



GEOFORSCHUNGSZENTRUM POTSDAM
STIFTUNG DES ÖFFENTLICHEN RECHTS

Scientific Technical Report

ISSN 1610-0956

Handb. 2.5
Handb. 2.5
Software User Manual
GPS
GeoForschungsZentrum Potsdam

Guochang Xu
Peter Schwintzer
Christoph Reigber

KSGSoft **(Kinematic/Static GPS Software)**

Software User Manual

Version of 1998



Scientific Technical Report STR98/19

KSGSoft (Kinematic/Static GPS Software)

Software User Manual

Table of Contents

● 1. Introduction	1
● 2. Characteristics of Program	1
● 3. Execution of Program	3
● 4. Control File	4
● 4.1 Control File Definitions	4
● 4.2 Standard Control Files	9
● 4.3 Debug Switches in Control File	13
● 4.4 Standard Control File Automatization	15
● 5. Kinematic GPS Data Processing	15
● 6. Static GPS Data Processing	16
● 7. Diagram of Program	16
● 8. Basic Principles Used	18
● 8.1 Data Platform	18
● 8.2 Adjustment Method	18
● 8.3 Kalman Filter	19
● 8.4 Modules Integrator	20
● 9. Acknowledgements	20
● 10. References	21
● 11. Appendixes	22
● 11.1 Appendix 1: List of functions of the software	22
● 11.2 Appendix 2: List of utilities programs	27

1. Introduction

The prototype version of Kinematic/Static GPS Software KSGSoft was developed in 1994 to fulfill the needs of precise GPS navigation in airborne gravimetry in the GeoForschungsZentrum (GFZ) Potsdam. It was first employed in processing data from a carborne GPS campaign taking place in the end of 1993 [1].

During 1995 and 1996, the software has been modified several times based on experiences from real GPS data processing and requirements from several scientific campaigns as well as on many discussions held with several scientists in GPS area. Comparisons have been made with several commercial software packages. KSGSoft performance proved to give accurate and reliable results while being more flexible and having more complete features. The software has been successfully used in aerogravimetry, aerophotogrammetry, airborne altimetry, shipborne oceanography, as well as carborne campaigns.

The software has been implemented and used in several international research partner institutes.

This manual outlines the characteristics of the software and describes how to use the software.

2. Characteristics of the Software

- KSGSoft is developed in C under the environment of a SUN workstation and can be directly implemented on PCs without any changes (computer independent).
- KSGSoft performs a combined adjustment and filtering of multiple kinematic/static stations, i.e.: data of several moving and fixed reference ground stations can be evaluated simultaneously.
- Least squares adjustment, block-wise least squares algorithm and Kalman filtering are used.

- Constraints of known distances between antennas of the moving platform can be introduced for the platform moving state monitoring and for modifying ambiguity resolution.
- Static results can be used as conditions for kinematic solution using multiple static references for reducing unknown number and modifying ambiguity resolution.
- Provides carrier phase independent velocity solution derived from instantaneous Doppler observations.
- Velocity information is included in Kalman filtering to strengthen physical basis of the Kalman algorithm.
- Very effective ambiguity search algorithm through introducing conditions after adjustment.
- Possible to determine iono-/tropospheric effects.
- The software can run in an automatic mode and is user-friendly designed for practical and scientific purposes.
- Debug functions have been integrated in the software and can be switched on and off from function to function, so that any processing step can be checked.
- GPS data preprocessing, such as data editing and cycleslip detection as well as gross error elimination, is included in the main procedure of the software.
- The software can process the block-wise correlated GPS data successively, and allows an increasing number of unknowns with time.
- The software is equipped with selection switches to apply different data processing methods and their combinations.
- The software is capable of applying both broadcast and precise IGS ephemerides.
- The software is equipped with all possible physical models such as the Earth tide model, relativity model, tropospheric models, and their related switches.

- There is no limitation for the number of sites, satellites, and data size.
 - The software is efficient in run time, sparing of memory of storage and flexibility of parameter estimation.
 - The software works under batch and manual modes with several levels of interactivity.
 - Adaptive Kalman filtering techniques are selectable by using switches. Robust estimation technique is also available.
 - Postprocessing and real time kinematic positioning and flight state monitoring are selectable.
 - Well documented in source code and user manual.
 - Implemented in several international research institutes, used for aerogravimetry, aerophotogrammetry, airborne altimetry, shipborne oceanography, carborne surveying, volcano monitoring as well as static applications.
 - Most C functions can be directly used in Fortran programming.
 - Many utilities programs.
-

3. Execution of Program

To start the program, one just needs to enter the following command line:

```
KSGSoft In_control_file
```

where In_control_file is an input control filename. If the In_control_file is default, the program will ask for input control filename or to run program in interactive mode. Edit the In_control_file before running the program is recommended. In interactive mode, necessary information as given in In_control_file will be requested for.

The definition and description of the input control file will be given in the next

section. The standard input control file may be created automatically, see subsection 4.4.

4. Control File

The essential work to run the program for GPS data processing is to write an input control file which is defined in a flexible form. Definitions of the input control file, and standard control file as well as standard definitions of debug switches are given in following subsections.

4.1 Control File Definitions

The detailed definitions of the input control file for the GPS data processing are given below in three parts: input control file definitions, input parameter definitions, and debug switch definitions.

Input control file definitions:

- Any line with a character * (star) will be considered as comment line.
- Every not-comment line should contain the character : (colon).
- Before : are contents of input, after : is identifier text.
- Only the identifier text defined here will be accepted.
- Any line with undefined identifier text will be overread.
- Blank line will be overread.
- The order of definition lines is optional if not specified.
- EOF marks the file end.

Detailed definitions of the input parameters:

- RINEX data file name
input rinex GPS data file name, the total station number will be counted automatically, file name is not allowed to be longer than 50 characters. If reference station has to be defined, then in static case the first station (related to given data file) will be assumed as ref_station, in kinematic case the static stations should be placed after the kinematic ones.

