

**RADIO OBSERVATORIO DE JICAMARCA**

**INSTITUTO GEOFÍSICO DEL PERÚ**



**JASMET: LIBRERÍAS DE PROCESAMIENTO DE DATOS**

**Julio Oscanoa**

**setiembre 2016**

**ÍNDICE**

1. Introducción
2. **Módulos desarrollados**

2.1. Meteor Detection

2.2. Phase Calibration

2.3. Correct Meteor Phases

2.4. Wind Profiler

2.5. Sky Map Plot

2.6. Wind Profiler Plot

2.7. Param Writer

2.8. Param Reader

1. **Descripción de scripts**

3.1. Monitoreo en tiempo real

3.2. Detección de meteoros inicial

3.3. Estimación de Vientos

3.4. Calibración de Fases

3.5. Estimación de Offset óptimo

1. **Ejecución de scripts**

4.1. Acceso a escritorio remoto

4.2. Montar disco de datos

4.3. Detección de meteoros

4.4. Estimación de vientos

4.5 Calibración de Fases

1. **Análisis de Gráficos**

5.1. Gráficos de espectros

5.2. RTI

5.3. Sky Map

5.4. Wind Profiler Plot

5.5. Phase Plot

5.6. Optimum Phase plots

1. **Bibliografía**
2. **Anexos**

JASMET: LIBRERÍAS DE PROCESAMIENTO DE DATOS

# 1. **INTRODUCCIÓN**

El presente documento realiza una descripción de los módulos desarrollados para el procesamiento de datos para el Jicamarca All-sky Specular Meteor (JASMET) radar.

El documento consta de las partes principales que se explican a continuación. Primero, se detallan cada uno de los módulos desarrollados dentro del entorno Signal Chain y se listan los parámetros de configuración utilizados. Luego, se describe mediante diagramas de flujos los scripts que se ejecutan para la detección de meteoros especulares y estimación de vientos horizontales. Finalmente, se describen cada uno de los gráficos que se obtienen durante el procesamiento de datos, con la finalidad de monitorear el correcto funcionamiento de las librerías.

**2. MÓDULOS DESARROLLADOS**

En esta sección se describirán cada uno de los módulos utilizados para el procesamiento de datos de JASMET. El primer módulo es *Meteor Detection*, el cual detecta los meteoros especulares dentro de las señales de voltaje y estima sus parámetros. Luego, se encuentra el módulo *Phase Calibration,* el cual estima el offset de fases óptimas, por canal, que permite obtener la mayor cantidad de ecos válidos para un determinado periodo de tiempo. Dichos offsets pueden ser introducidos a la detección de meteoros mediante la operación *Correct Meteor Phases*. Con los parámetros de los meteoros corregidos, se procede a estimar vientos horizontales con *Wind Profiler*.

Para el monitoreo del correcto funcionamiento de las librerías, se ha desarrollado dos gráficos *Sky Map*, el cual grafica la ubicación de los meteoros en coordenadas polares, y el *Wind Profiler Plot*, el cual grafica las estimaciones de vientos para un determinado tiempo. Finalmente, se ha desarrollado librerías de lectura y escritura de datos de tipo parámetros en formato HDF5 para poder guardar los ecos detectados y las estimaciones de vientos.

**2.1. Meteor Detection**

Módulo del tipo Operación que realiza la detección de ecos de meteoros especulares a partir de las señales de voltaje. El procedimiento seguido es el descrito en Holdsworth et al. 2004a [1], el cual se detalla a continuación:

1. **Detección de meteoros**, se forma la serie de potencia combinada de las cinco señales de de voltaje de recepción. Con dicha señal se estima el nivel de ruido y señal según [1]. Un candidato de eco de meteoro se detecta cuando por lo menos tres muestras contiguas superan un umbral seleccionado de SNR. Se graban los tiempos de inicio, pico y fin de cada meteoro.
2. **Remoción de detección múltiple**, se verifica que un mismo meteoro no se haya detectado múltiples veces en un rango de tiempo y alturas.
3. **Reestimación de meteoros**, con los candidatos a meteoro restantes, se corrige las fases teniendo de referencia el canal central. Luego, se reestima los tiempos de inicio, pico y fin de cada eco.
4. **Tiempo de Decay**, se evalúa el tiempo de decaimiento para verificar que solo se mantengan los “underdense echoes”. Se evalúa el tiempo de subida, decaimiento y que este último tenga un comportamiento exponencial.
5. **Velocidad Radial**, se calcula la velocidad radial del eco mediante la correlación cruzada.
6. **Cálculo de parámetros**, con los ecos restantes, se calculan los parámetros que se utilizarán en la estimación de vientos. Con las diferencias de fase entre canales, se calcula el ángulo de arribo y la altura mediante interferometría según Jones et al. 1998 [2].

**2.1.1. Entrada :** Señal de voltaje de los cinco canales

**2.1.2. Salida :** Ecos de Meteoros detectados

**2.1.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| hmin | int, float | Altura mínima para detectar meteoros |
| hmax | int, float | Altura máxima para detectar meteoros |
| azimuth | int, float | Ángulo a corregir para alinear ejes cardinales con el de JASMET |
| cohDetection | bool | Usar Detección coherente o no |
| cohDet\_timeStep | int, float | En la Detección coherente, # segundos del bloque |
| cohDet\_thresh | float | En la Detección coherente, umbral para corregir fases |
| noise\_timeStep | int, float | # segundos del bloque para cálculo de ruido |
| noise\_multiple | int, float | Multiplicador de nivel de ruido para definir umbral de nivel de señal |
| multDet\_timeLimit | int, float | Rango de tiempo (en s) en el que se buscan múltiples detecciones de un meteoro |
| multDet\_rangeLimit | int, float | Rango de alturas (en km) en el que se buscan múltiples detecciones de un meteoro |
| phaseThresh | float | Máxima diferencia de fases entre receptores para el meteoro |
| SNRThresh | Int, float | Umbral de SNR mínimo para el meteoro |

**2.2. Phase Calibration**

Módulo del tipo Operación que calcula el offset de fase óptima para detectar la mayor cantidad de ecos válidos posibles. A partir de los ecos de meteoros detectados, realiza un método iterativo que calcula la cantidad de meteoros válidos para determinadas correcciones en las fases. La corrección en las fases para los que hay una mayor cantidad de ecos válidos se considera los offsets óptimos. El procedimiento sigue Holdsworth et al. 2004b [1].

1. **Cálculo de Gammas**
2. **Cálculo de offsets óptimos**

**2.2.1. Entrada :** Ecos de meteoros detectados

**2.2.2. Salida :** Offset óptimo para las fases

**2.2.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| channelPositions | Pair list | Lista de pares que indica la posición de cada antena de recepción |
| hmin | int, float | Altura mínima para detectar meteoros |
| hmax | int, float | Altura máxima para detectar meteoros |
| nHours | int, float | Número de horas que acumula meteoros para calcular los offsets |

**2.3. Correct Meteor Phases**

Módulo del tipo Operación que corrige los offsets de fase de los canales, para una estimación óptima de ecos de meteoros. Se ingresa el offset por canal en la variable “phaseOffsets”. Dichos ángulos serán restados de la diferencias de fases calculadas para cada eco.

**2.3.1. Entrada :** Ecos de meteoros detectados

**2.3.2. Salida :** Ecos de meteoros detectados con fases corregidas

**2.3.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| phaseOffsets | Float list | Phase offset por canal |
| channelPositions | Pair list | Lista de pares que indica la posición de cada antena de recepción |
| hmin | int, float | Altura mínima para detectar meteoros |
| hmax | int, float | Altura máxima para detectar meteoros |
| azimuth | int, float | Corrección al azimuth para alinear arreglo con el norte geográfico |

**2.4. Wind Profiler**

Módulo del tipo Operación que hace las estimaciones de vientos horizontales a partir de los parámetros de los ecos de meteoros.

