Astronomía
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

INFORME CINEMÁTICA GALÁCTICA

Tarea 2

Estudiante: Camilo Núñez Barra
RUT: 20.533.326-6
Profesor: Leonardo Bronfman
Auxiliar: Paulina Palma

Curso: AS3201 Astronomía Experimental
Fecha: 30 de mayo de 2021

1. Introducción

2. Curva de rotación

2.1. Marco teórico

Sea P un punto a través de la línea de visión a longitud l y de distancia galactocéntrica $R = R_\odot \sin l$. Se supone P bajo un movimiento puramente circular con velocidad $\vec{v}(R)$, denominada velocidad rotacional, cuya componente a lo largo de la línea de visión, es decir, su velocidad radial medible a través del efecto Doppler, es $\vec{v}_\parallel$, formando un ángulo α entre ambas, por lo tanto, contrarrestando la velocidad del Sol $\vec{v}_\odot$ relativa a la línea de visión,

$$v_{\mathrm{LSR}}(R) = v(R)\cos\alpha - v_\odot \sin l. \tag{2.1}$$

Sea β el ángulo interior que forma el Sol, el punto P y el centro galáctico. Se aprecia que $\beta = 90 + \alpha$, y, usando el teorema del seno,

$$\frac{\sin l}{R} \equiv \frac{\sin\beta}{R_\odot} = \frac{\cos\alpha}{R_\odot}, \tag{2.2}$$

de modo que la ecuación 2.1 se reescribe como,

$$v_{\mathrm{LSR}}(R) = v(R)\frac{R_\odot \sin l}{R} - v_\odot \sin l, \tag{2.3}$$

y, definiendo la velocidad angular mediante la siguiente relación,

$$v(R) = \omega(R)R, \tag{2.4}$$

se obtiene la ecuación maestra de la cinemática de la Galaxia,

$$v_{\mathrm{LSR}}(R) = \left(\omega(R) - \omega(R_\odot)\right) R_\odot \sin l. \tag{2.5}$$

Para una misma velocidad $v_{\mathrm{LSR}}(R)$, existe una doble ambigüedad en la distancia heliocéntrica D del punto P que le corresponde, pues, dentro del círculo solar, la línea de visión intersecta a la circunferencia de radio R en un punto cercano y en otro lejano. Esto no ocurre fuera del círculo solar.

Es importante determinar la distancia D para poder calcular la densidad numérica y densidad de masa gracias a la luminosidad, y, si no se determina, solo se puede calcular la densidad de columna.

Sea $D_{\rm N}$ la distancia cercana y $D_{\rm F}$ la distancia lejana. Geométricamente, se obtiene,

$$D_{\substack{F\\N}} = R_\odot \cos l \pm \sqrt{R^2 - R_\odot^2 \sin^2 l} \qquad (2.6)$$

La velocidad máxima $v_{\rm LSR}^{\rm máx}(R)$ que se puede medir en la línea de visión ocurre en el punto P sobre esta que tiene la menor distancia galactocéntrica $R_{\rm mín} = R_\odot \sin l$, suponiendo ω monotónicamente decreciente con respecto a R. Este punto se denomina punto subcentral. Su distancia heliocéntrica es $D_{\rm N} = D_{\rm F} = R_\odot \cos l = D_{\rm tan}$ y su velocidad tangencial o rotacional es, gracias a la ecuación 2.3.

$$v_{\rm rot}(R_\odot \sin l) = v_{\rm LSR}(R_\odot \sin l) + v_\odot \sin l, \quad (2.7)$$

y su velocidad angular es,

$$\omega(R_\odot \sin l) = \frac{v_{\rm LSR}}{R_\odot \sin l} + \omega(R_\odot). \qquad (2.8)$$

Las velocidades rotacionales permitidas en la galaxia interna, en donde $R < R_\odot$ y luego $\omega(R) > \omega(R_\odot)$, para el cuarto cuadrante galáctico, en donde las longitudes está en el rango $270° < l < 360°$, de modo que $\sin l < 0$, cumplen, según la ecuación 2.5, con $v_{\rm rot} < v(R) < 0$.