- Coordinates x,y,z, TropCoef
input coordinates and troposphere coefficient of above mentioned station, this line should be put after the related line of Rinex file name, format: four float numbers.
Above Rinex filename and Coordinates lines could be repeated if there are more stations data used in processing. Total stations number is counted automatically.
- IGS data file name, only one file is allowed.
If this line default, the Broadcast data will be used.
- Broadcast ephemerides data file name, more data files are allowed.
Broadcasting orbit data files number is counted automatically.
- Document file name for output (record the input control file).
- Result file name for output (results in ellipsoid coordinate system).
- RESULT file name for output (results in cartesian coordinate system).
- Year, month, day of the measurement start epoch.
- Begin time hour, minute, second, format: three integers.
Due to reasons, the real begin epoch is the next one of above given.
- End time hour, minute, second, format: three integers.
- Icoordinates, <1,2,3>,
1 for using above inputed x,y,z,
2 for using that of given in RINEX file,
3 for using static coordinates given above and iterating the initial coordinates of kinematic stations.
- Time interval t1/t2, input integer t1 and t2 in second, format: two integers.
- Static/Kinematic, <1,2,3> for Static, Kinematic, both.
- Ref_sv, reference satellite, if Ref_sv<=0 or >=32 then it will be selected automatically.
- Kstation, number of kinematic stations.
- Cycleslip, criterion for cycleslip test, input integer t1 and t2,
Cycleslips_criterion = t1.t2
- M_cyctest, cycleslip test method, 0/1 for phases/Doppler.
- Igs_v, igs data version number <0,1,2>,
0 for old version with x,y,z,xdot,ydot,zdot,
1 for new version with x,y,z,dt_clock,
2 for GFZ version with x,y,z,dt_clock,xdot,ydot,zdot,zero.
- Run_n, number, <1,2,3,4>,
1 for normal run (standard run),
2 for cancel some ambiguities N in interactive module,

- 3 cancel some N from given file (Ncancel),
- 4 use N as known (ambiguity fixing, external information is allowed).
- DD, observable type used (methods),
 - 0 : only L1 phase used (no combination),
 - 1 : only L2 phase used (no combination),
 - 2 : ion-free combination $F_i(L1) - F_i(L2) * f(L2) / f(L1)$,
 - 3 : modified ion-free ambiguity-integer% combination:
 $77% * (F_i(L1) - F_i(L2) * f(L2) / f(L1))$,
 - 4 : wide lane combination : $F_i(L1) - F_i(L2)$,
 - 5 : general combination $Factor1 * F_i(L1) + Factor2 * F_i(L2)$,
 - 6 : code used, selection : P1, P2, C1, C2,
 - 7 : doppler used : D1,
 - n (n>=8) : code-phase combination method used,
 code is Code, phase is n-8 (0<=n-8<=5) related phase method.
- Dual Factor1, input integer i1 and i2, $i1/i2=Factor1$.
- Dual Factor2, input integer i3 and i4, $i3/i4=Factor2$.
- Lagrange polynomial order.
- Code, for selection of code data, <0,1,2,3,4,5> for
 auto,P1,P2,C1,C2,non.
- Frequency_number, single or double code data used, <1,2>.
- Trop_correction, <0,1,2> for no, yes, as unknown.
- Troposphere model used, <0,1,2> for non,H,S model.
- Clock_correction, <0,1,2,3> for no,Sat_clock,Receiver_c,total.
- Tide_correction, <0,1> for no, yes.
- TideCoef, tide coefficient of kinematic stations, 0/1 for airplane/car.
- Rela_correction, <0,1> for no/yes.
- Inv_n, <1,2> for Cholesky,Gaussjordan inverse method.
- Coord_ce (output forms), <1,2,3> for in cartesian,geodetic
 ellipsoid,both of ITRF coordinate system.
- N_n (criterion for cancel N), default <50>.
- Wc (code weight factor), default <1.0>.
- Wp (phase weight factor) default <100.0>.
- Wd (Doppler weight factor) default <0.0>.
- Condin (Condition number), default <0>.
- Distance of two kinematic antennas, default 4.566 meter.
- CPDV phase-doppler/code-doppler solution used <0,1>.
- Combine_code, <0,1> for no,yes code combination.
- Velocity, <0,1> for no,yes for solving velocity.

- Ambiguity fixing, <0,1--4,5> for no,yes,Teunissen.
- Ifilter, <0,1,2> for no,Kalman,adaptive Kalman filter.
- Robust, <0,1> for no,yes of robust estimation.
- Doppler_sign, <-1,1> for Berne_decoder, Landau_decoder.
- DopplerI, <0,1> for using phase, Doppler in data interpolating.
- ReceivC, <0,1,2> for receiver clock error correction, standard value <1>.
- Schue, <0,1> for further run with run-n=4 or no.
- Iearthrot, <0,1,2> for earth rotation effects, standard value <2>.
- Ixyz, <1,2,3,4> for no,yes for rotation correction, standard value <4>.
- Iluisa, <0,1,2>, coefficient of transmitting time tau, <1>.
- Itidesign, <-1,1>, theoretically sign should be 1.
- Itropsign, <-1,1>, static shows Itropsign must be 1.
- Iclocksign1, <-1,1>, theoretically sign should be 1.
- Iclocksign2, <-1,1>, theoretically sign should be 1.
- Iclocksign3, <-1,1>, theoretically sign should be 1.
- Istate, monitoring flystate 0/1 for no/yes.
- Igood, 0/1 for no/yes to read coordinates for state monitoring.
- Irfila, 0/1 for no/yes iterate filambda.
- Iotf, otf, 0/1 for no/yes.
- Iddfly, dd type used for flight state monitoring <8,9,10,11,12,13,14>.
- Imulstatic, multistatic kinematic, 0 for no, 2 for use static.
kinematic Ambiguity, static/kinematic ambiguity file names are Nfilesk/Nfilexk, they are computed under 0.
- Idis1, 0/1 for no/yes compute distance 2--3 stations.
- Idisn, true distance (known baseline) number.
- Itruedis, true distances of moving stations 1--2, 1--3, 1--4/2--3.

