Contents

```
import numpy as np
import os
from numpy import sqrt, cos, sin, pi, abs
import scipy as sp
import scipy.optimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
```

```
import sys
sys.path.append("/home/jtlaune/multi-planet-architecture/")
from helper import *
from plotting import *
import importlib
import os
import seaborn as sns
from scipy.ndimage import uniform_filter1d
j = 2
plt.rcParams.update({"font.size": 20,
     "figure.facecolor": "white",
     "figure.figsize": (6,6),
     #'font.family': 'serif',
     #"font.serif": ["DejaVu Serif"],
     #"text.usetex": False,
     "axes.linewidth": 3,
     'font.family': 'lmodern',
     'text.usetex': True,
     'text.latex.preamble': (            # LaTeX preamble
 r'\usepackage{lmodern}'
 # ... more packages if needed
     )
     })
# Create an array with the colors you want to use
#colors = ["#FF0B04", "#4374B3"]# Set your custom color palette
#sns.set_palette("dark")

pal = sns.color_palette("colorblind").as_hex()
blue = pal[0]
orange = pal[3]
palette = [blue, orange, *pal[1:2], *pal[4:]]
print(palette)
sns.set_palette(sns.color_palette(palette))

# important constants
deg_conv = 180./np.pi
```

1 standard picture h=0.1

1.1 preamble

```
class solve_eqeccs:
    def __init__(self, q, totmass, j, e1d, e2d, Tm1, Tm2, Te1, Te2, secterms=True):
self.q = q
self.mu2 = totmass/(1+q)
alpha_0 = (j/(j+1))**(2./3.)
self.alpha_0 = alpha_0
self.alpha_2 = 1./alpha_0
self.f1 = -A(alpha_0, j)
self.f2 = -B(alpha_0, j)
if secterms:
    self.C = C(alpha_0)
    self.D = D(alpha_0)
else:
    self.C = 0.
    self.D = 0.
self.e1d = e1d
self.e2d = e2d
self.Tm1 = Tm1
self.Tm2 = Tm2
self.Te1 = Te1
self.Te2 = Te2

    def dote1(self, e1, e2, theta1, theta2):
e1d = self.e1d
deriv = -self.mu2/self.alpha_2*(self.f1*sin(theta1) + self.D*e2*sin(theta1-theta2)) - (
return(deriv)
    def dote2(self, e1, e2, theta1, theta2):
e2d = self.e2d
deriv = -self.mu2/self.alpha_2*self.q*(self.f2*sin(theta2) + self.D*e1*sin(theta2-theta
return(deriv)
    def dotdpom(self, e1, e2, theta1, theta2):
#deriv = self.mu2/self.alpha_2*(self.f1*cos(theta1)/(e1*sqrt(self.alpha_0))
#         - self.q*cos(theta2)/e2
#         + 2*self.C/sqrt(self.alpha_0)
#         + self.D*e2/e1/sqrt(self.alpha_0)
#         -self.q*2*self.C/sqrt(self.alpha_2)
#         - self.q*self.D*e1/e2/sqrt(self.alpha_2))
deriv = (self.f1*e2*cos(theta1)/(sqrt(self.alpha_0))
 - self.q*self.f2*cos(theta2)*e1
 + 2*self.C/sqrt(self.alpha_0)*e1*e2
 + self.D*e2**2/sqrt(self.alpha_0)
 -self.q*2*self.C/sqrt(self.alpha_2)*e1*e2
 - self.q*self.D*e1**2/sqrt(self.alpha_2))
return(deriv)
    def doteta(self, e1, e2, theta1, theta2):
e1d = self.e1d
e2d = self.e2d
deriv = (self.q*sqrt(self.alpha_0)/(j*(self.q/self.alpha_0 + 1))
 *(1./self.Tm2 - 1./self.Tm1
   + 2*(e1)**2/self.Te1 - 2*(e2)**2/self.Te2)
 - self.q*sqrt(self.alpha_0)*2*(e1-e1d)*e1/self.Te1
 - 2*(e2-e2d)*e2/self.Te2)
return(deriv)
    def __call__(self, vec):
(e1, e2, theta1, theta2) = vec[:]
e1_deriv = self.dote1(e1, e2, theta1, theta2)
e2_deriv = self.dote2(e1, e2, theta1, theta2)
dpom_deriv = self.dotdpom(e1, e2, theta1, theta2)
eta_deriv =  self.doteta(e1, e2, theta1, theta2)
deriv_arr = np.array([e1_deriv,
      e2_deriv,
      dpom_deriv,
      eta_deriv])
#print(vec, deriv_arr)
return(deriv_arr)

def calc_eeqs(h, qs, totmass, j, e1d, e2d, Tm1s, Tm2s, Te1s, Te2s, secterms=True):
    Nqs = len(qs)
    e1s = np.zeros(Nqs)
    e2s = np.zeros(Nqs)
    theta1s = np.zeros(Nqs)
    theta2s = np.zeros(Nqs)

    for iq, q in enumerate(qs):
Te1 = 2*pi*Te1s[iq]
Te2 = 2*pi*Te2s[iq]
```

```
Tm1 = 2*pi*Tm1s[iq]
Tm2 = 2*pi*Tm2s[iq]
#if q <= 1:
#    Te2d = TW0*2*np.pi
#    Te1d = Te2d/q/alpha_0**0.5

#    Tm1 = Te1d/2.7/h**2
#    Tm2 = Te2d/2.7/h**2
#else:
#    Te1d = TW0*2*np.pi
#    Te2d = Te1d*q*alpha_0**0.5

#    Tm1 = Te2d/2.7/h**2
#    Tm2 = Te1d/2.7/h**2

if q > 1.:
    x0 = (0.01, 0.03, pi, 0.)
else:
    x0 = (0.05, 0.03, pi, 0.)
mfunc = solve_eqeccs(q, totmass, j, e1d, e2d, Tm1, Tm2, Te1, Te2, secterms=secterms)
sol = scipy.optimize.root(mfunc, x0, options={"diag":np.ones(4)*totmass, "maxfev":int(5
x = sol.x
e1 = x[0]
e2 = x[1]
theta1 = x[2]
theta2 = x[3]
if not sol.success:
    print(sol.message)
    e1 = -1
    e2 = -1
e1s[iq] = e1
e2s[iq] = e2
theta1s[iq] = theta1
theta2s[iq] = theta2
#print(x, mesg, infodict["nfev"])
#print(theta1-theta2)
    return(e1s, e2s, theta1s, theta2s)

os.chdir("/home/jtlaune/multi-planet-architecture/notes/")
```

1.2 get results

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
readme = "totmass q Te1 Te2 e1 e2 |g1-g2| \n " \
 "averages taken from 0.9xT; T = 20.xTe0; Te0=1000"
#################
# CONFIGURATION #
#################
j = 2
a0 = 1.0
alpha_0 = (j/(j+1))**(2./3.)
qs = np.array([0.5,0.75, 0.9, 1.1,1.5,2.])
Nqs = len(qs)
overwrite = True
totmass = 1e-3
e0 = 0.001

######################
# Varying parameters #
######################
Tw0 = 1000.
TeRatios = np.sqrt(qs)

####################
# THREADING ARRAYS #
####################
HS = np.ones(Nqs)*0.1
JS = np.ones(Nqs)*j
A0S = np.ones(Nqs)*a0
QS = np.array(qs)
MU2 = totmass/(1+QS)
MU1 = totmass - MU2
ALPHA_0 = alpha_0*np.ones(Nqs)
TE1 = Tw0*TeRatios
TE2 = Tw0/TeRatios
#TM1 = np.infty*np.ones(Nqs)
TM1 = TE1/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TM2 = TE2/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TS = 40.*np.maximum(TE1, TE2)
print(TS)
```

```
#E1_0 = np.minimum(0.1/sqrt(QS), 0.1*np.ones(Nqs))
#E2_0 = np.minimum(0.1*sqrt(QS), 0.1*np.ones(Nqs))
E1_0 = np.ones(Nqs)*e0
E2_0 = np.ones(Nqs)*e0
print(E1_0,E2_0)
E1DS = np.zeros(Nqs)
E2DS = np.zeros(Nqs)
CUTOFFS = TS
#ALPHA2_0 = (3/2.)**(2./3)*(1+E2_0**2+E1_0**2)
ALPHA2_0 = (1.7)**(2./3)*np.ones(Nqs)

NAMES = np.array([f"q{QS[i]:0.2f}" for i in range(Nqs)])

DIRNAMES = np.array([f"standard-h-{HS[i]:0.2f}-Tw0-{int(Tw0)}"
     for i in range(len(QS))])
DIRNAMES_NOSEC = np.array([DIRNAMES[i]+"-nosec" for i in range(Nqs)])

################
# WITH SECULAR #
################
results_arr = np.zeros((Nqs, 10))
results_arr[:,0] = totmass
results_arr[:,1] = QS
results_arr[:,2] = TE1
results_arr[:,3] = TE2
for i, name in enumerate(NAMES):
    data = np.load(os.path.join(DIRNAMES[i], name+".npz"))
    teval  = data["teval"]
    theta  = data["thetap"]
    a1     = data["a1"]
    a2     = data["a2"]
    e1     = data["e1"]
    e2     = data["e2"]
    g1     = data["g1"]
    g2     = data["g2"]
    L1     = data["L1"]
    L2     = data["L2"]
    x1     = data["x1"]
    y1     = data["y1"]
    x2     = data["x2"]
    y2     = data["y2"]

    it = int(len(teval)*0.9)

    results_arr[i,4] = np.average(e1[it:])
    results_arr[i,5] = np.average(e2[it:])
    results_arr[i,6] = np.average(np.abs(g1[it:]-g2[it:]))
    results_arr[i,7] = np.std(e1[it:])
    results_arr[i,8] = np.std(e2[it:])
    results_arr[i,9] = np.std(np.abs(g1[it:]-g2[it:]))
np.savetxt(f"standard-h-{h}-Tw0-{int(Tw0)}.txt", results_arr, header=readme)

