Example_parse_mpcorb_json

A Jupyter Notebook to illustrate various aspects of the mpc_orb package, primarily focusing on using the MPCORB class to read the contents of "mpcorb.json" files

MPC: April 2023

NB: The original version of this jupyter notebook is expected to live in

> https://github.com/Smithsonian/mpc-public/mpc_orb/demos/Example_parse_mpcorb_json.ipynb (https://github.com/Smithsonian/mpc-public/mpc_orb/demos/Example_parse_mpcorb_json.ipynb)

Notebook Contents

(0) Installation ... ¶

In order to execute the code in this notebook, the user must have installed the mpc_orb package onto their system via some command like

> "pip install mpc_orb"

(1) Basics of mpc_orb ...

Import modules from the mpc_orb package ...

- The "MPCORB" class provides functionality for parsing mpc_orb.json files
- The "filepaths" module defines some convenient directories & files

In [1]:
```
# Import the MPCORB class from the mpc_orb package ...
from mpc_orb import MPCORB, filepaths

# Other useful imports
import json
```

Define a filepath to an mpc_orb.json file

Here we use an example json file (included in the mpc*orb repository) for the object $2012 HN*{13}$ which is an object whose best fit orbit requires Yarkovski non-gravitational components.

In [8]:
```
# Define a filepath to an example json file provided in the package
demo_filepath = filepaths.demo_2012HN13
print(f'demo_filepath=\n {demo_filepath} \n')
```

Instantiate an MPCORB object & use it to parse the above json file

- NB The parsing is done by default "behind-the-scenes" upon instantiation
- When we print out the *variables* available within the MPCORB object, we see that there are many *dictionaries*, plus some "COM" and "CAR" objects that we will discuss in more detail below.

In [3]:
```
# Instantiate an MPCORB object & use it to parse the above json file
M = MPCORB(demo_filepath)

# Demonstrate the available variables
print('\n Here we print-out the instance variables for an "MPCORB" class
object ... \n')
for attribute in vars(M):
    print(f'\t{attribute:>20} : {type(M.__dict__[attribute])}')
```

```
 Here we print-out the instance variables for an "MPCORB" class object
...

                schema_json : <class 'NoneType'>
             categorization : <class 'dict'>
           designation_data : <class 'dict'>
                 epoch_data : <class 'dict'>
              magnitude_data : <class 'dict'>
                   moid_data : <class 'dict'>
          non_grav_booleans : <class 'dict'>
      orbit_fit_statistics : <class 'dict'>
             software_data : <class 'dict'>
               system_data : <class 'dict'>
                       COM : <class 'mpc_orb.parse.COORD'>
                       CAR : <class 'mpc_orb.parse.COORD'>
                         q : <class 'dict'>
                         e : <class 'dict'>
                         i : <class 'dict'>
                      node : <class 'dict'>
                   argperi : <class 'dict'>
                 peri_time : <class 'dict'>
                 yarkovski : <class 'dict'>
                         x : <class 'dict'>
                         y : <class 'dict'>
                         z : <class 'dict'>
                        vx : <class 'dict'>
                        vy : <class 'dict'>
                        vz : <class 'dict'>
```

Examine one of the best-fit orbital parameters, "q" (the pericenter distance)

- N.B. (1) A more thorough examination of *all* available parameters is provided below in (2) MPCORB Variables
- N.B. (2) A more thorough examination of the *describe* function is provided below in (3) The *Describe* Function

In [4]:
```
'''
Printing out the variable, "q"  (the pericenter distance),
we see that it is a dictionary, containing elements for
 - the best-fit value ('val'), and
 - the uncertainty on the best-fit value ('unc')

The describe function confirms that "q" is the "Cometary Pericenter Dist
ance", and has units of "au"
'''
print(M.q)

print(json.dumps(M.describe("q"), indent=4), '\n')
```

```
{'val': 0.97469103481812, 'unc': 1.37975e-08}
{
    "q": {
        "description": "Cometary Pericenter Distance",
        "units": [
            "au"
        ]
    }
}
```

(2) MPCORB Variables

In this section we examine all of the variables available within the MPCORB object

COM & CAR objects

As can be seen above, there are "COM" and "CAR" contained within the MPCORB Object

Here we display the variables available within the "CAR" object

In [5]:
```
# Demonstrate the attributes available in the "COM" coord-object contain
ed
print('\n CAR instance variables ... ')
for attribute in vars(M.CAR):
    print(f'\t{attribute:>20} : {type(M.CAR.__dict__[attribute])}')
```

```
 CAR instance variables ...
            coefficient_names : <class 'list'>
           coefficient_values : <class 'list'>
         coefficient_uncertainties : <class 'list'>
                  eigenvalues : <class 'list'>
                   covariance : <class 'dict'>
              covariance_array : <class 'numpy.ndarray'>
                  element_dict : <class 'dict'>
                             x : <class 'dict'>
                             y : <class 'dict'>
                             z : <class 'dict'>
                            vx : <class 'dict'>
                            vy : <class 'dict'>
                            vz : <class 'dict'>
                      yarkovski : <class 'dict'>
```

Access the Cartesian orbit elements

The Cartesian orbital elements within the MPCORB object can be accessed in a few different ways.