**2.4.1. Entrada :** Ecos de meteoros detectados

**2.4.2. Salida :** Estimaciones de vientos horizontales

**2.4.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| nHours | float | Número de horas por estimación de vientos |
| meteorsPerBin | int | Cantidad de ecos mínimos por bin altura/tiempo para proceder a estimación de vientos |
| hmin | int, float | Altura mínima para estimar vientos |
| hmax | int, float | Altura máxima para estimar vientos |

**2.5. Sky Map Plot**

Módulo del tipo Gráfico que grafica los ecos de meteoros detectados según su ubicación en el espacio.

**2.5.1. Entrada :** Ecos de meteoros detectados

**2.5.2. Salida :** Gráfico con los ecos detectados

**2.5.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| id | int | ID del gráfico |
| tmin | Int, float | Tiempo mínimo |
| tmax | int, float | Tiempo máximo |
| timerange | int, float | Rango de tiempo máximo en el |

**2.6. Wind Profiler Plot**

Módulo del tipo Gráfico que grafica las estimaciones de vientos horizontales en una escala de colores

**2.6.1. Entrada :** Estimaciones de vientos

**2.6.2. Salida :** Gráfico con los vientos horizontales

**2.6.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| id | int | ID del gráfico |
| xmin | Int, float | Tiempo mínimo |
| xmax | int, float | Tiempo máximo |
| ymin | int, float | Altura mínima |
| ymax | Int, float | Altura máxima |
| zmin | int, float | Velocidad mínima |
| zmax | int, float | Velocidad máxima |
| timerange | int, float | Rango de tiempo |

**2.7. Param Writer**

Módulos de escritura de archivos formato HDF5 para datos de tipo Parámetro. En este caso, se utilizan después de la detección de ecos de meteoros para guardar dichos ecos y poder prescindir de los datos de voltaje, los cuales ocupan mucho mayor espacio de memoria y demoran mucho más en procesar. A partir de un archivo HDF5 de meteoros se pueden hacer estimaciones de vientos y, de igual manera, se puede utilizar este módulo para guardar las estimaciones en archivos HDF5.

**2.6.1. Entrada :** Ecos detectados ó estimaciones de vientos

**2.6.2. Salida :** Archivos formato HDF5 con los datos de entrada

**2.6.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| path | string | Path donde se guardarán los archivos HDF5 |
| blocksPerFile | list | Número de bloques guardados por archivo |
| metadataList | list | Lista con los nombres de los atributos de “dataOut” que se guardarán en los archivos como parte de la metadata. |
| dataList | int, float | Lista con los nombres de los atributos de “dataOut” que se guardarán en los archivos como parte de la data. |

**2.7. Param Reader**

Módulos de lectura de archivos formato HDF5 creados con el módulo Param Writer. En el caso de los módulos para JASMET, se utilizarán para leer los archivos que contienen los ecos detectados, con el fin de poder utilizarlos para calibrar fases o estimar vientos.

**2.6.1. Entrada :** Archivos HDF5

**2.6.2. Salida :** Objeto con los datos almacenados

**2.6.3. Parámetros de configuración:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| path | string | Path donde están almacenados los archivos HDF5 |
| startDate | string | Fecha de inicio de los datos a procesar |
| endDate | string | Fecha de fin de los datos a procesar |
| startTime | string | Hora de inicio de los datos a procesar |
| endtime | string | Hora de fin de los datos a procesar |

**3. DESCRIPCIÓN DE SCRIPTS**

Una vez descrito las funciones de cada módulo, en esta sección se procede a describir los scripts utilizados para el procesamiento de datos. El primero es para el monitoreo en tiempo real de la adquisición de datos. Luego, está el programa que realiza la identificación de ecos de meteoros en las señales de voltaje. Con los ecos detectados, se procede a estimar vientos. En caso se necesite modificar los offsets de fases de los canales, se pueden utilizar los ecos detectados para estimar los offsets óptimos con el script de calibración de fases.

**3.1. Monitoreo en tiempo real “JASMET30\_Online\_spc.py”**

El siguiente script realiza el monitoreo de la señal adquirida en tiempo real, realiza observando el comportamiento espectral de la señal. No obstante, primero se debe procesar en el tiempo. Se sigue el siguiente procedimiento:

* *Interpolate Heights*, en la mayoría de campañas de JASMET, se realiza otros experimentos en paralelo, por lo que es común que sus pulsos de transmisión interfieran en un par de alturas de la señal de recepción de JASMET. Para estos casos, se interpolan dichas alturas para evitar que la interferencia se expanda a otras alturas cuando se proceda a la Decodificación.
* *Decode*, se decodifica la señal
* *Coherent Integration,* se integra para aumentar el SNR
* *Spectra Plot*, se plotea el espectro de la señal para corroborar la correcta remoción de interferencias y la visibilidad de los ecos de meteoros.
* *RTI*, de igual manera, permite monitorear las interferencias y ecos de meteoros.



**Figura 1.** Diagrama de bloques del Monitoreo en Tiempo Real

**3.2. Detección de Meteoros inicial “JASMET30\_MetDet.py”**

El siguiente script realiza una detección de ecos inicial con la totalidad de datos adquiridos por un día. La detección de ecos también se puede hacer en tiempo real. No obstante, para poder determinar si se está realizando de manera correcta, es necesario procesar datos de varias horas, por lo que es preferible usar este módulo en modo offline y con datos correspondiente a todo un día de adquisición.

* *Voltage Reader*, se debe leer los datos en bloques de 100 segundos.
* *Interpolate Heights*, *Decode* y *Coherent Integration*, se realiza el mismo procesamiento de la señal en el tiempo que el script anterior.
* *SM Detection*, en este caso, en lugar de pasarlo al dominio de frecuencia, se procede a detectar los ecos de meteoros en bloques de 100 segundos.
* *Param Writer*, los ecos obtenidos se almacenan en un archivo de formato HDF5, para poder ser utilizados posteriormente en estimación de vientos o calibración de fases.



**Figura 2.** Diagrama de bloques de la Detección de Meteoros

**3.3. Estimación de Vientos “JASMET30\_Winds.py”**

Con los ecos obtenidos, se procede a estimar los vientos. Debido a que la entrada son datos ya procesados, es posible realizar estimaciones de un día entero en minutos. El procedimiento que se sigue se detalla a continuación.

* *Param Reader*, se leen los ecos almacenados en archivos HDF5.
* *Correct SM Phases*, se corrigen los offsets de fases de cada canal.
* *Sky Map Plot*, gráfico que presenta los ecos de meteoros detectados y sus coordenadas polares en el espacio.
* *Wind Profiler*, operación que estima los vientos horizontales a partir de los ecos de meteoros.
* *Wind Profiler Plot*, gráfico que presenta las estimaciones de vientos calculados.
* *Param Writer*, se procede a almacenar los vientos en archivos HDF5 para su posterior análisis.



**Figura 3.** Diagrama de bloques de la Estimación de Vientos

**3.4. Calibración de fases “JASMET30\_PhaseCal.py”**

En caso el Sky Map Plot presente distribuciones de meteoros incorrectas o las estimaciones de vientos presente problemas (como se explicará más adelante) el usuario puede ejecutar el script de calibración de fases. Dicho script calculará los offsets de fases óptimo para cada canal y los almacenará en un archivo HDF5 para que se puedan utilizar luego en el script de Estimación de Vientos.

* *Param Reader*, se leen los ecos almacenados en archivos HDF5.
* *SM Phase Calibration*, estimación de offsets de fases óptimas
* *Phase Plot*, gráfico que presenta los offsets calculados
* *Param Writer*, se almacena los offsets en un archivo HDF5.



