104 - Rowing Analysis

Version `1_0` (version ID `10410`). Five sensing nodes, sampled at 100 Hz, measure sagittal rowing kinematics on one selected side. No feedback node is required. All angles are in degrees. Valid raw sensor values are preserved; invalid values are handled as described below.

Sensor placement

Select **Measured Side** (`right` by default, or `left`) before starting the trial. Place foot, shank and thigh nodes on that same side. The side setting changes the lateral mounting transform; flexion signs stay the same.

Sensor name	Placement	Positive sensor axes
`foot_sensor`	Securely on the dorsum of the foot/shoe	Y forward, Z up
`shank_sensor`	Lateral shank	Y up along the shank, Z laterally outward
`thigh_sensor`	Lateral thigh	Y up along the thigh, Z laterally outward
`sacrum_sensor`	Posterior sacrum, approximately S2	Y up, Z posterior
`lumbar_sensor`	Posterior lumbar region, approximately L1	Y up, Z posterior

On the right leg, lateral means toward the right; on the left, toward the left. Keep sensor axes aligned to the segment, and secure attachments against skin and clothing movement. Use consistent landmarks across sessions. The lumbar and sacrum nodes remain on the midline for either selected side.

Standing calibration

1. Start the trial while standing upright, legs straight, feet flat and parallel, facing forward. Avoid rotating the leg relative to the pelvis.

2. Hold still for **Standing Calibration Duration**, default 3 seconds. All five orientations are averaged using a sign-invariant quaternion mean.

3. Wait for the streamed status to read **Calibrated. Begin rowing.** Then sit down and row. No maximum flexion/extension capture is required.

4. To recalibrate, stand again and press the button on any of the five sensors. The previous calibration is invalidated immediately and a new standing capture begins.

The hub function button also requests recalibration when the system's **Control App with Button** setting is disabled. When that setting is enabled, the hub button retains its start/stop function; use a sensor button to recalibrate. Button release events are ignored. Stand still again after pressing a sensor button so the capture does not include movement from touching the sensor.

The capture restarts if a node moves more than **Calibration Motion Tolerance** (default 3 degrees), sensor data is invalid, or a mounted superior axis is more than 30 degrees from vertical for the shank/thigh or 60 degrees for the foot/back. The larger foot allowance accommodates shoe and dorsum mounting inclination; the back allowance preserves natural standing surface inclination. This posture check rejects obvious seated poses; it cannot establish anatomical neutral or correct a misplaced sensor. Until calibration completes, exported angles are zero placeholders and `data_valid` is 0. Repeat calibration after moving a sensor, reconnecting/reinitializing a node, or when accumulated drift is suspected. Recalibrating during a recorded trial creates a new `calibration_id`; analyse those portions separately.

Angles and signs

All relative angles are zeroed to the individual's standing posture. Positive values indicate flexion/dorsiflexion; extension/plantarflexion is negative. The knee angle is flexion (0 degrees standing), not the included angle (180 degrees standing).

Output	Segments/reference	Positive direction
`ankle_dorsiflexion_angle`	Foot relative to shank, sign reversed	Dorsiflexion; shank leaning forward over foot
`knee_flexion_angle`	Shank relative to thigh	Knee bending
`hip_flexion_angle`	Thigh relative to sacrum, sign reversed	Thigh moving anteriorly relative to pelvis
`lumbar_flexion_angle`	Lumbar relative to sacrum	Lumbar segment bending forward relative to pelvis
`sacral_inclination_angle`	Sacral sensor's mounted superior axis relative to gravity in its sagittal plane	Anterior surface inclination; standing offset is retained
`sacral_tilt_from_standing`	Sacral inclination minus standing inclination	Anterior tilt from standing

For example, sagittal segment pitches of foot 0, shank 30, thigh -70, sacrum 20, and lumbar 45 degrees, after neutral standing calibration, give ankle 30, knee 100, hip 90 and lumbar 25 degrees.

Sacral inclination is computed as `atan2(-g_x, g_z)`, where g is global superior expressed in the mounted sacrum frame (X anterior, Y left, Z superior). It is independent of global heading and retains the sensor's standing surface inclination. The standing-relative value is the wrapped difference of this inclination and its calibration value. These are **surface inclination measures**; neither is radiographic sacral slope, ASIS-PSIS pelvic tilt, or sacral endplate orientation. Lumbar flexion estimates the L1-to-sacrum surface-segment angle change; it does not resolve individual vertebral motion or absolute lordosis.

Quaternion and heading method

- Continue to use the existing `Rotation.py` wrapper for quaternion operations; incoming sensor quaternions are scalar first (`Quat1` through `Quat4`).

- Map each specified sensor mounting to a common right-handed segment frame: X anterior, Y left, Z superior.

- During standing capture, align each segment's initial heading to the sacrum, then remove the common starting heading. The forward segment axis supplies heading instead of raw sensor Euler yaw, avoiding mounting singularities.

- Store a fixed sensor/segment standing offset for each node. Subsequent segment orientation is `heading_correction * sensor_rotation * mounting * standing_offset`.

- For each adjacent pair, compute `proximal.inv() * distal`. Extract the first intrinsic Y-X-Z Cardan angle (mediolateral flexion first); reverse the hip and ankle signs to obtain positive hip flexion and dorsiflexion. This is a documented sagittal approximation, not a complete landmark-defined ISB model.

- No sample-by-sample yaw subtraction or forced planar motion is applied. Flexion becomes undefined at a 90-degree middle-axis rotation. The internal calculation returns NaN; the exported row has zero angles and `data_valid = 0`.