Here we explicitly demonstrate the contents of some key variables, and the different ways that they can be accessed.

In [6]:
```
# Demonstrate access to Cartesian elements

print('\n\t coefficient names ... ')
print('\t',M.CAR.coefficient_names )

print('\n\t coefficient values ... ')
print('\t',M.CAR.coefficient_values )

print('\n\t coefficient uncertainties ... ')
print('\t',M.CAR.coefficient_uncertainties )

print('\n\t element dictionary with combined values & uncertainties ...
')
for k,v in M.CAR.element_dict.items():
    print('\t',f'{k}:{v}' )

print('\n\t individual element access (x)... ')
print('\t',M.CAR.x )

print('\n\t covariance array ... ')
print('\t',M.CAR.covariance_array )




# Demonstrate that access to individual elements is also possible from t
he MPCORB object
print('-'*33)
print('\nMPCORB also has individual-element attributes ... ')

print('\n\t individual element access (x)... ')
print('\t',M.x )
```

```
        coefficient names ...
        ['x', 'y', 'z', 'vx', 'vy', 'vz', 'yarkovski']

        coefficient values ...
        [0.400637254703697, 1.72530013679644, -0.120928190519571, -0.0
102316591071472, 0.00429614246581105, -0.000349929761438383, -0.0011854
19262123]

        coefficient uncertainties ...
        [1.41332e-07, 2.24426e-08, 5.87873e-08, 3.40373e-10, 4.71794e-
10, 3.34717e-10, 1e-07]

        element dictionary with combined values & uncertainties ...
        x:{'val': 0.400637254703697, 'unc': 1.41332e-07}
        y:{'val': 1.72530013679644, 'unc': 2.24426e-08}
        z:{'val': -0.120928190519571, 'unc': 5.87873e-08}
        vx:{'val': -0.0102316591071472, 'unc': 3.40373e-10}
        vy:{'val': 0.00429614246581105, 'unc': 4.71794e-10}
        vz:{'val': -0.000349929761438383, 'unc': 3.34717e-10}
        yarkovski:{'val': -0.001185419262123, 'unc': 1e-07}

        individual element access (x)...
        {'val': 0.400637254703697, 'unc': 1.41332e-07}

        covariance array ...
        [[ 1.99747709e-14 -2.47579831e-15  3.87101874e-15  4.60192564e
-17
   6.44504078e-17 -1.33310099e-18 -6.43587493e-12]
 [-2.47579831e-15  5.03668761e-16 -1.56419288e-16 -4.43086368e-18
  -8.32243383e-18  1.51538836e-18  1.43727507e-12]
 [ 3.87101874e-15 -1.56419288e-16  3.45594667e-15  9.88296966e-18
   1.32178687e-17  1.42699944e-17 -1.14277528e-13]
 [ 4.60192564e-17 -4.43086368e-18  9.88296966e-18  1.15853646e-19
   1.48882275e-19 -4.99494773e-22 -9.56760386e-15]
 [ 6.44504078e-17 -8.32243383e-18  1.32178687e-17  1.48882275e-19
   2.22589311e-19  3.44719133e-21 -1.06279020e-14]
 [-1.33310099e-18  1.51538836e-18  1.42699944e-17 -4.99494773e-22
   3.44719133e-21  1.12035707e-19  5.67933081e-15]
 [-6.43587493e-12  1.43727507e-12 -1.14277528e-13 -9.56760386e-15
  -1.06279020e-14  5.67933081e-15  2.47188316e-08]]
--------------------------------

MPCORB also has individual-element attributes ...

        individual element access (x)...
        {'val': 0.400637254703697, 'unc': 1.41332e-07}
```

The *describe* function

The *describe* function can be used to access information / definitions for each and every attribute within an MPCORB instance.

