Thoughts on Controller

Notes by Rick Damiani of RRD Engineering LLC, April 2020

Introduction

The following is a compilation of unedited notes from Rick Damiani of his work debugging the Makani M600 crosswind controller implemented for use in KiteFAST-OS. This controller is comprised of code taken from the Makani M600 crosswind controller and modified to work with KiteFAST-OS. Implementing this code into KiteFAST proved to be significantly more complicated than anticipated. Makani engaged RRD Engineering to fix bugs in KiteFAST itself as well as in the controller. While the contract ended before the controller was fully operational in KiteFAST, all major bugs in both kiteFAST and the controller were fixed, and many improvements were made in the controller implementation. In addition, Rick developed an extremely useful post-processing data visualization tool (Jupyter notebook) that enables quick assessment of output from KiteFAST.

This document is essentially a log of progress cataloguing learnings and experience of the first attempt to implement a controller into KiteFAST, and may be useful for anyone interested in running KiteFAST-OS, especially if the interest includes modeling the Makani M600 energy kite.

Neither Makani nor RRD Engineering makes any claim about the accuracy of the information provided – it is simply presented as is – a work in progress.

Log

11/20/2019

Symptoms: The controller seems to be responsible for driving the Kite to the ground in KiteFAST

- I need to replicate the problem myself, running with controller and onshore
- I would like to run a straight and level flight, but it's not possible to run CSIM as straight & level for comparison
- I would like to remove the effect of the tether. I think Moordyn is creating quite a few problems, and it should be removed from the investigation for a while. CSIM can be run w/o a tether using tether release for an unpowered glide landing
- I am not convinced that the signs for ailerons and rudder/elevator must be reversed. Please let us fix this at first.
- It seems like even the aerotorque was way off at t=0 in STI's sims: see page 66. How so?
- Is there a way to run KiteFAST with the control settings from CSIM in an imposed fashion? i.e., pass the control-setting to KiteFAST from a dummy controller that passes the control-settings from a time-series file

SO I have worked on getting the laptop ready for post-processing and I have also worked on getting it and myself ready for CSIM running as well.

- I have run a Landbased case, with prescribed loop in KFAST. It ran. The circle position was already available.
- **Step 0.** I now want to create a case out of CSIM where I can run a loop starting at one specific time, running it for 30 seconds, or 1 loop, and then transfer that to KFAST, and see what happens there.
    - **Step A.** Then in KFAST the idea is to first run a prescribed trajectory as given by CSIM, extract tether forces and the other variables that would be passed to CSIM controller

- **Step B.** Then do a check on actuator disk, still in this case perhaps
- **Step C**. Then unconstrain the trajectory and activate controller fully

So first run csim by saving states and then rerunning from those states for ~30 seconds.

ALso, prepare the jupyter notebook to read out of csim the variables of interest. I think those are the ones passed to the controller (i.e. the estimated states by standalone csim).

Come back here after completing that step:

Constant wind speed=12 m/s --- No shear/veer --- (in CSIM, Parker Ranch wind direction changed to 0deg and air density, rho=1.225, to be consistent with the Landbased KiteFAST case)

Under makani type:

>run_sim --nowith_online_turbsim_databases -S -M M -t 600 --save_state_time 480 -l

when done, this has saved **controller_0_state_list.bin** ;the states at t=480s; then run for 40 s (i.e. till 520s)

>run_sim --nowith_online_turbsim_databases -S -M M -t 520 -l --load_state controller_0_state_list.bin

11/21/2019

⇒(11/21/2019) **I have completed step 0**, and now I am working on getting a similar KFAST run with prescribed trajectory. In the notebook, KFASTvsCSIM.ipynb, I have the script to get both initial conditions and the position file to be read in by MBDyn. I have a bit of a concern, as it may have issues with the Euler angles in the position-file due to jumps.

Well, I had all sorts of issues with Euler angles, and Rotational vectors coming out of scipy.Rotation.

I questioned my handling of matrices and all, and in the end, the problem was that **Rotation assumes the DCM to describe a rotation from Frame1 to Frame2 as Local-to-global, and not vice versa**!!! It cost me 2 days. Also I learned the following:

1. Rotation wants capital 'XYZ' for intrinsic rotations, i.e. those happening about each of the axes as they get updated, not the fixed global. Even though I would have used 'xyz', and did for a while.

2. No need to worry about 'zyx', always do 'xyz' for Euler, though in upper case, because the rot matrix is what we get from csim, and the Euler angles come from that and not vice versa..

3. My thinking of using Rotation was good all along , there is no magic or massaging to do, it works fine if you pass it the right matrix. However, the RotVecs may incur jumps, for some reason. This happens because the rotvec stays within [0,pi] in magnitude, so when it feels it goes over, (eps+pi) it changes direction (flips) and uses the magnitude =2pi-(eps+pi)=pi-eps. So I have made a simple algorithm that in 2 lines finds the flips and flips back the vectors adjusting magnitude to be: pi+(pi-eps).

3. The rotation vector is a vector whose components give you an actual axis of rotation, and the norm is the amplitude of the rotation (single) to go from Frame1 to Frame2 (still have to pass L2G to Rotation to get this). Note the rotation vector is also called Euler vector by MBdyn, but has nothing to do with Euler angles, even though, technically, the component units are rad.

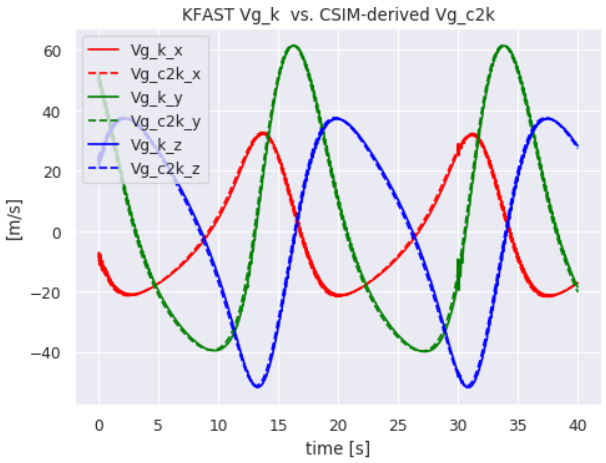
4. For KiteMain.set, just put in there the same vector (or the Euler angles) rather than the DCM, and so you have a 1-to-1 way of making sure at t=0 things are consistent between file driver (which has the rotation vector) and the initialization of mip_rf

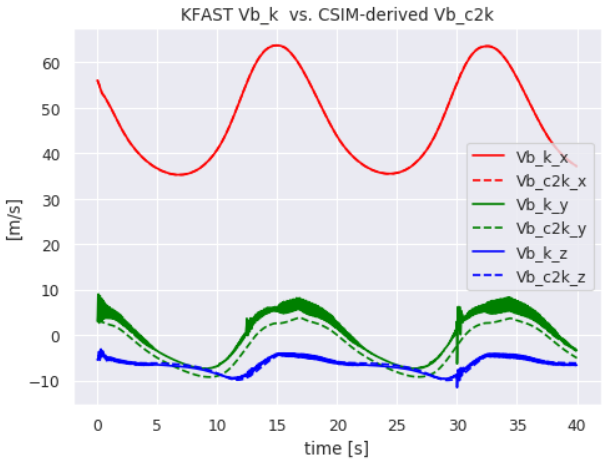
4. May have more problems with libz and blender: go here and follow the very first answer, and then point with a symlink to liz1.2.9 https://stackoverflow.com/questions/48306849/lib-x86-64-linux-gnu-libz-so-1-version-zlib-1-2-9-not-found/51317504

⇒(11/21/2019) **I have completed step A**, So I am building the infrastructure to read output from FAST, including variables to controller.

Findings

1. The apparent_wind in CSIM is apparent_wind_b, so in body coord sys, whereas KiteFAST is passing ground-based ones. Is STI's controller manipulating that appropriately? If that variable is only passed to then calculate alpha & beta, which are calculated correctly, then it is ok, but if the apparent wind is used in other ways, that may not be the case. In any case, the apparent wind from KFAST **should be passed in body ref frame instead**. Otherwise we are asking the controller to do the manipulation. So these questions are for Makani. Are these the real inputs you need for the controller? see below as well
2. As above, the accel, is passed both in body ref frame and in ground ref frame. Why? I have checked the Ag, Ab, and acc_norm, which seem to be dependent on each other, so why have them all?
3. The calculated alpha and beta I calculated with arctan, as arctan2 was 180 deg off
4. Whereas the ground-based velocities are a match, the Vb are not, even though the Euler angles are a match. So what is going on here? We need to understand why Vb and then alpha/beta are different. The two could be related. Yes they are, if I use the Vb_c_y instead of the k version, sideslip goes right on top of the CSIM one. So there is an issue with Vb_k_y. I could try dcm_fromatt_k *Vg_k and see what I get that way. Why do I get noise in the Vb and not in the Vg? Why is the VB_y off a bit, when the Vg are right on top of each other?





5. Need to understand units for rotor speeds, and whether the rotor_speeds I am getting from rotor sensor are in line with KFAST (the latter uses rad/s). It looks like units are fine, but kfast shows weird rotor_speeds.
6. The m**issing variable from KFAST is the generator torque,** I could potentially get it from MBDYN, as a reaction moment at the correct node but it is a long stretch. So, I would instrument the controller (KiteFastController.f90) to have an input and output file. The input file name, for the time being will be hardwired, otherwise we would need to change the preprocessor, and then the user element module as

well. Instead, we just want to modify the controller to have inputs/outputs. Inputs could be initial conditions, outputs are all the returned variables from C-controller.

7. Talking to Tobin: it looks like the flaps field contains all surface deflections (6 wing surfaces, 2 rud/ele) order should be port to starboard along positive y!!

   It looks to me as if this was not right in KiteFASTController.f90 line 3868, and I should verify/rectify that. I now have changed the mapping completely and recompiled.

8. The controller (says Tobin) does not know about the rotor accel, which I agree upon, however, I am not sure we have a model that calculates rotor accel, from KiteFastController.f90 I see **rtraccel** should come from **o%rtracc** which comes from external controller. I need to check **rotorload** calc routine. Looking at the KiteFASTMBD_Plan_OS.pdf, it looks like the rotor accel is not calculated within KFAST, but it must come from the outside. **So, we expect that rotor_speed and rotor_accel come from the controller**. **Confirmed**.
9. Tobin suggests checking the commanded rotor speeds to see whether they are more similar to what STI is passing into KFAST. So I will try and do that.
10. One thing I would like STI to do is to have a Ctrl_Init that takes the initial conditions from an external file, I can pass those initial conditions via KFAST similar to what is done done on KFC_STEP

12/12/2019 Prepare for Meeting

1. Work on notebook to see whether I remember all the main points out of it
2. gather questions for STI
3. Complete Step B that has to do with Actuator Disk. For this, I think it would be great to see how to get the **skew angle** out and see whether I can compare to CSIM. The AeroTorque is in KAD, and I am already plotting it, however, it would be great to have it available (as writeout) in KiteFastController too.

12/16/2019:

Modifications to code after Meeting

My findings were confirmed, so mapping was erratic, and we need to stick to CSIM order convention.

I went ahead and modified kitefastcontroller.f90 to take in more input data from the input file. These are then passed to controller_init at initialization and then passed back. However, I need to understand why the other inputs to the controller are not passed at init, i.e. Xg, dcm etc. They should be, there should not be guesses. The only "guesses" could be the actual rtrspeed, accel, bladepitch, gentorque, etc, but these are now read in from a separate input file. The kite states, however, should come straight from KiteFast.

I proposed to eliminate controller_init on the kfc.c &co., and stick to the controller_step, and at t=0.s, we would add the init phase. Doing so, the controller will have all of the Kite States (Xg,Vg,Acc, tether etc.) it needed. At Init, only Xg, Vg/Vb are available otherwise. The controller outputs are (as stated above) now input by the user in an input file for KiteFastController.f90.

I have found some weird things, and **fixed now the order of both flaps and rotors in Kfc.c &co.;** in KiteFastController.f90 i have come up with these clever indices to get things mapped:

```
cp=(i-1)*(p%numPylons+j) + (2-i)*(p%nRotors+1-j-p%numPylons)

cs=(i-1)*(p%numPylons-j+1) + (2-i)*(p%nRotors-p%numPylons+j)
```

without the use of ifs, these let me get from i,j to a single ‘c’ index within o%rotxxx.

