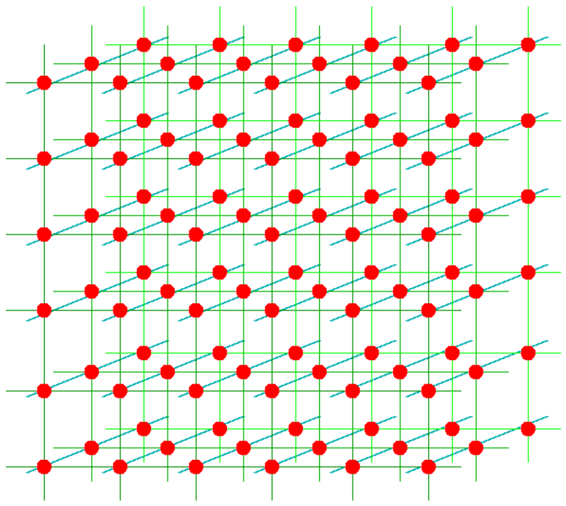
sigmond: Signal Extraction from Monte Carlo Data

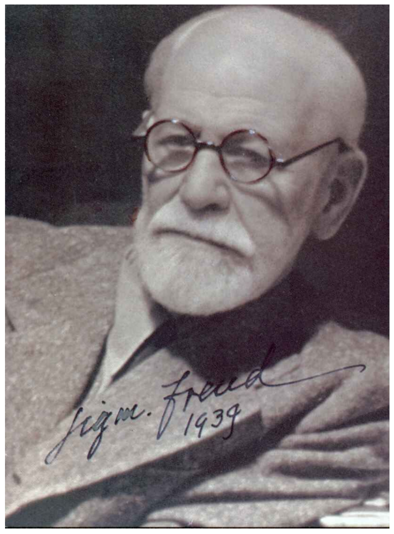
Colin Morningstar

September 15, 2020

Abstract

These notes detail how to use the software suite called `sigmond` for the analysis of Monte Carlo data in lattice QCD.





Contents

1 Introduction

sigmond is a software suite for the analysis of Monte Carlo data in lattice QCD. The name sigmond comes from signal extraction from Monte Carlo data, and also plays upon the name "Sigmund" from Sigmund Freud, a neurologist famous for his analytical thinking. sigmond currently runs only in batch mode. We assume sigmond is being used in a linux environment. These notes focus on the use of the software, so little is said here about how to compile the software.

The source software, build directories with make files, and documentation are bundled in the compressed tar file sigmond.tar.gz. Issue the command

```
tar xzvf sigmond.tar.gz
```

which creates a single directory sigmond in the current directory. cd into this directory, and one sees subdirectories named source, build, and doc. This user guide (LaTeX and pdf files) is stored in the doc subdirectory. All source files are contained in the source directory. Before compiling, be sure to edit the file sigmond/build/Makefile.inc. To compile sigmond in batch mode, cd into the build/batch directory and issue a make command.

Other libraries which are necessary include lapack, xmgrace (plotter), and Minuit2. Edit the file sigmond/build/Makefile.inc to specify the location of the needed libraries and header files on your system. You should change the SRC_DIR entry in this file to the full path name of the directory on your system where you untarred the sigmond directory.

In batch mode, a single input file must be given as a command line argument. The input file must contain an XML document which tells sigmond what data to analyze and what tasks to do. The structure and allowed content of the XML input is the subject of these notes and is described in detail below. If the input XML file is named input.xml, then a batch run is accomplished by the command

```
sigmond_batch input.xml
```

sigmond writes and reads binary data files. It can write out both bin and sampling files. A bin file contains Monte Carlo data of so-called simple observables corresponding to an integrand in a Monte Carlo integrand, something that can be defined on a single field configuration. Several of such successive measurements can be averaged into a so-called bin, and the file stores these bins for subsequent reading and analysis. Sampling files contain resampling values using either jackknife or bootstrap resampling. Other various files can be written and read, such as those containing information about correlation matrix rotations and transformations. Any file format written by sigmond can, of course, be read by sigmond. Currently, sigmond can read files in the format created by chroma_laph and last_laph. Of course, sigmond can be re-programmed to utilize other data formats by changing the source files in the source/data_handling subdirectory. sigmond also writes log files in XML format, as well as xmgrace files for plots. More on file formats is discussed later in these notes.

2 Overview of the XML input

The input XML document must have a root tag named `<SigMonD>`. Inside this root tag, there must be one tag named `<Initialize>`, and another tag named `<TaskSequence>`. Specification of the data files, the log file, and so on, must be located in the `<Initialize>` tag. The tasks to be performed using the data are then placed inside the `<TaskSequence>` tag. The input XML file must have the form given below.

```
<SigMonD>

    <Initialize>
        <ProjectName> Name Of Project </ProjectName>
        <LogFile> log_output.xml </LogFile>
        <KnownEnsemblesFile>/path/ensembles.xml</KnownEnsemblesFile> (optional)
        <EchoXML/>  (optional)
        <MCBinsInfo>  ...  </MCBinsInfo>
        <MCSamplingInfo> ... </MCSamplingInfo>
        <MCObservables>  ...  </MCObservables>
    </Initialize>

    <TaskSequence>
        <Task><Action>...</Action> ... </Task>
        <Task><Action>...</Action> ... </Task>
        ....
    </TaskSequence>

</SigMonD>
```

3 The data initializer tag

The `<Initialize>` tag must contain the data files, the log file, and so on. This tag must have the form:

```
<Initialize>
    <ProjectName> Name Of Project </ProjectName>
    <LogFile> log_output.xml </LogFile>
    <KnownEnsemblesFile>/path/ensembles.xml</KnownEnsemblesFile>
    <EchoXML/>
    <MCBinsInfo>  ...  </MCBinsInfo>
    <MCSamplingInfo> ... </MCSamplingInfo>
    <MCObservables>  ...  </MCObservables>
</Initialize>
```

– If `<ProjectName>` is missing, a default name will be created.

– If `<LogFile>` is missing, a default name for the log file is used. Output to this file will be in XML format.

– The tag `<KnownEnsemblesFile>` is optional and is described below.

– If `<EchoXML/>` is missing, the input XML will not be written (echoed) to the log file.

– The tag `<MCBinsInfo>` is mandatory: it specifies the ensemble, controls rebinning the data, and possibly omitting certain configurations in the ensemble if these is data corruption. This tag is described below.

– The tag `<MCSamplingInfo>` is mandatory. It controls the default resampling method: jackknife or bootstrap. This default method is assumed for all reading and writing sampling results to and from files. Note that both jackknife and bootstrap resampling can be done in any program execution, but only one can be used for reading/writing to files. A more detailed description of this tag is given below.

– `<MCObservables>` describes the data to be input for analysis. This tag is described in detail below.

3.1 The known ensembles information file

Various Monte Carlo ensembles are made known to `sigmond` in the ensembles XML file so that one only need specify an identification string to `sigmond` and `sigmond` will know all that it needs to know about the ensemble. The absolute path to this file can be specified in the `<KnownEnsemblesFile>` tag. If not given, a default location for this file has been stored during the compilation. This file must have information specified in the following XML format:

```
<KnownEnsembles>
  <Infos>
    <EnsembleInfo>...</EnsembleInfo>
    <EnsembleInfo>...</EnsembleInfo>
     ....
  </Infos>
  <CLSEnsembleWeights>
    <Ensemble>...</Ensemble>
     ....
  </CLSEnsembleWeights>
</KnownEnsembles>
```

with each ensemble in the `<Infos>` tags specified by

```
<EnsembleInfo>
   <Id>clover_s24_t128_ud840_s743</Id>
   <NStreams>4</NStreams>
   <NMeas>551</NMeas>
```

```
      <NSpace>24</NSpace>
      <NTime>128</NTime>
      <Weighted/>  (if has CLS weights; omit otherwise)
   </EnsembleInfo>
```

The entries in the `<CLSEnsembleWeights>` tag must have the form:

```
   <Ensemble>
      <Id>cls21_D200_r000</Id>
      <Weights> 0.999 0.998 ... </Weights>
   </Ensemble>
```

3.2 The bins information tag

The tag `<MCBinsInfo>` is mandatory: it identifies the Monte Carlo ensemble and controls rebinning the data, and possibly omitting certain configurations in the ensemble (for example, if certain measurements have been corrupted). The XML must have the form below:

```
<MCBinsInfo>
  <MCEnsembleInfo>clover_s24_t128_ud840_s743</MCEnsembleInfo>
  <TweakEnsemble>
     <Rebin>2</Rebin>
     <Omissions>2 7 11</Omissions>
  </TweakEnsemble>
</MCBinsInfo>
```

The tag `<MCEnsembleInfo>` tag is mandatory and identifies the ensemble of configurations upon which all calculations are based. The string inside this tag must match one of the ensemble identification strings known to `sigmond`. Known ensembles are stored in a file, usually named `ensembles.xml`, which was discussed above. Given one of the identifying strings specified in the file `ensembles.xml`, `sigmond` class knows how many Markov-chain streams are available, how many RHMC trajectory numbers are valid, and so on.

If you are using only a small fraction of the configurations, it is easier to use

```
  <TweakEnsemble>
    <KeepFirst>3</KeepFirst>  (index of first config to keep)
    <KeepLast>12</KeepLast>   (index of last config to keep)
  </TweakEnsemble>
```

To specify an ensemble not known to SigMonD, you must provide the following information: $\{n_{\rm meas},\ n_{\rm streams},\ n_x,\ n_y,\ n_z,\ n_t\}$ This is done by appending to the id string fields that give this information in the order above, separated by "`|`". Example:

```
   <MCEnsembleInfo>idname|800|1|24|24|24|36</MCEnsembleInfo>
```

The `<TweakEnsemble>` tag is optional. In the `<Rebin>` tag, a positive integer should be given. That number of successive Monte Carlo measurements are combined into a single bin. If there are N measurements, and `<Rebin>` is input as n, there will be N/n bins.

The `<Omissions>` tag, if present, should contain non-negative integers corresponding to the configurations (measurements) to omit. These indices are zero offset, that is, the first configuration in the ensemble is numbered zero.

3.3 The default sampling information tag

The tag `<MCSamplingInfo>` is mandatory. It controls the default resampling method: jackknife or bootstrap. This default method is assumed for all reading and writing sampling results to and from files. Note that both jackknife and bootstrap resampling can be done in any program execution, but only the default method can be used for reading/writing to files. For specifying jackknife resampling as the default, the XML input must have the form

```
<MCSamplingInfo>
   <Jackknife/>
</MCSamplingInfo>
```

To specify bootstrap resampling as the default, the XML input must have the form below, which provides details about the bootstrapping:

```
<MCSamplingInfo>
   <Bootstrapper>
      <NumberResamplings>2048</NumberResamplings>
      <Seed>6754</Seed>
      <BootSkip>127</BootSkip>
      <Precompute/>
   </Bootstrapper>
</MCSamplingInfo>
```

`<Seed>` must be an unsigned 32-bit integer $0 \ \ldots \ 2^{32}-1$. For a given bootstrap resampling, the Mersenne twister is called in sequence to generate the sample. Before generating the next resampling, the Mersenne twister is called `<BootSkip>` number of times. The `<BootSkip>` parameter allows more variety in how the bootstrap resamplings are generated. If the `<Precompute>` empty tag is present, then the bootstrap indices are computed all at once and stored in memory. If not present, the bootstrap indices are computed "on the fly". The bootstrap samples are generated using the Mersenne twister, but in a pseudorandom way that is repeatable. So a given resampling can be regenerated as needed. Including `<Precompute>` uses more memory. Try your computations with and without this tag to see which is faster, since speed depends on memory access speed.

3.4 The observables tag

The `MCObservables` tag specifies the data files to use in looking for the necessary Monte Carlo measurements, as well as the observables themselves, optionally. This tag must have the form:

```
<MCObservables>
    <BLCorrelatorData>              <-- basic LapH correlator data files
```

```
        <FileListInfo>...</FileListInfo>
        <FileListInfo>...</FileListInfo>
           ....
    </BLCorrelatorData>
    <BLVEVData>                   <-- basic LapH VEV data files
        <FileListInfo>...</FileListInfo>
        <FileListInfo>...</FileListInfo>
           ....
    </BLVEVData>
    <BinData>                     <-- bin files in sigmond format
        <FileName>...</FileName>
         ....
    </BinData>
    <SamplingData>                <-- resampling files in sigmond format
        <FileName>...</FileName>
         ....
    </SamplingData>
    <UseCheckSums/>  (optional)
    <Specifications>
          ...
        specifications of observables (optional)
          ...
    </Specifications>
</MCObservables>
```

Files containing basic LapH (see later) data for the temporal correlators to be analyzed must be specified in terms of `<FileListInfo>` tags inside a `<BLCorrelatorData>` tag. Similarly, files containing basic LapH data for any vacuum expectation values must be specified in terms of `<FileListInfo>` tags inside a `<BLVEVData>` tag. Bin files in `sigmond` format must be specified in a `<BinData>` tag, and sampling files in `sigmond` format must be listed in a `<SamplingData>` tag.

Each `<FileListInfo>` tag must have the form

```
<FileListInfo>
    <FileNameStub>  ...  </FileNameStub>
    <MinFileNumber> ...  </MinFileNumber> (default=0)
    <MaxFileNumber> ...  </MaxFileNumber>
    <FileMode>      ...  </FileMode>      (optional)
</FileListInfo>
```

Each `<FileListInfo>` tag contains information about a list of files having a common stub and a numerical suffix from a minimum value to a maximum value. For example, if `<FileNameStub>` contained `/data/correlators/corr`, `<MinFileNumber>` contained `10`, and `<MaxFileNumber>` contained `12`, then the three files named `corr.10`, `corr.11`, and `corr.12` in the directory `/data/correlators` would be read by `sigmond`. The `<FileMode>` tag is provided to indicate whether files may be overwritten. To allow for overwriting, the

`<FileMode>` tag must contain the word `overwrite`. Otherwise, files will be protected from possible overwriting.

Observables can be specified inside a `<Specifications>` tag. If no observables are specified, then all observables found while reading the files will be used. If any observables *are* specified, then only those observables will be considered for input: files corresponding to other observables will be ignored, and an exception is thrown if the file containing the data for any requested observable cannot be found.

Observables can be specified inside a `<Specifications>` tag as follows:

(a) a single correlator (no VEV subtraction)

```
<Correlator>
  <Source><Operator>...</Operator></Source>
  <Sink><Operator>...</Operator></Sink>
</Correlator>
```

(b) a single correlator with VEV subtraction

```
<CorrelatorWithVEV>
  <Source><Operator>...</Operator></Source>
  <Sink><Operator>...</Operator></Sink>
</CorrelatorWithVEV>
```

(c) a single VEV

```
<VEV><Operator>...</Operator></VEV>
```

(d) a Hermitian matrix of correlators (no VEV subtraction)

```
<HermitianCorrelationMatrix>
  <Operator>...</Operator>
  <Operator>...</Operator>
         ...
</HermitianCorrelationMatrix>
```

(e) a Hermitian matrix of correlators with VEV subtraction

```
<HermitianCorrelationMatrixWithVEVs>
  <Operator>...</Operator>
  <Operator>...</Operator>
       ...
</HermitianCorrelationMatrixWithVEVs>
```

(f) matrix of correlators (no Hermiticity, no VEV subtract)

```
<CorrelationMatrix>
  <Operator>...</Operator>
  <Operator>...</Operator>
       ...
</CorrelationMatrix>
```

(g) matrix of correlators with VEV subtract (no Hermiticity)

```
<CorrelationMatrixWithVEVs>
  <Operator>...</Operator>
  <Operator>...</Operator>
        ...
</CorrelationMatrixWithVEVs>
```

(h) set of observables in bin files (non basic LapH) of `sigmond` format

```
<ObsBins>
  <MCObservable>...</MCObservable>
  <MCObservable>...</MCObservable>
         ...
</ObsBins>
```

(i) set of observables in sampling files of `sigmond` format

```
<ObsSamplings>
  <MCObservable>...</MCObservable>
  <MCObservable>...</MCObservable>
        ...
</ObsSamplings>
```

Details concerning how to specify a QCD operator in an `<Operator>` tag and all observables in an `<MCObservable>` tag are described in the next section. In all of the above tags, the alternative tags `<BLOperator>`, `<GIOperator>`, `<OperatorString>`, `<BLOperatorString>`, and `<GIOperatorString>` can be used wherever an `<Operator>` is used. These tags are explained in the next section. If a tag `<AllHermitian/>` appears inside the `<Specifications>` tag, then all correlation matrices are treated as Hermitian.

