UV Continuum Subtraction Tasks in CASA

T. Tsutsumi

ver. 1.0
September 13, 2013

1 Introduction

This document describes the visibility domain continuum subtraction tools currently implemented as of the CASA 4.1 release and the current trunk code. There are two versions of the task for such functionality, namely, `uvcontsub` and `uvcontsub3`. Both are intended to achieve the same results and uses the same continuum fitting and subtraction code at the core of the code, but main differences are in processing framework.

The `uvcontsub` uses the calibrator tool (`cb`) as the framework for driving fitting and subtraction process while `uvconstsub3` uses relatively recently implemented framework shared by `statwt` task. The `uvcontsub` currently is the one end users use and maintained regular basis. The newer, `uvcontsub3`, was created for the intention to reduce the problem that `uvcontsub` has, — having many I/O for the intermediate results during the process. But since it has not been tested well, the task has been marked as experimental. There has been no new development or any maintenance of the code since the original developer of the `uvcontsub3` code left the project. I tried to capture the behavior of `uvcontsub3` here as well based on what I understood from decoding the relevant C++ code.

To moving forward to improve the *uv*-continuum task, the performance improvement of the calibrater based `uvcontsub` by fitting and subtraction in more on-the-fly way may be possible with the current development and improvement in the calibrater module. We also have been discussing that `mstransform` to provide the core functionality for `uvcontsub`. As the C++ code for `uvcontsub3` was implemented in `SubMS`, such discussion is more relevant to `uvcontsub3`.

2 Visibility-based Continuum Subtraction

There are several ways to subtract out the continuum emission from spectra line data. The methods usually divided into two categories, whether do it in image plane or visibility plane. The principle idea of visibility-based continuum subtraction has been around for more than two decades. For example, the approach had been studied and analyzed by Cornwell et al. (1992) and Sault (1994). For the continuum emission sufficiently bright relative to spectral line emission usually work well with the visibility-based approach. The technique also known to work well when the dominant continuum source is at or near phase center. The effectiveness tends to decline as distance to the source location with respect to the phase center increases. The effectiveness which has the same expression as in bandwidth smearing, can be parameterized as $\eta = \frac{\Delta\nu}{\nu}\frac{l_0}{\theta_{synth}}$ (Sault, 1994) in terms of the the distance

in the synthesized beam (l_0/θ_{synth}), where ν and $2\Delta\nu$ are the observing frequency and the bandwidth, respectively. In order to the method to work, $\eta << 1$ must be met. the restriction may be relaxed by fitting with higher terms but in general, if the brightest continuum emission lies beyond $\frac{\nu}{\Delta\nu}$ beams from the phase center the source need to be shifted before fitting.

In CASA, the tasks, `uvcontsub` and `uvcontsub3` are both implemented this visibility based approach by fitting line-free channels to estimate continuum emission.

3 Uvcontsub Task Processing Flow

Estimating the continuum emission and subtract the model from the visibility can be viewed as one of the calibration step. Thus `uvcontsub` is implemented using `cb` tool by solving for "A" - type (additive term).

Figure 1 shows the processing flow where the process are roughly divided into three stages, data preparation, fit and subtraction, and post-process.

In the data preparation stage, at first, it checks if the input MS is a multi-MS, sets up processing for parallelization including creating multi-MS structure for the output data. If `excludechans`=True, `fitspw` is used for exclusion criteria for creating correct spectral window and channel/frequency selections which will be used internally in the rest of the process. Since `cb` tool updates solutions on to the input MS, a working input MS is created by `ms.split` when CORRECTED_DATA exists or selections in output spectral windows and/or fields are made. Otherwise, the working copy of the MS is made by simple copying of the whole MS.

The main process for fitting and subtracting the continuum is done using the calibrater module. The selection defined in `fitspw` is used and `cb` solves for the continuum with the fit order specified and then stores the result in a temporary caltable on the disk. The `combine` parameter will be applied to specify data axes on which the data will be combined to generate a single solution. The processing of combining are common part of Calibrater so the behavior and interaction with other parameters such as `solint` should be (in principle) the same as in the other calibration tasks. So for example, combine='spw' will do fitting over frequency range acrosses spectral windows. While the process is driven by `cb`, the fitting and subtraction code was implemented in `MSVis::VBContinuumSubtractor` class. The `VBContinuumSubractor` performs least-square fitting with `fitorder` for complex visibilities separately fitting real and imaginary parts in a VisBuffer where a chunk of the visibility data is accessed. Fitting in complex visibilities rather amplitude and phase are important to avoid biases. The fit order is automatically downgraded to a lower order if there is insufficient number of unflagged data for user specified order. The fit solution stored in the temporary caltable is then applied to the working MS by running `cb.correct()`, to fill CORRECTED_DATA with the continuum-model-subtracted data. In addition, if `want_cont=True`, `cb.corrupt()` is run to fill the continuum model visibilities in the MODEL_DATA.

