Term Project (I) – Orbit of the Sun

Due Date: May 10, 2020, 40 points

Please submit both your solution and codes.

(a) (The Sun's orbit properties) Using the epicycle approximation, find the minimum and maximum distances to the Galactic center that the Sun attains in its orbit ($R_{\rm peri}$ and $R_{\rm apo}$). The Sun is located at $(X, Y, Z) = (8.2, 0, 0.025)$ kpc (Bland-Hawthorn & Gerhard 2016) with peculiar velocities with respect to LSR as $(U_\odot, V_\odot, W_\odot) = (11.1, 12.24, 7.25)$ km/s (Schonrich 2012). This means the Sun is moving towards the Galactic center and faster than LSR, and moving towards the North Galactic pole. The circular velocity at LSR is $V_{\rm LSR} = 240$ km/s (Reid et al. 2019 BeSSeL2 result). You may assume that the Oort's constants are $A = 15.06\,{\rm km\ s^{-1}\,kpc^{-1}}$, $B = -13.74\,{\rm km\ s^{-1}\,kpc^{-1}}$ (Irrgang et al. 2013). Hint: the guiding center of the Sun's orbit (R_g) is not the same as the Sun's current radius (R_0). The circular velocity at these two radii are not the same because the rotation curve is not completely flat around solar neighbourhood. The properties of the nearby rotation curve are described by the Oort's constants A and B. You could also ask yourself what is the rough phase of the Sun with its current radius and velocities in its epicyclic motion around the guiding center?

(b) (Solar orbit integration) Adopting the potential in Irrgang et al. (2013, A&A, 549, A137, Model I), try to use a simple time integration method to advance the motion of the Sun for 1 Gyr (read BT08, Page 196, Section 3.4 for more information on the orbital integration), and use appropriate units (for length, velocity, and time) to avoid extremely large or small numbers in your code. Make sure the orbital energy of the Sun is conserved better than $\sim 10^{-5}$. Plot the final orbits in the $X-Y$, $X-Z$, and $Y-Z$ plane. Estimate the angular frequency Ω, the radial frequency κ, and the vertical frequency ν, and the pericenter and apocenter radius from the orbit you integrated, then compare them to the values calculated by epicycle approximations in (a) and Table 1.2 in BT08. How different are the values?

(c) ($Z - V_Z$ phase space) Assuming that the vertical action J_Z of the Sun is conserved, derive the ratio of $|\Delta Z|$ (the vertical extension of the solar orbit) at $R_{\rm peri}$ and $R_{\rm apo}$ analytically. You may use the density profiles of the disk,

bulge, and halo of Irrgang et al. (2013, A&A, 549, A137, Model I) as in (b).

(d) ($Z-V_Z$ phase space) Plot the trajectory of the solar orbit in the $Z-V_Z$ phase space and $R-Z$ plane, where $R=\sqrt{X^2+Y^2}$. Find the ratio of $|\Delta Z|$ at $R_{\rm peri}$ and $R_{\rm apo}$, and compare with the analytical result in (c).

