Space Shuttle Rendezvous Targeting in Lua

by indy91

December 19, 2012

1 Introduction

Did you ever want to do the rendezvous with the ISS like the pros? Now you can.
This is (almost) the same program the astronauts used in the space shuttle for the onboard targeting. This is for advanced and expert users! As my LSTS launch autopilot this was primary made for myself, so please excuse any user unfriendly commands/functions.
The script consists of two parts:

- planning the necessary burns and corrections to rendezvous with another vehicle (by default the ISS)
- executing a RCS or an OMS-burn

2 Requirements

There are two scenarios, but I am not 100% sure which addons you will need to use the ISS by Thorton, so there is another scenario with the default ISS.
At least you need:

1. Attitude MFD
2. Shuttle Fleet V4.8
3. STS-120 Payload: http://orbithangar.com/download.php?ID=4267
4. (International Space Station v.3.2 by Thorton)

3 Function description

3.1 target

You can change the target by typing:

```
target = vessel.get_interface('ISS')
```

with your intended vessel.

3.2 `dv = orbit_tgt(hv,dt,offset,t_burn,[elev])`

Calculates the required velocity change.

Parameters

- hv (handle): the handle for your vessel. Normally this should be vif.
- dt (number): the Transfer time to the target [min].
- offset (vector): the offset vector in a LVC (Local Vertical Curvilinear) frame [kft]. offset=vec.set(-2,0,1) would target for 2000 feet behind and 1000 feet under the target.
- t_burn (number): the time of the burn in simulation time [s].
- elev (number): If you want the burn to occur at a special elevation angle to the target, you can specify the angle here [deg].

Return values

- dv (vector): The velocity change you have to do at t_burn to reach the desired orbit [m/s].
- Also the total Δv of the burn in feet per second is shown on the terminal.

3.3 `burn(hv,dv,t_burn)`

Executes a RCS burn.

Parameters

- hv (handle): the handle for your vessel. Normally this should be vif.
- dv (vector): This is the Δv vector you get from the orbit_tgt function or specify yourself [m/s].
- t_burn (number): the time of the burn in simulation time [s].

Notes This function will show messages how long until the burn at TIG-5 and TIG-1 minute. At TIG-30 seconds the automated burn procedure will start and wait for the right time to execute the burn. You should have Attitude MFD or any other attitude hold program off, because the RCS burn does not work when it has not sole control over the vehicle.

3.4 `oms_burn(hv,dv,t_burn)`

Executes an OMS burn.

Parameters

- hv (handle): the handle for your vessel. Normally this should be vif.
- dv (vector): This is the Δv vector you get from the orbit_tgt function or specify yourself [m/s].
- t_burn (number): the time of the burn in simulation time [s].

Notes The function works just as the normal burn function. You have to use the numbers given for the pitch and yaw for the maneuver e.g. in Attitude MFD.

3.5 `autopilot()`

An autopilot is included in the script, which uses the functions and procedures from the tutorial. You can start the autopilot by typing: autopilot(). Follow the instructions on the MFD.
Please note, that the autopilot will not work for every scenario or mission. If you want to be more flexible and realistic, go through the tutorial and learn how to do the rendezvous calculations and burns by yourself. I find it much more rewarding to recreate the rendezvous as near to the reality as possible.

4 Tutorial

4.1 NCC-burn

Start the Scenario 'STS-120 before NCC' open your Terminal MFD and start the script by typing:

```
run('Rendezvous')
```

To plan a burn you need the transfer time, offset target and time of the burn. The first burn you have to do in the scenario is the NCC-burn (For the rendezvous profile look at pages 1-5/6 of reference [1]). Reference data for the NCC-burn in an 210NM orbit per STS-120 Rendezvous Targeting Data (page 6-4 or 116):

```
T1: -0/00:57:42 (basetime: TI)
EL: 0
DT 57.7 (in minutes)
DX: -48.6 (in kft)
DY: 0
DZ: 1.2
```

So this might be confusing, but its not too hard. You can see that the NCC-burn is executed at a specific time before the TI-burn. We can assume the TI-burn will be at the time of your next apoapsis, so you can type:

```
t_TI=oapi.get_simtime()+apt(vif)
```

This sets the time for the TI-burn in simulation time. apt is a function to get the time to the next apoapsis and vif is a parameter for your vehicle.

```
t_NCC=t_TI-57*60-42
```

Now you have the time of the burn. The target of the burn is always the ISS or an offset target to the ISS. In this case the numbers mean -48600 feet behind the ISS and 1200 feet below it. You have to give the targeting function an offset vector so you type:

```
offset=vec.set(-48.6,0,1.2)
```

DT is the transfer time to the target (or offset). You can easily type:

```
t=57.7
```

Now you have everything you need. To get the delta velocity vector you write:

```
dv=orbit_tgt(vif,t,offset,t_NCC)
```

