Washington State University

Internationally Poised
Globally Engaged

Seminar @ AGH University

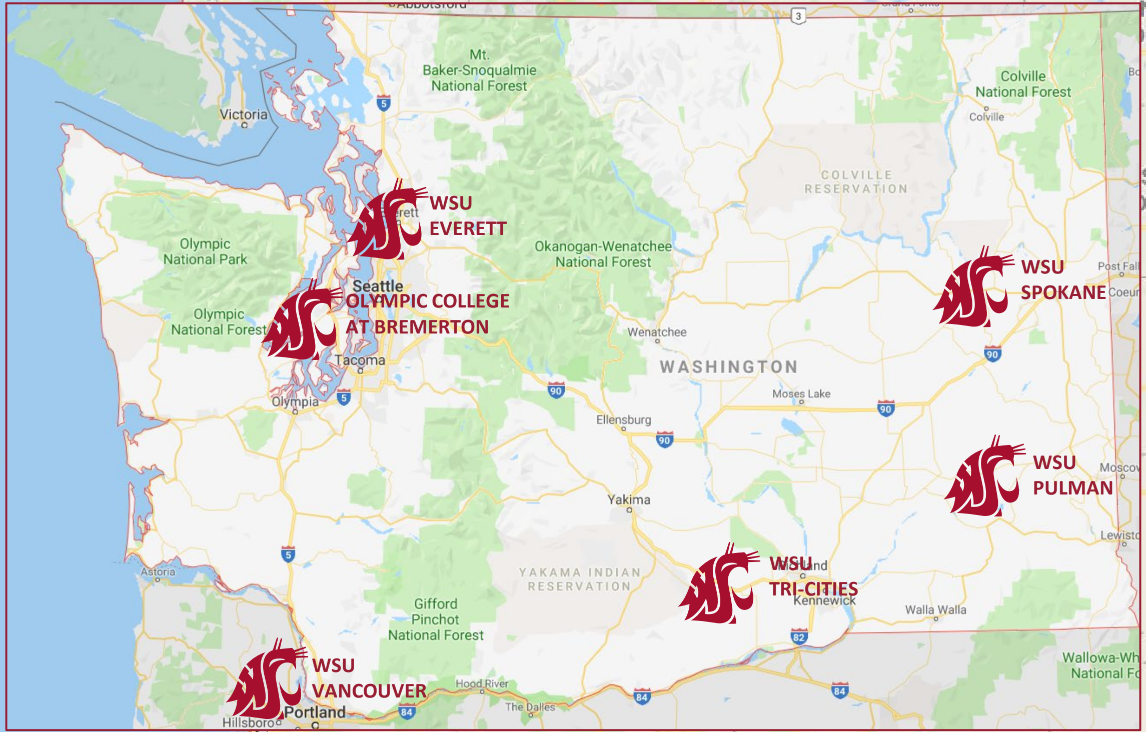
Joseph Iannelli, PhD, FHEA
Professor & Director
WSU Europe
Education & Research Liaison
to European Universities

Junes 2026

WASHINGTON STATE
UNIVERSITY

Welcomes You

State of Washington – Pacific Northwest



- Florid Economy - No State Income Tax
- ~ 8 Million Residents
- Maritime Ports: 1 Day Closer to Asian Markets than those in CA
- ~45% of Jobs in WA are Related to International Trade
- Affordable Renewable Energy Available to Companies e.g. BMW (Mainly Hydro, Wind)
- Leader in Sustainable Biofuels, Hydrogen
- Institute for Northwest Energy Futures

WASHINGTON STATE UNIVERSITY

Welcomes You

WSU: Original WA Land-Grant University Transforming Lives, Driving Global Innovation

- FACULTIES – 11 of Them

  Agricultural, Human, and Natural Resource Sciences - Arts & Sciences- Business – Communication – Education & Sport Sciences - Engineering and Architecture - Health Sciences – Medicine - Nursing - Pharmacy - Veterinary Medicine

- DEGREE PROGRAMS - Over 200

  Undergraduate: •95 Majors, •85 Minors, •100+ Certificates

  Post-graduate: •47 PhD Programs, •66 Master's Programs

WASHINGTON STATE UNIVERSITY

Welcomes You

WSU Europe Global Engagement Mission Goals

- Graduating internationally-minded diverse students for leadership in multicultural global societies, by expanding student exchanges for Study & Research abroad

