1. Nav Report (5/15/22)

We discuss the coordinate systems that may be used for evaluating the navigation system in the tests at STAR campus. The navigation system will nominally include two units, each capable of reporting position and orientation. For the test at STAR, the two units are the "b" system ABX-2 GPSS and the "p" system the APX IMU. As of 5/14/22, this report only considers the evaluation of the two systems for reporting orientation, where we mean the orientation of a solid body with respect to a local reference coordinate system. We begin with a discussion of reference frames and rotations.

1.1. Co-ordinate systems, orientation and rotations (skipped in this draft)

1.2. Application to navigation systems

Navigation system coordinates

For a navigation system, we want to specify the position and orientation of a solid body (e.g. a plane, car, etc). For the pilot of this object there is nominal orientation of the vehicle which corresponds to the direction the vehicle is pointed (not necessarily the direction it is moving). The pilot cares about the heading relative to a local reference, conventionally North, and the elevation the vehicle is pointed above or below horizontal. This direction defines the longitudinal axis or the roll axis of the vehicle. It's the direction the pilot sees out the front window. From the perspective of defining rotations. specifying the longitudinal axis is equivalent of specifying the up direction for local geographic coordinates - it takes two angles to specify a preferred direction on a sphere. There remains the rotation around the longitudinal axis. The reference direction for this rotation is with respect to an e_2 which is horizontal and right handed with respect to an e_1 which is forward and an e_3 which is down. From the pilot's perspective e_2 points to the right hand wing tip, and a positive roll corresponds to the right hand wing moving down and the left hand wing tip moving up.

To summarize, then, the navigation basis is $\{e_1, e_2, e_3\} = \{N, E, D\}$ and the orientation of the vehicle is defined by the basis vectors

$$R(\text{heading}, \text{pitch}, \text{roll}) = R_1(\text{roll})\, R_2(\text{pitch})\, R_3(\text{heading})$$

with the intermediate stages

$$e'''_i = R''_1\,(\text{roll})\, e''_i = R''_1\,(\text{roll})\, R'_2\,(\text{pitch})\, e'_i = R''_1\,(\text{roll})\, R'_2\,(\text{pitch})\, R_3(\text{heading})\, e_i$$

Here the rotations are defined and act on the basis established at that point of the sequence. Note that for navigation the longitudinal axis is e_1 whereas for Euler angles that preferred axis is labeled e_3. Also, note that $e_3 = D$, so a positive heading angle has an opposite horizontal 2D effect to the sense of a rotation by α for the first Euler angle.

Then, with

$$R_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_1 & s_1 \\ 0 & -s_1 & c_1 \end{pmatrix}, R_2 = \begin{pmatrix} c_2 & 0 & -s_2 \\ 0 & 1 & 0 \\ s_2 & 0 & c_2 \end{pmatrix}, R_3 = \begin{pmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

the coordinate basis due to heading, pitch, and roll is then

```
{R1 = (1 0 0; 0 c1 s1; 0 -s1 c1), R2 = (c2 0 -s2; 0 1 0; s2 0 c2), R3 = (c3 s3 0; -s3 c3 0; 0 0 1)};
Rtot = R1.R2.R3;
MatrixForm[Rtot];
```

$$R = \begin{pmatrix} c_2\, c_3 & c_2\, s_3 & -s_2 \\ c_3\, s_1\, s_2 - c_1\, s_3 & c_1\, c_3 + s_1\, s_2\, s_3 & c_2\, s_1 \\ c_1\, c_3\, s_2 + s_1\, s_3 & -c_3\, s_1 + c_1\, s_2\, s_3 & c_1\, c_2 \end{pmatrix}$$

where $c_i,\ s_i = \mathrm{Cos}[\theta_i],\ \mathrm{Sin}[\theta_i]$ and $\theta_1 = \text{heading},\ \ \theta_2 = \text{pitch},\ \ \theta_3 = \text{roll}.$

Note that one can work backwards ... If someone gives you R, then one can find heading, pitch and roll as

$\text{pitch} = \theta_2 = -\mathrm{ArcSin}[R_{13}]$
$\text{heading} = \theta_1 = \mathrm{ArcTan}[R_{33}/c_2,\ R_{23}/c_2]$
$\text{roll} = \theta_3 = \mathrm{ArcTan}[R_{13}/c_2,\ R_{12}/c_2]$

