TONALLI

Dra. Lucía Adame Villanueva, Dr. Carlos Román Zúñiga, Dr. Jesús Hernández, Dr. Ricardo López Valdivia y M. en C. Edilberto Sánchez Moreno
Grupo de Formación Estelar.
UNAM, Instituto de Astronomía, Ensenada

1. Descripción del proyecto.

El código tonalli es una herramienta desarrollada en el Grupo de Formación Estelar del Instituto de Astronomía sede Ensenada, para obtener el mejor ajuste para un espectro estelar en la banda H, observado por APOGEE-2 (*Apache Point Observatory Galactic Evolution Experiment 2* [1], del *Sloan Digital Sky Survey*, o *Catastro Digital Celeste Sloan*, en español, **Figura 1**) mediante un algoritmo genético asexual [2]. Esto se consigue interpolando espectros sintéticos estelares (provenientes de una biblioteca pre-calculada) para contrastarlos contra el espectro observado. La comparación se cuantifica con una figura de mérito, y el algoritmo genético asexual tiene como objetivo minimizar tal figura de mérito, es decir, el encontrar un espectro sintético que sea lo más parecido al espectro observado (ver **Figura 2**). Una vez que se encuentra el mejor espectro sintético, del cual suponemos que es el que mejor explica la realidad, es posible derivar los parámetros físicos de la estrella observada, como la temperatura efectiva, la gravedad superficial, su metalicidad y su velocidad de rotación, entre otros. **Figura 3**

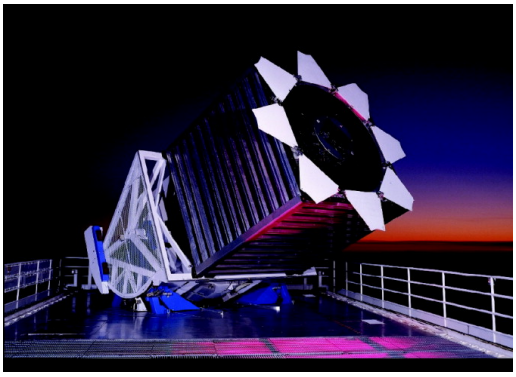

Figura 1. El telescopio 2.5 m del SDSS. Imagen tomada de [3]

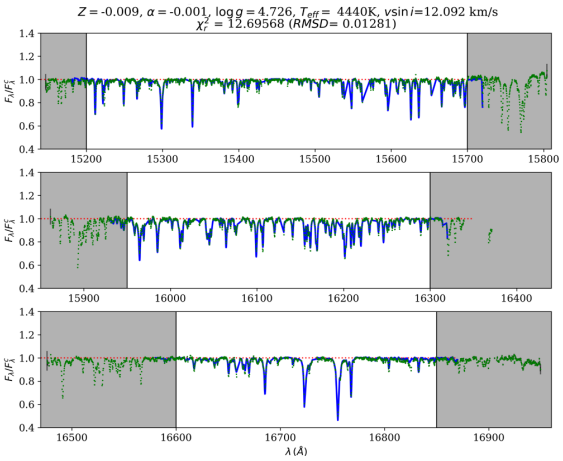

Figura 2. Espectro observado (en verde) comparado con un modelo de mejor ajuste (azul) obtenido por tonalli.

