HOWTO: Alfano Utilities and Controls

# Introduction

The Alfano package is used with the Goddard Mission Analysis Tool (GMAT) to implement continuous low-thrust orbit transfer trajectory simulations.

The Alfano package contains Python code to generate yaw control values governing the optimum thrust angles by orbit, using the Alfano direct optimal method, Figure 1.

Chart

Description automatically generated

Figure , Low Thrust Optimal Circle-to-Circle Orbit Transfer

In this figure the control variable corresponds to a series of yaw angles over the range of orbit ratio 1 – 10. The costate values are negative and correspond to the lambda i coefficient in Alfano’s direct optimization. The given costates will only reach the full orbit ratio of 10 in the values between approximately -0.6 and -0.2. A Geosynchronous orbit corresponds to an orbit ratio of 6.31.

GMAT is used with a python transcription of the Alfano phi function to assess the performance of costates with regard to time of flight and fuel efficiency. The figure shows that costates that are more negative are more aggressive with regard to inclination, thus there is a limit to how much inclination change this trajectory algorithm can produce.

The Alfano Transfer technique originated in an engineering note by Salvatore Alfano and William Weisel published in 1985 in the AIAA Journal of Guidance and Control as “Optimal Many-Revolution Orbit Transfer” (doi: 10.2514/3.19952).

The primary module for use with GMAT is the YawAngles.py procedure found in the Alfano/controls subpackage. As a usage example, the following GMAT mission script calls YawAngles.py in line 71.

1. %----------------------------------------
2. %---------- Arrays, Variables, Strings
3. %----------------------------------------
4. Create Array CONTROL[1,3];
5. Create Variable SMA\_INIT ORBIT\_R AOL REV REV\_LAST REV\_ERR T0\_AT\_REV T\_REV COSTATE CURRINCL;
6. Create Variable MORE;
7. GMAT CONTROL(1, 1) = 1;
8. GMAT SMA\_INIT = 6878.1366;
9. GMAT ORBIT\_R = 1;
10. GMAT AOL = 0;
11. GMAT REV = 0;
12. GMAT REV\_LAST = 5070;
13. GMAT REV\_ERR = 89;
14. GMAT T0\_AT\_REV = 0;
15. GMAT T\_REV = 0;
16. GMAT COSTATE = -0.52;
17. GMAT CURRINCL = 28.5;
18. GMAT MORE = -1;
19. %----------------------------------------
20. %---------- Mission Sequence
21. %----------------------------------------
22. BeginMissionSequence;
23. %BeginFiniteBurn 'BeginFiniteBurnRaiseAltitude' DefaultFB(EOTV);
24. %While EOTV.SMA < 11670
25. % GMAT EOTV.HET1.ThrustDirection1 = 1;
26. % GMAT EOTV.HET1.ThrustDirection2 = 0;
27. % GMAT EOTV.HET1.ThrustDirection3 = 0;
28. % Propagate 'Propagate Steps' DefaultProp(EOTV);
29. %EndFiniteBurn 'EndFiniteBurnRaiseAltitude' DefaultFB(EOTV);
30. Propagate 'Propagate to periapsis' TrimPropagator(EOTV) {EOTV.Earth.Periapsis};
31. BeginFiniteBurn 'BeginAlfanoBurn' ContinuousThrust(EOTV);
32. % If the trajectory defined by the costate is over-aggressive, the EOTV will arrive at inclination
33. % at lower altitude than desired.
34. % If the trajectory is not aggressive enough, the EOTV will arrive at inclination at higher altitude.
35. GMAT SMA\_INIT = EOTV.SMA;
36. GMAT REV\_LAST = EOTV.Earth.OrbitPeriod;
37. GMAT T0\_AT\_REV = EOTV.ElapsedSecs;
38. GMAT REV = REV + 1;
39. %While EOTV.SMA < 42159
40. While CURRINCL > 0.1
41. % The costate value "blends" inclination change with altitude change, such that the resultant trajectory
42. % should have arrived at the desired altitude when the inclination change is accomplished.
43. % Precision of the inclination result prevents consistent stopping condition.
44. % TODO - apply a differential correction to the costate and iterate.
45. GMAT AOL = EOTV.TA + EOTV.AOP;
46. % AOP is measured relative to the ascending node, TA is measured relative to the AOP.
47. % This works as long as the target plane for inclination change is the equatorial plane.
48. If AOL > 360.0
49. GMAT AOL = AOL - 360.0;
50. EndIf;
51. GMAT T\_REV = EOTV.ElapsedSecs - T0\_AT\_REV;
52. If T\_REV >= REV\_LAST
53. % T\_REV is elapsed time, REV\_LAST is earth orbit period at beginning of rev.
54. GMAT REV\_ERR = T\_REV - REV\_LAST;
55. GMAT REV\_LAST = EOTV.Earth.OrbitPeriod;
56. GMAT T0\_AT\_REV = EOTV.ElapsedSecs - REV\_ERR;
57. % TODO, better to interpolate EOTV.Earth.OrbitPeriod
58. % Write for debug:
59. %Write REV\_ERR { Style = Script, LogFile = true, MessageWindow = true }
60. GMAT REV = REV + 1;
61. EndIf;
62. If EOTV.SMA > 42159.5
63. % End the loop
64. GMAT CURRINCL = 0;
65. % GMAT EOTV.HET1.ThrustDirection1 = 1;
66. % GMAT EOTV.HET1.ThrustDirection2 = 0;
67. % GMAT EOTV.HET1.ThrustDirection3 = 0;
68. Write EOTV.Earth.SMA { Style = Script, LogFile = true, MessageWindow = true }
69. Write EOTV.INC { Style = Script, LogFile = true, MessageWindow = true }
70. Else
71. GMAT [CONTROL] = Python.YawAngles.get\_control\_onrev(COSTATE, AOL, EOTV.Earth.SMA, SMA\_INIT, MORE);
72. GMAT EOTV.HET1.ThrustDirection1 = CONTROL(1,1);
73. GMAT EOTV.HET1.ThrustDirection2 = CONTROL(1,2);
74. GMAT EOTV.HET1.ThrustDirection3 = CONTROL(1,3);
75. Propagate 'Propagate Steps' DefaultProp(EOTV);
    1. GMAT CURRINCL = EOTV.INC;
76. EndIf;
77. EndWhile;
78. Write REV { Style = Script, LogFile = true, MessageWindow = true }
79. EndFiniteBurn 'BeginAlfanoBurn' ContinuousThrust(EOTV);

