ME 596 Spacecraft Attitude Dynamics and Control

Attitude Determination

Professor Christopher D. Hall
Mechanical Engineering

University of New Mexico

October 18, 2021

Attitude Determination

> *"In many spacecraft attitude systems, the attitude observations are naturally represented as unit vectors. Typical examples are the unit vectors giving the direction to the sun or a star and the unit vector in the direction of the Earth's magnetic field. Almost all algorithms for estimating spacecraft attitude from vector measurements are based on minimizing a loss function proposed in 1965 by Grace Wahba."*
>
> – F. Landis Markley and Daniele Mortari, "How to Estimate Attitude from Vector Observations," American Astronautical Society / AIAA Astrodynamics Specialists Conference, Girdwood, Alaska, August 1999

In this module, we develop attitude determination algorithms that use unit vector observations. The best of these algorithms are based on Wahba's loss function.

We will develop the Sun vector and Magnetic field vector, the TRIAD algorithm, Wahba's Loss Function, the $\mathbf{q}$-Method, and the QUEST algorithm.

The Basic Idea of Attitude Determination

A spacecraft must "know" where it is pointing, and that knowledge is provided by the Attitude Determination Subsystem.

- Combination of **sensors** and mathematical **models** generates data and **algorithm** estimates the attitude
- Need at least two of: Sun vector, Star vector, Earth horizon vector, Nadir vector, Magnetic field vector
- **Sensor** measures components of vector in body frame, $\mathcal{F}_b$
- **Model** uses spacecraft position to compute components in inertial frame, $\mathcal{F}_i$
- **Algorithm** determines rotation matrix such that (for example)

$$\mathbf{s}_b = \mathbf{R}^{bi}\mathbf{s}_i \quad \text{and} \quad \mathbf{m}_b = \mathbf{R}^{bi}\mathbf{m}_i$$

where $\hat{\mathbf{s}}$ is the Sun vector, and $\hat{\mathbf{m}}$ is the magnetic field vector.

What You Need To Know

- What types of sensors are commonly used and something about how they work
  – The sensor's function is to provide a unit vector measurement of a particular direction, usually expressed in $\mathcal{F}_b$
- Associated mathematical models and how to use them
  – The model's function is to provide the unit vector expressed in a different frame, usually $\mathcal{F}_i$
- What algorithms are commonly used, how they work, and how to use them
- **Typical problem:** A spacecraft's position at epoch is given, along with the types of sensors and their measurements in the body frame. Determine the attitude.

Why do we need to know the position to determine the attitude?

Sun Sensor

Sun sensors, using photocells, are commonly used on spacecraft because they are simple, reliable, and fairly accurate.

The Sun sensor provides a measurement of the direction vector from the spacecraft to the Sun, expressed in $\mathcal{F}_b$, by measuring the angle between the Sun direction and the normal vector of the sensor.

The (unit) sun direction vector can be written as

$$\hat{\mathbf{s}} = \mathbf{s}_i{}^{\mathsf{T}} \left\{ \hat{\mathbf{i}} \right\} = \mathbf{s}_b{}^{\mathsf{T}} \left\{ \hat{\mathbf{b}} \right\}$$

The sun direction and sensor normal directions are related by

$$\hat{\mathbf{s}} \cdot \hat{\mathbf{n}} = \cos \alpha$$

The angle α is measured through the current difference between two photocells.

A single pair of photocells cannot determine $\mathbf{s}_b$, as the angles of rotation about $\hat{\mathbf{s}}$ and $\hat{\mathbf{n}}$ cannot be determined

Sun Sensor (continued)

In left illustration, unit Sun vector is $\hat{\mathbf{s}}$ and normal to photocell is $\hat{\mathbf{n}}$. In right illustration, two photocells have unit normals in $\hat{\mathbf{n}} - \hat{\mathbf{t}}$ plane, and $\hat{\mathbf{n}}$ is normal to sensor pair.

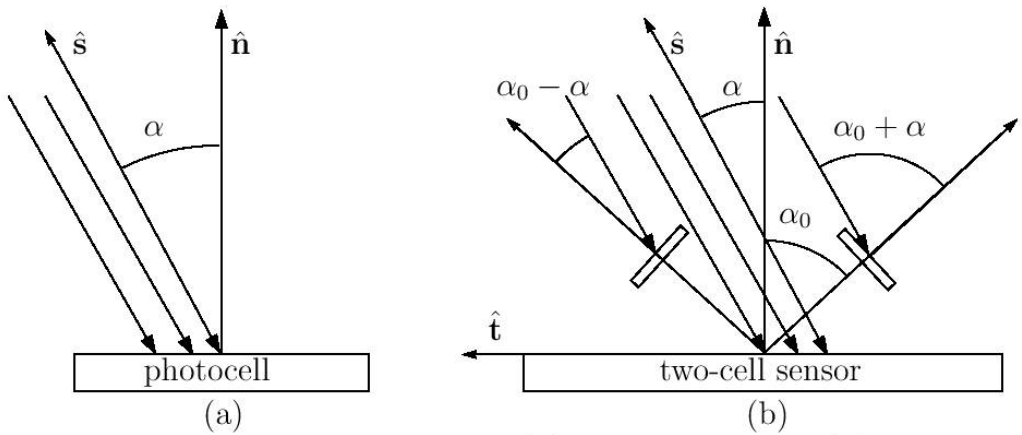


Figure 4.1: Photocells for Sun Sensors. (a) Single photocell. (b) Pair of photocells for measurement of α in $\hat{\mathbf{n}} - \hat{\mathbf{t}}$ plane.

Photocell produces current, I, proportional to $\cos\alpha$, where $\hat{\mathbf{s}} \cdot \hat{\mathbf{n}} = \cos\alpha$: $I(\alpha) = I(0)\cos\alpha$. Thus $I_1(\alpha) = I(0)\cos(\alpha_0 - \alpha)$, $I_2(\alpha) = I(0)\cos(\alpha_0 + \alpha)$, hence

$$\Delta I = I_2 - I_1 = I(0)\left[\cos(\alpha_0 + \alpha) - \cos(\alpha_0 - \alpha)\right] = 2I(0)\sin\alpha_0 \sin\alpha = C\sin\alpha$$

$$\Delta I = C\sin\alpha \Rightarrow \text{ current measurement provides } \alpha$$

Two Photocells = One Sun Sensor

Using two pairs of photocells, with $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$ perpendicular to $\hat{\mathbf{t}}$, measure two angles, α_1 and α_2.