2.2. Detalle del algoritmo

La curva de rotación se deriva de las velocidades terminales del espectro de CO para cada longitud galáctica l. El cuarto cuadrante galáctico tiene emisión con corrimiento al azul, por lo tanto, se escogen las velocidades terminales más negativas posibles para cada espectro y que cumplan cierta condición que se explica a continuación. El procedimiento corresponde al código 1.

Se utiliza la técnica de *sigma clipping* para obtener la base de ruido de los espectros con un número de 5 desviaciones estándar para el límite de recorte tanto superior como inferior, además de las iteraciones necesarias para que el algoritmo converja. Suponiendo que el promedio del ruido se anula, entonces la temperatura de ruido o valor RMS del ruido es igual a la desviación estándar del mismo.

La velocidad terminal se define para cada longitud l y para cada latitud b como aquella velocidad del primer punto del espectro de emisión cuya temperatura sea cinco veces mayor a la temperatura de ruido, considerando primero las velocidades del lado del corrimiento al azul del espectro, es decir, recorriendo el espectro desde la velocidad más negativa a la más positiva. Esta condición se muestra en la figura 1.

A continuación, se escoge para cada longitud l el máximo maximorum entre las velocidades terminales de todas las latitudes b.

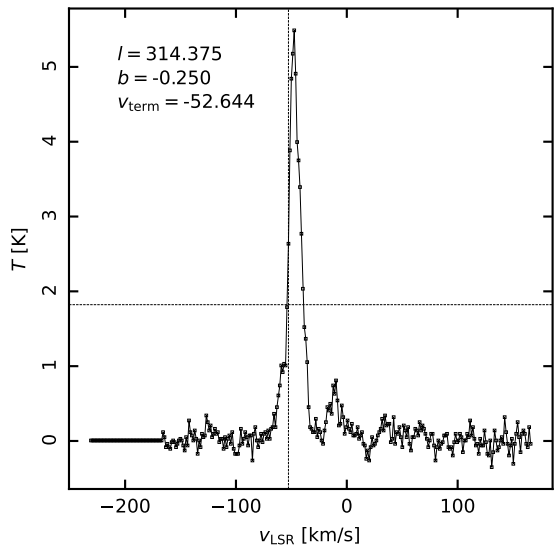

Figura 1: Un típico espectro de emisión, para las coordenadas $l = 314.385°$ y $b = -0.250°$, con la condición para elegir la velocidad terminal. La línea discontinua horizontal representa un nivel de 5σ de ruido. La primera temperatura del lado del corrimiento al azul en superar esta condición ocurre a una velocidad $v_{\rm LSR} = -52.644\,{\rm km\,s^{-1}}$, que corresponde a la línea discontinua vertical.

2.3. Curva de rotación: $v_{\rm rot}$ vs. R y ω vs. R

Se usa la ecuación 2.7 y los máximos maximorum de velocidades terminales para graficar en la figura 2 las velocidades rotacionales para cada longitud.

Se utiliza la ecuación 2.8 junto con los máximos maximorum de velocidades terminales para graficar en la figura 2 las velocidades angulares para cada longitud.