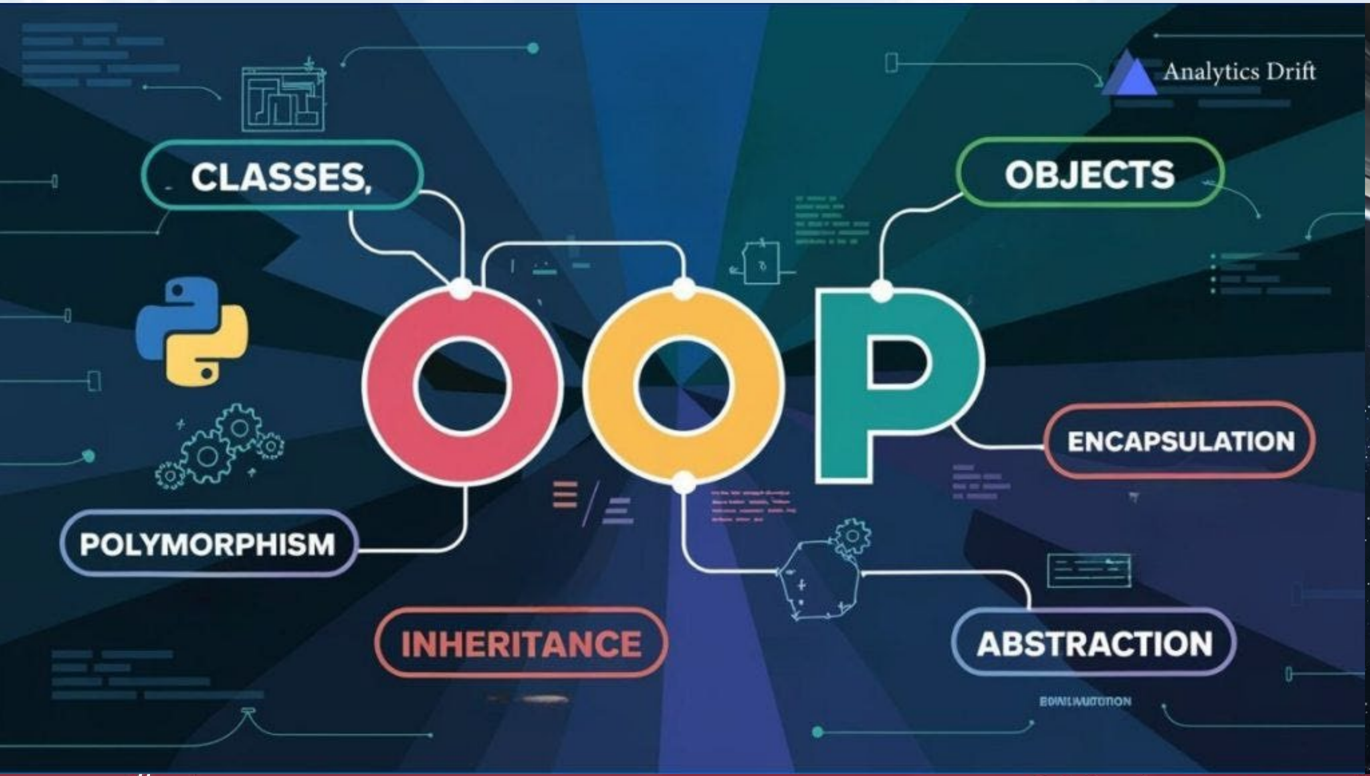
- Establishing international partnerships to advance knowledge more effectively than allowed by the resources of any single institution

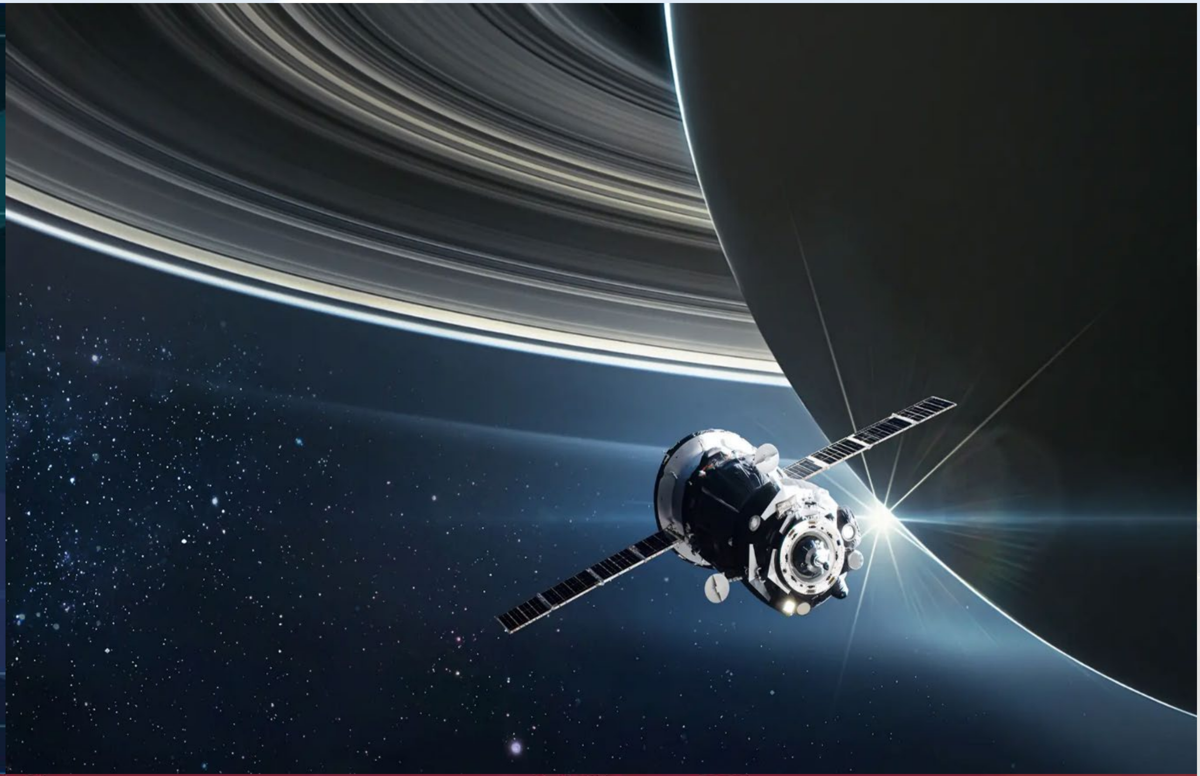
- Securing domestic & transatlantic funding to support international research collaborations and provide students with research-&-study-abroad fellowships that remove financial barriers

- Integrating education and research within international initiatives supported by domestic & transatlantic extramural funding

