Directions to implement tether line control:

1. Tether command. The controller is really just a table with commanded unstretched tether length vs time. In the code, we just interpolate this table at the given timestep and set the unstretched length equal to the appropriate value. Thus we assume the tether length follows our command perfectly.

The tether command is input into the code via a new input file called ReelCommand.THR. Right now I just have this as a 1-column file, first with all the time entries and then with the associated commanded lengths. You can modify this however you want. An example file is attached, which first extends the tether from 30 m to 300 m, and then retracts it again later to 30 m.

To generate this file, I use a Bezier curve profile due to its smoothness and ease of configuration. I have included two MATLAB scripts that will generate this file. The first just generates either a single extension or retraction. The other file generates and extension followed by a retraction. In the create\_bezier\_profile function, the first argument d is a smoothness parameter, and the second argument is the duration of extension and retraction. The first argument must always be smaller than the second argument. Play around with the arguments and watch what happens to the profile. When you run the script a new ReelCommand.THR file is generated.

2. Code modifications. To add this capability to the code, you will need to make a few modifications. First, make time and reel command vectors which hold the reel command from the input file ReelCommand.THR. This can be done by adding the following lines to the Tether Structure:

!!!!!!!!!!!!!!!!! Added for tether controller !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

real\*8 :: TCOM(200) = 0.0 ! Time indices for reel trajectory

real\*8 :: REELTRAJ(200) = 0.0 ! Unstretched length values for reel trajectory

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

You will also need to add a read in function to populate these vectors, say in the tether subroutine:

! Load in the reel in and reel out command trajectories

open(unit=94,file='ReelCommand.THR',status='old',iostat=openflag)

do i = 1,200

read(unit=94,fmt=\*,iostat=readflag) T%THR%TCOM(i)

end do

do i = 1,200

read(unit=94,fmt=\*,iostat=readflag) T%THR%REELTRAJ(i)

end do

close(94)

Ok, now that you have the data you can add the final piece. In the Control subroutine:

!!!!!!!!!!!!!!!!!!!!!!!!!!! SUBROUTINE CONTROL !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Add the following lines, which represent the tether controller. Note that the “controller” is really just a table interpolation, followed by recomputation of the bead properties:

! Tether Line Length Control

if (T%CS%TETHERCONTROLOFFON .eq. 1) then

! Extension according to precomputed Bezier curve profile

lencommand = 0.0

if (T%SIM%TIME .le. T%THR%TCOM(1)) then

lencommand = T%THR%REELTRAJ(1)

else if (T%SIM%TIME .ge. T%THR%TCOM(200)) then

lencommand = T%THR%REELTRAJ(200)

else

do i=2,200

if ((T%SIM%TIME .ge. T%THR%TCOM(i-1)) .and. (T%SIM%TIME .le. T%THR%TCOM(i))) then

lencommand = T%THR%REELTRAJ(i-1) + ((T%SIM%TIME-T%THR%TCOM(i-1))/(T%THR%TCOM(i)-T%THR%TCOM(i-1)))\*(T%THR%REELTRAJ(i)-T%THR%REELTRAJ(i-1))

end if

end do

end if

T%THR%LEN = lencommand

! Recompute Bead Properties

T%THR%EMASS = T%THR%MASSPUL\*T%THR%LEN/(T%THR%NBEADS)

T%THR%ELEN = T%THR%LEN/(T%THR%NBEADS+1)

T%THR%EKE = T%THR%KE/T%THR%ELEN

T%THR%EKV = T%THR%KV/T%THR%ELEN

T%THR%ECV = T%THR%CV/T%THR%ELEN

T%THR%SIGMA = (T%THR%EKV+T%THR%EKE)/T%THR%ECV

T%THR%KU = T%THR%EKE\*T%THR%EKV/T%THR%ECV

T%THR%CU = T%THR%EKE

end if

Note that I have hardcoded in a maximum vector length for TCOM and REELTRAJ of 200 elements. This can easily be changed by modifying all the appropriate variables of course.

I think that is it! I have attached an example TAPAS.f90 file with all of this implemented in case you need a reference. Of course it is an old version of the code now.