###################
# WITHOUT SECULAR #
###################
results_arr = np.zeros((Nqs, 10))
results_arr[:,0] = totmass
results_arr[:,1] = QS
results_arr[:,2] = TE1
results_arr[:,3] = TE2
for i, name in enumerate(NAMES):
    data = np.load(os.path.join(DIRNAMES_NOSEC[i], name+".npz"))
    teval  = data["teval"]
    theta  = data["thetap"]
    a1     = data["a1"]
    a2     = data["a2"]
    e1     = data["e1"]
    e2     = data["e2"]
    g1     = data["g1"]
    g2     = data["g2"]
    L1     = data["L1"]
    L2     = data["L2"]
    x1     = data["x1"]
    y1     = data["y1"]
    x2     = data["x2"]
    y2     = data["y2"]

    it = int(len(teval)*0.9)

    results_arr[i,4] = np.average(e1[it:])
    results_arr[i,5] = np.average(e2[it:])
    results_arr[i,6] = np.average(np.abs(g1[it:]-g2[it:]))
    results_arr[i,7] = np.std(e1[it:])
    results_arr[i,8] = np.std(e2[it:])
    results_arr[i,9] = np.std(np.abs(g1[it:]-g2[it:]))

np.savetxt(f"standard-h-{h}-Tw0-{int(Tw0)}-nosec.txt", results_arr, header=readme)
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
print(f"standard-h-{h}-Tw0-{int(Tw0)}-nosec.txt")
```

1.3 eeq eccentricities

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
fig, ax = plt.subplots(figsize=(6,6))
fontsize=24
results = np.loadtxt(f"standard-h-{h}-Tw0-{int(Tw0)}.txt")
Te1 = results[:,2]
Te2 = results[:,3]
ratio = Te1/Te2
e1avg = results[:,4]
e2avg = results[:,5]
Dpomega = results[:,6]
e1avg_std = results[:,7]
e2avg_std = results[:,8]
Dpomega_std = results[:,9]
ax.errorbar(QS, e1avg, yerr=e1avg_std, fmt="o", c="k", label=r"$e_1$", capsize=2)
ax.errorbar(QS, e2avg, yerr=e2avg_std, fmt="o", c="r", label=r"$e_2$", capsize=2)
ax.tick_params(which="both", labelsize=fontsize, width=3, length=6,
       bottom=True, top=True, left=True, right=True,
       direction="in", pad=10)
#ax.set_title(r"$q = $"+f"{q:0.1f}", fontsize=fontsize)

# there is only negligible difference
#results = np.loadtxt(f"standard-h-{h}-Tw0-{int(Tw0)}-nosec.txt")
#Te1 = results[:,2]
#Te2 = results[:,3]
#ratio = Te1/Te2
#e1avg = results[:,4]
#e2avg = results[:,5]
#Dpomega = results[:,6]
```

```
#e1avg_std = results[:,7]
#e2avg_std = results[:,8]
#Dpomega_std = results[:,9]
#ax.errorbar(ratio, e1avg, yerr=e1avg_std, fmt="o", c="k", label=r"w/o sec.", capsize=2
#ax.errorbar(ratio, e2avg, yerr=e2avg_std, fmt="o", c="r", label=r"w/o sec.", capsize=2

ax.set_ylabel(r"$e$", fontsize=36)
ax.set_xlabel(r"$q$", fontsize=36)
ax.set_xscale("log")
#ax.set_xlim((0.45,2.05))
#ax.set_ylim((0.0, 0.045))

ylab = ax.get_yticklabels()
ylab[0].set_visible(False)

ax.legend(ncol=2, loc="best", fontsize=22)
print(-TM2)

qplot = np.linspace(0.5,2.0,100)
TeRatiosplot = np.sqrt(qplot)
Te1plot = Tw0*TeRatiosplot
Te2plot = Tw0/TeRatiosplot
Tm1plot = Te1plot/3.46/h**2*(1*(qplot<1) - 1*(qplot>=1))
Tm2plot = Te2plot/3.46/h**2*(1*(qplot<1) - 1*(qplot>=1))
e1s, e2s, theta1s, theta2s = calc_eeqs(h, qplot, totmass, j, 0., 0.,
       -Tm1plot, -Tm2plot, Te1plot, Te2plot, secterms=True)
ax.plot(qplot, e1s, ls="--", c="k"  , label=(r"$e_1$"))
ax.plot(qplot, e2s, ls="--", c="r", label=(r"$e_2$"))

#e1s, e2s, theta1s, theta2s = calc_eeqs(HS[0], QS, totmass, j, 0., 0., -TM1, -TM2, TE1
##print(e2s)
#ax.plot(QS, e1s, ls="--", c="k"  ,   label=(r"$e_1$, w/o sec"))
#ax.plot(QS, e2s, ls="--", c="r", label=(r"$e_2$, w/o sec"))

os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
print(f"standard-eeqs-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.png")
fig.savefig(f"standard-eeqs-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.png", bbox_inches="tight"
```

1.4 eq theta

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
fig, ax = plt.subplots(figsize=(6,6))
fontsize=24
results = np.loadtxt(f"standard-h-{h}-Tw0-{int(Tw0)}.txt")
Te1 = results[:,2]
Te2 = results[:,3]
ratio = Te1/Te2
e1avg = results[:,4]
e2avg = results[:,5]
Dpomega = results[:,6]*deg_conv
e1avg_std = results[:,7]
e2avg_std = results[:,8]
Dpomega_std = results[:,9]*deg_conv
ax.tick_params(which="both", labelsize=fontsize, width=3, length=6,
       bottom=True, top=True, left=True, right=True,
       direction="in", pad=10)
ax.errorbar(QS, Dpomega, c="k", yerr=Dpomega_std, fmt="o", capsize=2)
ax.tick_params(which="both", labelsize=fontsize)
#ax.set_ylim((0., 0.05))
#ax.set_title(r"$q = $"+f"{q:0.1f}", fontsize=fontsize)

#results = np.loadtxt(f"standard-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}-nosec.txt")
#Te1 = results[:,2]
#Te2 = results[:,3]
#ratio = Te1/Te2
#e1avg = results[:,4]
#e2avg = results[:,5]
#Dpomega = results[:,6]*deg_conv
#e1avg_std = results[:,7]
#e2avg_std = results[:,8]
#Dpomega_std = results[:,9]*deg_conv
#ax.errorbar(QS, Dpomega, marker="x", c="gray", label=r"w/o sec.", yerr=Dpomega_std, f

ax.set_ylabel(r"$\Delta\varpi$", fontsize=36)
ax.set_xlabel(r"$q$", fontsize=36)

ax.legend(ncol=2, loc="lower right")
```

```
qplot = np.linspace(0.5,2.0,100)
TeRatiosplot = np.sqrt(qplot)
Te1plot = Tw0*TeRatiosplot
Te2plot = Tw0/TeRatiosplot
Tm1plot = Te1plot/3.46/h**2*(1*(qplot<1) - 1*(qplot>=1))
Tm2plot = Te2plot/3.46/h**2*(1*(qplot<1) - 1*(qplot>=1))
e1s, e2s, theta1s, theta2s = calc_eeqs(h, qplot, totmass, j, 0., 0.,
       -Tm1plot, -Tm2plot, Te1plot, Te2plot, secterms=True)
ax.plot(qplot, np.abs(theta1s-theta2s)*deg_conv, ls="--", c="k")

#e1s, e2s, theta1s, theta2s = calc_eeqs(HS[0], QS, totmass, j, 0., 0., -TM1, -TM2, TE1
#ax.plot(QS, np.abs(theta1s-theta2s)*deg_conv, ls="--", c="gray")
ax.set_xscale("log")