4 Monte Carlo observables and operators

A key concept used by `sigmond` is that of a Monte Carlo observable. Each observable must be associated with a *real-valued* quantity that can be estimated by our Monte Carlo path integrals. These can be simple quantities, such as the real or imaginary part of a temporal correlator for one time separation, that can be defined on a single gauge configuration, or much more complicated quantities, such as a fit parameter yielding a stationary-state energy, determined by fitting a decaying exponential to a temporal correlation function.

Many Monte Carlo observables involve quantum field operators. For example, the real and the imaginary parts of the vacuum expectation value of a QCD field operator are examples of Monte Carlo observables here. The real and the imaginary parts of the temporal correlation of two operators separated by some time interval are also Monte Carlo observables. So defining and specifying QCD operators is an important part of specifying Monte Carlo observables. In this section, a variety of operators are first discussed, and then the all-important `<MCObservable>` tag is discussed.

4.1 Basic LapH operators

A so-called *basic LapH* QCD operator can be specified in "long form" by an `<Operator>` or `<BLOperator>` tag, or in "short form" by an `<OperatorString>` or `<BLOperatorString>` tag. A basic LapH operator is an assemblage of covariantly-displaced LapH-smeared quark fields in a variety of spatial orientations. The XML format for specifying an operator must be of the form:

```
<BLOperator>
    <NumberOfHadrons> 2 </NumberOfHadrons>
        ....
</BLOperator>
```

or of the form (see below)

```
    <BLOperatorString> ... </BLOperatorString>
```

An `<BLOperator>` tag must contain a `<NumberOfHadrons>` tag. The rest of the XML depends on the number of hadrons. A "hadron" here means a baryon, a meson, or a glueball. It could also mean a tetraquark system, a pentaquark system, *etc.* It is a gauge-invariant localized quantity.

- Number of hadrons = 0: No further tags are required (this is the "default" constructor).

- Number of hadrons = 1: XML must include

```
<Hadron>
   ....described below....
</Hadron>
<LGIrrepRow> 3 </LGIrrepRow>
```

- Number of hadrons = 2, 3, 4,. . . : XML must include

```
<Total>
   <Isospin> triplet </Isospin>
   <IsoCGId> 0 </IsoCGId>} (assume zero if absent)
   <Momentum>  0 0 0  </Momentum>
   <LGIrrep> T1gm </LGIrrep>
   <LGCGId> 0 </LGCGId>} (assume zero if absent)
```

```
       <LGIrrepRow> 3 </LGIrrepRow>
    </Total>
    <Hadron1>
        ....
    </Hadron1>
    <Hadron2>
        ....
    </Hadron2>
       ....
```

In the `<Total>` tag, the `<Isospin>` tag must take a value such as `singlet`, `doublet`, `triplet`, *etc.* When there are three or more hadrons, an isospin Clebsch-Gordan occurrence identifying number must be specified in a `IsoCGId` tag: value 0 to one less than the number of times that the irrep specified in the `<Isospin>` tag occurs in the direct product of the single-hadron isospins. A value of zero for this tag is assumed if the tag is absent. If the irrep occurs more than once in the Clebsch-Gordan series of the direct product of single-hadron irreps, then a little group Clebsch-Gordan identifying number must be specified (value 0 to one less than the number of occurrences) in a `<LGCGId>` tag. A value of 0 is assumed if absent.

The constituent single-hadron operators inside an `<BLOperator>` tag must be described in separate tags named `<Hadron1>`, `<Hadron2>`, *etc.* If a single hadron is a meson or a baryon, then a single `<Hadron>` tag must occur. Single hadron tags must have the form

```
<Hadron1>
    <Flavor> eta </Flavor>
    <Momentum>  0 0 0  </Momentum>
    <LGIrrep> A1p </LGIrrep>
    <SpatialType> DDL </SpatialType>
    <SpatialIdNum> 4 </SpatialIdNum>
    <DispLength> 3 </DispLength>
</Hadron1>
```

Allowed meson `<Flavor>` tag values are `pion`, `eta`, `phi`, `kaon`, and `kbar`. Note that `pion` does *not* mean an actual pion, rather, it just means an isovector consisting of u, d quarks. `pion` is just a shorter name than `isovector_du`. Similarly, an `eta` means an isoscalar meson consisting of $\overline{u}u+\overline{d}d$ quarks. A `phi` is an isoscalar meson that is $\overline{s}s$. `kaon` refers to a strangeness $S=1$ meson, and `kbar` refers to a strangeness $S=-1$ meson. Allowed baryon `<Flavor>` tag values are `nucleon`, `delta`, `lambda`, `sigma`, `xi`, and `omega`. `<Momentum>` is the momentum of the particle in a chosen "reference" term of the total operator. The total momentum can be obtained by adding all of the single hadron momenta, and the total momentum is fixed. The total operator is a superposition of terms that are rotations, parity-transformations, and G-parity transformations of the reference term. Each `<Momentum>` tag must contain three integers (in units of $2\pi/L$, where L is the spatial extent of the L^3 lattice volume) which describe the momentum of the hadron in the reference term. `<LGIrrep>` specifies the irreducible representation of the little group corresponding to the hadron momentum.

If the single constituent is a glueball, then its `<Hadron>` tag must be of the form

```
<Hadron1>
    <Flavor> glueball </Flavor>
    <Momentum>  0 0 0  </Momentum>
    <LGIrrep> A1gp </LGIrrep>
    <SpatialType> TrEig </SpatialType>
</Hadron1>
```

If the single constituent is a tetraquark, then its `<Hadron>` tag must be of the form

```
<Hadron1>
   <Flavor> isotriplet_phi_pion </Flavor>
   <Momentum>  0 0 0  </Momentum>
   <LGIrrep> A1gm </LGIrrep>
   <SpatialType> QDX </SpatialType>
   <SpatialIdNum> 2 </SpatialIdNum>
   <DispLength> 3 </DispLength>
   <ColorType> 1 </ColorType>  (or -1)
</Hadron1>
```

Allowed tetraquark `<Flavor>` tag values are shown in the first column of the table below. When using the short hand `<BLOperatorString>` form (see below), tetraquarks are specified using the short notation in the second column below. The final `p` in the entries of the second column refers to color symmetric. For color antisymmetric operators, replace the `p` by `m`.

	(short notation)
isosinglet_eta_eta	tquuuu1p
isotriplet_eta_pion	tquudu3p
isosinglet_pion_pion	tqdudu1p
isotriplet_pion_pion	tqdudu3p
isoquintet_pion_pion	tqdudu5p
isodoublet_kaon_eta	tqsuuu2p
isodoublet_kaon_pion	tqsudu2p
isoquartet_kaon_pion	tqsudu4p
isotriplet_phi_pion	tqssdu3p
isosinglet_eta_phi	tquuss1p
isodoublet_kaon_phi	tqsuss2p
isosinglet_phi_phi	tqssss1p

An operator can also be specified by a short string inside an `<BLOperatorString>` tag. For example:

```
<BLOperatorString> glueball P=(0,0,0) A1gp_1 TrEig </BLOperatorString>
<BLOperatorString> pion P=(0,0,0) A1um_1 SD_5  </BLOperatorString>
<BLOperatorString> isotriplet_pion_pion A1um_1 CG_1 [P=(0,0,1) A1p LSD_1]
  [P=(0,0,-1) A2m TSD_2] </BLOperatorString>
<BLOperatorString> tquuuu1p P=(0,0,0) A1um_1 QDX_1</BLOperatorString>
```

If the `CG_1` token is absent, a value 0 is assumed. If the displacement length is omitted, then default values are used: 3 for mesons, 2 for tetraquarks and baryons.

4.2 General irreducible operators

Other QCD operators are called general irreducible operators and are specified inside a `<GIOperator>` tag. For example, a "rotated" operator or other linear superpositions of the basic operators are general irreducible operators. The XML format for specifying such an operator must be of the form

```
<GIOperator>
   <Isospin> triplet </Isospin>
   <Momentum>  0 0 0  </Momentum>
      or <MomentumSquared> 0 </MomentumSquared>
   <LGIrrep> T1gm </LGIrrep>
   <LGIrrepRow> 3 </LGIrrepRow>
   <IDName>a_string_no_whitespace</IDName> (24 char max)
   <IDIndex> 0 </IDIndex> (0 if absent)
</GIOperator>
```

or using the shorter form

```
<GIOperatorString>isotriplet P=(0,0,0) A1um_1 IDname 2</GIOperatorString>
```

The `<IDName>` tag content must be a string containing no white space and no more than 24 characters in length. The `<Isospin>` tag must take a value such as `singlet`, `doublet`, `triplet`, and so on.

You can also specify the momentum squared and/or leave out the irrep row (*e.g.,* if it was averaged over). For example,

```
  <GIOperatorString>isotriplet S=-1 PSQ=1 A2m IDname 2</GIOperatorString>
```

If the `IDIndex` token is absent, a value 0 is assumed.

4.3 Temporal correlators

Information about a temporal correlator can be specified with a `<Correlator>` tag using XML of the form

```
<Correlator>
   <Source><Operator>..</Operator></Source>
   <Sink><Operator>..</Operator></Sink>
</Correlator>
```

Inside the `<Source>` and `<Sink>` tags, one can use one of the following tag types: `<Operator>`, `<BLOperator>`, `<GIOperator>`, or the shorter `<OperatorString>`, `<BLOperatorString>`, or `<GIOperatorString>` tags.

A temporal correlator at one particular time separation is specified by XML of the form

```
<Correlator>
   <Source><Operator>..</Operator></Source>
   <Sink><Operator>..</Operator></Sink>
   <TimeIndex>..</TimeIndex>
   <HermitianMatrix/>            (optional)
   <SubtractVEV/>                (optional)
</Correlator>
```

where the `<TimeIndex>` specifies the requested time separation. If the `<HermitianMatrix>` tag is given, this facilitates input which automatically averages using the complex conjugate elements, if available. VEV subtraction is specified by the `<SubtractVEV/>` tag.

4.3.1 `CorrelatorMatrixInfo`

We may also specify a more general temporal correlation matrix, including whether VEV subtraction is needed and if the matrix is hermitian or not. The XML must be of the form

```
<CorrelatorMatrixInfo>
    <Operator>...</Operator>
    <Operator>...</Operator>
 (or   <OperatorString>...</OperatorString>)
          ...
    <HermitianMatrix/>    (optional)
    <SubtractVEV/>    (optional)
    <AssignName>descriptive_name</AssignName>
</CorrelatorMatrixInfo>
```

or recalled by

```
<CorrelatorMatrixInfo>
    <Name>descriptive_name</Name>
</CorrelatorMatrixInfo>
```

4.3.2 `DiagonalCorrelatorSet`

`sigmond` provides a class that is useful for storing information about a diagonal set of correlators. The CorrelatorInfo is available for each entry as well as information related to a fit to each correlator. The XML construction can take different forms, either with the `<Sequential>` tag to specify the listed operators in an explicit ordering or with `<OperatorIndex>` tags to give the operator orderings. Valid input XML can look like

(a)
```
<DiagonalCorrelatorSet>
        <DiagonalCorrelator>
           <OperatorString>pion P=(0,0,0) A1um_1 SS_0</OperatorString>
           <OperatorIndex>0</OperatorIndex>
        </DiagonalCorrelator>
        <DiagonalCorrelator>
           <OperatorString>kaon P=(0,0,0) A1u_1 SS_0</OperatorString>
           <OperatorIndex>1</OperatorIndex>
        </DiagonalCorrelator>
        <SubtractVEV/>   (optional)
     </DiagonalCorrelatorSet>
```

or

(b) `<DiagonalCorrelatorSet>`

```
    <Sequential>
        <OperatorString>pion P=(0,0,0) A1um_1 SS_0</OperatorString>
        <OperatorString>kaon P=(0,0,0) A1u_1 SS_0</OperatorString>
    </Sequential>
     <SubtractVEV/>   (optional)
  </DiagonalCorrelatorSet>
```

or

(c) from a rotated correlation matrix (6.8.2)

```
  <DiagonalCorrelatorSet>
    <RotatedSequential>
       <GIOperatorString>....</GIOperatorString> (can omit ID index)
       <NumberOfLevels>34</NumberOfLevels>
    </RotatedSequential>
    <SubtractVEV/>   (optional)
  </DiagonalCorrelatorSet>
```

or

(d) from a general correlation matrix

```
  <DiagonalCorrelatorSet>
    <CorrelatorMatrixInfo>
    ....
    </CorrelatorMatrixInfo>
  </DiagonalCorrelatorSet>
```

4.4 Monte Carlo observables

The all-important `<MCObservable>` tag is used to specify one particular Monte Carlo observable. This will be used extensively in many of the tasks, which will be introduced later in these notes. Each observable must be associated with a *real-valued* quantity that can be estimated by our Monte Carlo path integrals. These can be simple quantities, such as the real or imaginary part of a temporal correlator for one time separation, that can be defined on a single gauge configuration, or much more complicated quantities, such as a fit parameter yielding a stationary-state energy, determined by fitting a decaying exponential to a temporal correlation function.

An observable is termed *simple* if it can be associated with the integrand of a single path integral. Simple observables include the real or imaginary part of a temporal correlator for one time separation, and the real or imaginary part of a vacuum expectation value. Other observables are referred to as *nonsimple*.

The `<MCObservable>` tag is meant to encompass all observables of interest. Observables can be classified as *primary* or *secondary*: "primary" refers to operator VEVs and correlators of field operators, which can be of type "BasicLapH" or "GenIrrep", and "secondary" refers to fit parameters and other user-defined observables.

There are two types of primary observables currently implemented: vacuum expectation values of an operator, and temporal correlators of two operators for one time separation. A VEV is specified by the following XML:

```
<MCObservable>
   <VEV><Operator>..</Operator></VEV>
   <Arg>RealPart</Arg>
</MCObservable>
```

In the `<Arg>` tag, the content can be either `Re`, `RealPart`, `Im`, or `ImaginaryPart`. A correlator at a particular time separation is specified by XML for the form

```
<MCObservable>
   <Correlator>
      <Source><Operator>..</Operator></Source>
      <Sink><Operator>..</Operator></Sink>
      <TimeIndex>..</TimeIndex>
      <HermitianMatrix/>    (optional)
      <SubtractVEV/>    (optional)
   </Correlator>
   <Arg>RealPart</Arg>
</MCObservable>
```

Again, in the above, `<BLOperator>`, `<GIOperator>`, or the shorter `<OperatorString>`, `<BLOperatorString>`, or `<GIOperatorString>` tags can be used wherever an `<Operator>` tag is used. If the `<HermitianMatrix/>` tag is given, this facilitates input which automatically averages using the complex conjugate elements, if available. If `<Arg>` is omitted, then the real part is assumed.

