GWStrainPlotsSNR

August 14, 2018

1 Calculate the GW modes for each exoplanet that has the needed parameters in the dbase.

```
In [1]: # Started WEG  20180320.
        # See the gwtools.py which has utility and strain functions in it
        # and ExopDBase notebook that can download a new csv exop database.
```

2 References

P. Amaro-Seoane et al. "Triplets of supermassive black holes: astrophysics, gravitational waves and detection," MNRAS 402 2308-2320 (2010).

P. C. Peters and J. Mathews, "Gravitational Radiation from Point Masses in a Keplerian Orbit," Phys. Rev. 131 (1963) 435-440.

Michele Maggiore, "Gravitational Waves. Volume 1: Theory and Experiments," Oxford Univ. Press, 2008.

Shane Larson, "Sensitivity Curves for . . . " http://www.srl.caltech.edu/~shane/sensitivity/

Neil Cornish and Travis Robson, "The construction and use of LISA sensitivy curves," https://arxiv.org/abs/1803.01944

```
In [2]: import sys, os
        import numpy as np
        import urllib as ul
        import pandas as pd
        import gwTools as gwt
        import matplotlib.pyplot as plt
        %matplotlib inline
        import scipy as sp
        import scipy.interpolate as spint
```

3 CalTech Exop Database (from ExopDBase notebook)

3.1 Update or not from the CalTech database. Directories for the dbase and to save plots.

```
In [3]: thisDir = os.getcwd()  # This is the /python subdirectory.
        csvDir = thisDir + '/../dbases/'  # Will the /../ work on non-Unices?
        pixDir = thisDir + '/../pix/'
```

3.2 Search string, might want RA and DEC also.

In [4]:
```
# The search URL and search string/request.
exopURL = \
  "https://exoplanetarchive.ipac.caltech.edu/cgi-bin/nstedAPI/nph-\
nstedAPI?" ;

#searchString = \
#"table=exoplanets&select=pl_hostname,ra,dec&order=dec&format=CSV";*)

# The Below does NOT have right ascension and declination.  Will likely want them for 
# Can add later in its own Panda dataframe and/or merge into the main one in GWStrainP
# variables come from NASA Exoplanet Archive, the keywords are defined here:
#https://exoplanetarchive.ipac.caltech.edu/docs/API_exoplanet_columns.html
searchString = \
  "table=exoplanets&select=pl_hostname,pl_letter,pl_discmethod,pl_\
orbper,pl_orbsmax,pl_orbeccen,pl_bmassj,st_dist,st_mass,rowupdate,st_\
plx&order=dec&format=CSV";
```

3.3 Flags for fresh import and for saving the CSV file.

In [5]:
```
# Set to True to re-read the EXop Dbase from Caltech.  False to use csvFname below.
newImport = False;
#newImport = False;
saveFile = True;# Future work, when we do NOT want an intermediate file here would set
#saveFile = False;
```

In [6]:
```
# csv file below was downloaded earlier with code below.  newImport = False to use it.
# created.  This takes a few seconds.
csvFileName = csvDir + 'exopP_20180518_173344.csv'
if newImport and saveFile:
    myDateTimeStamp = gwt.dateTimeStamp()  # See the gwtools.py file with this and oth
    csvFileName = csvDir + 'exopP_' + myDateTimeStamp + '.csv'
    ofile = open(csvFname, 'w')
    with ul.request.urlopen(exopURL + searchString) as response:
        for aline in response:
            ofile.write( aline.decode('utf-8') )  # byte-string needs to be decoded. u
    ofile.close()
    print('Saved database file ' + csvFileName)
```

3.4 Read the CSV file and drop the rows/exops with NaN in the important fields. See ExopDBase.ipynb and re-run it for updating the dbase.

