# 18. Combination of Solutions

## 18.1 Motivation

The increasing number of permanent GPS stations all over the world and the associated big number of observations to be processed ask for sequential processing methods. A “conventional” processing of all observations in one step using, e.g., GPSEST may be appropriate for small campaigns (a few days with 24 hour sessions of about 10-20 sites). The computing power available today does not allow to go far beyond this limit.

The program ADDNEQ was therefore developed to compute multi-session solutions from the (statistically correct) combination of a set of single-session solutions. The theory of combining sequential solutions is well-known in geodesy since [*Helmert*, 1872]. Sequential adjustment techniques are in general independent of the observation types of the individual solutions. This implies, e.g., that even results from different techniques (classical geodetic techniques or space techniques GPS, SLR, VLBI, DORIS) might be combined. Here, we focus on the combination of GPS results, only.

Normal equations may be stored for a sequence of solutions including all possible types of unknown parameters (coordinates, troposphere, orbit parameters, Earth rotation parameters, nutation parameters, center of mass, satellite antenna offsets, etc.).

The special features of the normal equation stacking methods, described in Section 18.3, allow an extremely rapid and flexible computation of many solution types, *without* going back to the original observations (e.g., new definition of the geodetic datum, specification of a priori sigmas for different parameters types, etc.).

We focus on applications and different processing strategies using normal equations in Section 18.4. The computation of velocities from campaign results or from results achieved from permanent GPS networks is one important application. Another topic is the combination of GPS solutions of different analysis centers for the purpose of the *densification of the terrestrial reference frame* using GPS.

A description of the programs COMPAR and ADDNEQ is given in the Sections 18.6 and 18.7. Answers to “frequently asked questions” concerning parameter and normal equation handling conclude this chapter.

## 18.2 Basic Theory of Least-Squares Estimation

### 18.2.1 Least-Squares Estimation

The observation equations in the *Gauss-Markoff Model* (GMM) of full rank is given, e.g., by [*Koch*, 1988]

$$\boldsymbol{E}(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\beta} \quad ; \quad \boldsymbol{D}(\boldsymbol{y}) = \sigma^2 \boldsymbol{P}^{-1} \tag{18.1}$$

with

| | |
|---|---|
| $\boldsymbol{X}$ | $n \times u$ matrix of given coefficients with full rank $rank\ \boldsymbol{X} = u$; $\boldsymbol{X}$ is also called *design matrix*, |
| $\boldsymbol{\beta}$ | $u \times 1$ vector of unknowns, |
| $\boldsymbol{y}$ | $n \times 1$ vector of observations, |
| $\boldsymbol{P}$ | $n \times n$ positive definite weight matrix, |
| $n, u$ | number of observations, number of unknowns, |
| $\boldsymbol{E}(\cdot)$ | operator of expectation, |
| $\boldsymbol{D}(\cdot)$ | operator of dispersion, |
| $\sigma^2$ | variance of unit weight (variance factor). |

The observation equations of Chapter 9 may be written in this form. For $n > u$, the equation system $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ is not consistent. With the addition of the residual vector $\boldsymbol{e}$ to the observation vector $\boldsymbol{y}$, one obtains a consistent but ambiguous system of equations, also called system of *observation equations*:

$$\boldsymbol{y} + \boldsymbol{e} = \boldsymbol{X}\boldsymbol{\beta} \quad \text{with} \quad \boldsymbol{E}(\boldsymbol{e}) = \boldsymbol{\emptyset} \quad \text{and} \quad \boldsymbol{D}(\boldsymbol{e}) = \boldsymbol{D}(\boldsymbol{y}) = \sigma^2 \boldsymbol{P}^{-1}. \tag{18.2}$$

Eqns. (18.1) and (18.2) are formally identical. $\boldsymbol{E}(\boldsymbol{e}) = \boldsymbol{\emptyset}$, because $\boldsymbol{E}(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\beta}$, and $\boldsymbol{D}(\boldsymbol{e}) = \boldsymbol{D}(\boldsymbol{y})$ follows from the law of error propagation.
The *method of least-squares* asks for restrictions for the observation equations (18.1) or (18.2). The parameter estimates $\boldsymbol{\beta}$ should minimize the quadratic form

$$\Omega(\boldsymbol{\beta}) = \frac{1}{\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})' \boldsymbol{P} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{18.3}$$

where $(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})'$ is the transposed matrix of $(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$. The introduction of the condition $\Omega(\boldsymbol{\beta}) \rightarrow$ min. is necessary to lead us from the ambiguous observation Equations (18.1) or (18.2) to an unambiguous normal equation system (NEQ system) for the determination of $\boldsymbol{\beta}$.
The establishment of minimum values for $\Omega(\boldsymbol{\beta})$ leads to a system of $u$ equations $d\Omega(\boldsymbol{\beta})/d\boldsymbol{\beta} = \boldsymbol{\emptyset}$, also called *normal equations*.
The following formulae summarize the *Least-Squares Estimation* (LSE) in the Gauss-Markoff Model:
**Normal equations:**

$$\boldsymbol{X}'\boldsymbol{P}\boldsymbol{X}\widehat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{P}\boldsymbol{y} \tag{18.4}$$

**Estimates:**

$$\text{of } \boldsymbol{\beta} : \quad \widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{P}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{P}\boldsymbol{y} \tag{18.5}$$

of the (variance-)covariance matrix: $\boldsymbol{D}(\widehat{\boldsymbol{\beta}}) = \widehat{\sigma}^2(\boldsymbol{X'PX})^{-1}$ (18.6)

of the observations: $\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ (18.7)

of the residuals: $\widehat{\boldsymbol{e}} = \widehat{\boldsymbol{y}} - \boldsymbol{y}$ (18.8)

of the quadratic form: $\Omega = \widehat{\boldsymbol{e}}'\boldsymbol{P}\widehat{\boldsymbol{e}} = \boldsymbol{y}'\boldsymbol{Py} - \boldsymbol{y}'\boldsymbol{PX}\widehat{\boldsymbol{\beta}}$ (18.9)

of the variance of unit weight (variance factor): $\widehat{\sigma}^2 = \Omega/(n-u)$ (18.10)

**Degree of freedom / Redundancy:**

$$f = n - u \tag{18.11}$$

**Normal equation matrices:**

$$\boldsymbol{X'PX}, \quad \boldsymbol{X'Py}, \quad (\boldsymbol{y'Py}) \tag{18.12}$$

This algorithm is used in the parameter estimation program GPSEST ( Menu 4.5 ).

### 18.2.2 Parameter Pre-elimination

*Pre-elimination* of parameters is a basic procedure to reduce the dimension of the NEQ system *without* loosing information (apart from the parameters pre-eliminated). We do not give the mathematical proof, here. For more information see, e.g., [*Brockmann*, 1996].

The pre-elimination formulae basically compute the effect of the pre-eliminated parameters on the other (remaining) parameters of the normal equation system. As a result, the normal equation matrices (18.12) are modified. Pre-elimination, therefore, is NOT equivalent to cancelling the corresponding lines and columns of the normal equations.

Pre-elimination of parameters using covariance matrices as opposed to pre-elimination using normal equations is much easier. The determination of partial covariance matrices is identical to removing the corresponding rows and columns of the parameters, which have to be eliminated from the covariance matrix.

Pre-elimination of parameters is possible with both, program GPSEST ( Panel 4.5–2.4 , option `PARAMETER PRE-ELIMINATION`) and program ADDNEQ ( Panel 4.8.1–2 , option `PARAMETER PRE-ELIMINATION`). It is the responsibility of the user to decide at which stage of the processing to pre-eliminate parameters from the NEQ system (mainly a question of processing time and disk space). More information is given in Section 18.4 and in the examples in Chapter 4.

Let us distinguish between the pre-elimination options `BI` (before Inversion), `AI` (after inversion), and `EP` (epoch-wise, GPSEST only):

`BI` : Pre-elimination of a parameter *before inversion*.
Used mainly for ambiguity parameters in GPSEST (if there remain unresolved ambiguities from previous GPSEST runs) or for troposphere parameters in ADDNEQ (if they were stored previously in the normal equations) to reduce the number of unknowns in the combined solution.

`AI` : Pre-elimination of a parameter *after inversion*.
Used in GPSEST and ADDNEQ to store only the parameters of interest in the normal equation files. Ambiguity parameters have to be pre-eliminated using option `AI` in GPSEST if normal equations are stored.

`EP` : Pre-elimination of a parameter directly after each observation epoch.
Used in GPSEST for epoch-specific parameters such as, e.g., kinematic coordinates or stochastic ionosphere parameters.

### 18.2.3 Sequential Least-Squares Estimation

In this section we review the concept of sequential least-squares estimation techniques. The result of a LSE using all observations in one step is the same as when splitting up the LSE in different parts and combining the results later.
To prove the identity of both methods we first solve for the parameters according to the common adjustment in one adjustment step. Thereafter, we verify that the same result is obtained using a sequential adjustment.
Let us start with the observation equations:

$$\begin{array}{rclcl} \boldsymbol{y}_1 + \mathbf{e}_1 & = & \boldsymbol{X}_1\,\boldsymbol{\beta_c} & \text{with} & \boldsymbol{D}(\boldsymbol{y}_1) = \sigma_1^2 \boldsymbol{P}_1^{-1} \\ \boldsymbol{y}_2 + \mathbf{e}_2 & = & \boldsymbol{X}_2\,\boldsymbol{\beta_c} & \text{with} & \boldsymbol{D}(\boldsymbol{y}_2) = \sigma_2^2 \boldsymbol{P}_2^{-1}. \end{array} \tag{18.13}$$

In this case we divide the observation array $\boldsymbol{y}_c$ (containing all observations) into two independent observation series $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. We would like to estimate the parameters $\boldsymbol{\beta_c}$ common to both parts using both observation series $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. We assume furthermore, that there are no parameters which are relevant for one of the individual observation series, only. This assumption is meaningful if we pre-eliminate “uninteresting” parameters according to Section 18.2.2.
The proof of the equivalence of both methods is based on the assumption that both observation series are independent.
The division into two parts is suffuciently general. If both methods are leading to the same result, we might derive formulae for additional sub-divisions by assuming one observation series to be already the result of an accumulation of different observation series.

#### 18.2.3.1 Common Adjustment

In matrix notation we may write the observation equations (18.13) in the form:

$$\begin{gathered} \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta_c} \end{bmatrix} \\ \text{with} \quad \boldsymbol{D}(\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix}) = \sigma_c^2 \begin{bmatrix} \boldsymbol{P}_1^{-1} & \boldsymbol{\emptyset} \\ \boldsymbol{\emptyset} & \boldsymbol{P}_2^{-1} \end{bmatrix} \end{gathered} \tag{18.14}$$

which is equivalent to

$$\boldsymbol{y}_c + \boldsymbol{e}_c = \boldsymbol{X}_c \boldsymbol{\beta_c} \quad \text{with} \qquad \boldsymbol{D}(\boldsymbol{y}_c) = \sigma_c^2 \boldsymbol{P}_c^{-1}. \tag{18.15}$$

The matrices $\boldsymbol{y}_c$, $\boldsymbol{e}_c$, $\boldsymbol{X}_c$, $\boldsymbol{\beta_c}$, and $\boldsymbol{P}_c^{-1}$ may be obtained from the comparison of eqn. (18.15) with Eqn. (18.14). The independence of both observation series is given by the special form of the dispersion matrix (zero values for the off-diagonal elements). Substitution of the appropriate values for $\boldsymbol{y}_c$, $\boldsymbol{X}_c$ and $\boldsymbol{\beta_c}$ in Eqn. (18.4) leads to the normal equation system of the LSE:

$$\begin{bmatrix} \boldsymbol{X}_1' \boldsymbol{P}_1 \boldsymbol{X}_1 + \boldsymbol{X}_2' \boldsymbol{P}_2 \boldsymbol{X}_2 \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_1' \boldsymbol{P}_1 \boldsymbol{y}_1 + \boldsymbol{X}_2' \boldsymbol{P}_2 \boldsymbol{y}_2 \end{bmatrix}. \tag{18.16}$$

#### 18.2.3.2 Sequential Least-Squares Adjustment

In a first step the sequential LSE treats each observation series independently. An estimation is performed for the unknown parameters using only the observations of a particular observation series.

