# Report AT26-17: Inter-Inertial Calibration and Data Fusion for accurate ocean current measurement

Dehann Fourie
dehann.fourie@gmail.com

August 22, 2014

# Contents

# Nomenclature & Conventions

Throughout this work:

- Sentry body axis of FWD-STB-DWN is used as main body relative coordinate system.
- All body relative measurements throughout this work is intended as Phins to instrument, unless explicitly stated otherwise.
- Body frame values in this text refers to existing Sentry navigation solutions, assumed to be located at the Phins.
- Abstract inertial frame $^{i}$.
- Body frame $^{b}$.
- Microstrain instrument frame $^{\mu}$.
- Flat world navigation frame $^{w}$.
- Quaternions are used as standard rotation representation.
- Quaternion convention rotation from $a$ to $b$, is positive scalar first then vector: ${}^{b}_{a}\mathring{\mathbf{q}} = [+w, x, y, z]^{T}$.
- Quaternions are indicated with a small circle on top, vectors are bold and $^{T}$ indicates transpose.
- Quaternion conjugate is denoted by $\mathring{\mathbf{q}}^{*}$.
- Quaternion composition is denoted by $\oplus$, or ommited for shorthand notation: ${}^{c}_{a}\mathring{\mathbf{q}} = {}^{c}_{b}\mathring{\mathbf{q}}\,{}^{b}_{a}\mathring{\mathbf{q}}$;
- ${}^{b}\mathbf{v} = {}^{b}_{a}\mathbf{R}\,{}^{a}\mathbf{v}$;
- ${}^{b}\omega_{b}$ means: Body measured (Phins) rotation rate, given in the body reference frame;
- ${}^{b}\Delta\mathbf{v}_{\mu|b}$ is the velocity delta of the Microstrain, given body velocity, represented in the body frame;
- $\tilde{\mathbf{a}}$ is noisy acceleration measurement vector, including sensor bias offsets;
- $\hat{\mathbf{v}}$ is estimated velocity vector, where the carrot indicates that it is an estimated quantity;
- $\mathbf{R}\left({}^{b}_{a}\Psi\right)$ is the $SO(3)$ exponential map for Euler angle $\Psi$ from $a$ to $b$;

# Chapter 1

# Inter-Inertial Alignment

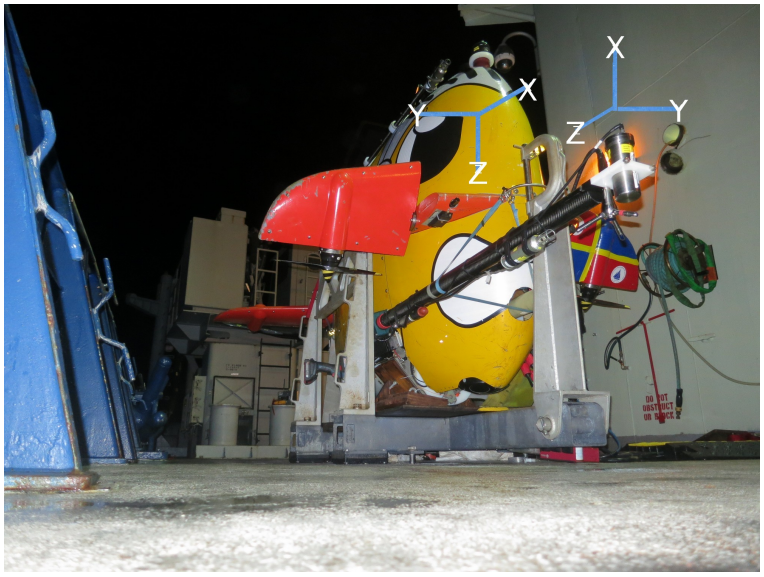
## 1.1 Introduction

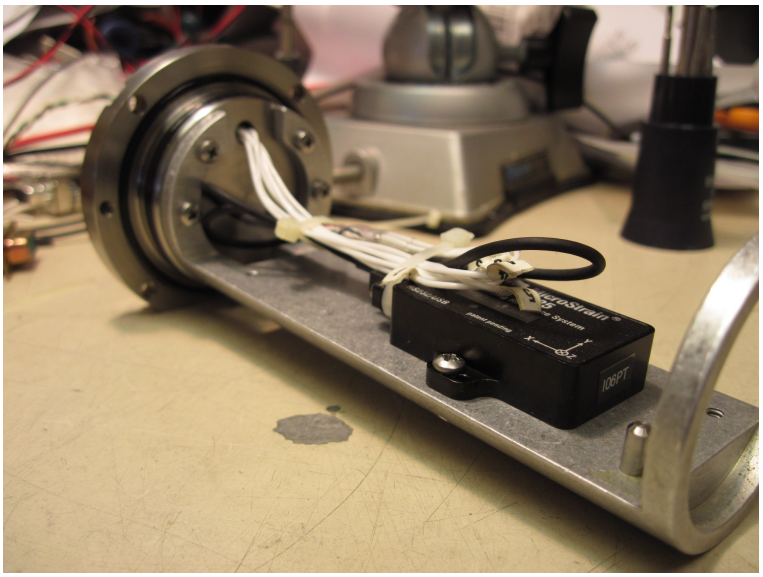
Cruise to Juan de Fuca, AT26-17. Heat flux and Optical communications experiment. Document presents algorithms and interaction with implemented `MATLAB` scripts. Document summarizes work and conversations had during the cruise. A definition of reference frames and other conventions for this work is given first. The main deliverables in this document are:

1. Coarse alignment from Phins to other sensors;
2. Fine alignment of Phins GVX to Microstrain (GX3) MSA3 strings; and
3. `MATLAB` functions to compute fine alignment between Phins and ADV mounted Microstrain;
4. Algebraic definition for calculating motion deltas of the ADV relative to Sentry for measuring ocean currents from dynamic base; and
5. Design of Kalman filter for fusing various measurements for estimation of ADV pole motion deltas.

## 1.2 Setup

A short account of what things looked like on AT26-17:


X
Y
Z
X
Y
Z



## 1.3 Coarse alignment

### 1.3.1 Rotation Alignment

We use direct conversions for the initial coarse alignment of the rotational transform between Phins and Microstrain:

$$ {}^{\mu}_{b}\mathring{\mathbf{q}} = \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}^{T} \tag{1.1} $$

The direct approach produces a good initialization for the fine alignment scheme below. However, for completeness; one could also use the Attitude Heading Reference System (AHRS) estimates from both IMU's to find a coarse alignment:

$$ {}^{\mu}_{b}\mathring{\mathbf{q}} = {}^{w}_{\mu}\mathring{\mathbf{q}}^{*} \oplus {}^{w}_{b}\mathring{\mathbf{q}} \tag{1.2} $$

### 1.3.2 Translation Alignment

CAD measurements were used to obtain initial translation estimates between Phins and the other instruments on Sentry.

Table 1.1: Linear translation from Phins center of measurement to (in Sentry body frame)

| FWD Microstrain | (w extra strap) | |
|---|---|---|
| X | 1958 $mm$ | $\pm 10$ $mm$ |
| Y | 408 $mm$ | $+10, \ -15$ $mm$ |
| Z | 731 $mm$ | $+10, \ -15$ $mm$ |
| CHA Microstrain | | |
| X | $mm$ | $\pm$ $mm$ |
| Y | $mm$ | $\pm$ $mm$ |
| Z | $mm$ | $\pm$ $mm$ |
| FWD DVL | (diamond face, | replacement) |
| X | 647 $mm$ | $\pm 5$ $mm$ |
| Y | 0 $mm$ | $\pm 8$ $mm$ |
| Z | 1060 $mm$ | $\pm 10$ $mm$ |
| AFT DVL | (diamond face) | |
| X | $-876$ $mm$ | $\pm 1$ $mm$ |
| Y | 0 $mm$ | $\pm 1$ $mm$ |
| Z | 1060 $mm$ | $\pm 10$ $mm$ |
| FWD ADV | (incl. 16 $cm$, | w extra strap) |
| X | 1895 $mm$ | $\pm 10$ $mm$ |
| Y | 408 $mm$ | $+10, \ -15$ $mm$ |
| Z | 990 $mm$ | $+15, \ -20$ $mm$ |

## 1.4 Fine alignment

Gyro rates can be used to extract a finer rotational alignment between the two IMUs. The approach followed is least squares optimization from a reasonable initial estimate – use the coarse alignment estimated from the section above. Microstrain biases should also be estimated for higher accuracy nominal alignment. A MATLAB implementation of this – based on Sentry dive 275 data – can be found by running:

```
extractFineAlignSegment.m
```

MSA and MSA3 data – converted to GVX and GX3 – from **Sentry dive 275** was used during development of the alignment scripts.

### 1.4.1 Batch optimization

A nonlinear least squares batch optimization is set up as follows:

$$\mathbf{\Theta}^* = \underset{\Theta}{\operatorname{argmin}} \left[ \sum_i^N \parallel f_i \left( \mathbf{\Theta}_i; \mathcal{Z}_i \right) \parallel^2 \right] \tag{1.3}$$

where $\Theta = \left[ {}^{\mu}_{\hat{\mu}}\Delta\Psi, \ {}^{\mu}\hat{\Delta}_{\omega} \right]^T$, ${}^{\mu}_{\hat{\mu}}\Delta\Psi$ is the Euler representation of the error between initial coarse alignment estimate and finer batch alignment; ${}^{\mu}\hat{\Delta}_{\omega}$ is Microstrain gyro bias estimate in the Microstrain instrument frame ${}^{\mu}$. $f_i$ is the $i^{\text{th}}$ measurement prediction and $\mathcal{Z}_i$ is the associated measurement.

$$f_i \left( \mathbf{\Theta}_i; \mathcal{Z}_i \right) = \mathbf{R} \left( {}^{\mu}_{\hat{\mu}}\Delta\Psi \right) \mathbf{R} \left( {}^{\hat{\mu}}_{b}\mathring{\mathbf{q}} \right) {}^{b}\omega \ - \ \left( {}^{\mu}\tilde{\omega} - {}^{\mu}\hat{\Delta}_{\omega} \right) \tag{1.4}$$

Optimization is done in the body frame. This is implemented in the MATLAB function:

`fineAlignRatesBias.m`

A unit test script for `fineAlignRatesBias.m` exists. To see how the fine alignment works, please run:

`unitTestBatchAlignment.m`

### Issue on MSA3 timestamp conversions

An issue was discovered with MSA3 timestamp conversions – this issue has not yet been fixed. The issue is easy to see, just plot `gx3.t` or `gx3.unix_time`. A sawtooth pattern should be obvious in the timestamp data. For this work, I decided not to resolve the timestamp conversion issues and continue skip through to the fine alignment task.