fig.subplots_adjust(wspace=.75)
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
print(f"standard-pomega-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.png")
fig.savefig(f"standard-pomega-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.png", bbox_inches="tigh
```

1.5 example run

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
readme = "totmass q Te1 Te2 e1 e2 |g1-g2| \n " \
    "averages taken from 0.9xT; T = 20.xTe0; Te0=1000"
#################
# CONFIGURATION #
#################
j = 2
a0 = 1.0
alpha_0 = (j/(j+1))**(2./3.)
#Nqs = 25
#qs = np.logspace(-2, 2, Nqs)
#Nqs = 10
#qs = np.linspace(0.5,2,Nqs)
qs = np.array([0.5, 0.75, 0.9, 1.1,1.5, 2.])
Nqs = len(qs)
overwrite = True
totmass = 1e-3
e0 = 0.001
```

```
######################
# Varying parameters #
######################
Tw0 = 1000.
TeRatios = np.sqrt(qs)

####################
# THREADING ARRAYS #
####################
HS = np.ones(Nqs)*0.1
JS = np.ones(Nqs)*j
A0S = np.ones(Nqs)*a0
QS = np.array(qs)
MU2 = totmass/(1+QS)
MU1 = totmass - MU2
ALPHA_0 = alpha_0*np.ones(Nqs)
TE1 = Tw0*TeRatios
TE2 = Tw0/TeRatios
#TM1 = np.infty*np.ones(Nqs)
TM1 = TE1/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TM2 = TE2/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TS = 40.*np.maximum(TE1, TE2)
print(TS)
#E1_0 = np.minimum(0.1/sqrt(QS), 0.1*np.ones(Nqs))
#E2_0 = np.minimum(0.1*sqrt(QS), 0.1*np.ones(Nqs))
E1_0 = np.ones(Nqs)*e0
E2_0 = np.ones(Nqs)*e0
print(E1_0,E2_0)
E1DS = np.zeros(Nqs)
E2DS = np.zeros(Nqs)
CUTOFFS = TS
#ALPHA2_0 = (3/2.)**(2./3)*(1+E2_0**2+E1_0**2)
ALPHA2_0 = (1.7)**(2./3)*np.ones(Nqs)

NAMES = np.array([f"q{QS[i]:0.2f}" for i in range(Nqs)])

DIRNAMES = np.array([f"standard-h-{HS[i]:0.2f}-Tw0-{int(Tw0)}"
     for i in range(len(QS))])
DIRNAMES_NOSEC = np.array([DIRNAMES[i]+"-nosec" for i in range(Nqs)])
```

```
i = -1
name = NAMES[i]
print(name)
data = np.load(os.path.join(DIRNAMES[i], name+".npz"))
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

fontsize=24
fig, ax = plt.subplots(3,2, figsize=(18,12))
tscale = 1.

iplt0 = np.where(teval > 1e2)[0][0]
teval = teval[iplt0:]

iplt = np.where(teval > 1e4)[0][0]

for axi in ax.flatten():
    axi.tick_params(which="major", labelsize=fontsize, width=3, length=8,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.tick_params(which="minor", labelsize=fontsize, width=3, length=4,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.set_xlim((teval[:iplt][0]/tscale, teval[:iplt][-1]/tscale))
    axi.set_xlabel(r"$t$ [y]", fontsize=fontsize)
    axi.yaxis.set_major_locator(MaxNLocator(4))
    axi.set_xscale("log")
```

```
ax[0,0].scatter(teval[:iplt]/tscale, a1[:iplt], s=2, alpha=0.05, c="k")
ax[0,0].scatter(teval[:iplt]/tscale, a2[:iplt], s=2, alpha=0.05, c="r")
ax[0,0].set_ylabel(r"semimajor axis", fontsize=fontsize)

ax[0,1].scatter(teval[:iplt]/tscale, (a2[:iplt]/a1[:iplt])**1.5, s=2, alpha=0.05, c="k
ax[0,1].set_ylabel(r"$P_2/P_1$", fontsize=fontsize)

ax[2,0].scatter(teval[:iplt]/tscale,e2[:iplt], s=2, alpha=0.05, c="r", label=r"$e_2$")
ax[2,0].scatter(teval[:iplt]/tscale,e1[:iplt], s=2, alpha=0.05, c="k", label=r"$e_1$")
ax[2,0].set_ylabel(r"$e$", fontsize=fontsize)
ax[2,0].set_ylim(0, 0.075)
ax[2,0].legend()
C0 = mpl.lines.Line2D([], [], color='k', marker="o", linestyle='None',
      markersize=10, label=r'$e_1$')
C1 = mpl.lines.Line2D([], [], color='r', marker="o", linestyle='None',
      markersize=10, label=r'$e_2$')

ax[2,0].legend(handles=[C0, C1], loc="upper left", ncol=2)


theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
ax[1,0].scatter(teval[:iplt]/tscale, theta1[:iplt]*deg_conv, s=2, alpha=0.05, c="k")
ax[1,0].set_ylabel(r"$\theta_1$", fontsize=fontsize)
ax[1,0].set_ylim(0, 2*np.pi*deg_conv)

ax[1,1].scatter(teval[:iplt]/tscale, theta2[:iplt]*deg_conv, s=2, alpha=0.05, c="r")
ax[1,1].set_ylabel(r"$\theta_2$", fontsize=fontsize)
ax[1,1].set_ylim(0, 2*np.pi*deg_conv)

ax[2,1].scatter(teval[:iplt]/tscale,np.abs(g1[:iplt]-g2[:iplt])*deg_conv, s=2, alpha=0
ax[2,1].set_ylabel(r"$|\varpi_1-\varpi_2|$", fontsize=fontsize)
ax[2,1].set_ylim(120, 240)
ax[2,1].axhline(y=180., c="green", ls="--", lw=3, label="$180^\circ$")
ax[2,1].legend()

fig.subplots_adjust(hspace=0.4, wspace=0.2)


os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
```

```
fig.savefig(f"standard-example-h-{h}-Tw0-{int(Tw0)}.png", bbox_inches="tight")
print(f"standard-example-h-{h}-Tw0-{int(Tw0)}.png")
print(TM1[i],TM2[i],TE1[i],TE2[i],QS[i])
```

2 MMR Hamiltonian

```
fontsize = 30
def Hhat_Rtheta(R, theta, delta):
    return(-3*(delta+1)*R+R**2-2*np.sqrt(2*R)*np.cos(theta))
def Hhat_xinu(xi, nu, delta):
    return(-3*(delta+1)*(xi**2+nu**2)
   +(xi**2+nu**2)**2-2*np.sqrt(2)*xi)
fig, ax = plt.subplots(1,3,figsize=(19,6))
xpos = np.linspace(-5,-0.01,500)
xneg = np.linspace(0.01,5,500)
deltapos = -1+(1./3./xpos)*(-sqrt(2) + 2*xpos**3)
deltaneg = -1+(1./3./xneg)*(-sqrt(2) + 2*xneg**3)
ax[0].plot(xpos, deltapos, c="k")
ax[0].plot(xneg, deltaneg, c="k")

deltares = np.linspace(0,10,1000)
x1s = np.zeros(len(deltares))
x2s = np.zeros(len(deltares))
x3s = np.zeros(len(deltares))
x4s = np.zeros(len(deltares))
for i, d0 in enumerate(deltares):
    p = np.array([2,0,-3*(d0+1), -sqrt(2)])
    proots = np.roots(p)
    xi = np.min(proots)
    H0 = Hhat_xinu(xi, 0., d0)
    q = np.array([1,0,-3*(d0+1),-2*np.sqrt(2),-H0])
    qroots = np.sort(np.roots(q))
    x1s[i] = qroots[0]
    x2s[i] = qroots[1]
    x3s[i] = qroots[2]
    x4s[i] = qroots[3]

ax[0].plot(x3s, deltares, c="r")
ax[0].plot(x4s, deltares, c="r")
```

```
ax[0].fill_betweenx(deltares, x3s, x4s, color="r", alpha=0.1)
#print(x3s)

#idcross = np.argmin(x2s)
#ax.plot(x2s[idcross:], deltapos[idcross:], c="r")
#ax.plot(x4s, deltapos, c="r")

ax[0].set_xlim((-5,5))
ax[0].set_ylim((-5,5))
ax[0].set_xlabel(r"$\xi$", fontsize=32)
ax[0].set_ylabel(r"$\delta$", fontsize=32)


R = np.linspace(0, 8, 1000)
t = np.linspace(0, 2*np.pi, 1000)
RR, TT = np.meshgrid(R,t)

delta = -0.5
ax[0].axhline(y=delta, ls="--", c="green")
XX = RR*np.cos(TT)
YY = RR*np.sin(TT)
levels = np.linspace(-2, 20, 8)
p = np.array([2,0,-3*(delta+1), -sqrt(2)])
proots = np.roots(p)
xi = np.max(proots)
ax[0].scatter(xi,delta,marker="x",c="cyan",s=100)
H0 = Hhat_xinu(xi, 0., delta)
ax[1].contour(XX, YY, Hhat_Rtheta(RR,TT,delta), levels=levels, colors="k", linestyles="
ax[1].scatter(xi**2,0,marker="x",c="cyan",s=100)
ax[1].text(-7, -7, r"$\delta=$ "+f"{delta:0.1f}", fontsize=32 )

delta = 1
ax[0].axhline(y=delta, ls="--", c="green")
p = np.array([2,0,-3*(delta+1), -sqrt(2)])
proots = np.roots(p)
xi = np.min(proots)
ax[0].scatter(proots,delta*np.ones(len(proots)),marker="x",c="magenta",s=100)
H0 = Hhat_xinu(xi, 0., delta)
levels = H0*np.flip(np.array([0.5, 1.0, 1.5, 2.0, 2.5, 3.0]))
print(levels)
```

```
ax[2].scatter(proots**2*np.sign(proots), np.zeros(len(proots)), marker="x",c="magenta"
ax[2].scatter(proots**2*np.sign(proots), np.zeros(len(proots)), marker="x",c="magenta"
ax[2].contour(XX, YY, Hhat_Rtheta(RR,TT,delta), levels=levels, colors="k", linestyles="
ax[2].text(-7, -7, r"$\delta=$ "+f"{delta:0.1f}", fontsize=32 )

for axi in ax:
    axi.axhline(y=0., ls="--", c="k", lw=1)
    axi.axvline(x=0., ls="--", c="k", lw=1)
    axi.tick_params(which="major", labelsize=fontsize, width=3,
      length=8, bottom=True, top=True, left=True, right=True,
      direction="in", pad=10)
    axi.tick_params(which="minor", labelsize=fontsize, width=3,
      length=4, bottom=True, top=True, left=True, right=True,
      direction="in", pad=10)
fig.subplots_adjust(wspace=0.35)
ax[1].set_ylabel(r"$R\sin\theta$",fontsize=32)
ax[2].set_ylabel(r"$R\sin\theta$",fontsize=32)
ax[1].set_xlabel(r"$R\cos\theta$",fontsize=32)
ax[2].set_xlabel(r"$R\cos\theta$",fontsize=32)


os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
fig.savefig(f"phasediag.png", bbox_inches="tight")
```

3 vary Te

3.1 get results

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
readme = ""
qs = [0.5, 1., 2.]
h=0.1

for q in qs:
    #################
    # CONFIGURATION #
    #################
    j = 2
    a0 = 1.0
```