In a second step the contribution of each sequential parameter estimation to the common estimation is computed.
Starting with the same observation equations as in the previous section, Eqns. (18.13), we may write

$$\begin{aligned} \boldsymbol{y}_1 + \boldsymbol{e}_1 &= \boldsymbol{X}_1\,\boldsymbol{\beta_1} \quad \text{with} \quad \boldsymbol{D}(\boldsymbol{y}_1) = \sigma_1^2 \boldsymbol{P}_1^{-1} \\ \boldsymbol{y}_2 + \boldsymbol{e}_2 &= \boldsymbol{X}_2\,\boldsymbol{\beta_2} \quad \text{with} \quad \boldsymbol{D}(\boldsymbol{y}_2) = \sigma_2^2 \boldsymbol{P}_2^{-1} \end{aligned} \tag{18.17}$$

or, in more general notation:

$$\boldsymbol{y}_i + \boldsymbol{e}_i = \boldsymbol{X}_i\,\boldsymbol{\beta_i} \quad \text{with} \quad \boldsymbol{D}(\boldsymbol{y}_i) = \sigma_i^2 \boldsymbol{P}_i^{-1},\ i = 1,2 \tag{18.18}$$

where the vector $\boldsymbol{\beta_i}$ denotes the values of the common parameter vector $\boldsymbol{\beta_c}$ satisfying observation series $\boldsymbol{y}_i$ only.

## First step: Solving for each individual NEQ

The normal equations for the observation equation systems $i = 1,2$ may be written according to Eqn. (18.4) as

$$\left[\ \boldsymbol{X}_i'\boldsymbol{P}_i\boldsymbol{X}_i\ \right]\left[\ \widehat{\boldsymbol{\beta}}_i\ \right] = \left[\ \boldsymbol{X}_i'\boldsymbol{P}_i\boldsymbol{y}_i\ \right] \tag{18.19}$$

$$\begin{aligned} \boldsymbol{D}(\widehat{\boldsymbol{\beta}}_i) &= \widehat{\sigma}_i^2\,(\boldsymbol{X}_i'\boldsymbol{P}_i\boldsymbol{X}_i)^{-1} \\ &= \widehat{\sigma}_i^2\boldsymbol{\Sigma}_i \quad \text{with} \quad i = 1,2. \end{aligned} \tag{18.20}$$

## Step 2: A posteriori LSE

In this a posteriori LSE step, the estimation for $\widehat{\boldsymbol{\beta}}_c$ is derived using the results of the individual solutions (18.19) and (18.20) obtained in the first step.
The pseudo-observation equations set up in this second step have the following form

$$\boldsymbol{y}_{II} + \boldsymbol{e}_{II} = \boldsymbol{X}_{II}\widehat{\boldsymbol{\beta}}_c \quad \text{with} \quad \boldsymbol{D}(\boldsymbol{y}_{II}) = \sigma_c^2 \boldsymbol{P}_{II}^{-1} \tag{18.21}$$

or more explicitly:

$$\begin{bmatrix} \widehat{\boldsymbol{\beta}}_1 \\ \widehat{\boldsymbol{\beta}}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}_{1_{II}} \\ \boldsymbol{e}_{2_{II}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix} \widehat{\boldsymbol{\beta}}_c \quad \text{with} \quad \boldsymbol{D}(\begin{bmatrix} \widehat{\boldsymbol{\beta}}_1 \\ \widehat{\boldsymbol{\beta}}_2 \end{bmatrix}) = \sigma_c^2 \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{\emptyset} \\ \boldsymbol{\emptyset} & \boldsymbol{\Sigma}_2 \end{bmatrix}.$$

The results of the individual estimations $\widehat{\boldsymbol{\beta}}_i$ and $\boldsymbol{\Sigma}_i$ are thus used to form the combined LSE. The interpretation of this pseudo-observation equation system is easy: Each estimation is introduced as a new observation using the associated covariance matrix as the corresponding weight matrix.
The normal equation system may be written as:

$$\boldsymbol{X}_{II}'\boldsymbol{P}_{II}\boldsymbol{X}_{II}\widehat{\boldsymbol{\beta}}_c = \boldsymbol{X}_{II}'\boldsymbol{P}_{II}\boldsymbol{y}_{II} \tag{18.22}$$

or more explicitly

$$\begin{aligned} &\left[\ \boldsymbol{I}', \boldsymbol{I}'\ \right] \begin{bmatrix} \boldsymbol{\Sigma}_1^{-1} & \boldsymbol{\emptyset} \\ \boldsymbol{\emptyset} & \boldsymbol{\Sigma}_2^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix} \widehat{\boldsymbol{\beta}}_c \\ &= [\boldsymbol{I}', \boldsymbol{I}'] \begin{bmatrix} \boldsymbol{\Sigma}_1^{-1} & \boldsymbol{\emptyset} \\ \boldsymbol{\emptyset} & \boldsymbol{\Sigma}_2^{-1} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}}_1 \\ \widehat{\boldsymbol{\beta}}_2 \end{bmatrix}. \end{aligned} \tag{18.23}$$

Substituting the results for $\boldsymbol{\Sigma}_i^{-1}$ we obtain

$$\left[\ \boldsymbol{X}_1'\boldsymbol{P}_1\boldsymbol{X}_1 + \boldsymbol{X}_2'\boldsymbol{P}_2\boldsymbol{X}_2\ \right]\ \widehat{\boldsymbol{\beta}}_c\ = \left[\ \boldsymbol{X}_1'\boldsymbol{P}_1\boldsymbol{y}_1 + \boldsymbol{X}_2'\boldsymbol{P}_2\boldsymbol{y}_2\ \right] \tag{18.24}$$

which is identical with Eqn. (18.19). This simple superposition of normal equations, also called *stacking* of normal equations, is always possible if the individual observation series are independent (which is the case if the dispersion matrix has the form (18.14)).

#### 18.2.3.3 Computation of the Combined RMS

In the previous section we only considered the combined parameter estimation. Sequential LSE leads to identical results for the a posteriori estimate of the variance of unit weight:

$$\Omega_c = \sum_{i=1}^{m}\boldsymbol{y}_i'\boldsymbol{P}_i\boldsymbol{y}_i - \sum_{i=1}^{m}\boldsymbol{y}_i'\boldsymbol{P}_i\boldsymbol{X}_i\widehat{\boldsymbol{\beta}}_c \tag{18.25}$$

$$\widehat{\sigma}_c^2 = \left(\sum_{i=1}^{m}\boldsymbol{y}_i'\boldsymbol{P}_i\boldsymbol{y}_i - \sum_{i=1}^{m}\boldsymbol{y}_i'\boldsymbol{P}_i\boldsymbol{X}_i\widehat{\boldsymbol{\beta}}_c\right)/f_c. \tag{18.26}$$

The importance of the "third normal equation part" $\boldsymbol{y'Py}$ (see Eqn. 18.12) is clearly seen in this formula. We refer to [*Brockmann*, 1996] for a complete discussion.

## 18.3 Special Features of Combining Normal Equations

Special features are "the salt in the soup" when dealing with normal equations. Here, we present only some important ones. Most of them may be derived from general parameter transformation rules applied to normal equations (see [*Brockmann*, 1996]).

### 18.3.1 Constraining Parameters

In general, the observations of a given type are *not* sensitive to *all* parameters in a theoretical model. In this case the normal equations (NEQs) are singular.
Additional information, or *constraints*, must be introduced into the least-squares solution to make the normal equations non-singular. Additional constraints may be useful also for parameters which would be estimated with a very high rms. Let us introduce "exterior" information concerning the parameters

$$\boldsymbol{H\beta} = \boldsymbol{w} + \boldsymbol{e}_w \quad \text{with} \quad \boldsymbol{D}(\boldsymbol{w}) = \sigma^2\boldsymbol{P}_w^{-1} \tag{18.27}$$

where

| | |
|---|---|
| $\boldsymbol{H}$ | $r \times u$ matrix with given coefficients with $rank\ \boldsymbol{H} = r$, |
| $r$ | number of constraining equations with $r < u$, |
| $\boldsymbol{\beta}$ | vector of unknown parameters with dimension $u \times 1$, |

| | |
|---|---|
| $\boldsymbol{w}$ | $r \times 1$ vector of known constants, |
| $\boldsymbol{e}_w$ | $r \times 1$ residual vector, and |
| $\boldsymbol{P}_w^{-1}$ | dispersion matrix of the introduced constraining equations with dimension $r \times r$. |

If the constraints are non-linear, a linearization has to be performed through a first-order Taylor series expansion.
We may interpret the constraints (18.27) as additional pseudo-observations, or as *fictitious observations*. That leads us to the observation equations:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{w} \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}_y \\ \boldsymbol{e}_w \end{bmatrix} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{H} \end{bmatrix} \widehat{\boldsymbol{\beta}} \quad \text{with} \quad \boldsymbol{D}(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{w} \end{bmatrix}) = \sigma^2 \begin{bmatrix} \boldsymbol{P}^{-1} & \boldsymbol{\emptyset} \\ \boldsymbol{\emptyset} & \boldsymbol{P}_w^{-1} \end{bmatrix} \tag{18.28}$$

or to the associated NEQ system:

$$(\boldsymbol{X}'\boldsymbol{P}\boldsymbol{X} + \boldsymbol{H}'\boldsymbol{P}_w\boldsymbol{H})\widehat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{P}\boldsymbol{y} + \boldsymbol{H}'\boldsymbol{P}_w\boldsymbol{w}. \tag{18.29}$$

The equation shows, that we may superpose the terms $\boldsymbol{H}'\boldsymbol{P}_w\boldsymbol{H}$ and $\boldsymbol{H}'\boldsymbol{P}_w\boldsymbol{w}$ to the original normal equation system to incorporate a priori information on the parameters. The values of these quantities have to be stored in the normal equation files (or in the SINEX files (see Section 24.8.13)). The terms must be removed if a “free” solution (without any a priori constraints) has to be created.
Constraints may be introduced in GPSEST and ADDNEQ for the following parameter types:

- coordinates: absolute constraints (station weights), station fixing, free network constraints,
- velocities: absolute and relative (concerning sites) constraints,
- troposphere: absolute and relative (in time) constraints,
- orbit: Keplerian, dynamical, stochastic parameters,
- center of mass,
- Earth rotation parameters: absolute constraints (UT1 and nutation absolute value has to be constrained to a VLBI value) and continuity constraints,
- satellite antenna offsets.

The most important features dealing mainly with the first three types of parameters are explained in more detail in the application part of this chapter.

