解析結果

2007 年 2 月 13 日

- 2006/06/09 に実施した JAXA MuPAL - α を用いた実験
- 温度補正、取付け補正あり

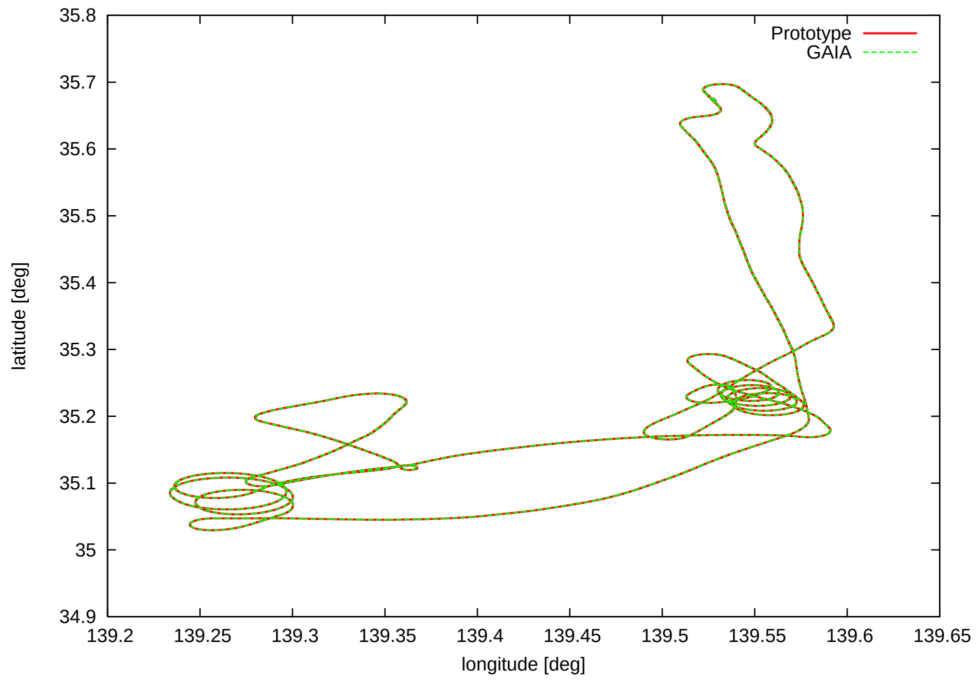

図 1　2D 履歴

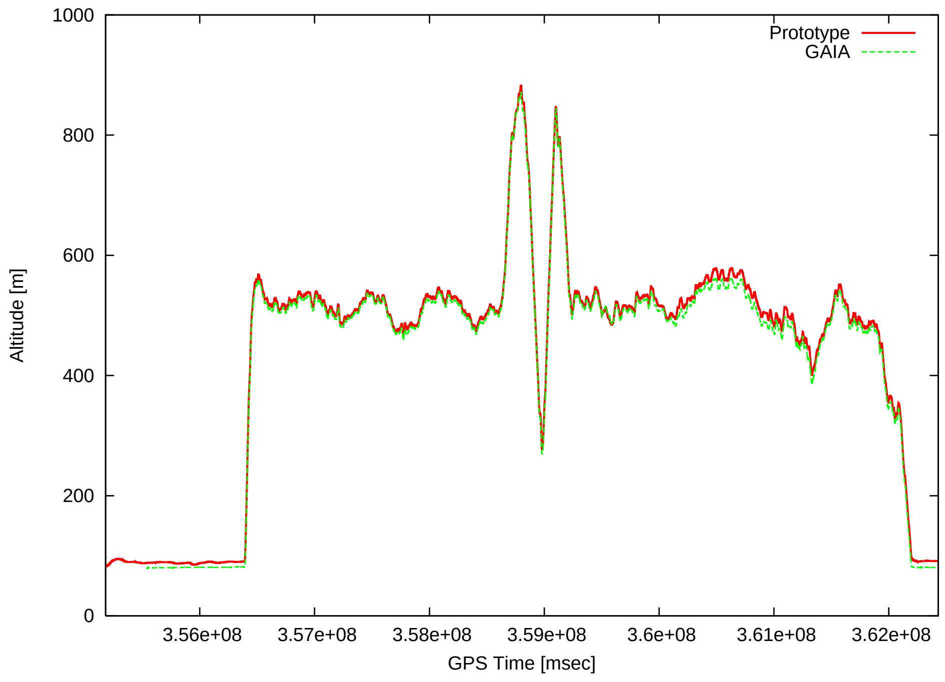