```
alpha_0 = (j/(j+1))**(2./3.)
overwrite = True
totmass = 1e-3

######################
# Varying parameters #
######################
Tw0 = 1000.
rats = np.array([0.25, 0.5, 0.75, 1.5, 2.5, 5, 10])
TeRatios = np.sqrt(rats)
Nqs = len(TeRatios)
qs = np.ones(Nqs)*q

####################
# THREADING ARRAYS #
####################
HS = np.ones(Nqs)*h
JS = np.ones(Nqs)*j
A0S = np.ones(Nqs)*a0
QS = np.array(qs)
MU2 = totmass/(1+QS)
MU1 = totmass - MU2
ALPHA_0 = alpha_0*np.ones(Nqs)
TE1 = Tw0*TeRatios
TE2 = Tw0/TeRatios
TM1 = TE1/3.46/HS**2*(1*(TeRatios<1) - 1*(TeRatios>=1))
TM2 = TE2/3.46/HS**2*(1*(TeRatios<1) - 1*(TeRatios>=1))
TS = 8.*np.maximum(TE1, TE2)
E1_0 = np.ones(Nqs)*0.1/sqrt(QS)
E2_0 = np.ones(Nqs)*0.1*sqrt(QS)
E1DS = np.zeros(Nqs)
E2DS = np.zeros(Nqs)
CUTOFFS = TS
ALPHA2_0 = (3/2.)**(2./3)*(1+E2_0**2+E1_0**2)
NAMES = np.array([f"ratio-{rats[i]}" for i in range(len(QS))])

DIRNAMES = np.array([f"./varyTe-q{QS[i]}-h-{h}-Tw0-{int(Tw0)}" for i in range(Nqs)
print(DIRNAMES)
DIRNAMES_NOSEC = np.array([DIRNAMES[i]+"-nosec" for i in range(Nqs)])
```

```
    ################
    # WITH SECULAR #
    ################
    results_arr = np.zeros((Nqs, 10))
    results_arr[:,0] = totmass
    results_arr[:,1] = QS
    results_arr[:,2] = TE1
    results_arr[:,3] = TE2
    for i, name in enumerate(NAMES):
data = np.load(os.path.join(DIRNAMES[i], name+".npz"))
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

it = int(len(teval)*0.9)

results_arr[i,4] = np.average(e1[it:])
results_arr[i,5] = np.average(e2[it:])
results_arr[i,6] = np.average(np.abs(g1[it:]-g2[it:]))
results_arr[i,7] = np.std(e1[it:])
results_arr[i,8] = np.std(e2[it:])
results_arr[i,9] = np.std(np.abs(g1[it:]-g2[it:]))
    np.savetxt(f"varyTe-q{q}-h-{h}-Tw0-{int(Tw0)}.txt", results_arr, header=readme)

    ###################
    # WITHOUT SECULAR #
    ###################
    results_arr = np.zeros((Nqs, 10))
    results_arr[:,0] = totmass
    results_arr[:,1] = QS
    results_arr[:,2] = TE1
    results_arr[:,3] = TE2
    for i, name in enumerate(NAMES):
data = np.load(os.path.join(DIRNAMES_NOSEC[i], name+".npz"))
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

it = int(len(teval)*0.9)

results_arr[i,4] = np.average(e1[it:])
results_arr[i,5] = np.average(e2[it:])
results_arr[i,6] = np.average(np.abs(g1[it:]-g2[it:]))
results_arr[i,7] = np.std(e1[it:])
results_arr[i,8] = np.std(e2[it:])
results_arr[i,9] = np.std(np.abs(g1[it:]-g2[it:]))

    np.savetxt(f"varyTe-q{q}-h-{h}-Tw0-{int(Tw0)}-nosec.txt", results_arr, header=read
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
```

3.2 eeq eccentricities

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
qs = [0.5, 1., 2.]
TM2 = -np.ones(Nqs)*5e4
fig, ax = plt.subplots(3, figsize=(8,14), sharex=True)
fontsize=32
for i,q in enumerate(qs):
    results = np.loadtxt(f"varyTe-q{q}-h-{h}-Tw0-{int(Tw0)}.txt")
    Te1 = results[:,2]
    Te2 = results[:,3]
    ratio = Te1/Te2
    e1avg = results[:,4]
    e2avg = results[:,5]
    Dpomega = results[:,6]
    e1avg_std = results[:,7]
    e2avg_std = results[:,8]
    Dpomega_std = results[:,9]
    ax[i].errorbar(ratio, e1avg, yerr=e1avg_std, c="k", label=f"$e_1$", fmt="o", capsiz
    ax[i].errorbar(ratio, e2avg, yerr=e2avg_std, c="r", label=f"$e_2$", fmt="o", capsiz
    ax[i].tick_params(which="both", labelsize=fontsize)
    #ax[i].set_ylim((0., 0.05))

############
# FIX THIS #
############

    Nplot = 100
    qplot = np.ones(Nplot)*q
    ratsplot = np.logspace(-1,1,Nplot)
    TeRatiosplot = np.sqrt(ratsplot)
    Te1plot = Tw0*TeRatiosplot
    Te2plot = Tw0/TeRatiosplot
    Tm1plot = Te1plot/3.46/h**2*(1*(TeRatiosplot<1) - 1*(TeRatiosplot>=1))
    Tm2plot = Te2plot/3.46/h**2*(1*(TeRatiosplot<1) - 1*(TeRatiosplot>=1))
    e1s, e2s, theta1s, theta2s = calc_eeqs(h, qplot, totmass, j, 0.,
   0., -Tm1plot, -Tm2plot, Te1plot, Te2plot, secterms=True)
    #print(ratio)
    #print(e1s)
    #print(e2s)
    ax[i].plot(ratsplot, e1s, ls="--", c="k")
    ax[i].plot(ratsplot, e2s, ls="--", c="r")


    ## there is no difference
    #results = np.loadtxt(f"varyTe-q{q}-h-{h}-Tw0-{int(Tw0)}-nosec.txt")
    #Te1 = results[:,2]
    #Te2 = results[:,3]
    #ratio = Te1/Te2
    #e1avg = results[:,4]
    #e2avg = results[:,5]
    #Dpomega = results[:,6]
    #ax[i].scatter(ratio, e1avg, marker="x", c="k", s=20)
    #ax[i].scatter(ratio, e2avg, marker="x", c="r", s=20)

    ax[i].tick_params(which="major", labelsize=fontsize, width=3, length=8,
   bottom=True, top=True, left=True, right=True,
   direction="in", pad=10)
    ax[i].tick_params(which="minor", labelsize=fontsize, width=3, length=4,
   bottom=True, top=True, left=True, right=True,
   direction="in", pad=10)
    ax[i].set_yticks([0.01, 0.03, 0.06, 0.09])
    ax[i].set_ylim((0.0,0.1))
    ax[i].set_xscale("log")

    ax[i].set_ylabel(r"$e$", fontsize=fontsize)
    ax[i].text(0.1, 0.1, r"$q=$ "+f"{q}", fontsize=fontsize, transform=ax[i].transAxes

ax[0].set_title("Eccentricity", fontsize=40, pad=20)
ax[0].legend(ncol=2)
ax[-1].set_xlabel(r"$T_{e,1}/T_{e,2}$", fontsize=fontsize)

fig.subplots_adjust(wspace=.75, hspace=0.)
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
fig.savefig(f"varyTe-eeqs-h-{h}-Tw0-{int(Tw0)}.png", bbox_inches="tight")
```

3.3 Dpomega

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
qs = [0.5, 1., 2.]
TM2 = -np.ones(Nqs)*5e4
fig, ax = plt.subplots(3, figsize=(8,14), sharex=True)
fontsize=32

palette = ["black", "red", "blue"]
sns.set_palette(sns.color_palette(palette))

for i,q in enumerate(qs):
    results = np.loadtxt(f"varyTe-q{q}-h-{h}-Tw0-{int(Tw0)}.txt")
    Te1 = results[:,2]
    Te2 = results[:,3]
    ratio = Te1/Te2
    e1avg = results[:,4]
    e2avg = results[:,5]
    Dpomega = results[:,6]
    e1avg_std = results[:,7]
    e2avg_std = results[:,8]
    Dpomega_std = results[:,9]*deg_conv
    ax[i].errorbar(ratio, Dpomega*deg_conv, yerr=Dpomega_std,
   label=r"$\Delta\varpi$", c=f"k", capsize=2,
   fmt="o",markersize=5,zorder=10)

    ax[i].tick_params(which="major", labelsize=fontsize, width=3, length=8,
   bottom=True, top=True, left=True, right=True,
   direction="in", pad=10)
    ax[i].tick_params(which="minor", labelsize=fontsize, width=3, length=4,
   bottom=True, top=True, left=True, right=True,
   direction="in", pad=10)
    ax[i].yaxis.set_major_locator(MaxNLocator(4))
    ax[i].set_ylabel(r"\Delta\varpi", fontsize=fontsize)
    ax[i].text(0.1, 0.1, r"$q=$ "+f"{q}", fontsize=fontsize, transform=ax[i].transAxes
    ax[i].set_xlabel(r"$T_{e,1}/T_{e,2}$", fontsize=fontsize)
    ax[i].tick_params(which="both", labelsize=fontsize)
    ax[i].set_xscale("log")

    ax[i].set_yticks([175, 180, 185 ])
    ax[i].set_ylim((170,190))

    Nplot = 100
    qplot = np.ones(Nplot)*q
    ratsplot = np.logspace(-1,1,Nplot)
    TeRatiosplot = np.sqrt(ratsplot)
    Te1plot = Tw0*TeRatiosplot
    Te2plot = Tw0/TeRatiosplot
    Tm1plot = Te1plot/3.46/h**2*(1*(TeRatiosplot<1) - 1*(TeRatiosplot>=1))
    Tm2plot = Te2plot/3.46/h**2*(1*(TeRatiosplot<1) - 1*(TeRatiosplot>=1))
    e1s, e2s, theta1s, theta2s = calc_eeqs(h, qplot, totmass, j, 0.,
   0., -Tm1plot, -Tm2plot, Te1plot, Te2plot, secterms=True)
```