### 18.3.2 Introducing Additional Parameters

The introduction of additional parameters is possible, even if these parameter types have not been set up in the individual normal equations. Site motion parameters, or site velocities, are an example for such parameter types. It is necessary, that the influence of these parameters may be neglected within the individual normal equations. For the site velocities, e.g., the influence of the site motion may actually be neglected for the time span of one day. How to estimate velocities is shown later in this chapter.
Another example for the setup of additional parameters is the estimation of Helmert parameters between individual solutions (see Section 24.8.14).

### 18.3.3 Independence of the A Priori Information

The used a priori values of the parameters are stored in the normal equations. This is important because the normal equations refer to the parameter increments (the difference between the estimated final parameter values and the a priori values). The normal equations may be transformed to an arbitrary set of a priori parameter values if higher order terms of the original non-linear observation equations may be neglected. The transformation to new a priori values is done automatically (if necessary) without user interaction.

### 18.3.4 Free Network Constraints

Free network solutions are optimal to define the geodetic datum with a minimum number of constraints, without fixing or constraining particular site coordinates. This option is well-suited to analyze inconsistencies in the reference site coordinates.
A geodetic datum may be defined in the following way:

- An a priori network is defined by selecting a list of sites (in the menu system of ADDNEQ "fixed" sites are used for this purpose).
- Helmert parameters may be specified (translations, rotations, scale). Depending on the selection of these parameters, the final parameter estimation has the property that the network results show no translations / rotations / scale with respect to the a priori network.

More information may be found in Section 18.7.3.

### 18.3.5 Reduction of the Number of Unknown Parameters

As opposed to adding new parameters, it is also possible to reduce the number of unknown parameters in the normal equations. An important application is the reduction of the number of troposphere parameters. If you have, e.g., estimated (and stored into NEQ files) 12 troposphere parameters per site and day, you have the possibility to reduce the number of parameters to 1, 2, 3, 4, or 6 values per day and site. This option is available in Panel 4.8.1–2.2, option `NUMBER OF PARAMETERS PER DAY`. See also Section 18.7.6.

### 18.3.6 Limitations of NEQ Stacking

Let us also mention what is *not* possible using ADDNEQ:

- model modifications which are highly time-dependent (e.g., a different tropospheric mapping function, different a priori tide model, etc.),
- ambiguity resolution,
- different basic observation types (e.g., to switch from $L_1$ and $L_2$ to $L_3$, etc.).

## 18.4 Applications and Strategies Using Normal Equations

It is possible to combine results based on normal equations without loss of information. Figure 18.1 shows an example how results are combined at CODE. Starting from cluster solutions, network solutions are created. These results are then used for long-arc applications. Finally, weekly, monthly, or annual solutions may be created.

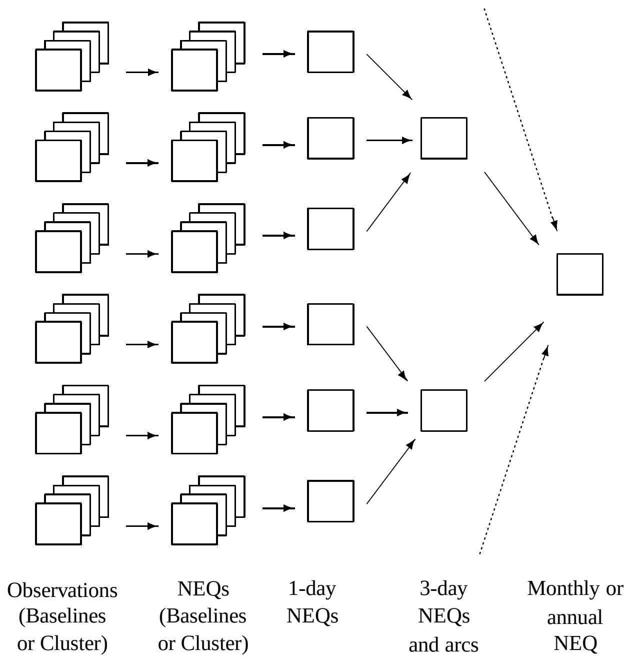

Observations (Baselines or Cluster)
NEQs (Baselines or Cluster)
1-day NEQs
3-day NEQs and arcs
Monthly or annual NEQ

**Figure 18.1:** Combination of the normal equations of different processing steps.

There is a wide area of applications for combination methods. Below, we briefly review some important applications when processing GPS observations.

Baseline processing mode:

Figure 18.2 demonstrates how baseline or cluster results are combined into a network solution.
The baseline processing scheme (also implemented in the BPE processing example; see also Chapter 4) has the advantage that the computational burden *increases only linearly with the number of sites*. It is a disadvantage, however, that *inter-baseline correlations* are not taken into account. For highest accuracy requirements we therefore recommend to process all observations with GPSEST using the correct handling of the correlations ( Panel 4.5–2 , option `CORRELATIONS`, select `CORRECT`). For big networks this may not be possible due to limited computer resources (memory and computing time). As a compromise between statistical correctness and computational efficiency you may define *clusters* of observations and process each cluster using the option "correct correlations". Afterwards you may combine the cluster normal equations (instead of baseline normal equations) into a network solution. The use of cluster definition files for processing clusters of observations is explained in Sections 24.8.30 and 24.8.31.

Multi-Days Solutions:

The creation of weekly or monthly solutions from daily solutions is sometimes useful to reduce the variations in the coordinate solutions. The noise of, e.g., weekly coordinate residuals is smaller by a factor of $1/\sqrt{7}$ in a weekly solution when compared to the daily solution.

Multi-Years Solutions or Multi-Campaigns Solutions:

The computation of "final" coordinates as a result of many days of continuous observations or several campaigns is the main goal of the combination of solutions. The program ADDNEQ was originally developed for this purpose. This includes also the detection of movements (estimation of velocities), which is described in Section 18.7.5.

Orbit Combination:

Orbit combination is probably not of great interest for the majority of users. The orbit combination method described in [*Beutler et al.*, 1996] and [*Brockmann*, 1996], (see also Chapter 8) is an extremely flexible tool for orbit determination purposes. Long arcs (e.g., 3-days-arcs) may be computed from short arcs (e.g., 1-day-arcs) in a very efficient way (gain of more than a factor of 10 in processing time). Many more options are available such as setting up stochastic parameters at the arc boundaries, splitting up of arcs, etc.

Combination of Solutions using Results of Different Processing Centers:

The combination of (GPS-) solutions derived by different analysis centers is a major activity within the IGS with the goal of densifying the International Terrestrial Reference Frame (ITRF). The *distributed processing concept* makes it possible that regional or local analysis centers (in the IGS naming convention: Regional Network Associated Analysis Centers RNAACs) may compute their sites of interest together with global IGS sites (also called *anchor sites*). These solutions may then be combined by Global Network Associated Analysis Centers (GNAACs) together with the global solutions of the IGS analysis centers to form a consistent network solution. It is not necessary in this concept that all contributing sites are processed by one analysis center. Combination strategies and results are shown by, e.g., [*Davies and Blewitt*, 1995] and [*Brockmann and Gurtner*, 1996].

The SINEX (see Section 24.8.13) exchange format is used for the combination of the results of different analysis centers (see Sections 7.3 and 24.8.13). Using the program SNXNEQ (see Section 7.3.3) it is possible to convert SINEX files (`.SNX`) into normal equation files (`.NEQ`), which may be used as input files for ADDNEQ. The combination of results derived from different software packages asks for the determination of normal equation rescaling factors to ensure that each contributing solution gets the "correct" weight. Rescaling factors may be specified using a special weighting (`.WGT`) file (see Section 24.8.14). The estimation of these factors using the methods of the *variance-covariance component estimation* is not supported, yet.

## 18.5 The Combination Programs ADDNEQ and COMPAR

Two programs are available to combine solutions in the *Bernese GPS Software* Version 4.2: ADDNEQ ( Menu 4.8.1 ) (or its new version ADDNEQ2 ( Menu 4.8.3 ), see also Chapter 19 )

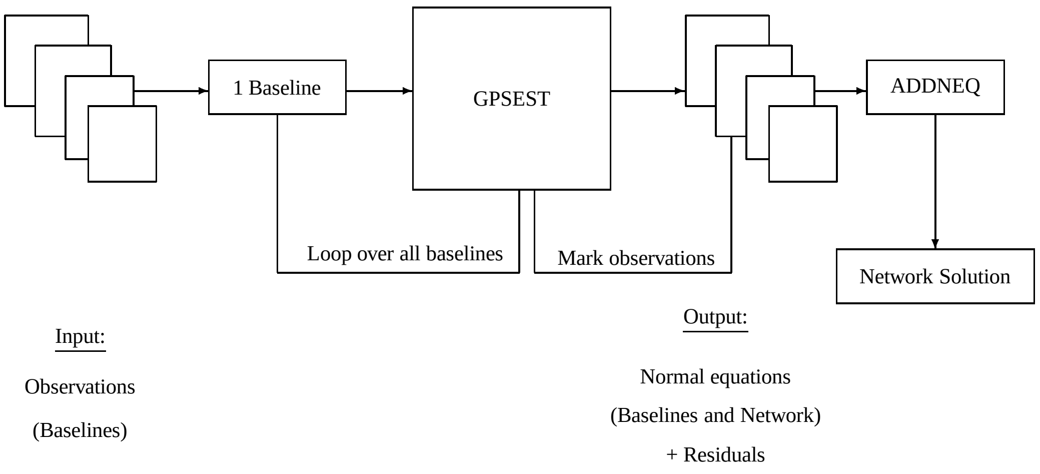
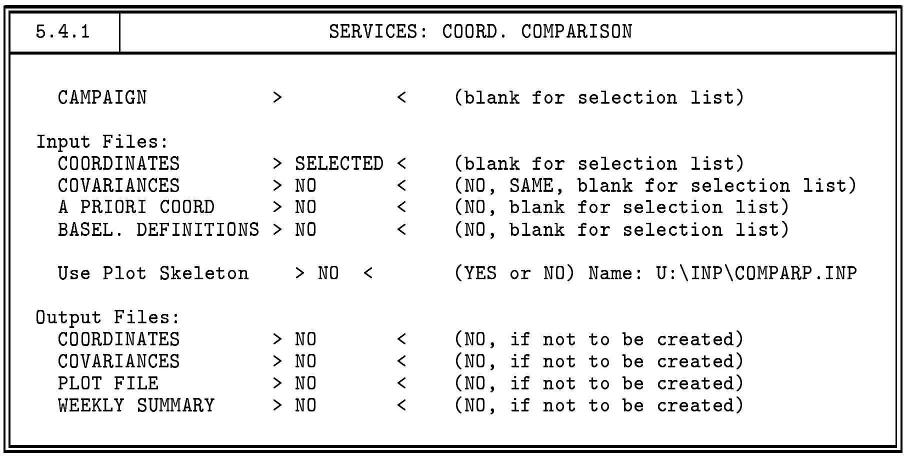


**Figure 18.2:** Processing scheme based on baseline (or bluster) processing.

and COMPAR ( Menu 5.4.1 ). ADDNEQ is based on normal equations and is able to handle all types of unknown parameters, COMPAR is based on covariance information of coordinates, only. In general, it is equivalent to combine solutions based on normal equations or based on covariance information. ADDNEQ is much more flexible, COMPAR is much simpler to use. The selection of the tool depends on the user requirements. We discuss both programs.