Denote Sun sensor reference frame as $\mathcal{F}_s = \left\{\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{t}}\right\}$

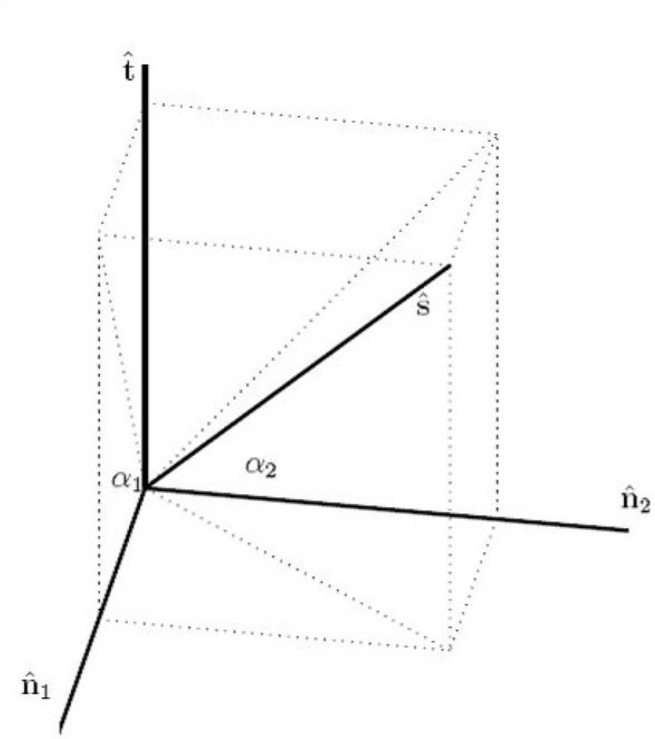
$\hat{\mathbf{n}}_1$ is normal vector for one pair of photocells, and $\hat{\mathbf{n}}_2$ is normal vector for second pair

Using α_1 and α_2, compute a non-unit vector Sun vector:

$$\mathbf{s}_s^* = \begin{bmatrix} 1 & \tan\alpha_1 / \tan\alpha_2 & \tan\alpha_1 \end{bmatrix}^{\mathsf{T}}$$

Normalize:

$$\mathbf{s}_s = \frac{\begin{bmatrix} 1 & \tan\alpha_1 / \tan\alpha_2 & \tan\alpha_1 \end{bmatrix}^{\mathsf{T}}}{\sqrt{{\mathbf{s}_s^*}^{\mathsf{T}} \mathbf{s}_s^*}}$$



Note well that this calculation is intended to illustrate the basic idea of Sun sensor measurement. In practice, internal calculations are more sophisticated.

Also, note that transformation from $\mathcal{F}_s$ to $\mathcal{F}_b$ is determined by orientation of Sun sensor on spacecraft: $\mathbf{s}_b = \mathbf{R}^{bs}\mathbf{s}_s$.

Sun Vector in Inertial Frame

A **mathematical model** for $\hat{\mathbf{s}}$ in $\mathcal{F}_i$ is required. That is, we need to know $\mathbf{s}_i$ as well as $\mathbf{s}_b$.

A convenient **algorithm** requires current time, expressed as Julian date, and returns position vector of Sun in $\mathcal{F}_i$.

We first present two algorithms for computing Julian date.

The first algorithm determines year, month, day, hour, minute and second from a two-line element set epoch.

The TLE epoch (line 1, columns 19–32) is of form `yyddd.ffffffff`, where `yy` is last two digits of the year, `ddd` is day of year (with January 1 being `001`), and `.ffffffff` is fraction of that day.

Thus, `01001.50000000` is noon universal time on January 1, 2001.

Sun Vector in Inertial Frame (continued)

Algorithm 4.1 (from Lecture Notes)

Given epoch from two-line element set, compute year, month, day, hour, minute and second.

Extract `yy`, `ddd`, and `.ffffffff`.

The year is either 19`yy` or 20`yy`, and for the next few decades the appropriate choice should be evident. Since old two-line element sets should normally not be used for current operations, the 19`yy` form is of historical interest only.

The day, `ddd`, begins with `001` as January 1, 20`yy`.

Example. Recent International Space Station TLE (from `celestrak.com`):

```
ISS (ZARYA)
1 25544U 98067A   18304.69640757  .00001215  00000-0  25814-4 0  9993
2 25544  51.6421  58.4548 0004250 357.2172 147.1380 15.53881671139746
```

`yy` = 18, so year = 2018. `ddd` = 304, 2018 is not leap year, so date = October 31. `0.69640757` (as fraction of day) = 16.71378168 hours. `0.71378168` (as fraction of hour) = `42.8269` minutes. `0.8269` as fraction of minute = `49.614048` seconds.

Thus, year=2018, month=10, day=31, hours=16, minutes=42, seconds=49.614048.

Sun Vector in Inertial Frame (continued)

Algorithm 4.2 (from Lecture Notes)

Julian date (JD) is time measured in days from epoch of January 1, 4713 BC, 1200 (noon).

Given year, month, day, hour, minute, second, compute the Julian date, JD.

$$\begin{aligned} \text{JD} \quad = \quad & 367\text{year} - \text{INT}\left\{\frac{7\left[\text{year} + \text{INT}\left(\frac{\text{month}+9}{12}\right)\right]}{4}\right\} \\ & + \text{INT}\left(\frac{275\text{month}}{9}\right) + \text{day} \\ & + 1,721,013.5 + \frac{\text{hour}}{24} + \frac{\text{minute}}{1440} + \frac{\text{s}}{86,400} \end{aligned}$$

Example. Using the data from previous example:

year=2018, month=10, day=31, hours=16, minutes=42, seconds=49.614048

Julian Date = 2,458,423.196407570 days

Note: In Matlab, the `fix` or `floor` functions can be used to calculate the `INT` functions in the algorithm. Note that `fix` rounds values towards zero, whereas `floor` rounds values to the nearest integer towards $-\infty$.

Sun Vector in Inertial Frame (continued)

Algorithm 4.3 (from Lecture Notes)

Given the Julian Date, JD, compute the distance to the Sun, s^*, and the unit vector direction to the Sun, $\mathbf{s}_i$, in $\mathcal{F}_i$.

$$T_{\text{UT1}} = \frac{\text{JD}_{\text{UT1}} - 2,451,545.0}{36,525}$$

Mean Longitude of Sun

$$\lambda_{M_{\text{Sun}}} = 280.460\,618\,4^\circ + 36,000.770\,053\,61 T_{\text{UT1}}$$

$$\text{Let } T_{\text{TDB}} \approx T_{\text{UT1}}$$

Mean Anomaly of Sun (with respect to Earth)

$$M_{\text{Sun}} = 357.527\,723\,3^\circ + 35,999.050\,34 T_{\text{TDB}}$$

Ecliptic Longitude of Sun

$$\lambda_{\text{ecliptic}} = \lambda_{M_{\text{Sun}}} + 1.914\,666\,471^\circ \sin M_{\text{Sun}} + 0.918\,994\,643 \sin 2M_{\text{Sun}}$$

UT1 is Univeral Time, which is the same as Greenwich Mean Time.
TDB is Barycentric Dynamic Time, one of many "times" used in astronomy.

Sun Vector in Inertial Frame (continued)

Algorithm 4.3 (from Lecture Notes, continued)

Distance to Sun in AUs

$$s^* = 1.000\,140\,612 - 0.016\,708\,617 \cos M_{\text{Sun}} - 0.000\,139\,589 \cos 2M_{\text{Sun}}$$

Elevation angle to Sun in degrees

$$\varepsilon = 23.439\,291^\circ - 0.013\,004\,2 T_{\text{TDB}}$$

$$\mathbf{s}_i = \begin{bmatrix} \cos \lambda_{\text{ecliptic}} \\ \cos \varepsilon \sin \lambda_{\text{ecliptic}} \\ \sin \varepsilon \sin \lambda_{\text{ecliptic}} \end{bmatrix}$$

The 3×1 matrix $\mathbf{s}_i$ is unit sun vector expressed in $\mathcal{F}_i$ at epoch.