YawAngles.py depends on the AlfanoLib.py from the utilities package. AlfanoLib.py implements the Alfano optimal control equations.

Rather than compute the trajectory functions in real time, GenerateControlTable.py is provided in the controls subpackage and uses AlfanoLib.py to generate a JSON file corresponding to Figure 1. YawAngles.py reads the created JSON file to iteratively provide the optimal low-thrust yaw angles to GMAT for each revolution of an Alfano circle-to-circle orbit raising maneuver.

Other Python procedures located in the utilities folder support research and validation of the output and are not used with GMAT.

# Building Alfano

The top-level Alfano folder is the folder above the source folder (alfano) and contains setup.py. Edit the setup.py file to update the version tag. Check the \_\_init\_\_.py version information and package requirements in the subfolders (controls and utilities) that they are consistent.

Run a command window as administrator.

Execute the source distribution build command in the top-level Alfano folder (above the src folder).

python setup.py sdist --formats tar

This will produce a tar file in the dist folder (Winzip is pissy about the gz format, so we use tar).

To deploy Alfano upack the tar file under /dist.

cd to dist/alfano-<version>, where version matches the version in the top level setup.py.

Execute the setup command from the dist/alfano-<version> folder.

python setup.py install

# Install Controls to GMAT

## Prerequisites

Once the Alfano project has been built, open the distribution folder (e.g., “Alfano/dist/alfano-<version>”) and copy the “alfano” folder to the python environment Lib/site-packages folder. Open this alfano folder and copy the contents of the “dist/alfano-<version>/alfano/controls” folder to “~/AppData/Local/GMAT/GMAT/userfunctions/python”.

It is important that the version of the control files copied to “GMAT/userfunctions/python” should match the version of AlfanoLib.py in “alfano/site-packages/utilities”. By always copying the controls files from the “site-packages/alfano/utilities” package, this constraint will be enforced.

## Generate Controls.json

Controls.json contains the computed Alfano costate values associated with vectors of the control variable using key:value dictionary format, ordered by increments of orbit ratio. The control variable (cv) is used to compute the yaw thrust vector as a function of spacecraft Argument of Latitude. This file MUST BE NEWLY GENERATED for each new build of AlfanoLib. If it is not rebuilt, the dictionary key from the file will not match the computation of costates during the initialization of YawAngles.py. GMAT will silently crash.

Once the controls are copied into the “GMAT/userfunctions/python folder”, the Controls.json file is generated with the GenerateControlTable.py”.

GenerateControlTable.py will also output an Excel workbook containing the same data as the JSON file, this is easier to read and can be used to identify the costate values.

PyQT file dialog is used to allow the user to place these files to a chosen path. It is necessary for YawAngles.py to find the exact JSON file that GenerateControlTables.py writes. The workbook is only used for engineering review. In order to ensure that YawAngles.py can find the controls file, an interface agreement between the two python procedures is that the path string which describes the location of the controls file is written to a file named “SavedJsonPath” in the current working directory.

To emphasize, it is necessary for YawAngles.py to read its control data from the file created by GenerateControlTable.py using the exact same build of AlfanoLib between them. GenerateControlTable.py must be executed anytime AlfanoLib is updated.

Under Windows 10, If the control table does not match the AlfanoLib computation, the signature of this problem can be seen by executing YawAngles. from the python terminal or within an IDE.

1. Traceback (most recent call last):
2. File "c:\Users\<user>\AppData\Local\GMAT\GMAT\userfunctions\python\YawAngles.py", line 266, in read\_controlfile
3. U = np.array(dct[str(l)])
4. KeyError: '-0.3666'
5. During handling of the above exception, another exception occurred:
6. Traceback (most recent call last):
7. File "c:\Users\<user>\\AppData\Local\GMAT\GMAT\userfunctions\python\YawAngles.py", line 343, in <module>
8. read\_controlfile(r'.\Controls.json')
9. File "c:\Users\<user>\\AppData\Local\GMAT\GMAT\userfunctions\python\YawAngles.py", line 271, in read\_controlfile
10. raise RuntimeError('Exception loading UbyRbyL dictionary: {0} for costate {1}.'.format(e.\_\_doc\_\_), l)
11. IndexError: tuple index out of range