Python Meeting the Cosmos and Differential Systems

Object-Oriented Python Programming in Computational Astrodynamics





WASHINGTON STATE UNIVERSITY

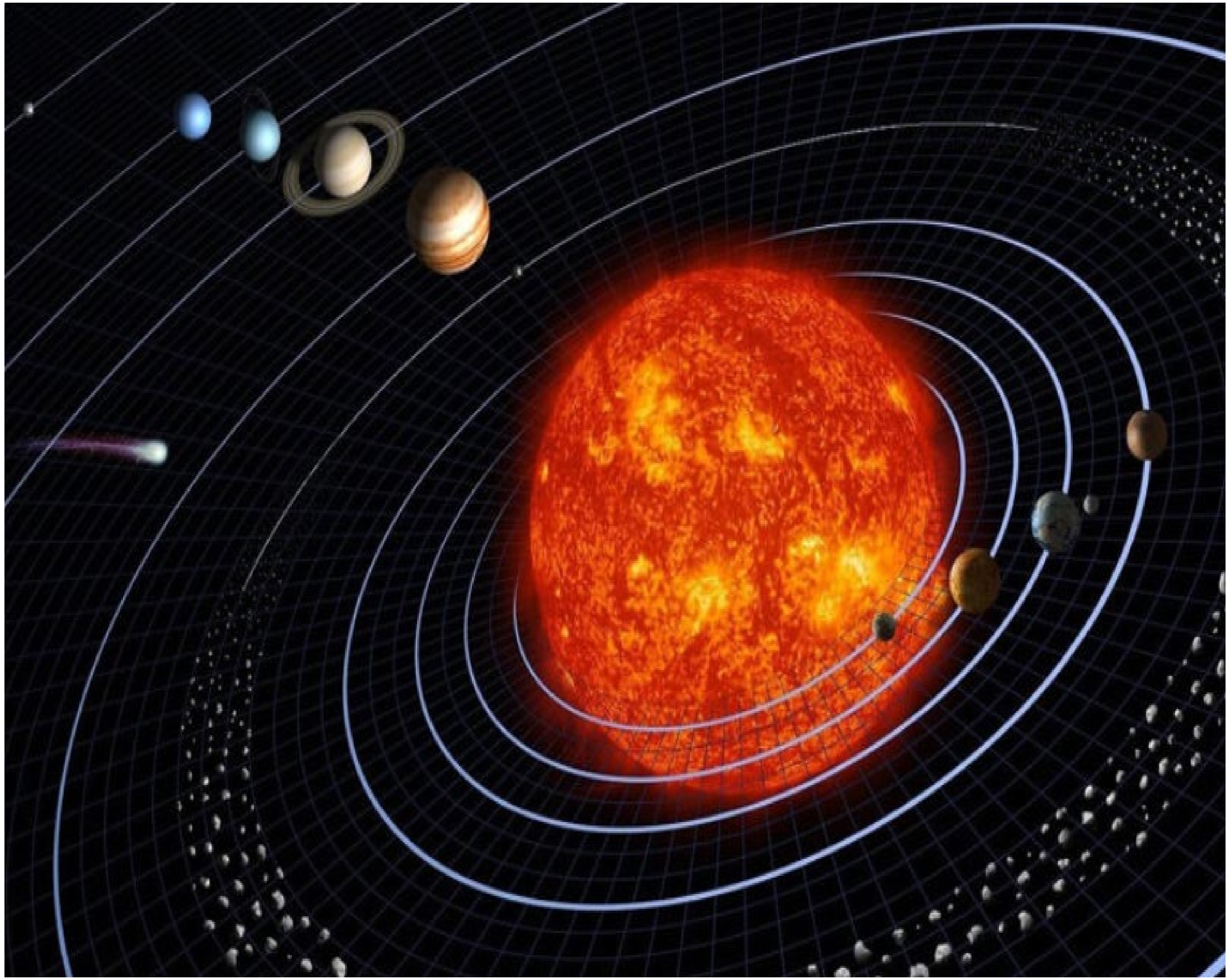
Welcomes You
Begrüßt Sie

Wita Cię
Vous Accueille

Contents

- Solar System and Initial-Value System
- Astrodynamics System – N-Body Problem
- Python Program Architecture – Libraries & Classes
- Python Inheritance
- Initial State for International Celestial Reference Frame
- Differential System & Jacobian Class Methods
- SciPy Differential Solver Class

Solar System and Initial-Value System



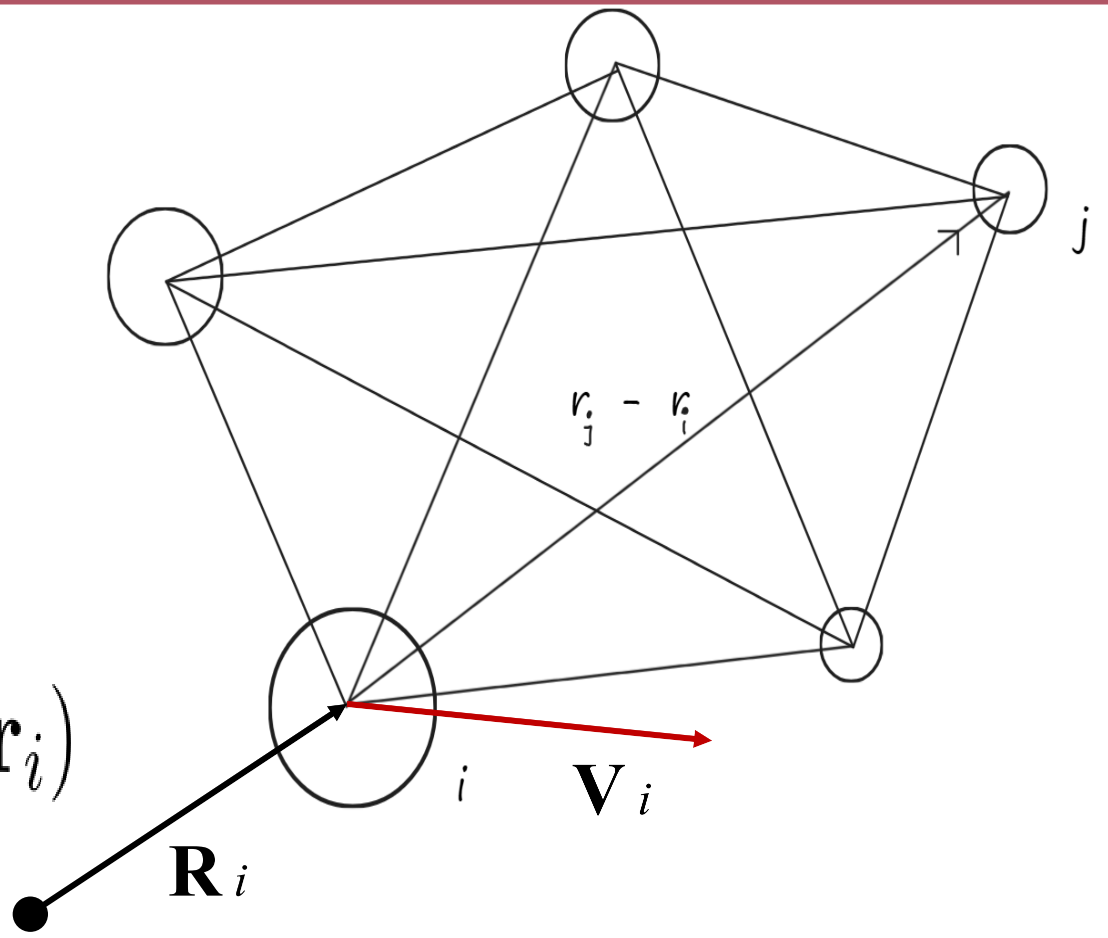
$$\begin{cases} \dfrac{d\mathbf{q}}{dt} = \mathbf{F}(t, \mathbf{q}(t)) \\ \mathbf{q}(t_0) = \mathbf{q}_0 \end{cases}$$