Example of use: Given $\mathbf{s}_b$ from Sun sensor measurements, and $\mathbf{s}_i$ from Algorithm 4.3, find $\mathbf{R}^{bi}$ such that $\mathbf{s}_b = \mathbf{R}^{bi}\mathbf{s}_i$.

Note, however, that we must have two independent measurements, so another attitude sensor is required.

Magnetometers

A magnetometer measures the vector direction of Earth's magnetic field, so that it returns the components of $\vec{\mathbf{m}}$ (also denoted $\vec{\mathbf{B}}$) in the magnetometer frame, $\mathcal{F}_m$. Note that the orientation of $\mathcal{F}_m$ with respect to $\mathcal{F}_b$ is fixed by the magnetometer's orientation in the spacecraft.

Model for $\mathbf{m}_i$ (all vectors in $\mathcal{F}_i$):

$$\mathbf{m}_i^* = \frac{R_\oplus^3 H_0}{r^3}\left[3\mathbf{d}^\mathsf{T}\hat{\mathbf{r}}\hat{\mathbf{r}} - \mathbf{d}\right]$$

$$\mathbf{m}_i^* = \frac{R_\oplus^3 H_0}{r^3}\begin{bmatrix} 3(\mathbf{d}^\mathsf{T}\hat{\mathbf{r}})\hat{r}_1 - \sin\theta'_m\cos\alpha_m \\ 3(\mathbf{d}^\mathsf{T}\hat{\mathbf{r}})\hat{r}_2 - \sin\theta'_m\sin\alpha_m \\ 3(\mathbf{d}^\mathsf{T}\hat{\mathbf{r}})\hat{r}_3 - \cos\theta'_m \end{bmatrix}$$

where the vector $\hat{\mathbf{d}}$ is the unit dipole direction, with components in the inertial frame:

$$\mathbf{d}_i = \begin{bmatrix} \sin\theta'_m\cos\alpha_m \\ \sin\theta'_m\sin\alpha_m \\ \cos\theta'_m \end{bmatrix}$$

Symbols used above are defined on next slide.

Magnetometers (continued)

Vector $\hat{\mathbf{r}}$ is a unit vector in direction of position vector of spacecraft: $\hat{\mathbf{r}} = \mathbf{r}/r$.

Constants $R_\oplus = 6378$ km and $H_0 = 30,115$ nT are radius of Earth and a constant characterizing Earth's magnetic field, respectively. Note that H_0 is continually updated.

In these expressions, $\alpha_m = \theta_{g0} + \omega_\oplus t + \phi'_m$
where θ_{g0} is the Greenwich sidereal time at epoch, $\omega_\oplus$ is the average rotation rate of the Earth, t is the time since epoch, and θ'_m and ϕ'_m are the coelevation and East longitude of the dipole, both of which slowly change with time.

Recent accepted values of these angles are $\theta'_m = 196.54°$ and $\phi'_m = 108.43°$.

Example of use: Given $\mathbf{s}_b$ from Sun sensor measurement, $\mathbf{s}_i$ from Algorithm 4.3, $\mathbf{m}_b$ from magnetometer measurement, and $\mathbf{m}_i$ from model above, find $\mathbf{R}^{bi}$ such that $\mathbf{s}_b = \mathbf{R}^{bi}\mathbf{s}_i$ and $\mathbf{m}_b = \mathbf{R}^{bi}\mathbf{m}_i$.

Typical Attitude Sensor Accuracies

Table: Sensor Accuracy Ranges

Sensor	**Accuracy**	**Characteristics and Applicability**
Magnetometers	1.0° (5000 km alt) 5.0° (200 km alt)	Attitude measured relative to Earth's local magnetic field. Magnetic field uncertainties and variability dominate accuracy. Usable only below $\sim$ 6,000 km.
Earth sensors	0.05° (GEO) 0.1° (LEO)	Horizon uncertainties dominate accuracy. Highly accurate units use scanning.
Sun sensors	0.01°	Typical field of view $\pm 30^\circ$
Star sensors	2 arc-sec	Typical field of view $\pm 6^\circ$
Gyroscopes	0.001 deg/hr	Requires periodically resetting reference
Directional antennas	0.01° to 0.5°	Typically 1% of the antenna beamwidth

Adapted from Larson and Wertz

TRIAD: A deterministic algorithm

Suppose we have a Sun sensor measurement $\mathbf{s}_b$ and associated model calculation $\mathbf{s}_i$, as well as a magnetometer measurement $\mathbf{m}_b$ and associated model calculation $\mathbf{m}_i$.

The TRIAD algorithm lets us construct the rotation matrix $\mathbf{R}^{bi}$ such that *either* $\mathbf{s}_b = \mathbf{R}^{bi}\mathbf{s}_i$ *or* $\mathbf{m}_b = \mathbf{R}^{bi}\mathbf{m}_i$. Question: Why can we not usually have both equalities be true?

Construct a triad of unit vectors of $\mathcal{F}_t$ ("t" denoting "temporary") with components known in $\mathcal{F}_b$ and in $\mathcal{F}_i$. Choose $\hat{\mathbf{t}}_1$ to be *either* $\hat{\mathbf{s}}$ *or* $\hat{\mathbf{m}}$.

$$\begin{aligned} \hat{\mathbf{t}}_1 &= \hat{\mathbf{s}} \\ \mathbf{t}_{1b} &= \mathbf{s}_b \\ \mathbf{t}_{1i} &= \mathbf{s}_i \end{aligned} \qquad \begin{aligned} \hat{\mathbf{t}}_2 &= \hat{\mathbf{s}} \times \hat{\mathbf{m}} \\ \mathbf{t}_{2b} &= \frac{\mathbf{s}_b^{\times}\mathbf{m}_b}{|\mathbf{s}_b^{\times}\mathbf{m}_b|} \\ \mathbf{t}_{2i} &= \frac{\mathbf{s}_i^{\times}\mathbf{m}_i}{|\mathbf{s}_i^{\times}\mathbf{m}_i|} \end{aligned} \qquad \begin{aligned} \hat{\mathbf{t}}_3 &= \hat{\mathbf{t}}_1 \times \hat{\mathbf{t}}_2 \\ \mathbf{t}_{3b} &= \mathbf{t}_{1b}^{\times}\mathbf{t}_{2b} \\ \mathbf{t}_{3i} &= \mathbf{t}_{1i}^{\times}\mathbf{t}_{2i} \end{aligned}$$

Then $\mathbf{R}^{bt} = [\mathbf{t}_{1b} \;\; \mathbf{t}_{2b} \;\; \mathbf{t}_{3b}]$, $\mathbf{R}^{ti} = [\mathbf{t}_{1i} \;\; \mathbf{t}_{2i} \;\; \mathbf{t}_{3i}]^{\mathsf{T}}$, and $\mathbf{R}^{bi} = \mathbf{R}^{bt}\mathbf{R}^{ti}$.

That's it!