Debug switches definitions

- Debug O_rinexh =<0,else> : all,non output of in rinex.c defined switch
- Debug O_rinexd =<0,1,else> : all,part,non
- Debug O_getdat =<0,1,else> : all,part,non
- Debug O_getdato =<0,else> : all,non
- Debug O_odat_tc =<0,else> : all,non
- Debug O_sdatdat =<0,else> : all,non
- Debug O_cycle =<0,else> : all,non
- Debug O_gpsdat =<-1,0,1,2,3,4,5,else> : all,odat,sdat,ddat,tdat,non
for one epoch, include cycleslip test results

- Debug O_rcon =<0,else> : all,non in control.c defined switch
- Debug O_igsdat =<0,else> : all,non in igs.c defined switch
- Debug O_lagrange =<0,else> : all,non
- Debug O_lagrangef=<0,else> : all,non
- Debug O_rigs =<0,else> : all,non (used in both r_igs_d.c and r_igs_h.c)
- Debug O_broadcast=<0,1,else> : all,part,non in broad.c defined switch
- Debug O_borbc =<0,else> : all,non
- Debug O_eph_check=<0,else> : all,non
- Debug O_broadcast_orbit=<0,else> : all,non
- Debug O_r_eph =<0,else> : all,non
- Debug O_refsat =<0,1,2,3,4,5> : all,...,non in orbit.c defined switch
- Debug O_getorb =<0,else> : all,non
- Debug O_tro =<0,else> : all,non
- Debug O_rot =<0,else> : all,non
- Debug O_rela =<0,else> : all,non
- Debug O_Rotats =<0,else> : all,non in orbit.c defined switch
- Debug O_SL_eph =<0,else> : all,non
- Debug O_tide =<0,1,else> : all in tide_1day,all in interpo_rotat.c,non
- Debug O_gcslsn =<0,1,2,3> : all,part,mini,non in ad_core.c defined
- Debug O_gcsls =<0,1,2,3> : all,part,mini,non
- Debug O_cls =<0,1,2,3> : all,part,mini,non
- Debug O_rls =<0,1,2,3> : all,part,mini,non
- Debug O_ls =<0,1> : all,non
- Debug O_chol =<0,,1,2,3> : all,part,mini,non
- Debug O_gaussj =<0,1,2,3> : all,part,mini,non
- Debug O_eq =<0,1,2,3,4,5,6,7>: 0--4:all--mini,7--non,5--6 for ambiguity if(O_eq>=8)O_eq=8 in dd_phase.c
- Debug O_norm =<0,1,2> : all,part,non in all normal equation functions
- Debug O_model =<0,1,2> : all,part,non in all model functions
- Debug O_main =<0,1,2,3> : all,part,mini,non in main.c
- Debug O_norm_cp =<0,1,2> : all,part,non in norm_dd_cp.c
- Debug O_kalman =<0,1,2> : all,part,non in kalman.c
- Debug O_teun =<0,1,2> : all,part,non in teun.c
- Debug O_robust =<0,1,2> : all,part,non in robust.c
- Debug O_screen =<0,1> : yes,no screen output
- Debug O_singlep =<0,1,2,3,4,5,6> : all,part,non in singlep.c

4.2 Standard Control Files

Following is an example of a standard input control file for static GPS data processing. GPS data measured on June 25th, 1996, at GFZ, Aero, AWI (three stations) and IGS orbit data are used for double differenced L3 solution.

```
* example of a standard input control file
* rinex file names, coordinates, orbit files, output files
gfz11770.96o : RINEX data filename
3800697.486 882054.506 5028789.350 1.0 : Coordinates xyz, TropCoef
aero1770.96o : RINEX data filename
3841330.731 715105.606 5024439.024 1.0 : Coordinates xyz, TropCoef
awi11770.96o : RINEX data filename
3756031.128 566718.052 5106554.790 1.0 : Coordinates xyz, TropCoef
gfz08592.sp3 : IGS data filename
*gfz11770.96n : Broadcast ephemerides data filename
d25a : Document filename for output
d25b : Result filename for output
d25c : RESULT filename for output
* to be defined parameters:
1 : Icoordinates
1996 6 25 : Year, month, day
18 40 0 : Begin time hour, minute, second
19 40 0 : End time hour, minute, second
1 1 : Time interval i1 i2=i1/i2
1 : Static/Kinematic/Both, <1,2,3>
0 : Ref_sv, <1--31>, else automatically selected
0 : Kstation, kinematic stations number, default 0
0 : Condin (Condition number), <0,1>, default 0
4.566 : Distance of two antennas, default 4.566 meter
2 2 : Cycleslip, criterion for cycleslip test, i1 i2=i1.i2, default 1.1
0 : M_cyctest, cycleslip test method, 0/1 for phases/Doppler
2 : DD data type used methods, <0,1,2,3,4,5,6,7,(8--13)>
1 1 : Dual Factor1, i1 i2=i1/i2
-1 1 : Dual Factor2, i3 i4=i3/i4
3 : which Code measurement used, <0,1,2,3,4,5>for auto,P1,P2,C1,C2,non
```

