Key Physical Fields Derived from a Single 9-DOF IMU on a Golf Club

Draft v1.4 — Jonah Sachs project notes

June 26, 2025

1 Setup & Notation

A single **9-DOF IMU** (*ax, ay, az, gx, gy, gz, mx, my, mz*) is epoxied just below the handle.

- Body axes follow the right-hand rule; $+\hat{z}$ points *down* the shaft.
- A sensor-fusion filter outputs the quaternion $q_{SI}(t)$ (sensor $\rightarrow$ inertial).
- Lever arm to head CG: $\mathbf{r} = [\,0, 0, -\ell]^{\mathsf{T}}$ with $\ell \approx 1.07\,\mathrm{m}$.

2 Raw Channels (9)

Symbol	Units	Comment
$\mathbf{a} = [a_x, a_y, a_z]$	m/s^2 or g	linear accel. incl. g
$\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$	rad/s	body rates
$\mathbf{m} = [m_x, m_y, m_z]$	µT	magnetic heading

3 Derived Golf Metrics (single-stick model)

Group	Metric	Method (plain-English)	Formula / Symbolic	Use
Pose	Euler angles (ϕ, θ, ψ)	Convert quaternion q_{SI} to roll–pitch–yaw with `QuatToEuler`.	$(\phi, \theta, \psi) = \text{QuatToEuler}(q_{SI})$	Shaft attitude at impact / takeaway
Timing				
	Backswing time T_b	First sign-change of ω_z (start→top).	$T_b = t_{\text{top}} - t_{\text{start}}$	Rhythm
	Downswing time T_d	Top→impact; use max $\|\mathbf{a}\|$ as impact.	$T_d = t_{\text{impact}} - t_{\text{top}}$	Rhythm
	Tempo ratio	Divide the two intervals.	$T_b : T_d$	3:1 goal
Speed / Power				
	Head speed v_h	Rotate $\mathbf{r}$ to world and cross with $\boldsymbol{\omega}_I$.	$v_h = \\|\boldsymbol{\omega}_I \times R_{SI}\mathbf{r}\\|$	Carry estimate
	Peak g	Stream the max of $\|\mathbf{a}\|$ per swing.	$a_{\text{pk}} = \max \|\mathbf{a}\|$	Mishit flag
	Centripetal force F_c	Use axial ω_z and lever arm ℓ.	$F_c = m\ell\omega_z^2$	Shaft load
Path / Plane				
	Plane tilt β	PCA on shaft axis over downswing; angle vs $+\hat{z}$.	$\beta = \arccos(\hat{n} \cdot \hat{z})$	Swing geometry
	Attack angle α	Take vertical component v_z of v_h.	$\alpha = \arcsin\big(v_z/\|\mathbf{v}\|\big)$	Up / down strike
	Club-path γ	Azimuth of v_h in ground plane.	$\gamma = \text{atan2}(v_y, v_x)$	Draw / fade
	Face angle[1]	Rotate stored face normal $\hat{f}_S$ by R_{SI}.	$\delta = \arccos(\hat{f}_I \cdot \hat{x})$	Open / closed
Quality				
	Release frame	Zero-cross of angular accel. $\dot{\omega}_z$.	$\dot{\omega}_z = 0$	Lag timing
	Smoothness S	Integrate squared jerk over swing.	$S = \int \|\dot{\mathbf{a}}\|^2 \, dt$	Compare swings
	Impact FFT	FFT of $\mathbf{a}$ in ± 3 ms.	PSD around impact	Contact quality

4 Extra Low-Bandwidth Logs

- Timestamp (ns)
- Battery voltage — HW QA
- Impact flag (boolean) — trims backswing data
- Club ID and length ℓ — correct head-speed calc

[1] One-time static calibration registers the club-face normal $\hat{f}_{\text{club}}$ in the sensor frame.