```
    ax[i].plot(ratsplot, deg_conv*np.abs(theta1s-theta2s), ls="--", c="k")

fig.subplots_adjust(wspace=1.0)
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
fig.savefig(f"varyTe-pomega-h-{h}-Tw0-{int(Tw0)}.png", bbox_inches="tight")
```

3.4 NOT INCLUDED epsilon est

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
qs = [0.5, 1., 2.]
TM2 = -np.ones(Nqs)*5e4
fig, ax = plt.subplots(figsize=(8,12))
fontsize=32
for i,q in enumerate(qs):
    results = np.loadtxt(f"q{q}-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.txt")
    Te1 = results[:,2]
    Te2 = results[:,3]
    ratio = Te1/Te2
    e1avg = results[:,4]
    e2avg = results[:,5]
    Dpomega = results[:,6]
    ax.plot(ratio, e1avg/e2avg, c=f"C{i}", label=f"$q={q}$", ls="-", lw=4)
    #ax.plot(ratio, e2avg, c=f"C{i}", label=r"$e_2$", ls="--")
    ax.tick_params(which="both", labelsize=fontsize)
    ax.set_ylim((0., 3))

    # there is no difference
    #results = np.loadtxt(f"q{q}-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}-nosec.txt")
    #Te1 = results[:,2]
    #Te2 = results[:,3]
    #ratio = Te1/Te2
    #e1avg = results[:,4]
    #e2avg = results[:,5]
    #Dpomega = results[:,6]
    #ax.scatter(ratio, e1avg, marker="x", c="k", label=r"w/o sec.")
    #ax.scatter(ratio, e2avg, marker="x", c="r", label=r"w/o sec.")

    alpha_0 = (j/(j+1))**(2./3.)
    gamma = q*sqrt(alpha_0)
    delta = j*(q/alpha_0 + 1)
    ####################################################################
    # I don't think this formula is correct. Check for a bug           #
    # or algebra error. Still get close though, increasing factor of q #
    ####################################################################
    eps2 = np.abs(ratio/(delta-1)*(0.5*Te2/e2avg**2/TM2[0] + delta/gamma - 1))
    eps = sqrt(eps2)

    ax.plot(ratio, eps, c=f"C{i}", ls="--", lw=4)

ax.set_ylabel(r"$\epsilon$", fontsize=fontsize)
ax.set_xlabel(r"$T_{e,1}/T_{e,2}$", fontsize=fontsize)

ax.legend(ncol=2, loc=4)
fig.subplots_adjust(wspace=.75)
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
fig.savefig(f"epsilon-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.png", bbox_inches="tight")
```

4 analytic vary Te

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
for q in [0.5, 1., 2.]:
    Tm2 = -5e4
    Tw0 = 1e3
    results = np.loadtxt(f"q{q}-Tm2-{-int(Tm2)}-Tw0-{int(Tw0)}.txt")
    Te1 = results[:,2]
    Te2 = results[:,3]
    ratio = Te1/Te2
    e1avg = results[:,4]
    e2avg = results[:,5]
    Dpomega = results[:,6]

    alpha_0 = (j/(j+1))**(2./3.)
    gamma = q*sqrt(alpha_0)
    delta = j*(q/alpha_0 + 1)
    eps2 = np.abs(ratio/(delta-1)*(0.5*Te2/e2avg**2/Tm2 + delta/gamma - 1))
    eps = sqrt(eps2) # fudge factor by eye at front

    #plt.plot(ratio, eps)
    plt.scatter(ratio, eps/(e1avg/e2avg))
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")

j = 2
N = 1000
Tw0 = 1000
Tm2 = 5e-4
TeRatios = np.linspace(0.1,10,N)
q = 0.5
alpha0 = (j/(j+1))**(2./3)
sqrtalpha0 = sqrt(alpha0)
sigma = (q*sqrtalpha0/j/(q*sqrtalpha0+1))*(TeRatios)
eps2 = np.abs((1+1/sigma)/(1-q*sqrtalpha0/sigma))
eps = sqrt(eps2)
e0 = 0.05
e1 = e0*sqrt(eps)
e2 = e0/sqrt(eps)
plt.plot(TeRatios, e1)
plt.plot(TeRatios, e2)
plt.twinx().plot(TeRatios, eps,c="r")
```

5 vary q

5.1 get results

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
ratios = [0.1, 0.5, 1.0, 2., 10.]

for ratio in ratios:
    #################
    # CONFIGURATION #
    #################
    j = 2
    a0 = 1.0
    alpha_0 = (j/(j+1))**(2./3.)
    Nqs = 5
    qs = np.linspace(0.5, 2., Nqs)
    overwrite = False
    totmass = 1e-3
```

```
######################
# Varying parameters #
######################
Tw0 = 1000.
TeRatios = np.sqrt(np.linspace(0.1, 10, Nqs))

####################
# THREADING ARRAYS #
####################
HS = np.zeros(Nqs)
JS = np.ones(Nqs)*j
A0S = np.ones(Nqs)*a0
QS = np.array(qs)
MU2 = totmass/(1+QS)
MU1 = totmass - MU2
TM1 = np.ones(Nqs)*np.infty
TM2 = -np.ones(Nqs)*5e4
ALPHA_0 = alpha_0*np.ones(Nqs)
TE1 = Tw0*ratio*np.ones(Nqs)
TE2 = Tw0/ratio*np.ones(Nqs)
TS = 5.*np.maximum(TE1, TE2)
E1_0 = np.ones(Nqs)*0.1/sqrt(QS)
E2_0 = np.ones(Nqs)*0.1*sqrt(QS)
E1DS = np.zeros(Nqs)
E2DS = np.zeros(Nqs)
CUTOFFS = TS
ALPHA2_0 = (3/2.)**(2./3)*(1+E2_0**2+E1_0**2)

NAMES = np.array([f"q{QS[i]:0.2f}" for i in range(Nqs)])

DIRNAMES = np.array([f"Teratio-{ratio:0.1f}-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}" for
#print(NAMES)
#print(DIRNAMES)
DIRNAMES_NOSEC = np.array([DIRNAMES[i]+"-nosec" for i in range(Nqs)])

################
# WITH SECULAR #
################
results_arr = np.zeros((Nqs, 7))
results_arr[:,0] = totmass
```

```
    results_arr[:,1] = QS
    results_arr[:,2] = TE1
    results_arr[:,3] = TE2
    for i, name in enumerate(NAMES):
data = np.load(os.path.join(DIRNAMES[i], name+".npz"))
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

it = int(len(teval)*0.9)

results_arr[i,4] = np.average(e1[it:])
results_arr[i,5] = np.average(e2[it:])
results_arr[i,6] = np.average(np.abs(g1-g2))
    np.savetxt(f"Teratio-{ratio}-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}-results.txt", result

    ###################
    # WITHOUT SECULAR #
    ###################
    results_arr = np.zeros((Nqs, 7))
    results_arr[:,0] = totmass
    results_arr[:,1] = QS
    results_arr[:,2] = TE1
    results_arr[:,3] = TE2
    for i, name in enumerate(NAMES):
data = np.load(os.path.join(DIRNAMES[i], name+".npz"))
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
```

```
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

it = int(len(teval)*0.9)

results_arr[i,4] = np.average(e1[it:])
results_arr[i,5] = np.average(e2[it:])
results_arr[i,6] = np.average(np.abs(g1-g2))

    np.savetxt(f"Teratio-{ratio:0.1f}-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}-results-nosec.t
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
```

5.2 eq ecc

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
readme = "totmass q Te1 Te2 e1 e2 |g1-g2| \n " \
 "averages taken from 0.9xT; T = 20.xTe0; Te0=1000"
TM2 = -np.ones(Nqs)*5e4
fig, ax = plt.subplots(1,5,figsize=(30,5))
ratios = [0.1, 0.5, 1.0, 2., 10.]
fontsize=32
for i,ratio in enumerate(ratios):
    print(ratio)
    results = np.loadtxt(f"Teratio-{ratio:0.1f}-Tm2-{-int(TM2[0])}-Tw0-{int(Tw0)}-resul
    Te1 = results[:,2]
    Te2 = results[:,3]
    e1avg = results[:,4]
    e2avg = results[:,5]
    Dpomega = results[:,6]
    qs = np.linspace(0.5, 2., Nqs)
    ax[i].scatter(qs, e1avg, c="C0", label=r"$e_1$")
    ax[i].scatter(qs, e2avg, c="C1", label=r"$e_2$")
    ax[i].tick_params(which="both", labelsize=fontsize)
    ax[i].set_ylim((0., 0.07))
    ax[i].set_title(r"$T_{e,1}/T_{e,2} = $"+f"{ratio:0.1f}", fontsize=fontsize)

    # there is no difference
    #results = np.loadtxt(f"Teratio-{ratio:0.1f}-Tm2-{-int(TM2[0])}-Tw0-{int(Tw0)}-resu
    #Te1 = results[:,2]
    #Te2 = results[:,3]
    #e1avg = results[:,4]
    #e2avg = results[:,5]
    #Dpomega = results[:,6]
    #qs = np.linspace(0.5, 2., Nqs)
    #ax[i].scatter(qs, e1avg, marker="x", c="k", label=r"w/o sec.")
    #ax[i].scatter(qs, e2avg, marker="x", c="r", label=r"w/o sec.")

