Notebook to illustrate the usage of the parse.MPCORB class to read the contents of "mpcorb.json" files

MPC: April 2023

NB: This jupyter notebook is expected to live in ../mpc-public/mpc_orb/demos/Example_parse_mpcorb_json.ipynb

Import the MPCORB class from the parse module in the mpc_orb directory ...

- NB: This assumes that mpc_orb has been installed via some command like

> "pip install mpc_orb"

In [3]:
```
# Import the MPCORB class from the mpc_orb package ...
from mpc_orb import MPCORB
```

Define a filepath to an example json file in the mpcorb format

In [2]:
```
# Import the convenience filepath-defn dictionary
from mpc_orb.filepaths import test_pass_mpcorb

# Define a filepath to an example json file provided in the package
filepath = test_pass_mpcorb[0]
print(f'filepath=\n {filepath} \n')
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call l
ast)
<ipython-input-2-d51a0f485ef0> in <module>
      1 # Import the convenience filepath-defn dictionary
----> 2 from mpc_orb.filepaths import test_pass_mpcorb
      3 
      4 # Define a filepath to an example json file provided in the pac
kage
      5 filepath = test_pass_mpcorb[0]

ImportError: cannot import name 'test_pass_mpcorb' from 'mpc_orb.filepa
ths' (/Users/matthewjohnpayne/Envs/mpc-public/mpc_orb/mpc_orb/filepath
s.py)
```

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

- NB The parsing is done by default "behind-the-scenes" upon instantiation

In [4]:

```
# Instantiate an MPCORB object & use it to parse the above json file
M = MPCORB(filepath)

# Demonstrate the available variables
print('\n MPCORB instance variables ... ')
for attribute in vars(M):
    print(f'\t{attribute:>20} : {type(M.__dict__[attribute])}')
```

```
 MPCORB instance variables ... 
                 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'>
```

There are COM & CAR objects contained within the MPCORB Object

In [5]:

```
# Demonstrate the attributes available in the "COM" coord-object contained
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 orbit elements from the MPCORB object in different ways

In [6]:

```
# Demonstrate access to Cartesian elements
print('\nformat = CAR = Cartesian ... ')
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 ...
')
print('\t',M.CAR.element_dict )

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

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


# Demonstrate access to Cartesian elements
print('-'*33)
print('\nformat = COM = Cometary ... ')
print('\n\t coefficient names ... ')
print('\t',M.COM.coefficient_names )

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

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

print('\n\t element dictionary with combined values & uncertainties ...
')
print('\t',M.COM.element_dict )

print('\n\t individual element access (e)... ')
print('\t',M.COM.e )

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


# Demonstrate that access to individual elements (Cartesian & Cometary)
is also possible from the MPCORB object
print('-'*33)
print('\nMPCORB also has individual-element attributes ... ')

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

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

```
format = CAR = Cartesian ...

         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': {'v
al': 1.72530013679644, 'unc': 2.24426e-08}, 'z': {'val': -0.12092819051
9571, 'unc': 5.87873e-08}, 'vx': {'val': -0.0102316591071472, 'unc': 3.
40373e-10}, 'vy': {'val': 0.00429614246581105, 'unc': 4.71794e-10}, 'v
z': {'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]]
---------------------------------

format = COM = Cometary ...

         coefficient names ...
         ['q', 'e', 'i', 'node', 'argperi', 'peri_time', 'yarkovski']

         coefficient values ...
         [0.97469103481812, 0.307980763141286, 4.0744770505194, 183.498
2668700383, 97.2208277743442, 59765.3930151203, -0.001185419262123]

         coefficient uncertainties ...
         [1.37975e-08, 1.08365e-08, 2.11995e-06, 1.6376e-05, 2.03201e-0
5, 1.27375e-05, 1e-07]

         element dictionary with combined values & uncertainties ...
         {'q': {'val': 0.97469103481812, 'unc': 1.37975e-08}, 'e': {'va
l': 0.307980763141286, 'unc': 1.08365e-08}, 'i': {'val': 4.074477050519
4, 'unc': 2.11995e-06}, 'node': {'val': 183.4982668700383, 'unc': 1.637
6e-05}, 'argperi': {'val': 97.2208277743442, 'unc': 2.03201e-05}, 'peri
_time': {'val': 59765.3930151203, 'unc': 1.27375e-05}, 'yarkovski': {'v
al': -0.001185419262123, 'unc': 1e-07}}

         individual element access (e)...
         {'val': 0.307980763141286, 'unc': 1.08365e-08}

         covariance array ...
         [[ 1.90372276e-16 -1.47434163e-16 -2.06614671e-15  3.06841464e
-14
  -9.87183438e-14 -9.86859910e-14 -6.65779067e-13]
 [-1.47434163e-16  1.17429953e-16  2.14707995e-15 -2.45688328e-14
   8.30960806e-14  8.57623116e-14  7.74123590e-13]
 [-2.06614671e-15  2.14707995e-15  4.49420376e-12  1.83626987e-11
  -1.12585073e-11  9.83443194e-12 -7.84563017e-12]
 [ 3.06841464e-14 -2.45688328e-14  1.83626987e-11  2.68173255e-10
  -3.00990352e-10 -4.89642128e-11 -3.46714657e-10]
 [-9.87183438e-14  8.30960806e-14 -1.12585073e-11 -3.00990352e-10
   4.12905974e-10  1.61251212e-10  8.64625977e-10]
 [-9.86859910e-14  8.57623116e-14  9.83443194e-12 -4.89642128e-11
   1.61251212e-10  1.62242759e-10  8.09227561e-10]
 [-6.65779067e-13  7.74123590e-13 -7.84563017e-12 -3.46714657e-10
   8.64625977e-10  8.09227561e-10  2.47188316e-08]]
--------------------------------

MPCORB also has individual-element attributes ...

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

         individual element access (e)...
         {'val': 0.307980763141286, 'unc': 1.08365e-08}
```