Secondary observables are specified using a `<ObsName>` tag and an unsigned integer `<Index>` tag. An optional `<Arg>` tag (assumed RealPart if absent) and an optional `<Simple/>` tag (if absent, the observable is assumed to be nonsimple) can also be included. In summary, the XML for a secondary operator takes the form

```
<MCObservable>
   <ObsName>T1upEnergy</ObsName>      (32 char or less, no white space)
   <Index>3</Index>                   (opt nonneg integer: default 0)
   <Simple/>                          (if simple observable)
   <Arg>RealPart</Arg>
</MCObservable>
```

5 File formats

`sigmond` assumes the Monte Carlo data files are in the format created by `chroma_laph` and `last_laph`, which use objects of a templated class named `IOMap<K,V>`, where `K` is a record key type and `V` is a data type, to write the data to various files. Each file created by `IOMap` contains in the following order:

(1) a character 'L' or 'B' indicating endianness
(2) a 32-character ID string
(3) address offset location where the map is stored in the file (8 bytes)
(4) a character 'Y' or 'N' describing if checksums are stored in the file
(5) a header string of any length
(6) the data records one after another
(7) the file map containing the keys and the offset locations of the records
(8) an ending character 'E'

Each record contains in the following order:

(1) the size in bytes of the record (excluding checksum)
(2) the data itself in binary format
(3) the checksum of the data (optionally).

Of course, `sigmond` can be re-programmed to utilize other data formats by changing the source files in the `source/data_handling` subdirectory.

6 The task sequence tag

The `<TaskSequence>` tag specifies the tasks to be performed using the Monte Carlo data. This tag is needed in batch mode, but can be omitted in interactive mode. If present, this tag must have the form:

```
<TaskSequence>
    <Task><Action>...</Action> ...  </Task>
    <Task><Action>...</Action> ...  </Task>
     ....
</TaskSequence>
```

Each `<Task>` tag must begin with an `<Action>` tag, and the `<Action>` tag must be a simple XML tag (*i.e.* it cannot contain any XML tags) containing the name of the task to be done. The rest of these notes are dedicated to describing the needed XML contents for the `<Task>` tags.

6.1 The Memory Management tasks

The simplest tasks that can be performed involve the management of data stored in memory.

6.1.1 `EraseData`

The `EraseData` task requires the specification of some set of `MCObservables` that are to have their data removed from memory, which includes the samplings associated with the observables listed. The XML for this task has the form

```
<Task>
    <Action>EraseData</Action>
    <MCObservable>...</MCObservable>
    <MCObservable>...</MCObservable>
      ...
</Task>
```

6.1.2 `EraseSamplings`

The function of the `EraseSamplings` task differs from the `EraseData` task in that it only erases the samplings for the observable. Thus, if the observable is simple, this task will still leave data associated with the observable. The form of the XML for this task is as follows

```
<Task>
    <Action>EraseSamplings</Action>
    <MCObservable>...</MCObservable>
    <MCObservable>...</MCObservable>
      ...
</Task>
```

6.1.3 `ClearMemory`

It is possible for the amount of data being analyzed to be so large that memory usage becomes a concern. Thus, a task called `ClearMemory` is provided that will clear all accumulated data from memory, including all samplings. This task can be called in between other tasks as many times as one likes. The appropriate format for this task is as follows:

```
<Task>
    <Action>ClearMemory</Action>
</Task>
```

6.1.4 `ClearSamplings`

The only difference between the `ClearSamplings` task and the `ClearMemory` task, is that the `ClearSamplings` will only clear samplings. Therefore, there will still exist data associated with any simple observables stored in memory. The form of the XML for this task is

```
<Task>
    <Action>ClearSamplings</Action>
</Task>
```

6.2 Reading and Writing tasks

The samplings associated with a particular observable can be written to disk or read from disk using the `WriteSamplingsToFile` task or the `ReadSamplingsFromFile` task, respectively.

6.2.1 `ReadFromFile`

This task will read bins or samplings in the file specified by the `<FileName>` tag and put them into memory. If any `<MCObservable>` tags are specified, then only those observables will have their samplings read into memory. The XML for this task has the form

```
<Task>
  <Action>ReadFromFile</Action>
   <FileType>bins</FileType> (or samplings)
   <FileName>name_of_file</FileName>
   <MCObservable>...</MCObservable>   (these are optional)
   <MCObservable>...</MCObservable>
</Task>
```

6.2.2 `WriteToFile`

This task allows any number of bins or samplings that are stored in memory to be written to a file (only if all samplings are available). The file is specified with the `<FileName>` tag. If the file specified does not exist, it will be created. If the file exists and the file mode is set to "overwrite", the old file will be destroyed and completely overwritten. If the file exists but the file mode is not set, then the header is checked for consistency and these samplings are added to the file, as long as the observables do not already exist in the file. The Monte Carlo observables associated with the desired bins or samplings are specified with `MCObservable>` tags. The XML for this task has the form

```
<Task>
  <Action>WriteToFile</Action> (uses samplings mode in <MCSamplingInfo> tag)
   <FileType>bins</FileType>  (or samplings)
   <FileName>name_of_file</FileName>
   <WriteMode>overwrite</WriteMode> (optional: protect, update, overwrite)
   <MCObservable>...</MCObservable> (these are needed)
   <MCObservable>...</MCObservable>
</Task>
```

6.2.3 `WriteCorrMatToFile`

This task writes an entire correlator matrix to a file, either as bins or as samplings. The required XML is

```
<Task>
  <Action>WriteCorrMatToFile</Action>
    <FileType>bins</FileType> (or samplings)
   <FileName>name_of_file</FileName>
   <WriteMode>overwrite</WriteMode> (optional: protect, update, overwrite)
<CorrelatorMatrixInfo>
  <BLOperatorString>....</BLOperatorString>
   ....
  <HermitianMatrix/>
  <SubtractVEV/>
</CorrelatorMatrixInfo>
<MinTimeSep>3</MinTimeSep>
<MaxTimeSep>25</MaxTimeSep>
<SeparateVEVWrite/>  (see below)
</Task>
```

Default write mode is `protect`. If type is `samplings`, the VEVs are NOT written to file by default; include the tag `<SeparateVEVWrite>` if you still want the VEVs separately written.

6.2.4 `GetFromPivot`

This task will read the samplings for fit energies (with reference if available) and amplitudes from a specified pivot file (see Sec. 6.8.3 for details). The XML for this task has the form

```
<Task>
  <Action>GetFromPivot</Action>
   <Type>SinglePivot</Type>
   <SinglePivotInitiate> ... </SinglePivotInitiate>
        (see Sec. 6.8.3 for details on reading pivot from file/memory)
   <ReferenceEnergy>  (optional: gives energies as a ratio over the ref.)
     <Name>kaon</Name><IDIndex>0<IDIndex>
   </ReferenceEnergy>
</Task>
```

6.3 The `PrintXML` tasks

The `PrintXML` task is used to print information about Monte Carlo observables. The results of this task will be output to the file specified in the `<LogFile>` tag, and will be in XML format. The `PrintXML` task must be one of the following types:

6.3.1 `MCValues`

Prints the mean, standard deviation, various jackknives, and the autocorrelation at a few Markov times for the specified observable. If the `<Verbose/>` or `<ShowBins/>` tag is used, then the value of the observable in each bin is output as well. The XML for this type must be of the form:

```
<Task>
    <Action>PrintXML</Action>
    <Type>MCValues</Type>
    <MCObservable> ... </MCObservable>   (must be simple)
    <MaxMarkovTime> ... </MaxMarkovTime> (optional: default is 4)
    <Verbose/> or <ShowBins/>          (optional)
</Task>
```

6.3.2 `MCBootstraps`

Prints the mean and standard deviation for the observable. Then prints the value of the observable for each of the bootstrap resamplings. The form of the XML for this type must be:

```
<Task>
    <Action>PrintXML</Action>
    <Type>MCBootstraps</Type>
    <MCObservable> ... </MCObservable>
</Task>
```

6.3.3 `MCJackknives`

Prints the estimate using the full ensemble, followed by the average and symmetric error computed from the jackknife resamplings. Then it prints the value of the observable for each of the jackknife resamplings. The form of the XML for this type must be:

```
<Task>
    <Action>PrintXML</Action>
    <Type>MCJackknives</Type>
    <MCObservable> ... </MCObservable>
</Task>
```

6.3.4 `MCHistogram`

The output will consist of the mean and standard deviation for the chosen observable. Then it prints the histogram constructed: the width of each bin, and information about the observable in each bin. Use the optional `<NumberOfBins>` tag to specify the number of bins for the histogram. The XML for this type must be of the form:

```
<Task>
    <Action>PrintXML</Action>
    <Type>MCHistogram</Type>
    <NumberOfBins>25</NumberOfBins>      (optional: default is 40)
    <MCObservable>... </MCObservable>
</Task>
```

6.3.5 `MCBootstrapHistogram`

Prints the mean and standard deviation for the specified observable. Then a histogram is constructed from the values of the observable in each bootstrap resampling. The histogram is then output: the width of each bin, and information about each bin in the histogram. Use the optional `<NumberOfBins>` tag to choose the number of bins for the histogram. The form of the XML for this type is:

```
<Task>
    <Action>PrintXML</Action>
    <Type>MCBootstrapHistogram</Type>
    <NumberOfBins>25</NumberOfBins>      (optional: default is 40)
    <MCObservable>... </MCObservable>
</Task>
```

6.3.6 `MCJackknifeHistogram`

Prints the mean and standard deviation for the specified observable. Then a histogram is constructed from the values of the observable in each jackknife resampling. The histogram is then output: the width of each bin, and information about each bin in the histogram. Use the optional `<NumberOfBins>` tag to choose the number of bins for the histogram. The form of the XML for this type is:

```
<Task>
    <Action>PrintXML</Action>
    <Type>MCJackknifeHistogram</Type>
    <NumberOfBins>25</NumberOfBins>      (optional: default is 40)
    <MCObservable>... </MCObservable>
</Task>
```

6.3.7 `BootstrapIndices`

Prints the available bootstrap indices and the bin indices. XML input must have the form:

```
    <Task>
     <Action>PrintXML</Action>
       <Type>BootstrapIndices</Type>
    </Task>
```

6.3.8 `TemporalCorrelator`

Prints the mean and symmetric error of the temporal correlator on each time slice.

```
<Task>
    <Action>PrintXML</Action>
    <Type>TemporalCorrelator</Type>
    <Correlator> ... </Correlator>
    <Arg>Re</Arg>
    <HermitianMatrix/>                      (optional)
    <SubtractVEV/>                          (optional)
    <SamplingMode>Bootstrap</SamplingMode>  (optional: Jackknife default)
</Task>
```

6.3.9 `EffectiveEnergy`

Prints the mean and symmetric error of the effective energy on each time slice for a particular correlator. The results are found using a particular sampling mode that can be specified in the XML. An effective energy requires a step size (since the effective energy is a discretization of a derivative); this can be specified with the `<TimeStep>` tag, which has a default value of 1. There are four forms of the correlator that can be assumed (specified with the `<EffEnergyType>`):

(1) TimeForward - $C(t) = Ae^{-mt}$
(2) TimeSymmetric - $C(t) = A(e^{-mt} + e^{-m(T-t)})$
(3) TimeForwardPlusConst - $C(t) = Ae^{-mt} + B$
(4) TimeSymmetricPlusConst - $C(t) = A(e^{-mt} + e^{-m(T-t)}) + B$

For each form of the correlator, the results will be such that the effective energy tends to m as t becomes large. Additionally, if you would like to reference these results in a later task, then use the `<EffEnergyIdName>` tag (must be a string with no spaces and 26 characters or less). The XML for this task must be of the form

```
<Task>
    <Action>PrintXML</Action>
    <Type>EffectiveEnergy</Type>
    <EffEnergyType>TimeSymmetric</EffEnergyType> (opt: TimeForward default)
    <TimeStep>3</TimeStep>                       (optional: 1 default)
    <EffEnergyIdName>PionA1um</EffEnergyIdName>  (optional)
    <Correlator>... </Correlator>
    <Arg>Re</Arg>
    <HermitianMatrix/>                      (optional)
    <SubtractVEV/>                          (optional)
    <SamplingMode>Bootstrap</SamplingMode>  (optional: Jackknife default)
</Task>
```

6.4 The `DoPlot` tasks

The `PrintXML` tag allowed information about Monte Carlo observables to be output in XML format, which soon becomes tiresome and ineffective when considering large data sets. It will often be more desirable to produce plots in order to visually display the information. The `DoPlot` task has been included in `sigmond` to fulfill this need. This task makes use of Grace: a free plotting tool for Unix-like operating systems. Thus, in order to use the `DoPlot` task Grace must be installed, which can be found in the package manager of most Linux distributions. Grace was a convenient choice, because it produces human-readable files that allow changes to be made with ease. There are four main types of plots that can be made:

Monte Carlo plots, histograms, temporal correlators plots, and effective energy plots. Most of these plots will use some of the same tags: `<PlotFile>` specifies the name of the file that the plot is stored in (grace plots usually use the `.agr` extension), `<SymbolColor>` specifies the color of the plot symbols to use, and `<SymbolType>` specifies the types of symbols to use for the plot points. For histograms, you can specify the color of the bar with `<BarColor>`.

6.4.1 Monte Carlo plots

This type of `DoPlot` task must have XML of the form

```
<Task>
    <Action>DoPlot</Action>
    <Type>MCValues</Type>                (or MCBootstraps or MCJackknives)
    <MCObservable> ... </MCObservable>
    <PlotFile> ... </PlotFile>
    <ObsName> ... </ObsName>             (optional: none default)
    <SymbolColor> ... </SymbolColor>     (optional: blue default)
    <SymbolType> ... </SymbolType>       (optional: circle default)
    <Rescale> ... </Rescale>             (optional: 1.0 default)
</Task>
```

There are three allowed types here:

– `MCValues` - Plots the estimate for the observable on each gauge configuration. On the plot you will see a solid line representing the mean, and two dotted lines representing the standard deviation.

– `MCBootstraps` - Plots the estimate for the observable on each bootstrap resampling. On the plot you will see a solid line representing the sample average, and two dotted lines representing the error.

– `MCJackknives` - Plots the estimate for the observable on each jackknife resampling. On the plot you will see a solid line representing the sample average, and two dotted lines representing the error (note that many times these errors will be off the plots, because the estimate on each jackknife resampling should be very near the ensemble average).

The `<ObsName>` tag specifies a name to appear on the plot.

6.4.2 Histograms

This type of plot is very similar to the Monte Carlo plots, except that it shows the results as a histogram. The task must have XML of the form

```
<Task>
    <Action>DoPlot</Action>
    <Type>MCHistogram</Type>
    <NumberOfBins>25</NumberOfBins>      <!-- optional: default is 40
    <MCObservable>... </MCObservable>
    <PlotFile> ... </PlotFile>
    <ObsName> ... </ObsName>             (optional: none default)
    <BarColor> ... </BarColor>           (optional: cyan default)
    <Rescale> ... </Rescale>             (optional: 1.0 default)
</Task>
```

where the `<Type>` tag can be `MCHistogram`, `MCBootstrapHistogram`, or `MCJackknifeHistogram`. Use the optional `<NumberOfBins>` tag to choose the number of bins for the histogram. The `<ObsName>` tag specifies a name to appear on the plot.