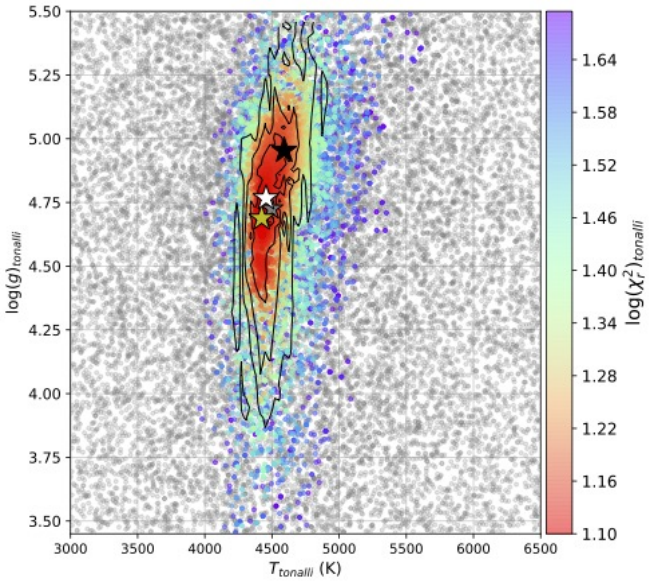

Figura 3. Gráfico de dispersión (temperatura contra logaritmo de la gravedad superficial) de todos los modelos interpolados por tonalli (círculos) para explicar el espectro observado de una estrella joven. El color indica aquellos modelos con una mejor figura de mérito, siendo los modelos en color rojo los modelos con la mayor probabilidad de ser el modelo de mejor ajuste, y en gris los modelos con la peor probabilidad de serlo. Las estrellas representan tres maneras estadísticas de obtener el mejor parámetro (estrellas blanca, gris y amarilla), mientras que la estrella de color negro representa una solución obtenida por otro trabajo.

2. Descripción de la instalación y ejecución del código Tonalli.

El código de Tonalli es un paquete desarrollado en Python 3, por lo que su instalación puede realizarse localmente utilizando la paquetería de de Anaconda3. La carpeta de instalación es proporcionada personalmente por el desarrollador y está integrada como se muestra en la **Figura 4.**

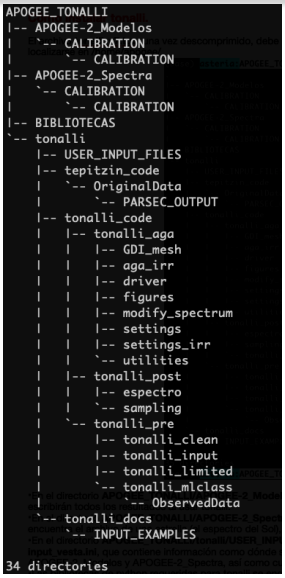

Figura 4. El archivo tonalli_v7.tgz, una vez descomprimido, debe contener esta estructura.

- En el directorio *APOGEE_TONALLI/APOGEE-2_Modelos/CALIBRATION/CALIBRATION/* se escribirán todos los resultados después de ejecutar Tonalli.
- En el directorio *APOGEE_TONALLI/APOGEE-2_Spectra/CALIBRATION/CALIBRATION/* se encuentra el archivo de una estrella (apStar-r12-VESTA.ts, el espectro del Sol).
- En el directorio APOGEE_TONALLI/tonalli/USER_INPUT_FILES/ se encuentra el archivo **input_vesta.ini**, que contiene información propias al código sobre dónde se localizan los directorios de calibración y resultados, así como la cantidad de núcleos del CPU que utilizará Tonalli para cada ejecución de una estrella, actualmente se utilizan 64 núcleos para realizar este proceso.
- Las bibliotecas adicionales de python requeridas para Tonalli, así como su instalación se encuentran descritas en el archivo **tonalli_install.sh**, por lo que se instalará Tonalli al ejecutar el archivo anterior.
- La ejecución de Tonalli utilizando una sola estrella se realiza de la siguiente forma:

  $ python **tonalli.py** *apStar-r12-VESTA* **input_vesta.ini**
- Para ejecutar N estrellas se puede utilizar puede utilizar un script como se muestra en la **Figura 5**.

```
(base) asteria:tonalli>more ejecuta_estrellas_par50.sh
python tonalli.py apStar-r12-21_LMi input_vesta.ini&
python tonalli.py apStar-r12-2M00391330+0308023 input_vesta.ini&
python tonalli.py apStar-r12-2M01054129+8719084 input_vesta.ini&
python tonalli.py apStar-r12-2M01075967+0159348 input_vesta.ini&
python tonalli.py apStar-r12-2M02515835+1122119 input_vesta.ini&
python tonalli.py apStar-r12-2M03033892-0539583 input_vesta.ini&
python tonalli.py apStar-r12-2M03082560+2619532 input_vesta.ini&
python tonalli.py apStar-r12-2M03182714+1510386 input_vesta.ini&
python tonalli.py apStar-r12-2M03183846+7216305 input_vesta.ini&
python tonalli.py apStar-r12-2M03470204+4125397 input_vesta.ini&
python tonalli.py apStar-r12-2M04425018+6644089 input_vesta.ini&
python tonalli.py apStar-r12-2M04575928+3416050 input_vesta.ini&
python tonalli.py apStar-r12-2M05100208-0704181 input_vesta.ini&
python tonalli.py apStar-r12-2M05373344+7441194 input_vesta.ini&
python tonalli.py apStar-r12-2M05414672+2252135 input_vesta.ini&
python tonalli.py apStar-r12-2M06563418+0109435 input_vesta.ini&
python tonalli.py apStar-r12-2M07090495+1525177 input_vesta.ini&
python tonalli.py apStar-r12-2M07385132-0527558 input_vesta.ini&
python tonalli.py apStar-r12-2M08054405+6822538 input_vesta.ini&
python tonalli.py apStar-r12-2M08102717+2530268 input_vesta.ini&
python tonalli.py apStar-r12-2M08113869+3227267 input_vesta.ini&
python tonalli.py apStar-r12-2M08172935-0359221 input_vesta.ini&
python tonalli.py apStar-r12-2M08423081+3411155 input_vesta.ini&
python tonalli.py apStar-r12-2M09221297+1232571 input_vesta.ini&
python tonalli.py apStar-r12-2M09421216-0746137 input_vesta.ini&
wait
python tonalli.py apStar-r12-2M10373678+1820584 input_vesta.ini&
```