I also found a bug in KiteAeroDyn where the Chi angle was being calculated. In substance, a “-” sign was missing, and the 180-complementary angle was being returned. This was misleading the calculation of AeroTorque. I fixed that.

Additionally, I **changed the rotor_control.c (Geoff’s model) to return the gen_torque and not the sum of that and aerotorque to KFAST**, this was also a major error. Yet, this rotor generator model should be revised by Makani some more, because I am not sure it is solid enough. The integration may also be a bit too crude, but I am not sure at this point it is a big issue, yet a more solid integrator perhaps, would be better.

As a result of these changes, I get more consistency between KFAST and CSIM: For instance the flaps now look much better, though the elevator is problematic, and I do not know how to explain that.

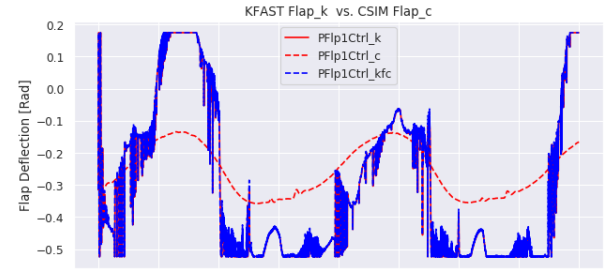
The rel-velocities (vrots) @rotors match pretty well, considering that CSIM has an effect of circulation where the top rotors are ‘accelerated’ and the bottom ones ‘decelerated’.

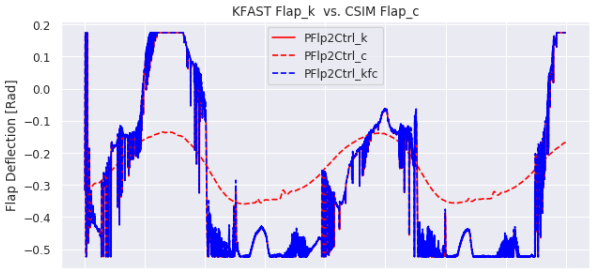
I have asked STI to clean up the code, and merge with my changes, in particular standardize nomenclature, remove the global data, unnecessary and confusing, and then prep-up for init to step change.

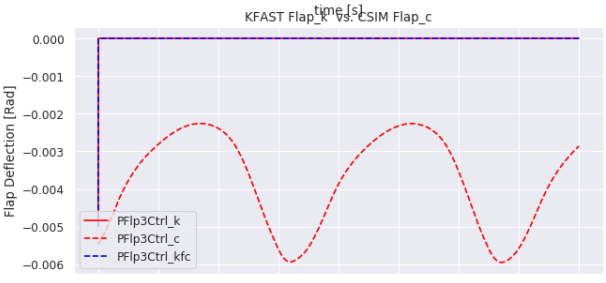
The rotor torques/speeds look better. So a lot of progress, and I would call **STEP B completed**.

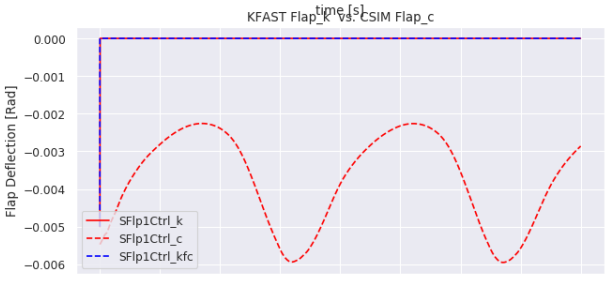
However, we should investigate **two issues still outstanding**:

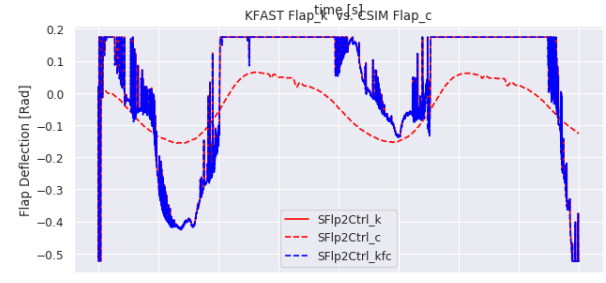
1. the v_y in body coordinates affects sideslip and reverts its sign w.r.t. CSIM;
2. the elevator commanded into K**FAST is way off**. See figures below.

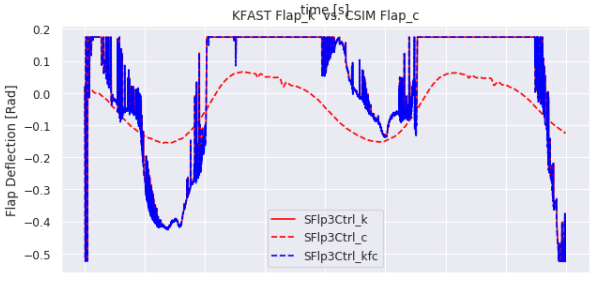

KFAST Flap_k vs. CSIM Flap_c
PFlp1Ctrl_k
PFlp1Ctrl_c
PFlp1Ctrl_kfc
Flap Deflection [Rad]
time [s]

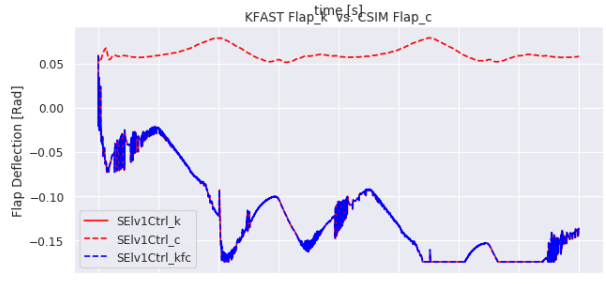

KFAST Flap_k vs. CSIM Flap_c
PFlp2Ctrl_k
PFlp2Ctrl_c
PFlp2Ctrl_kfc
Flap Deflection [Rad]
time [s]

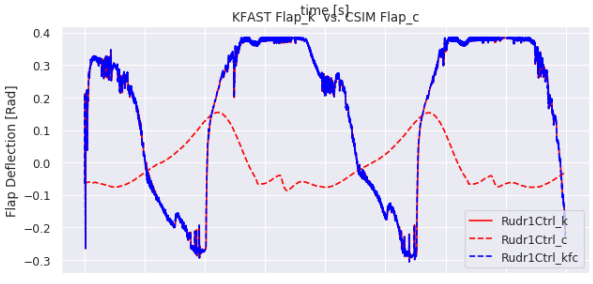

KFAST Flap_k vs. CSIM Flap_c
PFlp3Ctrl_k
PFlp3Ctrl_c
PFlp3Ctrl_kfc
Flap Deflection [Rad]
time [s]

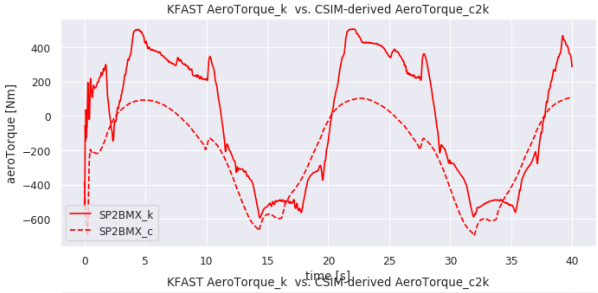

KFAST Flap_k vs. CSIM Flap_c
SFlp1Ctrl_k
SFlp1Ctrl_c
SFlp1Ctrl_kfc
Flap Deflection [Rad]
time [s]

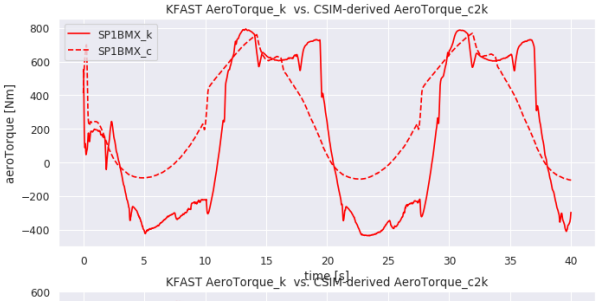

KFAST Flap_k vs. CSIM Flap_c
SFlp2Ctrl_k
SFlp2Ctrl_c
SFlp2Ctrl_kfc
Flap Deflection [Rad]
time [s]

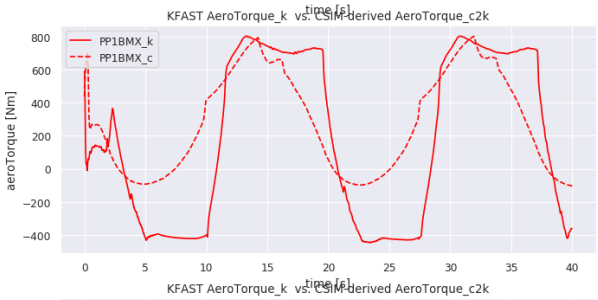

KFAST Flap_k vs. CSIM Flap_c
SFlp3Ctrl_k
SFlp3Ctrl_c
SFlp3Ctrl_kfc
Flap Deflection [Rad]
time [s]

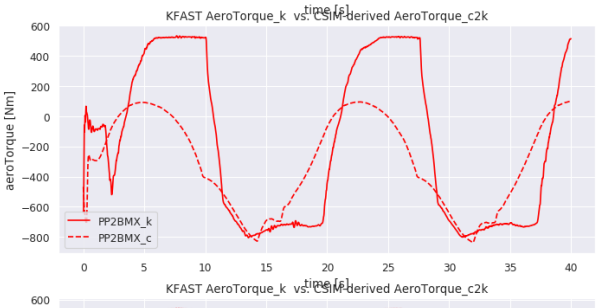

KFAST Flap_k vs. CSIM Flap_c
SElv1Ctrl_k
SElv1Ctrl_c
SElv1Ctrl_kfc
Flap Deflection [Rad]
time [s]

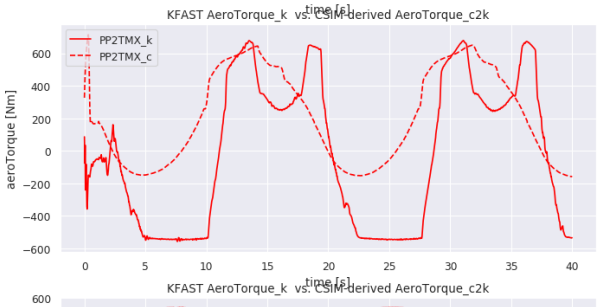

KFAST Flap_k vs. CSIM Flap_c
Rudr1Ctrl_k
Rudr1Ctrl_c
Rudr1Ctrl_kfc
Flap Deflection [Rad]
time [s]