6.4.3 `TemporalCorrelator`

This task plots a temporal correlator as a function of time. The XML for this task must be of the form

```
<Task>
    <Action>DoPlot</Action>
    <Type>TemporalCorrelator</Type>
    <Correlator>... </Correlator>
    <Arg>Re</Arg>
    <HermitianMatrix/>                    (optional)
    <SubtractVEV/>                        (optional)
    <SamplingMode>Bootstrap</SamplingMode>  (optional: Jackknife default)
    <PlotFile> ... </PlotFile>
    <CorrName> ... </CorrName>            (optional: none default)
    <SymbolColor> ... </SymbolColor>      (optional: blue default)
    <SymbolType> ... </SymbolType>        (optional: circle default)
    <Rescale> ... </Rescale>   (optional: 1.0 default)
</Task>
```

The `<CorrName>` tag is used to give a name for the plot. Writing `standard` between the tags will show the name specifying the correlator, *e.g.* $Re\ C_{AA}(t), A = \pi^{SSO}_{A1um}$.

6.4.4 `TemporalCorrelators`

This task plots the temporal correlators of a set of diagonal correlators as a function of time. The XML for this task must be of the form

```
<Task>
    <Action>DoPlot</Action>
    <Type>TemporalCorrelators</Type>
    <DiagonalCorrelatorSet>... </DiagonalCorrelatorSet>
    <Arg>Re</Arg>
    <HermitianMatrix/>                    (optional)
    <SubtractVEV/>                        (optional)
    <SamplingMode>Bootstrap</SamplingMode>  (optional: Jackknife default)
    <PlotFileStub> ... </PlotFileStub>
    <SymbolColor> ... </SymbolColor>        (optional: blue default)
    <SymbolType> ... </SymbolType>          (optional: circle default)
    <Rescale> ... </Rescale>   (optional: 1.0 default)
</Task>
```

For `<DiagonalCorrelatorSet>` usage, see Sec. 4.3.2.

6.4.5 `TemporalCorrelatorMatrixRescaled`

This tasks plots an entire temporal correlator matrix, rescaling the elements using the correlator matrix at a particular normalization time specified by the tag `<NormTime>`. In other words, the following matrix is plotted: $C_{ij}(t) = \mathcal{C}_{ij}(t)/\sqrt{\mathcal{C}_{ii}(t_N)\mathcal{C}_{jj}(t_N)}$, where $\mathcal{C}$ is the "raw" correlation matrix. The XML for this task must have the form

```
   <Task>
    <Action>DoPlot</Action>
      <Type>TemporalCorrelatorMatrixRescaled</Type>
      <RowInfo>
          <Operator>...</Operator> or <OperatorString>...</OperatorString>
          <FileSuffixLabel>pion_0</FileSuffixLabel> (integer used if absent)
          <PlotLabel>Pi0</PlotLabel>  (standard name used if absent)
      </RowInfo>
         ... other elements ...
      <NormTime>3</NormTime>
      <HermitianMatrix/>   (optional)
      <SubtractVEV/>   (optional)
      <DiagonalVerticalRange>2.0</DiagonalVerticalRange>
         -- all diagonal correlator plots will show -0.1..range
      <OffDiagonalVerticalRange>0.5</OffDiagonalVerticalRange>
         -- all offdiagonal from -range to range
      <SamplingMode>Bootstrap</SamplingMode>  (optional: Jackknife default)
      <PlotFileStub> ... </PlotFileStub>
      <SymbolColor> ... </SymbolColor>     (optional: blue default)
      <SymbolType> ... </SymbolType>       (optional: circle default)
   </Task>
```

6.4.6 `EffectiveEnergy`

This task plots the effective energy as a function of time. The XML for this task must be of the form

```
<Task>
    <Action>DoPlot</Action>
    <Type>EffectiveEnergy</Type>
    <EffEnergyType>TimeForward</EffEnergyType>
    <TimeStep>3</TimeStep>                  (optional: 1 default)
    <Correlator>... </Correlator>
    <Arg>Re</Arg>
    <HermitianMatrix/>                      (optional)
    <SubtractVEV/>                          (optional)
    <SamplingMode>Bootstrap</SamplingMode>  (optional: Jackknife default)
    <PlotFile> ... </PlotFile>
    <CorrName> ... </CorrName>              (optional: none default)
    <SymbolColor> ... </SymbolColor>        (optional: blue default)
    <SymbolType> ... </SymbolType>          (optional: circle default)
    <MaxErrorToPlot> ... </MaxErrorToPlot>  (optional)
</Task>
```

The `<MaxErrorToPlot>` tag specifies the maximum error allowed in plotting a point. For the usage of `<EffEnergyType>` and `<TimeStep>` see Sec. 6.3 on the `PrintXML` task of type `EffectiveEnergy`. See previous `DoPlot` type for usage of `<CorrName>`

6.4.7 `EffectiveEnergies`

This task plots the effective energies of a set of diagonal correlators as a function of time. The XML for this task must be of the form

```
<Task>
    <Action>DoPlot</Action>
    <Type>EffectiveEnergies</Type>
    <EffEnergyType>TimeForward</EffEnergyType>
    <TimeStep>3</TimeStep>                  (optional: 1 default
    <DiagonalCorrelatorSet>... </DiagonalCorrelatorSet>
    <SamplingMode>Bootstrap</SamplingMode>  (optional: Jackknife default)
    <PlotFileStub> ... </PlotFileStub>
    <SymbolColor> ... </SymbolColor>        (optional: blue default)
    <SymbolType> ... </SymbolType>          (optional: circle default)
    <MaxErrorToPlot> ...</MaxErrorToPlot>   (optional)
</Task>
```

For `<DiagonalCorrelatorSet>` usage, see Sec. 4.3.2.

6.5 The `DoFit` tasks

The `DoFit` task is one of the most important tasks that `sigmond` does! For a variety of different data, such as correlation functions, it carries out correlated-χ^2 fits to a variety of different models. These are described below. Four different minimizers are available: `Minuit2`, `LMDer`, `NL2Sol`, and `Minuit2NoGradient`.

6.5.1 `TemporalCorrelator`

The `DoFit` task here fits to temporal correlators using χ^2 minimization. The XML for this task must be of the form:

```
<Task>
    <Action>DoFit</Action>
    <Type>TemporalCorrelator</Type>
    <MinimizerInfo>
        <Method>Minuit2</Method>    (or LMDer, NL2Sol, Minuit2NoGradient)
        <ParameterRelTol>1e-6</ParameterRelTol>
        <ChiSquareRelTol>1e-4</ChiSquareRelTol>
        <MaximumIterations>1024</MaximumIterations>
        <Verbosity>Low</Verbosity>
    </MinimizerInfo>
    <SamplingMode>Bootstrap</SamplingMode>   (optional)
    <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
    <Uncorrelated/>  (optional) performs an uncorrelated fit
    <TemporalCorrelatorFit>
        <Operator>.... </Operator>
        <SubtractVEV/>
        <MinimumTimeSeparation>3</MinimumTimeSeparation>
        <MaximumTimeSeparation>12</MaximumTimeSeparation>
        <ExcludeTimes>4 8</ExcludeTimes>  (optional)
        <LargeTimeNoiseCutoff>1.0</LargeTimeNoiseCutoff>
        <Model>
          ... (See Sec. 6.5.10 for model dependent syntax)
        </Model>
        <DoEffectiveEnergyPlot> (optional)
            <PlotFile> ... </PlotFile>
            <CorrName>standard</CorrName>   (optional)
            <TimeStep>3</TimeStep>  (optional: 1 default)
            <SymbolColor> ... </SymbolColor>
            <SymbolType> ... </SymbolType>
            <MaxErrorToPlot> ...</MaxErrorToPlot> (optional)
            <Goodness>qual</Goodness>  "qual" or "chisq"
            <ShowApproach/>   (optional)
            <ReferenceEnergy> (optional: includes energy ratio on plot)
              <Name>kaon</Name><IDIndex>0</IDIndex>
            </ReferenceEnergy>
        </DoEffectiveEnergyPlot>
    </TemporalCorrelatorFit>
</Task>
```

- All the necessary information given to plotting method (*e.g.* Minuit) is passed through the `<MinimizerInfo>` tag. Inside this tag:

  - `<Method>` tag specifies the minimizer program, currently `Minuit2`, `LMDer`, `NL2Sol` & `Minuit2NoGradient` are supported. If the model is too complicated to provide a gradient routine (such as with phase shift and the RGL shifted zeta functions), use the “Minuit2NoGradient” method to performance minimizations without needing a gradient routine. LMDer and NL2Sol cannot be used in such cases. Minuit2 evaluates the gradient numerically.

  – `<ParameterRelTol>` specifies the relative tolerance of the parameter we are interested in finding out (relative tolerance is defined as a measure of the error relative to the size of each solution component. Roughly, it controls the number of correct digits in all solution components).
  – `<ChiSquareRelTol>` specifies the relative tolerance in the chi-square value we can allow.
  – `<MaximumIterations>` specifies the number of iterations in the minimizer program that we can allow.
  – `<Verbosity>` specifies the amount of information we want the minimizer program to provide us with.

– `<SamplingMode>`: refer to DoPlot.

– We are going to pass the necessary information for plotting the temporal correlator through `<TemporalCorrelatorFit>`.

  – `<MinimumTimeSeparation>` denotes the lower time limit in lattice-time unit.
  – `<MaximumTimeSeparation>` denotes the upper time limit in lattice-time unit.
  – `ExcludeTimes` can be used to exclude individual times from the fit.
  – `<LargeTimeNoiseCutoff>` is used to stop fitting after a the noise gets too large.
  – `<Operator>`: refer to DoPlot.

– `<Model>`: This is a base class and a variety of other classes are derived from here to which correspond different fit functions.

  – `<Type>`: This tag corresponds to different classes in temporal correlator model.

`<DoEffectiveEnergyPlot>` will plot the function after fitting.

– The `<PlotFile>` tag is used to specify the plot file that grace will produce. The file extension agr is commonly used for Grace output files, but it is not required. The file produced will be in a human-readable format, and can be altered as desired. See the Grace User Guide for more information.

– `<CorrName>` is used to give a name for the plot. It is an optional tag. Writing ‘standard’ between the tags is going to show the name specifying the correlator, e.g., $Re\, C_{AA}(t),\ A = \pi^{SSO}_{A1um}$.

– `<TimeStep>` is mentioned to keep record of ’step’ that we used to solve for energy using $C(t + step)/C(t)$.

– `<SymbolColor>` specifies the color of the points on the curve we plot, while `<SymbolType>` specifies the shape, e.g., triangleup (circle is the default shape).

– `<MaxErrorToPlot>` specifies the maximum error allowed in plotting a point.

– `<FitGoodness>` specifies the goodness of fit model used to fit to a function. It can be the quality of the fit or a χ^2/d.o.f..

– `<ShowApproach/>` shows how the effective energy is approached.

– `<ReferenceEnergy>` is for specifying a reference energy for the ratio $E_{\text{fit}}/E_{\text{ref}}$.

6.5.2 `LogTemporalCorrelator`

This task type performs a fit to the log of a single real-valued temporal correlator.

```
<Task>
 <Action>DoFit</Action>
   <Type>LogTemporalCorrelator</Type>
   <MinimizerInfo>                 (optional)
     <Method>Minuit2</Method>
     <ParameterRelTol>1e-6</ParameterRelTol>
     <ChiSquareRelTol>1e-4</ChiSquareRelTol>
     <MaximumIterations>1024</MaximumIterations>
     <Verbosity>Low</Verbosity>
   </MinimizerInfo>
   <SamplingMode>Bootstrap</SamplingMode>   (optional)
   <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
   <Uncorrelated/>  (optional) performs an uncorrelated fit
   <LogTemporalCorrelatorFit>
     <Operator>.... </Operator>
     <SubtractVEV/>             (as appropriate)
     <MinimumTimeSeparation>3</MinimumTimeSeparation>
     <MaximumTimeSeparation>12</MaximumTimeSeparation>
     <ExcludeTimes>4 8</ExcludeTimes>  (optional)
     <LargeTimeNoiseCutoff>1.0</LargeTimeNoiseCutoff>
     <LogModel>
         <Type>LogTimeSymSingleExponential</Type>
         <Energy>
            <Name>pion</Name><IDIndex>0</IDIndex> // default taskcount
         </Energy>
         <LogAmplitude>
            <Name>Amp</Name><IDIndex>0</IDIndex>
         </LogAmplitude>
     </LogModel>
   <DoEffectiveEnergyPlot> (optional)
      <PlotFile> ... </PlotFile>
      <CorrName>standard</CorrName>   (optional)
      <TimeStep>3</TimeStep>  (optional: 1 default)
      <SymbolColor> ... </SymbolColor>
      <SymbolType> ... </SymbolType>
      <MaxRelativeErrorToPlot> ...</MaxRelativeErrorToPlot> (optional)
      <Goodness>qual</Goodness>  "qual" or "chisq"
      <ShowApproach/>   (optional)
      <ReferenceEnergy> (optional: includes energy ratio on plot)
        <Name>kaon</Name><IDIndex>0</IDIndex>
      </ReferenceEnergy>
   </DoEffectiveEnergyPlot>
   </LogTemporalCorrelatorFit>
</Task>
```

6.5.3 `TemporalCorrelatorInteractionRatioFit`

For extracting the interaction energy or energy difference between an interacting energy level and the nearest non-interacting energy, the ratio of correlators

$$R(t) = \frac{C^{\rm int}(t)}{\prod_i C_i^{\rm non-int}(t)},$$

can be fit to the ansatz

$$R(t) = Ae^{-\Delta E t},$$

where $\Delta E = E_{\rm int} - E_{\rm non-int}$, $C^{\rm int}(t)$ is the (diagonal) correlator for the interacting level, and $C_i^{\rm non-int}(t)$ are the N correlators corresponding to the constituents of the closest non-interacting energy. For example, if the closest non-interacting energy to the level from $C^{\rm int}$ is $\pi(0)K(1)K(1)$, then

$$\begin{aligned} C_0^{\rm non-int}(t) &= \text{pion correlator for } \boldsymbol{P}^2 = 0, \\ C_1^{\rm non-int}(t) &= \text{kaon correlator for } \boldsymbol{P}^2 = 1, \\ C_2^{\rm non-int}(t) &= \text{kaon correlator for } \boldsymbol{P}^2 = 1. \end{aligned}$$

The resultant correlator must have (if specified) all VEVs subtracted in this task. The correlator ratio is a non-simple observable so is only available as resamplings.