State Vector / Array - Jacobian

$$\mathbf{q} \equiv \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix}, \quad \mathbf{F} \equiv \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix} \qquad \mathbf{J} \equiv \frac{\partial \mathbf{F}}{\partial \mathbf{q}} = \left\{ \frac{\partial F_i}{\partial q_j} \right\}$$

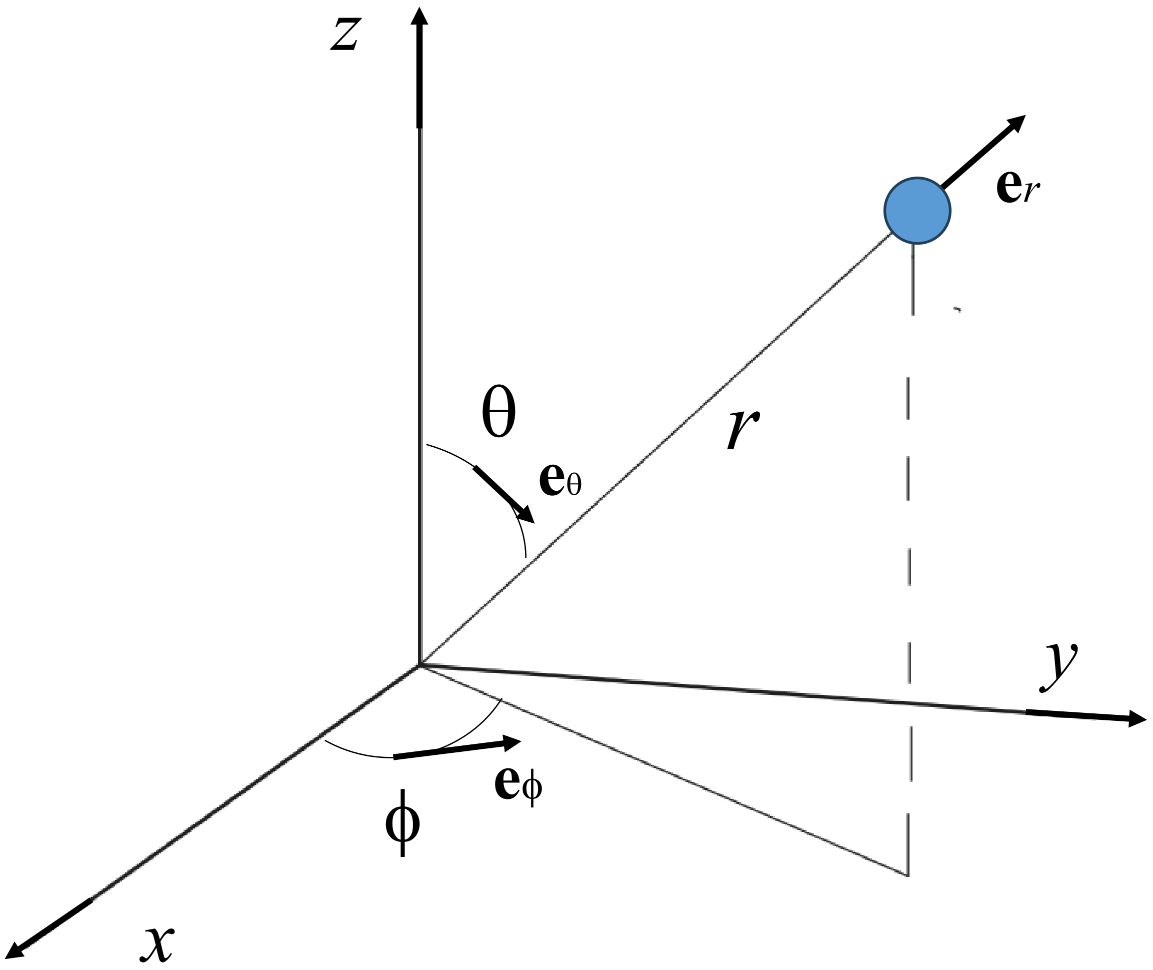
Astrodynamics System – N-Body Problem

$$\begin{cases} \dfrac{d\mathbf{R}_i}{dt} = \mathbf{V}_i(t) \\[2ex] \dfrac{d\mathbf{V}_i}{dt} = -\displaystyle\sum_{j=1, j\neq i}^{n} \frac{GM_j}{\|\mathbf{r}_j - \mathbf{r}_i\|^3} (\mathbf{r}_j - \mathbf{r}_i) \end{cases}$$