Use the *describe* function to access information / definitions for each attribute

In [10]:

```
# Demonstrate the available variables
print('\n MPCORB description ... ')
for attribute in vars(M):
    print(f'\n{M.describe(attribute)}')
```

```
 MPCORB description ...

{'schema_json': {'filepath': '/Users/matthewjohnpayne/Envs/mpc-public/m
pc_orb/mpc_orb/json_files/schema_json/mpcorb_schema.json'}}

{'categorization': {'description': 'Various different ways to categoriz
e / sub-set orbit / object types'}}

{'designation_data': {'description': 'The designations, numbers and nam
es that may be associated with the object'}}

{'epoch_data': {'description': 'Data concerning the orbit epoch: I.e. T
he date at which the best-fit orbital coordinates are correct [double-c
heck: Does TT ++ TDT???]'}}

{'magnitude_data': {'description': 'The absolute magnitude, H, and slop
e parameter, G, information derived from the fitted orbit in combinatio
n with the observed apparent magnitudes. '}}

{'moid_data': {'description': 'Calculated MOIDs (Minimum Orbital Interc
eption Distances) at Epoch'}}

{'non_grav_booleans': {'description': 'Booleans to indicate whether any
non-gravitational parameters are used in the orbit-fit. The actual fitt
ed values and their covariance properties are reported within the CAR a
nd COT parameter sections.'}}

{'orbit_fit_statistics': {'description': 'Summary fit statistics associ
ated with the best-fit orbit, the observations used, etc'}}

{'software_data': {'description': 'Details of the software used to perf
orm orbital fit and to create mpcorb output file'}}

{'system_data': {'description': 'Ephemeris model assumed when integrati
ng the motion of the object, and the frame of reference used to specify
the best-fit orbital elements. '}}

{'COM': {'description': 'Description of the best-fit orbit using cometa
ry coordinates (plus any non-gravs) in heliocentric coordinates. Contai
ns the best-fit orbit and covariance matrix.'}}

{'CAR': {'description': 'Cartesian Element Specification: Description o
f the best-fit orbit based on a cartesian coordinate system (plus any n
on-gravs). Contains the best-fit orbit and covariance matrix. Heliocent
ric coordinates.'}}

{'q': {'description': 'Cometary Pericenter Distance', 'properties': {'u
nit': {'description': 'Physical Units associated with Cometary Pericent
er Distance', 'enum': ['au']}}}}

{'e': {'description': 'Cometary Eccentricity', 'properties': {'unit':
{'description': 'Physical Units associated with Cometary Eccentricity',
'enum': ['null']}}}}

{'i': {'description': 'Cometary Inclination', 'properties': {'unit':
{'description': 'Physical Units associated with Cometary Inclination',
'enum': ['degrees']}}}}
```

```
{'node': {'description': 'Cometary Longitude of Ascending Node', 'prope
rties': {'unit': {'description': 'Physical Units associated with Cometa
ry Longitude of Ascending Node', 'enum': ['degrees']}}}}

{'argperi': {'description': 'Cometary Argument of Pericenter', 'propert
ies': {'unit': {'description': 'Physical Units associated with Cometary
Argument of Pericenter', 'enum': ['degrees']}}}}

{'peri_time': {'description': 'Cometary Time from Pericenter Passage',
'properties': {'unit': {'description': 'Physical Units associated with
Cometary Time from Pericenter Passage', 'enum': ['days']}}}}

{'yarkovski': {'description': 'Yarkovski Component', 'properties': {'un
it': {'description': 'Physical Units associated with Yarkovski non-grav
component', 'enum': ['10^(-10)*au/day^2']}}}}

{'x': {'description': 'Cartesian Position Component', 'properties': {'u
nit': {'description': 'Physical Units associated with Cartesian Positio
n Component', 'enum': ['au']}}}}

{'y': {'description': 'Cartesian Position Component', 'properties': {'u
nit': {'description': 'Physical Units associated with Cartesian Positio
n Component', 'enum': ['au']}}}}

{'z': {'description': 'Cartesian Position Component', 'properties': {'u
nit': {'description': 'Physical Units associated with Cartesian Positio
n Component', 'enum': ['au']}}}}

{'vx': {'description': 'Cartesian Velocity Component', 'properties':
{'unit': {'description': 'Physical Units associated with Cartesian Velo
city Component', 'enum': ['au/day']}}}}

{'vy': {'description': 'Cartesian Velocity Component', 'properties':
{'unit': {'description': 'Physical Units associated with Cartesian Velo
city Component', 'enum': ['au/day']}}}}

{'vz': {'description': 'Cartesian Velocity Component', 'properties':
{'unit': {'description': 'Physical Units associated with Cartesian Velo
city Component', 'enum': ['au/day']}}}}
```

```
In [ ]:
```