The required XML has format:

```
<Task>
 <Action>DoFit</Action>
   <Type>TemporalCorrelatorInteractionRatio</Type>
   <MinimizerInfo>                 (optional)
     <Method>Minuit2</Method>
     <ParameterRelTol>1e-6</ParameterRelTol>
     <ChiSquareRelTol>1e-4</ChiSquareRelTol>
     <MaximumIterations>1024</MaximumIterations>
     <Verbosity>Low</Verbosity>
   </MinimizerInfo>
   <SamplingMode>Bootstrap</SamplingMode>   (optional)
```

```
<CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
<Uncorrelated/>  (optional) performs an uncorrelated fit
<TemporalCorrelatorInteractionRatioFit>
  <Ratio>
     <Operator>...</Operator>
  </Ratio>
  <InteractingOperator>
     <Operator>...</Operator>
     <SubtractVEV/>    (optional)
  </InteractingOperator>
  <NonInteractingOperator>
     <Operator>...</Operator>
     <SubtractVEV/>    (optional)
  </NonInteractingOperator>
   ......
  <NonInteractingOperator>
     <Operator>...</Operator>
     <SubtractVEV/>    (optional)
  </NonInteractingOperator>
  <MinimumTimeSeparation>3</MinimumTimeSeparation>
  <MaximumTimeSeparation>12</MaximumTimeSeparation>
  <ExcludeTimes>4 8</ExcludeTimes>  (optional)
  <LargeTimeNoiseCutoff>1.0</LargeTimeNoiseCutoff>
  <Model>
      <Type>TimeSymSingleExponential</Type>
      <Energy>
         <Name>pion</Name><IDIndex>0</IDIndex> // default taskcount
      </Energy>
      <Amplitude>
         <Name>Amp</Name><IDIndex>0</IDIndex>
      </Amplitude>
  </Model>
  <DoEffectiveEnergyPlot> (optional)
     <PlotFile> ... </PlotFile>
     <CorrName>standard</CorrName>   (optional)
     <TimeStep>3</TimeStep>  (optional: 1 default)
     <SymbolColor> ... </SymbolColor>
     <SymbolType> ... </SymbolType>
     <MaxRelativeErrorToPlot> ...</MaxRelativeErrorToPlot> (optional)
     <Goodness>qual</Goodness>  "qual" or "chisq"
     <ShowApproach/>   (optional)
     <ReferenceEnergy> (optional: includes energy ratio on plot)
       <Name>kaon</Name><IDIndex>0</IDIndex>
     </ReferenceEnergy>
  </DoEffectiveEnergyPlot>
  </TemporalCorrelatorInteractionRatioFit>
</Task>
```

6.5.4 TwoTemporalCorrelator

The DoFit task here fits to two temporal correlators simultaneously. The plot is done for the first correlator with the second considered as the reference correlator. In the plot, fit value is given as a ratio of the first energy over the second. The XML for this task must be of the form:

```
<Task>
 <Action>DoFit</Action>
   <Type>TwoTemporalCorrelator</Type>
   <MinimizerInfo>                 (optional)
     <Method>Minuit2</Method>
     <ParameterRelTol>1e-6</ParameterRelTol>
     <ChiSquareRelTol>1e-4</ChiSquareRelTol>
     <MaximumIterations>1024</MaximumIterations>
     <Verbosity>Low</Verbosity>
   </MinimizerInfo>
   <SamplingMode>Bootstrap</SamplingMode>   (optional)
   <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
   <Uncorrelated/>  (optional) performs an uncorrelated fit
   <TwoTemporalCorrelatorFit>
     <CorrelatorOne>
       <Operator>.... </Operator>
       <SubtractVEV/>             (as appropriate)
       <MinimumTimeSeparation>3</MinimumTimeSeparation>
       <MaximumTimeSeparation>12</MaximumTimeSeparation>
       <ExcludeTimes>4 8</ExcludeTimes>  (optional)
       <LargeTimeNoiseCutoff>1.0</LargeTimeNoiseCutoff>
       <Model>
         ... (See Sec. 6.5.10 for model dependent syntax)
       </Model>
     </CorrelatorOne>
     <CorrelatorTwo>
       <Operator>.... </Operator>
       <SubtractVEV/>             (as appropriate)
       <MinimumTimeSeparation>3</MinimumTimeSeparation>
       <MaximumTimeSeparation>12</MaximumTimeSeparation>
       <ExcludeTimes>4 8</ExcludeTimes>  (optional)
       <LargeTimeNoiseCutoff>1.0</LargeTimeNoiseCutoff>
       <Model>
         ... (See Sec. 6.5.10 for model dependent syntax)
       </Model>
```

```
      </CorrelatorTwo>
      <EnergyRatio>
         <Name>pion</Name><IDIndex>0</IDIndex>
      </EnergyRatio>
      <DoEffectiveEnergyPlot>     (plot correlator 1 only)
        <PlotFile> ... </PlotFile>
        <CorrName>standard</CorrName>   (optional)
        <TimeStep>3</TimeStep>  (optional: 1 default)
        <SymbolColor> ... </SymbolColor>
        <SymbolType> ... </SymbolType>
        <MaxErrorToPlot> ...</MaxErrorToPlot> (optional)
        <Goodness>qual</Goodness>  "qual" or "chisq"
        <ShowApproach/>   (optional)
      </DoEffectiveEnergyPlot>
   </TwoTemporalCorrelatorFit>
</Task>
```

- Within `TwoTemporalCorrelatorFit`, correlators 1 and 2 are specified as in the `TemporalCorrelator` fit.
- `EnergyRatio` contains `MCObservable` info for the ratio of correlator fit energies: $E_{\text{Corr 1}}/E_{\text{Corr 2}}$

6.5.5 `TemporalCorrelatorTminVary`

This task does multiple fits to a single temporal correlator using a fixed t_{max} and varying t_{min}. The required XML is

```
<Task>
 <Action>DoFit</Action>
   <Type>TemporalCorrelatorTminVary</Type>
   <MinimizerInfo>                 (optional)
     <Method>Minuit2</Method>
     <ParameterRelTol>1e-6</ParameterRelTol>
     <ChiSquareRelTol>1e-4</ChiSquareRelTol>
     <MaximumIterations>1024</MaximumIterations>
     <Verbosity>Low</Verbosity>
   </MinimizerInfo>
   <SamplingMode>Bootstrap</SamplingMode>   (optional)
   <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
   <Uncorrelated/>  (optional) performs an uncorrelated fit
   <TemporalCorrelatorTminVaryFit>
     <Operator>.... </Operator>
     <SubtractVEV/>             (as appropriate)
     <TminFirst>3</TminFirst>
     <TminLast>3</TminLast>
```

```
  <Tmax>12</Tmax>
  <ExcludeTimes>4 8</ExcludeTimes>  (optional)
  <Model>
      <Type>TimeSymSingleExponential</Type>
      <Energy>
         <Name>pion</Name><IDIndex>0</IDIndex> // default taskcount
      </Energy>
      <Amplitude>
         <Name>Amp</Name><IDIndex>0</IDIndex>
      </Amplitude>
  </Model>
</TemporalCorrelatorTminVaryFit>
<DoEnergyDifference>
  <SpatialExtentNumSites>32</SpatialExtentNumSites>
  <Anisotropy>       (optional)
     <Name>aniso_fit_name</Name>
     <IDIndex>0</IDIndex>
  </Anisotropy>
  <ScatteringParticleEnergyFit>
     <IntMomSquared>4</IntMomSquared>
     <Name>scatting_part_atrest_energy_fit_name</Name>
     <IDIndex>0</IDIndex>
  </ScatteringParticleEnergyFit>
  <ScatteringParticleEnergyFit>
     <IntMomSquared>2</IntMomSquared>
     <Name>scatting_part_atrest_energy_fit_name</Name>
     <IDIndex>0</IDIndex>
  </ScatteringParticleEnergyFit>
     ...
</DoEnergyDifference>
<ChosenFitInfo>
  <Name>fit_obsname</Name>
  <IDIndex>0</IDIndex>
</ChosenFitInfo>
<PlotInfo>
   <PlotFile> ... </PlotFile>
   <CorrName>standard</CorrName>   (optional)
   <SymbolType> ... </SymbolType>
   <GoodFitSymbolColor> ... </GoodFitSymbolColor>
   <BadFitSymbolColor> ... </BadFitSymbolColor>
   <CorrelatedFitSymbolHollow/>  (optional)
   <UncorrelatedFitSymbolHollow/>  (optional)
   <QualityThreshold>qual</QualityThreshold>  (0.1 default)
   <CorrelatedThreshold>1.2</CorrelatedThreshold>  (1.0 default)
</PlotInfo>
```

```
</Task>
```

6.5.6 `LogTemporalCorrelatorTminVary`

This task does multiple fits to the log of a single temporal correlator using a fixed $t_{\max}$ and varying $t_{\min}$. The required XML is

```
<Task>
 <Action>DoFit</Action>
   <Type>LogTemporalCorrelatorTminVary</Type>
   <MinimizerInfo>                 (optional)
     <Method>Minuit2</Method>
     <ParameterRelTol>1e-6</ParameterRelTol>
     <ChiSquareRelTol>1e-4</ChiSquareRelTol>
     <MaximumIterations>1024</MaximumIterations>
     <Verbosity>Low</Verbosity>
   </MinimizerInfo>
   <SamplingMode>Bootstrap</SamplingMode>   (optional)
   <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
   <Uncorrelated/>  (optional) performs an uncorrelated fit
   <LogTemporalCorrelatorTminVaryFit>
     <Operator>.... </Operator>
     <SubtractVEV/>             (as appropriate)
     <TminFirst>3</TminFirst>
     <TminLast>3</TminLast>
     <Tmax>12</Tmax>
     <ExcludeTimes>4 8</ExcludeTimes>  (optional)
     <LogModel>
         <Type>TimeSymSingleExponential</Type>
         <Energy>
            <Name>pion</Name><IDIndex>0</IDIndex> // default taskcount
         </Energy>
         <LogAmplitude>
            <Name>Amp</Name><IDIndex>0</IDIndex>
         </LogAmplitude>
     </LogModel>
   </LogTemporalCorrelatorTminVaryFit>
   <DoEnergyDifference>
     <SpatialExtentNumSites>32</SpatialExtentNumSites>
     <Anisotropy>       (optional)
        <Name>aniso_fit_name</Name>
        <IDIndex>0</IDIndex>
     </Anisotropy>
     <ScatteringParticleEnergyFit>
        <IntMomSquared>4</IntMomSquared>
```

```
         <Name>scatting_part_atrest_energy_fit_name</Name>
         <IDIndex>0</IDIndex>
      </ScatteringParticleEnergyFit>
      <ScatteringParticleEnergyFit>
         <IntMomSquared>2</IntMomSquared>
         <Name>scatting_part_atrest_energy_fit_name</Name>
         <IDIndex>0</IDIndex>
      </ScatteringParticleEnergyFit>
         ...
    </DoEnergyDifference>
    <ChosenFitInfo>
      <Name>fit_obsname</Name>
      <IDIndex>0</IDIndex>
    </ChosenFitInfo>
    <PlotInfo>
       <PlotFile> ... </PlotFile>
       <CorrName>standard</CorrName>   (optional)
       <SymbolType> ... </SymbolType>
       <GoodFitSymbolColor> ... </GoodFitSymbolColor>
       <BadFitSymbolColor> ... </BadFitSymbolColor>
       <CorrelatedFitSymbolHollow/>  (optional)
       <UncorrelatedFitSymbolHollow/>  (optional)
       <QualityThreshold>qual</QualityThreshold>  (0.1 default)
       <CorrelatedThreshold>1.2</CorrelatedThreshold>  (1.0 default)
    </PlotInfo>
</Task>
```

6.5.7 TemporalCorrelatorInteractionRatioTminVary

This task does multiple fits to the ratio of correlation functions, as described above, using a fixed t_{max} and varying t_{min}. The required XML is

```
<Task>
 <Action>DoFit</Action>
   <Type>TemporalCorrelatorInteractionRatioTminVary</Type>
   <MinimizerInfo>                 (optional)
     <Method>Minuit2</Method>
     <ParameterRelTol>1e-6</ParameterRelTol>
     <ChiSquareRelTol>1e-4</ChiSquareRelTol>
     <MaximumIterations>1024</MaximumIterations>
     <Verbosity>Low</Verbosity>
   </MinimizerInfo>
   <SamplingMode>Bootstrap</SamplingMode>   (optional)
   <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
   <Uncorrelated/>  (optional) performs an uncorrelated fit
```

```
<TemporalCorrelatorInteractionRatioTminVaryFit>
  <Ratio>
     <Operator>...</Operator>
  </Ratio>
  <InteractingOperator>
     <Operator>...</Operator>
     <SubtractVEV/>    (optional)
  </InteractingOperator>
  <NonInteractingOperator>
     <Operator>...</Operator>
     <SubtractVEV/>    (optional)
  </NonInteractingOperator>
   ......
  <NonInteractingOperator>
     <Operator>...</Operator>
     <SubtractVEV/>    (optional)
  </NonInteractingOperator>
  <TminFirst>3</TminFirst>
  <TminLast>3</TminLast>
  <Tmax>12</Tmax>
  <ExcludeTimes>4 8</ExcludeTimes>  (optional)
  <Model>
      <Type>TimeSymSingleExponential</Type>
      <Energy>
         <Name>pion</Name><IDIndex>0</IDIndex> // default taskcount
      </Energy>
      <Amplitude>
         <Name>Amp</Name><IDIndex>0</IDIndex>
      </Amplitude>
  </Model>
</TemporalCorrelatorTminVaryFit>
<DoReconstructEnergy>
  <SpatialExtentNumSites>32</SpatialExtentNumSites>
  <Anisotropy>       (optional)
     <Name>aniso_fit_name</Name>
     <IDIndex>0</IDIndex>
  </Anisotropy>
  <ScatteringParticleEnergyFit>
     <IntMomSquared>4</IntMomSquared>
     <Name>scatting_part_atrest_energy_fit_name</Name>
     <IDIndex>0</IDIndex>
  </ScatteringParticleEnergyFit>
  <ScatteringParticleEnergyFit>
     <IntMomSquared>2</IntMomSquared>
     <Name>scatting_part_atrest_energy_fit_name</Name>
```

```
        <IDIndex>0</IDIndex>
      </ScatteringParticleEnergyFit>
        ...
    </DoReconstructEnergy>
    <ChosenFitInfo>
      <Name>fit_obsname</Name>
      <IDIndex>0</IDIndex>
    </ChosenFitInfo>
    <PlotInfo>
       <PlotFile> ... </PlotFile>
       <CorrName>standard</CorrName>   (optional)
       <SymbolType> ... </SymbolType>
       <GoodFitSymbolColor> ... </GoodFitSymbolColor>
       <BadFitSymbolColor> ... </BadFitSymbolColor>
       <CorrelatedFitSymbolHollow/>  (optional)
       <UncorrelatedFitSymbolHollow/>  (optional)
       <QualityThreshold>qual</QualityThreshold>  (0.1 default)
       <CorrelatedThreshold>1.2</CorrelatedThreshold>  (1.0 default)
    </PlotInfo>
</Task>
```

6.5.8 AnisotropyFromDispersion

In this task, a fit to the free-particle energies for various $\boldsymbol{P}^2$ is used to estimate the lattice anisotropy $\xi = a_s/a_t$. The fit model used for the observables is:

$$(a_tE)^2 = m_0^2 + \left(\frac{2\pi}{N_s}\right)^2 \frac{\boldsymbol{n}^2}{\xi^2}$$

where m_0^2 and ξ are the two model parameters, N_s is the spatial extent of the lattice in terms of number of sites in each of the three spatial directions, and $\boldsymbol{n}^2 \in \boldsymbol{N}$ is the integer square of the three momentum. Recall that $(a_s\boldsymbol{P})^2 = (2\pi/N_s)^2\,\boldsymbol{n}^2$. The XML must be of the form:

```
<Task>
 <Action>DoFit</Action>
    <Type>AnisotropyFromDispersion</Type>
    <MinimizerInfo>                  (optional)
      <Method>Minuit2</Method>
      <ParameterRelTol>1e-6</ParameterRelTol>
      <ChiSquareRelTol>1e-4</ChiSquareRelTol>
      <MaximumIterations>1024</MaximumIterations>
      <Verbosity>Low</Verbosity>
    </MinimizerInfo>
    <SamplingMode>Bootstrap</SamplingMode>   (optional)
    <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
    <Uncorrelated/>  (optional) performs an uncorrelated fit
    <AnisotropyFromDispersionFit>
      <SpatialExtentNumSites>24</SpatialExtentNumSites>
      <Energy>
        <Name>pion</Name><IDIndex>0</IDIndex>
        <IntMomSquared>0</IntMomSquared>
      </Energy>
      <Energy>
        <Name>pion</Name><IDIndex>1</IDIndex>
        <IntMomSquared>1</IntMomSquared>
      </Energy>
      <Energy>... </Energy>
      ...
      <Anisotropy>
        <Name>PionXi</Name><IDIndex>0</IDIndex>
      </Anisotropy>
      <RestMassSquared>
        <Name>PionRestMassSquared</Name><IDIndex>0</IDIndex>
      </RestMassSquared>
      <DoPlot>
        <PlotFile> ... </PlotFile>
        <ParticleName>pion</ParticleName>   (optional)
        <SymbolColor> ... </SymbolColor>
        <SymbolType> ... </SymbolType>
        <Goodness>qual</Goodness>  "qual" or "chisq"
      </DoPlot>
    </AnisotropyFromDispersionFit>
</Task>
```

– `SpatialExtentNumSites` is required to specify N_s, the number of lattice sites in the spatial directions.