    ax[i].set_ylabel(r"$e$", fontsize=fontsize)
    ax[i].set_xlabel(r"$q$", fontsize=fontsize)

ax[0].legend(ncol=2, loc="best")
fig.subplots_adjust(wspace=.75)
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
fig.savefig(f"Teratio-eeqs-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.png", bbox_inches="tight"
```

5.3 eq Dpomega

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/standard-compmass/")
readme = "totmass q Te1 Te2 e1 e2 |g1-g2| \n " \
 "averages taken from 0.9xT; T = 20.xTe0; Te0=1000"
TM2 = -np.ones(Nqs)*5e4
fig, ax = plt.subplots(figsize=(5,5))
ratios = [0.1, 0.5, 1.0, 2., 10.]
fontsize=24
for i,ratio in enumerate(ratios):
    results = np.loadtxt(f"Teratio-{ratio:0.1f}-Tm2-{-int(TM2[0])}-Tw0-{int(Tw0)}-resul
    Te1 = results[:,2]
    Te2 = results[:,3]
    e1avg = results[:,4]
    e2avg = results[:,5]
    Dpomega = results[:,6]
    qs = np.linspace(0.5, 2., Nqs)
    ax.plot(qs, Dpomega, c=f"C{i}", label=f"{ratio}")
    ax.tick_params(which="both", labelsize=fontsize)
    ax.set_title(r"$T_{e,1}/T_{e,2} $", fontsize=fontsize,pad=110)

    # there is no difference
    #results = np.loadtxt(f"Teratio-{ratio:0.1f}-Tm2-{-int(TM2[0])}-Tw0-{int(Tw0)}-resu
    #Te1 = results[:,2]
    #Te2 = results[:,3]
    #e1avg = results[:,4]
    #e2avg = results[:,5]
    #Dpomega = results[:,6]
    #qs = np.linspace(0.5, 2., Nqs)
    #ax[i].scatter(qs, Dpomega, marker="x", c="C1", label=r"w/o sec.")

    ax.set_ylabel(r"$\Delta\varpi$", fontsize=fontsize)
    ax.set_xlabel(r"$q$", fontsize=fontsize)

ax.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower left',
 ncol=2, mode="expand", borderaxespad=0.)
fig.subplots_adjust(wspace=.75)
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
fig.savefig(f"Teratio-pomega-Tm2-{-int(TM2[i])}-Tw0-{int(Tw0)}.png", bbox_inches="tight
```

6 ecc driving force

6.1 example

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/compmass-Rhat/")
#################
# CONFIGURATION #
#################
j = 2
a0 = 1.0
h = 0.1
alpha_0 = (j/(j+1))**(2./3.)
Nqs = 1
qs = np.ones(Nqs)*2
overwrite = True
totmass = 1e-3
```

```
Tw0 = 1000
TeRatios = sqrt(qs)

######################
# Varying parameters #
######################
E1_0 = np.ones(Nqs)*0.001
E2_0 = np.ones(Nqs)*0.001
E1DS = np.ones(Nqs)*0.3
E2DS = np.ones(Nqs)*0.0


#eccs = np.array([0.1])
#E1_0, E2_0 = np.meshgrid(eccs, eccs)
#E1_0 = np.flip(E1_0.flatten())
#E2_0 = np.flip(E2_0.flatten())

####################
# THREADING ARRAYS #
####################
HS = np.ones(Nqs)*h
JS = np.ones(Nqs)*j
A0S = np.ones(Nqs)*a0
QS = qs
MU2 = totmass/(1+QS)
MU1 = totmass - MU2
TE1 = Tw0*TeRatios
TE2 = Tw0/TeRatios
TM1 = TE1/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TM2 = TE2/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TS = 15.*np.maximum(TE1, TE2)
ALPHA_0 = alpha_0*np.ones(Nqs)
##############################################################
# BUG: SETTING CUTOFF TO T RESULTS IN DIFFERENCES BETWEEN T #
# VALUES. LIKELY A FACTOR OF 2PI THING.                      #
##############################################################
cutoff_frac = 1.0
CUTOFFS = TS*cutoff_frac
ALPHA2_0 = (3/2.)**(2./3)*1.
NAMES = np.array([f"e1d-{E1DS[i]:0.3f}-e2d-{E2DS[i]:0.3f}"
  for i, qit in enumerate(QS)])

DIRNAMES = np.array([f"./driveTe-h-{h:0.2f}-Tw0-{Tw0}" for i
in range(Nqs)])
DIRNAMES_NOSEC = np.array([DIRNAMES[i]+"_NOSEC" for i in range(Nqs)])

i = 0
dirname = DIRNAMES[i]
name = NAMES[i]+".npz"
print(name)
data = np.load(os.path.join(dirname, name))
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

alpha = a1/a2
theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
f1 = A(alpha, j)
f2 = B(alpha, j)
barg1 = np.arctan2(e2*np.sin(g2), e2*np.cos(g2) + f2*e1/f1)
barg2 = np.arctan2(e1*np.sin(g1), e1*np.cos(g1) + f1*e2/f2)
hattheta1 = np.arctan2(e1*sin(theta1) + f2/f1*e2*sin(theta2),
      e1*cos(theta1) + f2/f1*e2*cos(theta2))
hattheta1 = hattheta1+2*pi*(hattheta1<0.)

fontsize=24
fig, ax = plt.subplots(4,2, figsize=(18,16))
tscale = 1.
```

```
iplt0 = np.where(teval > 9e2)[0][0]
teval = teval[iplt0:]

#iplt = np.where(teval > 5e3)[0][0]
iplt = len(teval)

ax[0,0].scatter(teval[:iplt]/tscale, a1[:iplt], s=2, alpha=0.05, c="k")
ax[0,0].scatter(teval[:iplt]/tscale, a2[:iplt], s=2, alpha=0.05, c="r")
ax[0,0].set_ylabel(r"semimajor axis", fontsize=fontsize)
ax[0,0].set_ylim((0.3, 1.5))
C0 = mpl.lines.Line2D([], [], color='k', marker="o", linestyle='None',
      markersize=10, label=r'$a_1$')
C1 = mpl.lines.Line2D([], [], color='r', marker="o", linestyle='None',
      markersize=10, label=r'$a_2$')
ax[0,0].legend(handles=[C0, C1], loc="upper right", ncol=2)

ax[0,1].scatter(teval[:iplt]/tscale, (a2[:iplt]/a1[:iplt])**1.5, s=2, alpha=0.05, c="k
ax[0,1].set_ylabel(r"$P_2/P_1$", fontsize=fontsize)
ax[0,1].patch.set_visible(False)
#ax[0,1].set_ylim((1.5,1.75))

ax[1,0].scatter(teval[:iplt]/tscale, theta1[:iplt]*deg_conv, s=2, alpha=0.05, c="k")
ax[1,0].set_ylabel(r"$\theta_1$", fontsize=fontsize)
ax[1,0].set_ylim(0, 2*np.pi*deg_conv)

ax[1,1].scatter(teval[:iplt]/tscale, theta2[:iplt]*deg_conv, s=2, alpha=0.05, c="k")
ax[1,1].set_ylabel(r"$\theta_2$", fontsize=fontsize)
ax[1,1].set_ylim(0, 2*np.pi*deg_conv)

ax[2,0].scatter(teval[:iplt]/tscale,e1[:iplt], s=2, alpha=0.05, c="k", label=r"$e_1$")
ax[2,0].set_ylabel(r"$e_1$", fontsize=fontsize)
ax[2,0].set_ylim(0, 0.4)

ax[2,1].scatter(teval[:iplt]/tscale, e2[:iplt], s=2, alpha=0.05, c="k", label=r"$e_1$"
ax[2,1].set_ylabel(r"$e_2$", fontsize=fontsize)
ax[2,1].set_ylim(0, 0.4)

ax[3,0].scatter(teval[:iplt]/tscale, deg_conv*hattheta1[:iplt], s=2, alpha=0.05,c="k")
ax[3,0].set_ylim((0., 360.))
```

```
ax[3,0].set_ylabel(r"$\hat\theta$", fontsize=fontsize)


Dpom = (g2[:iplt]-g1[:iplt])%(2*pi)
Dpom = Dpom - 2*pi*(Dpom>pi)
ax[3,1].scatter(teval[:iplt]/tscale, Dpom*deg_conv, s=2, alpha=0.05, c="k")
ax[3,1].set_ylabel(r"$\varpi_1-\varpi_2$", fontsize=fontsize)
ax[3,1].set_ylim((-180., 180.))
ax[3,1].axhline(y=0., c="r", ls="--", lw=3, label="$180^\circ$")
#ax[3,1].legend()

fig.subplots_adjust(hspace=0.4, wspace=0.2)

for axi in ax.flatten():
    axi.tick_params(which="major", labelsize=fontsize, width=3, length=8,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.tick_params(which="minor", labelsize=fontsize, width=3, length=4,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.set_xlim((teval[:iplt][0]/tscale, teval[:iplt][-1]/tscale))
    axi.set_xlabel(r"$t$ [y]", fontsize=fontsize)
    axi.yaxis.set_major_locator(MaxNLocator(4))
    axi.set_xscale("log")


os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
print(f"driving-example-h-{h}-Tw0-{int(Tw0)}.png")
fig.savefig(f"driving-example-h-{h}-Tw0-{int(Tw0)}.png", bbox_inches="tight")
```

7 apsidal alignment explanation

```
os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/runs/compmass-Rhat/")

########################
# eccentricity driving #
########################
q = 2
totmass = 1e-3
```