図 2　高度

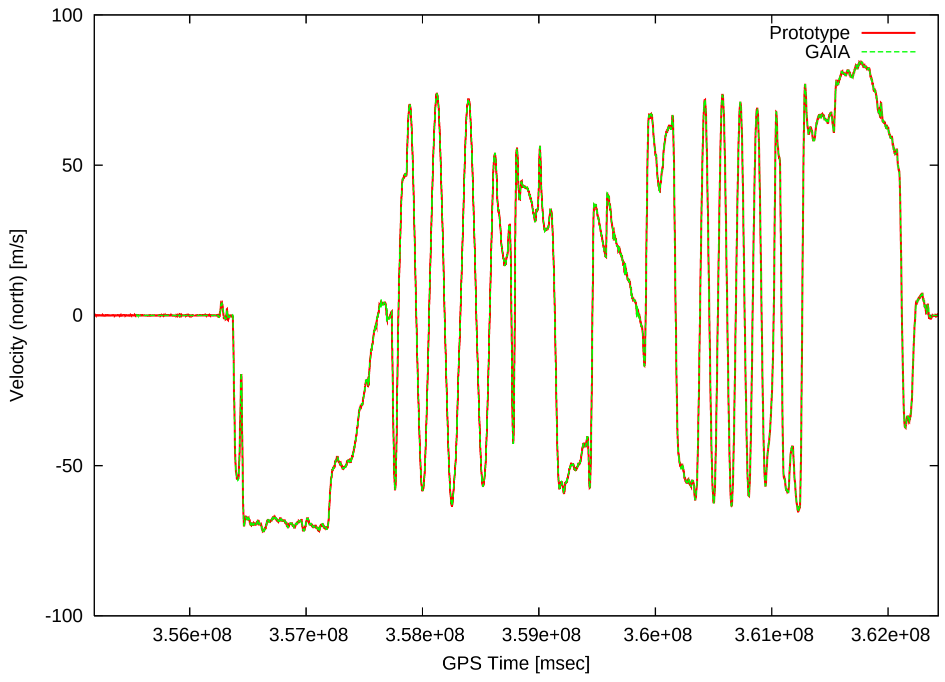

図 3　北方向速度

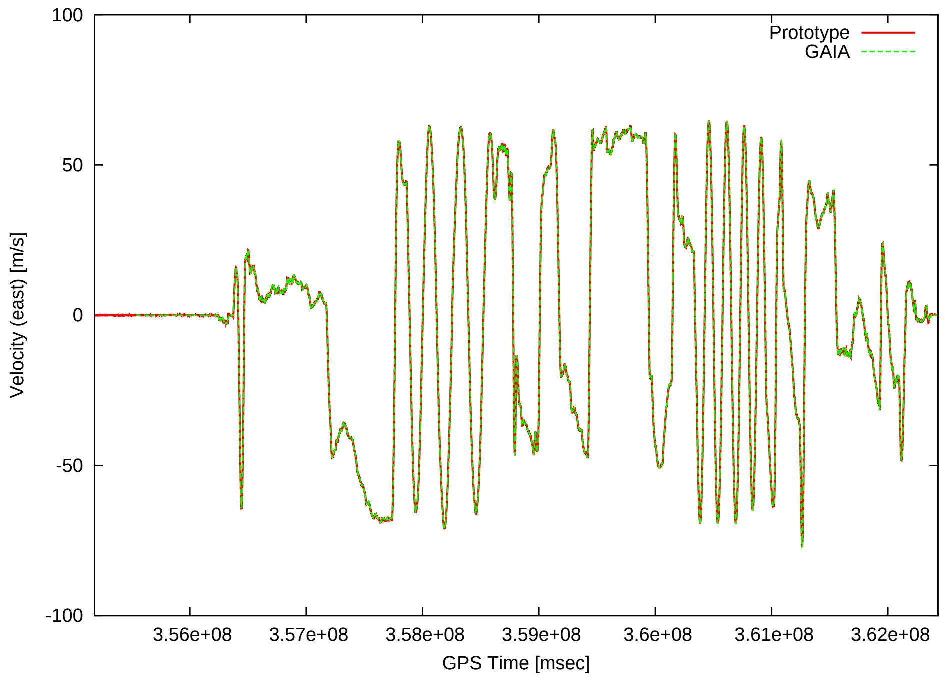

図 4　東方向速度

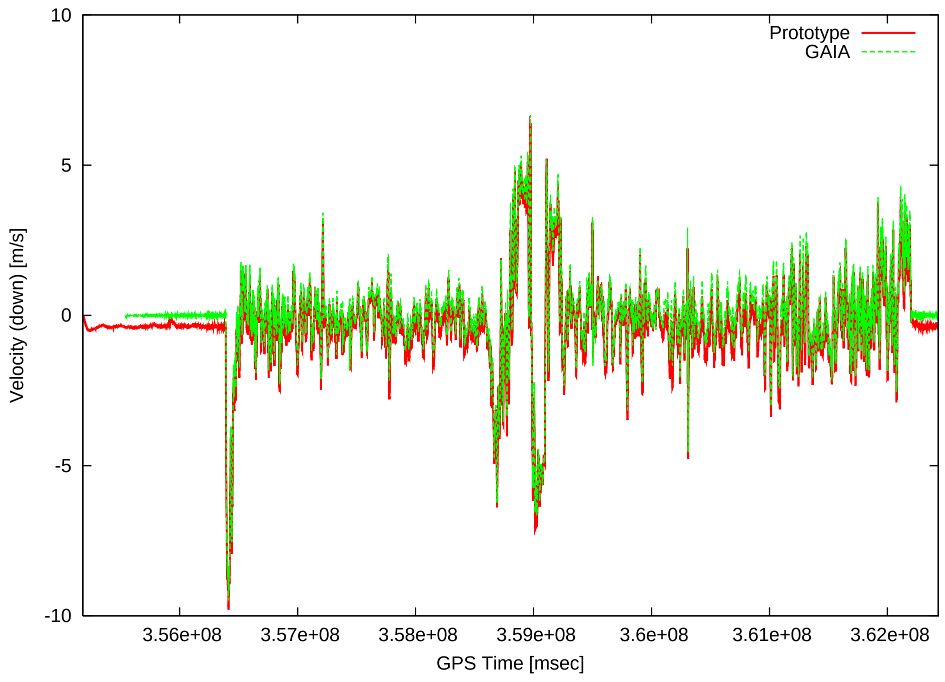

図 5　重力方向速度