## 18.6 Combination Program COMPAR

Menu 5.4.1 is used to prepare a run of program COMPAR. All important features of this program are activated using Panel 5.4.1 .


```
5.4.1                       SERVICES: COORD. COMPARISON

  CAMPAIGN           >          <    (blank for selection list)

Input Files:
  COORDINATES        > SELECTED <    (blank for selection list)
  COVARIANCES        > NO       <    (NO, SAME, blank for selection list)
  A PRIORI COORD     > NO       <    (NO, blank for selection list)
  BASEL. DEFINITIONS > NO       <    (NO, blank for selection list)

  Use Plot Skeleton    > NO  <       (YES or NO) Name: U:\INP\COMPARP.INP

Output Files:
  COORDINATES        > NO       <    (NO, if not to be created)
  COVARIANCES        > NO       <    (NO, if not to be created)
  PLOT FILE          > NO       <    (NO, if not to be created)
  WEEKLY SUMMARY     > NO       <    (NO, if not to be created)
```

The program COMPAR is used to compare different coordinate sets (select input files COORDINATES) without allowing for additional Helmert parameters between the different sets. We mentioned already that the program is also suited to compare the coordinates using the associated variance-covariance information (select COVARIANCES). Keep in mind, that with this program you do not have the flexibility to change the constraints specified in GPSEST, or to change the geodetic datum. It is, e.g., not possible to combine coordinate sets which were computed using different fixed or heavily constrained sites. We therefore recommend to use the program ADDNEQ if a statistically correct combination should be performed.

The program is well-suited to study coordinate repeatabilities and baseline results. By setting two options you may activate the printing of baseline repeatabilities in the output file. The first option is `BASEL. DEFINITIONS` in Panel 5.4.1 (see Section 24.8.29) and the second one has to be set in Panel 5.4.1–1, (`Repeatability option`; select `LOCAL or GEOcentric`). All this information may also be obtained using ADDNEQ. For an output description we refer to Section 18.7.7, because most of the output information is very similar to the output of ADDNEQ.

It is possible to create a summary file (`WEEKLY SUMMARY`) using COMPAR. Agencies participating in the IGS densification project may use this possibility to automatically create a summary file for their weekly submission of SINEX results.

## 18.7 Combination Program ADDNEQ

### 18.7.1 General Introduction

Most of the input options in ADDNEQ are identical to the options available in program GPSEST. This includes the handling of different parameter types (e.g., coordinates, troposphere, orbits, center of mass, etc.). We therefore will not repeat all different input options in detail, here. Furthermore, we refer to the available `HELP` panels if questions concerning a specific input option arise.
We put the emphasis on the differences with respect to the parameter estimation using GPSEST and to the additional features, which are available in ADDNEQ, only.

### 18.7.2 Differences to GPSEST

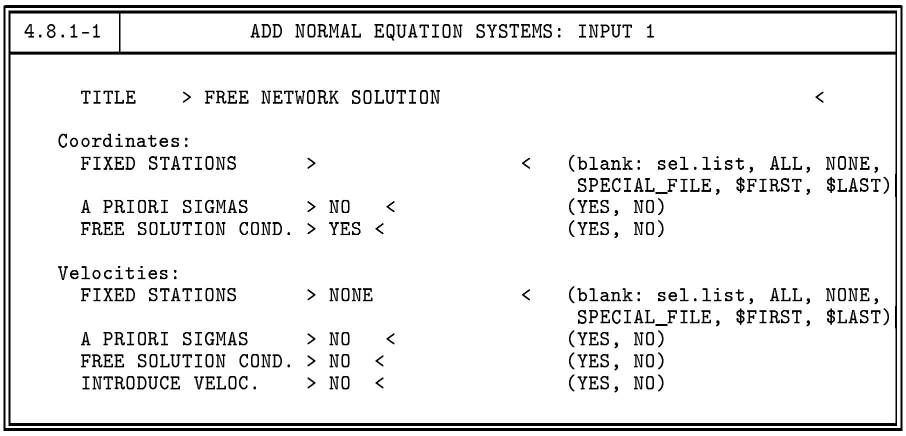
Below we summarize important differences between GPSEST and ADDNEQ:

- An a priori coordinate file has to be specified in GPSEST, only. ADDNEQ does not need such a file because the information is already stored in the normal equations. A coordinate file, specified in Panel 4.8.1, (see Figure 18.4) option `UPDATE CRD.` is only used as a master file to create a coordinate output file (specified in Panel 4.8.1–0, option `COORDINATES`).
- Fixing site coordinates in GPSEST is equivalent to not setting up these parameters as unknowns (not recommended, if normal equations are stored; see Section 18.8). Fixing site coordinates in ADDNEQ is equivalent to specifying an a priori sigma of 0.01 mm.
- Troposphere handling is much simpler than in GPSEST. In ADDNEQ you only have the possibility to specify a general absolute and a relative a priori sigma. These values are valid for all troposphere parameters stored in the normal equations. It is not possible to handle different sites in a different way (e.g., to constrain the troposphere parameters of some sites more than others).
- ADDNEQ features, which are NOT available in GPSEST:
  - Velocity estimation (more details are given below),

– Creation of a SINEX file (see e.g. Section 24.8.13 and Section 7.3),
– Free network solutions (more details are given below),
– Long-arc computation based on one-day-arcs (not explained in detail in this documentation),
– Special handling of the Earth rotation parameters (see Chapter 14).

## 18.7.3 Free Coordinate Solutions

Let us focus on some important aspects when generating so-called “free network solutions” (compare also Section 18.3.4):



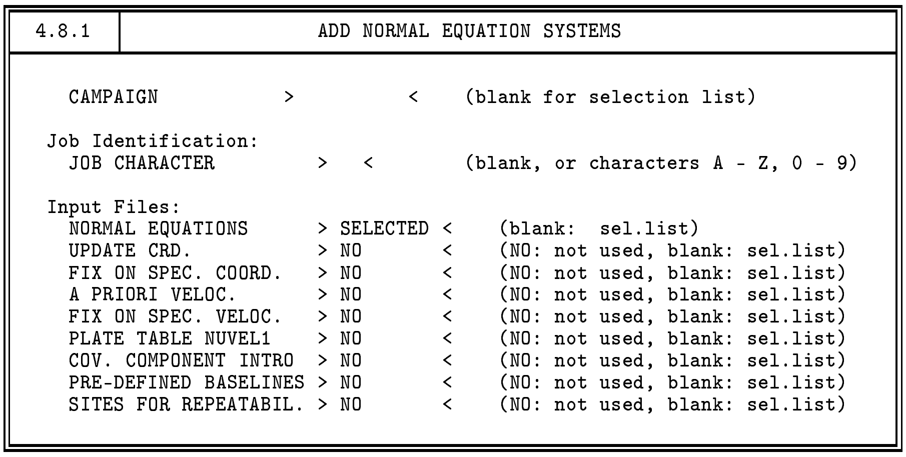
```
4.8.1-1            ADD NORMAL EQUATION SYSTEMS: INPUT 1

   TITLE    > FREE NETWORK SOLUTION                               <

 Coordinates:
   FIXED STATIONS      >                  <   (blank: sel.list, ALL, NONE,
                                               SPECIAL_FILE, $FIRST, $LAST)
   A PRIORI SIGMAS     > NO   <               (YES, NO)
   FREE SOLUTION COND. > YES <                (YES, NO)

 Velocities:
   FIXED STATIONS      > NONE             <   (blank: sel.list, ALL, NONE,
                                               SPECIAL_FILE, $FIRST, $LAST)
   A PRIORI SIGMAS     > NO   <               (YES, NO)
   FREE SOLUTION COND. > NO  <                (YES, NO)
   INTRODUCE VELOC.    > NO  <                (YES, NO)
```

**Figure 18.3:** Panel 4.8.1–1 options to define the geodetic datum of a solution.

- You may activate the free network solutions in Panel 4.8.1–1 (see Figure 18.3).
- “Fixed” stations have to be specified to define the a priori sites used for defining the reference network. Please select `FIXED STATIONS` using `blank`, using `ALL` sites, or using a `SPECIAL_FILE`. Do **not** use `A PRIORI SIGMAS` in this case.
- Identical free network options may also be defined for the velocities.
- You have to introduce Helmert constraints in Panel 4.8.1–1.1 (or Panel 4.8.1–1.2 for the velocities). Introduction of, e.g., 3 translation constraints is comparable (in view of the number of constraints) to keeping one site fixed (one site velocity vector fixed, respectively). We recommend to introduce 3 translation constraints for the definition of the geodetic datum of the coordinates. If you would like to align your solution to a specific network (and not to ITRF given by the used IGS orbits) you may also specify 3 rotation constraints. The scale of your solution should be estimated from the GPS data without any restrictions from the a priori network.
  In the case of velocities we recommend to introduce 3 translation parameters. Scale and rotation constraints have the same effects as the translation constraints, if you specify `NO` “`FIX ON SPEC. VELOC.`” file in Panel 4.8.1 (see Figure 18.4).
- The residuals of each individual solution with respect to the combined solution (see Section 18.7.7) are computed after applying a 7-parameter transformation (independently of the specified Helmert parameters) using all sites (independently of the selected “fixed” sites). The

application of the free network option has many advantages, in particular if you compare different solutions with only a small number of common sites or if an unique definition of the geodetic datum for **all** contributing solutions is difficult to realize.

- If a special file for fixing coordinates is specified (in Panel 4.8.1 (see Figure 18.4) for coordinates using the option `FIX ON SPEC. COORD.` or for velocities using the option `FIX ON SPEC. VELOC.`), the free network constraints are computed using the coordinate or velocity values of the selected "fixed" sites in these files instead of using the a priori coordinates originally used in GPSEST or instead of using a zero-velocity field. More information will be given in the next section.



```
4.8.1                    ADD NORMAL EQUATION SYSTEMS

  CAMPAIGN           >         <    (blank for selection list)

 Job Identification:
  JOB CHARACTER         >   <        (blank, or characters A - Z, 0 - 9)

 Input Files:
  NORMAL EQUATIONS      > SELECTED <    (blank:  sel.list)
  UPDATE CRD.           > NO       <    (NO: not used, blank: sel.list)
  FIX ON SPEC. COORD.   > NO       <    (NO: not used, blank: sel.list)
  A PRIORI VELOC.       > NO       <    (NO: not used, blank: sel.list)
  FIX ON SPEC. VELOC.   > NO       <    (NO: not used, blank: sel.list)
  PLATE TABLE NUVEL1    > NO       <    (NO: not used, blank: sel.list)
  COV. COMPONENT INTRO  > NO       <    (NO: not used, blank: sel.list)
  PRE-DEFINED BASELINES > NO       <    (NO: not used, blank: sel.list)
  SITES FOR REPEATABIL. > NO       <    (NO: not used, blank: sel.list)
```

**Figure 18.4:** The first option input (Panel 4.8.1) of ADDNEQ.

### 18.7.4 Fixing Coordinates or Velocities on Special Values

Fixing coordinates or velocities on given values (different from the original a priori values) is useful when defining the geodetic datum with fixed sites.

The option may be useful if data were processed using "bad" a priori coordinates (but not exceeding a few decimeters). For the final parameter estimation you then may define the geodetic datum by fixing site coordinates to more meaningful values (e.g. the ITRF92, ITRF93, or ITRF94 values).
The procedure works only if the differences (new − old a priori coordinates) are still in the linear domain of the observation equations.