Sample Model for One Celestial Body

$$
\begin{cases}
\dfrac{dr}{dt} = v_r \\
\dfrac{dv_r}{dt} = r\omega_\theta^2 + r\omega_\phi^2 \sin^2\theta - \dfrac{GM}{r^2} \\
\dfrac{d\phi}{dt} = \omega_\phi \\
\dfrac{d\omega_\phi}{dt} = -2\dfrac{v_r\omega_\phi}{r} - 2\omega_\phi\omega_\theta\dfrac{\cos\theta}{\sin\theta} \\
\dfrac{d\theta}{dt} = \omega_\theta \\
\dfrac{d\omega_\theta}{dt} = -2\dfrac{v_r\omega_\theta}{r} + \omega_\phi^2 \sin\theta\cos\theta
\end{cases}
$$



Differential Equations Reflected in Python

$$
\begin{cases}
\dfrac{dr}{dt} = v_r \\
\dfrac{dv_r}{dt} = r\omega_\theta^2 + r\omega_\phi^2 \sin^2\theta - \dfrac{GM}{r^2} \\
\dfrac{d\phi}{dt} = \omega_\phi \\
\dfrac{d\omega_\phi}{dt} = -2\dfrac{v_r\omega_\phi}{r} - 2\omega_\phi\omega_\theta\dfrac{\cos\theta}{\sin\theta} \\
\dfrac{d\theta}{dt} = \omega_\theta \\
\dfrac{d\omega_\theta}{dt} = -2\dfrac{v_r\omega_\theta}{r} + \omega_\phi^2 \sin\theta\cos\theta
\end{cases}
\qquad
\begin{cases}
q[0] = r \\
q[1] = v_r \\
q[2] = \phi \\
q[3] = \omega_\phi \\
q[4] = \theta \\
q[5] = \omega_\theta
\end{cases}
$$

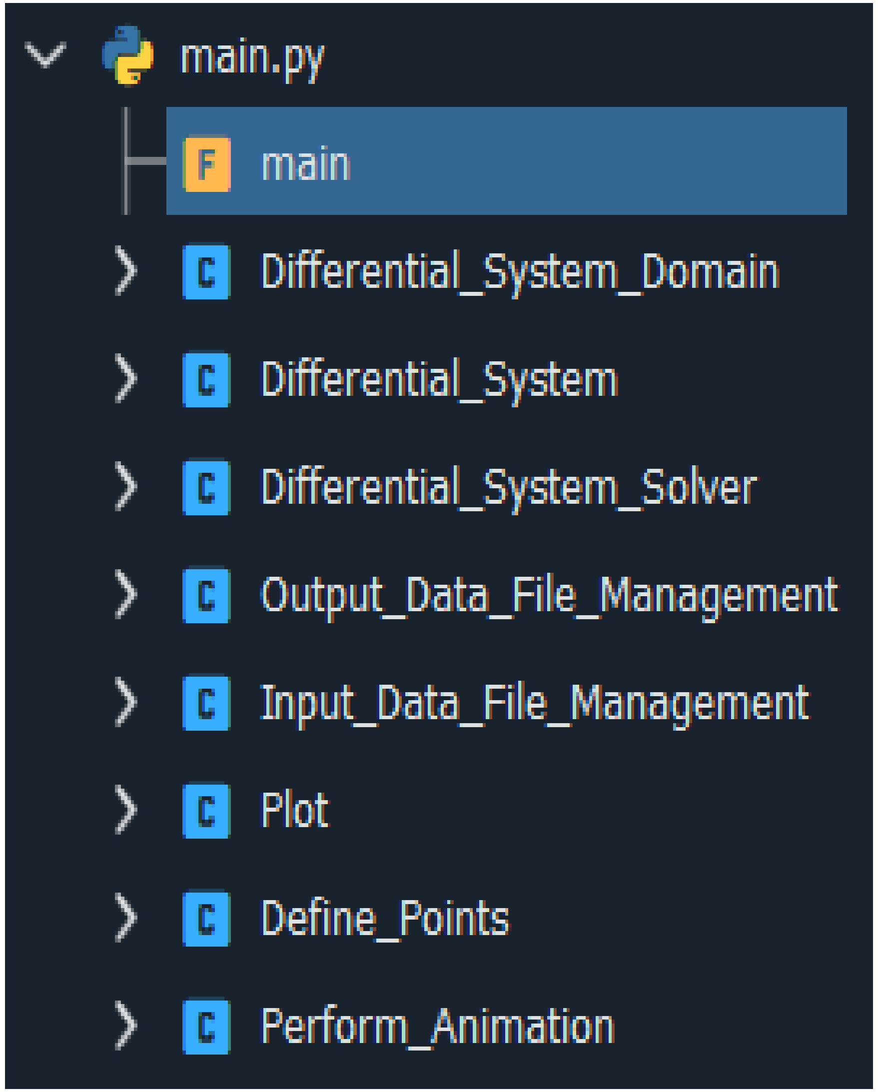
```
dq_dt[ 0 ] = q[ 1 ]

dq_dt[ 1 ] = q[ 0 ] * ( q[ 3 ]  * math.sin( q[ 4 ] ) )**2.0 - GM / q[ 0 ]**2.0 + q[ 0 ] * q[ 5 ]**2.0

dq_dt[ 2 ] = q[ 3 ]

dq_dt[ 3 ] = -2.0 * q[ 1 ] * q[ 3 ] / q[ 0 ] - 2.0 * q[ 3 ] * q[ 5 ] * math.cos( q[ 4 ] ) / math.sin( q[ 4 ] )

dq_dt[ 4 ] = q[ 5 ]

dq_dt[ 5 ] = -2.0 * q[ 1 ] * q[ 5 ] / q[ 0 ] + q[ 3 ] * q[ 3 ]  * math.cos( q[ 4 ] ) * math.sin( q[ 4 ] )
```