1 : Frequency_number used <1,2>
0 : Combine_code, <0,1> for no,yes code combination
0 : Velocity, <0,1> for no,yes
1 : Ambiguity fixing, <0,1--4,5> for no,yes,teunissen
0 : CPDV phase-doppler/code-doppler solution used <0/1>
0 : Ifilter, <0,1,2> for no,Kalman,adaptive Kalman filtering
0 : Robust estimation for weight regulation, <0,1> for no/yes
1 : Igs_v (igs data version), <0,1,2>
1 : Run_n (program run number), <1,2,3,4>
3 : Coord_ce (output form), <1,2,3>
50 : N_n (criterion for cancel N), default <50>
0 1 : Wc (code weight factor), i1 i2=i1.i2, default 0.1
100 0 : Wp (phase weight factor), i1 i2=i1.i2, default 100.0
0 0 : Wd (doppler weight factor), default 0.0
7 : Lagrange polynomial order, default 7
1 : Trop_correction, <0,1,2> for no, yes, as unknown
2 : Troposphere model used, <0,1,2> for non,H,S model
1 : Clock_correction, <0,1,2,3> for no,Sat_clock,Receiver_c,total
1 : Tide_correction, <0,1> for no, yes
0.0 : TideCoef, tide coefficients of kinematic stations
1 : Rela_correction, <0,1> for no, yes
1 : Inv_n, <1,2> for Cholesky,Gaussjordan inverse method
-1 : Doppler_sign, depends on decoder, <-1,1> for Berne_, Landau_decoder
0 : DopplerI, <0,1> for using phase, Doppler in data interpolating
1 : ReceivC, <0,1,2> for receiver clock error correction, standard value <1>
0 : Schue, <0,1> for further or no further run with run_n=4
2 : Iearthrot, <0,1,2> for earth rotation effects, standard value <2>
4 : Ixyz, <1,2,3,4> for no,yes for rotation correction, standard value <4>
1 : Iluisa, <0,1,2>, coefficient of transmitting time tau
1 : Itidesign, <-1,1>, theoretically sign should be 1
1 : Itropsign, <-1,1>, static shows Itropsign must be 1
1 : Iclocksign1, <-1,1>, theoretically sign should be 1
1 : Iclocksign2, <-1,1>, theoretically sign should be 1
1 : Iclocksign3, <-1,1>, theoretically sign should be 1
0 : Istate, monitoring flystate 0/1 for no/yes
0 : Igood, 0/1 for no/yes to read coordinates for state monitoring
1 : Irfila, 0/1 for no/yes iterate filambda
0 : Iotf, otf, 0/1 for no/yes

8 : Iddfly, dd type used for flight state monitoring <8,9,10,11,12,13,14>
 0 : Imulstatic, multistatic kinematic, 0 for no, 2 for use static
 * kinematic Ambiguity, static/kinematic ambiguity file names
 * are Nfilesk/Nfilexk, they are computed under 0
 1 : Idis1, 0/1 for no/yes compute distance 2--3 stations
 0 : Idisn, true distance number
 4.3333 : Itruedis, true distances of 1--2, 1--3, 1--4/2--3
 EOF : end of input file.

Following is an example of a standard input control file for kinematic GPS data processing. GPS data measured on Sep. 21st, 1996, at Air1, Air2, Airp, Norw (four stations) and IGS orbit data are used for double differenced L3 solution. Two static reference stations are used. The distance of both kinematic antennas is 4.566 meter.

* example of a standard input control file

* rinex file names, coordinates, orbit files, output files

43392650.96o : RINEX data filename
 3460996.5010 501135.8470 5316566.4950 1.0 : Coordinates xyz, TropCoef
 61072650.96o : RINEX data filename
 3460996.5010 501135.8470 5316566.4950 1.0 : Coordinates xyz, TropCoef
 airp2650.96o : RINEX data filename
 3435384.7030 525262.6620 5330297.2570 1.0 : Coordinates xyz, TropCoef
 norw2650.96o : RINEX data filename
 3357900.4909 395466.5280 5390144.8078 1.0 : Coordinates xyz, TropCoef
 igs08716.sp3 : IGS data filename

*airp2650.96n: Broadcast ephemerides data filename

d21k2s2a.4 : Document filename for output

d21k2s2b.4 : Result filename for output

d21k2s2c.4 : RESULT filename for output

* to be defined parameters:

3 : Icoordinates

1996 9 21 : Year, month, day

8 30 0 : Begin time hour, minute, second

10 10 0 : End time hour, minute, second

1 1 : Time interval i1 i2=i1/i2

2 : Static/Kinematic/Both, <1,2,3>

31 : Ref_sv, <1--31>, else automatically selected

2 : Kstation, kinematic stations number, default 0

0 : Condin (Condition number), <0,1>, default 0
 4.566 : Distance of two antennas, default 4.566 meter
 2 2 : Cycleslip, criterion for cycleslip test, i1 i2=i1.i2, default 1.1
 0 : M_cyctest, cycleslip test method, 0/1 for phases/Doppler
 10 : DD data type used methods, <0,1,2,3,4,5,6,7,(8--13)>
 1 1 : Dual Factor1, i1 i2=i1/i2
 -1 1 : Dual Factor2, i3 i4=i3/i4
 3 : which Code measurement used, <0,1,2,3,4,5>for auto,P1,P2,C1,C2,non
 1 : Frequency_number used <1,2>
 1 : Combine_code, <0,1> for no,yes code combination
 1 : Velocity, <0,1> for no,yes
 0 : Ambiguity fixing, <0,1--4,5> for no,yes,teunissen
 0 : CPDV phase-doppler/code-doppler solution used <0/1>
 0 : Ifilter, <0,1,2> for no,Kalman,adaptive Kalman filtering
 0 : Robust estimation for weight regulation, <0,1> for no/yes
 1 : Igs_v (igs data version), <0,1,2>
 1 : Run_n (program run number), <1,2,3,4>
 3 : Coord_ce (output form), <1,2,3>
 50 : N_n (criterion for cancel N), default <50>
 0 1 : Wc (code weight factor), i1 i2=i1.i2, default 0.1
 100 0 : Wp (phase weight factor), i1 i2=i1.i2, default 100.0
 0 0 : Wd (doppler weight factor), default 0.0
 7 : Lagrange polynomial order, default 7
 1 : Trop_correction, <0,1,2> for no, yes, as unknown
 2 : Troposphere model used, <0,1,2> for non,H,S model
 1 : Clock_correction, <0,1,2,3> for no,Sat_clock,Receiver_c,total
 1 : Tide_correction, <0,1> for no, yes
 0.0 0.0 : TideCoef, tide coefficients of kinematic stations
 1 : Rela_correction, <0,1> for no, yes
 1 : Inv_n, <1,2> for Cholesky,Gaussjordan inverse method
 -1 : Doppler_sign, depends on decoder, <-1,1> for Berne_, Landau_decoder
 0 : DopplerI, <0,1> for using phase, Doppler in data interpolating
 1 : ReceivC, <0,1,2> for receiver clock error correction, standard value <1>
 0 : Schue, <0,1> for further or no further run with run_n=4
 2 : Iearthrot, <0,1,2> for earth rotation effects, standard value <2>
 4 : Ixyz, <1,2,3,4> for no,yes for rotation correction, standard value <4>
 1 : Iluisa, <0,1,2>, coefficient of transmitting time tau