Constraining and also fixing of coordinates or velocities *without* specifying a special fixing file means that the constraints are set up with respect to the a priori values originally used in GPSEST when saving the normal equations. For coordinates, the a priori values are taken from the normal equation file in which a specific site appears first.
For velocities, the default reference is a zero-velocity field.
If you specify a special fixing file (in Panel 4.8.1 (see Figure 18.4) for coordinates using option `FIX ON SPEC. COORD.` or for velocities using option `FIX ON SPEC. VELOC.`), the same restrictions are set up as without specifying these files, but now with $\boldsymbol{w} \neq \emptyset$ (see Eqn. 18.27), with $\boldsymbol{w}$ as the difference between the original a priori values and the new specified a priori values. These new a priori values are used from the specified fixing file for the selected fixed sites, only.

Note that it is not possible to save normal equations in this case (see Section 18.8). Note also that *special fixing files are* **not** *used for the constraints* (set up using options `A PRIORI SIGMAS` in Panel 4.8.1–1 (see Figure 18.3)).

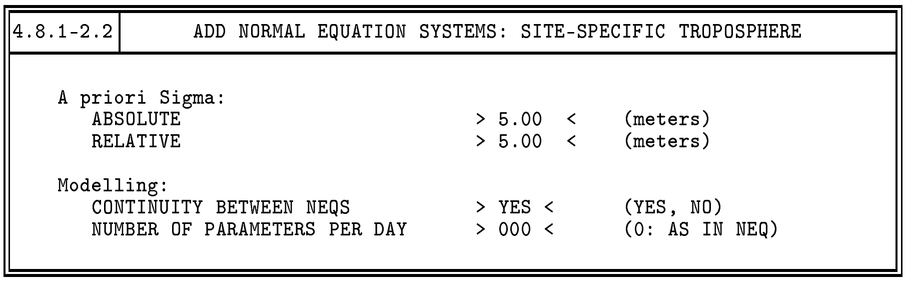
### 18.7.5 Site Velocity Estimation

The estimation of site velocities is an important application of ADDNEQ. Velocity estimation is possible if you have processed data covering a long time span. The quality of individual coordinate estimates is an important factor as well (see e.g. [*Brockmann*, 1996]). It is easy to invoke a velocity estimation: Specify `YES` in Panel 4.8.1–1 (see Figure 18.3), option `A PRIORI SIGMAS`. Then you will get a list of the sites for which you may specify a priori sigmas in units of mm per year. This list is similar to the list you get if you specify `A PRIORI SIGMAS` for site coordinates. Without specifying a priori sigmas no site velocities are estimated. We recommend to use a value of e.g. 999.99 mm/yr per component if you would like to perform a free velocity estimation. We recommend to solve for horizontal velocities, only (to specify e.g. 0.01 mm/yr for the vertical components of the velocities) if you do not have very long time spans of data.
Velocities are also set up if you select "fixed stations" using Panel 4.8.1–1 (see Figure 18.3), option `FIXED STATIONS` for velocities. The consequences of "fixed" velocities was explained already in Section 18.7.3 and Section 18.7.4.
Estimating *one velocity common to several site occupations* is not supported by the menu system. If you edit the I-file of ADDNEQ (see Chapter 3 or Section 24.9) and place an asterisk "`*`" behind the *station number* (see Section 24.8.1) and the associated a priori velocity sigmas, identical velocities will be estimated for all sites with the same station number. If sites with different station numbers are processed, you may use the station problem file (see Section 24.4.13) to change station numbers for this purpose. Keep in mind that in all other cases the site velocities are set up using the *station name*, only.

### 18.7.6 Tuning Troposphere Estimates

Troposphere parameterization very much depends on the size of the GPS network and the session lengths involved. It is not possible to come up with a list stating for all possible applications the number of troposphere parameters per day and site necessary and their associated a priori sigmas. You have to find the optimal parameterization and the optimal values for the a priori sigmas by tests of your own ("trial and error"). In Chapters 4 and 12 you find more information.
ADDNEQ may assist you to find the optimum:

- Store the troposphere parameter information in the NEQ files in GPSEST (see Section 18.2.2).
- Use ADDNEQ and modify the troposphere options in Panel 4.8.1–2.2 (see Figure 18.5): Reduce the number of unknown parameters according to Section 18.3.5 using option `NUMBER OF PARAMETERS PER DAY` and modifying the a priori sigmas `ABSOLUTE` and `RELATIVE`. Furthermore, it may be useful to force `CONTINUITY BETWEEN` the troposphere estimates of two consecutive `NEQS` (the specified relative sigmas are used for this purpose; pre-elimination of the troposphere parameters (option `BI`) is not allowed in this case).
- Check if the coordinate repeatabilities, listed in the ADDNEQ program output (see Section 18.7.7), are improving or not.



```
4.8.1-2.2        ADD NORMAL EQUATION SYSTEMS: SITE-SPECIFIC TROPOSPHERE

   A priori Sigma:
      ABSOLUTE                             > 5.00  <     (meters)
      RELATIVE                             > 5.00  <     (meters)

   Modelling:
      CONTINUITY BETWEEN NEQS              > YES <       (YES, NO)
      NUMBER OF PARAMETERS PER DAY         > 000 <       (0: AS IN NEQ)
```

**Figure 18.5:** The option input ( Panel 4.8.1–2.2 ) of ADDNEQ to modify the parameterization and the a priori constraints of the troposphere.

### 18.7.7 Output Description

The ADDNEQ output file consists of the following parts:

`LIST OF INPUT AND OUTPUT FILENAMES`
A list of the input files used and the stored output files (information given in the N-file).

`LIST OF NORMAL EQUATION FILES`
A list of the normal equation files used (information given in the F-file) including basic information like the number of parameters contained in the normal equations, title, and the re-scaling value (may be specified using a `.WGT` file according to Section 24.8.14).

`LIST OF STATIONS`
The list contains station number, station name (see Section 24.8.1), a flag if velocities are estimated (not in the example below), the total number of coordinate "observations", and a table showing in which files each site was observed. The flags used in the table are explained at the bottom of the list.

```
TOTAL NUMBER OF STATIONS:    38
NUM  STATION       VELO R #FIL 12345
-----------------------------------------------------------------------------
186  CAGL 12725M003          2 XX
155  MATE 12734M008B         3 XX  X
122  MASP 31303M002          2 XX
176  VILL 13406M001          2 XX
154  MADR 13407S012          2 NN
158  ZIMM 14001M004          4 NNNN
...  .... .........          . .....
FLAGS:    W: WEIGHTS , F: FIXED, N: FREE NETWORK RESTRICTIONS, X: FREE ESTIM.
```

`INPUT OPTIONS`
A summary of important input options is included:

- a priori sigmas for coordinates and velocities,
- free network constraints,
- orbit model information,
- troposphere information.

`STATISTICS FOR ESTIMATED PARAMETERS`
The statistics of estimated parameters may look as follows (extracted from a solution covering 3 years of continuous global GPS data processing):

```
-----------------------------------------------------------------------------------
STATISTIC OF SOLVED FOR PARAMETERS     #PARAMETERS   #PRE-ELIMINATED        #NO-OBS
-----------------------------------------------------------------------------------


STATION COORDINATES                        309         36  (BEFORE INV)        0
SATELLITE ANTENNA OFFSETS                    6          0                     47
CENTER OF MASS                               3          0                      0
STATION VELOCITIES                         285          0                      0
-----------------------------------------------------------------------------------
NUMBER OF SOLVE FOR PARAMETERS             603         36                     47
-----------------------------------------------------------------------------------
```

For each parameter type the following statistical information is included: number of unknown parameters, number of pre-eliminated parameters (pre-elimination of particular sites from particular solutions may be achieved according to Section 24.4.13), and number of parameters with no observations (e.g., parameters that were set up originally with a priori sigmas, but did not figure in any observation equation).

`SHORT SOLUTION STATISTICS`

For the same example the solution statistics has the form:

```
 TOTAL NUMBER OF PARAMETERS                    :    3148110
 TOTAL NUMBER OF OBSERVATIONS                  :  101705205
 NUMBER OF SINGLE DIFF. FILES                  :      83491
-----------------------------------------------

 A POSTERIORI SIGMA OF UNIT WEIGHT  :    0.0027
 SIGMA OF COORDINATE GROUP          :    0.0270
 (ESTIMATION OF A MORE REALISTIC SCALING
  FACTOR OF THE COVARIANCE MATRIX)
```

The solution summary contains information concerning the total number of parameters (including all unknowns which may have been pre-eliminated in previous runs), the number of observations, and the total number of processed single difference files. The value `A POSTERIORI SIGMA OF UNIT WEIGHT` ($\hat{\sigma}^2$ in Eqn. (18.6), same meaning as in program output of GPSEST) is an important indicator for the quality of the used receivers and for the quality of the solution (see Chapter 4). This value is furthermore used as the scaling factor for the estimated covariance matrix.

The value `SIGMA OF COORDINATE GROUP` is computed if only coordinates are involved in the combination. The value is derived as a group rms from weighted coordinate repeatabilities [*Brockmann*, 1996] and is better suited (gives a more realistic covariance matrix) than the estimated `A POSTERIORI SIGMA OF UNIT WEIGHT`. You may, e.g., multiply the formal rms values of the coordinates (see below) with the ratio `SIGMA OF COORDINATE GROUP/A POSTERIORI SIGMA OF UNIT WEIGHT` to get more realistic rms values, or, you may derive an approximate value from the (unweighted) repeatability summary (block `COMPARISON OF STATION COORDINATES...`), or, as a third possibility, use the (weighted) rms values specified in block `MEAN VALUES OF GEOCENTRIC X,Y,Z - COORDINATES`.

In the case of free network solutions this coordinate group rms value may be too pessimistic, because the coordinate repeatabilities are computed without allowing for Helmert parameters. In that case you should use the second possibility to derive more realistic rms values for your coordinates.

`RESULTS OF COMBINED SOLUTION FOR STATION COORDINATES`

In this block a list of coordinate estimates is given in the following way (identical to GPSEST):

```
TOTAL NUMBER OF STATIONS:   103
NUM  STATION NAME    PARAMETER    A PRIORI VALUE       NEW VALUE   NEW-A PRIORI RMS ERROR
-----------------------------------------------------------------------------------------

413  QUIN 40433M004  X           -2517230.9430      -2517230.9590    -0.0160    0.0001
                     Y           -4198595.2572      -4198595.1676     0.0896    0.0001
                     Z            4076531.3765       4076531.2646    -0.1119    0.0001

                     HEIGHT           1105.8994          1105.7750    -0.1244    0.0001
                     LATITUDE     39 58 28.398327    39 58 28.396976  -0.0417    0.0001
                     LONGITUDE  -120 56 39.929157  -120 56 39.931674  -0.0596    0.0001
...  .............
```

The example needs no further explanation.

In columns 100 to 132, in the lines corresponding to `HEIGHT`, `LATITUDE`, and `LONGITUDE` you find information concerning the error ellipsoids including some projections for each site.

```
RMS ERROR   3-D ELLIPSOID       2-D ELLIPSE
--------------------------------------------
0.0013      0.0013    0.3                           (a)
0.0002      0.0001   86.6      0.0001   86.7        (b)
0.0001      0.0002   -0.1      0.0002               (c)
 (1)         (2)      (3)       (4)      (5)
```

6 values (3 main axis parameters and 3 angles) characterize the three-dimensional ellipsoid and 3 values (2 main axis parameters and 1 angle) describe the two-dimensional error ellipse.