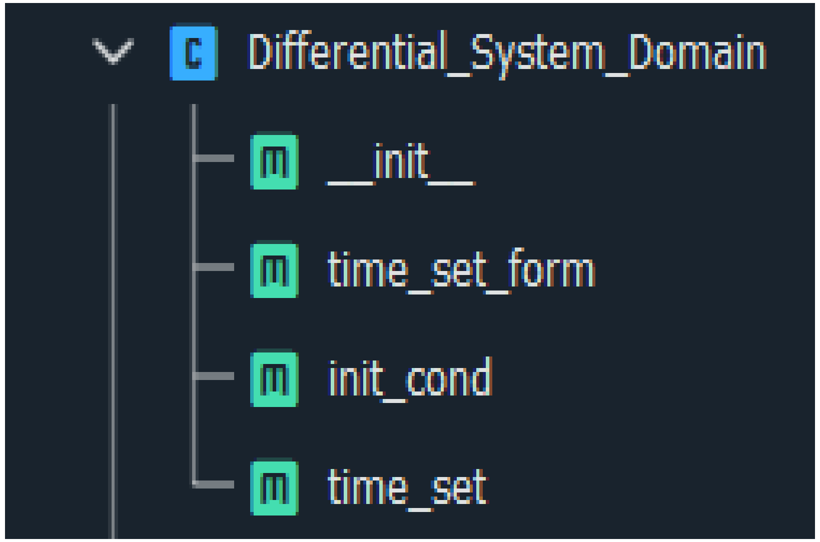
Python Program Architecture – Libraries & Classes

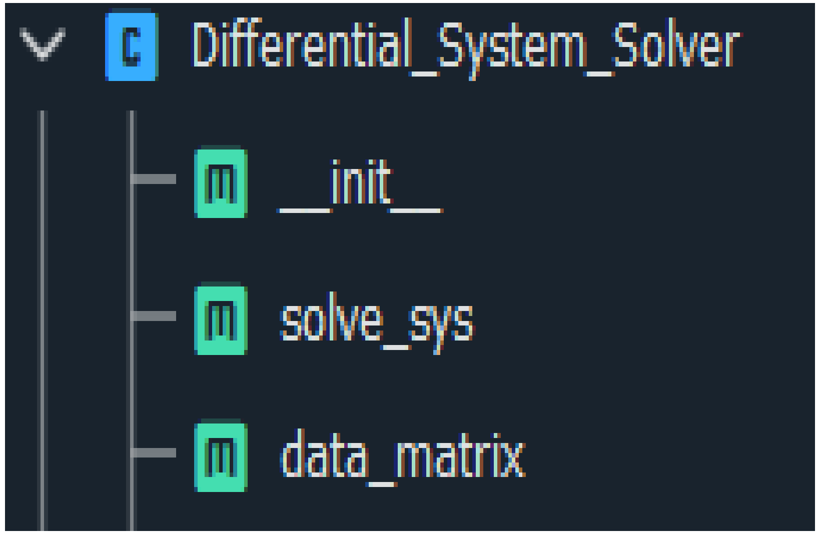
```
#-----------------------------------

import math
import numpy as np
from   scipy.integrate import solve_ivp


import matplotlib.pyplot as plt
import matplotlib.animation as animation
from   collections import deque


#-----------------------------------
```

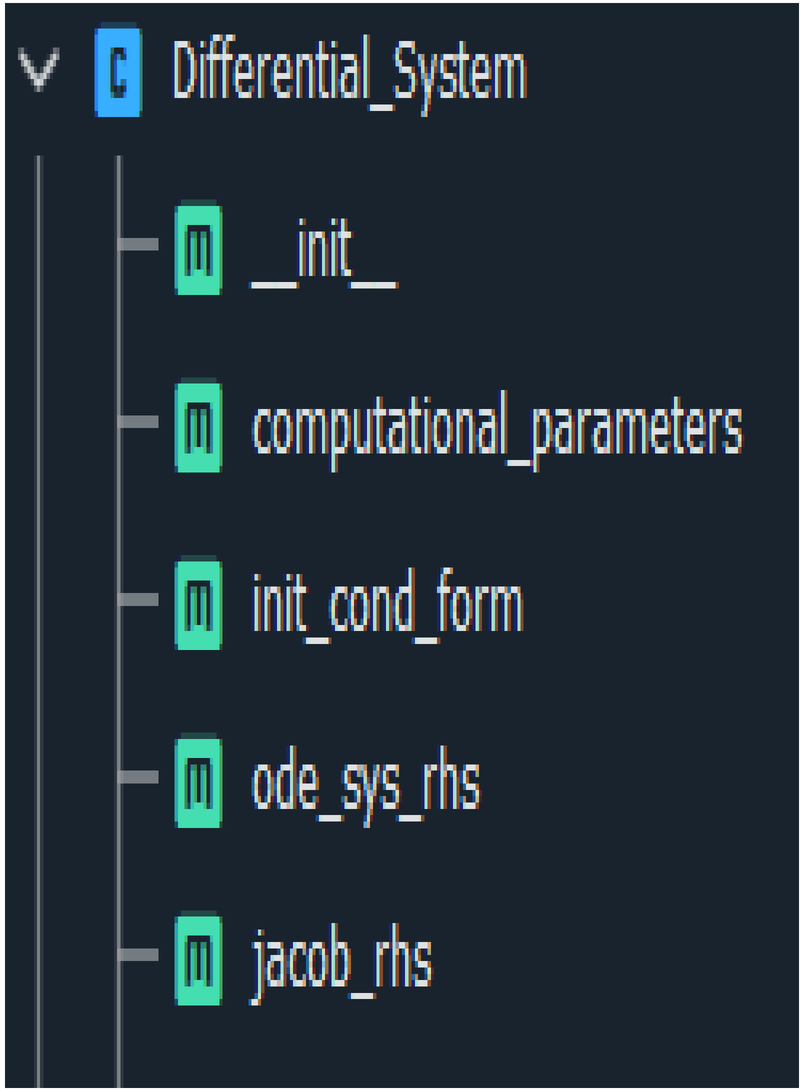
```
def main():
    #-------------------------------------------------
    #     Baseline class objects

    my_diff_sys     = Differential_System()

    my_solution     = Differential_System_Solver()

    my_data_mngmt   = Output_Data_File_Management()

    my_plot         = Plot()
    #-------------------------------------------------
    # object utilization

    my_solution.solve_sys( my_diff_sys )

    my_data_mngmt.set_data( my_solution.data )
    my_data_mngmt.save_matrix( )

    my_plot.set_labels( my_diff_sys.labels )
    my_plot.chart_it( my_solution.data )
    #-------------------------------------------------
    # animation

    coordinates      = Define_Points ( my_solution.cart_coord )
    coordinates.point_coordinates()

    action = Perform_Animation( my_diff_sys.delta_t,
                                my_diff_sys.info, coordinates )
    action.set_up_animation()
    #-------------------------------------------------
```