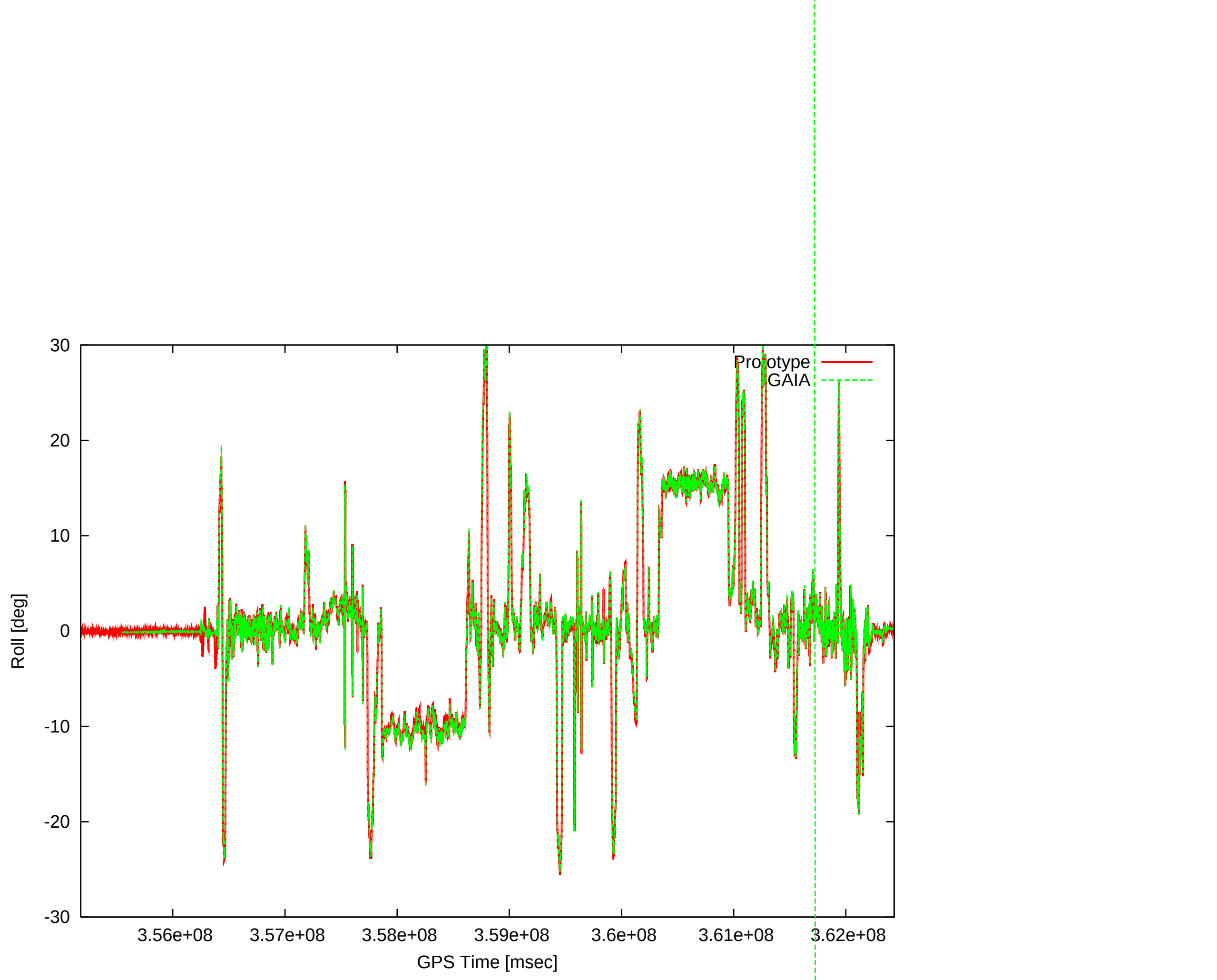
30
20
10
0
-10
-20
-30
Roll [deg]
3.56e+08
3.57e+08
3.58e+08
3.59e+08
3.6e+08
3.61e+08
3.62e+08
GPS Time [msec]
Prototype
GAIA

図 6　ロール