### Extracting useful portion of data

The methodology was to extract a reasonable portion of the Sentry275 data, from which batch optimization scripts could be run to extract a fine alignment estimate. Approximately 5 hours of data was used as the baseline for development work. This data segment starts at:

1405990800.108002 *s*
Which is near:
7/22/2014, 01:00

The choice of this data segment is twofold: GX3 timestamp is monotonically increasing during this time period, and there is notable vehicle motion during the early portion of this data segment. Note, the latter portion of the data shows Sentry driving a lawn mower pattern with little variation in pitch and roll. Please refer to fig. 1.1 for described data segment.

### Compute Rates from GVX Attitude

The GVX data strings do not store gyro rates from the Phins. We desire the rotation rates from the Phins system and use the following attitude difference technique to deduce what the rotation body rates were.

$$\mathbf{S} = \frac{\mathbf{R}\left({}^{b_k}_{w}\mathring{\mathbf{q}}\right)\mathbf{R}\left({}^{w}_{b_{k-1}}\mathring{\mathbf{q}}\right) - \mathbf{I}_3}{\Delta t} \tag{1.5}$$

$${}^{b}\omega \approx \begin{bmatrix} S_{3,2} \\ S_{1,3} \\ S_{2,1} \end{bmatrix} \tag{1.6}$$

A `MATLAB` implementation is included:

`computeRatesFromRPY.m`

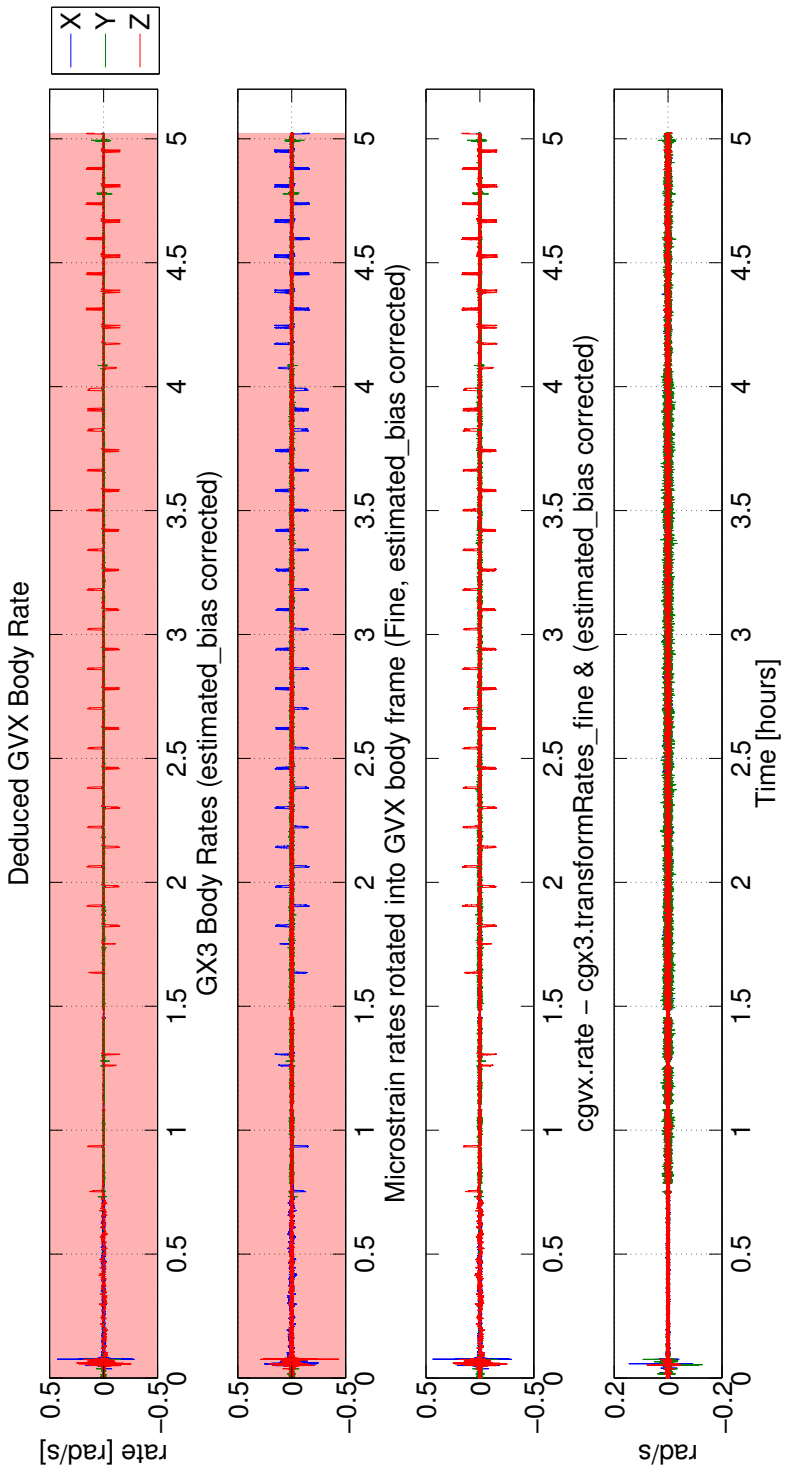



Figure 1.1: Fine alignment plots from Sentry dive 275. Top shows rates deduced from Phins (GVX) roll, pitch and heading values at 5 $Hz$. Next is raw ADV Microstrain rates which are subsampled from 25 $Hz$ stream – these include estimated bias correction. Second to bottom are Microstrain rates rotated into Sentry body with estimated fine alignment quaternion. Bottom figure is the euclidean difference between Phins deduced rates and bias corrected and rotated Microstrain rates. These deltas represent the ADV pole motion relative to Sentry. The shaded region represents data used in batch fine alignment optimization. This plot is generated by `extractFineAlignSegment.m`.

### 1.4.2 Results from `extractFineAlignSegment.m` – Sentry275

The fine alignment results with two `BATCH_SIZE`s is given. We choose to present two sets of results since variation in Phins and Microstrain clock rates are not optimizated in this work. While this is certainly feasible, we have limited the scope of work done.

A batch optimization over the first 33 *mins* of the data segment was done (`BATCH_SIZE` = 10000), where we assume the delta time rate influence is less dominant. (Easy to copy-paste format)

$\overset{\mu}{_b}\mathring{\mathbf{q}}_{275,10000} =$

```
[ 0.005676811771873
 -0.701250366720770
 -0.002424574121515
  0.712888363225562 ]
```

$\overset{\mu}{_b}\mathbf{R}_{275,10000} =$

```
[ -0.016431393964184  -0.004693399121072  -0.999853979987620
   0.011494333088488  -0.999923790496872   0.004504831319650
  -0.999798924583961  -0.011418634027673   0.016484089228629 ]
```

And estimated Microstrain gyro bias:

$\hat{\mu}\hat{\Delta}_{\omega,275,10000} =$

```
[ 0.003050655782702
 -0.001034052497962
  0.000051480918584 ]
```

A batch optimization over all 5 *hours* of the data segment was done (`BATCH_SIZE` = 90409), where we assume the delta time rate may influence rotational accuracy.

$\overset{\mu}{_b}\mathring{\mathbf{q}}_{275,90409} =$

```
[ 0.004984950143904
 -0.703290442617208
 -0.002573971779266
  0.710880495065545 ]
```

$\overset{\mu}{_b}\mathbf{R}_{275,90409} =$

```
[ -0.010715407190709  -0.003466908148503  -0.999936578287157
   0.010707907156199  -0.999937049882685   0.003352162920803
  -0.999885253803191  -0.010671308251721   0.010751855985143 ]
```

And estimated Microstrain gyro bias:

$\hat{\mu}\hat{\Delta}_{\omega,275,90409} =$

```
[ 0.002969066763825
 -0.001224708974738
  0.000168953578443 ]
```

Note, the longer batch segments may in fact yield a more accurate stable Microstrain gyro bias estimate.

### 1.4.3 Expected output

```
 extractFineAlignSegment and may take several minutes to run
(Parallel toolbox is being used, parfor)
Computation time depends most heavily on size of batch optimization size (
BATCH_SIZE=20000)

Sizes of gvx and gx3: 90409  90409
Starting parallel pool (parpool) using the 'local' profile ... connected to 4 workers.

coarse_bQu =

                    0
  -0.707106781186547
                    0
   0.707106781186547


Solver stopped prematurely.

lsqnonlin stopped because it exceeded the function evaluation limit,
options.MaxFunEvals = 600 (the default value).

Optimized cost is: 11.3973

residMisalQ_coarse =

   0.939371290331095
  -0.051548237552723
   0.242165983201782
   0.237234050437150

residMisalQ_coarse SANITY check -- should be near quaternion identity [1;0;0;0]

FINE ROTATIONAL ALIGNMENT using batch least squares
---------------------------------------------------

ENABLE_GYRO_BIAS_EST =

     1


Local minimum possible.

lsqnonlin stopped because the final change in the sum of squares relative to
its initial value is less than the selected value of the function tolerance.

<stopping criteria details>

Optimized cost is: 0.8869

fine_bQu =
```

```
   0.005587299201945
  -0.701485175726093
  -0.002606518723649
   0.712657376573276


gx3GyroBias =

   0.002988978228068
  -0.001049076133330
   0.000014999841264


Solver stopped prematurely.

lsqnonlin stopped because it exceeded the function evaluation limit,
options.MaxFunEvals = 600 (the default value).

Optimized cost is: 11.5092

residMisalQ_fine =

   0.998515553674528
  -0.021942313663781
   0.047674858699111
  -0.014571609010349

residMisalQ_fine SANITY check -- should be near quaternion identity [1;0;0;0]

Elapsed time is 577.252200 seconds.
End of extractFineAlignSegment
Please note, q2R([0;-7071;0;7071]) == q2R([0;0.7071;0;-0.7071])
```

Please note the that SANITY checks do not enable the bias estimation aspect of the batch optimization process – this will result in additional error in its output.