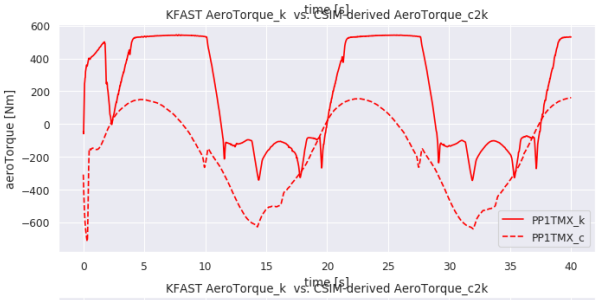

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
SP2BMX_k
SP2BMX_c

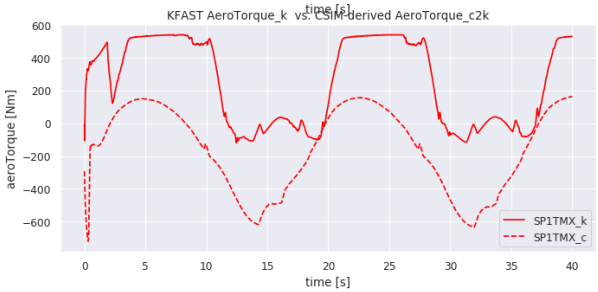

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
SP1BMX_k
SP1BMX_c

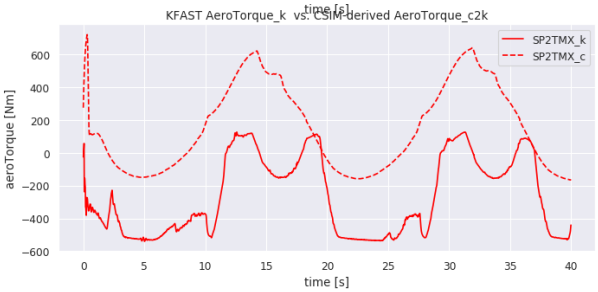

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
PP1BMX_k
PP1BMX_c

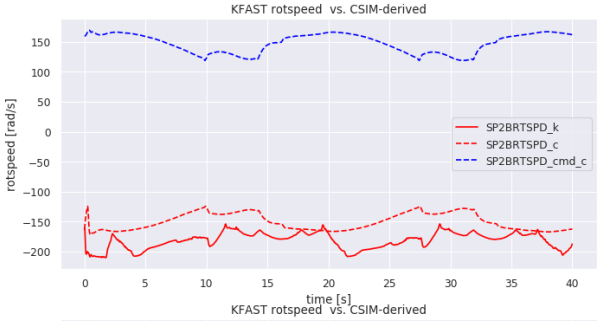

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
PP2BMX_k
PP2BMX_c

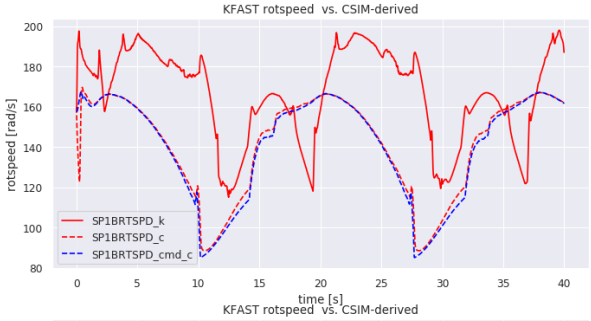

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
PP2TMX_k
PP2TMX_c

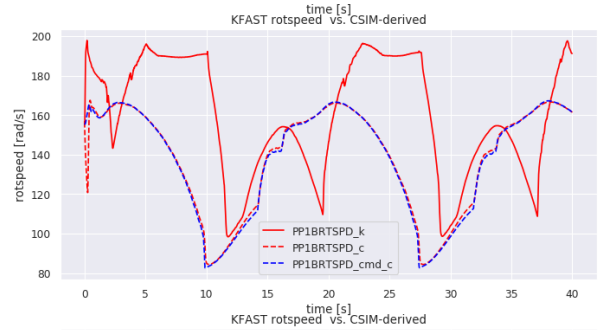

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
PP1TMX_k
PP1TMX_c

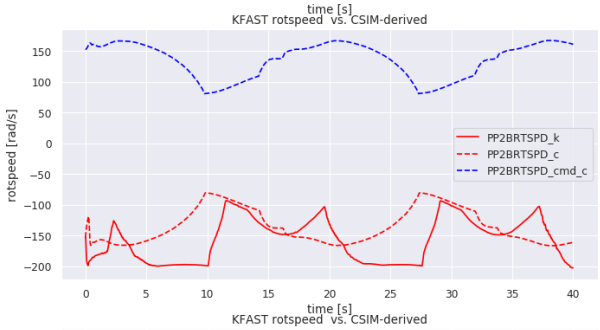

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
SP1TMX_k
SP1TMX_c

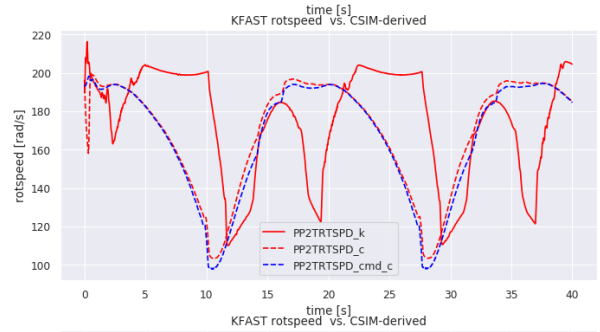

KFAST AeroTorque_k vs. CSIM-derived AeroTorque_c2k
aeroTorque [Nm]
time [s]
SP2TMX_k
SP2TMX_c

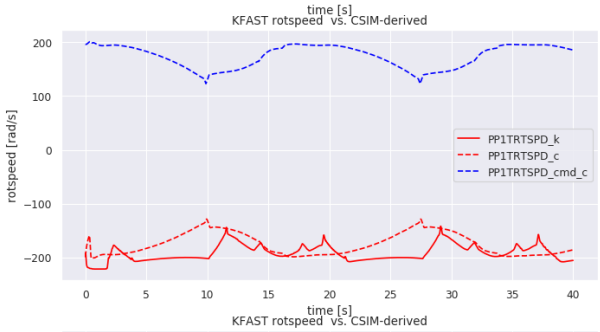

KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
SP2BRTSPD_k
SP2BRTSPD_c
SP2BRTSPD_cmd_c

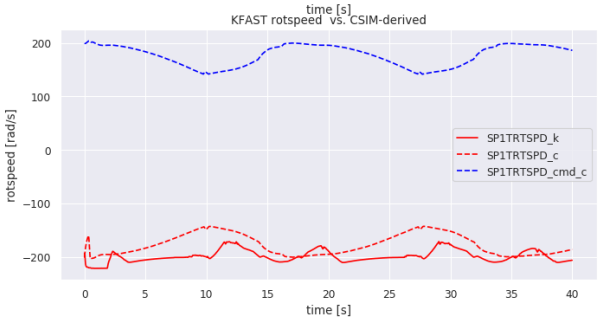

KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
SP1BRTSPD_k
SP1BRTSPD_c
SP1BRTSPD_cmd_c

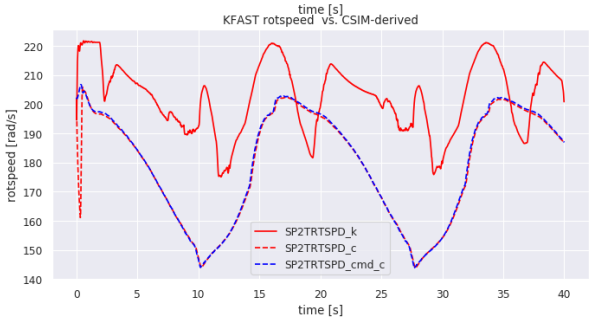

KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
PP1BRTSPD_k
PP1BRTSPD_c
PP1BRTSPD_cmd_c

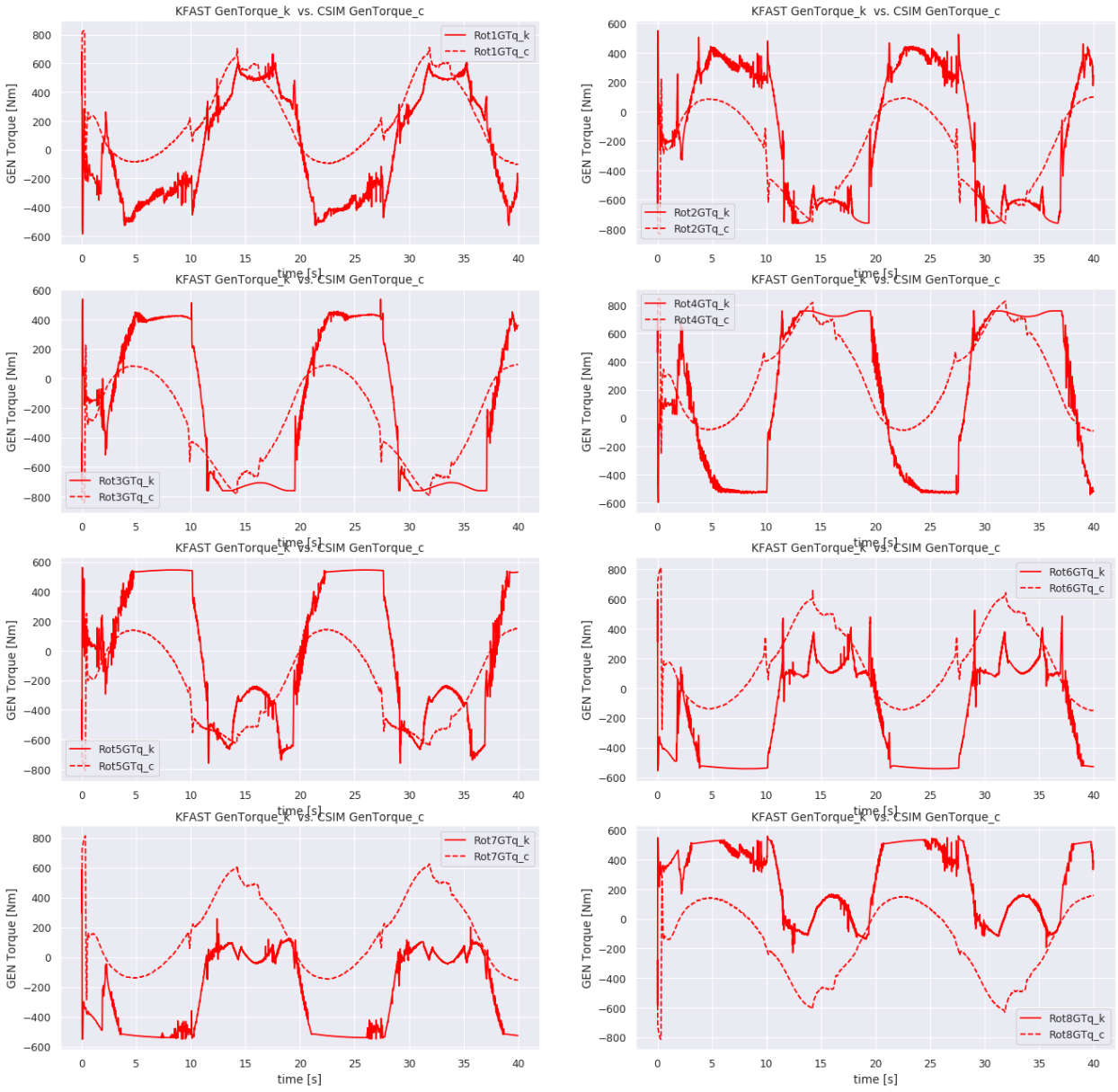

KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
PP2BRTSPD_k
PP2BRTSPD_c
PP2BRTSPD_cmd_c


KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
PP2TRTSPD_k
PP2TRTSPD_c
PP2TRTSPD_cmd_c


KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
PP1TRTSPD_k
PP1TRTSPD_c
PP1TRTSPD_cmd_c


KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
SP1TRTSPD_k
SP1TRTSPD_c
SP1TRTSPD_cmd_c

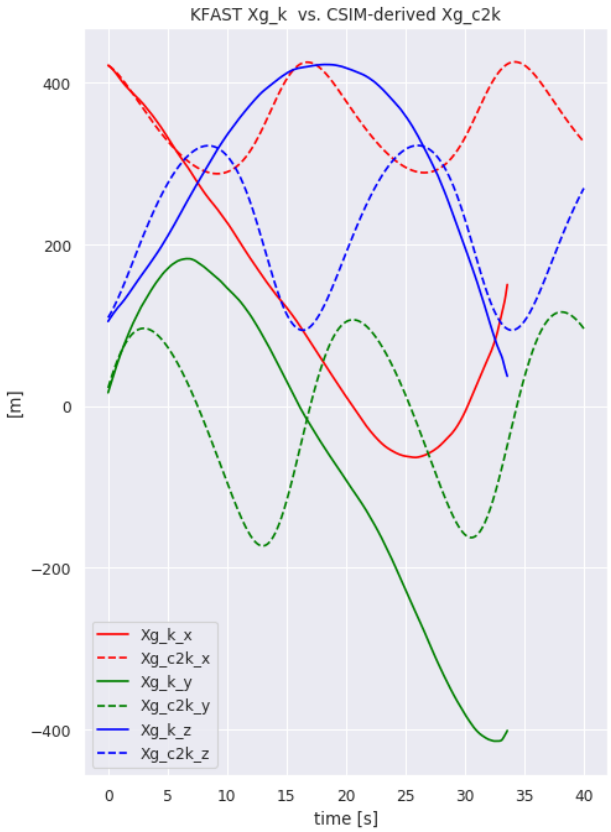

KFAST rotspeed vs. CSIM-derived
rotspeed [rad/s]
time [s]
SP2TRTSPD_k
SP2TRTSPD_c
SP2TRTSPD_cmd_c