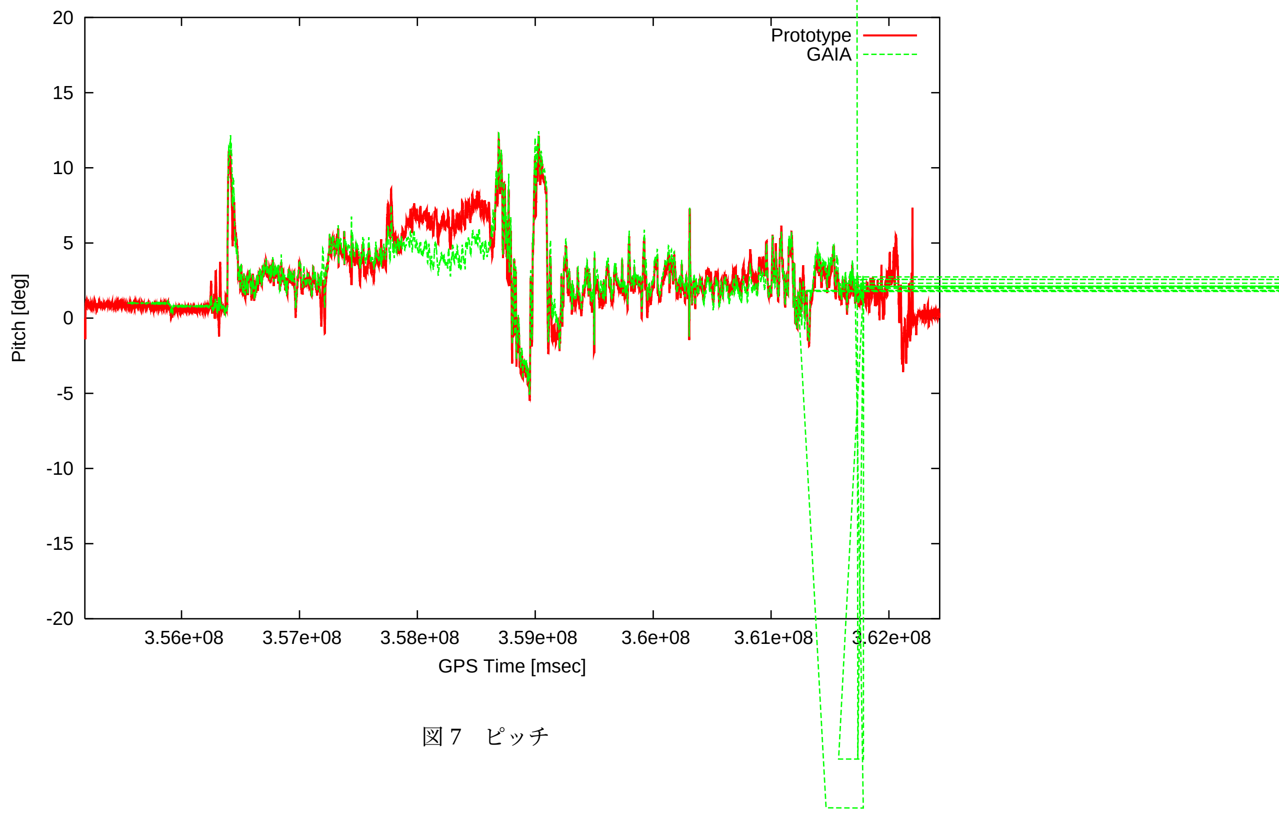
20
15
10
5
0
-5
-10
-15
-20
Pitch [deg]
3.56e+08
3.57e+08
3.58e+08
3.59e+08
3.6e+08
3.61e+08
3.62e+08
GPS Time [msec]
Prototype
GAIA