TRIAD Example

Example 4.2 Suppose a spacecraft has two attitude sensors that provide the following measurements of the two vectors $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$:

$$\begin{aligned} \mathbf{v}_{1b} &= [0.8273 \;\; 0.5541 \;\; -0.0920]^{\mathsf{T}} \\ \mathbf{v}_{2b} &= [-0.8285 \;\; 0.5522 \;\; -0.0955]^{\mathsf{T}} \end{aligned}$$

These vectors have known inertial frame components of

$$\begin{aligned} \mathbf{v}_{1i} &= [-0.1517 \;\; -0.9669 \;\; 0.2050]^{\mathsf{T}} \\ \mathbf{v}_{2i} &= [-0.8393 \;\; 0.4494 \;\; -0.3044]^{\mathsf{T}} \end{aligned}$$

Applying the TRIAD algorithm, we construct the components of the vectors $\hat{\mathbf{t}}_j, j = 1, 2, 3$ in both the body and inertial frames:

$$\begin{aligned} \mathbf{t}_{1b} &= [0.8273 \;\; 0.5541 \;\; -0.0920]^{\mathsf{T}} \\ \mathbf{t}_{2b} &= [-0.0023 \;\; 0.1671 \;\; 0.9859]^{\mathsf{T}} \\ \mathbf{t}_{3b} &= [0.5617 \;\; -0.8155 \;\; 0.1395]^{\mathsf{T}} \end{aligned}$$

and

$$\begin{aligned} \mathbf{t}_{1i} &= [-0.1517 \;\; -0.9669 \;\; 0.2050]^{\mathsf{T}} \\ \mathbf{t}_{2i} &= [0.2177 \;\; -0.2350 \;\; -0.9473]^{\mathsf{T}} \\ \mathbf{t}_{3i} &= [0.9641 \;\; -0.0991 \;\; 0.2462]^{\mathsf{T}} \end{aligned}$$

TRIAD Example (continued)

Substitute these results into the two rotation matrices $\mathbf{R}^{bt} = [\mathbf{t}_{1b} \;\; \mathbf{t}_{2b} \;\; \mathbf{t}_{3b}]$, and $\mathbf{R}^{ti} = [\mathbf{t}_{1i} \;\; \mathbf{t}_{2i} \;\; \mathbf{t}_{3i}]^{\mathsf{T}}$. Then calculate the desired result:

$$\mathbf{R}^{bi} = \mathbf{R}^{bt}\mathbf{R}^{ti} = \begin{bmatrix} 0.4156 & -0.8551 & 0.3100 \\ -0.8339 & -0.4943 & -0.2455 \\ 0.3631 & -0.1566 & -0.9185 \end{bmatrix}$$

Applying this rotation matrix to $\mathbf{v}_{1i}$ gives $\mathbf{v}_{1b}$ exactly, because we used $\hat{\mathbf{v}}_1$ as the "exact" vector in the formulation; however, applying it to $\mathbf{v}_{2i}$ does not give $\mathbf{v}_{2b}$ exactly.

If we know *a priori* that sensor 2 is more accurate than sensor 1, then we can use $\hat{\mathbf{v}}_2$ as the exact measurement, hopefully leading to a more accurate estimate of $\mathbf{R}^{bi}$.

Try your hand at making up your own TRIAD problems. Make up an $\mathbf{R}^{bi}$, make up two "sensor" vectors in $\mathcal{F}_i$, then rotate them into $\mathcal{F}_b$. Then you have the inputs for a TRIAD problem, that should solve both conditions exactly. Add some "noise" to the vectors in $\mathcal{F}_b$ and see how it affects your TRIAD solution.

Hint: The `rand` function in `Matlab` is useful in creating example problems.

Other Attitude Determination Methods

Suppose we have more than two vectors. In this case, TRIAD won't work without modification.

As noted in the quote at the beginning of this module, "Almost all algorithms for estimating spacecraft attitude from vector measurements are based on minimizing a loss function proposed in 1965 by Grace Wahba."

Wahba's Problem: Minimize the "loss function"

$$J\left(\mathbf{R}^{bi}\right) = \frac{1}{2}\sum_{k=1}^{N} w_k \left|\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}\right|^2$$

J is a non-negative scalar that is a function of a rotation matrix applied to the given vectors. The w_k are positive "weights" to be chosen by the attitude determination analyst[1]. In the sequel, we investigate Wahba's Problem using

- Numerical optimization
- $\bar{\mathbf{q}}$-Method
- QUEST

[1] See § 5.5 of Markley and Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*, Springer, 2014.

Wahba's Loss Function

Suppose we have ≥ 2 vectors, $\hat{\mathbf{v}}_k, k = 1, ..., N$ in frames $\mathcal{F}_i$ and $\mathcal{F}_b$

Ideal attitude determination is finding the rotation matrix $\mathbf{R}^{bi}$ so that

$$\mathbf{v}_{kb} = \mathbf{R}^{bi}\mathbf{v}_{ki}, \quad k = 1, ..., N$$

Note that subscript k is index for individual vectors, whereas b and i subscripts denote the frames in which vectors are represented.

The ideal situation does not generally exist. Sensor errors mean that the ideal relationship above is not possible. That is, for any choice $\mathbf{R}^{bi}$, at least *some* of the vectors will not satisfy the ideal relationship, so for some value of k, $|\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}|$ will not be zero. (Compare with TRIAD results.)

One engineering solution to using noisy data effectively is the Least-Squares method. Grace Wahba posed least-squares for attitude determination in 1966.

Wahba's "loss function":

$$J\left(\mathbf{R}^{bi}\right) = \frac{1}{2}\sum_{k=1}^{N} w_k \left|\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}\right|^2$$

G. Wahba. "A Least-Squares Estimate of Satellite Attitude," Problem 65.1. SIAM Review, pp. 385-386, July 1966.

Wahba's Loss Function (continued)

Suppose we have ≥ 2 vectors, $\hat{\mathbf{v}}_k, k = 1, ..., N$ in frames $\mathcal{F}_i$ and $\mathcal{F}_b$

Ideal attitude determination is finding the rotation matrix $\mathbf{R}^{bi}$ so that

$$\mathbf{v}_{kb} = \mathbf{R}^{bi}\mathbf{v}_{ki}, \quad k = 1, ..., N$$

Note that subscript k is index for individual vectors, whereas b and i subscripts denote the frames in which vectors are represented.

The ideal situation does not generally exist. Sensor errors mean that the ideal relationship above is not possible. One engineering solution to using noisy data effectively is the Least-Squares method.

Grace Wahba posed least-squares for attitude determination in 1966.

Wahba's "loss function" is the sum of

$$J\left(\mathbf{R}^{bi}\right) = \frac{1}{2}\sum_{k=1}^{N} w_k \left|\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}\right|^2$$

Minimizing the Loss Function

Recall that to minimize a function, you differentiate with respect to the independent variable and set equal to zero.

$$J\left(\mathbf{R}^{bi}\right) = \frac{1}{2}\sum_{k=1}^{N} w_k \left|\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}\right|^2$$

In the loss function, the independent variable is the attitude, $\mathbf{R}^{bi}$, which can be expressed in terms of Euler angles, Euler axis and angle, or quaternions. Thus,

$$J(\mathbf{R}^{bi}) = J(\boldsymbol{\theta}) = J(\mathbf{a}, \Phi) = J(\bar{\mathbf{q}})$$

We could, for example, use two of the given vectors with TRIAD algorithm, then extract $\boldsymbol{\theta}$, $(\mathbf{a}, \Phi)$, or $\bar{\mathbf{q}}$ from resulting estimate for $\mathbf{R}^{bi}$, and use this approximation as an initial "guess" in search for the attitude that minimizes J.