– Each energy (already in memory) to be included in the fit must be specified in an `Energy` tag with $\boldsymbol{n}^2$ specified by `IntMomSquared`.

– The two model parameters, ξ and m_0^2, are specified by `Anisotropy` and `RestMassSquared` respectively.

– `DoPlot` produces a plot of $(a_t E)^2$ against $\boldsymbol{n}^2$ including the fit information.

6.5.9 `LatticeDispersionRelation`

Fit the free-particle energies squared for various three-momenta squared to the lattice dispersion relation

$$\sinh^2(a_t E) = \sinh^2(m_{\text{rest}}) + d \sum_j \sin^2(2\pi n_j / N_s)$$

where the rest mass m_{rest} and displacement length d are the two model parameters, and N_s is extent of the lattice in terms of number of sites in each of the three spatial directions, and n is the vector of integers describing the momentum direction. Recall that $P_i = 2\pi n_i/L$ with $L = N_s a_s$. The XML input should have the form

```
<Task>
 <Action>DoFit</Action>
   <Type>LatticeDispersionRelation</Type>
   <MinimizerInfo>                 (optional)
     <Method>Minuit2</Method>
     <ParameterRelTol>1e-6</ParameterRelTol>
     <ChiSquareRelTol>1e-4</ChiSquareRelTol>
     <MaximumIterations>1024</MaximumIterations>
     <Verbosity>Low</Verbosity>
   </MinimizerInfo>
   <SamplingMode>Bootstrap</SamplingMode>   (optional)
   <CovMatCalcSamplingMode>Bootstrap</CovMatCalcSamplingMode> (optional)
   <Uncorrelated/>  (optional) performs an uncorrelated fit
   <LatticeDispersionRelationFit>
     <SpatialExtentNumSites>24</SpatialExtentNumSites>
     <Energy>
       <Name>pion</Name><IDIndex>0</IDIndex>
       <IntMom>0 0 0</IntMom>
     </Energy>
     <Energy>
       <Name>pion</Name><IDIndex>1</IDIndex>
       <IntMom>0 0 1</IntMom>
     </Energy>
     <Energy>... </Energy>
     <RestMass>
       <Name>PionRestMassSquared</Name><IDIndex>0</IDIndex>
     </RestMass>
     <Disp>
       <Name>DispTerm</Name><IDIndex>0</IDIndex>
     </Disp>
     <DoPlot>
       <PlotFile> ... </PlotFile>
       <ParticleName>pion</ParticleName>   (optional)
       <SymbolColor> ... </SymbolColor>
       <SymbolType> ... </SymbolType>
       <Goodness>qual</Goodness>  "qual" or "chisq"
     </DoPlot>
   </LatticeDispersionRelationFit>
</Task>
```

6.5.10 Fit models

Currently implemented fit forms for correlator and log of correlator fits are listed below.

– `TimeForwardSingleExponential` - fit function: Ae^{-mt}

```
<Model>
    <Type>TimeForwardSingleExponential</Type>
    <Energy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </Energy>
    <Amplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </Amplitude>
</Model>
```

`<Energy>` tag specifies the name and ID index by which the operator can be recalled. Name and ID index of the `<Amplitude>` tag specifies the constant A .

– `TimeSymSingleExponential` - fit function: $A(e^{-mt} + e^{-m(T-t)})$

```
<Model>
    <Type>TimeSymSingleExponential</Type>
    <Energy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </Energy>
    <Amplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </Amplitude>
</Model>
```

– `TimeForwardSingleExponentialPlusConstant` - fit function: $Ae^{-mt} + B$

```
<Model>
    <Type>TimeForwardSingleExponentialPlusConstant</Type>
    <Energy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </Energy>
    <Amplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </Amplitude>
    <AddedConstant>
        <Name>B</Name><IDIndex>0</IDIndex>
    </AddedConstant>
</Model>
```

We have AddedConstant tag here to specify an extra constant B.

– TimeSymSingleExponentialPlusConstant - fit function: $A(e^{-mt} + e^{-m(T-t)}) + B$

```
<Model>
    <Type>TimeSymSingleExponentialPlusConstant</Type>
    <Energy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </Energy>
    <Amplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </Amplitude>
    <AddedConstant>
        <Name>B</Name><IDIndex>0</IDIndex>
    </AddedConstant>
</Model>
```

– TimeForwardTwoExponential - fit function: $Ae^{-mt}(1 + Be^{-D^2t})$

```
<Model>
    <Type>TimeForwardTwoExponential</Type>
    <FirstEnergy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </FirstEnergy>
    <FirstAmplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </FirstAmplitude>
    <SqrtGapToSecondEnergy>
        <Name>pionprime</Name><IDIndex>0</IDIndex>
    </SqrtGapToSecondEnergy>
    <SecondAmplitudeRatio>
        <Name>B</Name><IDIndex>0</IDIndex>
    </SecondAmplitudeRatio>
</Model>
```

FirstEnergy tag specifies the ground state energy (m, in the expression above) and SqrtGapToSecondEnergy specifies D in the expression which is related to the first excited state. SecondAmplitudeRatio specifies B in the expression above.

– TimeSymTwoExponential - fit function:
$Ae^{-mt}(1 + Be^{-D^2t}) + Ae^{-m(T-t)}(1 + Be^{-D^2(T-t)})$

```
<Model>
    <Type>TimeSymTwoExponential</Type>
    <FirstEnergy>
```

```
        <Name>pion</Name><IDIndex>0</IDIndex>
    </FirstEnergy>
    <FirstAmplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </FirstAmplitude>
    <SqrtGapToSecondEnergy>
        <Name>pionprime</Name><IDIndex>0</IDIndex>
    </SqrtGapToSecondEnergy>
    <SecondAmplitudeRatio>
        <Name>B</Name><IDIndex>0</IDIndex>
    </SecondAmplitudeRatio>
</Model>
```

– `TimeForwardTwoExponentialPlusConstant` - fit function: $Ae^{-mt}(1 + Be^{-D^2t}) + C$

```
<Model>
    <Type>TimeForwardTwoExponentialPlusConstant</Type>
    <FirstEnergy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </FirstEnergy>
    <FirstAmplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </FirstAmplitude>
    <SqrtGapToSecondEnergy>
        <Name>pionprime</Name><IDIndex>0</IDIndex>
    </SqrtGapToSecondEnergy>
    <SecondAmplitudeRatio>
        <Name>B</Name><IDIndex>0</IDIndex>
    </SecondAmplitudeRatio>
    <AddedConstant>
        <Name>C</Name><IDIndex>0</IDIndex>
    </AddedConstant>
</Model>
```

– `TimeSymTwoExponentialPlusConstant` - fit function:
$A[e^{-mt}(1 + Be^{-D^2t}) + e^{-m(T-t)}(1 + Be^{-D^2(T-t)})] + C$

```
<Model>
    <Type>TimeSymTwoExponentialPlusConstant</Type>
    <FirstEnergy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </FirstEnergy>
    <FirstAmplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </FirstAmplitude>
    <SqrtGapToSecondEnergy>
        <Name>pionprime</Name><IDIndex>0</IDIndex>
    </SqrtGapToSecondEnergy>
    <SecondAmplitudeRatio>
        <Name>B</Name><IDIndex>0</IDIndex>
    </SecondAmplitudeRatio>
    <AddedConstant>
        <Name>C</Name><IDIndex>0</IDIndex>
    </AddedConstant>
</Model>
```

– `TimeForwardGeomSeriesExponential` - fit function: $\frac{Ae^{-mt}}{1-Be^{-D^2t}}$

```
<Model>
    <Type>TimeForwardGeomSeriesExponential</Type>
    <FirstEnergy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </FirstEnergy>
    <FirstAmplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </FirstAmplitude>
    <SqrtGapToSecondEnergy>
        <Name>pionprime</Name><IDIndex>0</IDIndex>
    </SqrtGapToSecondEnergy>
    <SecondAmplitudeRatio>
        <Name>B</Name><IDIndex>0</IDIndex>
    </SecondAmplitudeRatio>
</Model>
```

– `TimeSymGeomSeriesExponential` - fit function: $A\left(\frac{e^{-mt}}{1-Be^{-D^2t}}+\frac{e^{-m(T-t)}}{1-Be^{-D^2(T-t)}}\right)$

```
<Model>
    <Type>TimeSymGeomSeriesExponential</Type>
    <FirstEnergy>
        <Name>pion</Name><IDIndex>0</IDIndex>
    </FirstEnergy>
    <FirstAmplitude>
        <Name>A</Name><IDIndex>0</IDIndex>
    </FirstAmplitude>
    <SqrtGapToSecondEnergy>
        <Name>pionprime</Name><IDIndex>0</IDIndex>
    </SqrtGapToSecondEnergy>
    <SecondAmplitudeRatio>
        <Name>B</Name><IDIndex>0</IDIndex>
    </SecondAmplitudeRatio>
</Model>
```

– `LogTimeForwardSingleExponential` - fit function: $\ln(A) - mt$

```
<Model>
    <Type>LogTimeForwardSingleExponential</Type>
    <Energy>
       <Name>pion</Name><IDIndex>0</IDIndex>
    </Energy>
    <LogAmplitude>
       <Name>Amp</Name><IDIndex>0</IDIndex>
    </LogAmplitude>
</Model>
```

– `LogTimeForwardTwoExponential` -fit function: $\ln(A) - mt + \ln(1 + Be^{-D^2 t})$

```
<Model>
    <Type>LogTimeForwardTwoExponential</Type>
    <FirstEnergy>
       <Name>pion</Name><IDIndex>0</IDIndex>
    </FirstEnergy>
    <LogFirstAmplitude>
       <Name>Amp0</Name><IDIndex>0</IDIndex>
    </LogFirstAmplitude>
    <SqrtGapToSecondEnergy>
       <Name>pionprime</Name><IDIndex>0</IDIndex>
    </SqrtGapToSecondEnergy>
    <SecondAmplitudeRatio>
       <Name>Amp1</Name><IDIndex>0</IDIndex>
    </SecondAmplitudeRatio>
</Model>
```

6.6 The `DoChecks` task

This task is designed for checking the correlator data for any possible errors or corruption. The correlator matrix to check is specified with the `<CorrelatorMatrix>` tag. The `<MinTimeSep>` and `<MaxTimeSep>` tags are used to specify the temporal range to consider.

6.6.1 `TemporalCorrelationMatrix`

This `DoChecks` task checks that correlators and VEVs, if required, are present, and checks each observable for "outliers", which might indicate corrupt data. Outliers are identified as follows: the data is sorted, then the data range $a \dots b$ for the middle half of the data points is found. Let the mid-range be $m = (a + b)/2$ then outliers are points outside the range $m - v \dots m + v$ where $v = s(b - a)/2$, and s is an outlier scale specified with the `<OutlierScale>` tag. Additionally this task checks for any VEVs or diagonal correlators at the minimum time separation that are consistent with zero. The XML must be of the form

```
<Task>
    <Action>DoChecks</Action>
    <Type>TemporalCorrelatorMatrix</Type>
    <CorrelatorMatrix>
        <Operator>...</Operator>
        <Operator>...</Operator>
             ...
        <HermitianMatrix/>
        <SubtractVEV/>                    (optional)
    </CorrelatorMatrix>
    <MinTimeSep>3</MinTimeSep>
    <MaxTimeSep>25</MaxTimeSep>
    <Verbose/>                            (optional)
    <OutlierScale>12.5</OutlierScale>   (optional: default 9.5)
</Task>
```

6.6.2 TemporalCorrelationMatrixIsHermitian

This DoChecks task checks whether the temporal correlation matrix is Hermitian. The XML must be of the form

```
<Task>
    <Action>DoChecks</Action>
    <Type>TemporalCorrelatorMatrixIsHermitian</Type>
    <CorrelatorMatrix>
        <Operator>...</Operator>
        <Operator>...</Operator>
             ...
    </CorrelatorMatrix>
    <MinTimeSep>3</MinTimeSep>
    <MaxTimeSep>25</MaxTimeSep>
    <Verbose/>                          (optional)
</Task>
```

6.7 The DoObsFunction task

Once you have calculated a few observables, you may be interested in combining them in some way. The DoObsFunction task has been implemented for this purpose. The task makes use of a <Result> tag in order to specify how the newly formed observable is to be stored in memory. Recall from Sec. 4.4 how secondary Monte Carlo observables are specified; the <Name> and <IDIndex> tags (which will be inside the <Result> tag) are used as the identifying information for the secondary observable that will be stored in memory.