図 7　ピッチ

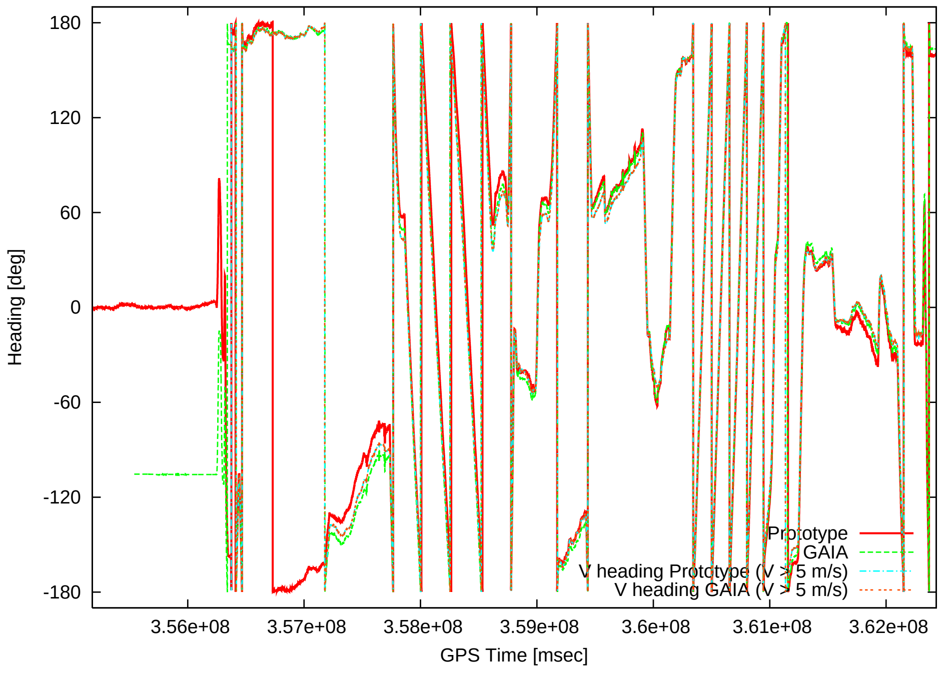

図 8　ヨー

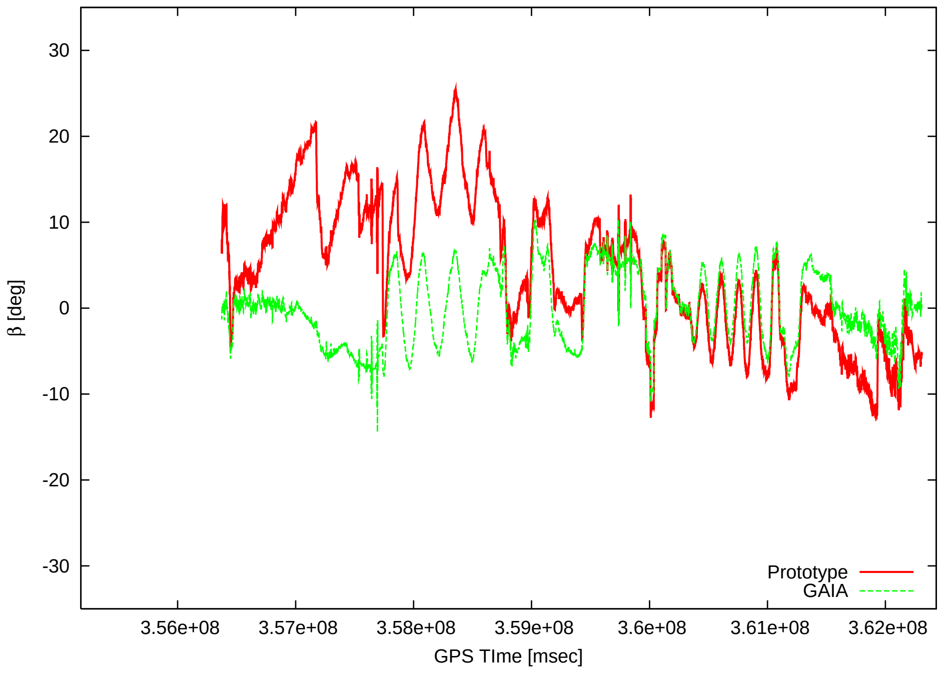

図 9　対地速度 横滑り角

表 1　プロトタイプと GAIA の差異 (GPS Time 3.58E+08[msec] 以降)

	平均値	標準偏差	最悪値
水平距離 [m]	5.37884222304374	2.17047608681524	10.6350963598937
高度 [m]	8.80062791711247	4.49189531057675	18.1365544813652
北方向速度 [m/s]	0.00251701338254768	0.131769439369425	1.4427641514
東方向速度 [m/s]	-0.00465839112641328	0.130486115746505	-1.2872197828
重力方向速度 [m/s]	-0.326693661917974	0.170983111347295	-0.93187559
ロール [deg]	-0.00907130978994206	0.226962158890542	-0.912784827
ピッチ [deg]	0.105385288014087	1.01461403062255	3.223850122
ヘディング [deg]	1.61262277226337	6.95933536841826	18.95288208