In multivariable differentiation, we calculate the gradient of the function with respect to the independent variables. For example, if we use Euler angles, we differentiate J as follows:

$$\nabla J = \begin{bmatrix}\partial J/\partial\theta_1 & \partial J/\partial\theta_2 & \partial J/\partial\theta_3\end{bmatrix}^{\mathsf{T}} = \mathbf{0}$$

In principle, we can carry out derivatives (recall Euler angle versions of $\mathbf{R}^{bi}$).

Minimizing J (Newton's Method, Euler Angles)

Solving $\nabla J = \mathbf{0}$ requires solving three nonlinear equations, each of which depends (in general) on all three Euler angles. Together the three scalar functions comprise a single "vector" function of a "vector" variable.

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial \theta_2} = \frac{\partial J}{\partial \theta_3} \quad = \quad 0$$

First: Apply Newton's method for finding root of scalar function of scalar variable, $F(x) = 0$. Denote (unknown) solution by x^*, and initial guess by x_n.

$$F(x^*) = F(x_n + \Delta x) = F(x_n) + \frac{\partial F}{\partial x}(x_n)\Delta x + \mathcal{O}(\Delta x^2) = 0$$

Assume x_n is a close enough guess so that higher order terms are negligible and solve for the correction, Δx.

$$\Delta x = -\left[\frac{\partial F}{\partial x}(x_n)\right]^{-1} F(x_n)$$

A hopefully closer estimate is

$$x_{n+1} = x_n - \left[\frac{\partial F}{\partial x}(x_n)\right]^{-1} F(x_n)$$

Repeat until "close enough."

Minimizing J (continued)

Now, consider a "vector" function, $\mathbf{F}(\mathbf{x}) = \mathbf{0}$:

$$\mathbf{F} = \nabla J = \frac{\partial J}{\partial \mathbf{x}} = \left[\frac{\partial J}{\partial x_1} \quad \frac{\partial J}{\partial x_2} \quad \frac{\partial J}{\partial x_3}\right]^{\mathsf{T}}$$

Newton's method for vector functions of vector variables is:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left[\frac{\partial \mathbf{F}}{\partial \mathbf{x}}(\mathbf{x}_n)\right]^{-1} \mathbf{F}(\mathbf{x}_n)$$

The expression $\partial \mathbf{F}/\partial \mathbf{x}$ represents a 3×3 *Jacobian* matrix whose elements are

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \frac{\partial^2 J}{\partial x_1 \partial x_3} \\ \frac{\partial^2 J}{\partial x_2 \partial x_1} & \frac{\partial^2 J}{\partial x_2^2} & \frac{\partial^2 J}{\partial x_2 \partial x_3} \\ \frac{\partial^2 J}{\partial x_3 \partial x_1} & \frac{\partial^2 J}{\partial x_3 \partial x_2} & \frac{\partial^2 J}{\partial x_3^2} \end{bmatrix}$$

Calculating all these derivatives is possible, but tedious.

Normally one uses numerical differentiation, i.e., a central or one-sided finite difference scheme.

The next slide presents the numerical differentiation calculations.

Minimizing J (continued)

Use numerical differentiation to compute first and second partial derivatives of J with respect to the independent variables ($x_i = \theta_i$).

The first derivatives, e.g., $\partial J/\partial x_1$ are computed by first calculating $J(\mathbf{x})$ for the current guess of x_1, x_2, and x_3. Remember that J is a scalar that depends on the attitude ($\mathbf{x}$), the attitude sensor and model vectors, and the weights w_k.

Then a small value is added to x_1, $x_1 + \delta x_1$, and J is recalculated, leading to:

$$\frac{\partial J}{\partial x_1} \approx \frac{J(x_1 + \delta x_1, x_2, x_3) - J(x_1, x_2, x_3)}{\delta x_1}$$

This approach is applied to compute all the derivatives, including the second derivatives in the Jacobian.

To compute second derivative $\partial^2 J/(\partial x_1 \partial x_2)$, denote $\partial J/\partial x_1$ by J_{x_1}, and use:

$$\frac{\partial^2 J}{\partial x_1 \partial x_2} \approx \frac{J_{x_1}(x_1, x_2 + \delta x_2, x_3) - J_{x_1}(x_1, x_2, x_3)}{\delta x_2}$$

Similar steps are repeated to calculate ∇J and the Jacobian.

Fortunately, an elegant solution to Wahba's Problem exists and we do not need to apply the extremely important Newton's method (but you should be well aware of the method!).

Minimizing $J(\bar{\mathbf{q}})$ ($\bar{\mathbf{q}}$-Method)

Grace Wahba posed her problem as a mathematical puzzle, and many solutions have been developed. We will consider the analytical solution developed by Paul Davenport. First, expand the loss function:

$$\begin{aligned} J\left(\mathbf{R}^{bi}\right) &= \frac{1}{2}\sum_{k=1}^{N} w_k \left|\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}\right|^2 \\ &= \frac{1}{2}\sum w_k (\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki})^{\mathsf{T}}(\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}) \\ &= \frac{1}{2}\sum w_k (\mathbf{v}_{kb}^{\mathsf{T}}\mathbf{v}_{kb} + \mathbf{v}_{ki}^{\mathsf{T}}\mathbf{v}_{ki} - 2\mathbf{v}_{kb}^{\mathsf{T}}\mathbf{R}^{bi}\mathbf{v}_{ki}) \end{aligned}$$

Vectors are normalized to unity, so $\mathbf{v}_{kb}^{\mathsf{T}}\mathbf{v}_{kb} = \mathbf{v}_{ki}^{\mathsf{T}}\mathbf{v}_{ki} = 1$. Therefore, the loss function becomes

$$J = \sum w_k (1 - \mathbf{v}_{kb}^{\mathsf{T}}\mathbf{R}^{bi}\mathbf{v}_{ki})$$

Minimizing $J(\mathbf{R})$ is the same as minimizing $J'(\mathbf{R}) = -\sum w_k \mathbf{v}_{kb}^{\mathsf{T}}\mathbf{R}^{bi}\mathbf{v}_{ki}$ or *maximizing* the **gain** function

$$g(\mathbf{R}) = \sum w_k \mathbf{v}_{kb}^{\mathsf{T}}\mathbf{R}^{bi}\mathbf{v}_{ki}$$

Make sure you know what all the subscripts and superscripts mean here.

Minimizing $J(\bar{\mathbf{q}})$ (continued)

Recall that $J(\mathbf{R}^{bi}) = J(\bar{\mathbf{q}})$, so $g(\mathbf{R}^{bi}) = g(\bar{\mathbf{q}})$.