In [7]:
```
print('Using database file ' + csvFileName)
with open(csvFileName, 'r') as ifile:
    print(ifile.readline(), '\n', ifile.readline() )  #Print a couple of lines and res

    ifile.seek(0);
```

```
        dbData = pd.read_csv(ifile)  # Read in the whole file to a Panda Dataframe, handle
        #ifile.close()  # Should close when you leave the "with."
```

```
Using database file /home/gabella/Documents/astro/exop/exoplanetsMath/python/../dbases/exopP_20
pl_hostname,pl_letter,pl_discmethod,pl_orbper,pl_orbsmax,pl_orbeccen,pl_bmassj,st_dist,st_mass

 HD 142022 A,b,Radial Velocity,1928.00000000,2.930000,0.530000,4.44000,35.87,0.90,2018-04-26,27
```

```
In [8]: dbData.head(10)  # NaN's show up when the field has no data.  Need both masses, eccent
        # and distance.
```

```
Out[8]:   pl_hostname pl_letter    pl_discmethod   pl_orbper  pl_orbsmax  \
        0  HD 142022 A         b  Radial Velocity  1928.00000        2.93
        1     HD 39091         b  Radial Velocity  2151.00000        3.38
        2  HD 137388 A         b  Radial Velocity   330.00000        0.89
        3      GJ 3021         b  Radial Velocity   133.71000        0.49
        4     HD 63454         b  Radial Velocity     2.81805        0.04
        5    HD 212301         b  Radial Velocity     2.24571        0.03
        6      CHXR 73         b          Imaging         NaN      210.00
        7       CT Cha         b          Imaging         NaN      440.00
        8    HD 196067         b  Radial Velocity  3638.00000        5.02
        9     HD 68402         b  Radial Velocity  1103.00000        2.18

           pl_orbeccen  pl_bmassj  st_dist  st_mass   rowupdate  st_plx
        0       0.5300      4.440    35.87     0.90  2018-04-26   27.88
        1       0.6405     10.270    18.21     1.10  2014-07-23   54.92
        2       0.3600      0.200    38.45     0.68  2018-04-26   26.01
        3       0.5110      3.370    17.62     0.90  2014-05-14   56.76
        4       0.0000      0.250    35.80     0.42  2018-04-26   27.93
        5       0.0000      0.510    52.72     1.55  2018-04-26   18.97
        6          NaN     12.569      NaN     0.35  2014-05-14     NaN
        7          NaN     17.000   165.00      NaN  2014-05-14     NaN
        8       0.6600      6.900    43.57     1.29  2014-05-14   22.95
        9       0.0300      3.070    78.00     1.12  2016-11-10   12.82
```

3.5 Drop the exops/rows with NaN (missing values) in the following fields:

3.5.1 pl_orbeccen (eccentricity), pl_orbper (orbital period), pl_obsmax (semimajor axis), pl_bmassj (planet mass), st_dist (distance to host star), st_mass (stellar mass)

```
In [9]: # {"pl_hostname", "pl_letter", "pl_discmethod", "pl_orbper", \
        # "pl_orbsmax", "pl_orbeccen", "pl_bmassj", "st_dist", "st_mass", \
        # "rowupdate", "st_plx"}
        print('Length all data, dbData ', len(dbData) )
        aData = dbData.dropna(axis = 0, how = 'any', subset = ['pl_orbeccen'])
        print('Length with pl_orbeccen\t', len(aData) )
```

```
aData = aData.dropna(axis = 0, how = 'any', subset = ['pl_orbper'])
print('Length with pl_orbper\t', len(aData) )

aData = aData.dropna(axis = 0, how = 'any', subset = ['pl_orbsmax'])
print('Length with pl_orbsmax\t', len(aData) )

aData = aData.dropna(axis = 0, how = 'any', subset = ['pl_bmassj'])
print('Length with pl_bmassj\t', len(aData) )

aData = aData.dropna(axis = 0, how = 'any', subset = ['st_dist'])
print('Length with st_dist\t', len(aData) )

aData = aData.dropna(axis = 0, how = 'any', subset = ['st_mass'])
print('Length with st_mass\t', len(aData) )
```

```
Length all data, dbData  3726
Length with pl_orbeccen         1192
Length with pl_orbper         1192
Length with pl_orbsmax         1023
Length with pl_bmassj         941
Length with st_dist         853
Length with st_mass         846
```

3.6 So aData is the working exoplanet data frame after filtering, as a Panda DataFrame. Later should consider filling in missing data with Kepler or other calculations.

3.7 Physical Constants, made explicit here. The CalTech exop dbase has an FAQ on the units they use for each parameter. General URL https://exoplanetarchive.ipac.caltech.edu/ and the one for units under Support>Documentation>Table Column Definitions> Confirmed Planets https://exoplanetarchive.ipac.caltech.edu/applications/DocSet/index.html?doctree=/docs/docme

```
In [10]: # Some scipy.constants for comparison mostly.
         from scipy.constants import speed_of_light, gravitational_constant, c, G, pi

         massSun = 1.989e30; #(*kg *)
         massJ = 1.898e27; #(* kg *)
         massE = 5.972e24; #(* kg *)
         massJe = massJ/massE; #(* Jupiter mass is 317.9 earth masses *)
         massJs = massJ/massSun; #(* relative to the sun's mass *)

         pc = 30.86e15; #(* meters, parsec *)
         au = 149.6e9; #(* meters, astron unit *)

         cee = 299792458.0; #(* meters/s, speed of light *)
         print('Compare my cee ', cee, ' and scipy.constants ', speed_of_light)
```