In post-process stage, the output MS contains the continuum-subtracted data in DATA

column is generated from the working MS by `ms.split()`. If the continuum data is requested, another `split` will be run to extract MODEL_DATA column and write to the output DATA column.

4 Uvcontsub3 Task Processing Flow

Since the `uvcontsub` potentially runs multiple split up to three times within the task as described in the previous section, `uvcontsub3` was implemented to address this problem. The task made some improvements in speed and amount of temporary disk usage by supporting on-the-fly fitting and subtraction. The code is implemented in the `ms` tool and uses `SubMS` classes.

From the view of the task interface, the supported parameters are also slightly different from `uvcontsub`. For `uvcontsub3`, `solint` is not available so that the fit is always done in per integration. The `combine` parameter only supports `spw`. Also, the `excludechans` sub-parameter is not available for `fitspw`. There is no option to create the fitted continuum data. The reason this option is omitted may be, generally, aside from the confirmation purpose, the continuum model contains not only continuum emission model but also contains contributions from any residual errors, and there for it does not represent a true continuum model. Few data selection parameters are added, namely, `intent` and `observation`.

Without much of pre-processing, the task calls `ms.contsub()` where all the fitting for continuum and subtraction from the visibility data take place. Figure 2 mostly shown the processing steps internal to the subMS methods. The fitting to the visibility data is done on VisBuffers and applied on on the fly to the working MS during Main table filling process in `SubMS::makeSubMS()`, which write out a new continuum-subtracted MS. So, this part is equivalently doing `cb.setsolve(), cb.solve(), and cb.apply()` in `uvcontsub` except all its done in on-the-fly. This processing uses `VisBufferGroup`, where it stores copies of VisBuffers. This `GroupProcessor/VIsBufferGroup` framework is used in `statwt` task as well.

5 Moving Forward

The following is a random list of things that I think we should consider in moving forward to improve the visibility-based continuum subtraction task.

- One feature missing in both of the current `uvcontsub` and `uvcontsub3` tasks, is ability to shift the visibility data so in the case of the dominant continuum source not located at/near the phase center is shifted. As described in 2, this is important feature to have so that the tasks can be applicable wider cases.
- The way continuum-subtracted data and continuum model data stored in output MS(es). The current way of output the continuum model is output in a separate
- The way to exclude the data with some criteria during fitting