For the $\bar{\mathbf{q}}$-Method, we restate problem in terms of $\bar{\mathbf{q}} = \begin{bmatrix}\mathbf{q}^{\mathsf{T}} & q_4\end{bmatrix}^{\mathsf{T}}$, for which

$$\mathbf{R} \quad = \quad (q_4^2 - \mathbf{q}^{\mathsf{T}}\mathbf{q})\mathbf{1} + 2\mathbf{q}\mathbf{q}^{\mathsf{T}} - 2q_4\mathbf{q}^{\times}$$

We substitute this expression into the gain function. The algebra is straightforward and eventually leads to

$$g(\bar{\mathbf{q}}) \quad = \quad \bar{\mathbf{q}}^{\mathsf{T}}\mathbf{K}\bar{\mathbf{q}}$$

where $\mathbf{K}$ is a symmetric 4×4 matrix given by

$$\begin{aligned}
\mathbf{K} &= \begin{bmatrix} \mathbf{S} - \sigma\mathbf{I} & \mathbf{Z} \\ \mathbf{Z}^{\mathsf{T}} & \sigma \end{bmatrix} \\
\text{with } \mathbf{B} &= \sum_{k=1}^{N} w_k \left(\mathbf{v}_{kb}\mathbf{v}_{ki}^{\mathsf{T}}\right) \\
\mathbf{S} &= \mathbf{B} + \mathbf{B}^{\mathsf{T}} \\
\mathbf{Z} &= \begin{bmatrix} B_{23} - B_{32} & B_{31} - B_{13} & B_{12} - B_{21} \end{bmatrix}^{\mathsf{T}} \\
\sigma &= \mathrm{tr}\,[\mathbf{B}]
\end{aligned}$$

Note well that $\mathbf{K}$ depends entirely on the sensor/model vectors and the weights.

Minimizing $J(\bar{\mathbf{q}})$ (continued)

We have redefined Wahba's problem as one of maximizing $g(\mathbf{R}^{bi})$, and by substituting the expression for $\mathbf{R}(\bar{\mathbf{q}})$ into g, we have

$$g(\bar{\mathbf{q}}) = \bar{\mathbf{q}}^{\mathsf{T}}\mathbf{K}\bar{\mathbf{q}}$$

Note that the maximum of this function is ∞, since if we choose larger and larger magnitudes of $\bar{\mathbf{q}}$, g becomes larger and larger.

However, recall that $\bar{\mathbf{q}}$ is a 4×1 matrix, and that three parameters are the minimum required to determine attitude uniquely. Any four-parameter representation of attitude has a single constraint relating the parameters.

For quaternions, the constraint is

$$\bar{\mathbf{q}}^{\mathsf{T}}\bar{\mathbf{q}} \quad = \quad 1$$

This constraint means that the maximum of g is not ∞, since $\bar{\mathbf{q}}$ is bounded.

In solving the maximization problem, we have to incorporate the constraint, using a Lagrange multiplier, λ. If you don't remember this concept from calculus, I recommend that you dust off your Calculus textbook, or visit the `Wikipedia` page for `Lagrange multiplier`.

Minimizing $J(\bar{\mathbf{q}})$ (continued)

To maximize g, we differentiate with respect to $\bar{\mathbf{q}}$.

Since the quaternion elements are not independent the constraint must also be satisfied.

Adding the constraint to the gain function with a Lagrange multiplier yields a new function, sometimes called the *Lagrangian function*:

$$g'(\bar{\mathbf{q}}) \quad = \quad \bar{\mathbf{q}}^{\mathsf{T}}\mathbf{K}\bar{\mathbf{q}} - \lambda(\bar{\mathbf{q}}^{\mathsf{T}}\bar{\mathbf{q}} - 1))$$

Differentiating g' with respect to $\bar{\mathbf{q}}$, and setting it equal to zero leads to

$$\mathbf{K}\bar{\mathbf{q}} \quad = \quad \lambda\bar{\mathbf{q}}$$

where $\mathbf{K}$ is known, and both λ and $\bar{\mathbf{q}}$ are unknown.

You should recognize this equation as an Eigenproblem, which gives us license to go on an eigenhunt!

Observation: The quaternion (attitude) that minimizes Wahba's loss function is an eigenvector of the 4×4 matrix $\mathbf{K}$.

Minimizing $J(\bar{\mathbf{q}})$ (continued)

The Lagrangian gain function $g'(\bar{\mathbf{q}})$ has a stationary value (maximum, minimum, or inflection) when

$$\mathbf{K}\bar{\mathbf{q}} = \lambda\bar{\mathbf{q}}$$

Since $\mathbf{K}$ is known, and both λ and $\bar{\mathbf{q}}$ are unknown, this equation defines an Eigenvalue problem.

Since $\mathbf{K}$ is a 4×4 matrix, there are in general four eigenvalues and four eigenvectors. Furthermore, since $\mathbf{K}$ is a real symmetric matrix, the eigenvalues are all real, and the eigenvectors are all real and orthogonal. As a consequence, $\mathbf{K}$ can be written as

$$\mathbf{K} = \sum_{j=1}^{4} \lambda_j \bar{\mathbf{q}}_j \bar{\mathbf{q}}_j{}^{\mathsf{T}}$$

where λ_j are the eigenvalues, and $\bar{\mathbf{q}}_j$ are the eigenvectors.

We can, without any loss of generality, order the eigenvalues and eigenvectors such that $\lambda_{\text{max}} = \lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$.

Question: How do we choose the "right" eigenvalue eigenvector pair?

Minimizing $J(\bar{\mathbf{q}})$ (continued)

We know that the solution to Wahba's problem lies in choosing the "right" eigenvector of the 4×4 real symmetric matrix $\mathbf{K}$.

Recall that the gain function (before adding the constraint and Lagrange multiplier) is

$$g(\bar{\mathbf{q}}) = \bar{\mathbf{q}}^{\mathsf{T}} \mathbf{K} \bar{\mathbf{q}}$$

Each of the four solutions to the eigenvalue problem satisfies

$$\mathbf{K}\bar{\mathbf{q}} = \lambda \bar{\mathbf{q}}$$

Substitute the solution into the gain function:

$$g(\bar{\mathbf{q}}) = \bar{\mathbf{q}}^{\mathsf{T}} \mathbf{K} \bar{\mathbf{q}} = \bar{\mathbf{q}}^{\mathsf{T}} \lambda \bar{\mathbf{q}} = \lambda \bar{\mathbf{q}}^{\mathsf{T}} \bar{\mathbf{q}} = \lambda$$

This equality is true for each $(\lambda, \bar{\mathbf{q}})$ pair for $\mathbf{K}$. Thus, the eigenvalue that maximizes $g(\bar{\mathbf{q}})$ is the **largest eigenvalue** of $\mathbf{K}$.

The eigenvector $\bar{\mathbf{q}}$ corresponding to the largest eigenvalue λ_{max} of $\mathbf{K}$ is the least-squares optimal estimate of the attitude.

`Matlab` **note:** The function `eig(K)` returns eigenvalues and eigenvectors of $\mathbf{K}$.

q-Method Illustrative Example

For this example, we will construct the data required to apply the $\bar{\mathbf{q}}$-Method.