Note that although this was developed to find basis vectors for h,p,r, and vice versa, relative to the Nav basis of $\{N,\ E,\ D\}$, it can be used to find the equivalent basis and angles for any coordinate system relative to any basis set. Lastly, it is the property of unitary transformations that the elements of U_{ij} are given by the projections of the basis vectors of the two coordinate bases onto each other. i.e. the transformation from basis a to basis b is given by $\left(U^{\mathrm{AB}}\right)_{\mathrm{ij}} = \left(e^A\right)_i \cdot \left(e^B\right)_j$, where the dot indicates the inner product. For real spaces this is simply $\left(R^{\mathrm{AB}}\right)_{\mathrm{ij}} = \left(e^A\right)_i \cdot \left(e^B\right)_j$ where all quantities are real. Accordingly, if one knows the two sets of coordinate basis vectors, one can compute R, and then find heading, pitch, roll angles that would be needed to reorient the basis vectors form the first system to the second. In the case where one starts from $\{N,\ E,\ D\}$ and one knows the orientation basis vectors of the vehicle, one can determine the h,p,r needed to produce that orientation.

Relative orientation of ABX-2 and APX

The previous discussion gives us two mechanisms for determining the alignment of the ABX-2 and APX navigation systems.

method 1: From the h,p,r for each system, find the rotation matrices relative to Nav

$$R_p = \begin{pmatrix} c'_2\, c'_3 & c'_2\, s'_3 & -s'_2 \\ c'_3\, s'_1\, s'_2 - c'_1\, s'_3 & c'_1\, c'_3 + s'_1\, s'_2\, s'_3 & c'_2\, s'_1 \\ c'_1\, c'_3\, s'_2 + s'_1\, s'_3 & -c'_3\, s'_1 + c'_1\, s'_2\, s'_3 & c'_1\, c'_2 \end{pmatrix} \text{ and } R_b = \begin{pmatrix} c_2\, c_3 & c_2\, s_3 & -s_2 \\ c_3\, s_1\, s_2 - c_1\, s_3 & c_1\, c_3 + s_1\, s_2\, s_3 & c_2\, s_1 \\ c_1\, c_3\, s_2 + s_1\, s_3 & -c_3\, s_1 + c_1\, s_2\, s_3 & c_1\, c_2 \end{pmatrix}$$

so that $e'''_p = R_p\, e$ and $e'''_b = R_b\, e$. Then rewrite one relation by it’s inverse, e.g. $e = R_p^{-1}\, e'''_p$, so that

$$e'''_b = R_b\, R_p^{-1}\, e'''_p = R_{\mathrm{bp}}\, e'''_p$$

where R_{bp} is the orientation rotation which give the relative alignment of the ABX-2 relative to the APX. This can be done basis vector by basis vector (9 total numbers) or it can be condensed into three angles corresponding to the relative heading pitch and roll. NB! relative heading defined i nthis sense is *not* the same as heading_APX - heading_ABX-2. They will be similar for small angles of pitch and roll but not identical.

method 2: From the h,p,r for each system, find the explicit basis vectors e'''_p and e'''_b. Using these basis vectors, populate R_{bp} and proceed to find relative h,p,r. Or if all we need is to find the offset between the longitudinal axes compute $e'''_{p_1} \cdot e'''_{b1}$

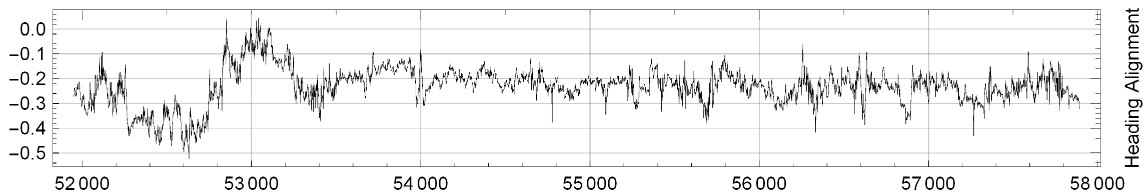
Alignment of APX, ABX-2

I took the input files provided by Quin, which give time, heading, pitch, roll for each system. Since the samples don’t always align in time, for each of the six angles, I produced a linear interpolation as a function of time, and limited the time domain to where there was data from both systems. I then followed method 2 and constructed $e'''_p,\ e'''_b$ as a function of time. These were used to construct relative heading, pitch and roll. in full transparency I didn’t construct the full R, but just the components need. For example, relative pitch comes from $\text{pitch} = \theta_2 = -\mathrm{ArcSin}[R_{13}] = -\mathrm{ArcSin}\left[e'''_{\mathrm{p1}} \cdot e'''_{\mathrm{b3}}\right]$. In addition to relative heading, pitch, and roll, I also find the space angle between the two longitudinal axes $\theta_{\mathrm{pb}} = \mathrm{ArcCos}[R_{11}]$