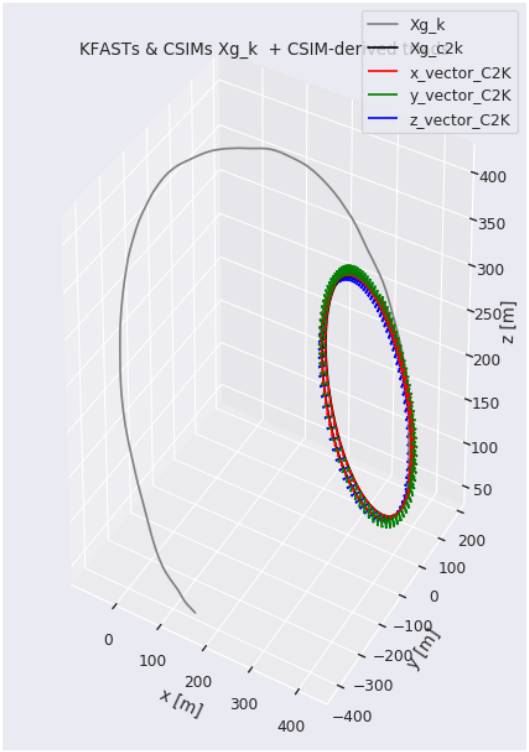


What worries me the most is the elevator. I want to ask @Ruth to check on those airfoil tables to make sure positive is positive etc. No need to create new airfoil tables for now, just checking the current ones. **Airfoil tables were checked and are correct.**

I tried to let the controller take control of the kite in a specific simulation, starting from the usual point:

Then another question surfaced today: as seen in the figure below, the kite is now following a much larger circle. I am not sure this is due to some hardcoded info or a result of the controller still being so off, but this could be an issue. I.e., ar**e the playbook settings changing from CSIM to KFAST**? **this would be pretty major.** Additionally, to note is that the **simulation crashed after some 33s** with the message posted below the figure. So, all in all still not seeing significant progress.





debug - t = 33.560000

dcm_g2b_c = [-0.3744, 0.8509, 0.3685],[-0.1369, -0.4438, 0.8856],[0.9171, 0.2812, 0.2826]

pqr_c = [6.8916, 0.3031, 2.4189]

acc_norm_c = 6.8916

Xg_c = [-150.4350, -401.3999, -37.0579]

Vg_c = [-103.8132, 39.0786, 56.7818]

Vb_c = [85.5955, -89.7108, 12.4046]

Ag_c = [-67.4643, 192.0041, 112.5482]

Ab_c = [102.1942, -110.9645, 176.9953]

rho_c = 1.0260

apparent_wind_c = [91.8132, -39.0786, -56.7818]

tether_force_c = [11258.3906, -386390.5000, 171417.5469]

wind_g_c = [-12.0000, 0.0000, 0.0000]

AeroTorque (csim) = [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 621.8189, -770.2850]

Tether Roll Angle (rad): = [0.8061]

Tether Tension (N): = [422857.3580]

kFlapA_c (kfas frame) = [A1=0.1745, A2=0.1745, A4=0.0000, A5=0.0000, A7=-0.5236, A8=-0.5236, El=-0.1745, Ru=-0.0857]

kFlapA_c (csim frame) = [0.1745, 0.1745, 0.0000, 0.0000, -0.5236, -0.5236, 0.1745, -0.0857]

Gen_Torque (csim frame) = [-577.9652, 530.7135, 383.8497, -460.4379, 392.4433, -401.9729, -451.8109, 493.8967]

Rotor_Speed (csim frame) = [-190.2507, 207.8521, 225.9110, -238.7850, 279.2491, -272.7446, -239.2514, 218.0681]

Rotor_Accel (csim frame) = [-370.4905, 340.2010, 246.0575, -295.1525, 251.5662, -257.6749, 108.9795, -177.1720]

VSM_UpdateStates: Maximum number of Newton iterations of 40 has been reached before convergence.

The residual, 428.58, is larger than the target threshold of, 1.00000E-04. Using solution from

previous timestep.

MD_UpdateStates: NaN state detected.

error status 4: AssRes_Onshore: MD_UpdateStates: NaN state detected.

controller_end

KFC_End errStat - 0 errMsg - controller ending

An error occurred during the execution of MBDyn; aborting...

One step that I wanted to try was to augment KiteFastController.f90 to read in inputs as coming from CSIM, and see what happens to the trajectory. I am not sure it will work, because it is an open-loop approach, but it could tell something?? That requires creating a new input file containing the ctrl outputs from the notebook. Then augmenting the f90 and its input file to read in this new data file. I think it may **be worth trying even just to check that the elevator is doing what one expects for instance**.

This has now been done. The kite flies but it is not great. I will create a new notebook just for the free_sim case.

1/21/2020 Things to Try Next

1. Aero forces and moments, 1st check dcm transform, 2nd remove rotor force/mom components; I cannot remove moments easily because of moment arms, so I am better off leaving that to an update of the output of KiteAD instead **⇒this shows a deficit in Fz or lift, this may be due to the airfoils being so low efficacy? Could try to factor the lift somehow or add a Cl0 somehow to Cl. However, while this is all true, I do not think it will help much, because the kite should try and gain more AOA to compensate for low lift, and it should still make the turn. Instead, it looks like it gives up and straightens out in the free case. ⇒Actually Tobin and M.A. told me that because of limits imposed by the controller, it would not pitch up enough to make up for it, so …⇒the new airfoil put the lift where it should be in z, but the resulting Fx seems so much larger than CSIM**
2. Read in the log from kfc.c as a new file and data to look at path_radius_cmd and k_geom_cmd tether_roll_cmd ⇒this shows very messy cmd dynamics; I believe the controller requires smoother signals coming in

3. Update to latest CSIM set of routines for the controller .c folder?==>**for now, let us try to get the kite to fly with what we have got, in the future we will want to have something that can connect to CSIM**
4. flap frequency response where does it come from? check inputs to controller for high freq, and check freq of flaps **response ~25Hz is what I see**
5. wish list of outputs: KiteVxyz**i**,KiteTAxyzi,(gluecode); KiteFxyz**b**, FM_Aero_**b**, FM_AeroRot_b (airframe+rotor) (KiteAD), DCM_G2B_K and DCM_G2B_C (gluecode/KFASTcontroller)==>
    a. **I have added AirfFxyzi, AirfMxyzi airframe only loads, still in Intertial Ref though. in KAD**.
    b. **KiteTVxyzi,KiteTAxyzi, as well as KiteRVxyzi,KiteRAxyzi, in KiteFAST.f90**
    c. **MIPDCM1-9 components (1,1) (1,2) ..(3,3) of the DCM in KiteFAST**
6. Tobin and Michael were concerned about the constrained case in terms of where those speeds come from. I am not sure how it is done, but I think it is integrated normally, and then the position is fixed on a constraint which is described by the position file. I also noted that there may be a phase issue when plotting things, for instance adding a shift to Vg_k reduced the difference between the velocities: ax.plot(time_k[idx_k],np.roll(Vg_k[:,ii],-7)-Vg_c2k[1:,ii],label='Vg_k_'+coords[ii],color=colors[ii])

   In general, as shown in that cell, the kfast time is 2ms away from that of csim, not too big a deal.

7. I think we should **filter the heck out of any variable coming out of FAST before it gets into the controller**
8. One thing to note in the rotors is that **the rotors do not start in thrust at the same place as in CSIM**. Also RotorSpeeds jump like crazy, and then the other quantities do too. Is it possible that they are the cause of a lot of grief? Michael told me that the motor mixer is trying to get the 'thrust_moment_avail' as far as motor torque and thrust cmd, then it tries to protect the motor if the cmds are unreasonable. In all of this there are effects on the motors where the top and bottom ones are expecting the relative velocity to be offset by this law:

   C_P = 1 - (v / v_freestream)^2 # #

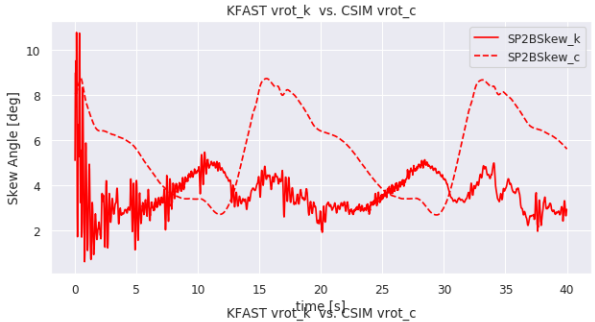
   There is a significant difference in airspeeds between the top # and bottom propellers caused by the lift of the wing. These # pressure coefficients are derived from CFD with the slatted # kite at 4 deg alpha (https://goo.gl/yfkJJS) 'local_pressure_coeff': 0.1448 if r in bottom_row else -0.1501, ⇒I have **hardcoded this correction in KiteAeroDyn**. **I think it is worth checking some more as to why the gentorque is so noisy, and why do we get that huge drop in rotorspeed right from the getgo ⇒Update 1/27/2020 : Talking to Geoff, it sounds like the motor controller may be riding along multiple limits.** TO check this, try and get the cmd_speeds for the rotors and plot against what we have, i.e. the actual (integrated) speeds coming into KFAST**.** Also, the sideslip issue: see if the new CG is part of the problem, so let us reanalyze that, perhaps getting the state_estimate values besides the sim values. If I could get the rest of the telemetry out from the c-controller that would be great, but I am not sure we can, as in missing parts of the c-suite of codes (?)
    a. **Also model must be modified so that we can have the correct shaft tilt. This may require some rework of the preprocessor, and understanding exactly how the nacelle is positioned, there are 2 nodes in the same location, and I do not know why. Still trying to figure it out, but the rotor mass is thrown into KiteFAST and not MBDYN. So the CG from MBDYN will not be accurate, same for mass and inertia. So I suggest that this be modified, but it will take quite a bit of work. The first step may be to correct the rotor and nacelle node alignment. Still unclear to me why there are 2 nodes.**
    b. **I have modified the shaft tilt, but the skew angle got worse (see below), perhaps check that the correction for the velocity, which I have also done, is now done properly and normal to the new rotor plane.**
    c. **The commanded and actual rotor speeds are basically on top of each other, so the commanded speeds are jittery and noisy, due to the noise of the incoming signals, in particular the acceleration.**

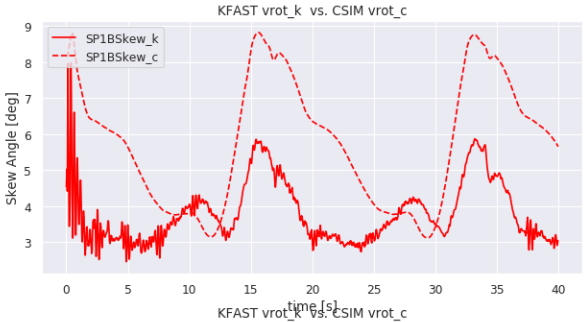
d. **The jitter in the gentorque comes from the acceleration that is super noisy; also because Ab_x is basically a dead signal (constant) the controller will command something really odd. So, let us check what is going on with the ingestion of the position coordinates and orientation into KiteFAST/BDDyn.**

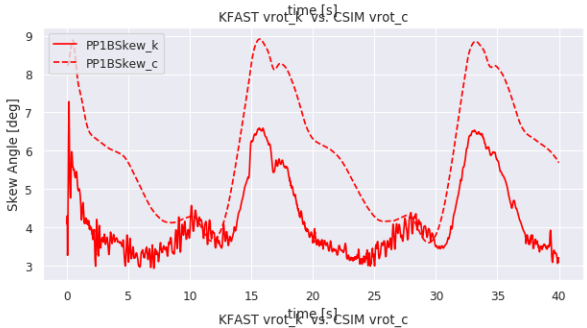
I have tried to run with new airfoil tables, and while lift is much closer (AirfFz), the moment_y is totally off, but that might be due to the reference point used for the moments (though it seems too off). Nonetheless to try and reconcile all the forces to the CM and see what I get. In the free-case, with VSM on, it crashes after a few time-steps with SRGETF problem (singularity), not sure where that comes from, but it might be due to the VSM solver. If I get rid of it, we get to some 10sec, but the kite crashes.