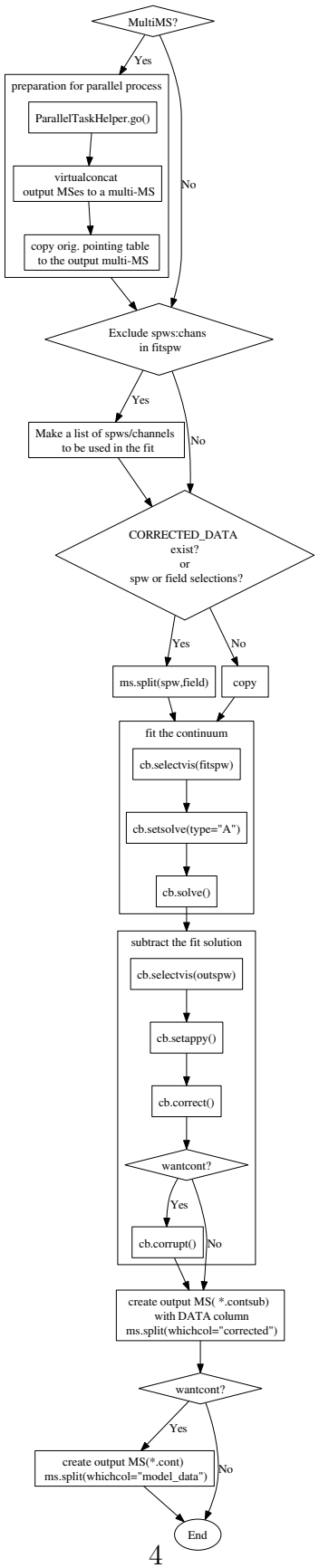

Figure 1: uvcontsub task flowchart

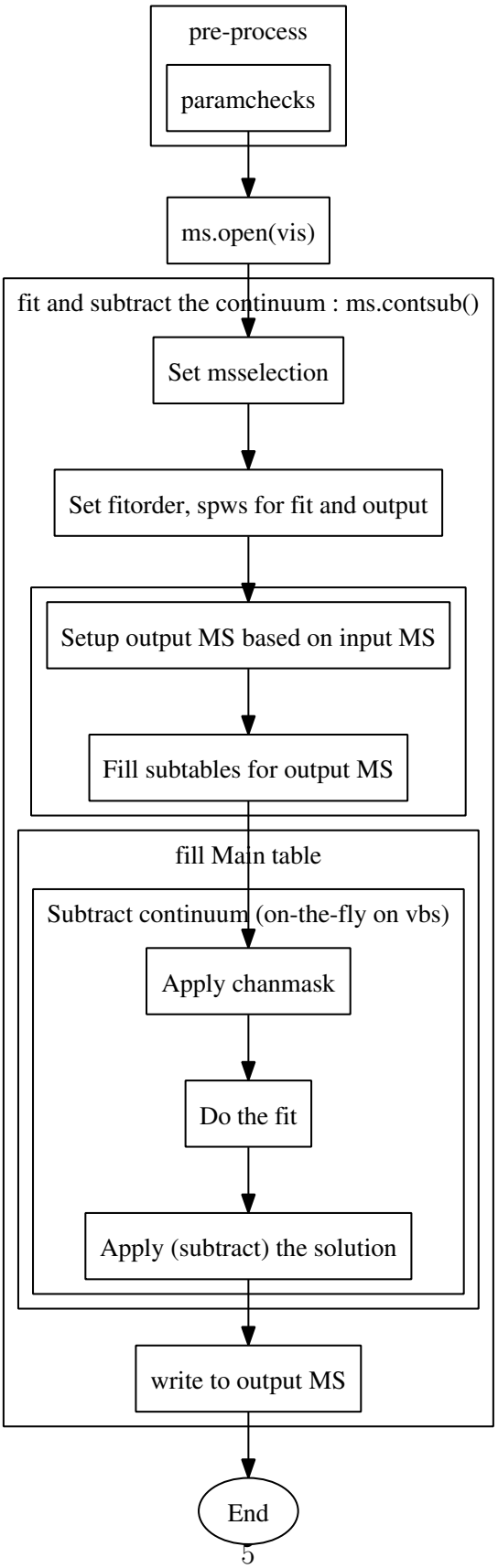

Figure 2: uvcontsub3 task flowchart

6 Questions from Crystal as Posted in the JIRA ticket, CAS-4633

While some of the questions posted on the JIRA tickets are addressed in the previous sections, for clarity I answer each of the questions here.

1. Q: What happens when combine='spw'? Does it find the the combined fit to all spws in frequency space, or does it average the spw and then fit? How is frequency overlap handled? What if there are multiple datasets and you set solint (larger than int) would it cross observation id boundaries?

   A: combine='spw' should try to a fit to combined data including all the specified spws. For this no averaging is done before the fit. Currently only 'spw' and 'scan' are permitted in `combine`. With `combine`='spw,scan' and with setting `solint` > int. But it probably do not cross the different obsids if there are multiple obsids.

2. Q: Is there any reason not to run CVEL before uvcontsub?

   A: Functionality wise, it should be fine.

3. Q: Why is no quantitative information about the goodness of fit given? Why doesn't it report the spectral index when fitorder=1. Is such info already generated but not printed to the logger?

   A: The code which responsible for fitting (`VBContinuumSubtractor`) currently does not keep record of the chi-square, etc, although such information should be easily retrievable with a proper method call to `Fitter` code. Since the fit is done for each integration, how it should be reported may need some considerations.

4. Q: How much does S/N affect the fit? Time-smearing is an easily quantifiable effect, if the solution can really be improved by larger solint shouldn't we reformulate to stay within a reasonable time-smearing limit but improve S/N? i.e. for a 1 km baseline 30 seconds still gives very little time-smearing, this would be a factor of sqrt (5) in S/N for most ALMA spectral line datasets (6 second native averaging).

   A: I think the principle of the visibility based approach is to estimate the continuum per integration. By doing per integration it would not effected by time dependent errors. So it is more accurate than do averaging before fit. But probably worth do some experiments.

5. Q: The previous incarnations of uv-continuum subtraction worked poorly if the brightest emission was not at the phase center and it was recommended to shift the phase center to the brightest continuum emission first. We don't even have this option, does that mean our code is somehow insensitive to this effect?

   A: As described in previous sections, our implementation follows the visibility-based subtraction approach. So it will be effected if the bright continuum emission is large distance from the phase center. I think the option to shift the source to phase center has to be available.

References

T. J. Cornwell, J. M. Uson, and N. Haddad. Radio-interferometric imaging of spectral lines - the problem of continuum subtraction. *Astronomy and Astrophysics*, 258:583–590, May 1992. URL `http://adsabs.harvard.edu/abs/1992A%26A...258..583C`.

R. J. Sault. An analysis of visibility-based continuum subtraction. *Astronomy and Astrophysics Supplement Series*, 107:55–69, October 1994. URL `http://adsabs.harvard.edu/abs/1994A%26AS..107...55S`.