1 : Itidesign, <-1,1>, theoretically sign should be 1
 1 : Itropsign, <-1,1>, static shows Itropsign must be 1
 1 : Iclocksign1, <-1,1>, theoretically sign should be 1
 1 : Iclocksign2, <-1,1>, theoretically sign should be 1
 1 : Iclocksign3, <-1,1>, theoretically sign should be 1
 0 : Istate, monitoring flystate 0/1 for no/yes
 0 : Igood, 0/1 for no/yes to read coordinates for state monitoring
 1 : Irfile, 0/1 for no/yes iterate filambda
 0 : Iotf, otf, 0/1 for no/yes
 8 : Iddfly, dd type used for flight state monitoring <8,9,10,11,12,13,14>
 0 : Imulstatic, multistatic kinematic, 0 for no, 2 for use static
 * kinematic Ambiguity, static/kinematic ambiguity file names
 * are Nfilesk/Nfilek, they are computed under 0
 0 : Idis1, 0/1 for no/yes compute distance 2--3 stations
 0 : Idisn, true distance number
 4.3333 : Itruedis, true distances of 1--2, 1--3, 1--4/2--3
 EOF : end of input file.

4.3 Standard Debug Switches in Control File

The standard debug switches are defined as following:

* Definitions of standard debug switches

1 : Debug O_rinexh, <0,else>:all,non in rinex.c defined switch
 2 : Debug O_rinexd, <0,1,else> :all,part,non
 2 : Debug O_getdat, <0,1,else> :all,part,non
 1 : Debug O_getdato, <0,else> :all,non
 1 : Debug O_odat_tc, <0,else> :all,non
 1 : Debug O_sdtat, <0,else> :all,non
 2 : Debug O_cycle, <0,1,else> :all,part,non output
 4 : Debug O_gpsdat, <-1,0,1,2,3,else> :all,o-,s-,d-,tdat,non
 1 : Debug O_rcon, <0,else> :all,non
 1 : Debug O_igsdat, <0,else> :all,non in igs.c defined switch
 1 : Debug O_lagrange, <0,else>:all,non
 1 : Debug O_lagrangef, <0,else>:all,non
 1 : Debug O_rigs, <0,else> :all,non (used in r_igs_d,r_igs_h)
 2 : Debug O_broadcast, <0,1,else> : all,part,non in broad.c

1 : Debug O_borbc, <0,else>:all,non
1 : Debug O_eph_check, <0,else>:all,non
1 : Debug O_broadcast_orbit, <0,else>:all,non
1 : Debug O_r_eph, <0,else>:all,non
5 : Debug O_refsats, <0,1,2,3,4,5>:all,....,non in orbit.c
1 : Debug O_getorb, <0,else>:all,non
1 : Debug O_tro, <0,else>:all,non
1 : Debug O_rot, <0,else>:all,non
1 : Debug O_rela, <0,else>:all,non
1 : Debug O_Rotats, <0,else>:all,non
1 : Debug O_SL_eph, <0,else>:all,non
2 : Debug O_tide, <0,1,else>:all(tide_1day),all(interpo_rotat),non
0 : Debug O_gcslns, <0,1,2,3>:all,part,mini,non in ad_core.c
0 : Debug O_gcsls, <0,1,2,3>:all,part,mini,non
0 : Debug O_cls, <0,1,2,3>:all,part,mini,non
0 : Debug O_rls, <0,1,2,3>:all,part,mini,non
1 : Debug O_ls, <0,1>:all,non
3 : Debug O_chol, <0,,1,2,3>:all,part,mini,non
3 : Debug O_gaussj, <0,1,2,3>:all,part,mini,non
7 : Debug O_eq, <0,1,2,3,4,5,6,7>:<0--4:all--mini,7--non,5--6 for ambiguity>
3 : Debug O_norm, <0,1,2>:all,part,non in all normal equations
0 : Debug O_model, <0,1,2>:all,part,non in all model functions
4 : Debug O_main, <0,1,2,3,4>:all,part,mini,rot,non in main.c
4 : Debug O_norm_cp, <0,1,2>:all,part,non in norm_dd_cp.c
2 : Debug O_kalman =<0,1,2> :all,part,non in kalman.c
2 : Debug O_teun =<0,1,2> :all,part,non in teun.c
0 : Debug O_robust =<0,1,2> :all,part,non in robust.c
1 : Debug O_screen =<0,1> :yes,no screen output
7 : Debug O_singlep =<0,1,2,3,4,5,6> :all,part,non in singlep.c
4 : Debug O_specula = <0,1,2,3,4> :all,part,non
4 : Debug O_ambifix = <0,1,2,3,4> :all,part,non
4 : Debug O_ambifib = <0,1,2,3,4> :all,part,non

4.4 Standard Control File Automatization

Even though the input control file is quite simple defined and can be edited with any text editor, however, an automatic generation of the standard input control file is needed and helpful. Two small programs are designed for this purpose and named as Ana_dat and Ana_afile. Then Ana_dat analyses the given GPS Rinex data and creates the related information files (called as a_files). Then Ana_afile works interactively or reads a given example input control file for necessary information and creates the standard input control files for use. The outputs of Ana_afile are input control files with name prefix In merged with an increasing integer number as suffix, like In0, In1, In2,

Then the whole GPS data processing procedures can be carried out by enter the following command lines successively:

```
Ana_dat  
Ana_afile  
KSGSoft In0  
KSGSoft In1  
.....
```

Such command lines can be written into a batch file and then the whole process can be done automatically. Integrating the function of Ana_dat.c and Ana_afile.c into the KSGSoft.c, the software can run full automatically under standard parameter conditions. Thus, not much training and experience is required to operate the software.

5. Kinematic GPS Data Processing

In kinematic GPS data processing, it has to be emphasized that in the related input control file the kinematic data filenames must be put before the static reference station's data filenames. In case of only one kinematic station used in GPS data processing, the computer screen shows the output context:

hour, minute, second, obs-number, single diff. obs-number, double diff. obs-number, triple diff. obs-number, xdot, ydot, zdot, ambiguity-number, dx, dy, dz

The result data file includes context:

hour, minute, second, 3 velocity components, 3 position components

In ITRF cartesian coordinate system, velocity and position components are given in vx, vy, vz, x, y, z, respectively. In ITRF geodetic ellipsoid coordinate system, these are given in v-south, v-east, v-height, latitude, longitude, height, respectively. The velocity components have always the unit meter/second, position components and height are given in meter, angels have unit degree.