### 1.4.4 Confirm Microstrain to MSA3 gyro axis conversions

The purpose of this test was to confirm that rate information from the DSL Microstrain driver to MSA3 data strings and `read_msa3.m` script are positive xyz, since this convention was in question due to these comments from the DSL code base:

```
 LLW seems to be the first to have changed coordinates for microstrain logging
for Nereus in 2007 -- later comments suggest the hacks were removed
Current LCM logger by JCK says:
"// Assign attitude rate output.
// @@@ Convention is messed up here."
and also:
"Crude approx ok for coordinate frame at microstrain.  These are
probably wrong - unclear what convention is being used in microstrain functions."
```

A basic test sequence was logged with the Microstrain-ADV sensor cluster – filename:

`strainAdv_postdive_test20140805_1623.DAT`

The test sequence is a simple hand held rotation about the Microstrain mechanical positive X then negative X axis; in similar succession for the Y and Z axes. Please see fig. 1.2 for raw data from this test, which confirms direct, and sign consistent, xyz data representation.

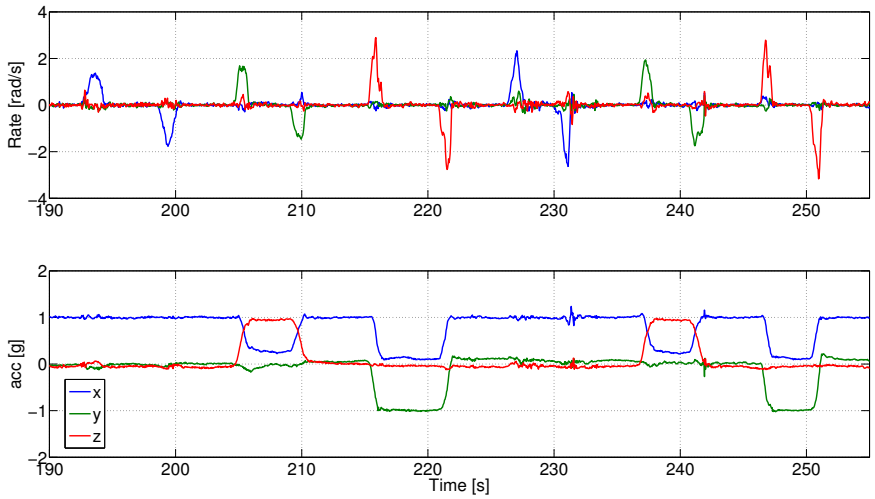



Figure 1.2: Microstrain gyro to MSA3 data: Sign and axes consistency checks; all seems OK.

# Chapter 2

# Dynamic platform measurement

## 2.1 Motion Deltas

We are working towards measuring the ocean current velocity in the world frame, ${}^{w}\mathbf{v}_{\mu}$. To do so, we need to compute the world frame velocity of the ADV mounted Microstrain – assuming we will then convert to velocity of ADV sense location in the world frame.

To obtain reasonable accuracy, we must subtract the motion delta of the ADV at the end of pole from Sentry body measured rates. The school of thought is to use a Microstrain IMU – which is to be slaved to the more accurate Phins IMU – to measure the ADV motion relative to the accurate Sentry navigation solution. The purpose is to do accurate ocean current velocity estimates from Sentry as a moving base. The ocean current measurements should be resolved to the world frame.

The Microstrain near ADV measures both rotation rates and accelerations. Here we develop expressions to compute Sentry Phins to Microstrain motion delta. First, we define two sets of equations, each using either just Mirostrain gyroscopes, or Microstrain accelerometers. Each have their own advantages and should therefore be fused in a suitable manner.

### 2.1.1 Gyroscope based motion deltas

The cross product of delta rotation rates and translational offset will compute the velocity of the Microstrain sensor, $\mu$, relative to the Sentry Phins, $b$. Starting with the delta rotation rates as:

$$
\begin{aligned}
{}^{b}\Delta\omega_{\mu|b} &= {}^{b}\omega_{\mu} - {}^{b}\omega_{b} && (2.1)\\
&= {}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\omega_{\mu} - {}^{b}\omega_{b} && (2.2)
\end{aligned}
$$

which gives the delta velocities in the body frame

$$
{}^{b}\Delta\mathbf{v}_{\mu|b} = {}^{b}\Delta\omega_{\mu|b} \times \overrightarrow{\mathbf{l}}_{b\to\mu} \tag{2.3}
$$

$\overrightarrow{\mathbf{l}}_{b\to\mu}$ is the linear translation vector from the Phins measurement center to the Microstrain – coarse values were given in table 1.1. ${}^{b}\Delta\mathbf{v}_{\mu|b}$ is the velocity delta of the Microstrain, given body velocity, represented in the body frame.