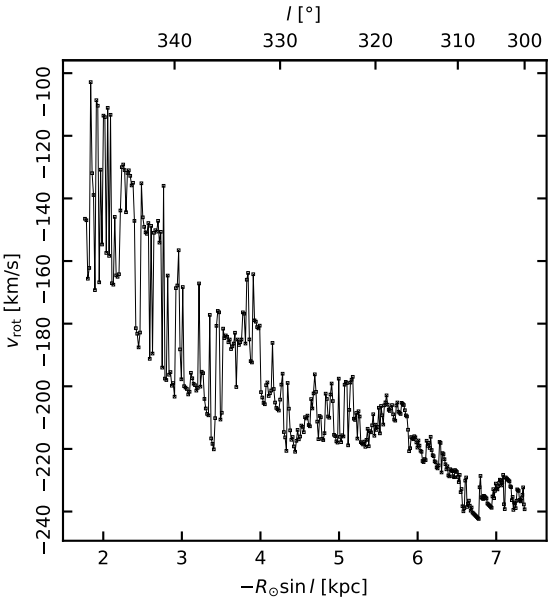

Figura 2: Curva de rotación del cuarto cuadrante galáctico. Velocidad rotacional, correspondiente al máximo maximorum de velocidades terminales de las latitudes, en función de la distancia galactocéntrica (eje inferior) y la longitud (eje superior).

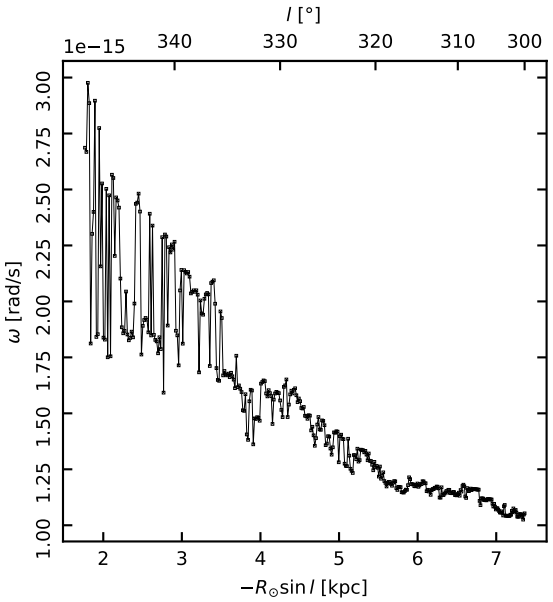

Figura 3: Curva de rotación del cuarto cuadrante galáctico. Velocidad angular de cada máximo maximorum de velocidad terminal en función de la distancia galactocéntrica (eje inferior) y la longitud (eje superior).

3. Corrugación del plano

3.1. Marco teórico

El disco galáctico no es plano, sino que tiene una corrugación de aproximadamente 150 pc, que es mucho menor comparado con los aproximadamente 15 kpc de radio. La corrugación de la altura Z a través la distancia galactocéntrica R se puede asimilar a un parche de un tambor de espesor minimal, puesto que la Galaxia tiene una estructura ondulatoria que depende en coordenadas cilíndricas de θ y r y que funciona como una onda espiral de densidad y en R hay modos normales de vibración, luego, se puede expresar analíticamente como una composición de funciones de Bessel, pero para este informe se hace un análisis más sencillo a partir de los datos medidos.

Gracias al algoritmo de la sección anterior, se toma como la posición de densidad máxima para cada longitud l a la latitud b donde se encuentra el máximo maximorum de velocidad terminal.

Se sabe también de la sección anterior que en este punto subcentral tangente al círculo la distancia galactocéntrica es $D = R_\odot \sin l$ y la distancia heliocéntrica es $D = R_\odot \sin l$. Geométricamente, la altura para este punto respecto al ecuador galáctico es,

$$Z = R_\odot \sin l \tan b, \tag{3.1}$$

pudiendo aproximar $\tan \approx b$ para un régimen de pequeñas oscilaciones.

3.2. Detalle del algoritmo

Se utiliza la latitud b del máximo maximorum de velocidad para cada longitud l del código 1 para calcular en el código 2 la altura en cada longitud del disco según la ecuación 3.1.

3.3. Corrugación del plano: Z vs. R

La figura 4 muestra la corrugación del disco galáctico en función del radio galactocéntrico.