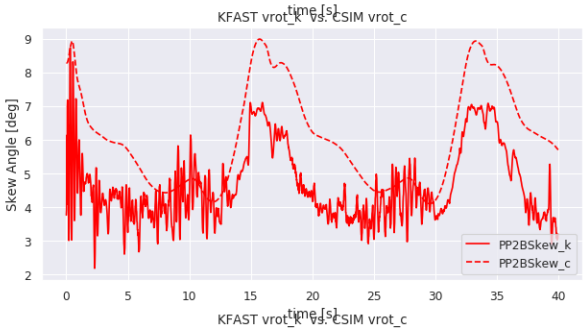
9. I **want to check the m%FusOLoads%Force(:,1) why index 1 all the time? Also why is AirFFx>0? it should not contain any rotor, so check it** ! has to do with the mesh, I think that is ok.
10. Why is the tether force larger ? play with E in case⇒done, the mean is now the same, but it is forcing the solution in a way.
11. Check the LPF in the STI stuff, is it really working? if it does then try and filter it all. ==>I Have played with filtering of pqr_f and Ab_f, and indeed it does smooth things out.

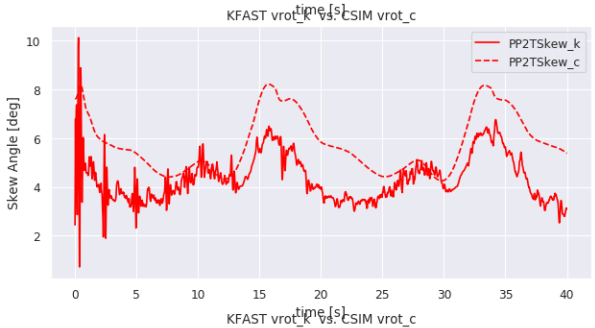
The skew angle below is obtained by setting the RFs in the nacelle and rotor nodes to 3degs pitched down, and same for the vrels at the rotors below:

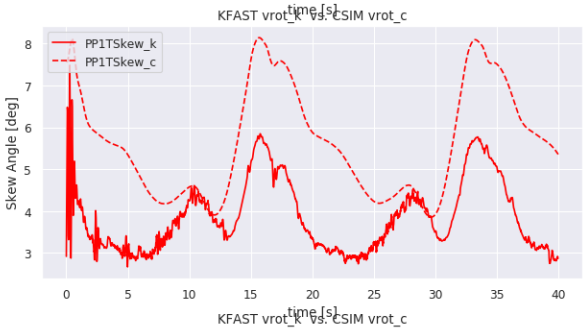


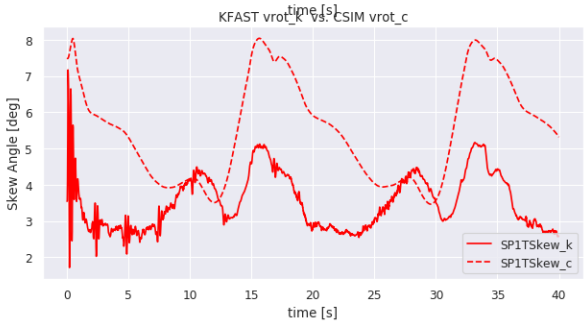


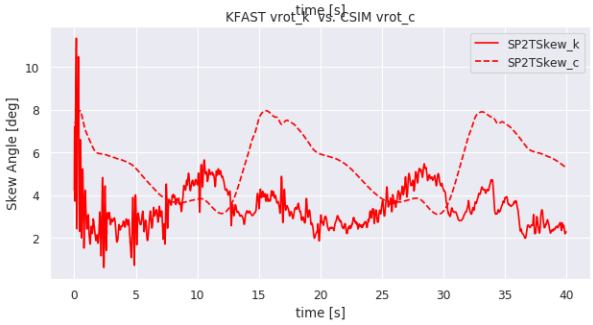




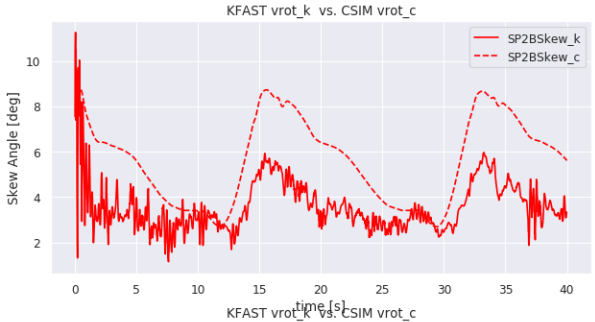


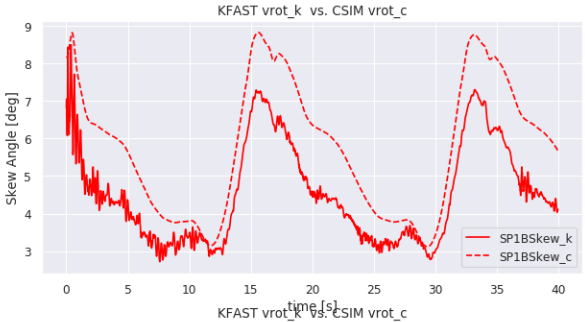


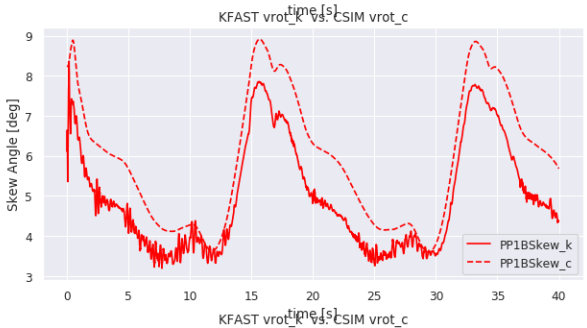


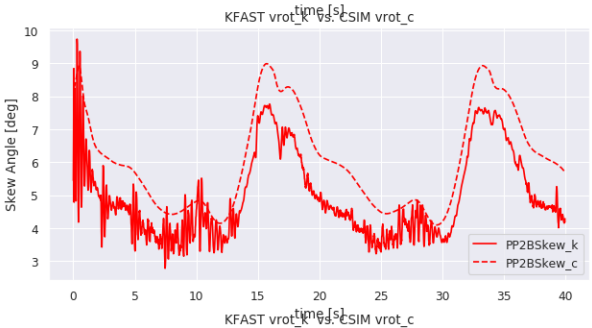


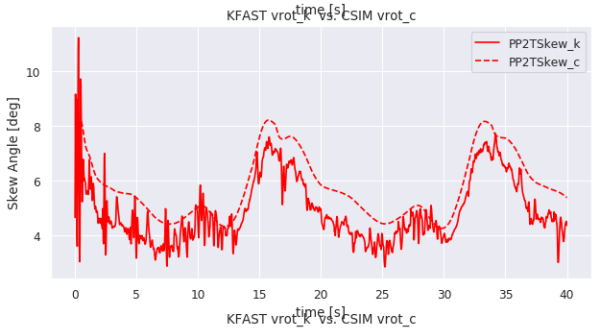
with a straight rotor axis the skew angles are better overall as can be seen below