Figura 5. Script secuencial utilizado para ejecutar 25 estrellas del código Tonalli.

Los requerimientos computacionales así como las bibliotecas adicionales requeridas para la ejecución del código Tonalli se presentan en la **Tabla 1.**

Tabla 1. Requerimientos de cómputo y bibliotecas adicionales.

Requerimientos de cómputo	
Nombre del código	tonalli.v7
Lenguaje de programación	Python3
Bibliotecas requeridas	astropy, GetDist, matplotlib, numpy, pandas, parmap, PyAstronomy, scikit-learn, scipy, specutils, statsmodels, terminaltables.
Almacenamiento	7 GB de Bibliotecas espectrales
Memoria RAM	20 GB por espectro
Tiempo de ejecución	2 Hrs por espectro
Archivo de Salida	11 MB
Procesadores por corrida	64

3. Características técnicas del servidor local en el Instituto de Astronomía Ensenada.

En el Instituto de Astronomía Ensenada, el proyecto TONALLI cuenta con un servidor local dedicado a este proyecto, el cual cuenta con las siguientes características.

- Procesador AMD Ryzen™ Threadripper™ 3990X ,(2.9 up to 4.3 GHz)
- Memoria RAM de 256 GB
- SSD 1TB, PCI Express 4.0, NVMe, M.2 Velocidad de lectura: 7400 MB/s, Velocidad de escritura: 6400 MB/s.

4. Descripción de ajuste a ejecución en Grid.

Se elaboró un script adaptado a las condiciones de ejecución propias de la Grid UNAM, el cual realiza la siguiente secuencia:

1. Descarga la paquetería propia del código almacenado en el S3 cargado previamente por el usuario.
2. Instala Python 3.10 en el directorio de trabajo local, así como las bibliotecas propias del código.
3. Ejecuta la secuencia el código Tonalli *N* veces de acuerdo a la lista de parámetros previamente cargada por el usuario.
4. Comprime la carpeta de resultados generados por el código.
5. Transfiere el comprimido al sistema de almacenamiento S3 cargado por el usuario previo a la ejecución del script.
6. Borra la carpeta donde se ejecutó el script.

Adicionalmente se enlistan las consideraciones técnicas que se tomaron en cuenta antes y durante la ejecución del código.

- Se realizó una corrida inicial con una estrella en tres entidades de la Grid UNAM (DGTIC, Nucleares y Astronomía Ensenada) con el objetivo de identificar los alcances del script utilizado para la ejecución del código. Se realizaron observaciones y correcciones con respecto a la estrategia utilizada antes y después de ejecutarlos.
- Se redujo la carga de trabajo en los nodos de la Grid UNAM de 64 núcleos a 24 núcleos, esto considerando que el código ha estado operando en los servidores locales del Instituto de Astronomía sede Ensenada con 64 núcleos.

- Se ejecutó la prueba total de 25 estrellas en las tres entidades mencionadas anteriormente, haciendo un total de 75 ejecuciones del código Tonalli.
- Los resultados obtenidos en tiempo de ejecución son resultado de las diferentes arquitecturas en los diferentes nodos de procesamiento.

5. Tiempo de ejecución.

Los resultados de tiempo de ejecución realizados en las diferentes entidades de la Grid UNAM se muestran en la **Tabla 2** y **Tabla 3**.