In [7]:
```
# Demonstrate the available variables
print('\n MPCORB description ... ')
for attribute in vars(M):
    print(json.dumps(M.describe(attribute), indent=4), '\n')
```

```
 MPCORB description ...
{
    "schema_json": {
        "filepath": "schema_json/mpcorb_schema_latest.json"
    }
}

{
    "categorization": {
        "description": "Various different ways to categorize / sub-set
orbit / object types",
        "allowed properties": [
            "object_type_str",
            "object_type_int",
            "orbit_type_str",
            "orbit_type_int",
            "orbit_subtype_str",
            "orbit_subtype_int",
            "parent_planet_str",
            "parent_planet_int"
        ]
    }
}

{
    "designation_data": {
        "description": "The designations, numbers and names that may be
associated with the object",
        "allowed properties": [
            "permid",
            "packed_primary_provisional_designation",
            "unpacked_primary_provisional_designation",
            "orbfit_name",
            "packed_secondary_provisional_designations",
            "unpacked_secondary_provisional_designations",
            "iau_name"
        ]
    }
}

{
    "epoch_data": {
        "description": "Data concerning the orbit epoch: I.e. The date
at which the best-fit orbital coordinates are correct ",
        "allowed properties": [
            "timesystem",
            "timeform",
            "epoch"
        ]
    }
}

{
    "magnitude_data": {
        "description": "The absolute magnitude, H, and slope parameter,
G, information derived from the fitted orbit in combination with the ob
served apparent magnitudes. ",
```

```
        "allowed properties": [
            "H",
            "G",
            "G1",
            "G2",
            "G12",
            "photometric_model"
        ]
    }
}

{
    "moid_data": {
        "description": "Calculated MOIDs (Minimum Orbital Interception
Distances) at Epoch",
        "allowed properties": [
            "Venus",
            "Earth",
            "Mars",
            "Jupiter",
            "moid_units"
        ]
    }
}

{
    "non_grav_booleans": {
        "description": "Booleans to indicate whether any non-gravitatio
nal parameters are used in the orbit-fit. The actual fitted values and
their covariance properties are reported within the CAR and COT paramet
er sections.",
        "allowed properties": [
            "non_gravs",
            "non_grav_model",
            "non_grav_coefficients"
        ]
    }
}

{
    "orbit_fit_statistics": {
        "description": "Summary fit statistics associated with the best
-fit orbit, the observations used, etc",
        "allowed properties": [
            "sig_to_noise_ratio",
            "snr_below_3",
            "snr_below_1:",
            "U_param",
            "score1",
            "score2",
            "orbit_quality",
            "normalized_RMS",
            "not_normalized_RMS",
            "nobs_total",
            "nobs_total_sel",
            "nobs_optical",
            "nobs_optical_sel",
```

```
            "nobs_radar",
            "nobs_radar_sel",
            "arc_length_total",
            "arc_length_sel",
            "nopp",
            "numparams"
        ]
    }
}

{
    "software_data": {
        "description": "Details of the software used to perform orbital
fit and to create mpcorb output file",
        "allowed properties": [
            "fitting_software_name",
            "software_version",
            "fitting_datetime",
            "mpcorb_schema_version",
            "mpcorb_schema_sha256",
            "mpcorb_creation_datetime"
        ]
    }
}

{
    "system_data": {
        "description": "Ephemeris model assumed when integrating the mo
tion of the object, and the frame of reference used to specify the best
-fit orbital elements. ",
        "allowed properties": [
            "eph",
            "refplane",
            "EclipticObliquityArcseconds",
            "refframe",
            "force_model"
        ]
    }
}

{
    "COM": {
        "description": "Description of the best-fit orbit using cometar
y coordinates (plus any non-gravs) in heliocentric coordinates. Contain
s the best-fit orbit and covariance matrix.",
        "allowed properties": [
            "coefficient_specification",
            "coefficient_names",
            "coefficient_values",
            "coefficient_uncertainties",
            "eigenvalues",
            "covariance"
        ]
    }
}

{
```

```
    "CAR": {
        "description": "Cartesian Element Specification: Description of
the best-fit orbit based on a cartesian coordinate system (plus any non
-gravs). Contains the best-fit orbit and covariance matrix. Heliocentri
c coordinates.",
        "allowed properties": [
            "coefficient_specification",
            "coefficient_names",
            "coefficient_values",
            "coefficient_uncertainties",
            "eigenvalues",
            "covariance"
        ]
    }
}

{
    "q": {
        "description": "Cometary Pericenter Distance",
        "units": [
            "au"
        ]
    }
}

{
    "e": {
        "description": "Cometary Eccentricity",
        "units": [
            "null"
        ]
    }
}

{
    "i": {
        "description": "Cometary Inclination",
        "units": [
            "degrees"
        ]
    }
}

{
    "node": {
        "description": "Cometary Longitude of Ascending Node",
        "units": [
            "degrees"
        ]
    }
}

{
    "argperi": {
        "description": "Cometary Argument of Pericenter",
        "units": [
            "degrees"
```

```
            ]
        }
    }

    {
        "peri_time": {
            "description": "Cometary Time from Pericenter Passage",
            "units": [
                "days"
            ]
        }
    }

    {
        "yarkovski": {
            "description": "Yarkovski Component",
            "units": [
                "10^(-10)*au/day^2"
            ]
        }
    }

    {
        "x": {
            "description": "Cartesian Position Component",
            "units": [
                "au"
            ]
        }
    }

    {
        "y": {
            "description": "Cartesian Position Component",
            "units": [
                "au"
            ]
        }
    }

    {
        "z": {
            "description": "Cartesian Position Component",
            "units": [
                "au"
            ]
        }
    }

    {
        "vx": {
            "description": "Cartesian Velocity Component",
            "units": [
                "au/day"
            ]
        }
    }
```

```
{
    "vy": {
        "description": "Cartesian Velocity Component",
        "units": [
            "au/day"
        ]
    }
}

{
    "vz": {
        "description": "Cartesian Velocity Component",
        "units": [
            "au/day"
        ]
    }
}
```


In [ ]: