_configuration!(sys::System, original\_spins::Vector{Vector{Int}})

Sets the initial spin configuration in a given system.

Impetus: For linear spin-wave theory style calculations, the magnetic structure is known by the data analyst.

* **Parameters:**
  + sys: A System instance
  + original\_spins: Vector of spins

set\_system\_params!(sys::System, parameters::Dict{String, Float64})

Sets parameters to a given system.

Impetus: To perform optimizations with Sunny, there needed to be a way to update the system parameters repeatedly.

* **Parameters:**
  + sys: A System instance
  + parameters: Dictionary of parameters

sunny\_dispersion(sys::System, swt::SpinWaveTheory, parameters::Dict{String, Float64}, q::SVector{3, Float64})

Calculates dispersion in a system using Sunny model.

Impetus: This function is actually taking the Sunny dispersion and selecting out the most intense mode. Because the eigensolvers are returning un-ordered lists of eigenvalues, there needed to be a way to track modes throughout the Brillouin Zone.

* **Parameters:**
  + sys: A System instance
  + swt: A SpinWaveTheory instance
  + parameters: Dictionary of parameters
  + q: 3D point

calculate\_covariance(objective\_function\_map, param\_ranges)

Calculates the covariance matrix for a given objective function map and parameter ranges. Note: Not functioning properly yet.

Impetus: The data analyst will want to know the quality of their solution, and have a way to share metrics of the quality.

* **Parameters:**
  + objective\_function\_map: A map of objective function values for different parameters
  + param\_ranges: Ranges of parameters for which covariance is calculated

**Classes**

ExpData

Represents experimental data.

Impetus: As an organizational principle, there needed to be a container for all aspects of the experiment, broken into discrete visualizations. This implementation only considers dispersion data, but may be extended to other paradigms.

* **Parameters:**
  + name: A string to identify the dataset.
  + Q: A vector of SVector representing the position in the reciprocal space.
  + E: A vector of SVector representing the energy at corresponding Q points.
  + direction: The principal axis of the data set, could be a vector or a string if no well-defined direction.

ModelData

Represents the data of a model.

Impetus: As an organizational principle, there needed to be a container for the model that is being compared to the observables. Presently, this ModelData lives in the ExpDataGroup, but that may be conceptually incorrect.

* **Parameters:**
  + name: A string to identify the model.
  + model\_Q: A vector of SVector representing the position in the reciprocal space for the model.
  + model\_E: A vector of SVector representing the energy at corresponding model\_Q points for the model.
  + residuals: A vector of SVector representing the difference between experimental E and model E.
  + direction: The principal axis of the model data set.
  + model\_Q\_dense: A vector of SVector representing the denser points in the reciprocal space for the model.
  + model\_E\_dense: A vector of SVector representing the energy at corresponding model\_Q\_dense points for the model.

ExpDataGroup

A group of experimental data.

Impetus: As an organizational principle, there needed to be a grouping container where the discrete parts of the experiment could be put together and passed to an optimizer.

* **Parameters:**
  + files: A vector of ExpData.
  + model: A vector of ModelData.
  + all\_Q: A vector of all the Q points from all the ExpData in files.
  + all\_E: A vector of all the E points from all the ExpData in files.
  + model\_all\_E: A vector of all the model\_E points from all the ModelData in model.

AbstractCalculatorWrapper

An abstract type to wrap the calculation methods.

SunnyCalculatorWrapper

A subtype of AbstractCalculatorWrapper to wrap the Sunny calculation methods.

Impetus: These wrappers were a way to have updatable mutation of the calculators as required by optimization. The AbstractCalculatorWrapper was needed to deal with the concept that there may be a countable but >2 set of possible calculators (e.g. extending beyond dispersion examples). The SunnyCalculatorWrapper is dispatched if the parameters contain SpinWaveTheory and System Sunny objects.

* **Parameters:**
  + dispersion: A function that describes the dispersion relation of the system.
  + sys: A System object for the Sunny calculation.
  + swt: A SpinWaveTheory object for the Sunny calculation.

AnalyticalCalculatorWrapper

A subtype of AbstractCalculatorWrapper to wrap the analytical calculation methods.

Impetus: This calculator allows users to make analytical calculators when either phenological descriptions are all that are available or that the analytical solution to the problem is well-suited and the speed of the analytical version is desired.

* **Parameters:**
  + dispersion: A function that describes the dispersion relation of the system.

ParameterWrapper

Represents the parameters for a model.

Impetus: This wrapper was invoked as a way to have updatable mutation of the parameters as required by optimization.

* **Parameters:**
  + values: A dictionary with parameter names as keys and their corresponding values as values.
  + param\_to\_optimize: A dictionary with parameter names as keys and a Boolean as value, indicating whether the parameter is to be optimized or not.
  + param\_bounds: A dictionary with parameter names as keys and their corresponding upper and lower limits as Tuple values.

AbstractModel

An abstract type for all models.

AnalyticalModel

A subtype of AbstractModel to define an analytical model.

Impetus: These models are a grouping that keeps together essential model information.

* **Parameters:**
  + name: A string to identify the model.
  + parameters: A ParameterWrapper object to hold the parameters of the model.
  + calculator: An AnalyticalCalculatorWrapper object to calculate the model values.

SunnyModel

A subtype of AbstractModel to define a Sunny model.

Impetus: A Sunny model has different needs than an analytical model.

* **Parameters:**
  + name: A string to identify the model.
  + parameters: A ParameterWrapper object to hold the parameters of the model.