Program Classes









Python Inheritance

```
#---------------------------------------------------
class Differential_System_Domain( object ):

    #------------------------------------
    def __init__( self ):

        self.labels = [ "Differential System Solution", "time", "values"]

        self.time_set_form()
```

```
#---------------------------------------------------
class Differential_System( Differential_System_Domain ):

    #------------------------------------
    def __init__( self ):

        Differential_System_Domain.__init__( self )

        self.computational_parameters()
        self.init_cond_form()
```

Initial State for International Celestial Reference Frame

```
#--------------------------------------
def init_cond_form( self ):

    #------------------------------------------------------

    self.q_in =  np.zeros(  self.numb_of_eq, dtype = float )
    q_in        = self.q_in

    #------------------------------------------------------
    #-------------------------------------------------------------------
    # set initial conditions

    q_in[ 0 ] =   3.173312775112640E+00
    q_in[ 1 ] =   2.593912393192030E-02
    q_in[ 2 ] =  2.129419344727240E+00
    q_in[ 3 ] =   1.756346981419310E-01
    q_in[ 4 ] =   1.562460620249420E+00
    q_in[ 5 ] =   -3.718348831566720E-03
```

NASA Horizons System

Horizons Web Application

Save/Load Settings...

1 Ephemeris Type: Vector Table

2 Edit Target Body: **Jupiter**

3 Edit Coordinate Center: **Sun (body center)** [500@10]

4 Edit Time Specification: Start=**2026-6-9 10:26:00** TDB , Stop=**2026-6-10 10:26:00**, Step=**1** (days)

5 Edit Table Settings: *defaults*

Differential System Class Method

```
#----------------------------------------
def ode_sys_rhs( self, t, q ):  # implement the differential equations in this function
                               # parameters may be specified through the self.param[] dictionary

    #-----------------------------------------------------
    dq_dt = np.zeros(  self.numb_of_eq, dtype = float )

    par   = self.param
    #-----------------------------------------------------
    #-----------------------------------------------------


    GM      = par[ "GM" ]


    #-----------------------------------------------------
```