**Figura 4.** Diagrama de bloques de la Calibración de Fases

**3.4. Estimación de Offset de fase óptima “Optimum\_Offset.py”**

**Ver Anexo A**

El script de Calibración de Fases dará como resultado una lista de fases óptimas por canal para los meteoros de un determinado intervalo de tiempo (por default 1 hora). Consecuentemente, tendremos varios valores posibles por canal, y el offset óptimo no es simplemente el promedio.

Uno de los factores que más influencian es la hora para el cual se hizo el cálculo. En horas del día hay considerablemente menor cantidad de meteoros que en horas de la noche y madrugada. Por lo tanto, los offsets calculados en horas de madrugada serán más acertadas, lo cual se puede validar viento el gráfico *Phase Plot*. En dicho gráfico, los offsets calculados en horas de día tienden a variar considerablemente entre intervalo e intervalo, mientras que en horas de la noche se mantienen parcialmente constantes.

El presente script hace la limpieza correspondiente de los outliers para poder hacer una estimación de offset óptimo. Primero lee los archivos HDF5 con los offsets calculados en el paso anterior y procede a calcular la media y elimina los valores que se encuentren a un valor determinado alejados de la media.

Finalmente, grafica las fases e imprime los valores óptimos.

**4. EJECUCIÓN DE SCRIPTS**

**4.1. Acceso a escritorio remoto**

Los scripts que procesan los datos se encuentran en el computador de procesamiento en la caseta de VIPIR. En consecuencia, para ejecutar dichos programas, será necesario acceder mediante un visor de escritorio remoto.

1. Entrar a Vinagre
2. Seleccionar “Conectar”
3. En “Protocolo”: VNC (Virtual Network Computing)
4. En “Equipo”: 10.10.40.37 (**Verificar que IP**)
5. Hacer click en “Conectar”
6. Clave: “soporte”
7. Aceptar

**4.2. Montar disco de datos de computador de adquisición**

Los datos se encuentran en el computador de adquisición. Por lo tanto, para procesarlos será necesario montar el disco a la computadora de procesamiento.

1. Abrir un Terminal (CTRL + ALT + T)
2. Ejecutar el siguiente comand (**Verificar que IP**)

sudo mount -t cifs //10.10.40.37/JASMET /mnt/jars -o password=operator,username=operator

**4.3. Monitoreo en tiempo real**

1. Abrir un Terminal (CTRL + ALT + T)
2. Entrar a la carpeta con los scripts de procesamiento (**Verificar carpeta**)

cd workspace/schainroot/source/schainpy/scripts/

1. Verificar el script de Monitoreo en Tiempo Real “JASMET30\_Online\_spc.py” (**3.1**). Se debe verificar que las siguiente variables sean correctas:
   * path, directorio donde se encuentran los archivos .r a procesar
   * pathfig, directorio donde se grabarán las imágenes
   * startDate, fecha de inicio para buscar datos
   * endDate, fecha de fin para buscar datos
   * startTime, hora de inicio para buscar datos
   * endTime, hora de fin para buscar datos
2. Ejecutar el script

python JASMET30\_Online\_spc.py

**4.4. Detección de meteoros**

1. Abrir un Terminal (CTRL + ALT + T)
2. Entrar a la carpeta con los scripts de procesamiento (**Verificar carpeta**)

cd workspace/schainroot/source/schainpy/scripts/

1. Verificar el script de Detección de Meteoros Inicial “JASMET30\_MetDet.py” (**3.2**). Se debe verificar que las siguiente variables sean correctas:
   * path, directorio donde se encuentran los archivos .r a procesar
   * pathfig, directorio donde se grabarán las imágenes
   * pathfile, directorio donde se almacenarán los archivos HDF5 con los ecos
   * startDate, fecha de inicio para buscar datos
   * endDate, fecha de fin para buscar datos
   * startTime, hora de inicio para buscar datos
   * endTime, hora de fin para buscar datos
2. Ejecutar el script

python JASMET30\_MetDet.py

**4.4. Estimación de Vientos**

1. Abrir un Terminal (CTRL + ALT + T)
2. Entrar a la carpeta con los scripts de procesamiento (**Verificar carpeta**)

cd workspace/schainroot/source/schainpy/scripts/

1. Verificar el script de Estimación Vientos Inicial “JASMET30\_Winds.py” (**3.3**). Se debe verificar que las siguiente variables sean correctas:
   * phaseOffsets, offsets de fases por canal a corregir obtenidos de “Optimum\_Offset.py” (**3.5**)
   * path, directorio donde se encuentran los archivos HDF5 con los ecos de meteoros
   * pathfig, directorio donde se grabarán las imágenes
   * pathfile, directorio donde se almacenarán los archivos HDF5 con los vientos
   * startDate, fecha de inicio para buscar datos
   * endDate, fecha de fin para buscar datos
   * startTime, hora de inicio para buscar datos
   * endTime, hora de fin para buscar datos
2. Ejecutar el script

python JASMET30\_Winds.py

**4.5. Calibración de Fases**

**4.5.1 Cálculo de Offsets óptimos por intervalo de tiempo**

1. Abrir un Terminal (CTRL + ALT + T)
2. Entrar a la carpeta con los scripts de procesamiento (**Verificar carpeta**)

cd workspace/schainroot/source/schainpy/scripts/

1. Verificar el script de Calibración de Fases “JASMET30\_PhaseCal.py” (**3.4**). Se debe verificar que las siguiente variables sean correctas:
   * path, directorio donde se encuentran los archivos HDF5 con los ecos de meteoros
   * pathfig, directorio donde se grabarán las imágenes
   * pathfile, directorio donde se almacenarán los archivos HDF5 con los offsets
   * startDate, fecha de inicio para buscar datos
   * endDate, fecha de fin para buscar datos
   * startTime, hora de inicio para buscar datos
   * endTime, hora de fin para buscar datos
2. Ejecutar el script

python JASMET30\_PhaseCal.py

**4.5.2 Cálculo de Offsets óptimo final**

1. Crear un nuevo script de Python y copiar el código de Estimación de Fase Óptima “Optimum\_Offset.py” (**Anexo A**).
2. Abrir un nuevo Terminal
3. Entrar a la carpeta donde creo el script “Optimum\_Offset.py”
4. Verificar el script de Estimación de Fase Óptima (**Anexo A**). Se debe verificar que el path donde se encuentran los archivos HDF5 con los offsets calculados es el correcto.
5. Ejecutar el script de Estimación de Fase Óptima “Optimum\_Offset.py” (**Anexo A**).

python Optimum\_Offset.py

1. La fase óptima a introducir en el script de Estimación de Vientos aparecerá impresa en la consola.

**5. ANÁLISIS DE GRÁFICOS**

En esta sección, se analizará cada uno de los gráficos producidos para el monitoreo de los datos adquiridos con JASMET. Asimismo, para cada caso, se dará una representación típica correcta y otra errónea, junto con los posibles orígenes de los errores.

**5.1. Gráficos de espectros**

En la Figura 5, se puede observar como “normalmente” se ven los ecos de meteoros en los gráficos de espectros de Signal Chain. Los ecos se observan como picos de potencia en una altura determinada, los cuales tienen una velocidad radial baja, debido a que representa la velocidad radial del viento, mas no del meteoro.