4. Ajuste de modelo de masa

4.1. Marco teórico

Se quiere encontrar una fórmula analítica para la curva de rotación basándose en un modelo

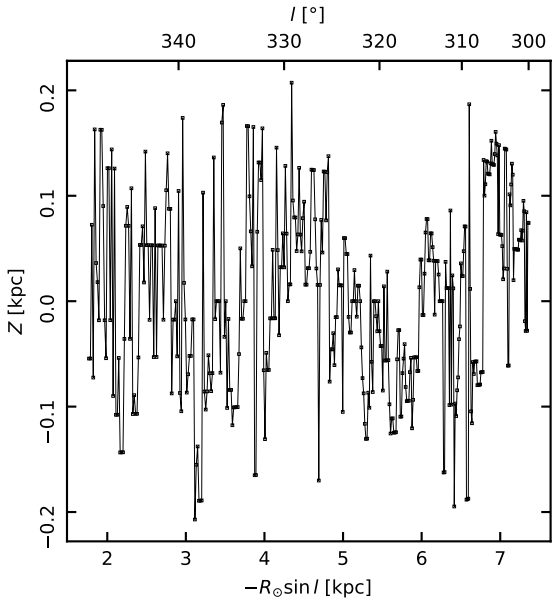

Figura 4: Corrugación del plano. Altura Z de cada punto subcentral con respecto a la distancia galactocéntrica (eje inferior) y la longitud correspondiente (eje superior).

físico que considere la distribución de masa de la galaxia.

Se iguala para cierta partícula de masa m en rotación pura la fuerza de gravedad con la fuerza centrípeta,

$$G\frac{M(R)m}{R^2} = m\frac{v_{\text{rot}}^2(R)}{R}, \tag{4.1}$$

donde G es la constante de gravitación universal y M es la masa de la galaxia dentro de un radio R determinado por la distancia galactocéntrica de la partícula. Luego, la velocidad de rotación de la curva de rotación de la figura 2.7 se puede modelar por,

$$v_{\text{rot}}(R) = \sqrt{G\frac{M(R)}{R}}. \tag{4.2}$$

Se estudian cinco modelos de distribución de masa para la galaxia, listados a continuación.

- Masa puntual en el centro de la galaxia. $M(R) = M_0$. M_0 es el parámetro libre de la masa puntual.
- Disco uniforme. $M(R) = \pi R^2 s_0$. s_0 es el parámetro libre de densidad superficial uniforme de masa.
- Esfera uniforme. $M(R) = \frac{4}{3}\pi R^3 \rho_0$. ρ_0 es el parámetro libre de densidad uniforme volumétrica de masa.
- Disco uniforme con una masa puntual en el centro. $M(R) = \pi R^2 s_0 + M_0$. Dos parámetros libres, s_0 y M_0.
- Esfera uniforme con una masa puntual en el centro. $M(R) = \frac{4}{3}\pi R^3 \rho_0 + M_0$. Dos parámetros libres, rho_0 y M_0.

4.2. Detalla del algoritmo

El código 3 muestra el sigueinte procedimiento. Se evalúan los cinco modelos de distribución de masa en la ecuación 4.2 y se realiza un ajuste no lineal de mínimos cuadrados según el algoritmo de Levenberg-Marquardt para cada modelo, asumiendo una desviación estándar de la unidad para las velocidades de rotación. Además, se calcula el error RMS de cada modelo con respecto a las velocidades de rotación medidas para determinar el mejor ajuste.

4.3. Ajuste de los modelos de la curva de rotación

La tabla 1 muestra el resultado del ajuste de mínimos cuadrados con los parámetros óptimos para cada modelo de distribución de masa de la Galaxia y el error RMS con respecto a la velocidad rotacional medida.

La figura 5 muestra el gráfico de los ajustes de los cinco modelos de distribución de masa de la Galaxia.