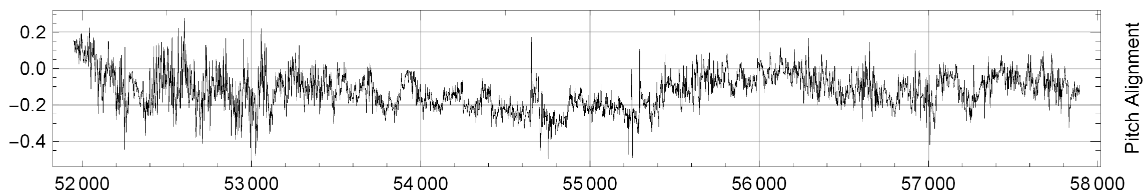
In[ ]:=

```
Plot[{getrph2[t][[1]] / °}, {t, tt0, tt0 + dt0}, ImageSize → paperwid,
 AspectRatio → .15, GridLines → Automatic, FrameLabel → {"", "", "", "Heading Alignment"}]
Plot[{getrph2[t][[2]] / °}, {t, tt0, tt0 + dt0}, ImageSize → paperwid, AspectRatio → .15,
 GridLines → Automatic, FrameLabel → {"", "", "", "Pitch Alignment"}]
Plot[{getrph2[t][[3]] / °}, {t, tt0, tt0 + dt0}, ImageSize → paperwid, AspectRatio → .15,
 GridLines → Automatic, FrameLabel → {"", "", "", "Roll Alignment"}]
Plot[{ArcCos[eb12[t].ep12[t]] / °}, {t, tt0, tt0 + dt0}, ImageSize → paperwid, AspectRatio → .15,
 GridLines → Automatic, FrameLabel → {"", "", "", "space angle of long axis"}]
```

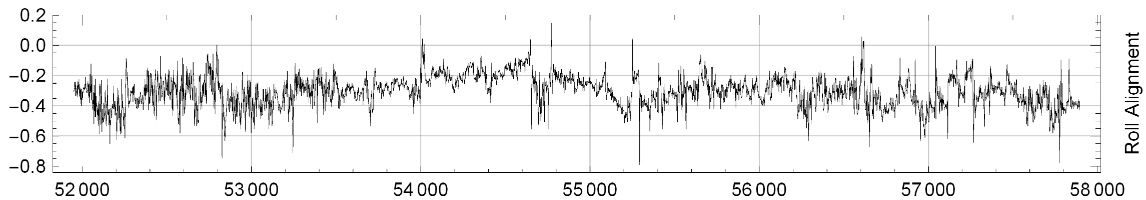
Out[ ]=



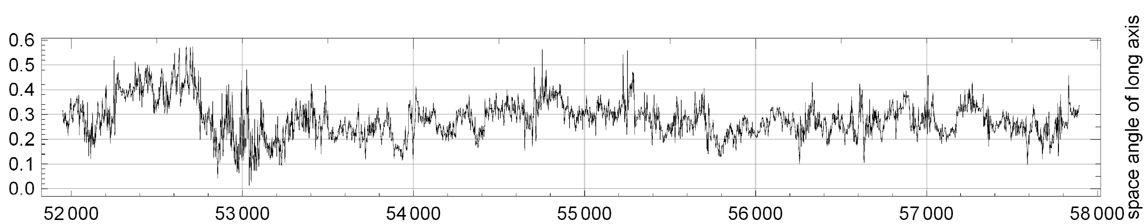
Out[ ]=



Out[ ]=



Out[ ]=



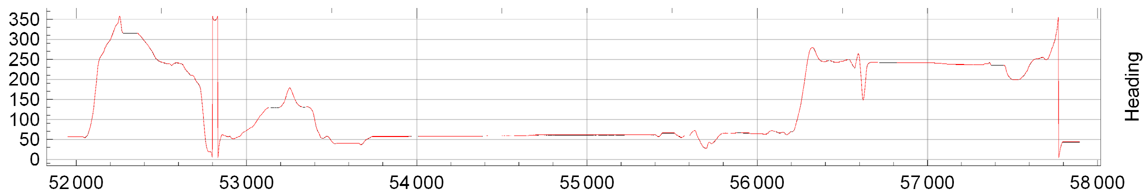
In principle, if the trailer was perfectly rigid platform, these relative alignments would be constant in time, at least up to the measurement accuracy of the two instruments. In practice, the excursions seem somewhat larger than the .05 deg allowed. Especially for the first 20 mins or so (1200 secs). The mis-alignment of the two forward directions appears to be mostly due to relative heading, but with some contributions due to relative pitch around t=55000.