**Figura 5.** Gráfico de espectros con EEJ débil. Marcados con rojo se ven tres meteoros y de naranja el EEJ, el cual es débil en las horas de madrugada

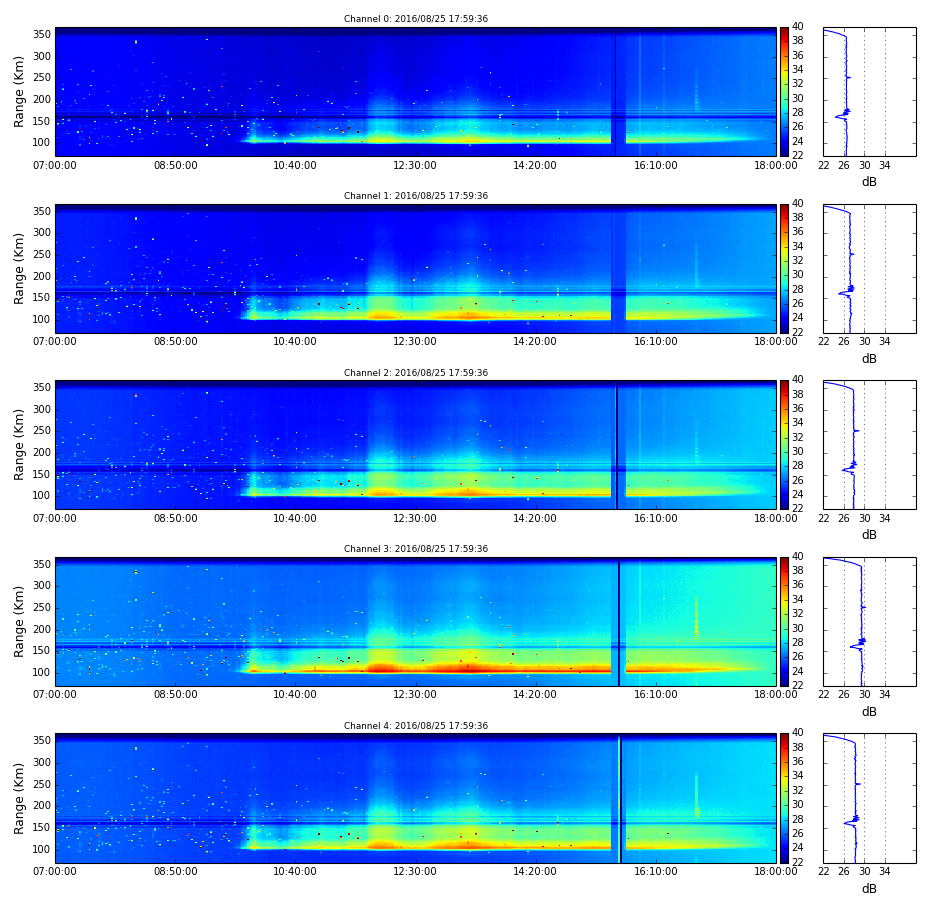
En la Figura 6, se observa el gráfico para una hora de la mañana, en la que se puede apreciar que el EEJ presenta una gran interferencia que casi impidge ver los ecos de meteoros.



**Figura 6.** Gráfico de espectros con EEJ fuerte. Marcados con rojo se ven tres meteoros, los son casi cubiertos por la gran potencia del EEJ

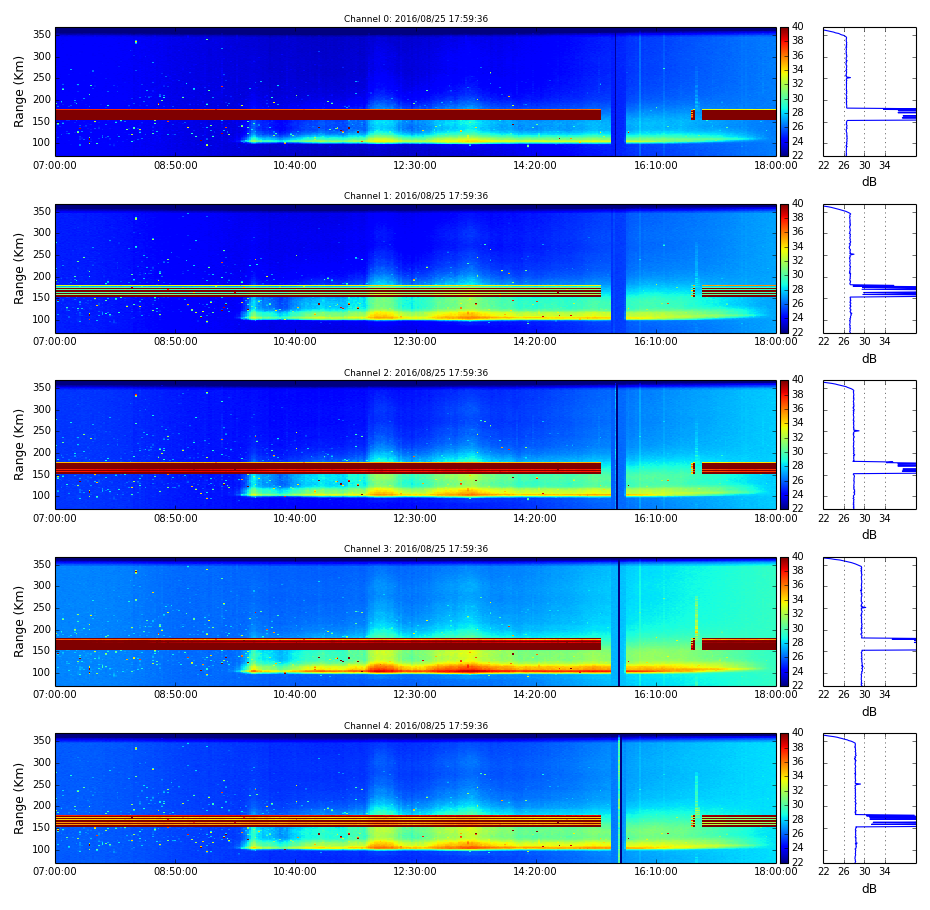
**5.2. RTI**

En la Figura 7, se puede apreciar un típico RTI de una adquisición de datos. Los ecos de meteoros se aprecian como puntos brillantes, lo cual representa picos breves de potencia en la señal.



**Figura 7.** Gráfico de RTI. Los ecos de meteoros se ven como picos de potencia en la señal.

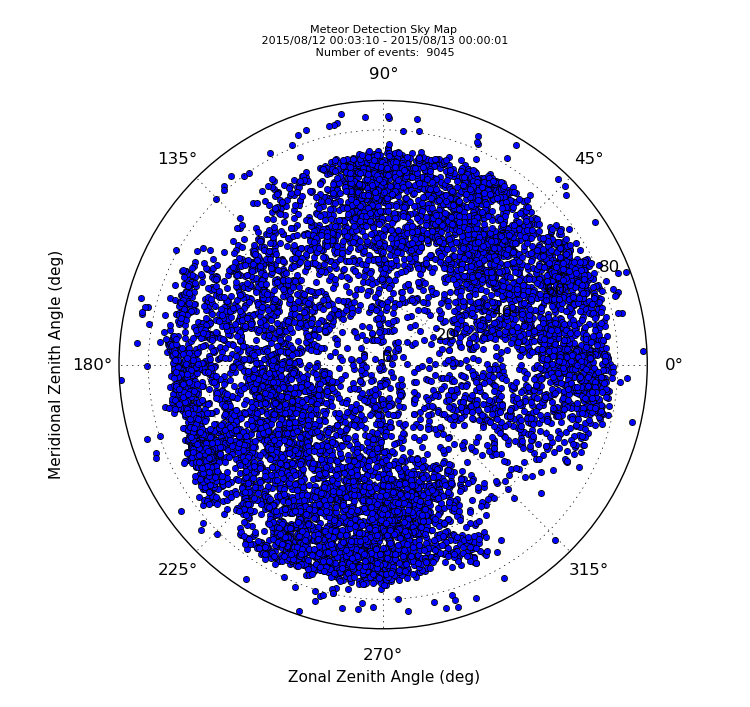
Por otro lado, en algunas campañas de JASMET, se corre otros experimentos en paralelo, lo que puede ocasionar interferencias en la señal, como se aprecia en la Figura 8. En dicho gráfico hay una interferencia alrededor de los 150 km que se debe a un pulso de transmisión de otro experimento. Para estos casos, se debe interpolar las alturas afectadas por el pulso de transmisión. En general, son solo dos o tres muestras de altura, los cuales se expanden durante la decodificación de la señal.