5. Análisis y conclusiones

Modelo	M_0 $M_\odot$	s_0 $M_\odot\,kpc^{-2}$	ρ_0 $M_\odot\,kpc^{-3}$	RMS $km\,s^{-1}$
Punto	$3{,}655 \times 10^{10} \pm 1{,}321 \times 10^{9}$	—	—	68,738
Disco	—	$6{,}419 \times 10^{8} \pm 5{,}501 \times 10^{6}$	—	17,248
Esfera	—	—	$8{,}569 \times 10^{7} \pm 1{,}883 \times 10^{6}$	43,337
Disco + Punto	$3{,}928 \times 10^{9} \pm 3{,}422 \times 10^{8}$	$5{,}814 \times 10^{8} \pm 6{,}851 \times 10^{6}$	—	14,749
Esfera + Punto	$1{,}259 \times 10^{10} \pm 4{,}343 \times 10^{8}$	—	$6{,}127 \times 10^{7} \pm 1{,}112 \times 10^{6}$	21,326

Cuadro 1: Parámetros y error obtenido del ajuste no lineal de míminos cuadrados para los modelos de distribución de masa de la Galaxia con respecto a la velocidad rotacional medida.

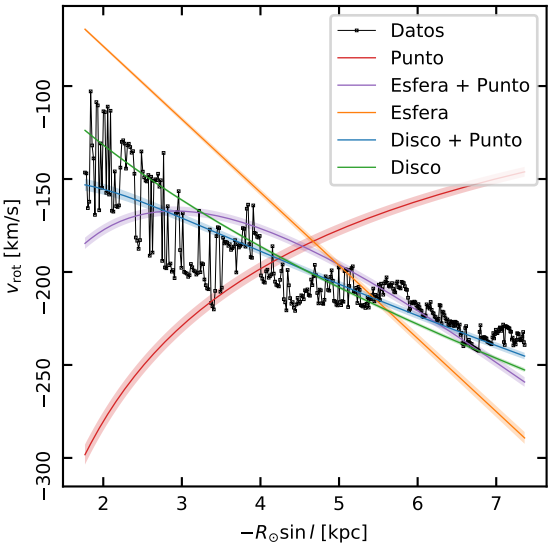

Figura 5: Ajuste de los modelos de distribución de masa de la Galaxia con respecto a la curva de rotación medida. Línea negra es datos medidos. Línea naranja es modelo de masa puntual central. Línea verde es modelo de disco uniforme. Línea naranja es modelo de esfera uniforme. Línea azul es modelo de disco uniforme con masa puntual central. Línea morada es modelo de esfera uniforme con masa puntual central. Las cinco líneas de ajustes tienen un área respectiva que representa el error de los parámetros óptimos.

