Unmanned Aerial Systems
Bancroft's method and Quaternions

For questions: slynen@ethz.ch

Abstract

This exercise is intended to improve the understanding of the GPS navigation equation. Additionally there is an optional part on Quaternions to help you understand their use in representing attitudes of rigid bodies. The results of this exercise will be used in the next exercises and are required to implement the final visual-inertial optical-flow system.

1 Solving the navigation equation using Bancroft's method

As discussed in the lecture, Bancroft's method presents an elegant way of reformulating the navigation equation such that it can be solved as a least squares problem. The algorithm can be summarized as follows:

1) Organize your data into $\mathbf{B}$, $\vec{\mathbf{a}}$ and $\vec{\mathbf{e}}$.

2) Solve the quadratic equation for $\mathbf{\Lambda_1}$ and $\mathbf{\Lambda_1}$.

3) Solve the least-squares solutions $\vec{\mathbf{u}}_1^*$ and $\vec{\mathbf{u}}_2^*$ and pick the one that makes sense.

Your task is to implement the Bancroft algorithm in Matlab using the data provided below. Write a function talking as input the values x_k, y_k, z_k and t_k and calculating the position in ECEF (earth-centered, earth-fixed) coordinates.

1.1 Sample data

Here is some actual satellite data to from 6 satellites for you to use. Geocentric coordinates are given in meters and time is given in nanoseconds.

i (Satellite number)	x_k (meters)	y_k (meters)	z_k (meters)	T_i - t_i (nanoseconds)
1	14177553.47	-18814768.09	12243866.38	70446329.64
2	15097199.81	-4636088.67	21326706.55	75142197.81
3	23460342.33	-9433518.58	8174941.25	78968497.20
4	-8206488.95	-18217989.14	17605231.99	69887173.01
5	1399988.07	-17563734.90	19705591.18	67231182.38
6	6995655.48	-23537808.26	-9927906.48	80796265.09

1.2 Hints

- The data is provided in the file `gps_data.mat`.
- Speed of light: c = 299792458 m/s = 0.299792458 m/nsec.
- Radius of the earth: r = 6378 km = 6378000 m

- Implement the function `bancroft` and `lorenz_product` as prepared in the templates.
- The Matlab command `solve` can aid finding symbolic solutions of algebraic equations.

2 Quaternions (optional)

Convention

There are different conventions to represent quaternions. We use the same notation as in the lecture, which is the Hamilton notation:

$$q = q_1 + q_2 i + q_3 j + q_4 k \tag{1}$$

2.1 Calculations with quaternions

Note: These questions lead you through the derivation of several quaternion functions which are needed in the implementation part of this exercise. Feel free to skip them and go straight to the implementation part if you feel familiar enough with Quaternion math or want to solve them at the same time.

a) Show that $q \otimes p = M_q * p$ by calculating M_q. **Hint:** Multiply the quaternion elements and apply the rules for sign changes for the imaginary parts.

b) Convert the quaternion q into a rotation matrix $C = [f(q)]_{3x3}$ using one of the formulae from the lecture.

c) Convert the rotation matrix $C = [f(q)]_{3x3}$ into a quaternion q using the algorithm from the lecture.

2.2 Programming with quaternions

In MATLAB program the following scripts

a) A function `mulquat(q,p)` that multiplies two quaternions q and p as in $q \otimes p$

b) A function `quat2rot(q)` that converts the quaternion q into a rotation matrix C

c) A function `rot2quat(C)` that converts a rotation matrix C to the quaternion q

d) A function `skew(v)` that generates the skew symmetric matrix $\lfloor v_X \rfloor$ for the 3x1-vector v. Try to define some properties of $\lfloor v_X \rfloor$. (i.e. what is $\lfloor v_X \rfloor^T$, $\lfloor v_X \rfloor \vec{w}$, $\lfloor v_X \rfloor + \lfloor w_X \rfloor$, $\lfloor v_X \rfloor^n$)