Differential Equations Reflected in Python

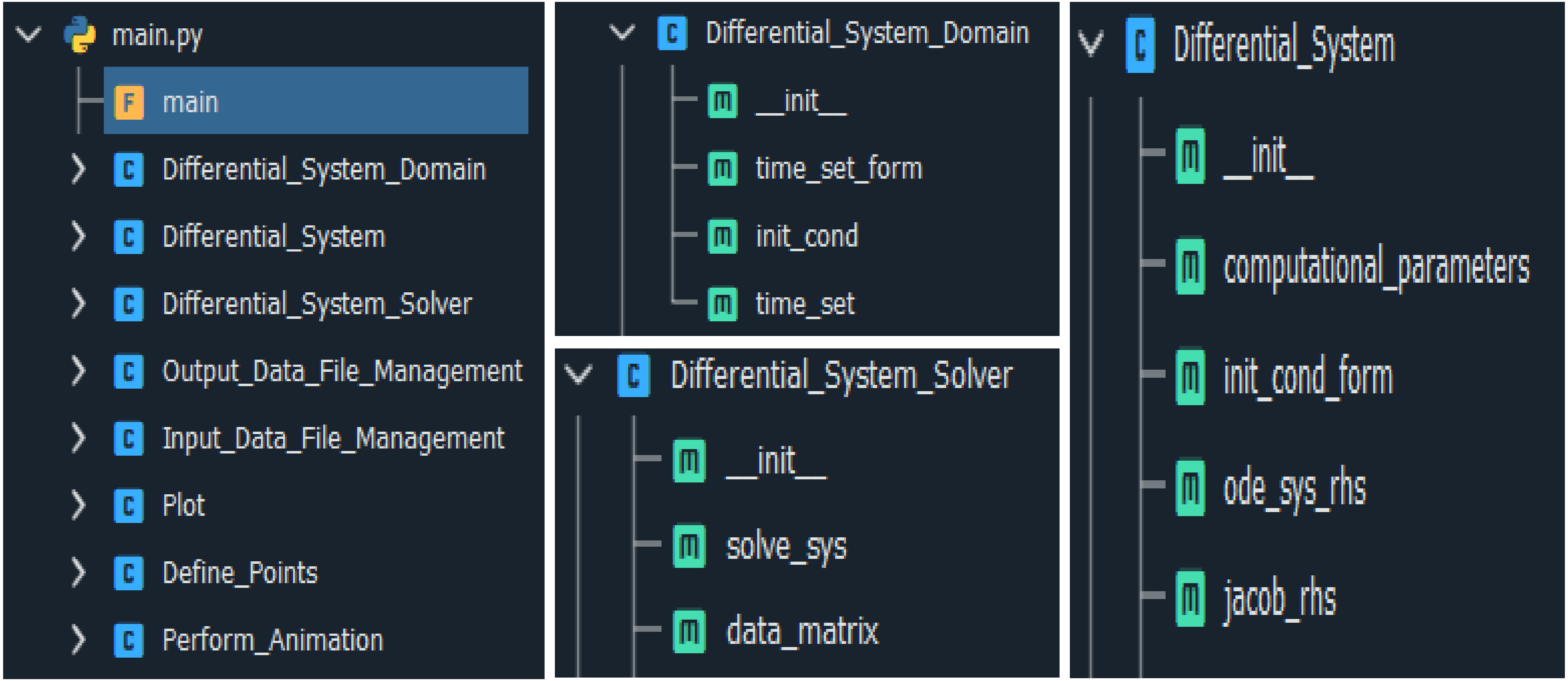
$$
\begin{cases}
\dfrac{dr}{dt} &= v_r \\
\dfrac{dv_r}{dt} &= r\omega_\theta^2 + r\omega_\phi^2 \sin^2\theta - \dfrac{GM}{r^2} \\
\dfrac{d\phi}{dt} &= \omega_\phi \\
\dfrac{d\omega_\phi}{dt} &= -2\dfrac{v_r\omega_\phi}{r} - 2\omega_\phi\omega_\theta\dfrac{\cos\theta}{\sin\theta} \\
\dfrac{d\theta}{dt} &= \omega_\theta \\
\dfrac{d\omega_\theta}{dt} &= -2\dfrac{v_r\omega_\theta}{r} + \omega_\phi^2 \sin\theta\cos\theta
\end{cases}
\qquad
\begin{cases}
q[0] &= r \\
q[1] &= v_r \\
q[2] &= \phi \\
q[3] &= \omega_\phi \\
q[4] &= \theta \\
q[5] &= \omega_\theta
\end{cases}
$$

```
dq_dt[ 0 ] = q[ 1 ]

dq_dt[ 1 ] = q[ 0 ] * ( q[ 3 ]  * math.sin( q[ 4 ] ) )**2.0 - GM / q[ 0 ]**2.0 + q[ 0 ] * q[ 5 ]**2.0

dq_dt[ 2 ] = q[ 3 ]

dq_dt[ 3 ] = -2.0 * q[ 1 ] * q[ 3 ] / q[ 0 ] - 2.0 * q[ 3 ] * q[ 5 ] * math.cos( q[ 4 ] ) / math.sin( q[ 4 ] )

dq_dt[ 4 ] = q[ 5 ]

dq_dt[ 5 ] = -2.0 * q[ 1 ] * q[ 5 ] / q[ 0 ] + q[ 3 ] * q[ 3 ]  * math.cos( q[ 4 ] ) * math.sin( q[ 4 ] )
```

Jacobian Class Method

```
def jacob_rhs( self, t, q ): # implement the differential-equation jacobian in this function
                             # parameters may be specified through the self.param[] array
    #------------------------------------------------------
    jac_mtrx = np.zeros( ( self.numb_of_eq, self.numb_of_eq ), dtype = float )
    par = self.param
    #------------------------------------------------------
    #------------------------------------------------------

    GM        = par[ "GM" ]

    #------------------------------------------------------

    # dq_dt[ 0 ] = q[ 1 ]

    jac_mtrx[ 0 ][ 0 ] =   0.0
    jac_mtrx[ 0 ][ 1 ] =   1.0
    jac_mtrx[ 0 ][ 2 ] =   0.0
    jac_mtrx[ 0 ][ 3 ] =   0.0
    jac_mtrx[ 0 ][ 4 ] =   0.0
    jac_mtrx[ 0 ][ 5 ] =   0.0

    # dq_dt[ 1 ] = q[ 0 ] * ( q[ 3 ]  * math.sin( q[ 4 ] ) )**2 - GM / q[ 0 ]**2.0 + q[ 0 ] * q[ 5 ]**2.0

    jac_mtrx[ 1 ][ 0 ] =   ( q[ 3 ]  * math.sin( q[ 4 ] ) )**2.0 + 2.0 * GM / q[ 0 ]**3.0 + q[ 5 ]**2.0
    jac_mtrx[ 1 ][ 1 ] =   0.0
    jac_mtrx[ 1 ][ 2 ] =   0.0
    jac_mtrx[ 1 ][ 3 ] =   2.0 * q[ 0 ] * q[ 3 ] * math.sin( q[ 4 ] )**2.0
    jac_mtrx[ 1 ][ 4 ] =   2.0 * q[ 0 ] * ( q[ 3 ]**2.0 ) * math.sin( q[ 4 ] ) * math.cos( q[ 4 ] )
    jac_mtrx[ 1 ][ 5 ] =   2.0 * q[ 0 ] * q[ 5 ]
```

Program Classes



WASHINGTON STATE
UNIVERSITY

Welcomes You
Begrüßt Sie

Wita Cię
Vous Accueille

SciPy Differential Solver Class

```
#-------------------------------------------------------
class Differential_System_Solver( object ):

    #----------------------------------------
    def __init__( self ):

        self.comp_sol = []

    #----------------------------------------
    #----------------------------------------
    def solve_sys( self, diff_sys ):

        self.diff_sys = diff_sys

        time_range, time = diff_sys.time_set()

        q_in = diff_sys.init_cond()

        print('...working...')
        self.comp_sol = solve_ivp( diff_sys.ode_sys, time_range, q_in,
                                  method = 'Radau', t_eval = time, dense_output = True,
                                  atol = 1.0e-13, rtol = 1.0e-13, jac = diff_sys.jacob )


        self.data_matrix()
```

Summary

- Solar System and Initial-Value System
- Astrodynamics System – N-Body Problem
- Python Program Architecture – Libraries & Classes
- Python Inheritance
- Initial State for International Celestial Reference Frame
- Differential System & Jacobian Class Methods
- SciPy Differential Solver Class