**Figura 8.** Gráfico de RTI con interferencia.

**5.1. Sky Map**

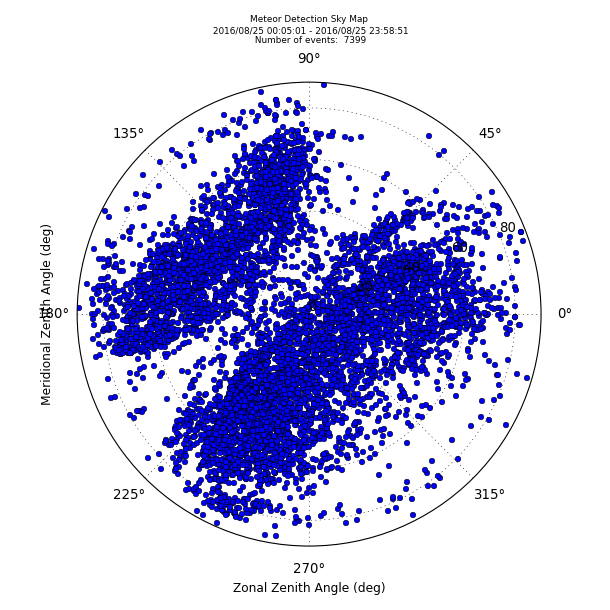
En la Figura 9, se puede observar un Sky Map de meteoros típico. En dicho gráfico, se ilustran los ecos de meteoros detectados en función de sus coordenadas polares (azimuth y zenith). Un Sky Map correcto presenta una distribución de meteoros aleatoria, de manera que, si es que se grafican los meteoros de todo un día, deberían aparecer distribuidos uniformemente. Cabe resaltarse que, según el diseño de JASMET, hay una franja diagonal en la que no se observarán meteoros



**Figura 9.** Sky Map de ecos de meteoros detectados.

Los ecos deben tener una distribución aleatoria uniforme a excepción de una franja diagonal.

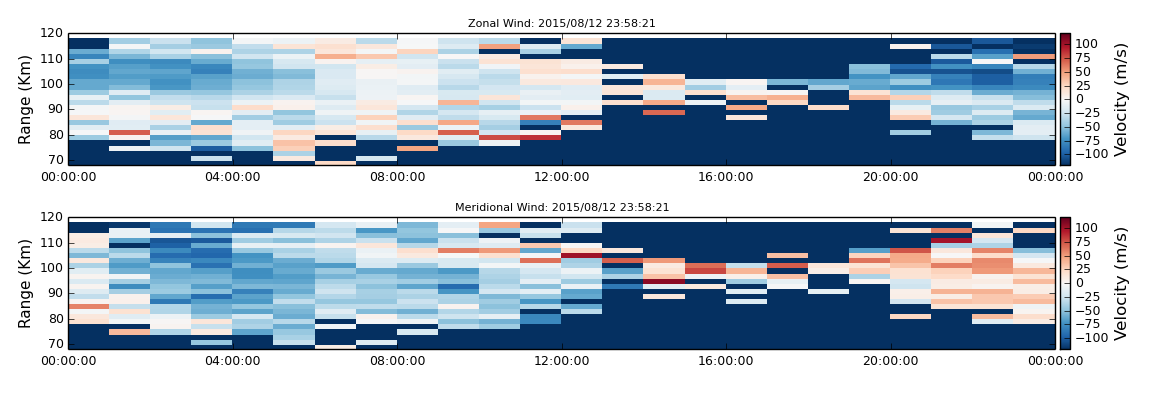
En caso no se observe un gráfico como el anterior, es muy probable que sea debido a un problema con los offsets de fase de cada canal. En la Figura 10, se observa un Sky Map erróneo, debido a la falta de corrección de las fases. La distribución deja de ser uniforme en toda la extensión del Sky Map.



**Figura 10.** Sky Map de ecos de meteoros erróneo. La falta de corrección de fases ocasiona que la distribución deje de ser uniforme.

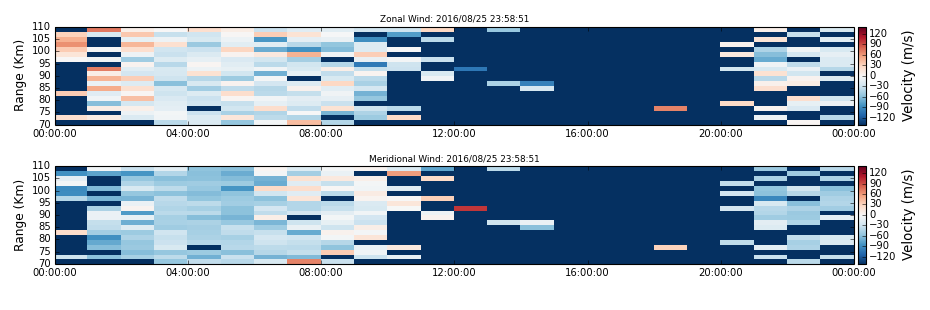
**5.1. Wind Profiler Plot**

En cuanto a las estimaciones de vientos, se debe verificar que no presenten variaciones bruscas en el tiempo, debido a que los vientos horizontales a dichas alturas (70-110 km) son relativamente constantes en el tiempo. Por otro lado, es importante resaltar que para horas de la tarde la cantidad de estimaciones se reducen, debido a que la cantidad de meteoros es considerablemente menor. En la Figura 11, se observa que para las horas entre la 13:00 y las 19:00 solo estimaciones para un par de alturas.



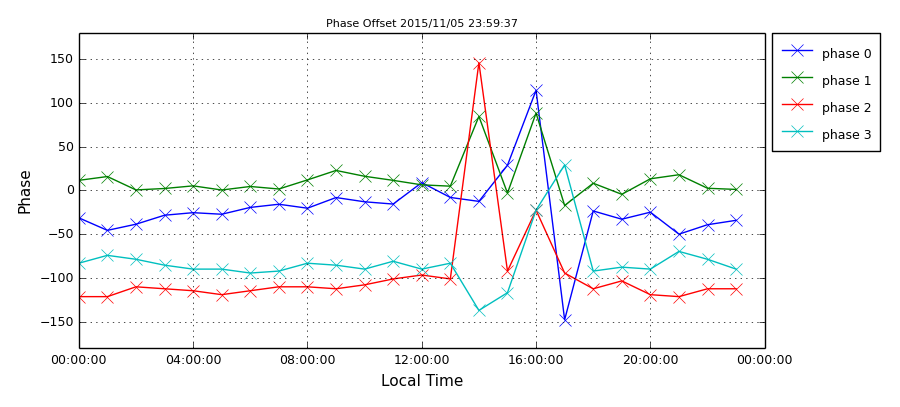
**Figura 11.** Wind Profiler Plot. Las estimaciones de vientos deben variar lentamente en el tiempo. Asimismo, durante las horas del día hay menos estimaciones

Un gráfico de estimaciones de vientos erróneo es, ciertamente, complicado de identificar en ciertos casos. En la Figura 12, se tiene un Wind Profiler Plot estimado con ecos de meteoros con offsets de fases sin corregir. Si bien presenta algunas variaciones bruscas, dichos cambios no son tan notorios. La única observación que se podría hacer es que casi no hay ninguna estimación para horas de la tarde (12:00 - 20:00). Dicho particularidad se puede explicar con el hecho de no haber corregido los offsets. Cuando no se realiza la corrección, la cantidad de ecos válidos se reduce y, por ende, el número de estimaciones de vientos disponibles.