In addition, the DoObsFunction task may be used to form linear combinations of multiple correlator matrices for various purposes. eg. averaging matrices of equivalent total $\boldsymbol{P}^2$. The current available operations are:

6.7.1 `Ratio`

This type provides a way to compute ratios of observables. Commonly, the energies we extract are expressed as ratios over the kaon mass. This task makes determining these ratios very simple. The XML must be of the form:

```
<Task>
    <Action>DoObsFunction</Action>
    <Type>Ratio</Type>
    <Result>
        <Name>result-name</Name>
        <IDIndex>0</IDIndex>
    </Result>
    <Numerator>
        <MCObservable> ... </MCObservable>
    </Numerator>
    <Denominator>
        <MCObservable> ... </MCObservable>
    </Denominator>
    <Mode>Jackknife</Mode>       (optional)
      (or Bootstrap or Current [default] or Bins)
</Task>
```

6.7.2 `LinearSuperposition`

This type produces a linear combination of observables of your choosing (note that the coefficients must be real). The XML must be of the form:

```
<Task>
    <Action>DoObsFunction</Action>
    <Type>LinearSuperposition</Type>
    <Result>
        <Name>result-name</Name>
        <IDIndex>0</IDIndex>
    </Result>
    <Summand>
        <MCObservable> ... </MCObservable>
        <Coefficient>3.2</Coefficient>      (must be real)
    </Summand>
    <Summand>
        <MCObservable> ... </MCObservable>
        <Coefficient>-5.7</Coefficient>     (must be real)
    </Summand>
        ...
    <Mode>Jackknife</Mode>       (optional)
      (or Bootstrap or Current [default] or Bins)
</Task>
```

6.7.3 `BoostEnergy`

This type takes an energy and boosts it to a new frame. For boosting an energy specified in the `<FrameEnergy>` tag to a frame new frame using the momentum specified by the `<IntMomSquared>` tag. If the `<BoostToCM/>` tag is present it is assumed that the energy is in a lab frame specified by the `<IntMomSquared>` tag and the resulting energy is boosted to the center of momentum frame. The XML must be of the form:

```
<Task>
    <Action>DoObsFunction</Action>
    <Type>BoostEnergy</Type>
    <BoostToCM/>     (optional)
    <Result>
        <Name>result-name</Name>
        <IDIndex>0</IDIndex>
    </Result>
    <IntMomSquared>4</IntMomSquared>
    <SpatialExtentNumSites>32</SpatialExtentNumSites>
    <FrameEnergy>
        <MCObservable> ... </MCObservable>
    </FrameEnergy>
    <Anisotropy>
        <MCObservable> ... </MCObservable>
    </Anisotropy>
    <Mode>Jackknife</Mode> (optional)
      (or Bootstrap or Current [default] or Bins )
    <ReferenceEnergy>   (optional)
        <MCObservable> ... </MCObservable>
    </ReferenceEnergy>
</Task>
```

6.7.4 `CorrelatorMatrixTimeDifference`

This type produces a time subtracted correlator matrix as to remove the need for VEV subtraction. For a given correlator matrix, $C(t) - C(t+1)$ is computed. The XML must be of the form:

```
<Task>
   <Action>DoObsFunction</Action>
   <Type>CorrelatorMatrixTimeDifferences</Type>
   <NewOperatorOrderedList>
     <Operator>...</Operator>
         ...
   </NewOperatorOrderedList>
   <OriginalOperatorOrderedList>
     <Operator>...</Operator>
```

```
         ...
   </OriginalOperatorOrderedList>
   <MinimumTimeSeparation>3</MinimumTimeSeparation>
   <MaximumTimeSeparation>12</MaximumTimeSeparation>
   <HermitianMatrix/>  (if hermitian)
   <Mode>samplings</Mode> ( or bins )
   <WriteToFile>
      <FileName>name</FileName>
      <FileType>bins</FileType> (or samplings)
      <WriteMode>overwrite</WriteMode> (protect, update, overwrite)
   </WriteToFile>
</Task>
```

`<Mode>` and `<FileType>` must be the same if both are present; however, both do not need to be present. At least one of these tags must be present.

6.7.5 `CorrelatorMatrixSuperposition`

This type combines several correlation matrices to make a resultant matrix:

$$R_{ij} = \sum_k d[k]_i d[k]_j C[k]_{ij}$$

where k labels the different matrices, i and j are the row and column, and the superposition coefficients are specified in $d[k]_i$. The linear superposition coefficients are given at the operator level. Each correlator matrix to combine must be given in an `<OperatorOrderedList>` tag, which includes `<Item>...` tags. Each item in each list must have an `<Operator>` tag, and an optional `<Coefficient>` tag; if absent, a coefficient equal to 1 is assumed. All coefficients must be real. The XML must be of the form:

```
<Task>
 <Action>DoObsFunction</Action>
   <Type>CorrelatorMatrixSuperposition</Type>
   <ResultOperatorOrderedList>
     <Operator>...</Operator>
         ...
   </ResultOperatorOrderedList>
   <OperatorOrderedList>
     <Item><Operator>...</Operator><Coeffient>-1.0</Coefficient></Item>
         ...
   </OperatorOrderedList>
   <OperatorOrderedList>
          ...
   </OperatorOrderedList>
          ...
   <MinimumTimeSeparation>3</MinimumTimeSeparation>
   <MaximumTimeSeparation>12</MaximumTimeSeparation>
   <HermitianMatrix/>  (if hermitian)
   <Mode>samplings</Mode> ( or bins )
   <WriteToFile>
      <FileName>name</FileName>
      <FileType>bins</FileType> (or samplings)
      <WriteMode>overwrite</WriteMode> (protect, update, overwrite)
   </WriteToFile>
</Task>
```

`<Mode>` and `<FileType>` must be the same if both are present; however, both do not need to be present. At least one of these tags must be present.

6.7.6 `TransformCorrelatorMatrix`

This task performs the following transformation on a correlation matrix C:

$$C^{\rm trans}_{ij} = T^{\dagger}_{ik} C_{km} T_{mj}.$$

If C is $N \times N$, then the transformation matrix T must be $N \times M$, where $M \leq N$. In terms of operators, this means the transformed operators W_k are given in terms of the original operators O_m by

$$\bar{W}_j = \bar{O}_m T_{mj}, \qquad W_i = T^{\dagger}_{ik} O_k.$$

The input XML must have the form

```
  <Task>
   <Action>DoObsFunction</Action>
     <Type>TransformCorrelatorMatrix</Type>
       <TransformedOperator>   (creation operator)
          <OpName><Operator>...</Operator></OpName>
          <OpTerm>
               <Operator>...</Operator>
               <Coefficient>(0.43,-0.121)</Coefficient>
          </OpTerm>
           ... other opterms
       </TransformedOperator>
          ...   other transformed operators
     <MinimumTimeSeparation>3</MinimumTimeSeparation>
     <MaximumTimeSeparation>12</MaximumTimeSeparation>
     <HermitianMatrix/>  (if hermitian)
     <SubtractVEV/> (optional)
     <Mode>samplings</Mode> ( or bins )
     <WriteToFile>
        <FileName>name</FileName>
        <FileType>bins</FileType> (or samplings)
        <WriteMode>overwrite</WriteMode> (protect, update, overwrite)
        <SeparateVEVWrite/> (optional for FileType=samplings: default no)
```

```
    </WriteToFile>
  </Task>
```

`<Mode>` and `<FileType>` must be the same if both are present; however, both do not need to be present. At least one of these tags must be present.

6.7.7 `ReconstructEnergy`

This task obtains the total energy after having extracted the energy difference from a ratio fit. The required XML must have the format:

```
<Task>
 <Action>DoObsFunction</Action>
   <Type>ReconstructEnergy</Type>
   <SpatialExtentNumSites>32</SpatialExtentNumSites>
   <Anisotropy>       (optional)
      <Name>aniso_fit_name</Name>
      <IDIndex>0</IDIndex>
   </Anisotropy>
   <Result>
      <Name>result-name</Name>
      <IDIndex>0</IDIndex>
   </Result>
   <EnergyDifferenceFit>
      <Name>diff_fit_name</Name>
      <IDIndex>0</IDIndex>
   </EnergyDifferenceFit>
   <ScatteringParticleEnergyFit>
      <IntMomSquared>4</IntMomSquared>
      <Name>scatting_part_atrest_energy_fit_name</Name>
      <IDIndex>0</IDIndex>
   </ScatteringParticleEnergyFit>
   <ScatteringParticleEnergyFit>
      <IntMomSquared>2</IntMomSquared>
      <Name>scatting_part_atrest_energy_fit_name</Name>
      <IDIndex>0</IDIndex>
   </ScatteringParticleEnergyFit>
      ...
   <Mode>samplings</Mode> (default: current sampling method)
                (or Bootstrap or Jackknife )
</Task>
```

6.7.8 `ReconstructAmplitude`

This task obtains the amplitude of the original correlator extracting an amplitude from a fit to a ratio correlator. The required XML is

```
<Task>
 <Action>DoObsFunction</Action>
   <Type>ReconstructAmplitude</Type>
   <Result>
      <Name>result-name</Name>
      <IDIndex>0</IDIndex>
   </Result>
   <EnergyDifferenceAmplitudeFit>
      <Name>diff_fit_name</Name>
      <IDIndex>0</IDIndex>
   </EnergyDifferenceAmplitudeFit>
   <ScatteringParticleAmplitudeFit>
      <Name>scatting_part_atrest_amp_fit_name</Name>
      <IDIndex>0</IDIndex>
   </ScatteringParticleAmplitudeFit>
   <ScatteringParticleAmplitudeFit>
      <Name>scatting_part_atrest_amp_fit_name</Name>
      <IDIndex>0</IDIndex>
   </ScatteringParticleAmplitudeFit>
      ...
   <Mode>samplings</Mode> (default: current sampling method)
                  (or Bootstrap or Jackknife )
</Task>
```

6.7.9 EnergyDifference

This task is used for getting the difference (subtraction) between the Monte Carlo estimates of two energies. The XML format is

```
<Task>
 <Action>DoObsFunction</Action>
   <Type>EnergyDifference</Type>
   <SpatialExtentNumSites>32</SpatialExtentNumSites>
   <Anisotropy>       (optional)
      <Name>aniso_fit_name</Name>
      <IDIndex>0</IDIndex>
   </Anisotropy>
   <Result>
      <Name>result-name</Name>
      <IDIndex>0</IDIndex>
   </Result>
   <EnergyFit>
      <Name>fit_name</Name>
      <IDIndex>0</IDIndex>
   </EnergyFit>
   <ScatteringParticleEnergyFit>
      <IntMomSquared>4</IntMomSquared>
      <Name>scatting_part_atrest_energy_fit_name</Name>
      <IDIndex>0</IDIndex>
   </ScatteringParticleEnergyFit>
   <ScatteringParticleEnergyFit>
      <IntMomSquared>2</IntMomSquared>
      <Name>scatting_part_atrest_energy_fit_name</Name>
      <IDIndex>0</IDIndex>
   </ScatteringParticleEnergyFit>
      ...
   <Mode>samplings</Mode> (default: current sampling method)
                  (or Bootstrap or Jackknife )
</Task>
```

6.7.10 Exp

This task evaluates the Monte Carlo estimate of the exponential of some other Monte Carlo estimate. The required XML is

```
<Task>
 <Action>DoObsFunction</Action>
   <Type>Exp</Type>
   <Result>
      <Name>result-name</Name><IDIndex>0</IDIndex>
   </Result>
   <InObservable><MCObservable> ... </MCObservable></InObservable>
   <Mode>samplings</Mode> (default: current sampling method)
                  (or Bootstrap or Jackknife or bins )
</Task>
```

6.7.11 Log

This task evaluates the Monte Carlo estimate of the natural logarithm of some other Monte Carlo estimate. The required XML is

```
<Task>
 <Action>DoObsFunction</Action>
   <Type>Log</Type>
   <Result>
      <Name>result-name</Name><IDIndex>0</IDIndex>
   </Result>
   <InObservable><MCObservable> ... </MCObservable></InObservable>
   <Mode>samplings</Mode> (default: current sampling method)
                  (or Bootstrap or Jackknife or bins )
</Task>
```

6.7.12 `Copy`

This task is useful for copying data from one observable to another. The required XML is

```
<Task>
 <Action>DoObsFunction</Action>
   <Type>Copy</Type>
   <Result>
      <Name>result-name</Name><IDIndex>0</IDIndex>
   </Result>
   <InObservable><MCObservable> ... </MCObservable></InObservable>
   <Mode>samplings</Mode> (default: current sampling method)
                 (or Bootstrap or Jackknife or bins )
</Task>
```

6.8 Correlation Matrix Analysis

When one wishes to analyze a correlation matrix for the energy spectrum and the overlap amplitude factors, it is not feasible to directly fit to the entire "raw " correlation matrix given the (usually) large number of parameters that would be needed. Instead, we first perform a *rotation* or *diagonalization* of the matrix to make the extraction of energies and overlap factors by least squares fitting practical. We have developed three approaches for this extraction procedure that use a so-called *Correlation Matrix Pivot*: the Principal Axes method, the Rolling Pivot method and the Single Pivot method. However, only the single pivot method is currently available in `sigmond`.

To apply the Single Pivot method, one first creates a single pivot using the tag `<SinglePivotInitiate>`. The result of this initiation can be saved to file with a `<WritePivotToFile>` tag for subsequent use in other runs, or the pivot can be directly used. Later runs can initiate a pivot by reading from file with a `<ReadPivotFromFile>` tag, or if it is already stored in persistent memory from a previous task, a `<GetFromMemory>` tag can be used. Regardless of how the pivot is initialized, the second step is to use the pivot to rotate a correlation matrix using the vb¡WriteRotatedCorrToFile¿ tag. The third step is to perform fits to the diagonal elements of the rotated correlation matrix. These fit results can be saved to file, especially the amplitude factors. The last step is to use the pivot information and the fit amplitudes to determine the operator overlap Z factors using a `<DoCorrMatrixZMagSquares>` tag. The initial ordering of the levels is based on the diagonalization at time separation τ_D. However, this ordering may not agree with that from the final fit energies. By inserting fit energy information for all levels, the level ordering can be changed to agree with increasing fit energy.

For given Hermitian correlation matrix, three time slices are chosen: $\tau_N \leq \tau_0 < \tau_D$. The matrix at rescaling time τ_N is used to rescale the correlation matrix:

$$C_{ij}(t) = \mathcal{C}_{ij}(t) / \Big(\mathcal{C}_{ii}(\tau_N) \mathcal{C}_{jj}(\tau_N) \Big)^{1/2},$$

where $\mathcal{C}$ is the "raw" correlation matrix. For metric time τ_0 and diagonalization time τ_D, the following procedure is followed:

1. The eigenvalues and eigenvectors of the $N \times N$ matrix $C(\tau_0)$ are determined using the full ensemble of configurations. Let $L_{0\max}$ denote the eigenvalue of largest magnitude. Put the $N_0 \leq N$ eigenvectors associated with eigenvalues greater than $L_{0\max}\times$ `MinimumInverseConditionNumber` into the columns of a matrix named P_0. Define the $N_0 \times N_0$ matrices

$$\begin{aligned} \tilde{C}(\tau_0) &= P_0^\dagger C(\tau_0) P_0, \\ \tilde{C}(t) &= P_0^\dagger C(t) P_0, \\ \tilde{G}(t) &= \tilde{C}(\tau_0)^{-1/2} \tilde{C}(t) \tilde{C}(\tau_0)^{-1/2}. \end{aligned}$$

2. Solve for the eigenvalues and eigenvectors of $\tilde{G}(\tau_D)$ using the full ensemble only. Let L_{tmax} denote the eigenvalue of largest magnitude. Put the $N_P \leq N_0$ eigenvectors associated with eigenvalues greater than $L_{\text{tmax}}\times$ `inimumInverseConditionNumber` into the columns of a matrix called $\tilde{V}_D$.

3. Evaluate the diagonal elements of the "rotated" correlation matrix

$$\tilde{D}(t) = \tilde{V}_D^\dagger \tilde{G}(t) \tilde{V}_D,$$

on the individual bins. On the full ensemble, we will have

$$\tilde{D}(tau0) = 1, \qquad \tilde{D}(\tau_D) = \text{diagonal}.$$

For different resamplings, the above relations will not be true.

If there are VEVs, these are also rotated and subtracted. An additional rephasing is done so the rotated VEVs are real and positive. In the single pivot method, this rotation is done bin-by-bin. Only the diagonal elements of the "rotated" correlation matrix are determined since these are the only ones needed (currently).