6. Static GPS Data Processing

In static GPS data processing, it has to be emphasized that the related input control file the reference station's data filename must be put before the other station's data filenames. The computer screen shows:

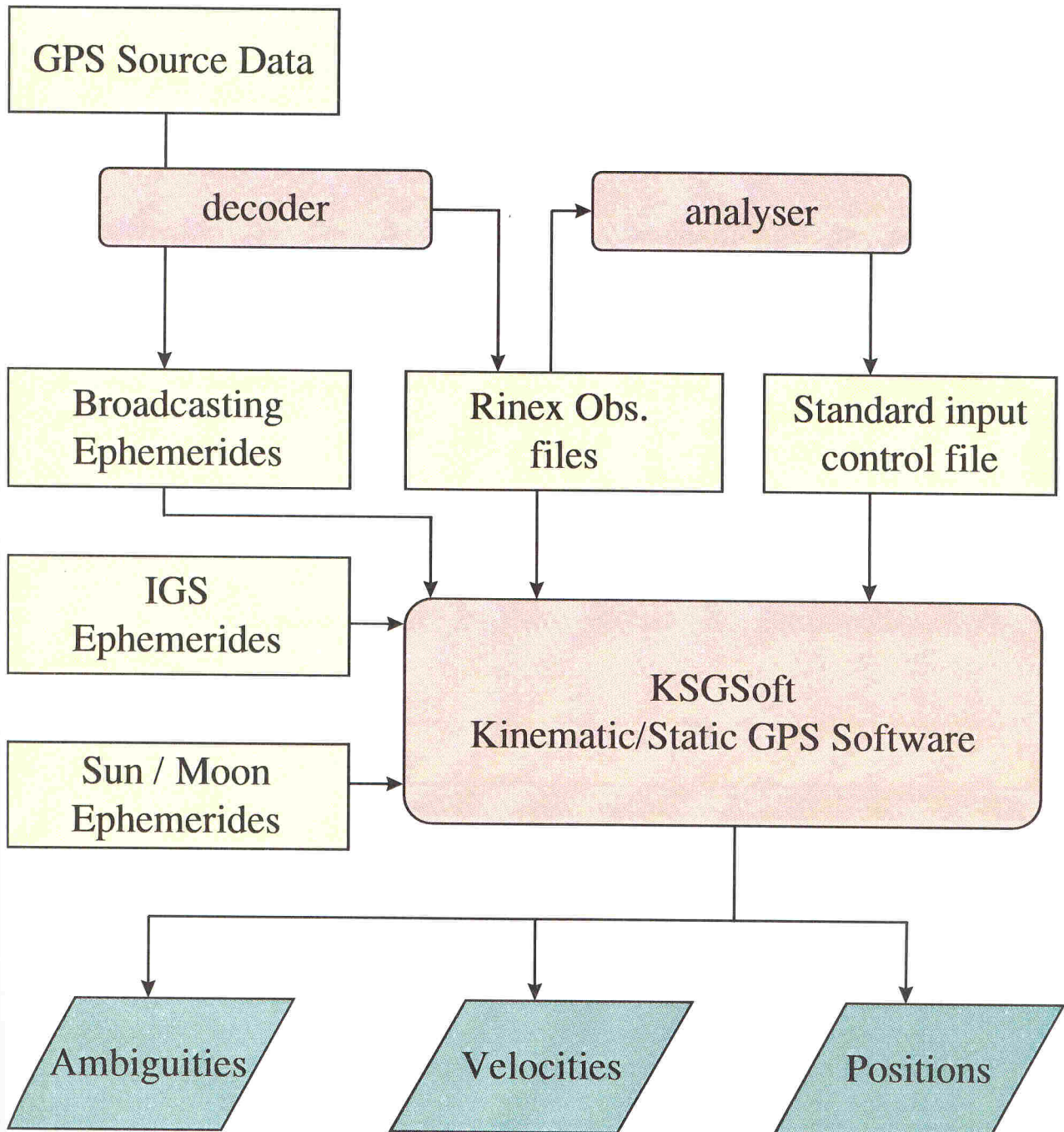
hour, minute, second

At the end of data processing, the detail adjusted results including coordinates will be given. The related output file is just a mirror file of the results output on the screen.

7. Diagram of Program

See next page.

GPS Kinematic / Static Data Standard Processing Scheme



8. Basic Principles Used

The software has four essential packages, they are brief outlined in following subsections. For more detailed reference, literature [2] is recommended.

8.1. Data Platform

The basic part of the kinematic/static software is the data-platform. It's function is to prepare GPS and orbit data, and to provide entire corrections, such as tropospheric correction, the Earth tide correction, relativity correction, clock error correction, transmitting time correction, the Earth rotation correction, time differences correction. GPS data of every stations are read only for a few epoches each time for use, so that the GPS station number and total observation time are unlimited by data processing. Cycle slips testing and differences forming are carried out. IGS orbit data or broadcasting ephemerides are also prepared for use. Several help-functions, such as coordinates transform, coordinates rotation, Helmert transformation as well as data interpolation tools are also presented.

Based on such a data-platform, any GPS data processing modules can be implemented.

8.2 Adjustment method

Least squares (LS) adjustment principle is the basic one used in the software. For static case the least squares (LS) adjustment algorithm is directly used for determining the complete unknowns. For kinematic case, there are two groups of unknowns. One changes with the time (e.g. coordinates) and another does not (e.g. ambiguities). A block-wise adjustment algorithm is specially suitable for kinematic case to separate the time dependent unknowns and time independent unknowns, so one can solve for position every epoch in one hand, and meanwhile in another hand, one can take the updated ambiguity information for further use. The coordinates solution modified as the ambiguity information accumulated.