So lets now collect all velocity terms to find Microstrain velocity in the world frame:

$$
{}^{w}\mathbf{v}_{\mu} = {}^{w}\mathbf{v}_{b} + {}^{w}_{b}\mathbf{R}\left({}^{b}\omega_{b} \times \overrightarrow{\mathbf{l}}_{b\to\mu} + {}^{b}\Delta\mathbf{v}_{\mu|b}\right) \tag{2.4}
$$

We can simplify the expression:

$$
{}^{w}\mathbf{v}_{\mu} = {}^{w}\mathbf{v}_{b} + {}^{w}_{b}\mathbf{R}\left(\left[{}^{b}\omega_{b} + {}^{b}\Delta\omega_{\mu|b}\right] \times \overrightarrow{\mathbf{l}}_{b\to\mu}\right) \tag{2.5}
$$

This concludes the Microstrain velocity estimate in the world frame using Microstrain gyroscopes.

### 2.1.2 Accelerometer based motion deltas

We want the Microstrain velocity in the world frame, $w$:

$$ {}^{w}\mathbf{v}_{\mu} = {}^{w}\mathbf{v}_{b} + {}^{w}_{b}\mathbf{R}\left({}^{b}\omega_{b} \times \overrightarrow{\mathbf{l}}_{b\to\mu} + {}^{b}\Delta\mathbf{v}_{\mu|b}\right) \tag{2.6} $$

#### Using vehicle accelerations

First we compute the Microstrain velocity deltas in vehicle body frame $b$ – using Microstrain acceleration measurements.

$$ {}^{b}\Delta\mathbf{v}_{\mu|b} \approx \int {}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\mathbf{a}_{\mu} - {}^{b}\mathbf{a}_{b}\; d\tau \tag{2.7} $$

where we have assumed the change ${}^{b}_{\mu}\mathbf{R}$ due to ADV pole motion is negligible. This model, however, is overly simplistic. We have no mechanization doesn't allow for Microstrain accelerometer bias correction. With the foresight of data fusion between gyroscopes and accelerometers, we introduce the hooks for correcting Microstrain accelerometer biases, ${}^{\mu}\hat{\Delta}_{a,\mu}$:

$$ {}^{b}\Delta\mathbf{v}_{\mu|b} \approx \int {}^{b}_{\mu}\mathbf{R}\left({}^{\mu}\tilde{\mathbf{a}}_{\mu} - {}^{\mu}\hat{\Delta}_{a,\mu}\right) - {}^{b}\mathbf{a}_{b}\; d\tau \tag{2.8} $$

Lets redevelop these expressions, but using vehicle velocity, rather than acceleration, information. If we were to use raw Phins accelerations, we would probably have to estimate their biases also, which is best done with the DVL. We assume an INS/DVL fused result is already available and therefore avoid double work.

#### Using vehicle velocities

The advantage of working from vehicle velocities ${}^{b}\mathbf{v}$ is that we can use the DVL, or an INS/DVL fused estimate, as information source. It boils down to choice – when do we fuse information from different sensors.

$$ {}^{b}\Delta\mathbf{v}_{\mu|b} \approx \int {}^{b}_{\mu}\mathbf{R}\left({}^{\mu}\tilde{\mathbf{a}}_{\mu} - {}^{\mu}\hat{\Delta}_{a,\mu}\right)\; d\tau - \int {}^{b}\mathbf{a}_{b}\; d\tau \tag{2.9} $$

$$ \approx \int {}^{b}_{\mu}\mathbf{R}\left({}^{\mu}\tilde{\mathbf{a}}_{\mu} - {}^{\mu}\hat{\Delta}_{a,\mu}\right)\; d\tau - {}^{b}\mathbf{v} \tag{2.10} $$

# Chapter 3

# Data fusion

The presented algorithm is capable of real-time operation (recursive least squares). An alternative approach would be to compute a batch solution for ocean flow vectors from raw sensor data using direct optimization techniques. A feasible batch solution is possible and would likely be a more accurate post-processing approach.

Lastly, the derivation here has not included the Phins acceleration, since they have not available in the logged data from this cruise. We can certainly add Phins accelerations to this data fusion process. The assumption here is, the ${}^{b}\mathbf{v}$ estimate is from the already fused INS/DVL system.

## 3.1 Complementary Kalman filter based fusion

We want to fuse accelerometer and gyroscope based measurements of ADV mounted Microstrain motion deltas to achieve higher accuracy ocean current measurement. Lets take a complementary filtering approach to fuse rate and acceleration information.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \eta \tag{3.1}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}) + \nu \tag{3.2}$$

Process noise, $\eta$, and measurement noise, $\nu$, are discussed further in section 3.2.