Use `Matlab`'s `rand` function to generate two vectors in $\mathcal{F}_i$:

$$\mathbf{v}_{1i} = \begin{bmatrix} 0.2673 \\ 0.5345 \\ 0.8018 \end{bmatrix} \mathbf{v}_{2i} = \begin{bmatrix} -0.3124 \\ 0.9370 \\ 0.1562 \end{bmatrix}$$

The "exact" attitude is defined by a 3-1-3 Euler angle sequence with a 30° rotation for each angle. The true attitude is represented by the rotation matrix between $\mathcal{F}_i$ and $\mathcal{F}_b$:

$$\mathbf{R}^{bi}_{exact} = \begin{bmatrix} 0.5335 & 0.8080 & 0.2500 \\ -0.8080 & 0.3995 & 0.4330 \\ 0.2500 & -0.4330 & 0.8660 \end{bmatrix}$$

We will add some random "noise" to the $\mathcal{F}_b$ vectors, and then compare the TRIAD and $\bar{\mathbf{q}}$-Method solutions for $\mathbf{R}^{bi}$.

q-Method Illustrative Example (continued)

If sensors measured without error, then in $\mathcal{F}_b$ the vectors would be:

$$\mathbf{v}_{1b_{exact}} = \begin{bmatrix} 0.7749 \\ 0.3448 \\ 0.5297 \end{bmatrix} \mathbf{v}_{2b_{exact}} = \begin{bmatrix} 0.6296 \\ 0.6944 \\ -0.3486 \end{bmatrix}$$

Sensor measurements are not perfect however, and to model this uncertainty we introduce some error into the body-frame sensor measurements. A uniformly distributed random error is added to the sensor measurements, with a maximum value of $\pm 5^\circ$. The two "measured" vectors are:

$$\mathbf{v}_{1b} = \begin{bmatrix} 0.7814 \\ 0.3751 \\ 0.4987 \end{bmatrix} \mathbf{v}_{2b} = \begin{bmatrix} 0.6163 \\ 0.7075 \\ -0.3459 \end{bmatrix}$$

Thus, we have the two $\mathcal{F}_i$ vectors and these two $\mathcal{F}_b$ vectors.

We will apply TRIAD and the $\bar{\mathbf{q}}$-Method.

Note that it's important in calculations to normalize vectors. If you check the norm of, e.g., $\mathbf{v}_{1b}$, it isn't unity.

q-Method Illustrative Example (continued)

Using the TRIAD algorithm and assuming $\hat{\mathbf{v}}_1$ is the "exact" vector, the satellite attitude is estimated to be:

$$\mathbf{R}_{triad}^{bi} = \begin{bmatrix} 0.5662 & 0.7803 & 0.2657 \\ -0.7881 & 0.4180 & 0.4518 \\ 0.2415 & -0.4652 & 0.8516 \end{bmatrix}$$

A useful approach to measuring the quality of the attitude estimate makes use of the orthonormal nature of the rotation matrix; *i.e.*, $\mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{1}$. Since the TRIAD algorithm's estimate is not perfect, we compare the following to the identity matrix:

$$\mathbf{R}_{error}^{bi} = \mathbf{R}_{triad}^{bi^{\mathsf{T}}}\mathbf{R}_{exact}^{bi} = \begin{bmatrix} 0.9992 & 0.03806 & 0.0094 \\ -0.0378 & 0.9989 & -0.0268 \\ -0.0104 & 0.02645 & 0.9996 \end{bmatrix}$$

If $\mathbf{R}_{triad}^{bi}$ were perfect, then the result would be the identity matrix.

However, $\mathbf{R}_{error}^{bi}$ is in fact the rotation matrix that relates the attitude estimated by TRIAD with the "exact" attitude, and corresponds to an Euler axis / angle. The Euler angle Φ can be used to characterize the error.

q-Method Illustrative Example (continued)

$\mathbf{R}_{error}^{bi}$ is the rotation matrix relating the exact attitude to the attitude estimated by the Triad algorithm. The principal Euler angle of this matrix, and therefore the attitude error of the estimate, is $\Phi = 2.72^{\circ}$. For later comparison, the loss function is $J(\mathbf{R}_{triad}^{bi}) = 7.3609 \times 10^{-4}$.

Using the $\bar{\mathbf{q}}$-Method with the same inertial and measured vectors, and with weights $w_k = 1$, produces the $\mathbf{K}$ matrix:

$$\mathbf{K} = \begin{bmatrix} -1.1929 & 0.8744 & 0.9641 & 0.4688 \\ 0.8744 & 0.5013 & 0.3536 & -0.4815 \\ 0.9641 & 0.3536 & -0.5340 & 1.1159 \\ 0.4688 & -0.4815 & 1.1159 & 1.2256 \end{bmatrix}$$

The largest eigenvalue and corresponding eigenvector of $\mathbf{K}$ are:

$$\lambda_{max} = 1.9996 \qquad \bar{\mathbf{q}} = [0.2643 \quad -0.0051 \quad 0.4706 \quad 0.8418]^{\mathsf{T}}$$

The corresponding rotation matrix is:

$$\mathbf{R}_{q}^{bi} = \begin{bmatrix} 0.5570 & 0.7896 & 0.2575 \\ -0.7951 & 0.4173 & 0.4402 \\ 0.2401 & -0.4499 & 0.8602 \end{bmatrix}$$

$\mathbf{q}$-Method Illustrative Example (continued)

As we did for the TRIAD solution, we determine the quality of this solution by computing the Euler angle of $\mathbf{R}_q^{bi^\mathsf{T}} \mathbf{R}_{exact}^{bi}$, as well as the loss function $J(\mathbf{R}_q^{bi})$.

For the $\mathbf{q}$-Method estimate of attitude, the attitude error and loss function values are:

$$\begin{aligned} \Phi &= 1.76^\circ \\ J &= 3.6954 \times 10^{-4} \end{aligned}$$

As expected, the $\mathbf{q}$-Method finds the attitude matrix which minimizes the loss function. The attitude error is also lower than for the TRIAD algorithm.

However, other measurement errors could create a case where the attitude error is actually larger for the $\mathbf{q}$-Method, even though it minimizes the loss function. We return to this point after the QUEST algorithm example.

QUEST is a fast algorithm for finding $\mathbf{q}$ without having to use an eigenvalue solver (which is computationally expensive).

Recommendation: Use the approach from this example to create another example, with different initial vectors, different "exact" rotation, and so forth. Share (optional!) your examples and observations in the Discussion Forum.

QUaternion ESTimator (QUEST)

In 1981, Malcolm Shuster published a novel application of the $\mathbf{q}$-Method, known as QUEST. QUEST is the standard attitude estimation algorithm used in most spacecraft today.

Recall that the least-squares optimal attitude minimizes Wahba's loss function

$$\begin{aligned} J &= \frac{1}{2}\sum_{k=1}^{N} w_k \left|\mathbf{v}_{kb} - \mathbf{R}^{bi}\mathbf{v}_{ki}\right|^2 \\ J &= \sum w_k (1 - \mathbf{v}_{kb}{}^{\mathsf{T}}\mathbf{R}^{bi}\mathbf{v}_{ki}) \end{aligned}$$

and maximizes the gain function

$$g = \sum w_k \mathbf{v}_{kb}{}^{\mathsf{T}}\mathbf{R}^{bi}\mathbf{v}_{ki} = \lambda_{max}$$

Review the development of the $\mathbf{q}$-Method to refresh your memory.