The best ambiguity solution will be obtained at the end of the whole kinematic surveying, so a repeat computation of the kinematic coordinates by using best known ambiguity is necessary if a homogeneous coordinates solution is desired.

8.3 Kalman filter

Kalman filtering method is specially suitable for On The Fly kinematic applications. It can be brief outlined as follows.

For epoch k , from GPS observations one can form the normal equation

$$M*Z = B$$

and compute the value $|M|$, where M is normal matrix, B is computed known vector, Z is unknown vector, $|M| = L^T * P * L$, L is observation vector, L^T is transposed L , P is weight matrix.

Update the M matrix by

$$M = M + M_.$$

Then solve the problem

$$M*Z = B$$

by using least squares method and obtain the Z and Q , where $Q = \text{Inverse}(M)$.

Using system equation or velocity information, one has Z' and Qz' , where $Z' = dZ/dt$, Qz' is covariance matrix of Z' . Then one can predict $Z(k+1)$ and $M_.$ for the next epoch by using

$$Z(k+1) = Z + Z'*dt, M_ = \text{Inverse}(Q + Qz'*dt*dt).$$

Repeat for the next epoch.



8.4 Modules Integrator

Switches are given for selection of data processing models and their combinations, such as code solution, L1-phase solution, L2-phase solution, ionosphere-free solution, wide-lane, narrow-lane and user defined combination, code-phase combined solution, as well as Doppler velocity solution. Switches are also given for selections such as internal and external ambiguity fixing, weight setting, correction model selection, etc.

9. Acknowledgements

Gratefully acknowledged are the helpful discussions held with Dr. Hans-Juergen Euler of Leica AG (Switzerland), Dr. Oscar L. Colombo of NASA/GSFC (USA), Dr. Shengyun Zhu of GFZ Potsdam (Germany), and Dr. Danan Dong of JPL (USA) during the software development. EU-project AGMASCO real data was provided to fully test of the performance of this software.

10. References

- Xu, G.; Hehl, K.; Angermann, D. (1994): GPS Software Development for Use in Aerogravimetry: Strategy, Realization, and First Results, Proceedings of ION GPS-94, p1637-1642
- Hofmann-Wellenhof, B; Lichtenegger, H.; Collins, J. (1992): GPS Theory and Practice, Springer-Verlag, Wien
- Wang, G.; Chen, Z.; Chen, W.; Xu, G. (1988): The Principle of the GPS Precise Positioning System, Surveying Publishing House, Peking
- Hostetter, G. H. (1987): Handbook of Digital Signal Processing, Engineering Applications, Academic Press, INC.
- Schwarz, K.P.; Cannon, M.E.; Wong, R.V.C. (1989): A Comparison of GPS Kinematic Models for the Determination of Position and Velocity Along a Trajectory, Manuscripta Geodaetica 14, 345-353
- Xu, G.; Bastos, L.; Timmen, L. (1997): GPS Kinematic Positioning in AGMASCO Campaigns -- Strategic Goals and Numerical Results, Proceedings of ION GPS-97 Conference in Kansas City (USA), p1173-1183
- Euler, H.J.; Landau, H. (1992): Fast GPS ambiguity resolution on-the-fly for real-time applications. Proceedings of 6th Int. Geod. Symp. on satellite Positioning. Columbus, Ohio, 17-20.
- Teunissen, P.J.G. (1995): The least-squares ambiguity decorrelation adjustment: a method for fast GPS integer ambiguity estimation. Jurnal of Geodesy, 70, 65-82.
- Goad, C.; Remondi, B. (1984): Initial Relative Positioning Results Using the Global Positioning System, Bulletin Geodesique, 58, 193-210.
- Cannon, M.E.; Lachapelle, G.; Szarmes, M.; Herbert, J; Keith, J.; Jokerst, S. (1997): DGPS Kinematic Carrier Phase Signal Simulation Analysis for Precise Velocity and Position Determination, Proceedings of ION NTM 97, Santa Monica, CA.

11. Appendixes

11.1 Appendix 1: List of functions of the software

Complete functions used in this software are listed bellow:

ksg44.c --- main program of KSGSoft.

rinexy1.c --- functions package related to Rinex data files.

int getline() --- get a line from data file pointer fp into string s.
void r_rinexh() --- read the head part of Rinex data file.
void r_rinexd() --- read the data part of Rinex data file.
void getdat() --- get the gps data at time rt for all stations.
void getdat_o() --- get Rinex data for all stations with Iepoches.
void t_hms() --- change time t into hour, minute, sec.
void id_i12j12() --- change observable id to sta_i1,i2, sat_j1,j2.
void odat_tc() --- interpolate Rinex data so that all observations are made at the same time odat_t.
void sddat() --- form single, double, triple difference data sd, dd, td.
double line_interpo() --- linear interpolation for data at time t, data used: y1, y2, t1, t2.
double line2_interpo() --- interpolation for data at time t, data used: y0, y1, y2, t0, t1, t2.
double line2_integrate() --- interpolation for data at time t, data used: y0, y1, y2, t0, t1, t2.
void id_obs() --- set column values of C1, C2, P1, P2, L1, L2, D1, D2 using header data information.
void cycleslips_test() --- check the cycle slips.
void gps_dat() --- read Rinex data, single point positioning, compute clock error, correct the time differences of different stations, form single, double and triple differences, cycle slips test.

controly.c--- functions package related to input control file.

void r_con() --- read the input control file.
int indexx() --- search in string s for character c and return index.
void w_info() --- write information to file.
void DOP() --- DOPs calculation.
void wr_condat() --- write/read data idN00, N0tb, N00, m00, ...
void time_debug() --- set debug parameters from time to time.
int overread() --- overread some strings.

igsnew.c --- functions package related to IGS data.

void igs() --- read igs gps orbit data.
void igsdat() --- read igs gps orbit data for use.
double lagrange() --- lagrange interpolate function for equi-deltat case.
double lagrangef() --- lagrange interpolate function.
void r_igs_h() --- read the standard igs file header data.
void r_igs_d() --- read the standard igs data file.

broad.c --- functions package related to broadcasting ephemerides.

void broadcast() --- transfer broadcasting ephemerides into IGS
format.
void r_eph() --- read the standard broadcast orbits data files.
void eph_check() --- check the broadcast data read.
void DdE_e() --- change all D, d, E in the string s to e.
void broadcast_orbit() --- calculate the satellite position at epoch t.
double Kepler_equation() --- solve the Kepler equation.
void borbc() --- calculate the orbit using broadcasting ephemerides
for needed time.
int findisat() --- find out the common satellite number.

orbit.c --- functions package related to orbit.

int getorb() --- calculate the related satellite orbit.
int refsat() --- determine the reference satellite.

correct.c --- functions package related to correction models.

void XYZBLH() --- calculate geodetic coordinates from cartesian
coordinates.