Heading alignment is adapted from Core-Algorithms/YawCorrection.py. That method supplies **initial yaw alignment**, not continuous drift estimation. Gravity constrains inclination but not heading. Independent gyro yaw drift after calibration can still bias relative joint angles. A fixed footplate alone does not identify the changing heading errors of all other nodes. Reliable continuous correction would require a validated motion constraint/estimator, functional calibration, or an external heading reference. Recalibrate between bouts and validate against known poses and a reference system before research collection.

Saved and streamed data

Derived fields are the six angles above plus:

- `time`: sample count divided by 100 Hz, seconds since trial start.

- `side`: selected side, `left` or `right`.

- `is_calibrated`: integer 1 when standing calibration exists, otherwise 0.

- `data_valid`: integer 1 only when calibration exists, all raw channels are finite and present, and all six derived angles are finite. Otherwise 0. It does not assess measurement accuracy or detect accumulated drift.

- `raw_data_valid`: integer 1 when the complete five-node raw packet contains finite numeric values for every SDK raw channel, otherwise 0. It can be 1 during standing capture even while `data_valid` is 0.

- `calibration_id`: 0 before first calibration, incremented on every successful capture. Use it to detect reference changes within a trial.

- `user_defined_status`: standing capture progress or connection/posture status.

Sensor sampling is explicitly requested with `sample_rate: 100`, matching `datarate: 100`, which the app uses for its time and calibration calculations. BodyNet starts/stops acquisition and recording through the inherited `pre_work()`/`post_work()` lifecycle. No additional acquisition loop is started.

Any invalid derived row exports **all six angles as 0.0**, with `data_valid = 0`. Always filter on `data_valid == 1` before analysis: a placeholder is not a measured zero. Flags use integer 0/1 because the SDK XLSX writer mishandles Python boolean numeric cells. This also keeps exported numeric fields finite for standard JSON parsing and Excel.

Raw sensor channels continue through Sage's normal saving and streaming paths:

- Valid raw values are preserved and the incoming packet is never mutated.

- CSV/H5 retain nonfinite numeric raw values; missing or nonnumeric raw channels are represented as NaN. Missing nodes are padded to the five-node schema so the SDK writer can still record a row.

- XLSX and live JSON replace missing/nonfinite raw channels with 0.0 and mark `raw_data_valid = 0`. For live transport only, a missing/nonfinite `Sampletime` uses another node's finite timestamp, or a sample-count timestamp if none is available, because BodyNet's websocket sender requires a finite timestamp.

- A finite but zero-norm quaternion has `data_valid = 0`; `raw_data_valid` can still be 1 because the raw channels themselves are present and finite.

Previous angle values are not repeated for invalid rows. A drop-out retains the standing reference, but node reinitialization or movement requires manual standing recalibration. BodyNet may stop the trial on a Bluetooth disconnect.

The save-mode choices remain CSV, H5, XLSX and none. `config.json` and `snapshot.json` start with the same five-node settings and no old device bindings. BodyNet reads `snapshot.json` first and uses `config.json` only as a fallback; change active settings through the Sage UI or update `snapshot.json`. Change the side before the trial; switching it requires a new standing capture.

Research basis and limitations

The implementation follows the research practices of explicitly reporting segment placements, neutral standing references, relative segment rotations, and separate pelvis/sacrum inclination and lumbar motion:

- Lebleu et al. (2020), Lower Limb Kinematics Using Inertial Sensors during Locomotion compares sensor-to-segment calibration approaches using standing plus functional movements. Static standing here is combined with the specified assumed mounting axes; it does not supply their functional axis calibration.

- The validity of using inertial measurement units to monitor the torso and pelvis sagittal plane motion of elite rowers (2022) supports studying torso and pelvis sagittal angles with IMUs in rowing. Its validation results do not transfer automatically to this hardware/model.

- Wu et al. (2002), ISB recommendations for ankle, hip and spine joint coordinates motivates explicit proximal/distal frames and angle conventions. The present five surface sensors do not locate the anatomical landmarks required for full conformity with those recommendations.

Sensor placement, soft-tissue motion, incorrect standing posture and heading drift all affect the measurements. The numerical tests verify angle conventions and app behavior; this version has not been validated against motion capture or tested live on five Sage nodes. No additional smoothing is applied, so there is no undocumented filter latency; offline filtering choices should be reported.

Development checks

With NumPy, SciPy and packaging installed, run:

```
python -m unittest discover -s tests -v
```

The suite uses physical synthetic sensor poses on both sides and a stub of the Sage lifecycle. It checks mounting axes, flexion beyond 90 degrees, angle signs, initial heading offsets, rigid-body motion, sacral standing offsets, quaternion signs, calibration/recalibration, movement and seated-pose rejection, invalid sensor data, strict JSON packets, finite XLSX payloads, integer flags, complete save schemas and preservation of valid raw values.

To additionally exercise selected methods from a local BodyNet checkout:

```
BODYNET_REPO=/path/to/bodynet python -m unittest discover -s tests -v
```

`tests/test_bodynet_compatibility.py` extracts and runs actual SDK method bodies without importing hardware drivers or starting worker processes. These checks cover websocket JSON, XLSX cells, field validation, both hub-button modes, 100 Hz startup, and snapshot precedence. They skip when `BODYNET_REPO` is unset. This verifies the selected SDK paths, not a complete live hardware trial.

To regenerate the PDF after editing this guide, install ReportLab in a development environment and run `python tools/build_user_guide.py`.