ソース

ソースコード 1　source.cpp

```cpp
/*
 *
 */
#include <iostream>
#include <iomanip>
#include <strstream>
#include <exception>
#include <math.h>
#include <time.h>

#if _MSC_VER >= 1400
#define sprintf sprintf_s
#endif

using namespace std;

typedef double accuracy;

#include "common.h"

#include "INS_GPS2.h"
#include "INS_GPS_BE.h"
#include "sensor/MEMS.h"

#include "util/logger.h"

typedef MEMS_Accelerometer mems_a;
typedef MEMS_Gyro mems_g;

//#define DUMP_UPDATE
#define DUMP_CORRECT

//#define BIAS_ON
//#define COV_OUT
```

```cpp

#define DUMP_TARGET 9
const char *label[DUMP_TARGET] = {"東経[秒]", "北緯[秒]", "高度[m]", "北方向速度[m/s]", "東方向速度[m/s]", "重力方向速度[m/s]", "Ψ[deg]", "Θ[deg]", "Φ[deg]"};

#ifdef BIAS_ON
typedef INS_GPS2_BiasEstimated<accuracy> NAV;
#else
typedef INS_GPS2<accuracy> NAV;
#endif

class CovObserver : public Loggable{
  protected:
    const NAV &m_nav;
  public:
    CovObserver(const NAV &nav) : m_nav(nav){}
    ~CovObserver(){}

    void inspect(ostream &out) const {
      NAV &nav(*(const_cast<NAV *>(&m_nav)));

      Matrix<accuracy> P(nav.getFilter().getP());
      out << scientific << setprecision(8);
      for(int i = 0; i < P.rows(); i++){
        for(int j = i; j < P.columns(); j++){
          out << P(i, j) << "\t";
        }
      }
    }
};

class Status{
  private:
    bool bool_init;
    NAV nav;
    CovObserver cov;
    Logger logger;

  public:
    Status() : bool_init(false), nav(), cov(nav), logger("additional.txt"){
#ifdef BIAS_ON
      nav.beta_accel() *= 0.1;
      nav.beta_gyro() *= 0.1;//mems_g.BETA;
#endif

      Matrix<accuracy> P(nav.getFilter().getP());
      {
        P(0, 0) = P(1, 1) = P(2, 2) = 1E+1;
        P(3, 3) = P(4, 4) = P(5, 5) = 1E-8;
        P(6, 6) = 1E+2;
        P(7, 7) = P(8, 8) = P(9, 9) = 1E-4; //CRS03 1E-1; //-4
#ifdef BIAS_ON
        P(10, 10) = P(11, 11) = P(12, 12) = 1E-2;
        P(13, 13) = P(14, 14) = P(15, 15) = 1E-8;
#endif
      }
      Matrix<accuracy> Q(nav.getFilter().getQ());
      {
        Q(0, 0) = pow(ACC_STD / ACC_1G * SCALE_1G, 2); //pow(pow(0.01, mems_a.WN_SLOPE) * mems_a.WN_INTERCEPT / mems_a.SF, 2);
        Q(1, 1) = pow(ACC_STD / ACC_1G * SCALE_1G, 2); //pow(pow(0.01, mems_a.WN_SLOPE) * mems_a.WN_INTERCEPT / mems_a.SF, 2);
        Q(2, 2) = pow(ACC_STD / ACC_1G * SCALE_1G, 2); //pow(pow(0.01, mems_a.WN_SLOPE) * mems_a.WN_INTERCEPT / mems_a.SF, 2);
        Q(3, 3) = pow(deg2rad(GYRO_STD / GYRO_1DEG), 2); //pow(pow(0.01, mems_g.WN_SLOPE) * mems_g.WN_INTERCEPT / mems_g.SF, 2);
        Q(4, 4) = pow(deg2rad(GYRO_STD / GYRO_1DEG), 2); //pow(pow(0.01, mems_g.WN_SLOPE) * mems_g.WN_INTERCEPT / mems_g.SF, 2);
        Q(5, 5) = pow(deg2rad(GYRO_STD / GYRO_1DEG), 2); //pow(pow(0.01, mems_g.WN_SLOPE) * mems_g.WN_INTERCEPT / mems_g.SF, 2);
        Q(6, 6) = 1E-6; //1E-14
#ifdef BIAS_ON
        Q(7, 7) = 1E-8;//1E-10;
        Q(8, 8) = 1E-8;//1E-10;
        Q(9, 9) = 1E-8;//1E-10;
        Q(10, 10) = 1E-10;//pow(mems_g.ROOT_N, 2) * 1. / 100 * 16; //sim6と同じ設定
        Q(11, 11) = 1E-10;//pow(mems_g.ROOT_N, 2) * 1. / 100 * 16;