`void BLHXYZ()` --- calculate cartesian coordinates from geodetic coordinates.
`double tro_hopfield()` --- modified Hopfield troposphere model.
`double tro_saasta()` --- Saastamoinen troposphere model.
`double zenith()` --- zenith distance of the satellite *j* related to station *i*.
`void rotation()` --- transform a vector or a matrix between geocentric coord. system and local coord. system.
`double rela_tivity()` --- calculate the relativistic effects.

detlef.c --- functions package related to transformation.

`void TRANF()` --- transformation between ϕ, λ, h and x, y, z
`double zenith1()` --- zenith distance of the satellite *j* related to station *i*.

tidex.c --- functions package related to tidal correction.

`void tide()` --- calculate the Earth tide corrections for every station.
`void tide_1day()` --- create one day tidal correction data for stations.
`void tide_t()` --- interpolate tide displacement at epoch hour.
`double JD()` --- calculate JD (returned) using UTC time.
`double JD_GPST()` --- calculate JD (returned) using GPS time.
`void GPST()` --- calculate GPS day of week.
`void YMDHNW_JD()` --- calculate year, month, day, hour, as well as GPS day of week, GPS week from JD.
`void TDT_GPST()` --- calculate TDT from GPS time.
`void UTC_GPST()` --- calculate UTC from GPST time.
`void Rotation_M()` --- calculate three rotational matrixes R_p, R_n, R_s .
`void interpo_rotat()` --- Interpolate the S_r and L_r from SL_{eph} using given GPS time, and rotate S_r and L_r to the Earth fixed coordinates system.
`void D_e()` --- change all *D* in the string *s* to *e*.
`int SL_eph()` --- get needed Sun and Moon ephemerides.

ad_core.c --- functions package related to adjustment and filtering.

`int gauss_jordan()` --- Gauss-Jordan algorithmus.
`int chol_inv()` --- Cholesky's decomposition method.

double gcs_lsn1() --- use least squares adjustment method to solve
 a linear observation equation system with a group wise correlated
 equation system and a group wise added new observations and some
 new unknowns.
double gcs_ls2() --- an inverser function of **gcs_ls1()**.
double gcs_ls1() --- use least squares adjustment method to solve
 a linear observation equation system with a group wise correlated
 equation system and a group wise added new observations.
double condi_ls() --- use least squares adjustment method to solve
 a linear observation equation system with a condition equation system.
double ls() --- least squares adjustment.
double recurs_ls() --- recursive least squares adjustment.
void pm_low_cl_v2() --- output a n*n matrix in lower case.
void rm_low_cl_v2() --- read a n*n matrix in lower case.
void pm_2d_cl_v2() --- output a m*n matrix.
void rm_2d_cl_v2() --- read a n*n matrix.
void p2v_v2() --- output two vectors.
void r2v_v2() --- read two vectors.
void pstr1v_v2() --- output a variable.
double rstr1v_v2() --- read a variable.
void pv_1d_v2() --- output a vector.
void rv_1d_v2() --- read a vector.

eqdd_s42.c --- void **eq_dd_s()** : form static obs. equations.

eqdd_k57.c --- void **eq_dd_s5()** : form kinematic obs. equations.


eqdd_d85.c --- void **eq_dd_k7()** : form Doppler obs. equations.

normdd_s.c --- void **edit_norm_dd()** : form static normal equation.

normdd_k.c --- void **norm_dd_k()** : form kinematic normal equation.

c pha33.c --- void **norm_dd_cp()** : form the four blocks normal equation
 and solve them block weis.

dis_con.c --- void **dis_con()** : approximated distance condition.



c_pha3k4.c --- void norm_dd_cpk() : form the four blocks normal equation and solve them block weis under conditions.

filenam.c --- void filenam() : strcat the integer j as character to string file_nam.

ambifix.c --- functions package related to ambiguity fixing.

void ambifix() --- use conditioned least squares method to solve the ambiguity fixing problem.

void NINi() --- compute to be searched ambiguity possibility number and values.

void min2m0() --- remember the two minimum m01, m02.

void getNk() --- give ambiguity test vector Nk.

void NisNk() --- separate ambiguity vector W as several groups.

double dfix() --- double fix function.

ambifixb.c --- void ambifixb() : function for ambiguity fixing in flight state monitoring.

yalman.c --- void kalman() : function of Kalman filtering.

singlep3.c

void earthrot() --- earth rotation correction.

double distance() --- distance between two points.

void transrot() --- transmitting time and Earth rotation effect.

void transrot1() --- transmitting time and Earth rotation effect.

void singlep() --- using code data for single point positioning.

double twoi2f() --- change two integers to a float number.

void PC12() --- decide P1,P2,d1,d2 column number.

void checkDefine() --- check the defined values.

specula.c --- void speculation() : use multiple static results.

dis3.c --- void discon() : introduction of distance condition.

xu.h --- header file of main program ksg44.c.

11.2 Appendix 2: List of utilities programs

sse2x35.c --- GPS Rinex data analysis program.

ana_a53.c --- input control file creation (automatically) program.

changein.c --- change all input control files for one parameter.

coortran.c --- coordinates transformation program.

crinex.c --- clean Rinex data file.

d_sat.c --- delete one satellite from Rinex file.

finds.c --- find a string from all software packages.

read4.c --- read result files for graphics.

rn.c --- ion-free ambiguity fixing computation function.

u7.c --- compare results program.

zhu.c --- transfer broadcasting orbits to IGS orbit.

urinx.c --- uniform Rinex files.