```
Tw0 = 1000
TeRatio = sqrt(q)
Te1 = Tw0*TeRatio
Te2 = Tw0/TeRatio
filename = "./driveTe-h-0.10-Tw0-1000/e1d-0.300-e2d-0.000.npz"
data = np.load(filename)
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

alpha = a1/a2
theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
f1 = A(alpha, j)
f2 = B(alpha, j)
barg1 = np.arctan2(e2*np.sin(g2), e2*np.cos(g2) + f2*e1/f1)
barg2 = np.arctan2(e1*np.sin(g1), e1*np.cos(g1) + f1*e2/f2)
hattheta1 = np.arctan2(e1*sin(theta1) + f2/f1*e2*sin(theta2),
      e1*cos(theta1) + f2/f1*e2*cos(theta2))
hattheta1 = hattheta1+2*pi*(hattheta1<0.)

edrive_analytic_expr = (e1**2*Te2 + e2**2*Te1)/(e1*e2*(Te1+Te2))
edrive_numerical_expr = cos(g2-g1)

############
# large e0 #
############
totmass = 1e-3
Tw0 = 10000
```

```
q = 2
TeRatio = sqrt(q)
Te1 = Tw0*TeRatio
Te2 = Tw0/TeRatio

filename = "./e0large-h-0.10-Tw0-10000-cut-0.30/e10-0.300-e20-0.300.npz"
data = np.load(filename)
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

alpha = a1/a2
theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
f1 = A(alpha, j)
f2 = B(alpha, j)
barg1 = np.arctan2(e2*np.sin(g2), e2*np.cos(g2) + f2*e1/f1)
barg2 = np.arctan2(e1*np.sin(g1), e1*np.cos(g1) + f1*e2/f2)
hattheta1 = np.arctan2(e1*sin(theta1) + f2/f1*e2*sin(theta2),
      e1*cos(theta1) + f2/f1*e2*cos(theta2))
hattheta1 = hattheta1+2*pi*(hattheta1<0.)

e0large_analytic_expr = (e1**2*Te2 + e2**2*Te1)/(e1*e2*(Te1+Te2))
e0large_numerical_expr = cos(g2-g1)

fontsize=24
fig, ax = plt.subplots(1,2, figsize=(12,6))
tscale = 1.
```

```
iplt0 = np.where(teval > 9e2)[0][0]
teval = teval[iplt0:]

#iplt = np.where(teval > 5e3)[0][0]
iplt = len(teval)

ax[0].scatter(teval[:iplt]/tscale,  edrive_analytic_expr[:iplt], s=2, alpha=0.05, c="r"
ax[0].scatter(teval[:iplt]/tscale, edrive_numerical_expr[:iplt], s=2, alpha=0.05, c="k"
ax[0].set_ylim((-1,10))

ax[1].scatter(teval[:iplt]/tscale,  e0large_analytic_expr[:iplt], s=2, alpha=0.05, c="
ax[1].scatter(teval[:iplt]/tscale, e0large_numerical_expr[:iplt], s=2, alpha=0.05, c="
ax[1].set_ylim((-1,10))
```

8 large initial ecc

8.1 example

```
os.chdir("/home/jtlaune/multi-planet-architecture/runs/compmass-Rhat/")
#################
# CONFIGURATION #
#################
j = 2
a0 = 1.0
h = 0.1
alpha_0 = (j/(j+1))**(2./3.)
Nqs = 1
qs = np.ones(Nqs)*2
overwrite = True
totmass = 1e-3
#e2d = 0.0
#e1d = 0.2
Tw0 = 10000
TeRatios = sqrt(qs)

######################
# Varying parameters #
######################
```

```
E1_0 = np.ones(Nqs)*0.3
E2_0 = np.ones(Nqs)*0.3
E1DS = np.ones(Nqs)*0.0
E2DS = np.ones(Nqs)*0.0


#eccs = np.array([0.1])
#E1_0, E2_0 = np.meshgrid(eccs, eccs)
#E1_0 = np.flip(E1_0.flatten())
#E2_0 = np.flip(E2_0.flatten())

####################
# THREADING ARRAYS #
####################
HS = np.ones(Nqs)*h
JS = np.ones(Nqs)*j
A0S = np.ones(Nqs)*a0
QS = qs
MU2 = totmass/(1+QS)
MU1 = totmass - MU2
TE1 = Tw0*TeRatios
TE2 = Tw0/TeRatios
TM1 = TE1/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TM2 = TE2/3.46/HS**2*(1*(qs<1) - 1*(qs>=1))
TS = 3.*np.maximum(TE1, TE2)
ALPHA_0 = alpha_0*np.ones(Nqs)
#############################################################
# BUG: SETTING CUTOFF TO T RESULTS IN DIFFERENCES BETWEEN T #
# VALUES. LIKELY A FACTOR OF 2PI THING.                     #
#############################################################
cutoff_frac = 0.3
CUTOFFS = TS*cutoff_frac
ALPHA2_0 = (3/2.)**(2./3)*1.
NAMES = np.array([f"e10-{E1_0[i]:0.3f}-e20-{E2_0[i]:0.3f}"
  for i, qit in enumerate(QS)])

DIRNAMES = np.array([f"./e0large-h-{h:0.2f}-Tw0-{Tw0}-cut-{cutoff_frac:0.2f}" for i
in range(Nqs)])
DIRNAMES_NOSEC = np.array([DIRNAMES[i]+"_NOSEC" for i in range(Nqs)])
```

```
i = -1
name = NAMES[i]
print(name)
data = np.load(os.path.join(DIRNAMES[i], name+".npz"))
teval  = data["teval"]
theta  = data["thetap"]
a1     = data["a1"]
a2     = data["a2"]
e1     = data["e1"]
e2     = data["e2"]
g1     = data["g1"]
g2     = data["g2"]
L1     = data["L1"]
L2     = data["L2"]
x1     = data["x1"]
y1     = data["y1"]
x2     = data["x2"]
y2     = data["y2"]

alpha = a1/a2
theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
f1 = A(alpha, j)
f2 = B(alpha, j)
barg1 = np.arctan2(e2*np.sin(g2), e2*np.cos(g2) + f2*e1/f1)
barg2 = np.arctan2(e1*np.sin(g1), e1*np.cos(g1) + f1*e2/f2)
hattheta1 = np.arctan2(e1*sin(theta1) + f2/f1*e2*sin(theta2),
      e1*cos(theta1) + f2/f1*e2*cos(theta2))
hattheta1 = hattheta1+2*pi*(hattheta1<0.)

fontsize=24
fig, ax = plt.subplots(4,2, figsize=(18,16))
tscale = 1.

iplt0 = np.where(teval > 9e2)[0][0]
teval = teval[iplt0:]

#iplt = np.where(teval > 5e3)[0][0]
iplt = len(teval)
```

```
ax[0,0].scatter(teval[:iplt]/tscale, a1[:iplt], s=2, alpha=0.05, c="k")
ax[0,0].scatter(teval[:iplt]/tscale, a2[:iplt], s=2, alpha=0.05, c="r")
ax[0,0].set_ylabel(r"semimajor axis", fontsize=fontsize)
ax[0,0].set_ylim((0.8, 1.5))
C0 = mpl.lines.Line2D([], [], color='k', marker="o", linestyle='None',
      markersize=10, label=r'$a_1$')
C1 = mpl.lines.Line2D([], [], color='r', marker="o", linestyle='None',
      markersize=10, label=r'$a_2$')
ax[0,0].legend(handles=[C0, C1], loc="upper right", ncol=2)

ax[0,1].scatter(teval[:iplt]/tscale, (a2[:iplt]/a1[:iplt])**1.5, s=2, alpha=0.05, c="k
ax[0,1].set_ylabel(r"$P_2/P_1$", fontsize=fontsize)
#ax[0,1].set_zorder(axp.get_zorder()+1)
ax[0,1].patch.set_visible(False)
ax[0,1].set_ylim((1.45,1.6))

ax[1,0].scatter(teval[:iplt]/tscale, theta1[:iplt]*deg_conv, s=2, alpha=0.05, c="k")
ax[1,0].set_ylabel(r"$\theta_1$", fontsize=fontsize)
ax[1,0].set_ylim(0, 2*np.pi*deg_conv)

ax[1,1].scatter(teval[:iplt]/tscale, theta2[:iplt]*deg_conv, s=2, alpha=0.05, c="k")
ax[1,1].set_ylabel(r"$\theta_2$", fontsize=fontsize)
ax[1,1].set_ylim(0, 2*np.pi*deg_conv)

ax[2,0].scatter(teval[:iplt]/tscale,e1[:iplt], s=2, alpha=0.05, c="k", label=r"$e_1$")
ax[2,0].set_ylabel(r"$e$", fontsize=fontsize)
ax[2,0].set_ylim(0, 0.35)

ax[2,1].scatter(teval[:iplt]/tscale, e2[:iplt], s=2, alpha=0.05, c="k", label=r"$e_1$"
ax[2,1].set_ylabel(r"$e$", fontsize=fontsize)
ax[2,1].set_ylim(0, 0.35)

ax[3,0].scatter(teval[:iplt]/tscale, deg_conv*hattheta1[:iplt], s=2, alpha=0.05,c="k")
ax[3,0].set_ylim((0., 360.))
ax[3,0].set_ylabel(r"$\hat\theta$", fontsize=fontsize)


Dpom = (g2[:iplt]-g1[:iplt])%(2*pi)
Dpom = Dpom - 2*pi*(Dpom>pi)
ax[3,1].scatter(teval[:iplt]/tscale, Dpom*deg_conv, s=2, alpha=0.05, c="k")
```