Tabla 2. Muestra los resultados de tiempo de ejecución del código TONALLI utilizando una estrella con 24 núcleos de procesamiento.

Entidad	Tiempo en instalación	Tiempo de Ejecución	Tiempo total
jamatu.astrosen.unam.mx (IA-Ensenada)	8 minutos	4 hrs, 50 min.	4 hrs, 58 min.
submit.grid.unam.mx (DGTIC)	3 minutos	1 hrs, 20 min.	1 hrs, 23 min.
cuezcomatl.nucleares.unam.mx (LAMOD)	3hrs, 10min,	2 hrs, 44 min.	5 hrs, 54 min.

Tabla 3. Muestra los resultados de tiempo de ejecución del código TONALLI utilizando 25 estrellas con 24 núcleos de procesamiento.

Entidad	Tiempo en instalación	Tiempo de Ejecución	Tiempo total
jamatu.astrosen.unam.mx (IA-Ensenada)	8 min	5 días.	5 días
submit.grid.unam.mx (DGTIC)	3 min	33 hrs 20 min.	33 hrs, 23 min.
cuezcomatl.nucleares.unam.mx (LAMOD)	2 hrs 27 min	50 hrs 34 min.	53 hrs, 1 min.

6. Cantidad de recursos utilizados en Grid vs Recursos locales

En la **Tabla 4** se presenta una tabla comparativa utilizando el servidor local y los recursos utilizados en la Grid UNAM utilizando solo una estrella.

Tabla 4. Comparación en tiempos de ejecución del código Tonalli entre el servidor local y los diferentes nodos de la Grid UNAM con solo una estrella.

Entidad	CPU	Memoria	Tiempo de ejecución	Almacenamiento
Local (IA-Ensenada)	**25**	**22 GB**	**1 hrs 6 min.**	**7 GB**
jamatu.astrosen.unam.mx (IA-Ensenada)	**25**	**22 GB**	**4 hrs, 50 min.**	**7 GB**
submit.grid.unam.mx (DGTIC)	**25**	**22 Gb**	**1 hrs, 20 min.**	**7 GB**
cuezcomatl.nucleares.unam.mx (Nucleares)	**25**	**22 GB**	**2 hrs, 44 min.**	**7 GB**

7. Comparación de resultados : recursos propios vs Grid

Es observable que el tiempo de trabajo aumentó en algunos de los nodos de la Grid UNAM en comparación con el servidor local del Instituto de Astronomía sede Ensenada, esto se debe a la arquitectura individual de cada cluster de la Grid UNAM. Sin embargo, como se puede ver en la **Tabla 3**, el uso de los recursos compartidos entre los diferentes nodos de la Grid UNAM benefició significativamente a este proyecto, pues en la misma tabla se aprecia que el mejor rendimiento de este código se registró en el cluster de DGTIC donde los resultados en procesamiento son bastante comparables a los generados localmente.

8. Conclusiones

Bajo la arquitectura de la Grid UNAM se realizó la ejecución de 75 espectros estelares entre las tres entidades mencionadas anteriormente (LAMOD, DGTIC y IA Ensenada), al finalizar las corridas se obtuvieron resultados en tiempo de producción similar a los que se reciben en el servidor local del proyecto. Estos tiempos de ejecución se pueden mejorar al cambiar la estrategia utilizada en la ejecución por manejo de contenedores. Si bien los resultados en tiempo se acercan a los que se obtienen localmente, el uso de la Grid UNAM le permitiría al usuario generar más análisis espectrales utilizando más recursos computacionales una vez que localmente se cuenta con un solo servidor para dicho proyecto.

Referencias:

[1] Majewski, S. R., “The Apache Point Observatory Galactic Evolution Experiment (APOGEE)”, The Astronomical Journal, vol. 154, no. 3, 2017. doi:10.3847/1538-3881/aa784d.

[2] Cantó, J., Curiel, S., y Martínez-Gómez, E., “A simple algorithm for optimization and model fitting: AGA (asexual genetic algorithm)”, Astronomy and Astrophysics, vol. 501, no. 3, pp. 1259–1268, 2009. doi:10.1051/0004-6361/200911740.

[3] Gunn, J. E., “The 2.5 m Telescope of the Sloan Digital Sky Survey”, The Astronomical Journal, vol. 131, no. 4, pp. 2332–2359, 2006. doi:10.1086/500975.