        Q(12, 12) = 1E-10;//pow(mems_g.ROOT_N, 2) * 1. / 100 * 16;
#endif
      }
      //Q *= 2;
      nav.getFilter().init();
#ifdef COV_OUT
      logger.add(cov);
#endif
    }

    NAV &get_nav() {return nav;}

    static void dump_label(){
      cout << "longitude" << "\t"
           << "latitude" << "\t"
           << "height" << "\t"
           << "v_north" << "\t"
           << "v_east" << "\t"
           << "v_down" << "\t"
           << "Yaw(Ψ)" << "\t" //Ψ (yaw)
           << "Pitch(Θ)" << "\t" //Θ (pitch)
           << "Roll(Φ)" << "\t" //Φ (roll)
           << "Azimuth(α)" << "\t"; //α (azimuth)
#ifdef BIAS_ON
      cout << "bias_accel(X)" << "\t" //Bias
           << "bias_accel(Y)" << "\t"
           << "bias_accel(Z)" << "\t"
           << "bias_gyro(X)" << "\t"
           << "bias_gyro(Y)" << "\t"
           << "bias_gyri(Z)" << "\t";
#endif
    }

    void dump(const int &itow){
      cout << itow << "\t"
           << setprecision(10)
           << rad2deg(nav.longitude()) << "\t"
           << rad2deg(nav.latitude()) << "\t"
```

```
            << nav.height() << "\t"
            << nav.v_north() << "\t"
            << nav.v_east() << "\t"
            << nav.v_down() << "\t"
            << rad2deg(nav.heading()) << "\t" //Ψ (yaw) <– q_{g}^{b}
            << rad2deg(nav.euler_theta()) << "\t" //Θ (pitch) <– q_{n}^{b}
            << rad2deg(nav.euler_phi()) << "\t" //Φ (roll) <– q_{n}^{b}
            << rad2deg(nav.azimuth()) << "\t"; //α (azimuth)
#ifdef BIAS_ON
      cout << nav.bias_accel().getX() << "\t" // Bias
            << nav.bias_accel().getY() << "\t"
            << nav.bias_accel().getZ() << "\t"
            << nav.bias_gyro().getX() << "\t"
            << nav.bias_gyro().getY() << "\t"
            << nav.bias_gyro().getZ() << "\t";
#endif
      logger.out() << itow << "\t";
      logger.flush();
      logger.out() << endl;
    }

    void update(const A_Packet &current, const A_Packet &next){

#define pow2(x) ((x) * (x))
      //cout << current.acc_x() << ' ' << current.acc_y() << ' ' << current.acc_z() << endl;
      //cout << current.gyro_x() << ' ' << current.gyro_y() << ' ' << current.gyro_z() << endl;
      //cout << sqrt(pow2(current.acc_x()) + pow2(current.acc_y()) + pow2(current.acc_z())) << endl;

      if(!bool_init){return;}

      //cout << "update: " << current.itow << ' ' << current.interval(next) << endl;
      nav.update((current.accel() + next.accel()) / 2, (current.gyro() + next.gyro()) / 2, INTERVAL_UNIT * current.interval(next));

#ifdef DUMP_UPDATE
      cout << "U\t";
      dump(current.itow);
      cout << endl;
#endif
    }

    void correct(const G_Packet &g_packet){
      //cout << g_packet.acc_2d << "," << g_packet.acc_v << endl;

      static int count = 0;

      if(bool_init){
        //cout << "correct: " << g_packet.itow << endl;

        nav.correct(g_packet.convert());
        //nav.correct(g_packet.convert2());

        /*if(count == 50){
          nav.initAttitude(deg2rad(-105), nav.euler_theta(), nav.euler_phi());
        }*/
        count++;