The lines are marked with (a), (b), and (c), the columns with (1) - (5). These marks are not included in the output file. The values (not extracted from the previous coordinate example) are defined as follows:

| | |
|---|---|
| Column (1) : | contains the rms errors in height (a), latitude (b), and longitude (c). These values are the intersections of the three-dimensional error ellipsoid with the coordinate axes of the local geodetic coordinate system. |
| Column (2) : | contains the lengths in meters of the principal axes of the three-dimensional error ellipsoid. |
| Column (3) : | contains in the first line (a) the zenith distance in degrees ($^o$) of the longest axis (2a), in the second line (b) the azimuth, counted positive in the direction to east, of the axis (2b) and in the third line (c) the elevation angle of the same axis (2b). |
| Column (4) : | contains the lengths in meters of the principal axes of the two-dimensional error ellipse in the horizontal plane. |
| Column (5) : | contains the azimuth in degrees ($^o$) of the principal axis (4b), counted positive in the direction to east, in the horizontal plane. |

This information is useful to produce plots of error ellipses as shown in Figure 18.6. The big error ellipse for the site `SFER` is a consequence of the small number of observations. The shape of the error ellipses is typical for ambiguity-resolved regional solutions. Regional solutions without ambiguity resolution usually show larger rms values in the east-west direction by a factor of about 2.

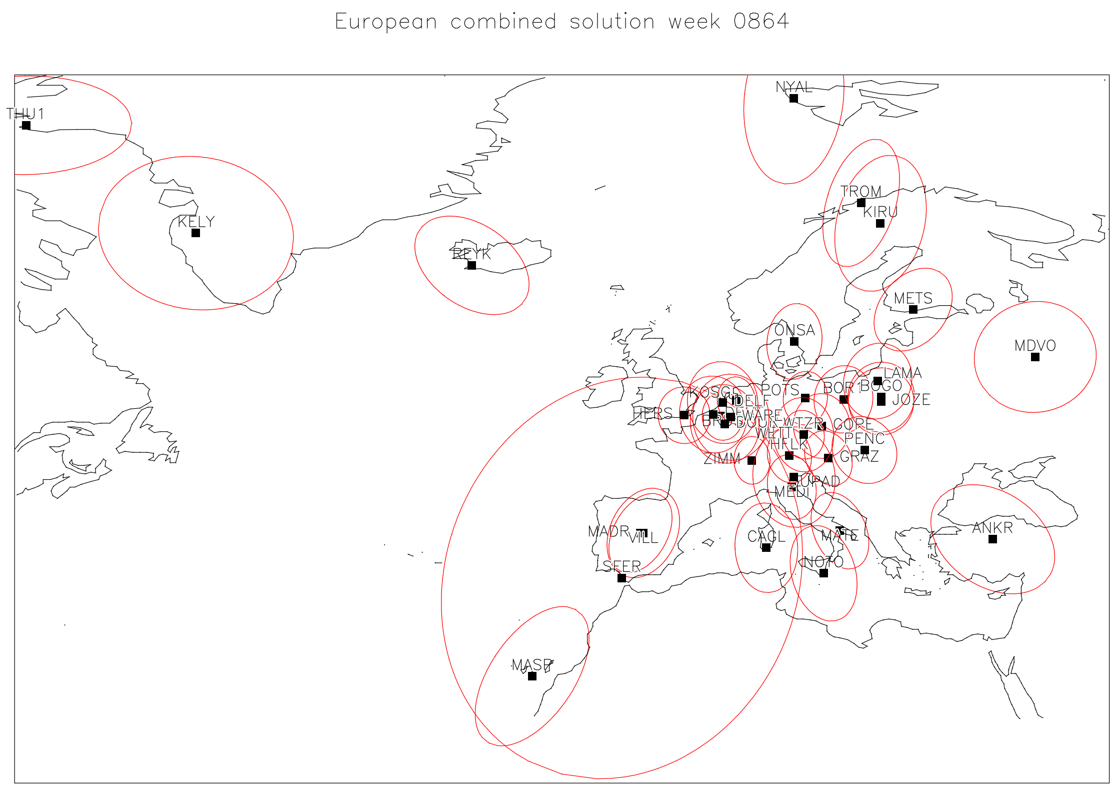


**Figure 18.6:** Error ellipses using the values of the GPSEST or ADDNEQ program output.

`Results for other parameter types`

The results for the other parameter types (e.g. troposphere parameters, orbits, etc.) are not explained in more detail here. The information given in the program output should be sufficient.

`MEAN VALUES OF GEOCENTRIC X,Y,Z - COORDINATES`

In this block we summarize the coordinate estimates in the geocentric coordinate system together with the associated rms information. The formal rms of the previous block is repeated (`RMS1`). The formal rms error of the station position $\texttt{RMS1-XYZ} = \sqrt{\texttt{RMS1}_X^2 + \texttt{RMS1}_Y^2 + \texttt{RMS1}_Z^2}$ is given in the last column. The second rms value `RMS2` is derived from weighted coordinate repeatabilities, similar to the computation of the group rms in the output block `SHORT SOLUTION STATISTICS`. In this case we consider one coordinate component as a group.

The `EPOCH` of the coordinates listed here and also in the resulting coordinate file (see Section 24.8.1) is usually the middle of the observation time span. Only in one case we refer the resulting coordinates to a different epoch: if you specify a special coordinate fixing file (see Section 18.7.4) without specifying a special velocity fixing file to propagate the coordinates to the middle observation epoch, we retain the epoch of the special coordinate fixing file as reference epoch for the resulting coordinates.

```
------------------------------------------------------------------------------------------
MEAN VALUES OF GEOCENTRIC X,Y,Z - COORDINATES
RMS1: FORMAL ACCURACY OF EACH COORDINATE COMPONENT FROM COMBINED SOLUTION
RMS2: RMS OF WEIGHTED AVERAGE OF EACH COORDINATE COMPONENT
------------------------------------------------------------------------------------------
EPOCH: 1996-07-24 11:59:45
VELOCITY MODEL INTRODUCED TO INDIVIDUAL SOLUTIONS:  ZERO VELOCITY FIELD


NUM  STATION      #FIL FLG     X (M)   RMS1  RMS2        Y (M)     RMS1  RMS2  ...RMS1-XYZ
------------------------------------------------------------------------------------------

186  CAGL 12725M003  2 M 4893378.9431 0.0008 0.0015   772649.6258 0.0002 0.0004 ... 0.0011
155  MATE 12734M008B 3 M 4641949.7189 0.0007 0.0016  1393045.2699 0.0002 0.0023 ... 0.0010
122  MASP 31303M002  2 M 5439192.2475 0.0011 0.0031 -1522055.6314 0.0003 0.0038 ... 0.0013
... ..... .........    .  .
```

`LIST OF RMS VALUES`
: Each NEQ system (corresponding to one NEQ file) is resolved individually applying the new specified input options. The list of rms values is useful to have a good overview about the impact of each contributing solution.

```
FILE  FILE NAME                          RMS     #OBS #PAR(ORIG.) #PAR (SOLVED)  MJD
-----------------------------------------------------------------------------------------

  1   M:/COMB_GLO/OUT/EUR08577.NEQ     0.0038   838976     10226      114    50246.49983
  2   M:/COMB_GLO/OUT/EUR08587.NEQ     0.0024   870151      9936      114    50253.49983
  3   M:/COMB_GLO/OUT/EUR08597.NEQ     0.0035   899299      9733      111    50260.49983
  4   M:/COMB_GLO/OUT/EUR08607.NEQ     0.0036   884395      9959      114    50267.49983
  5   M:/COMB_GLO/OUT/EUR08617.NEQ     0.0036   768605     12046      114    50274.49983
  6   M:/COMB_GLO/OUT/EUR08627.NEQ     0.0034   838939     11874      114    50281.49983
  7   M:/COMB_GLO/OUT/EUR08637.NEQ     0.0036   833443     10256      114    50288.49983
  8   M:/COMB_GLO/OUT/EUR08647.NEQ     0.0037   915804     12007      114    50295.49983
-----------------------------------------------------------------------------------------
                                       0.0037  6849612     85251      123    50256.99983
```

`FINAL LIST OF COVFACTOR VALUES`
: This list shows the rescaling values. Values $\neq 1.0$ are possible by specifying a `.WGT` file (see Section 24.8.14).

`COMPARISON OF STATION COORDINATES...`
: In this block we compare the results of the *site coordinates* for each individual solution with the combined solution. The *coordinate repeatabilities,* in units of mm, are given in the local north-east-up coordinate system. The intention is to detect outliers (see also block `OUTLIER DETECTION`). In the column `RMS` we compute an unweighted rms for the estimation of a single coordinate component (for more information see the explanations concerning block `SHORT SOLUTION STATISTICS`).
Specifying a `.PLT` file (see Section 24.8.11) saves the same information into a file in a different format, better suited as input to conventional plot tools.
We mentioned already in Section 18.3.4 that in the case of free network solutions the residuals are obtained after a Helmert transformation. In the case of long time series including

velocity estimation you may wish to remove the effect caused by the estimated velocity. Edit the I-file, search for the string `REPET.PLT` and insert there a “2” instead of “0”. The default plot option is “0” to retain the linear development of the coordinate components in the residuals.

```
 ----------------------------------------------------------------------------------
 COMPARISON OF STATION COORDINATES WITH RESPECT TO THE COMBINED SOLUTION IN MM
 - UNWEIGHTED RMS OF INDIVIDUAL COORDINATE RESIDUALS
 ----------------------------------------------------------------------------------

 TOTAL NUMBER OF STATIONS:    41
 ----------------------------------------------------------------------------------

 NUM  STATION         #FIL C   RMS     1     2     3     4     5     6     7     8
 ----------------------------------------------------------------------------------

 186  CAGL 12725M003    8  N    1.2   -1.0  -0.7   0.7   0.1   0.3   2.4  -0.5   1.6
                           E    1.6   -0.7   1.2  -2.7  -0.3  -0.7   0.9   1.7   2.2
                           U    4.6   -1.7  -4.8   0.3  -5.8  -4.4   6.7   4.2   2.4

 158  ZIMM 14001M004    8  N    1.7    3.3   0.0  -0.3  -1.6  -0.6   1.7   0.8  -1.7
                           E    1.7   -2.7  -0.9  -0.6   2.5   1.3  -1.6   0.0  -0.5
                           U    3.9    5.6   6.2   1.5   0.3  -3.9  -3.7   0.0  -1.6
 ...  .............
```

`UNWEIGHTED RMS VALUES WITH RESPECT TO THE COMBINED SOLUTION`

This block contains a summary of the *repeatabilities for each file*. This information is well-suited to find out whether particular normal equations contain problems. Because unweighted rms values are computed it may occur, that outliers for a single site for a particular solution (e.g., due to a small number of observations), degrade the summary values for that specific solution.

`FAC` is the ratio of the repeatability and the formal rms value. The information contained in this line may also be saved to a `.WGT` file using Panel 4.8.1–0, option `COVARIANCE COMPON`. Furthermore, we use these values for the outlier detection described below.

```
 TOTAL NUMBER OF STATIONS:    41
 ----------------------------------------------------------------------------------

                      #FIL C   RMS     1     2     3     4     5     6     7     8
 ----------------------------------------------------------------------------------

      COMBINATION       8  N    1.7    1.7   2.0   1.5   1.7   2.0   1.3   1.4   2.2
                           E    1.7    2.2   1.8   1.5   1.6   1.7   1.1   1.7   2.0
                           U    4.7    5.3   5.2   3.1   5.3   4.2   4.8   5.6   4.4
                         #STA    41     38    38    37    38    38    38    38    38
                          FAC         4.54  7.67  5.48  5.00  4.71  5.23  4.82  4.60
```

`WEIGHTED RMS VALUES WITH RESPECT TO THE COMBINED SOLUTION`

This block contains the same information as in the previous block, but now the rms values are computed using the formal rms values for each coordinate component estimation. Outliers due to a small number of observations do not play a major role, here. This information is available for free network solutions, only.