There are four important tasks that can be used to help accomplish the complete analysis of a correlation matrix:

– `<Action>DoCorrMatrixRotation</Action>`

– `<Action>DoRotCorrMatInsertFitInfos</Action>`

– `<Action>DoCorrMatrixZMagSquares</Action>`

– `<Action>DoCorrMatrixRelabelEnergyPlots</Action>`

6.8.1 Rotation procedure

The task `DoCorrMatrixRotation` is done first, then fits to the diagonal element of the rotated correlation matrix are done. From these fits, the level energies and amplitudes are obtained. The task `DoRotCorrMatInsertFitInfos` can be used to insert the fit energies and fit amplitudes into memory, optionally reordering the level indices by ascending fit energy. Then the amplitudes can be used finally to evaluate operator overlap factors in a `DoCorrMatrixZMagSquares` task. The plots from the fits to the rotated diagonal correlators have an initial level ordering. If you reorder the levels by ascending fit values, then the task `DoCorrMatrixRelabelEnergyPlots` can be used to modify the plots to agree with the new ordering.

To accomplish all of this, make sure to write the pivot to a file in the first task `DoCorrMatrixRotation` with a `<WritePivotToFile>` tag. Then do all of the fits to the diagonal rotated correlators, and write these energy and amplitude fits into a samplings file(s). Then do a `DoRotCorrMatInsertFitInfos` task, reading the pivot from file and inserting fit infos, specifying the sampling files containing the energies and amplitudes. One can reordering the energies at this point. Store the pivot in memory with an `<AssignName>` tag. In the final task `DoCorrMatrixZMagSquares`, initiate the pivot using a `<GetFromMemory>`. Note: fit energies and amplitudes from any given pivot file can be read into memory using the `GetFromPivot` task described in Sec. 6.2.4

6.8.2 The `DoCorrMatrixRotation` task

Given the (usually) large number of parameters in our correlator data, it is not feasible to fit to the entire 'raw' correlation matrix. Instead this task is used to *rotate* or *diagonalize* the matrix to make the extraction of energies and overlap factors by least squares fitting more practical. We have developed three approaches for this extraction procedure that use the so-called *Correlation Matrix Pivot*: the Principal Axes method, the Rolling Pivot method and the Single Pivot method. However, only the single pivot method is currently available in `sigmond`.

The input XML must be of the the form:

```
<Task>
   <Action>DoCorrMatrixRotation</Action>
   <MinTimeSep>3</MinTimeSep>
   <MaxTimeSep>30</MaxTimeSep>
   <RotateMode>bins</RotateMode> (or samplings or samplings_unsubt
                                  or samplings_all)
   <Type>SinglePivot</Type>
   <SinglePivotInitiate> ... </SinglePivotInitiate>         (depends on type)
   <WriteRotatedCorrToFile>    (optional)
      <RotatedCorrFileName>rotated_corr_bins</RotatedCorrFileName>
      <WriteMode>overwrite</WriteMode> (default protect, update, overwrite)
   </WriteRotatedCorrToFile>
   <PlotRotatedEffectiveEnergies>   (optional)
     <SamplingMode>Jackknife</SamplingMode> (or Bootstrap: optional)
     <EffEnergyType>TimeForward</EffEnergyType> (optional)
         ( or TimeSymmetric, TimeSymmetricPlusConst, TimeForwardPlusConst)
     <TimeStep>3</TimeStep>  (default 1)
     <PlotFileStub>stubname</PlotFileStub> (produces several files with
                                            numerical suffices 0,1,...)
     <SymbolColor>blue</SymbolColor>
     <SymbolType>circle</SymbolType>
```

```
      <MaxErrorToPlot>1.0</MaxErrorToPlot>
    </PlotRotatedEffectiveEnergies>
    <PlotRotatedCorrelators>         (optional)
      <SamplingMode>Jackknife</SamplingMode> (or Bootstrap: optional)
      <PlotFileStub>stubname</PlotFileStub> (produces several files with
                                             numerical suffices 0,1,...)
      <Arg>Re</Arg>  (or Im)
      <SymbolColor>blue</SymbolColor>
      <SymbolType>circle</SymbolType>
      <Rescale>1.0</Rescale>          (optional)
    </PlotRotatedEffectiveEnergies>
</Task>
```

The `<Type>` specifies which kind of pivot to create. The tag `<SinglePivotInitiate>` applies only for a type of SinglePivot. For other types, there will be a different tag instead of `<SinglePivotInitiate>`. (Sadly, these other pivot types are not yet implemented.)

The choices for `RotateMode` are as follows:

– If you wish to rotate and save to file by bins, use `<RotateMode>`bins`</RotateMode>`.

– If you wish to rotate and save to file only the vev-subtracted correlators by samplings, use `<RotateMode>`samplings`</RotateMode>`.

– If you wish to rotate and save to file only the unsubtracted correlators and the vevs by samplings, use `<RotateMode>`samplings_unsubt`</RotateMode>`.

– If you wish to rotate and save to file the unsubtracted correlators, the the vevs, and the vev-subtracted correlators by samplings, use `<RotateMode>`samplings_all`</RotateMode>`.

To save, write, and/or plot results:

– `<WriteRotatedCorrToFile>` writes the rotated correlation matrix to file (as bins or samplings).

– `<PlotRotatedCorrelators>` plots the rotated diagonal correlators.

– `<PlotRotatedEffectiveEnergies>` plots effective energies for the rotated diagonal correlators.

6.8.3 `SinglePivot`

Since diagonalization at large times can amplify errors, this method performs the diagonalization using the full ensemble with one choice of metric time, τ_0, and only one other time, τ_D. Hence, only a single pivot is performed. Using these diagonalized matrices then, a fully pivoted correlator matrix can be obtained, rotated bin by bin. The `SinglePivot` tasks must all be contained within the `<SinglePivotInitiate>` tags:

```
<Task>
   <Action>DoCorrMatrixRotation</Action>
   <MinTimeSep>3</MinTimeSep>
   <MaxTimeSep>30</MaxTimeSep>
   <Type>SinglePivot</Type>
   <SinglePivotInitiate>
      <CorrelatorMatrixInfo>
      ...
      </CorrelatorMatrixInfo>
      ... tasks ...
   </SinglePivotInitiate>
   ...
</Task>
```

Based on the input XML, it either calculates and creates a new object (new memory), reads a previously calculated object from file (new memory), or gets a pointer to a previously calculated object saved in the task handler data map (no new memory).

Create a new pivot:
The format of the input XML to create a new pivot is as follows:

```
<SinglePivotInitiate>
   <RotatedCorrelator>
     <GIOperator>...</GIOperator>
   </RotatedCorrelator>
   <AssignName>PivTester</AssignName>  (optional)
   <CorrelatorMatrixInfo> ... </CorrelatorMatrixInfo>
   <ImprovedOperators> ... </ImprovedOperators>  (optional)
   <NormTime>3</NormTime>
   <MetricTime>6</MetricTime>
   <DiagonalizeTime>12</DiagonalizeTime>
   <MinimumInverseConditionNumber>0.01</MinimumInverseConditionNumber>
   <NegativeEigenvalueAlarm>-0.01</NegativeEigenvalueAlarm>  (optional)
   <CheckMetricErrors/>    (optional)
   <CheckCommonMetricMatrixNullSpace/>    (optional)
   <WritePivotToFile>    (optional)
      <PivotFileName>pivot_test</PivotFileName>
      <Overwrite/>
   </WritePivotToFile>
   <PrintTransformationMatrix/> (optional)
    <SetImaginaryPartsZero/>  (optional: use carefully!!)
    <SetToZero>   (optional: noise reduction)
      <Correlator>
         <Source><Operator>..</Operator></Source>
         <Sink><Operator>..</Operator></Sink>
      </Correlator>
    </SetToZero>     (if hermitian, other correlator will be zeroed too)
</SinglePivotInitiate>
```

– The `<RotatedCorrelator>` tag specifies the operators to be rotated. Any integer index specified is *ignored*. If the matrix to be rotated is $N \times N$, then the N rotated operators will all have the same isospin and irrep labels and ID name, but the ID index will vary from 0 to $N - 1$.

– If `<AssignName>` is present, the pivot is inserted into the task handler data map, and can be accessed using this ID tag. If not present, the pivot is not stored in persistent memory. If the pivot is written to file, this ID name is NOT put into the file, allowing subsequent programs to assign whatever name they wish.

– The `<CorrelatorMatrixInfo>` tag specifies the correlator matrix of operators to be rotated. The tag `<HermitianMatrix>` **must** be present. The operators in this tag can be Basic LapH operators, general irrep operators, or so-called "improved operators" which are linear combinations of other operators. If improved operators are present, then an `<ImprovedOperators>` tag must be present as a child of the `<SinglePivotInitiate>` tag. If the tag `<ImprovedOperators>` is present, this means that some or all of the operators are linear combinations of another set of operators. The linear combinations must then be given within this tag in the form:

```
<ImprovedOperators>
  <ImprovedOperator>
    <OpName>
      <GIOperatorString>isotriplet P=(0,0,0) A1gp_1 Rot 0</GIOperatorString>
    </OpName>
    <OpTerm>
      <BLOperatorString>pion P=(0,0,0) A1gp_1 SD_0</BLOperatorString>
      <Coefficient>(-0.0593735752248,0.0421528577847)</Coefficient>
    </OpTerm>
       ...
  </ImprovedOperator>
    ....
</ImprovedOperators>
```

– `<NormTime>`, `<MetricTime>`, and `<DiagonalizeTime>` specify the normalization, metric, and diagonalization times respectively.

– The tag `MinimumInverseConditionNumber` has already been explained above, but its purpose is to remove noisy states. States that are not sufficiently independent of the other states can become dominated by noise. The fractional errors in the diagonal elements of $\mathcal{C}(\tau_0), \mathcal{C}(\tau_D)$ are printed out for informational purposes.

– If the tag `<NegativeEigenvalueAlarm>` is set to a negative value, then if any eigenvalues are less than this value, this fact is reported.

- If the tag `<CheckMetricErrors>` is present, the eigenvalues of the largest and smallest magnitudes of $C(\tau_0)$ are determined by jackknife and the condition number with errors is output. This can help in determining the `<MinimumInverseConditionNumber>` to use.
- If the tag `<CheckCommonMetricMatrixNullSpace>` is present, then in $\tilde{G}(\tau_D)$, the null space of $\tilde{G}(\tau_D)$ is checked to see that it contains the entire null space of $\tilde{C}(\tau_0)$. This is a desirable property when removing noisy eigenvectors. Finding this false indicates caution in interpreting the overlap factors.
- If the `<WritePivotToFile>` tag is assigned, the information in the pivot is written out to a file so that it can be input by later sigmond runs. The pivot file is an `IOMap` (a single integer key) with an XML header string and binary data. The XML header string contains
  - the correlator matrix info
  - the rotated correlator ID info
  - the norm time, metric time, diagonalize time
  - the minimum inverse condition number

  The binary data contains
  - the rotation (transformation) matrix (key = 0)
  - the matrix needed to compute the overlap Zmag squares (key = 1)
- `<PrintTransformationMatrix/>` prints out the transformation matrix in XML format to the logfile.
- If you happen to know that the correlator matrix is not only Hermitian, but real and symmetric, you can specify the tag `<SetImaginaryPartsZero>`, which sets all imaginary parts to zero. This can help reduce statistical errors. However, use carefully!!
- If you happen to know that certain correlators are negligible, you can set them to zero in order to prevent noise from these correlators filtering into other correlators upon diagonalization. Use the `<SetToZero>` tag carefully.

Set up previously created pivot from a file:
Input XML to set up a previously created pivot saved in a file is as follows:

```
<SinglePivotInitiate>
   <ReadPivotFromFile>
      <PivotFileName>pivot_file</PivotFileName>
   </ReadPivotFromFile>
   ... other tasks ...
</SinglePivotInitiate>
```

Set up previously created pivot from memory:
Input XML to set up a previously created pivot saved in memory is as follows:

```
<SinglePivotInitiate>
   <GetFromMemory>
      <IDName>PivTester</IDName>
   </GetFromMemory>
   ... other tasks ...
</SinglePivotInitiate>
```

6.8.4 The `DoRotCorrMatInsertFitInfos` task

After fits to the diagonal elements of the rotated correlators are done to obtain the fit energies and the amplitudes, this information can be inserted into memory, optionally reordering the levels according to increasing fit energy. The needed amplitudes can either be individually specified in the XML below, or if they are given the same name with the ID index equal to the level number, a short form is available:

```
   <RotatedAmplitudeCommonName>Amp</RotatedAmplitudeCommonName>
```

The XML format for this task is:

```
<Task>
  <Action>DoRotCorrMatInsertFitInfos</Action>
   <Type>SinglePivot</Type>
   <SinglePivotInitiate> ... </SinglePivotInitiate> (depends on type)
   <ReorderByFitEnergy/>  (optional)
   <EnergyFit>
      <Level>0</Level>
      <Name>A</Name><IDIndex>0</IDIndex> (name of fit energy observable)
   </EnergyFit>
       ... one for each level
    (specify amplitude using short form:)
   <RotatedAmplitudeCommonName>Amp</RotatedAmplitudeCommonName>
    (or specify each individually:)
   <RotatedAmplitude>
      <Level>0</Level>
      <Name>A</Name><IDIndex>0</IDIndex>
   </RotatedAmplitude>
       ...  (needed for all levels)
</Task>
```

6.8.5 The `DoCorrMatrixRelabelEnergyPlots` task

If you reorder the levels by ascending fit values, then the task `DoCorrMatrixRelabelEnergyPlots` can be used to modify the plots to agree with the new ordering. The original plot files are specified either by a stub (stub_0.agr, stub_1.agr, ...) or the file names are specified in order. The revised plot files are similarly specified. If no revised plot files are given, the original files are overwritten. The XML must have the form

```
  <Task>
     <Action>DoCorrMatrixRelabelEnergyPlots</Action>
     <Type>SinglePivot</Type>
     <SinglePivotInitiate> ... </SinglePivotInitiate> (depends on type)
        ...use <GetFromMemory> ...
     <OriginalPlotFiles>
        <PlotFileStub> ... </PlotFileStub>
         or  <PlotFile>name0</PlotFile> ....
     </OriginalPlotFiles>
     <RevisedPlotFiles>   (optional)
        <PlotFileStub> ... </PlotFileStub>
         or  <PlotFile>name0</PlotFile> ....
     </RevisedPlotFiles>
  </Task>
```

6.8.6 The `DoCorrMatrixZMagSquares` task

After a correlator matrix has been rotated and energies extracted for the rotated diagonal correlators, an idea of which quantum mechanical operators contribute to/have non-zero overlaps with each level is often needed. (See spectroscopy notes for details)

Before computing the Zmag squares, the amplitudes from the rotated correlator fits are needed. These must be have been "inserted" into the pivot. XML format for correlator matrix Zmag squares computation:

```
<Task>
   <Action>DoCorrMatrixZMagSquares</Action>
   <Type>SinglePivot</Type>
   <SinglePivotInitiate> ... </SinglePivotInitiate> (depends on type)
        ...use <GetFromMemory> ...
   <DoPlots>                                (optional)
      <PlotFileStub> ... </PlotFileStub>
      <BarColor> ... </BarColor>            (optional: cyan default)
      <ZMagSqPlot>
        <BLOperatorString>...</BLOperatorString>
        <ObsName> ... </ObsName>
        <FileSuffix> ... </FileSuffix>      (optional: default is index)
      </ZMagSqPlot>
      ....
   </DoPlots>
</Task>
```

7 Conclusion

It is hoped that you find these notes helpful for successfully using `sigmond` , a software suite for the analysis of Monte Carlo data in lattice QCD. The name `sigmond` comes from signal extraction from Monte Carlo data, and also plays upon the name "Sigmund" from Sigmund Freud, a neurologist famous for his analytical thinking. `sigmond` currently runs only in batch mode. These notes focused on the use of the software; contact the author of these note if you need assistance with installation. Enjoy!