The output vector will be saved in dv and this vector will also be shown in the MFD. Additionally there will be the total dv in feet/second. If this number is higher than 6 fps the space shuttle would do an OMS-burn, a number lower than 6 a RCS-burn. In this case the number should say something around 2.77 fps. If not, something went wrong and look again at the numbers you typed in. Now you probably have some time left so type:

```
term.out(t_NCC)
```

to see when the burn will be executed. 5 minutes/300 seconds before the burn you can do another calculation of the dv (just use

```
dv=orbit_tgt(vif,t,offset,t_NCC)
```

again) but with Nonspherical gravity sources off it should be accurate enough. Now you have to execute a RCS burn. First I have to say, that there is no attitude control in the script. So you need something like Attitude MFD to hold the attitude. In the velocity mode set your Pitch, Yaw and Roll to 0. To execute the NCC use the function 'burn':

```
burn(vif,dv,t_NCC)
```

You will see the messages '5 minutes to burn' and '1 minute to burn' appear on the Terminal MFD. Now NOTICE: It is very important that you deactivate the attitude hold BEFORE the burn begins, because with it on the RCS burn will not work. You can of course leave the attitude hold on until shortly before the burn begins. At T-30 seconds the 'burn autopilot' is armed. You should now deactivate the attitude hold and watch a beautiful, short burn from the outside.

4.2 TI-burn

So NCC is done, what next? You already told the computer the time for the next burn, the TI-burn. The data for the TI-burn in the reference data are:

```
T1: 0/00:00:00 (basetime: TI)
EL: 0
DT: 76.9
DX: -0.9
DY: 0
DZ: 1.8
```

That means you have to change some variables:

```
t=76.9
offset.x=-0.9 (you could also set a new offset vector)
offset.z=1.8
```

That was all you have to change for the TI-burn. Now you can calculate a first dv for the burn:

```
dv=orbit_tgt(vif,t,offset,t_TI)
```

The total dv should be just under 10 fps. That means you have to do an OMS burn. By typing:

```
term.out(t_TI)
```

you see, that you have enough time to prepare for the burn. Don't forget to activate the attitude hold again and you can also leave it on during the OMS burn. Again ca. 5 minutes before the TI-burn you can do another dv calculation. Now start the OMS-Burn mode:

```
oms_burn(vif,dv,t_TI)
```

Now you see two new numbers. These represent the direction of the burn. The first number is the required yaw, the second the required pitch (I normally get around yaw=+12, pitch=-11). You should use this numbers in the Attitude MFD. You don't have to use the exact numbers, two or three digits after the comma are enough. The burn works the same way as the RCS burns, only that you can leave on the attitude hold. After the burn you should return to the normal attitude (0,0,0).

4.3 MC1-burn

20 minutes after the TI there is the first midcourse correction (MC1):

```
T1: 0/00:20:00 (basetime: TI)
EL: 0
DT: 56.9
DX: -0.9
DY: 0
DZ: 1.8
```

The targeting data is the same, only everything is 20 minutes later. By now you know how to calculate the dv:

```
t=56.9
t_MC1=t_TI+20*60
dv=orbit_tgt(vif,t,offset,t_MC1)
```

If the TI was perfect this would result in a 0 fps burn, but I normally get around 0.04-0.06 fps (without nonspherical gravity). Again the burn is executed by typing:

```
burn(vif,dv,t_MC1)
```

4.4 MC2-burn

MC2 is a little different. The time of ignition is based on the elevation of the ISS:

```
T1: 0/00:49:54 (basetime: TI)
EL: 29.07
DT: 27.0
DX: -0.9
DY: 0
DZ: 1.8
```

The elevation angle is an additional parameter for the orbit_tgt function and the time for MC2 only an initial guess:

```
t=27
t_MC2=t_TI+49*60+54
elev=29.07
dv=orbit_tgt(vif,t,offset,t_MC2,elev)
```

Unfortunately you have to type in the new time for the MC2 by hand:

```
t_MC2=
```

whatever the new burntime is. Again for the burn:

```
burn(vif,dv,t_MC2)
```

4.5 MC3-burn

MC3 is just another midcourse correction, you know how to do that by now, don't you? ;)

```
T1: 0/00:17:00 (basetime: MC2)
EL: 0
DT: 10.0
DX: -0.9
DY: 0
DZ: 1.8

t=10
t_MC3=t_MC2+17*60
dv=orbit_tgt(vif,t,offset,t_MC3)
burn(vif,dv,t_MC3)
```

4.6 MC4-burn

After the MC3-burn you will see the ISS getting bigger and bigger. But you are still not even on a collision course. So for the MC4 you have to change some offset parameters:

```
T1: 0/00:27:00 (basetime: MC2)
EL: 0
DT: 13.0
DX: 0
DY: 0
DZ: 0.6

offset.x=0
offset.z=0.6
t=13
t_MC4=t_MC2+27*60
dv=orbit_tgt(vif,t,offset,t_MC4)
burn(vif,dv,t_MC4)
```

This a rather long burn (ca. 1.6 fps), but you should use the RCS for it. After this last burn you will end up 600 feet directly below the ISS.
If you are experienced with the tools I suggest you try to recreate the STS-39 (or another) mission. Take a look at the special focus chapter in reference [2].

References

[1] STS-120 Rendezvous Flight Data File: http://www.nasa.gov/centers/johnson/pdf/193893main_RNDZ_120_F_1.pdf

[2] History of Space Shuttle Rendezvous: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110023479_2011024697.pdf