`OUTLIER DETECTION`

The residuals in the block `COMPARISON OF STATION COORDINATES...` are analyzed for outliers using the information about the individual formal rms values. The "detection level" for the components `NORTH`, `EAST`, and `UP` is 3 times of the mean formal rms of all contributing files (the column `RMS` in block `UNWEIGHTED RMS VALUES WITH RESPECT...`). Outliers of several meters may disturb outlier detection. You may pre-eliminate sites of particular solutions using a station problem file (see Section 24.4.13).

```
----------------------------------------------------------------------------
OUTLIER DETECTION USING THE MEAN REPEATABILITY RMS OF EACH COMPONENT
DETECTION LEVEL (RESIDUUM/RMS): 3.00
NORTH:  0.006, EAST:  0.006, UP:  0.018 (M)
----------------------------------------------------------------------------
FILE  STATION           COMPONENT    RESIDUUM(M)     RMS(M) RMS*FAC(M)   GRP

   1  MATE 12734M008        U           -0.0205     0.0011     0.0062
   4  SFER 13402M004        N           -0.0056     0.0002     0.0013
   5  MATE 12734M008B       N           -0.0091     0.0002     0.0011
   5  MATE 12734M008B       E           -0.0068     0.0001     0.0008
   8  ANKR 20805M002        N           -0.0074     0.0002     0.0012
```

`UNWEIGHTED RMS VALUES OF THE COMPARISON BETWEEN THE SOLUTIONS`

Similar comparisons as those in block `UNWEIGHTED RMS VALUES WITH RESPECT...` are also performed between individual solutions (instead of comparing each solution with the combination). To reduce the amount of output we give this information only if less than 16 normal equation files are processed.

```
TOTAL NUMBER OF STATIONS:    41
------------------------------------------------------------------------------

                      FIL C          1     2     3     4     5     6     7
------------------------------------------------------------------------------

                       2  N          1.9
                          E          1.7
                          U          6.1
                       #STA           38

                       3  N          2.0   2.1
                          E          2.4   2.0
                          U          6.8   6.1
                       #STA           37    37

                       4  N          2.8   2.5   2.0
                          E          3.2   2.5   2.3
                          U          8.2   6.1   6.1
                       #STA           37    37    37
                       ....         ....  ....  ....  ....
```

`COMPARISON OF BASELINE LENGTHS`

If you defined baselines using a (`.BSL`) file (see Section 24.8.29) in ADDNEQ you also get baseline statistics (residuals in latitude, longitude, height, and length) for those baselines. The relative error (expressed in `PPM` = parts per million) is computed as the ratio of the rms of the quantity considered (in column `D(LGT)`) and the baseline length. The `RMS` values are

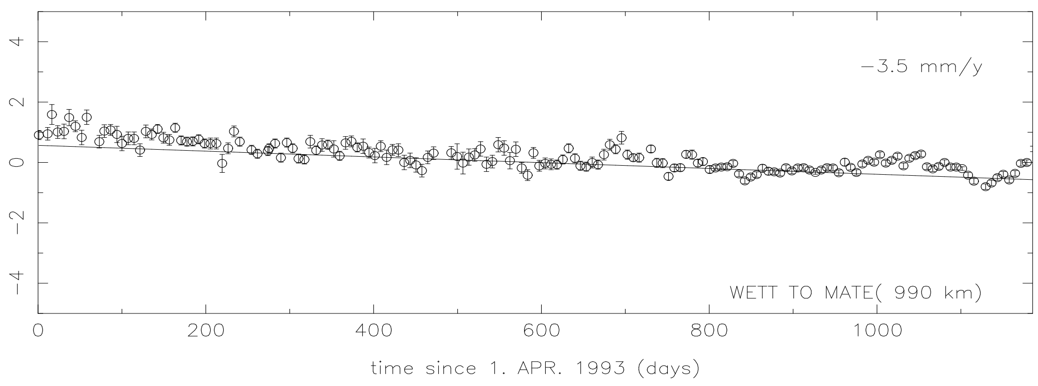


**Figure 18.7:** Plot of the baseline length residuals and the associated rms errors (in cm) derived from the ADDNEQ (or COMPAR) output.

the formal rms values derived from the individual solutions. An example for the development of the baseline length and the associated formal rms errors (column (`RMS`)) is given in Figure 18.7.7. The velocity value given in the first line refers to the baseline length.

```
BASELINE : WETT 14201M009    TO  MATE 12734M008   VELOCITIES:-0.0035 +- 0.00003
-------------------------------------------------------------------------------
     FILE   BASE.LENGTH    D(LAT)    D(LON)    D(HGT)    D(LGT)    (PPM)  (RMS)
       1    989988.2166   -0.0051   -0.0167    0.0114    0.0091   0.0092 0.0005
       2    989988.2171   -0.0106   -0.0055    0.0031    0.0095   0.0096 0.0007
       3    989988.2235   -0.0154   -0.0026   -0.0283    0.0159   0.0161 0.0011
       4    989988.2176   -0.0076   -0.0062    0.0184    0.0101   0.0102 0.0007
    ...     ...........   .......   .......   .......   .......  ....... ......
     164    989988.2039    0.0021    0.0045   -0.0533   -0.0036  -0.0037 0.0003
     165    989988.2073   -0.0012    0.0035   -0.0410   -0.0003  -0.0003 0.0003
     166    989988.2075    0.0002    0.0025   -0.0239    0.0000   0.0000 0.0003
     167    989988.2120   -0.0060    0.0026   -0.0228    0.0045   0.0045 0.0003
-------------------------------------------------------------------------------
     164    989988.2076    0.0048    0.0054    0.0163    0.0051   0.0052 0.0005
```

## 18.8 Handling Parameters and NEQ Files in Programs GPSEST and ADDNEQ

Let us address in this section a few important aspects concerning the parameters and the NEQ files as they occur in programs GPSEST and ADDNEQ.

| | |
|---|---|
| GPSEST | Do **not** use fix sites when saving normal equations, because GPSEST does not set up the coordinate parameters for fixed sites. Constrain the stations (using, e.g., an a priori sigma of 0.0001 m) instead. |
| GPSEST | Do not resolve ambiguities and create the NEQ file in the same program run. Solve ambiguities first, then introduce the resolved ambiguities in the second run and generate the NEQ file. |
| GPSEST | Select pre-elimination of type `AI` for ambiguity parameters when saving the NEQs. |
| GPSEST | Select pre-elimination of types `BI` or `AI` for troposphere, if you do not want to change the troposphere parameter handling later in ADDNEQ, select `NO` if you want to include them in the NEQs. |
| ADDNEQ | In Panel 4.8.1–2, option `SITE-SPECIFIC TROPOSPHERE`, say `YES` to allow the specification of a priori sigmas (if you are not sure whether there are troposphere parameters in the NEQs). |
| ADDNEQ | Fixing sites is allowed (as opposed to GPSEST), because fixing is equivalent to constraining coordinates to an a priori sigma of 0.01 mm in ADDNEQ. |
| ADDNEQ | The generation of a SINEX file is not allowed (NEQ file is allowed), if free network solutions are generated because of a singular a priori covariance matrix. The use of the inverted a priori covariance matrix would solve the problem, but this information is not supported by most of the SINEX reading routines. |
| ADDNEQ | Saving a NEQ file is not possible (SINEX file is possible), if you specify a special coordinate file for fixed sites in Panel 4.8.1. This limitation is not important, because you may create normal equations containing all information without this special file. |
| ADDNEQ | Saving a NEQ file is not possible (SINEX file is possible) if velocities are solved for. ADDNEQ supports the estimation of velocities, but not the handling and combination of estimated coordinates **and** velocities. |

# 19. Stacking of Normal Equations Using the New ADDNEQ2 Program

The program ADDNEQ2 was developed during the years 1997 and 1998 as a replacement of the older program ADDNEQ. The new program has not only the whole functionality of its predecessor but it adds many new options, too. ADDNEQ2 is the first Bernese program written in Fortran 90. It uses features of the new language standard like dynamical memory allocation, structures, modules etc. The main idea behind the development of ADDNEQ2 was the common way of handling different parameter types. This approach is described in the following section.

In the *Bernese GPS Software* Version 4.2, the program ADDNEQ2 is delivered in an experimental stage. It will, however, replace ADDNEQ in the next release of the *Bernese GPS Software*. Users interested in long time series are advised to archive their normal equations in both, `NEQ` and `NQ0`, formats (GPSEST generates both file types by default).

## 19.1 Time-Dependent Parameters

Despite the fact that only data from so-called static GPS applications are processed by the *Bernese GPS Software*, many parameters are clearly time-dependent. A typical example is the tropospheric delay. Even such "stable" parameters as the coordinates of static receivers in the terrestrial reference frame become time-dependent, if data stemming from long periods are processed. Those parameters have to be modeled as a function of time at some point in the processing procedure.

### 19.1.1 Piece-Wise Constant Function

The simplest way to introduce a time-dependency for a particular parameter consists of defining a piece-wise constant function according to the Figure 19.1.

In the *Bernese GPS Software* piece-wise constant functions are used for the modeling of station-specific tropospheric delay. The time information corresponding to this type of parameters consists of the center of the validity interval and the interval length.

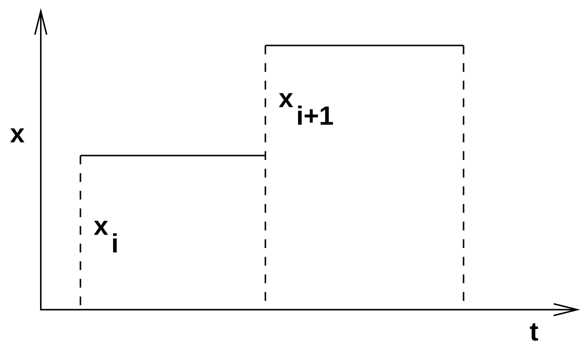


**Figure 19.1:** Piece-wise constant function.

### 19.1.2 Piece-Wise Linear Function

Using a piece-wise constant function is simple but also inconvenient, in many cases "unphysical" because of the discontinuities at the interval boundaries. Actually, this kind of parameterization has only "historical" reasons and we plan to replace it in all cases where continuity is expected from the physical point of view.