6. Anexos

Código 1: Curva de rotación

```python
#!/usr/bin/python3

import numpy as np
from astropy.io import fits
from astropy.stats import SigmaClip
import matplotlib.pyplot as plt
from astropy import units as u

R0 = 8.5*u.kpc
v0 = 220*u.km/u.s

def get_values_from(header: fits.header.Header, axis: int) -> np.ndarray:
    """Get real coordinate values instead of pixels from `header` at `axis`."""
    naxis = header[f"NAXIS{axis}"]  # number of pixels on `axis`
    crpix = header[f"CRPIX{axis}"]  # reference pixel for `axis`
    crval = header[f"CRVAL{axis}"]  # coordinate value at `crpix`
    cdelt = header[f"CDELT{axis}"]  # pixel spacing for `axis`
    return crval + (1 - crpix + np.arange(naxis)) * cdelt

cube = fits.open("../southgal_fixbadc.fits")
data = cube[0].data  # indexing: [galactic latitude, galactic longitude, velocity]
header = cube[0].header

v = get_values_from(header, 1)  # VELO-LSR
l = get_values_from(header, 2)  # GLON-FLT
b = get_values_from(header, 3)  # GLAT-FLT

v_maximorum = np.zeros_like(l)
b_v_maximorum = np.zeros_like(l)

sigclip = SigmaClip(sigma_lower=3.5, sigma_upper=3.5, maxiters=None, cenfunc=np.mean, stdfunc=np.
    std)
T_noise = sigclip(data, axis=2)
rms = np.sqrt(np.mean(T_noise**2, axis=2))
T_rms = np.repeat(rms[..., np.newaxis], len(v), axis=2)
iv_terminal = (data > 5*T_rms).argmax(axis=2)
for il in range(len(l)):
    v_terminal = np.zeros_like(b)
    for ib in range(len(b)):
        i = iv_terminal[ib][il]
        v_terminal[ib] = v[i] if i else np.nan
    i = np.nanargmin(v_terminal)
    v_maximorum[il] = v_terminal[i]
    b_v_maximorum[il] = b[i]

vrot = v0*np.sin(l*np.pi/180.)+v_maximorum*u.km/u.s
R = R0*np.sin(l*np.pi/180.)

if __name__ == "__main__":

    plt.rcParams.update({'font.size': 7})
    fig, ax = plt.subplots(figsize=(3.25, 3.25))
    ax.plot(v, data[14][115], c='k', lw=0.25, marker='s', markersize=1, mfc="none", markeredgewidth
    =0.25)
    ax.axhline(5*rms[14][115], c='k', lw=0.25, ls="--")
    ax.axvline(v[iv_terminal[14][115]], c='k', lw=0.25, ls="--")
    ax.set_xlabel(r"$v_\mathrm{LSR}$ [km/s]")
    ax.set_ylabel(r"$T$ [K]")
    ax.text(0.1, 0.9, f"$l=${l[115]:.3f}", ha='left', va='center', transform=ax.transAxes)
    ax.text(0.1, 0.85, f"$b=${b[14]:.3f}", ha='left', va='center', transform=ax.transAxes)
    ax.text(0.1, 0.8, r"$v_\mathrm{term}=$"+f"{v[iv_terminal[14][115]]:.3f}", ha='left', va='center
    ', transform=ax.transAxes)
    ax.yaxis.set_tick_params(rotation=90)
    ax.tick_params(direction="in", top=True, right=True)
    fig.savefig("../informe/rsc/vterminal.pdf")
    plt.show()

```

```
    w0 = v0/R0.to(u.km)

    w = w0+(v_maximorum*u.km/u.s)/R.to(u.km)

    plt.rcParams.update({'font.size': 7})
    fig, ax = plt.subplots(figsize=(3.25, 3.25))
    ax.plot(-R, vrot, c='k', lw=0.25, marker='s', markersize=1, mfc="none", markeredgewidth=0.25)
    ax.set_xlabel(r"$-R_{\odot}\sin\,l$ [kpc]")
    ax.set_ylabel(r"$v_\mathrm{rot}$ [km/s]")
    ax.yaxis.set_tick_params(rotation=90)
    ax.tick_params(direction="in", top=False, right=True)
    axl = ax.secondary_xaxis("top", functions=(lambda R: (180./np.pi*np.arcsin(-R/R0.value)-360)%
    360, lambda l: -R0.value*np.sin(l*np.pi/180.)))
    axl.set_xlabel(r"$l$ [$^{\circ}$]")
    axl.tick_params(direction="in", top=True)
    fig.savefig("../informe/rsc/vrot.pdf")
    plt.show()

    plt.rcParams.update({'font.size': 7})
    fig, ax = plt.subplots(figsize=(3.25, 3.25))
    ax.plot(-R, w, c='k', lw=0.25, marker='s', markersize=1, mfc="none", markeredgewidth=0.25)
    ax.set_xlabel(r"$-R_{\odot}\sin\,l$ [kpc]")
    ax.set_ylabel(r"$\omega$ [rad/s]")
    ax.yaxis.set_tick_params(rotation=90)
    ax.tick_params(direction="in", top=False, right=True)
    axl = ax.secondary_xaxis("top", functions=(lambda R: (180./np.pi*np.arcsin(-R/R0.value)-360)%
    360, lambda l: -R0.value*np.sin(l*np.pi/180.)))
    axl.set_xlabel(r"$l$ [$^{\circ}$]")
    axl.tick_params(direction="in", top=True)
    fig.savefig("../informe/rsc/w.pdf")
    plt.show()
```