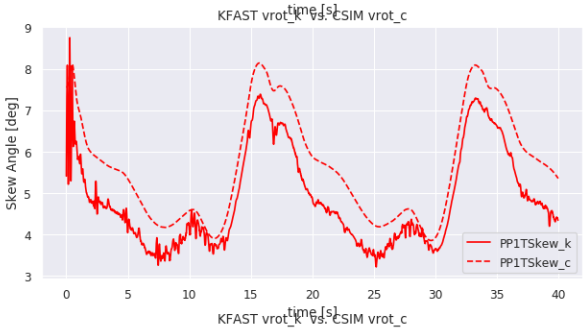


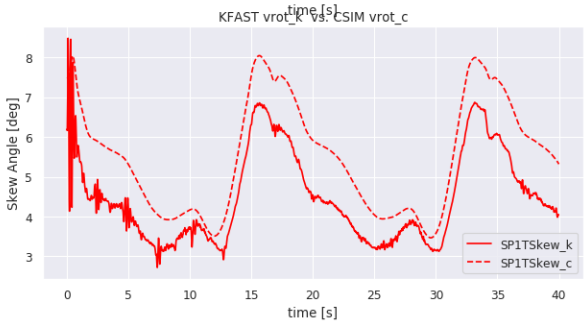


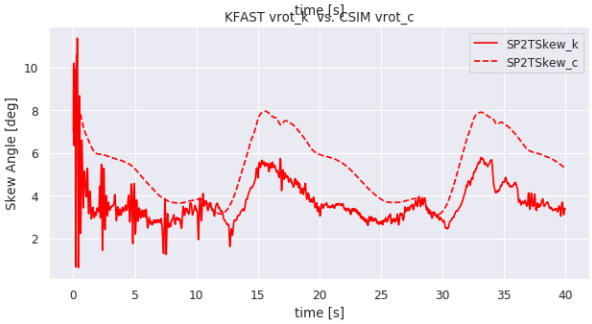




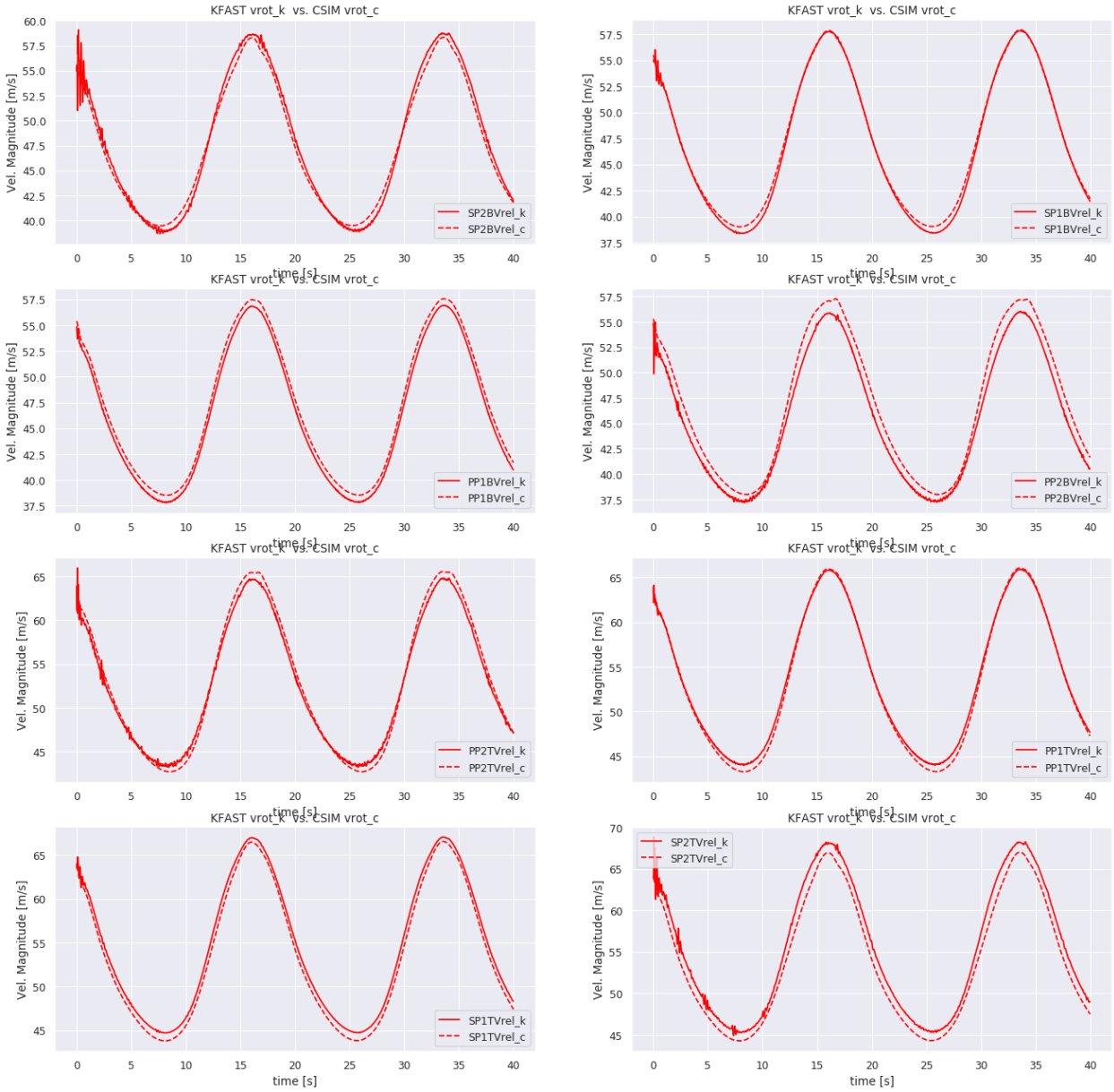




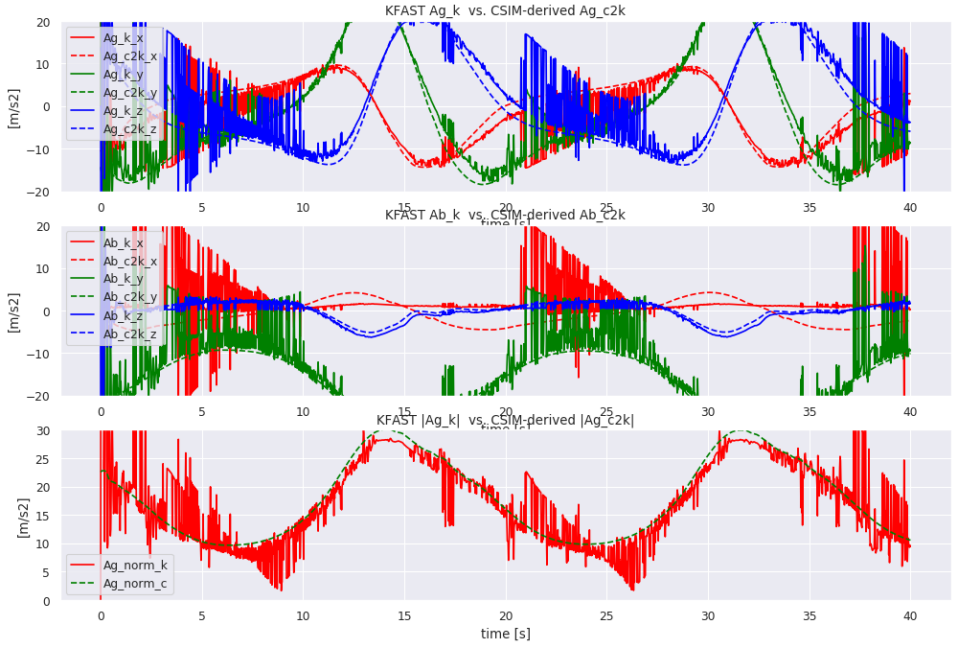




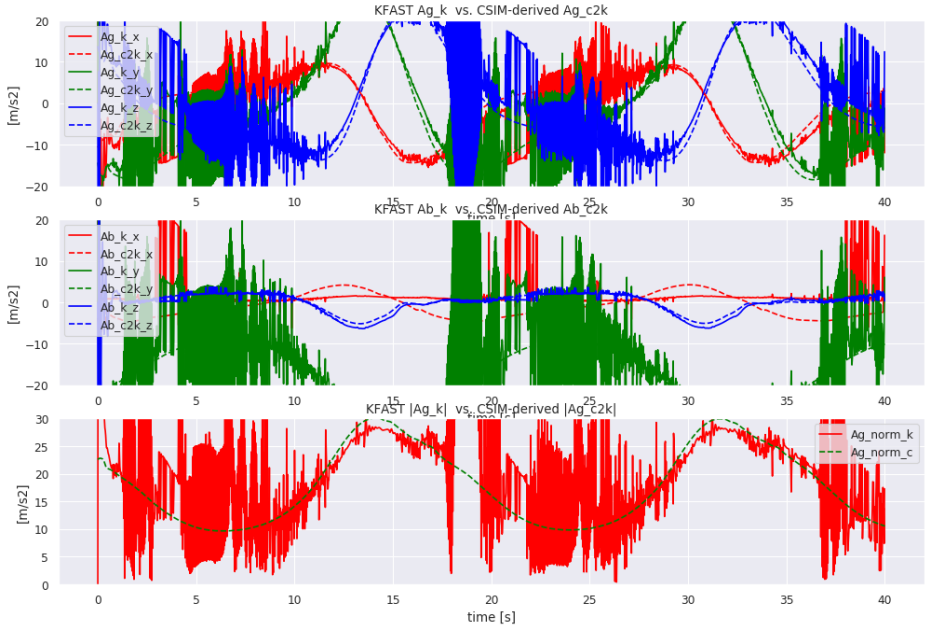
For the velocity magnitudes there is no difference as was to be expected.



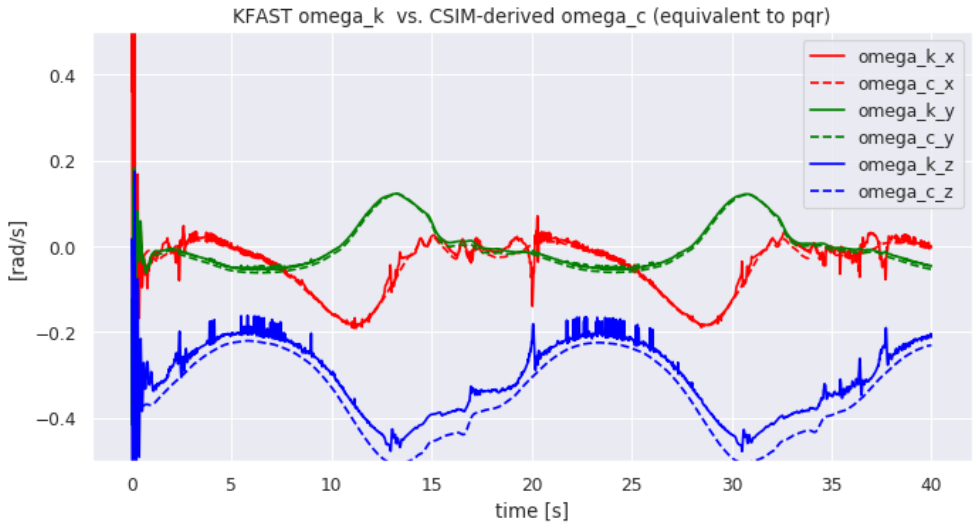
12. So I have f**iltered the heck out of Ab_f**, and things get better, in the sense that the controller starts smoothing out a bit, for instance with a 0.1Hz cut off this is what I get for the Accel:



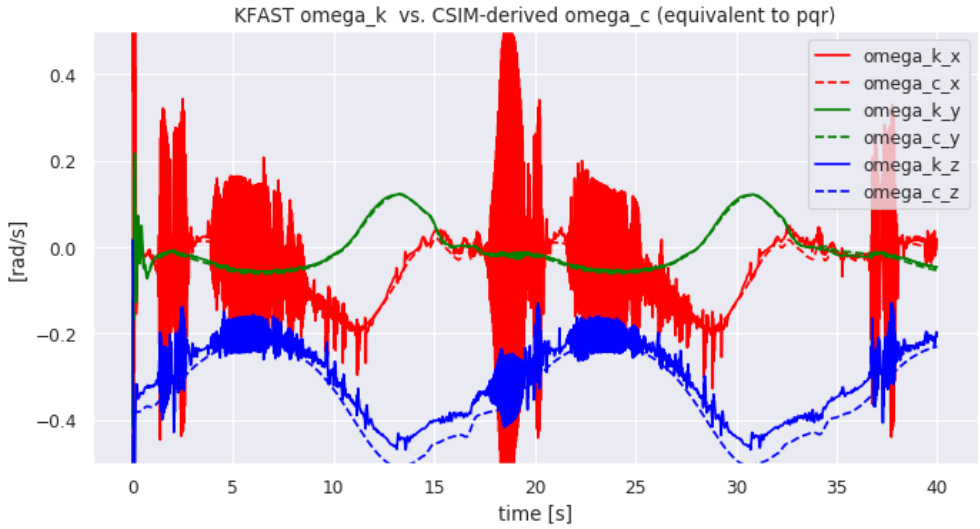
vs. non-smoothed

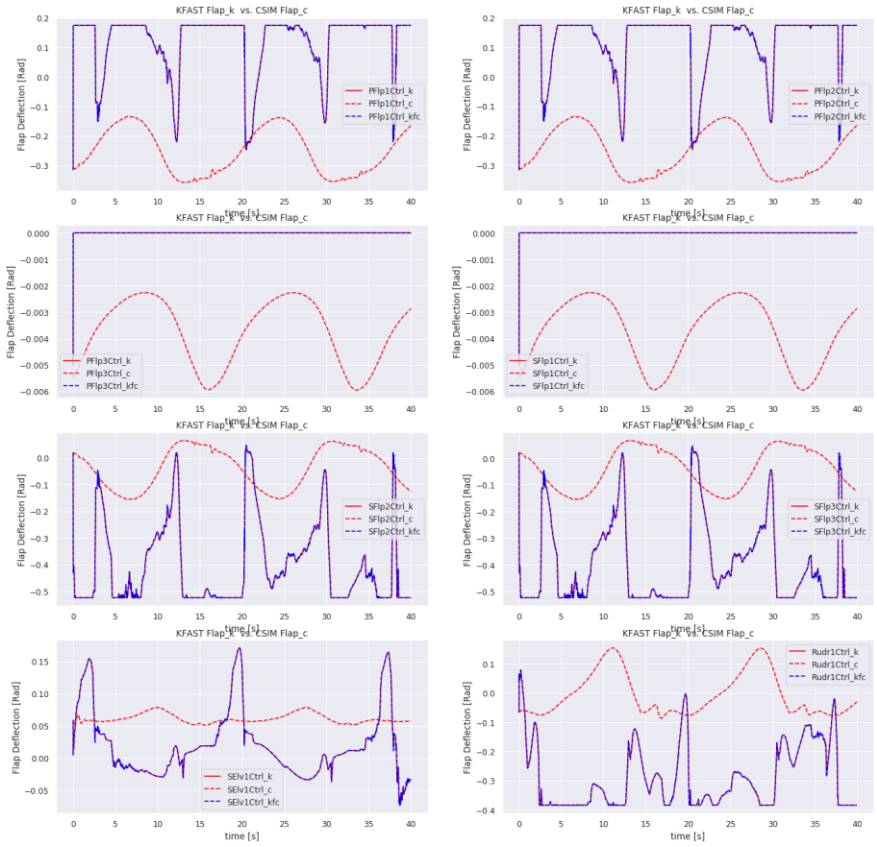


So as you can see the smoothing of Ab has an impact also on the Ag. This is likely because the rotors are calmer and so are the flap deflections and all. So given this, I think the largest contribution to the problem comes from the noise in the accel, which must be removed somehow, and then from the fact that Ab_x is way off. This is reflected by the AirFx which is positive whereas CSIM has it negative. Also look at how smoother the omegas get as a result of smoother flap deflections as well.

