SPIRE Fourier Transform - User Guide

Prepared by:
Andres Rebolledo (andres.rebolledo@uleth.ca) and Peter Davis (peter.davis@uleth.ca),
University of Lethbridge, Canada.
Version 1.0; March 14, 2005
SPIRE-UOL-DOC-002495

Table of Contents:

Purpose

The purpose of this document is to familiarize users with the tools to efficiently execute the task Fourier Transform through Jython scripts and a Java class. This document is not meant to describe the package in detail which is done in the Technical Note on the ICC Work Package Fourier Transformation, SPIRE-UOL-NOT-002204.

Background

WPFourierTransform is a package of Java classes which reduce data from the SPIRE FTS. The inputs to the package are Spectrometer Data Products which are produced by the Engineering Data Process (EDP). The output of the package is a Spectrometer Detector Spectrum (SDS) Product. The package will be included as part of the Herschel Interactive Analysis software package and is located in herschel.spire.ia.ft. The class SpecCalculation is the main wrapper class which calls various sub-tasks.

Tools

Three tools are at different stages in their development to efficiently execute the FT task:

1) RunFT Jython Script

This script is designed to open data products from disk and produce the resulting spectra. The input products are stored in FITS format. To use this script you must identify the products you want to reduce, and the parameters you want to use for

reducing the data. Parameter values are set by the script. Reduced data may be output to screen in plot windows, and reduced and intermediate data steps may be saved to disk.

2) EDP -> FT Script

This script is not yet available as it depends on the integration with the EDP task. When completed, this script will run identically to RunFT, except instead of loading products from FITS, products will be created using the EDP.

3) SpecCalculationTest

This is a Java class which operates similar to RunFT. However, since it is a Java class, to change parameter values you must recompile the class. This makes using this class considerably less convenient than the Jython script.

Required Data Products

There are two sets of products needed to run the task:

1) Spectrometer Data Products

These products include the Spectrometer Detector Timeline (SDT), Stage Mechanism Timeline (SMECT), and House Keeping Timeline (HKT). These products are produced by the EDP and are required to run the task. For more information on the data products see the SPIRE: FTS-related Data Products document (available on the SPIRE Bulletin Board > Data Products).

2) Calibration Products

These products are currently derived manually as instrument calibration is still pending. Calibration products include LVDT DC at ZPD calibration, LVDT DC to OPD calibration, OE to OPD calibration, band limit calibration, known non-linear phase calibration, and bad pixel calibration. For more information see the Task Fourier Transform Calibration Products document.

- Of these products, LVDT DC at ZPD is the only required product.

- The OE to OPD may be used, but this is optional.

- The bad pixel calibration may be used to mask bad pixels. The task BadPixelChecker may be used to create the bad pixel calibration product.

Setting Parameters

Many parameters can be set in the FT task and its sub-tasks. In order to organize all these parameters, the class SpecCalculationParams was created to store these parameters. Essentially, this class is a series of getters and setters for each parameter. In addition, calibration products are contained within an array in this class.

1) Task Control

Each sub-task in the FT task has an execution level which indicates whether or not to run a task, and if so, what type of debugging to use. Note that some tasks are not optional. The following execution levels are available, and are defined in interface SpireConstants:

SpireConstants.NO_EXEC:	Don't run the task. Some tasks need to be run, like RegSampledIfgmCreation.
SpireConstants.EXEC_QUIET:	Run silently, no messages except error messages.
SpireConstants.EXEC_MSG:	Run with debug messages.
SpireConstants.EXEC_PLOT:	Run with debug messages, plus plots for the first pixel it processes.
SpireConstants.EXEC_ALL:	Run with debug messages, plus plots for all pixels. NB: THIS WILL CLUTTER YOUR SCREEN! Also, coaddition and deglitching tasks need EXEC_ALL in order to show plots.

2) Task Output

Output from each sub-task may be saved to FITS file. The save files are named according to the root file name and task name, and are stored in the output file location. More details of task output are included in the Jython script.

3) Task Parameters

As the pipeline runs, each task is fed the appropriate data products. However, there are various additional task parameters which may be set according to user preference. The following table lists the tasks, whether or not it is an optional task (mandatory tasks are marked in yellow), and the parameters fed to each task.

Task Name	Optional	Parameters
BadPixelChecker	Yes	
RegSampledIfgmCreation	No	`interpolType, phaseShift`
DriftRemoval	Yes	`removeDriftPolyDeg`
IfgmDeglitcher	Yes	`deglitchType`
RegSampledApodization	Yes	`ifgmApodFunction`
RegSampledFT (double sided)	No	`dsZeropad`

Task Name	Optional	Parameters
RegSampledPhaseCorrection	Yes	`phaseCorrectPolyDeg, pcfApodFunction, pcfSize`
IfgmCoaddition	Yes	`coaddIfgmUpDown`
RegSampledApodization	Yes	`ifgmApodFunction`
RegSampledFT (single sided)	No	`ssZeropad`
SpecCoaddition	Yes	`coaddSpectraUpDown`

For information on the parameter, see the script or Javadoc documentation.

File Naming and Paths (only RunFT and SpecCalculationTest)

The RunFT Jython script allows the user to load FITS files from disk and then reduce the data.

The first step is to define the file name and paths to the products. Each product should contain a common root name, followed identification of what kind of product it is. A common input path and output path must also be defined for data products. Names and paths are described in more detail in the commented code.

Task Execution

Once the SpecCalculationParams class has been set up, the SpecCalculation task can be executed to produce spectra. Using the model of retrieving input products, setting up SpecCalculationParams, and executing the task, it is possible to use the FT task in different scripts.