Código 2: Corrugación del plano galáctico

```
#!/usr/bin/python3

import numpy as np
import matplotlib.pyplot as plt

from curvaderotacion import R0, R, l, b_v_maximorum

def Z(l: float, b: float) -> float:
    return R0*np.cos(l*np.pi/180.)*np.tan(b*np.pi/180.)

plt.rcParams.update({'font.size': 7})
fig, ax = plt.subplots(figsize=(3.25, 3.25))
ax.plot(-R, Z(l, b_v_maximorum), c='k', lw=0.25, marker='s', markersize=1, mfc="none",
    markeredgewidth=0.25)
ax.set_xlabel(r"$-R_{\odot}\sin\,l$ [kpc]")
ax.set_ylabel(r"$Z$ [kpc]")
ax.yaxis.set_tick_params(rotation=90)
ax.tick_params(direction="in", top=False, right=True)
axl = ax.secondary_xaxis("top", functions=(lambda R: (180./np.pi*np.arcsin(-R/R0.value)-360)%360,
    lambda l: -R0.value*np.sin(l*np.pi/180.)))
axl.set_xlabel(r"$l$ [$^{\circ}$]")
axl.tick_params(direction="in", top=True)
fig.savefig("../informe/rsc/Z.pdf")
plt.show()
```

Código 3: Ajuste de los modelos de distribución de masa de la Galaxia

```
#!/usr/bin/python3

import numpy as np
import matplotlib.pyplot as plt
from astropy import units as u
from scipy.optimize import curve_fit

from curvaderotacion import R, vrot