Rearranging the two expressions for g and J provides a useful result:

$$\lambda_{max} = \sum w_k - J \qquad \text{Exercise}$$

For the optimal eigenvalue, the loss function should be "small."

QUEST (continued)

We have the following equality involving the optimal eigenvalue, the weights, and the optimal value of the loss function:

$$\lambda_{max} = \sum w_k - J$$

Since the optimal loss function should be small, a good approximation for the optimal eigenvalue is

$$\lambda_{max} \approx \sum w_k$$

This approximation makes a good first guess for a Newton-Raphson solution of the characteristic polynomial for $\mathbf{K}$:

$$p(\lambda) \quad = \quad \lambda^4 - (a+b)\lambda^2 - c\lambda + (ab + c\sigma - d)$$

where

$$\begin{aligned} a &= \sigma^2 - \mathrm{tr}(\mathrm{adj}\,\mathbf{S}), & b &= \sigma^2 + \mathbf{Z}^\mathsf{T}\mathbf{Z} \\ c &= \det \mathbf{S} + \mathbf{Z}^\mathsf{T}\mathbf{S}\mathbf{Z}, & d &= \mathbf{Z}^\mathsf{T}\mathbf{S}^2\mathbf{Z} \end{aligned}$$

Review $\mathbf{q}$-Method development for definitions of σ, $\mathbf{S}$, and $\mathbf{Z}$.

QUEST (continued)

QUEST is an approach to solving for the maximum eigenvalue of $\mathbf{K}$, which from the $\mathbf{q}$-Method we know corresponds to the eigenvector that is the quaternion that solves Wahba's Problem.

Given the characteristic polynomial of $\mathbf{K}$,

$$p(\lambda) \quad = \quad \lambda^4 - (a+b)\lambda^2 - c\lambda + (ab + c\sigma - d)$$

and the initial guess, $\lambda_n = \Sigma w_k$, we can improve the estimate using Newton-Raphson

$$\begin{aligned} \lambda_{n+1} &= \lambda_n - p(\lambda_n)/p'(\lambda_n) \\ p'(\lambda) &= 4\lambda^3 - 2(a+b)\lambda - c \end{aligned}$$

Typically 2-3 iterations are required to reach machine precision.

This part of QUEST gives the maximum **eigenvalue**, and before doing an example, we need to develop the rest of the algorithm for computing the associated **eigenvector**, which is the **quaternion** that solves Wahba's problem.

QUEST (continued)

Given λ_{max}, how do we get the corresponding eigenvector?

One approach is to introduce the Rodrigues parameters, yet another set of attitude variables:

$$\mathbf{p} = \frac{\mathbf{q}}{q_4} = \mathbf{a} \tan \frac{\Phi}{2}$$

The eigenproblem is rearranged as

$$[(\lambda_{max} + \sigma)\mathbf{1} - \mathbf{S}]\,\mathbf{p} = \mathbf{Z}$$

Solve the linear system for $\mathbf{p}$, and then the quaternion is calculated by

$$\bar{\mathbf{q}} = \frac{1}{\sqrt{1 + \mathbf{p}^{\mathsf{T}}\mathbf{p}}} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

Thus concludes the derivation of QUEST.

Summary: Solve characteristic polynomial for λ_{max} with N-R using $\lambda_n = \sum w_k$ as initial guess. Solve the eigenproblem linear system for $\mathbf{p}$. Calculate $\bar{\mathbf{q}}$.

QUEST Illustrative Example

Repeat the $\mathbf{q}$-Method Example using QUEST, with $w_k = 1, k = 1, 2$.

First, use $\lambda = \sum w_k = 2$ and see how well this crude estimate (let's call it the *Q0* estimate) does.

The Rodrigues parameter calculation leads to

$$\mathbf{p} = [0.3139 \quad -0.0062 \quad 0.5590]^{\mathsf{T}}$$

The quaternion calculation leads to

$$\bar{\mathbf{q}} = [0.2643 \quad -0.0052 \quad 0.4706 \quad 0.8418]^{\mathsf{T}}$$

After calculating the rotation matrix, denoted $\mathbf{R}_{Q0}^{bi}$, we calculate the angle Φ for $\mathbf{R}_{error}^{bi} = \mathbf{R}_{Q0}^{bi^{\mathsf{T}}} \mathbf{R}_{exact}^{bi}$, and the loss function $J(\mathbf{R}_{Q0}^{bi})$.

$$\begin{aligned} \Phi &= 1.77^{\circ} \\ J &= 3.6957 \times 10^{-4} \end{aligned}$$

Note that both these error measures are slightly larger than those for the exact solution using the $\mathbf{q}$-Method.

The computational cost of this *Q0* estimate is significantly smaller than that for the $\mathbf{q}$-Method.

QUEST Illustrative Example (continued)

Now we carry out the refinement steps for QUEST, using $\lambda = \sum w_k = 2$ as an initial guess for finding the largest eigenvalue of the characteristic polynomial

$$p(\lambda) = \lambda^4 - (a+b)\lambda^2 - c\lambda + (ab + c\sigma - d)$$

using Newton-Raphson iteration

$$\begin{aligned} \lambda_{n+1} &= \lambda_n - p(\lambda_n)/p'(\lambda_n) \\ p'(\lambda) &= 4\lambda^3 - 2(a+b)\lambda - c \end{aligned}$$

Newton-Raphson algorithm for QUEST

1. Set $\lambda_n = \sum w_k$, `ITER` = 0, `MAXITER`=10, `TOL`=10^{-14}
2. Calculate $p_n = p(\lambda_n)$, $p'_n = p'(\lambda_n)$
3. While (`ITER<MAXITER` AND $p_n >$ `TOL`)
   - Set $\lambda_n = \lambda_n - p_n/p'_n$
   - Calculate $p_n = p(\lambda_n)$, $p'_n = p'(\lambda_n)$, `ITER = ITER+1`
4. If (`ITER<MAXITER`) Then $\lambda_{max} = \lambda_n$ Else `Failed to converge`

For the example, Newton-Raphson converges in 3 iterations. As we should expect, the result is numerically the same as the results for the $\mathbf{q}$-Method.

Attitude Determination Summary

A spacecraft must "know" where it is pointing, and that knowledge is provided by the Attitude Determination Subsystem.

- The ADS uses **sensors** to measure vectors in $\mathcal{F}_b$, and mathematical **models** to calculate the vectors in $\mathcal{F}_i$
- Need at least two vectors – Sun vector and Magnetic field vector used here
- The TRIAD algorithm is a constructive method in which the two vectors are used to assemble $\mathbf{R}^{bi}$ directly. (There are extensions to TRIAD, not covered here.)
- Wahba's Loss Function J is a well-established least-squares measure of the quality of an attitude estimate
- Wahba's Problem is "*Find the attitude that minimizes* J"
- The $\mathbf{q}$-Method is an elegant closed-form solution to Wahba's Problem, but is computationally expensive due to need to compute the eigenvalue decompisition of a 4×4 matrix
- QUEST is a powerful and computationally superior approach to solving Wahba's Problem, with a single linear system solve and a few iterations of Newton-Raphson to obtain the same results as the $\mathbf{q}$-Method

This module concludes our treatment of Attitude Kinematics topics. Next up: Rigid Body Dynamics.