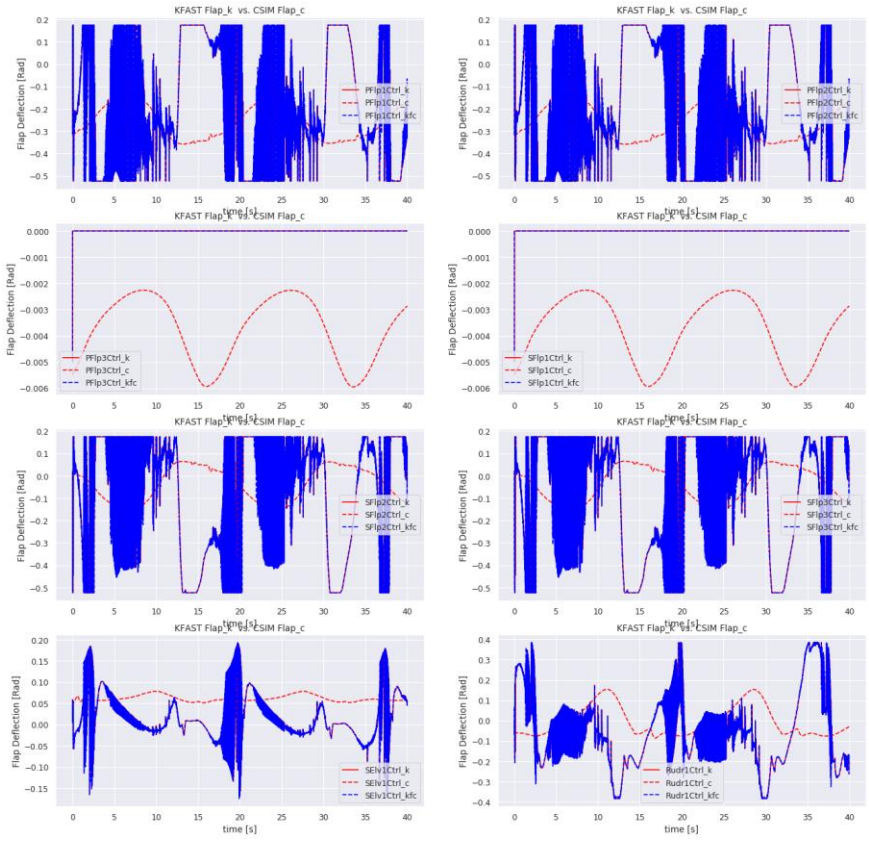


vs. non-smoothed



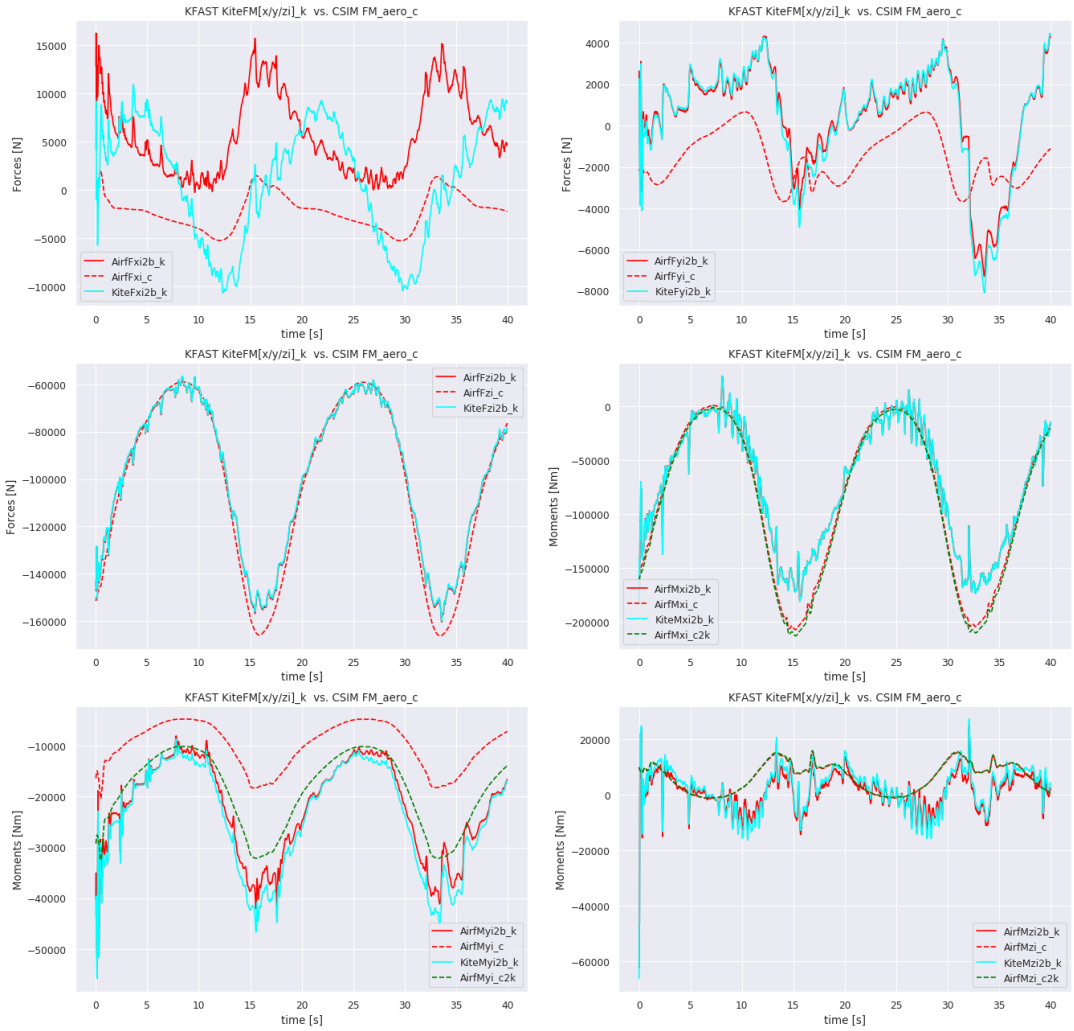


vs. nonsmoothed

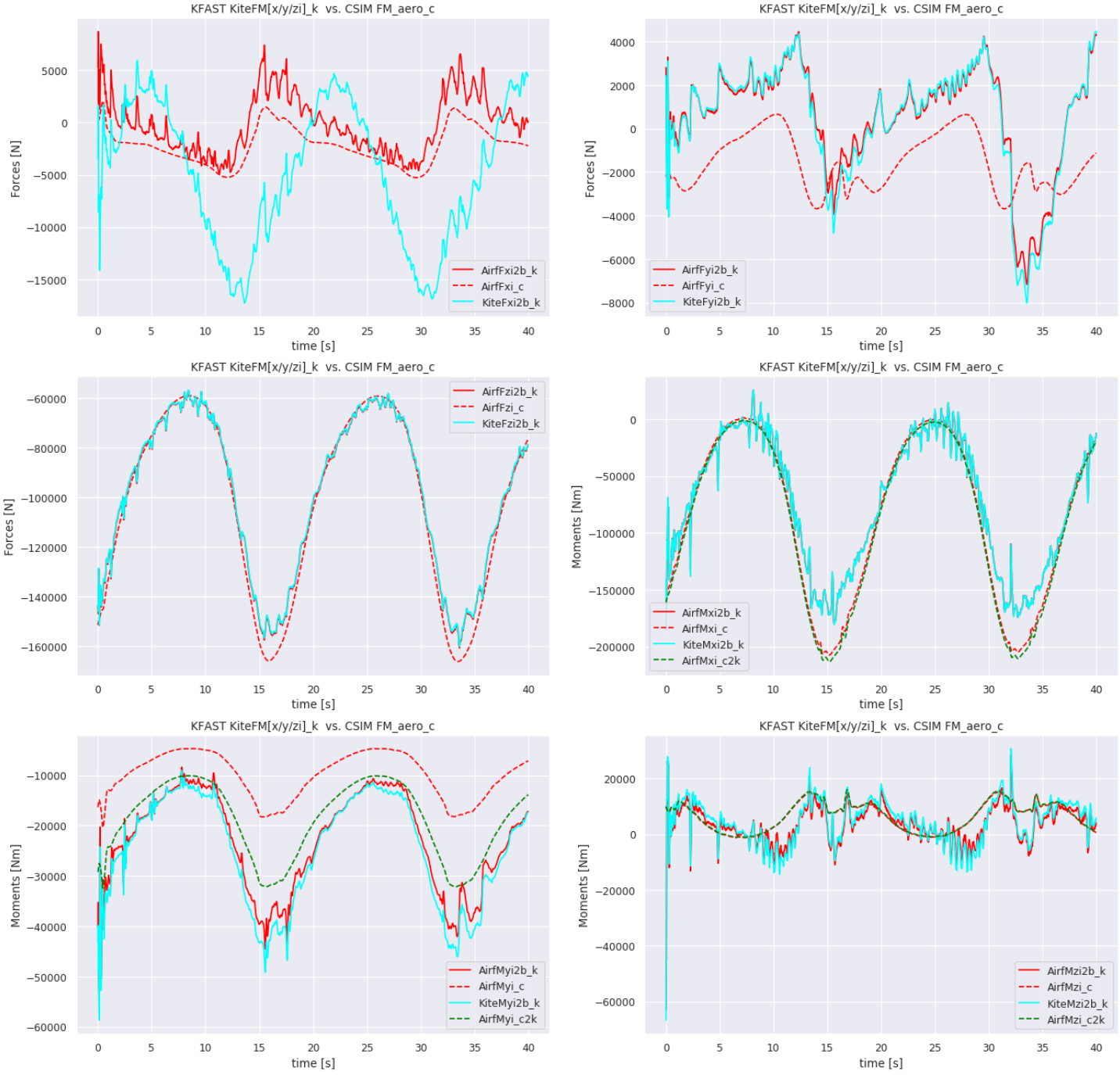


(Side note: I confirmed with Jerome that: 1. Xg contains coordinates of the kite body coordinate system origin, expressed in ground frame. 2. Apparent Wind is the velocity at the kite origin. 3. Vb and Ab are in body frame, Vg and Ag are in ground frame. They are velocity and acceleration of the body frame origin. So everytihng should match KFAST)

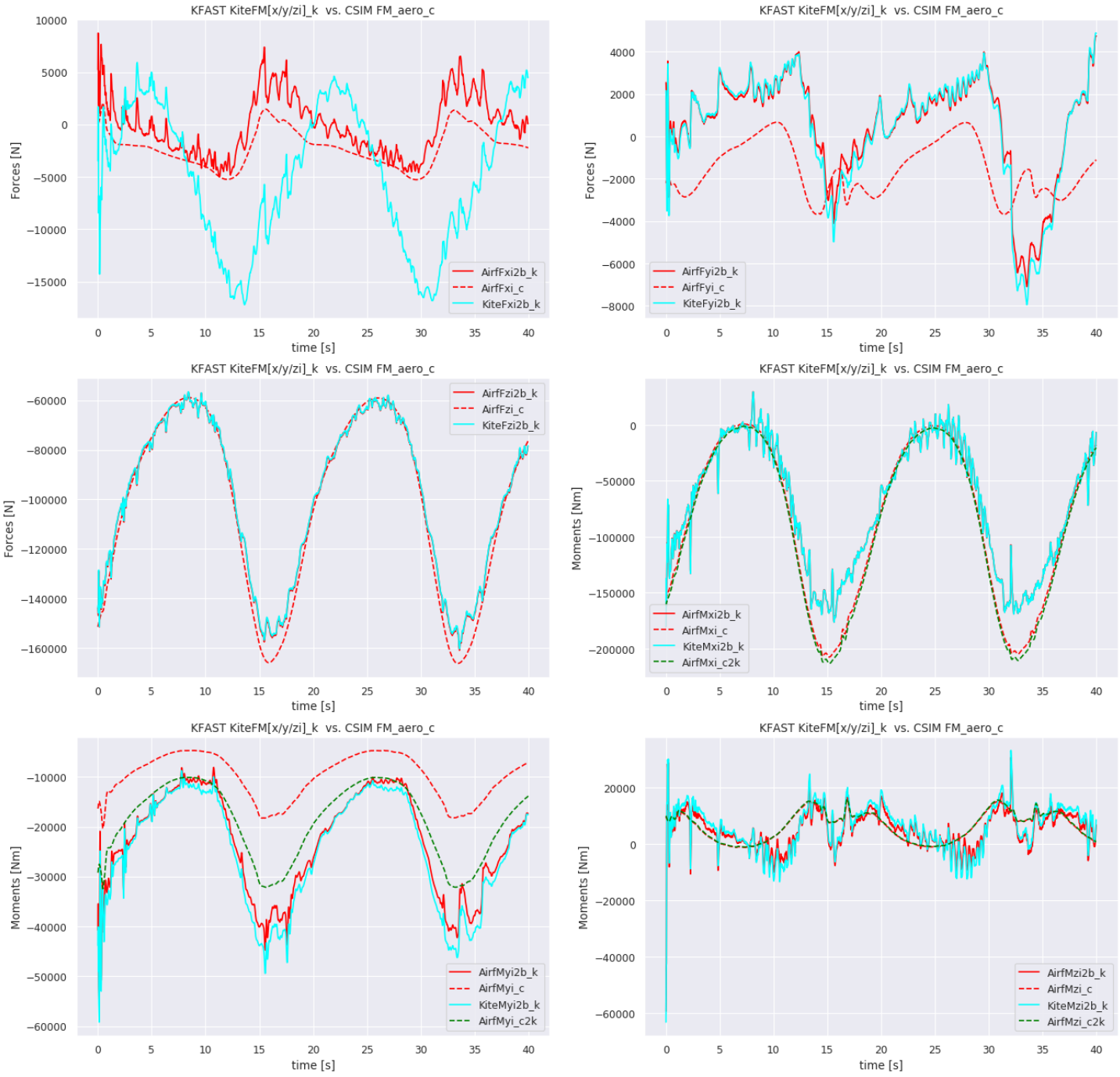
13. We have a problem wit**h the x component of the airframe loads. Also the moments are totally off. So the drag is likely ultra-underestimated**.
14. The position and Vb, and Ab should match better than what we have gotten. So let us investigate why that is the case. ⇒I found a problem in the input file, the MIP node was not referred to correctly, and so data were coming from a node not at (0,0,0). Now that is all fixed, **and we do not have much difference in the Xg and Vb_y and sideslip are now fine**. Though, t**he accelerations are all 0, which is a problem**.
15. Ruth prepared a new set of aifoils for the pylons and those seem to have been effective at reducing the Fy component; below is a run where I injected CSIM ctrl commands into FAST (via KFCmode=2), and as you can see we are much closer to CSIM now with both moments and Fy, though Fy still is positive, which pushes the kite out of the loop, it should be negative or toward the center of the loop for the kite to turn.