```

```
G = 4.302e-6*u.kpc*u.km**2/u.s**2/u.Msun

vrot = -vrot.value
R = -R.value

def vel(R: float, M: float) -> float:
    return np.sqrt(G.value*M/R)

def pointlike_mass_model(M0: float) -> float:
    """`M0` point mass in the center of the Galaxy."""
    return M0

def uniform_disk_mass_model(R: float, s: float) -> float:
    """`s` uniform superficial mass density."""
    return np.pi*R**2*s

def uniform_sphere_mass_model(R: float, rho: float) -> float:
    """`rho` uniform volumetric mass density."""
    return 4/3*np.pi*R**3*rho

def uniform_disk_and_pointlike_mass_model(R: float, s: float, M0: float) -> float:
    return uniform_disk_mass_model(R, s) + pointlike_mass_model(M0)

def uniform_sphere_and_pointlike_mass_model(R: float, rho: float, M0: float) -> float:
    return uniform_sphere_mass_model(R, rho) + pointlike_mass_model(M0)

popt_point, pcov_point = curve_fit(lambda R, M0: vel(R, pointlike_mass_model(M0)),
                                   R, vrot, check_finite=False)
perr_point = np.sqrt(np.diag(pcov_point))
popt_disk, pcov_disk = curve_fit(lambda R, s: vel(R, uniform_disk_mass_model(R, s)),
                                 R, vrot, check_finite=False)
perr_disk = np.sqrt(np.diag(pcov_disk))
popt_sphere, pcov_sphere = curve_fit(lambda R, rho: vel(R, uniform_sphere_mass_model(R, rho)),
                                     R, vrot, check_finite=False)
perr_sphere = np.sqrt(np.diag(pcov_sphere))
popt_disk_point, pcov_disk_point = curve_fit(lambda R, s, M0: vel(R,
    uniform_disk_and_pointlike_mass_model(R, s, M0)),
                                             R, vrot, check_finite=False)
perr_disk_point = np.sqrt(np.diag(pcov_disk_point))
popt_sphere_point, pcov_sphere_points = curve_fit(lambda R, rho, M0: vel(R,
    uniform_sphere_and_pointlike_mass_model(R, rho, M0)),
                                                  R, vrot, check_finite=False)
perr_sphere_point = np.sqrt(np.diag(pcov_sphere_points))

print("M0 =", popt_point, "+-", perr_point, "RMS =",
      np.sqrt(np.mean((vrot-vel(R, pointlike_mass_model(*popt_point)))**2)))
print("s0 =", popt_disk, "+-", perr_disk, "RMS =",
      np.sqrt(np.mean((vrot-vel(R, uniform_disk_mass_model(R, *(popt_disk+perr_disk))))**2)))
print("rho0 =", popt_sphere, "+-", perr_sphere, "RMS =",
      np.sqrt(np.mean((vrot-vel(R, uniform_sphere_mass_model(R, *popt_sphere)))**2)))
print("s0 M0 =", popt_disk_point, "+-", perr_disk_point,
      "RMS =", np.sqrt(np.mean((vrot-vel(R, uniform_disk_and_pointlike_mass_model(R, *
    popt_disk_point)))**2)))
print("rho0 M0 =", popt_sphere_point, "+-", perr_sphere_point,
      "RMS =", np.sqrt(np.mean((vrot-vel(R, uniform_sphere_and_pointlike_mass_model(R, *
    popt_sphere_point)))**2)))

plt.rcParams.update({'font.size': 7})
fig, ax = plt.subplots(figsize=(3.25, 3.25))
ax.plot(R, -vrot,
        c='k', lw=0.25, marker='s', markersize=1, mfc="none", markeredgewidth=0.25, label="Datos")
ax.plot(R, -vel(R, pointlike_mass_model(*popt_point)),
        c="tab:red", lw=0.5, label="Punto")
ax.fill_between(R, -vel(R, pointlike_mass_model(*(popt_point+perr_point))),
                -vel(R, pointlike_mass_model(*(popt_point-perr_point))),
                facecolor="tab:red", alpha=0.25)
ax.plot(R, -vel(R, uniform_sphere_and_pointlike_mass_model(R, *popt_sphere_point)),
        c="tab:purple", lw=0.5, label="Esfera + Punto")
ax.fill_between(R, -vel(R, uniform_sphere_and_pointlike_mass_model(R, *(popt_sphere_point+
    perr_sphere_point))),
                -vel(R, uniform_sphere_and_pointlike_mass_model(R, *(popt_sphere_point-
    perr_sphere_point))),
```

```
                facecolor="tab:purple", alpha=0.25)
ax.plot(R, -vel(R, uniform_sphere_mass_model(R, *popt_sphere)),
        c="tab:orange", lw=0.5, label="Esfera")
ax.fill_between(R, -vel(R, uniform_sphere_mass_model(R, *(popt_sphere+perr_sphere))),
                -vel(R, uniform_sphere_mass_model(R, *(popt_sphere-perr_sphere))),
                facecolor="tab:orange", alpha=0.25)
ax.plot(R, -vel(R, uniform_disk_and_pointlike_mass_model(R, *popt_disk_point)),
        c="tab:blue", lw=0.5, label="Disco + Punto")
ax.fill_between(R, -vel(R, uniform_disk_and_pointlike_mass_model(R, *(popt_disk_point+
    perr_disk_point))),
                -vel(R, uniform_disk_and_pointlike_mass_model(R, *(popt_disk_point-perr_disk_point)
    )),
                facecolor="tab:blue", alpha=0.25)
ax.plot(R, -vel(R, uniform_disk_mass_model(R, *popt_disk)),
        c="tab:green", lw=0.5, label="Disco")
ax.fill_between(R, -vel(R, uniform_disk_mass_model(R, *(popt_disk+perr_disk))),
                -vel(R, uniform_disk_mass_model(R, *(popt_disk-perr_disk))),
                facecolor="tab:green", alpha=0.25)
ax.legend(loc="upper right")
ax.set_xlabel(r"$-R_{\odot}\sin\,l$ [kpc]")
ax.set_ylabel(r"$v_\mathrm{rot}$ [km/s]")
ax.yaxis.set_tick_params(rotation=90)
ax.tick_params(direction="in", top=True, right=True)
fig.savefig("../informe/rsc/massmodels.pdf")
plt.show()
```