The results for alignment suggest some how the platform is not rigid. If so, that could be due to stresses on the platform introducing strains, which are in effect measured by the nav siystems. With that in mind, here are the timelines for heading, pitch and roll. Black is APX, Red is ABX-2

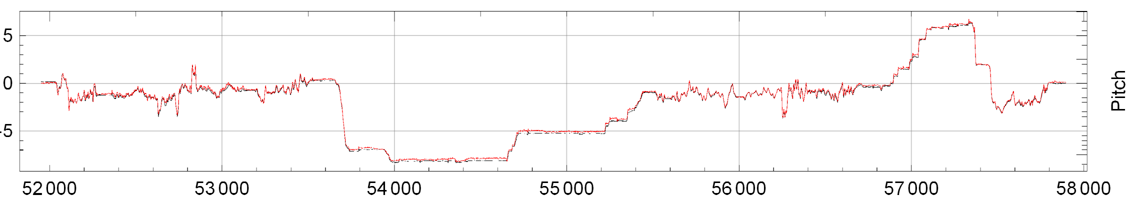
In[ ]:=

```
Plot[{hpd[t], hbd[t]}, {t, tt0, tt0 + dt0}, ImageSize → paperwid, AspectRatio → .15,
 GridLines → Automatic, FrameLabel → {"", "", "", "Heading"}, PlotRange → All]
Plot[{ppd[t], pbd[t]}, {t, tt0, tt0 + dt0}, ImageSize → paperwid, AspectRatio → .15,
 GridLines → Automatic, FrameLabel → {"", "", "", "Pitch"}, PlotRange → All]
Plot[{rpd[t], rbd[t]}, {t, tt0, tt0 + dt0}, ImageSize → paperwid, AspectRatio → .15,
 GridLines → Automatic, FrameLabel → {"", "", "", "Roll"}, PlotRange → All]
```

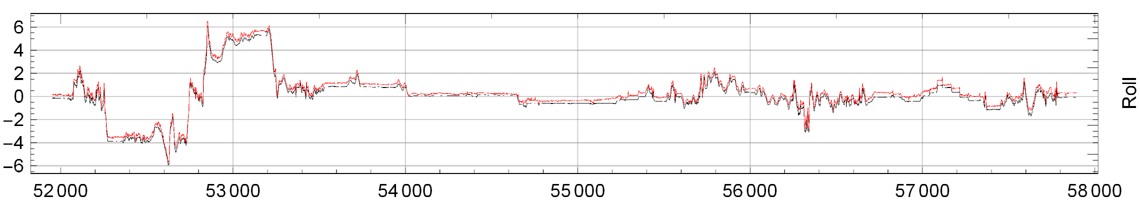
Out[ ]=



Out[ ]=



Out[ ]=

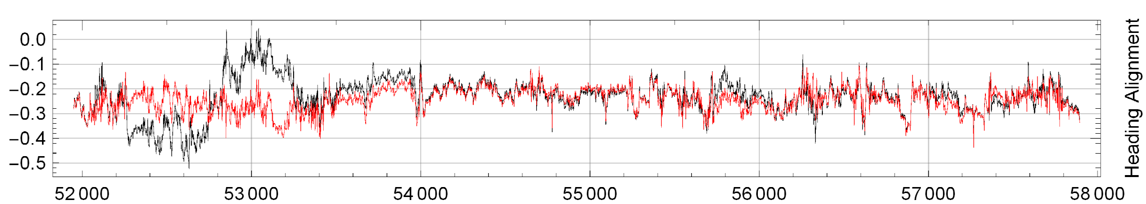


It is tempting to correlate roll with the relative heading. The last figure here shows black as relative heading from above, and red has a correction which is -.04 * roll. The roll is the real roll relative to Nav coordinates. Pretty clear that there is some correlation, but I don’t have a mechanical model.

In[ ]:=

```
Plot[{getrph2[t][[1]] / °, getrph2[t][[1]] / ° - .04 rbd[t]}, {t, tt0, tt0 + dt0}, ImageSize → paperwid,
 AspectRatio → .15, GridLines → Automatic, FrameLabel → {"", "", "", "Heading Alignment"}]
```

Out[ ]=



1.3. More stuff skipped in this draft