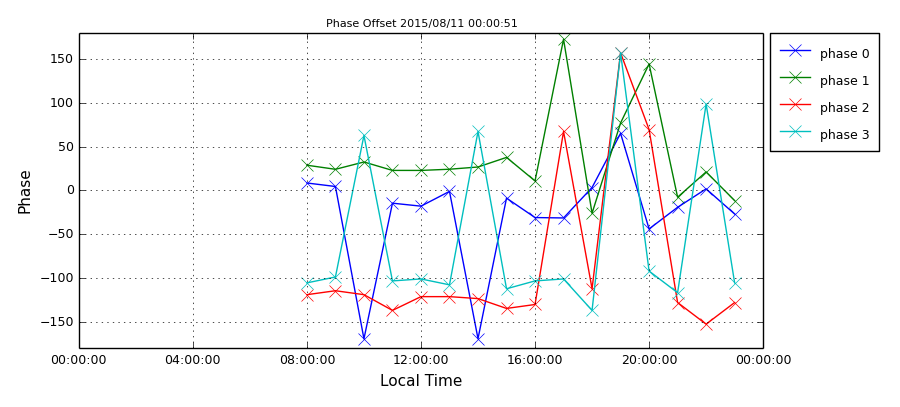


**Figura 12.** Wind Profiler Plot erróneo. Hay algunas variaciones bruscas en los vientos zonales. Asimismo, casi no hay estimaciones de vientos en la tarde

**5.2. Phase Plot**

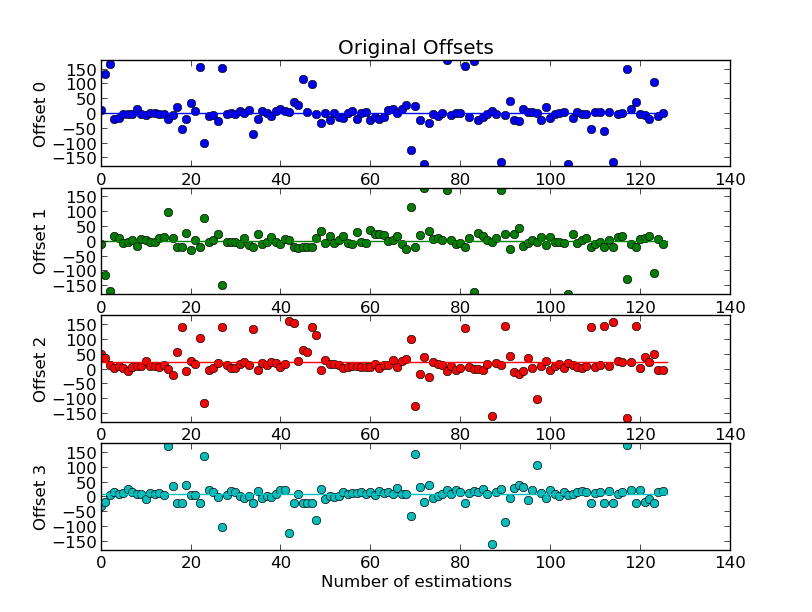


**Figura 13.** Phase Plot

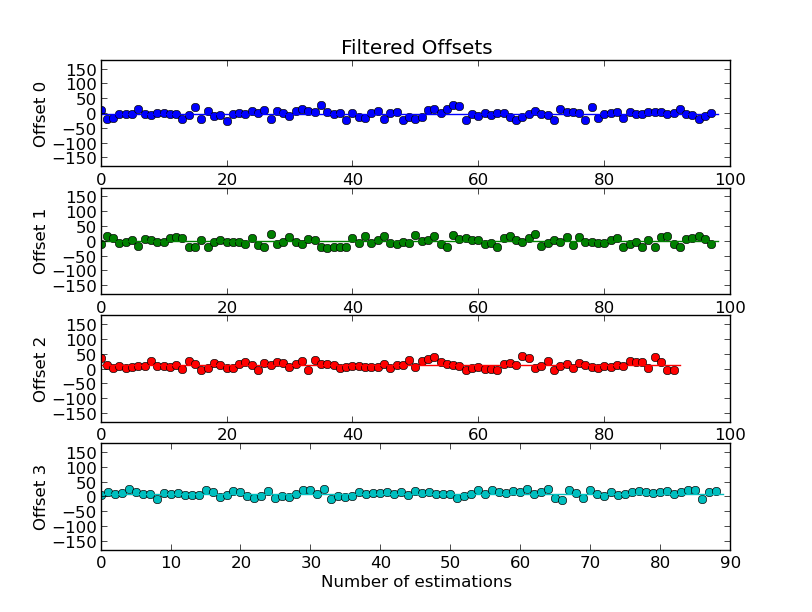


**Figura 14.** Phase Plot Erróneo

**5.3. Optimum Phase Plot**



**Figura 15.** Offsets de fases originales



**Figura 16.** Offset de fases filtrados

**6. BIBLIOGRAFÍA**

**7. ANEXOS**

**A. Optimum\_Offset.py**

import h5py

import numpy

import matplotlib.pyplot as plt

import glob

import os

#---------------------- Functions ---------------------

def findFiles(path):

dirList = []

fileList = []

for thisPath in os.listdir(path):

dirList.append(os.path.join(path,thisPath))

dirList.sort()

for thisDirectory in dirList:

files = glob.glob1(thisDirectory, "\*.hdf5")

files.sort()

for thisFile in files:

fileList.append(os.path.join(thisDirectory,thisFile))

return fileList

def readFiles(fileList):

meteors\_array = numpy.zeros((1,4))

for thisFile in fileList:

#Leer

f1 = h5py.File(thisFile,'r')

grp1 = f1['Data']

grp2 = grp1['data\_output']

meteors1 = grp2['table0'][:]

meteors\_array = numpy.vstack((meteors\_array,meteors1))

#cerrar

f1.close()

meteors\_array = numpy.delete(meteors\_array, 0, axis=0)

meteors\_list = [meteors\_array[:,0],meteors\_array[:,1],meteors\_array[:,2],meteors\_array[:,3]]

return meteors\_list

def estimateMean(offset\_list):

mean\_off = []

axisY\_off = []

axisX\_off = []

for thisOffset in offset\_list:

mean\_aux = numpy.mean(thisOffset, axis = 0)

mean\_off.append(mean\_aux)

axisX\_off.append(numpy.array([0,numpy.size(thisOffset)]))

axisY\_off.append(numpy.array([mean\_aux,mean\_aux]))

return mean\_off, axisY\_off, axisX\_off

def plotPhases(offset0, axisY0, axisX0, title):

f, axarr = plt.subplots(4, sharey=True)

color = ['b','g','r','c']

# plt.grid()

for i in range(len(offset0)):

thisMeteor = offset0[i]

thisY = axisY0[i]

thisX = axisX0[i]

thisColor = color[i]

opt = thisColor + 'o'

axarr[i].plot(thisMeteor,opt)

axarr[i].plot(thisX, thisY, thisColor)

axarr[i].set\_ylabel('Offset ' + str(i))

plt.ylim((-180,180))

axarr[0].set\_title(title + ' Offsets')

axarr[3].set\_xlabel('Number of estimations')

return

def filterOffsets(offsets0, stdvLimit):

offsets1 = []

for thisOffset in offsets0:

pstd = numpy.std(thisOffset)\*stdvLimit

pmean = numpy.mean(thisOffset)

outlier1 = thisOffset > pmean - pstd

outlier2 = thisOffset < pmean + pstd

not\_outlier = numpy.logical\_and(outlier1,outlier2)

thisOffset1 = thisOffset[not\_outlier]

offsets1.append(thisOffset1)

return offsets1

#---------------------- Setup ---------------------------

path = '/home/propietario/Imágenes/JASMET/201608/phase'

stdvLimit = 0.5

#---------------------- Script ---------------------------

fileList = findFiles(path)

offsets0 = readFiles(fileList)

mean0, axisY0, axisX0 = estimateMean(offsets0)

plotPhases(offsets0, axisY0, axisX0, 'Original')

offsets1 = filterOffsets(offsets0, stdvLimit)

mean1, axisY1, axisX1 = estimateMean(offsets1)

plotPhases(offsets1, axisY1, axisX1, 'Filtered')

print "Original Offsets: %.2f, %.2f, %.2f, %.2f" % (mean0[0],mean0[1],mean0[2],mean0[3])

print "Filtered Offsets: %.2f, %.2f, %.2f, %.2f" % (mean1[0],mean1[1],mean1[2],mean1[3])

plt.show()

**B. Publicación en base de datos Madrigal**

Para poder publicar los datos de las estimaciones de los vientos en la base de datos madrigal es necesario convertir los archivos HDF5 a un formato CEDAR compatible con madrigal; ello se realiza usando el script JASMET\_Madrigal.py (Anexo C).