```

```

#ifdef DUMP_CORRECT
        cout << "C\t";
        dump(g_packet.itow);
        cout << endl;
#endif
      }else if(!bool_init && g_packet.acc_2d < 100.){
        bool_init = true;
        /*cout << g_packet.llh[0] << ','
             << g_packet.llh[1] << ','
             << g_packet.llh[2] << endl;*/
        nav.initPosition(deg2rad(g_packet.llh[1]), deg2rad(g_packet.llh[0]), g_packet.llh[2]);
        nav.initVelocity(0., 0., 0.);
        nav.initAttitude(deg2rad(0), deg2rad(0), 0.);
      }
    }
};



int main(){

  Status status;

  cout << "Type\t" << "ITOW(ms)\t";
  Status::dump_label();
  cout << endl;

  char buf[1024];
  strstream line(buf, sizeof(buf), strstream::in);

  A_Packet *a_packet_current(NULL), *a_packet_next(NULL);
  G_Packet *g_packet(NULL);

  while(!cin.eof()){
    cin.getline(buf, sizeof(buf));
    buf[cin.gcount()] = '\0';
    //cout << buf << endl;

    char header;
    line.seekg(0);

    line >> header;
    if('A' == header){
      delete(a_packet_current);
      a_packet_current = a_packet_next;
      a_packet_next = new A_Packet(line);
      if(a_packet_current){
        status.update(*a_packet_current, *a_packet_next);
      }
    }else if('G' == header){
      try{
        g_packet = new G_Packet(line);
        if(a_packet_next && (a_packet_next->interval(*g_packet) > 5)){continue;}
      }catch(exception &e){
        line.clear();
      }
    }

    if(g_packet){
      status.correct(*g_packet);
      delete(g_packet);
      g_packet = NULL;
    }
  }

  cerr << "P:" << status.get_nav().getFilter().getP() << endl;
  delete(a_packet_next);
}
```