(e) [optional, 10 bonus points] Try to plot the orbit of the Sun using the potential of Irrgang et al. (2013, A&A, 549, A137, Model I, same as in b) while replace the bulge component with a rotating bar. You may assume the pattern speed of the bar is $40\,{\rm km\ s^{-1}\,kpc^{-1}}$, the mass of the bar is $1.5\times10^{10}M_\odot$, the semi-major axis length of the bar is 5 kpc and the axial ratio (major-to-minor) is 2.5 (Bland-Hawthorn & Gerhard 2016). You can also assume that the potential of the bar follows the Ferrers potential with $n=1$ (BT08, Page 95, Section 2.5.3). A C/C++ source code for calculating the Ferrers bar potential and forces with the parameters above is pasted below (major axis of the bar is along y-axis in this code). You are encouraged to adapt it into other programming languages. How is the conservation of Jacobi constant E_J of the orbit you get in this potential? How different is this orbit and the orbit in (b)?

```
#if SINGLE_PRECISION_ENABLED
using Real = float;
#else
using Real = double;
#endif

// provide the coordinates x,y,z of the point at a given time,
// return the force (fx,fy,fz) and potential (pot) at this point
// need to specify the major axis: aferrs, minor axis: bferrs
// the mass of the bar: barmass, and the bar pattern speed: phibar
// usage: ferrersbar(pot,fx,fy,fz,x,y,z,time);

static void ferrersbar(Real &potb, Real &fxb, Real &fyb, Real &fzb,
                       Real x, Real y, Real z, Real time)
{

  Real a2,b2,x2,y2,z2,lambda,dlambda;
  Real Fxrot,Fyrot,Fzrot,xrot,yrot,zrot;
  Real sinth, costh, tanth;
  Real w00, w01, w10, w11, w20, w02, w21, w12, w30, w03;
  Real angle;

  Real gconst = 4.302;   \\ gravity constant [kpc/(1.0e6Msun)*(km/s)^2]
```

```
Real aferrs = 5.;                       \\ kpc
Real bferrs = 2.;                       \\ kpc
Real barmass = 1.5e4;                   \\ 1.0e6 Msun
Real phibar = 40*1.02269032;            \\ km/s/kpc to Gyr^-1;
Real coefferr = -15/16*gconst*barmass; \\ n=1 case

Real rad = sqrt(x*x+y*y);
Real radius = sqrt(x*x+y*y+z*z);
Real dr = 1./radius;
Real ti = time*(unitT/Gyr);

angle = -1*ti*phibar;

xrot = x*cos(angle)-y*sin(angle);
yrot = x*sin(angle)+y*cos(angle);
zrot = z;

x2 = xrot * xrot;
y2 = yrot * yrot;
z2 = zrot * zrot;

a2  = aferrs*aferrs;
b2  = bferrs*bferrs;


if ( (y2/a2+x2/b2+z2/b2) > 1.0 ){

   lambda = (-(a2+b2-x2-y2-z2) + sqrt(SQR(a2+b2-x2-y2-z2)
          - 4.0*(a2*b2-y2*b2-x2*a2-z2*a2)))/2.0;
}

else {

   lambda = 0.0;

}


dlambda = (b2+lambda)*sqrt(a2+lambda);
costh = sqrt((b2+lambda)/(a2+lambda));
sinth = sqrt(1.0-SQR(costh));
tanth = sinth/costh;
```

```
  w00 = log((1.0+sinth)/costh)*2.0/sqrt(a2-b2);
  w10 = 2.0/sqrt((a2-b2)*SQR(a2-b2)) * (log((1.0+sinth)/costh)-sinth);
  w01 = SQR(tanth)*sinth/sqrt(SQR(a2-b2)*(a2-b2)) - w10/2.0;
  w11 = (w01 - w10)/(a2-b2);
  w20 = 2.0/3.0*(1.0/dlambda/(a2+lambda) - w11);
  w02 = 1.0/4.0*(2.0/dlambda/(b2+lambda) - w11);
  w21 = (w11 - w20)/(a2-b2);
  w12 = (w02 - w11)/(a2-b2);
  w30 = 2.0/5.0*(1.0/dlambda/SQR(a2+lambda) - w21);
  w03 = 1.0/6.0*(2.0/dlambda/SQR(b2+lambda) - w12);

  // n=1 case
  potb = coefferrs*(w00+x2*(x2*w02+2.0*y2*w11-2.0*w01)
                       +y2*(y2*w20+2.0*z2*w11-2.0*w10)
                       +z2*(z2*w02+2.0*x2*w02-2.0*w01));

  Fxrot = -4.* coefferrs * xrot * (x2*w02 + y2*w11 + z2*w02 - w01);
  Fyrot = -4.* coefferrs * yrot * (x2*w11 + y2*w20 - w10);
  Fzrot = -4.* coefferrs * zrot * (z2*w02 + y2*w11 + x2*w02 - w01);

  fxb = Fxrot*cos(-angle)-Fyrot*sin(-angle);
  fyb = Fxrot*sin(-angle)+Fyrot*cos(-angle);
  fzb = Fzrot;


  return;

}
```