Procedimiento a seguir:

* Transferencia de archivos al servidor de Madrigal:
  + Abrir un terminal (Ctrl + Alt +T):

scp -P 6633 -r path\_datos geodatos@jro-db.igp.gob.pe:/home/geodatos/JASMET

* + Donde path\_datos es la carpeta de que contiene los datos de los vientos (por ejemplo /home/usuario/JASMET/2016\_08/winds)
  + De ser necesario tambien podremos transferir el script JASMET\_Madrigal.py

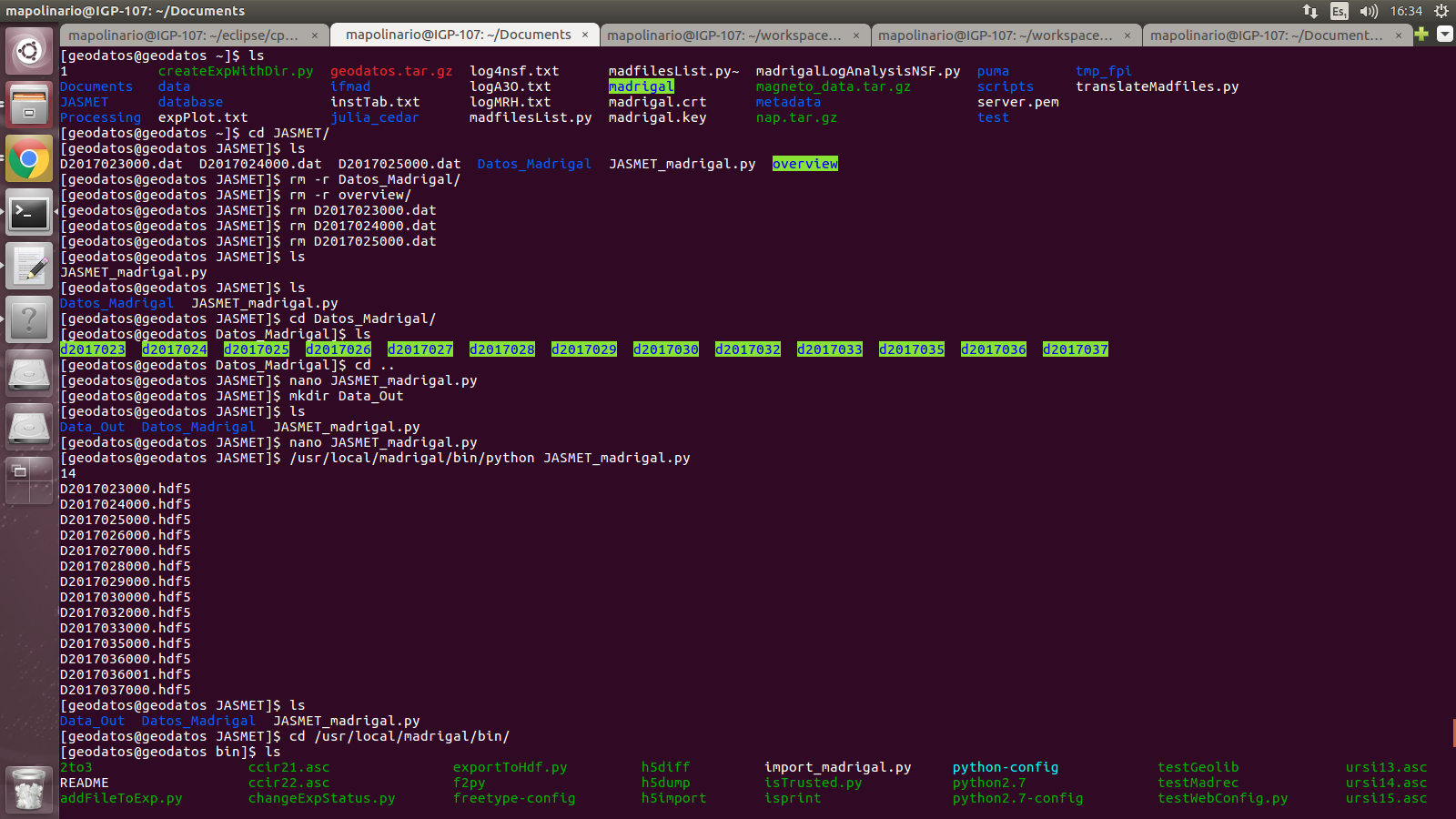
scp -P 6633 path geodatos@jro-db.igp.gob.pe:/home/geodatos/JASMET

* + Donde path sería la ruta del script (por ejemplo /home/usuario/JASMET/JASMET\_Madrigal.py )
* Conversion de formatos:
  + Abrir un terminal y conectar al servidor de madrigal mediante ssh, con el comando:

ssh geodatos@jro-db.igp.gob.pe -p 6633

* + La clave es: ge0d4t0sJR0!
  + Nos movemos a la carpeta que contiene el script JASMET\_Madrigal.py
  + Modificamos las variables path y pathout en el script con las direcciones de los directorios donde están los datos de los vientos y donde deseamos guardar los datos generados (.dat)
  + Ejecutamos el script con el comando:

/usr/local/madrigal/bin/python JASMET\_Madrigal.py

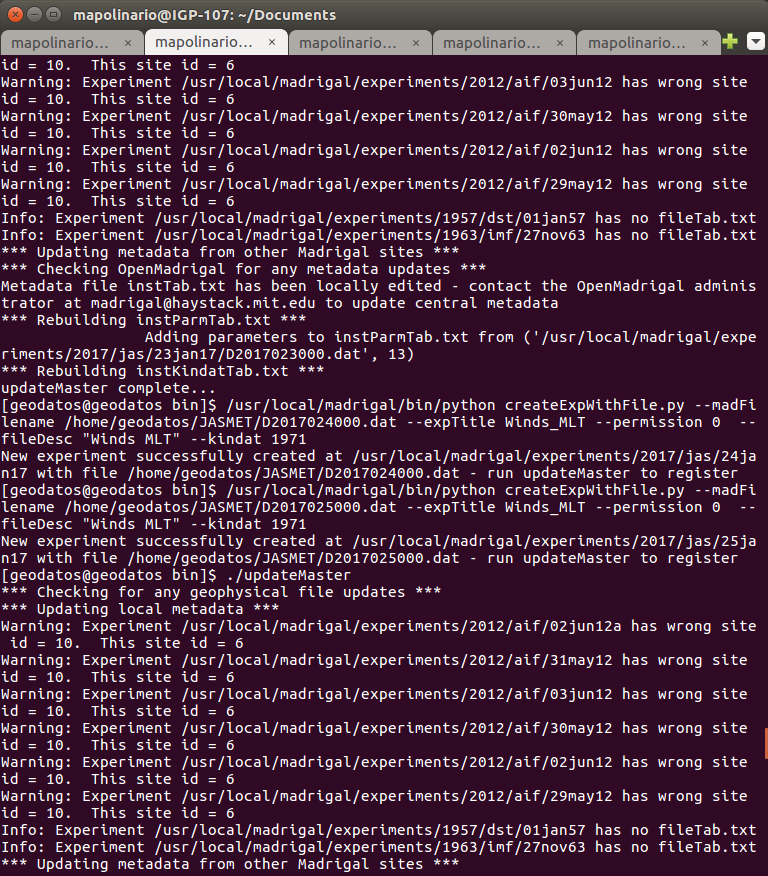


* Publicar datos en Madrigal:
  + Una vez logueados en el servidor de madrigal nos movemos al siguiente directorio:

cd /usr/local/madrigal/bin

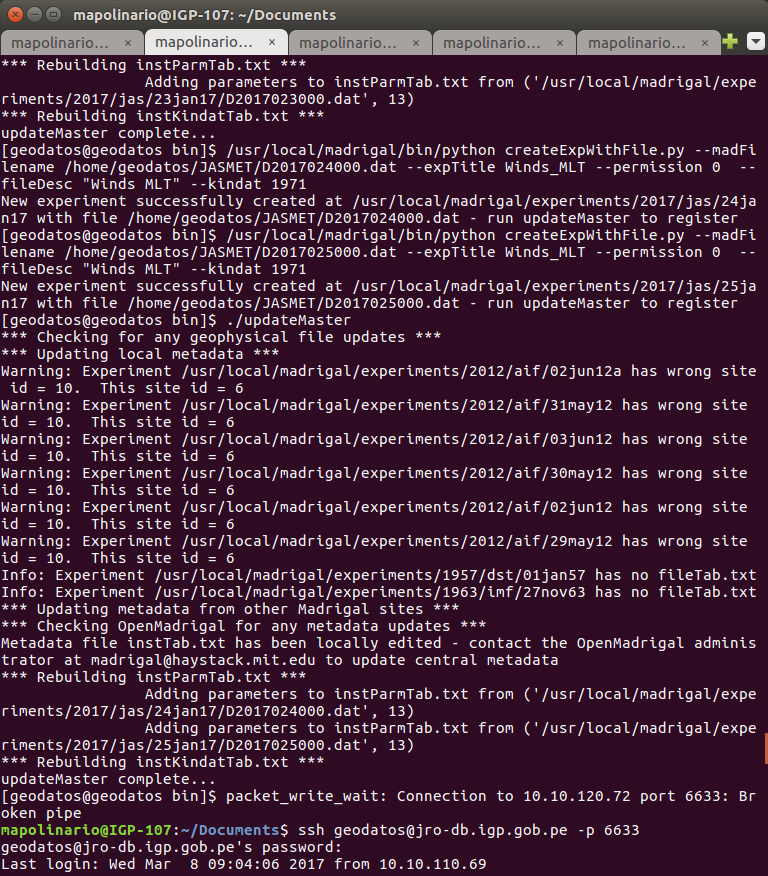
* + Usamos el siguiente comando:

/usr/local/madrigal/bin/python createExpWithFile.py --madFilename /home/geodatos/JASMET/D2017023000.dat --expTitle Winds\_MLT --permission 0 --fileDesc "Winds MLT" --kindat 1971



* + Donde “/home/geodatos/JASMET/D2017023000.dat” es la ruta da cada archivo .dat generado por el script JASMET\_Madrigal.py
  + Continuamos ejecutando el comando anterior para todos los datos que se quieren publicar y finalmente actualizamos la base de datos para poder visualizar los resultados en la web:

./updateMaster



**C. JASMET\_Madrigal.py**

import os, os.path

import types

from datetime import date

import datetime

import madrigal.metadata

import madrigal.cedar

import h5py

import glob

def fecha\_fcn(namefile):

year = int(namefile[1:5])

day = int(namefile[5:8])

relativedate = date(year,1,1)

date\_experiment = date.fromordinal(date.toordinal(relativedate)+day-1)

return date\_experiment

def findFiles(path):

dirList = []

fileList = []

nameList = []

for thisPath in os.listdir(path):

dirList.append(os.path.join(path,thisPath))

dirList.sort()

for thisDirectory in dirList:

files = glob.glob1(thisDirectory, "\*.hdf5")

files.sort()

for thisFile in files:

fileList.append(os.path.join(thisDirectory,thisFile))

nameList.append(thisFile)

return fileList, nameList

path = "/home/developer/JASMET/datos" #path de los archivos \*.hdf5

pathout = "/home/developer/JASMET/"

#namefile = glob.glob1(path,"\*.hdf5")[0]

fileList, nameList = findFiles(path)

for k in range(len(fileList)):

################# Read Data ###################

datos = h5py.File(fileList[k],"r") #leemos el archivo de datos

fecha = fecha\_fcn(nameList[k]) #establecemos la fecha del experimento

################# Data

print nameList[k]

tabla0 = datos["Data"]["data\_output"]["table0"][:,:]

tabla1 = datos["Data"]["data\_output"]["table1"][:,:]

utctime = datos["Data"]["utctimeInit"]

utctime = utctime.value[0]

################# Metadata

heightList = datos["Metadata"]["heightList"][:]

a\_dim\_dataout = datos["Metadata"]['array dimensions'][0]

a\_dim\_utctime = datos["Metadata"]['array dimensions'][1]

outputInterval = datos["Metadata"]['outputInterval']

timeZone = datos["Metadata"]["timeZone"]

typedata = datos["Metadata"]["type"]

datos.close()

GDALT = []

for f in range(len(utctime)):

GDALT.append(list(heightList))

tabla0lst = [range(25) for f in range(len(utctime))]

tabla1lst = [range(25) for f in range(len(utctime))]

################# sample data #################

kinst = 13 # instrument identifier

modexp = 700 # id of mode of experiment

kindat = 1971 # id of kind of data processing

nrow = 25 # all data records have 5 2D rows

tiempo\_init = datetime.datetime.utcfromtimestamp(utctime[0])

for f in range(len(utctime)):

for l in range(25):

if tabla0[f][l] != tabla0[f][l]:

tabla0lst[f][l] = 'missing'

else:

tabla0lst[f][l] = tabla0[f][l]

if tabla1[f][l] != tabla1[f][l]:

tabla1lst[f][l] = 'missing'

else:

tabla1lst[f][l] = tabla1[f][l]

################# end sample data #################

newFile = pathout+nameList[k][0:11]+'.dat'

# create a new Madrigal file

cedarObj = madrigal.cedar.MadrigalCedarFile(newFile, True)

# create all data records - each record lasts one minute

startTime = datetime.datetime(fecha.year, fecha.month, fecha.day, tiempo\_init.hour - 5 , tiempo\_init.minute, tiempo\_init.second, 0)

recTime = datetime.timedelta(0,3600)

for recno in range(len(utctime)):

endTime = startTime + recTime

dataRec = madrigal.cedar.MadrigalDataRecord(kinst,

kindat,

startTime.year,

startTime.month,

startTime.day,

startTime.hour,

startTime.minute,

startTime.second,

startTime.microsecond/10000,

endTime.year,

endTime.month,

endTime.day,

endTime.hour,

endTime.minute,

endTime.second,

endTime.microsecond/10000,

('GDLATR','GDLONR'), #latitud y longitud

('gdalt', 'vn2', 'vn1'), #altura, velocidad de iones este y velocidad de iones norte

nrow)

# set 1d values

dataRec.set1D('GDLATR', -1.19500e+01)

dataRec.set1D('GDLONR', -7.68700e+01)

# set 2d values

for n in range(nrow):

dataRec.set2D('gdalt', n, GDALT[recno][n])

dataRec.set2D('vn2', n, tabla0lst[recno][n]) # meridional wind

dataRec.set2D('vn1', n, tabla1lst[recno][n]) # zonal wind

# append new data record

cedarObj.append(dataRec)

startTime += recTime

# write new file

cedarObj.write()

# next, use the cedar.CatalogHeaderCreator class to add catalog and header

catHeadObj = madrigal.cedar.CatalogHeaderCreator(newFile)

catHeadObj.createCatalog(principleInvestigator="Marco Milla", sciRemarks=" ")

catHeadObj.createHeader(analyst=" ", comments=" ")

catHeadObj.write()