```
ax[3,1].set_ylabel(r"$\varpi_1-\varpi_2$", fontsize=fontsize)
ax[3,1].set_ylim((-180., 180.))
ax[3,1].axhline(y=0., c="r", ls="--", lw=3, label="$180^\circ$")
#ax[3,1].legend()

fig.subplots_adjust(hspace=0.4, wspace=0.2)

for axi in ax.flatten():
    axi.tick_params(which="major", labelsize=fontsize, width=3, length=8,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.tick_params(which="minor", labelsize=fontsize, width=3, length=4,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.set_xlim((teval[:iplt][0]/tscale, teval[:iplt][-1]/tscale))
    axi.set_xlabel(r"$t$ [y]", fontsize=fontsize)
    axi.yaxis.set_major_locator(MaxNLocator(4))
    axi.set_xscale("log")

os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
print(f"./e0large-h-{h:0.2f}-Tw0-{Tw0}-cut-{cutoff_frac:0.2f}.png")
fig.savefig(f"./e0large-h-{h:0.2f}-Tw0-{Tw0}-cut-{cutoff_frac:0.2f}.png", bbox_inches="
```

9 phase space paths

```
fig, ax = plt.subplots(2,3,figsize=(18,12))
j = 1

filter_size = 1
plt_skip = 250

####################
# STANDARD EXAMPLE #
####################
fname = "/home/jtlaune/multi-planet-architecture/runs/standard-compmass/standard-h-0.1
data = np.load(fname)
teval  = data["teval"][::plt_skip]
theta  = uniform_filter1d(data["thetap"], filter_size)[::plt_skip]
a1     = uniform_filter1d(data["a1"], filter_size)[::plt_skip]
```

```
a2     = uniform_filter1d(data["a2"], filter_size)[::plt_skip]
e1     = uniform_filter1d(data["e1"], filter_size)[::plt_skip]
e2     = uniform_filter1d(data["e2"], filter_size)[::plt_skip]
g1     = uniform_filter1d(data["g1"], filter_size)[::plt_skip]
g2     = uniform_filter1d(data["g2"], filter_size)[::plt_skip]
L1     = uniform_filter1d(data["L1"], filter_size)[::plt_skip]
L2     = uniform_filter1d(data["L2"], filter_size)[::plt_skip]
x1     = uniform_filter1d(data["x1"], filter_size)[::plt_skip]
y1     = uniform_filter1d(data["y1"], filter_size)[::plt_skip]
x2     = uniform_filter1d(data["x2"], filter_size)[::plt_skip]
y2     = uniform_filter1d(data["y2"], filter_size)[::plt_skip]

alpha = a1/a2
theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
f1 = A(alpha, j)
f2 = B(alpha, j)
barg1 = np.arctan2(e2*np.sin(g2), e2*np.cos(g2) + f2*e1/f1)
barg2 = np.arctan2(e1*np.sin(g1), e1*np.cos(g1) + f1*e2/f2)
hattheta1 = np.arctan2(e1*sin(theta1) + f2/f1*e2*sin(theta2),
      e1*cos(theta1) + f2/f1*e2*cos(theta2))
hattheta1 = hattheta1+2*pi*(hattheta1<0.)
hattheta1 = pi - hattheta1
fontsize=24
ehat = np.sqrt(f1**2*e1**2 + f2**2*e2**2 - 2*np.abs(f1*f2)*e1*e2*cos(g1-g2))

ax[0,0].scatter(e2*cos(theta2), e2*sin(theta2), c=teval)
ax[1,0].scatter(ehat*cos(hattheta1), ehat*sin(hattheta1), c=teval)

########################
# ECCENTRICITY DRIVING #
########################
fname = "/home/jtlaune/multi-planet-architecture/runs/compmass-Rhat/driveTe-h-0.10-Tw0-
data = np.load(fname)
teval  = data["teval"][::plt_skip]
theta  = uniform_filter1d(data["thetap"], filter_size)[::plt_skip]
a1     = uniform_filter1d(data["a1"], filter_size)[::plt_skip]
a2     = uniform_filter1d(data["a2"], filter_size)[::plt_skip]
e1     = uniform_filter1d(data["e1"], filter_size)[::plt_skip]
e2     = uniform_filter1d(data["e2"], filter_size)[::plt_skip]
```

```
g1     = uniform_filter1d(data["g1"], filter_size)[::plt_skip]
g2     = uniform_filter1d(data["g2"], filter_size)[::plt_skip]
L1     = uniform_filter1d(data["L1"], filter_size)[::plt_skip]
L2     = uniform_filter1d(data["L2"], filter_size)[::plt_skip]
x1     = uniform_filter1d(data["x1"], filter_size)[::plt_skip]
y1     = uniform_filter1d(data["y1"], filter_size)[::plt_skip]
x2     = uniform_filter1d(data["x2"], filter_size)[::plt_skip]
y2     = uniform_filter1d(data["y2"], filter_size)[::plt_skip]

alpha = a1/a2
theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
f1 = A(alpha, j)
f2 = B(alpha, j)
barg1 = np.arctan2(e2*np.sin(g2), e2*np.cos(g2) + f2*e1/f1)
barg2 = np.arctan2(e1*np.sin(g1), e1*np.cos(g1) + f1*e2/f2)
hattheta1 = np.arctan2(e1*sin(theta1) + f2/f1*e2*sin(theta2),
      e1*cos(theta1) + f2/f1*e2*cos(theta2))
hattheta1 = hattheta1+2*pi*(hattheta1<0.)
hattheta1 = pi - hattheta1
fontsize=24
ehat = np.sqrt(f1**2*e1**2 + f2**2*e2**2 - 2*np.abs(f1*f2)*e1*e2*cos(g1-g2))

ax[0,1].scatter(e2*cos(theta2), e2*sin(theta2), c=teval)
ax[1,1].scatter(ehat*cos(hattheta1), ehat*sin(hattheta1), c=teval)

##############################
# LARGE INITIAL ECCENTRICITY #
##############################
fname = "/home/jtlaune/multi-planet-architecture/runs/compmass-Rhat/e0large-h-0.10-Tw0-
data = np.load(fname)
teval  = data["teval"][::plt_skip]
theta  = uniform_filter1d(data["thetap"], filter_size)[::plt_skip]
a1     = uniform_filter1d(data["a1"], filter_size)[::plt_skip]
a2     = uniform_filter1d(data["a2"], filter_size)[::plt_skip]
e1     = uniform_filter1d(data["e1"], filter_size)[::plt_skip]
e2     = uniform_filter1d(data["e2"], filter_size)[::plt_skip]
g1     = uniform_filter1d(data["g1"], filter_size)[::plt_skip]
g2     = uniform_filter1d(data["g2"], filter_size)[::plt_skip]
L1     = uniform_filter1d(data["L1"], filter_size)[::plt_skip]
```

```
L2     = uniform_filter1d(data["L2"], filter_size)[::plt_skip]
x1     = uniform_filter1d(data["x1"], filter_size)[::plt_skip]
y1     = uniform_filter1d(data["y1"], filter_size)[::plt_skip]
x2     = uniform_filter1d(data["x2"], filter_size)[::plt_skip]
y2     = uniform_filter1d(data["y2"], filter_size)[::plt_skip]

alpha = a1/a2
theta1 = (theta+g1)%(2*np.pi)
theta2 = (theta+g2)%(2*np.pi)
f1 = A(alpha, j)
f2 = B(alpha, j)
barg1 = np.arctan2(e2*np.sin(g2), e2*np.cos(g2) + f2*e1/f1)
barg2 = np.arctan2(e1*np.sin(g1), e1*np.cos(g1) + f1*e2/f2)
hattheta1 = np.arctan2(e1*sin(theta1) + f2/f1*e2*sin(theta2),
      e1*cos(theta1) + f2/f1*e2*cos(theta2))
hattheta1 = hattheta1+2*pi*(hattheta1<0.)
hattheta1 = pi - hattheta1
fontsize=24
ehat = np.sqrt(f1**2*e1**2 + f2**2*e2**2 - 2*np.abs(f1*f2)*e1*e2*cos(g1-g2))

ax[0,2].scatter(e2*cos(theta2), e2*sin(theta2), c=teval)
ax[1,2].scatter(ehat*cos(hattheta1), ehat*sin(hattheta1), c=teval)

for axi in ax.flatten():
    axi.tick_params(which="major", labelsize=fontsize, width=3, length=8,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.tick_params(which="minor", labelsize=fontsize, width=3, length=4,
    bottom=True, top=True, left=True, right=True,
    direction="in", pad=10)
    axi.axhline(y=0, ls="--", c="k")
    axi.axvline(x=0, ls="--", c="k")
    axi.set_xlim((-0.3,0.3))
ax[0,0].set_ylabel(r"$e_2\sin\theta_2$", fontsize=30)
ax[1,0].set_ylabel(r"$\hat e\sin\hat\theta$", fontsize=30)
for i in range(3):
    ax[0,i].set_xlabel(r"$e_2\cos\theta_2$", fontsize=30)
    ax[0,i].set_ylim((-0.2,0.2))
for i in range(3):
    ax[1,i].set_xlabel(r"$\hat e\cos\hat\theta$", fontsize=30)
    ax[1,i].set_ylim((-0.15,0.15))
    ax[1,i].set_xlim((-0.4,0.4))
plt.subplots_adjust(hspace=0.3)
ax[0,0].set_title(r"Standard", fontsize=36, pad=20)
ax[0,1].set_title(r"$e_2$-driving", fontsize=36, pad=20)
ax[0,2].set_title(r"Large $e_0$", fontsize=36, pad=20)

ax[0,0].set_xlim((-0.08,0.08))
ax[0,0].set_ylim((-0.05,0.05))
ax[1,0].set_ylim((-0.02,0.02))
ax[1,1].set_xlim((-0.1,0.5))


os.chdir("/home/jtlaune/Dropbox/multi-planet-architecture/docs/apsidal-alignment/")
fig.savefig(f"phasediagsex.png", bbox_inches="tight")
```