16. on 2/8/2020 I have hardwired a Cd of 0.075 to sum to all Cd's calculated (VSM.f90). So this + the rel vel at the motors are the two hardwired things so far in the code. Then I am filtering the accels and pqr in the controller side of things. Remember what has been hardwired and make sure to add to KAD as inputs in the future. Below is the improvement in Fx. If we could get Fy to behave better we might have a better chance at making the turn. Might need to get the total forces for each component to check against CSIM.

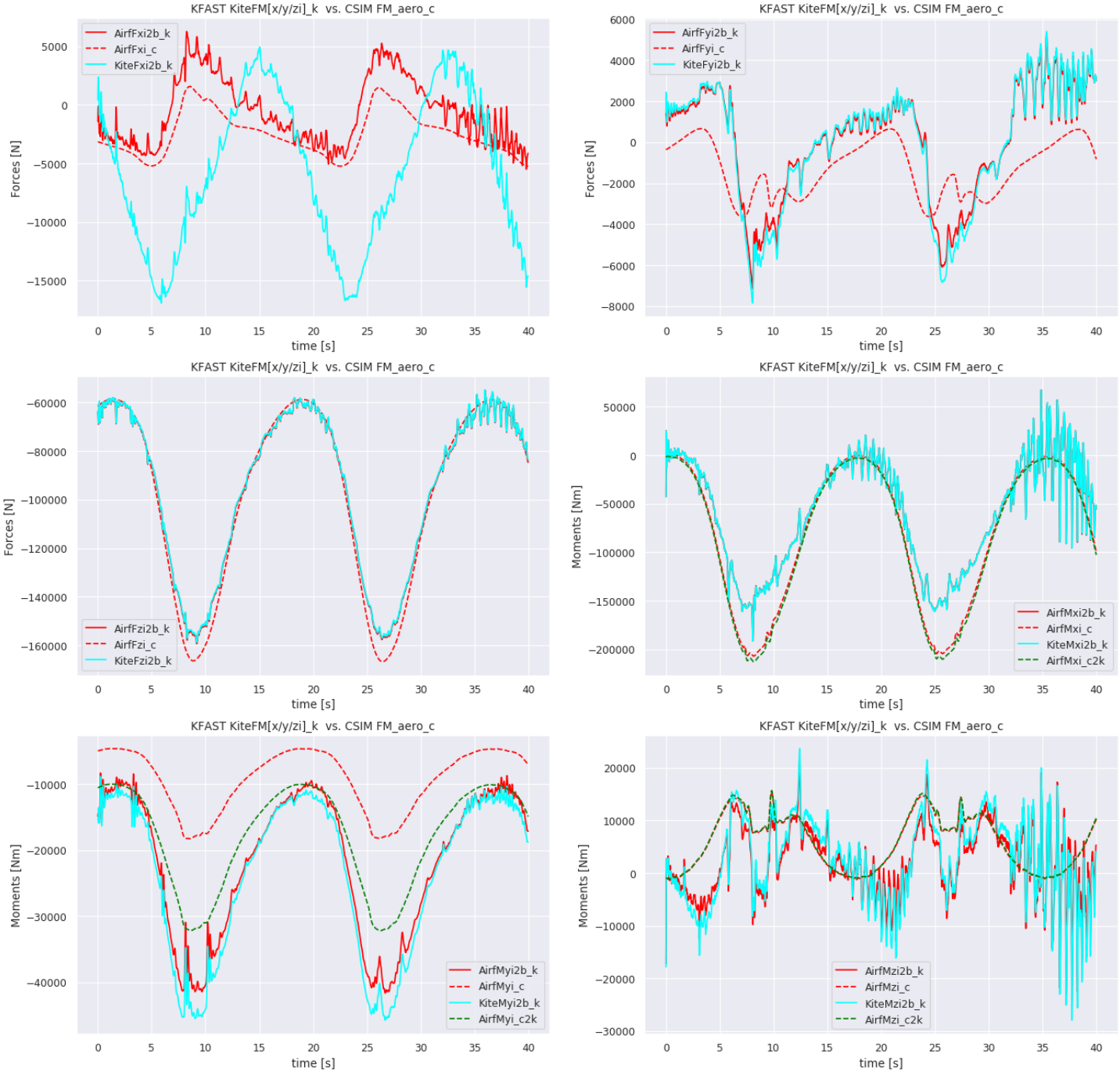


After fixing the dihedral (remember >0 on both sides) not much has changed as to be expected:

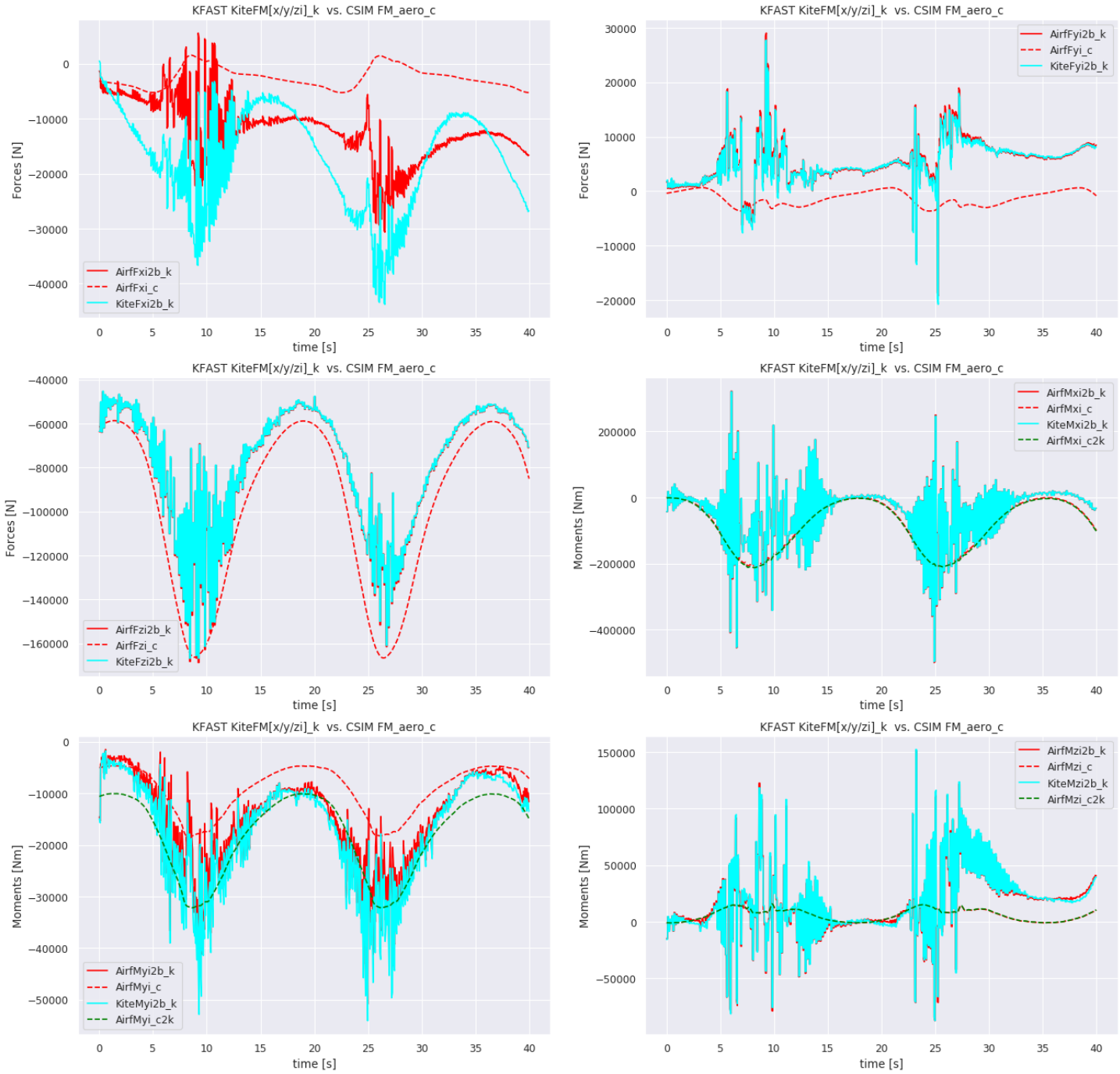


Nonetheless we still have to figure out why we have a positive AirfFy. We shall check the pylon aero next, comparing it to kiteAD standalone, and then try a new CSIM run, i.e. starting from a new point at the top of the circle.

There was a problem with one airfoil, and fixing the 1620 airfoil has improved a bit, but we are still a bit off, and a bit too positive as you can see below with high_start point:



I have run the hi_start case in double precision and VSM on as you see below this seems to be much worse. I confirmed that the oscillations come from a larger aero_dt in the KiteAD input file. So, it is worth keeping 0.01s. Yet, I cannot compile in DP and I was just forcing ReKi=8 in SingPrec.f90 which now has been reverted (3/17/20). I am not sure how to overcome the DP issue at this point. Yet that seems to be needed to run with VSM on.



I am trying to keep k_aero_cmd and k_geom_cmd constant. Added overwrites in crosswindpath.c. Also realized that Ab_f gets the effect of Justin's global structure, which is not good.

I have created a new KFC_Driver that can feed states to the STI controller based on a user input file, which I also created based on CSIM states. I have also created a notebook that can post-process outputs of a KFC_Driver standalone run (STIvsCSIM.ipynb). In this process, I found a few bugs in the STI wrapper and controller. Nothing major, but small improvements. In the process I have also increased the readability and formatting of the STI logger, and then augmented KiteFASTController.f90 to also spit out the tether_force_b, which has brought some unpleasant results. One of those is that the x-component of the tension seems wrong, ie. it looks as if it is a propulsion (see Figure below). How this can happen is a bit troublesome, because I could not find the bug. The y and z component seem fine. Obviously this creates a problem in the forces that the kite will feel, yet, the roll calculation is performed by controller_conversion.c and does not use the sign of Tx, so that does not seem to be affected. One more thing, I re-derived the components of the

tether_forces from tension roll and pitch, but I used a ground reference system, therefore in my standalone run there may be a problem. Well, actually the derivation does not know about which reference frame it is derived.