### 3.1.1 Estimated states

We must choose the best fitting states, $\mathbf{x}$, to estimate: We could use either the vehicle velocity, Microstrain velocity, or Microstrain velocity deltas as the estimator state. We choose the Microstrain velocity deltas as the desired state estimate. Therefore 9-states at time $k$ – delta velocities in the body frame, Microstrain gyro and accelerometer bias:

$$\mathbf{x}_k = \begin{bmatrix} {}^{b}\Delta\mathbf{v}_{\mu|b} \\ {}^{\mu}\Delta_{\omega,\mu} \\ {}^{\mu}\Delta_{a,\mu} \end{bmatrix} \tag{3.3}$$

and input at time $k$

$$\mathbf{u}_k = \begin{bmatrix} {}^{b}\omega_b \\ {}^{\mu}\tilde{\omega}_{\mu} \\ {}^{\mu}\tilde{\mathbf{a}}_{\mu} \\ {}^{b}\mathbf{v}_k \\ {}^{b}\mathbf{v}_{k-1} \end{bmatrix} \tag{3.4}$$

This choice allows all Kalman filter equations to be **linear time invariant** (LTI).

### 3.1.2 State measurement model

Choose the Microstrain gyroscope measurements as the long term – low frequency – motion delta information source. We do not have to integrate gyroscope rates to get to velocity deltas and does not suffer sensor bias integration. We therefore introduce the gyroscope based motion deltas through the Kalman Filter measurement model $\mathbf{y} = \mathbf{h}(\mathbf{x}) + \mathbf{j}(\mathbf{u})$.

$$\begin{aligned}
\mathbf{y}_k = {}^{b}\Delta\hat{\mathbf{v}}_{\mu|b} &= \left({}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\hat{\omega}_{\mu} - {}^{b}\omega_b\right) \times \overrightarrow{l}_{b\to\mu} && (3.5)\\
&= \left({}^{b}_{\mu}\mathbf{R}\left({}^{\mu}\tilde{\omega}_{\mu} - {}^{\mu}\hat{\Delta}_{\omega,\mu}\right)\right) \times \overrightarrow{l}_{b\to\mu} - {}^{b}\omega_b \times \overrightarrow{l}_{b\to\mu} && (3.6)\\
&= -\left({}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\hat{\Delta}_{\omega,\mu}\right) \times \overrightarrow{l}_{b\to\mu} + \left({}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\tilde{\omega}_{\mu}\right) \times \overrightarrow{l}_{b\to\mu} - {}^{b}\omega_b \times \overrightarrow{l}_{b\to\mu} && (3.7)\\
&= \mathbf{L}_{\mu\to b}\,{}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\hat{\Delta}_{\omega,\mu} - \mathbf{L}_{\mu\to b}\,{}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\tilde{\omega}_{\mu} + \mathbf{L}_{\mu\to b}\,{}^{b}\omega_b && (3.8)\\
&= \mathbf{H}\,\mathbf{x}_k + \mathbf{J}\,\mathbf{u}_k && (3.9)
\end{aligned}$$

where

$$\left[\overrightarrow{l}_{b\to\mu}\times\right] = \mathbf{L}_{\mu\to b} = \begin{bmatrix} 0 & -l_z & l_y \\ l_z & 0 & -l_x \\ -l_y & l_x & 0 \end{bmatrix} \tag{3.10}$$

$$\mathbf{H}\,\mathbf{x}_k = \begin{bmatrix}\mathbf{0} & \mathbf{L}_{\mu\to b}\,{}^{b}_{\mu}\mathbf{R} & \mathbf{0}\end{bmatrix} \begin{bmatrix} {}^{b}\Delta\mathbf{v}_{\mu|b} \\ {}^{\mu}\Delta_{\omega,\mu} \\ {}^{\mu}\Delta_{a,\mu} \end{bmatrix} \tag{3.11}$$

$$\mathbf{J}\,\mathbf{u}_k = \begin{bmatrix}\mathbf{L}_{\mu\to b} & -\mathbf{L}_{\mu\to b}\,{}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\tilde{\omega}_{\mu} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \begin{bmatrix} {}^{b}\omega_b \\ {}^{\mu}\tilde{\omega}_{\mu} \\ {}^{\mu}\tilde{\mathbf{a}}_{\mu} \\ {}^{b}\mathbf{v}_k \\ {}^{b}\mathbf{v}_{k-1} \end{bmatrix} \tag{3.12}$$

Look carefully at the change of signs, since the order of multiplication on the cross products were switched.

To recap: The data fused Sentry to ADV pole velocity delta, at time $k$, is extracted with:

$${}^{b}\Delta\hat{\mathbf{v}}_{\mu|b} = \mathbf{H}\,\mathbf{x}_k + \mathbf{J}\,\mathbf{u}_k \tag{3.13}$$

### 3.1.3 State propagation model

Choose Microstrain accelerometers as short term – high frequency – motion delta information source. Accelerations must be integrated to estimate velocities and can therefore not stand alone as the long term stable motion delta information source. Accelerometers should be an excellent source of motion deltas, if used in a DC decoupled, or bias compensated, configuration. We use the input model, $\mathbf{g}(\mathbf{u})$, to introduce accelerometer motion deltas to the combined state estimate.