5 Reference End-to-End Workflow (pseudo-Python)

The code fragment below shows how one **nine-value IMU packet**

$$[\,a_x, a_y, a_z,\ \omega_x, \omega_y, \omega_z,\ m_x, m_y, m_z\,]$$

Listing 1: Minimal processing loop

```
# ---- constants & one-time calibration ---------------------------------
FS        = 200.0                    # sample rate, Hz
R_S       = np.array([0, 0, -1.07])        # lever arm in sensor axes (m)
F_S       = np.array([1, 0,  0])           # face normal after static calib
Z_S       = np.array([0, 0,  1])           # shaft axis in sensor frame
Z_I       = np.array([0, 0,  1])           # up-direction in world frame
BUF_AXES, BUF_TS = [], []                  # ring buffers for PCA & tempo

# ---- main loop ---------------------------------------------------------
for pkt in imu_stream():                  # unpack one packet
    ax, ay, az, gx, gy, gz, mx, my, mz, ts = pkt

    a_S = np.array([ax, ay, az])
    w_S = np.array([gx, gy, gz])
    m_S = np.array([mx, my, mz])

    # (1) sensor fusion -> quaternion and rotation matrix
    q_SI = madgwick.updateIMU(w_S, a_S)
    R_SI = quat_to_rotmat(q_SI)           # 3 x 3 matrix
    w_I  = R_SI @ w_S

    # (2) rigid-stick kinematics
    r_I  = R_SI @ R_S
    v_I  = np.cross(w_I, r_I)             # club-head velocity

    # (3) keep history for tempo and plane PCA
    BUF_AXES.append(R_SI @ Z_S)           # shaft Dir. in world frame
    BUF_TS.append(ts)

    # (4) simple g-threshold marks impact
    if np.linalg.norm(a_S) > 30.0:        # adjust threshold as needed
        # --- tempo -----------------------------------------------------
        sign_flip = np.where(np.diff(np.signbit(w_I[2])))[0][0]
        Tb = ts - BUF_TS[0]
        Td = ts - BUF_TS[sign_flip]

        # --- swing plane via PCA -------------------------------------
        X = np.stack(BUF_AXES[sign_flip:])
        X = X - X.mean(axis=0)
        C = X.T @ X / (len(X) - 1)        # covariance 3 x 3
        eigval, eigvec = np.linalg.eigh(C)
        n_hat = eigvec[:, 0]              # normal = eigenvector min var
        beta = np.degrees(np.arccos(abs(n_hat @ Z_I)))

        # --- path, attack, face --------------------------------------
        v      = v_I
```

```
    alpha  = np.degrees(np.arcsin(v[2] / np.linalg.norm(v)))
    gamma  = np.degrees(np.arctan2(v[1], v[0]))
    f_I    = R_SI @ F_S
    delta  = np.degrees(np.arctan2(f_I[1], f_I[0]))

    # --- speed / power ------------------------------------------
    v_head = np.linalg.norm(v)
    g_peak = max(np.linalg.norm(a_S), 0)   # here just current packet
    F_cent = 0.205 * abs(w_I[2])**2 * abs(R_S[2])   # 205 g head mass

    # --- smoothness (jerk) --------------------------------------
    jerk = np.gradient([np.linalg.norm(a) for a in BUF_AXES],
                       1.0/FS, edge_order=2)
    S_smooth = np.trapz(jerk**2, dx=1.0/FS)

    emit_metrics(dict(Tb=Tb, Td=Td, beta=beta, alpha=alpha,
                      gamma=gamma, delta=delta, v_head=v_head,
                      g_peak=g_peak, F_cent=F_cent,
                      S_smooth=S_smooth, ts=ts))

    BUF_AXES.clear()
    BUF_TS.clear()
```

Conceptual flow

1. **Fusion** $\{\mathbf{a}_S, \boldsymbol{\omega}_S, \mathbf{m}_S\} \to q_{SI}$ using Madgwick/Kalman.
2. **Rigid kinematics** $\mathbf{v}_I = \boldsymbol{\omega}_I \times (R_{SI}\mathbf{r}_S)$.
3. **Buffers** collect shaft directions and time-stamps until impact.
4. **Impact** detected with a simple $|\mathbf{a}|$ threshold.
5. **Metrics** tempo, plane PCA, path, attack, face, power, quality.

No long-term integration of $\mathbf{a}$ is performed, so drift stays negligible.