Working from eq. (2.10), we take Microstrain velocity in the vehicle body frame and convert to velocity space through differentiation:

$$\begin{aligned}
\frac{d}{dt}\,{}^{b}\Delta\mathbf{v}_{\mu|b} = {}^{b}\Delta\mathbf{a}_{\mu} &= {}^{b}_{\mu}\mathbf{R}\left({}^{\mu}\mathbf{a}_{\mu} - {}^{\mu}\hat{\Delta}_{a,\mu}\right) - \frac{d}{dt}\,{}^{b}\mathbf{v} && (3.14)\\
&= -{}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\Delta_{a,\mu} + {}^{b}_{\mu}\mathbf{R}\,{}^{\mu}\tilde{\mathbf{a}}_{\mu} - \frac{d}{dt}\,{}^{b}\mathbf{v} && (3.15)
\end{aligned}$$

Writing this in standard state space form:

$$\frac{d}{dt}\,\mathbf{x}_k = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k \tag{3.16}$$

where:

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & -{}^{b}_{\mu}\mathbf{R} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.17}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & {}^{b}_{\mu}\mathbf{R} & -\frac{1}{\Delta t}\mathbf{I} & \frac{1}{\Delta t}\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.18}$$

### 3.1.4 Discrete Kalman filter

The Kalman filter formulation described above is linear and time invariant (**LTI**). We can convert this filter to the discrete domain with constant matrices – following the convention:

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k + \mathbf{\Lambda}\eta \tag{3.19}$$

Process noise shaping matrix is identity, since we are dealing with a full LTI estimator: $\mathbf{\Lambda} = \mathbf{I}$. This simplifies the expressions for discrete time process noise covariance matrix $Q_d$. Recall the increase in state uncertainty during estimator state propagation:

$$\mathbf{P}_{k+1|k} = \mathbf{\Phi}\mathbf{P}_{k|k}\mathbf{\Phi}^T + \mathbf{Q}_d \tag{3.20}$$

This mechanization therefore requires we compute discrete equivalents $\mathbf{\Phi}$, $\mathbf{\Gamma}$ and $\mathbf{Q}_d$. To obtain these matrices, please follow standard composite matrix exponential techniques for continuous to discrete time conversions. Use of Pade‘ approximation for matrix exponential is suggested, and is readily implemented in `MATLAB`'s `expm.m` function.

## 3.2 Process and Measurement noise covariances

We also need to define the correct covariances for process and measurement noise. Starting with continuous process noise (normally distributed):

$$\eta \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_c\right) \tag{3.21}$$

$$\mathbf{Q}_{c;1,1} = {\sigma_{VRW-\mu strain}}^2 + \frac{2}{\Delta t}{\sigma_{^b\mathbf{v}}}^2 \tag{3.22}$$

$$\mathbf{Q}_{c;2,2} = {\sigma_{\omega,BI,\mu strain}}^2 \tag{3.23}$$

$$\mathbf{Q}_{c;3,3} = {\sigma_{a,BI,\mu strain}}^2 \tag{3.24}$$

$$\tag{3.25}$$

where $ARW$ is angle random walk, $VRW$ velocity random walk and $BI$ is bias instability associated with Gauss-Markov process lumped together with un-modeled $\frac{1}{f}$ bias variation.

Similarly for measurement noise:

$$\nu \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_\sigma\right) \tag{3.26}$$

$$\mathbf{R}_\sigma = diag\left[\mathbf{L}_{\mu\to b}\left({\sigma_{\omega,BI,\mu strain}}^2 + {\sigma_{ARW,\mu strain}}^2 + {\sigma_{ARW,Phins}}^2\right)\right] \tag{3.27}$$

$$\tag{3.28}$$

We follow standard spectral density convention, $\Omega$, for converting between IMU specifications and signal covariances:

$${\sigma_\eta}^2 = \frac{\Omega_\eta}{\Delta t} \tag{3.29}$$

### 3.2.1 Safety Factor

As always, it is a good idea to introduce a safety factor to the process and measurement noise covariance matrices. A good start should be:

$$\mathbf{R}_\sigma \leftarrow 1.5 \times \mathbf{R}_\sigma \tag{3.30}$$

Then we expect a final process noise safety factor to be around $\alpha = 2$ to ensure some filter robustness

$$\mathbf{Q}_c \leftarrow \alpha \times \mathbf{Q}_c \tag{3.31}$$

however, we suggest starting with $\alpha$ much larger, for example 10 for initial filter trials. Once confidence has been established in the filter implementation, you can reduce $\alpha$. The smaller $\alpha$ is, the more accurate the state estimate becomes, but increases the risk of filter divergence.

This case has a linear filter, so divergence should be easily detectable and recoverable. An adaptive Kalman filter can be achieved by dynamically varying $\alpha$ based on confidence bounds in measurement residual and innovation covariance estimates.

### 3.2.2 Initial State Covariance

Tthe following initial state covariace matrix, $\mathbf{P}_0$, is suggested to help with initial convergence tests:

$$\mathbf{P}_0 = \begin{bmatrix} 100 \times \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 5 \times \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 10 \times \mathbf{I} \end{bmatrix} \tag{3.32}$$