A better but still rather simple solution is the *linear or piece-wise linear function*. In this case each parameter $x$ in each particular time interval $\langle t_1,\ t_2 \rangle$ is modeled according to one of the following figures:

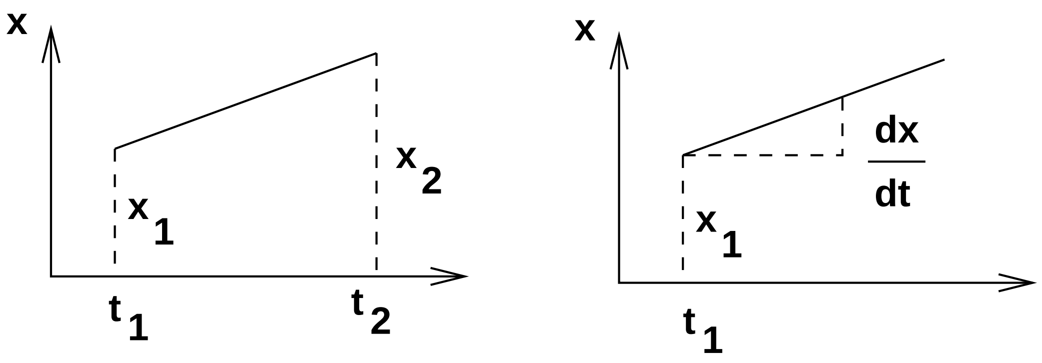


**Figure 19.2:** Modeling of time-dependent parameters by $x_1, x_2$ resp. $x_1, \dot{x}_1$.

Both types of parameterizations

$$\tilde{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ \dfrac{\mathrm{d}\,x}{\mathrm{d}\,t} \end{pmatrix}$$

are equivalent. For the transition between these two parameterizations we may use the following transformation equation

$$\mathbf{x} = \begin{pmatrix} 1 & 0 \\ \dfrac{-1}{t_2 - t_1} & \dfrac{1}{t_2 - t_1} \end{pmatrix} \widetilde{\mathbf{x}} = \mathbf{C}\,\widetilde{\mathbf{x}}\,, \tag{19.1}$$

and according to this the new system of normal equations reads as

$$\mathbf{C}^T\,\mathbf{N}\,\mathbf{C}\,\widetilde{\mathbf{x}} = \mathbf{C}^T\,\mathbf{b}\,. \tag{19.2}$$

Although both approaches are equivalent from the mathematical point of view, we prefer to use the parameters

$$\widetilde{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\,.$$

Consequent use of this kind of parameterization is made in the new program ADDNEQ2. With this definition there is no need to distinguish between “offsets” and “drifts” (only offsets are necessary). Furthermore, we have the advantage of a very simple form of the condition for the continuity at the interval boundaries (see Section 19.2.8).

In ADDNEQ2 the time validity of each parameter is characterized by two values: the epoch $t_i$ that is the center of the validity interval, and the length of the validity interval. If the piece-wise linear modeling is used, this length is set to zero and the actual value of the parameterized quantity in epoch $t \neq t_i$ is computed using the linear interpolation between two parameters. A non-zero interval length means that the piece-wise constant function is used.

## 19.2 Parameter Manipulations

The following operations may be performed with parameters in the new program ADDNEQ2:

(1) Changing the auxiliary parameter information.
(2) Rescaling the normal equation systems.
(3) A priori transformation of the coordinates to a different reference frame.
(4) Changing the a priori values.
(5) Changing the validity interval.
(6) Parameter elimination.
(7) Parameter stacking.
(8) Constraining of the parameters.
(9) Changing the parameterization of the time-dependent quantities.
(10) Expansion of the normal equation system.

These basic operations represent the functionality of the ADDNEQ2 program. Every actual operation performed by ADDNEQ2 may be expressed as a combination of basic operations listed above. Let us now discuss the individual operations:

### 19.2.1 Changing the Auxiliary Parameter Information

This is a very simple operation which does not have any influence on the system of normal equations. It includes renaming of stations and changing receiver and antenna names. This manipulation is done by the subroutine RDSTACRX immediately after reading the NEQ system from a file.

### 19.2.2 Rescaling the Normal Equation Matrices

Rescaling of normal equations is important, e.g., if two or more NEQ systems have to be combined, where each of them stems from different processing software or strategies. A typical example is given by the use of different sampling rates for GPS observations. Due to the time-correlations of GPS observations, the results remain (almost) the same, but the variance-covariance matrices change if different sampling rates are used.

The problem is solved by the following transformation: assuming the original NEQ system

$$\mathbf{N}\,\mathbf{x} = \mathbf{b}\;, \tag{19.3}$$

the new system reads as

$$\kappa\,\mathbf{N}\,\mathbf{x} = \kappa\,\mathbf{b}\;, \tag{19.4}$$

where $\kappa$ is the a priori scale factor. Its statistical meaning is the ratio of the variances:

$$\kappa = \frac{\sigma^2_{old}}{\sigma^2_{new}}\;. \tag{19.5}$$

In the program ADDNEQ2 the rescaling is performed by the subroutine APRHELM if the a priori information on the scale factor is provided.

### 19.2.3 A priori Transformation of Coordinates Into a Different Reference Frame.

A parameter transformation must be used for the transition between two reference frames. Let us assume that we want to combine normal equations stemming from the solution with fixed coordinates in, e.g., ITRF93 (denoted by index “93”) with a second set of normal equations where the ITRF94 (denoted further by index “94”) was used. Both reference frames are related by the 7-parameter transformation

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix}_{94} = \underbrace{(1+\mu)\begin{pmatrix} 1 & \gamma & -\beta \\ -\gamma & 1 & \alpha \\ \beta & -\alpha & 1 \end{pmatrix}}_{\mathbf{R}} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix}_{93} + \underbrace{\begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{pmatrix}}_{\mathbf{\Delta}}\;. \tag{19.6}$$

Introducing the a priori values for the coordinates (in matrix notation)

$$\mathbf{X}_{94} = \mathbf{X}^0_{94} + \tilde{\mathbf{x}} \tag{19.7a}$$

$$\mathbf{X}_{93} = \mathbf{X}^0_{93} + \mathbf{x}\;, \tag{19.7b}$$

we may write

$$\mathbf{X}^0_{94} + \tilde{\mathbf{x}} = \mathbf{R}\,(\mathbf{X}^0_{93} + \mathbf{x}) + \mathbf{\Delta}\;. \tag{19.8}$$

Assuming small rotation angles $\alpha, \beta, \gamma$, we may simplify the last equation by setting

$$\mathbf{R}\,\mathbf{x} \doteq \mathbf{x}\,. \tag{19.9}$$

Thus the relation between old and new parameters is of the form

$$\mathbf{x} = \tilde{\mathbf{x}} + (\mathbf{X}^0_{94} - \mathbf{R}\,\mathbf{X}^0_{93} - \boldsymbol{\Delta})\,. \tag{19.10}$$

The transition between two different a priori reference frames is thus nothing but a change of the a priori values of the station coordinates. In ADDNEQ2 the subroutine APRHELM is responsible for the transformation of the a priori coordinates. Then the NEQ system is transformed by the subroutine APRTRANS as described in the following section.

### 19.2.4 Changing the A Priori Values

Changing the a priori values is based on the following relation between the old and new parameters:

$$\mathbf{x} = \tilde{\mathbf{x}} + (\mathbf{X}^0_{new} - \mathbf{X}^0_{old})\,. \tag{19.11}$$

A priori values do not have any influence on the normal equation matrix **N**, but they change the vector **b** on the right-hand side of the NEQ system:

$$\tilde{\mathbf{b}} = \mathbf{b} - \mathbf{N}(\mathbf{X}^0_{new} - \mathbf{X}^0_{old})\,. \tag{19.12}$$

The transformation of a priori values is performed by the subroutine APRTRANS immediately after reading the NEQ system. A priori values are changed, if different values are provided as input (this is, e.g., always true for the station coordinates). If the system contains some drift parameters (this is the case if the system stems from the older programs, e.g., from GPSEST), the a priori values of all drift parameters are automatically transformed to zero (there are the Earth orientation parameters as an example).

### 19.2.5 Changing the Validity Interval

Changing the validity interval usually precedes parameter stacking (see below). Two variants of this operation are provided. The first variant is used, if some quantity is modeled by the (piece-wise) constant function:

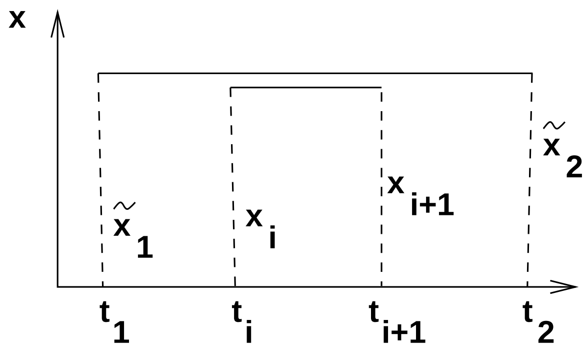


**Figure 19.3:** Changing the validity interval for the constant function.

Such an operation is trivial. It does not have any influence on the system of normal equations. It just changes the time interval information in the parameter description structure.

The situation is more complicated when a certain quantity is modeled by the (piece-wise) linear function:

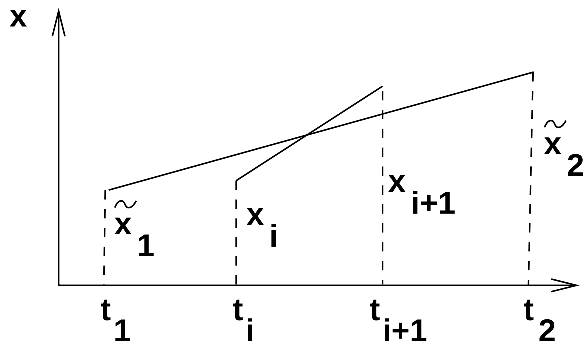


**Figure 19.4:** Changing the validity interval for the linear function.

The time-dependent value $x(t)$ may be expressed in two ways:

$$x(t) = x_i \frac{t_{i+1} - t}{t_{i+1} - t_i} + x_{i+1} \frac{t - t_i}{t_{i+1} - t_i} \tag{19.13}$$

$$x(t) = \tilde{x}_1 \frac{t_2 - t}{t_2 - t_1} + \tilde{x}_2 \frac{t - t_1}{t_2 - t_1} . \tag{19.14}$$

Comparing both expressions we obtain the transformation matrix for the interval $\langle t_i; t_{i+1} \rangle$:

$$\mathbf{x} = \begin{pmatrix} \frac{t_2 - t_i}{t_2 - t_1} & \frac{t_i - t_1}{t_2 - t_1} \\ \frac{t_2 - t_{i+1}}{t_2 - t_1} & \frac{t_{i+1} - t_1}{t_2 - t_1} \end{pmatrix} \tilde{\mathbf{x}} = \mathbf{C}\, \tilde{\mathbf{x}} . \tag{19.15}$$

This matrix may be extended for an arbitrary number of subintervals. Using Equation (19.15) the subroutine PARTRANS transforms the NEQ system according to Equation (19.2). Changing the validity interval has an influence on the normal equation matrix **N** and on vector **b** (right-hand side of NEQ system).

### 19.2.6 Parameter Elimination

Pre-elimination of parameters was described in Section 18.2.2. It is performed by the subroutine PARELIMI. One may decide whether the parameters are pre-eliminated before or after the stacking of normal equations. Before pre-elimination, the parameters may be either constrained or transformed.

### 19.2.7 Parameter Stacking

A simple, but basic operation of the ADDNEQ2 program is called the parameter stacking. If we want to stack the parameters $x_i$ and $x_{i+1}$ into one parameter $x_i$, we can use following parameter transformation:

$$\begin{pmatrix} \vdots \\ \mathbf{x}_i \\ \mathbf{x}_{i+1} \\ \mathbf{x}_{i+2} \\ \vdots \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & \cdots & 0 \\ 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}}_{\mathbf{C}} \begin{pmatrix} \vdots \\ \mathbf{x}_i \\ \mathbf{x}_{i+2} \\ \vdots \end{pmatrix} . \tag{19.16}$$

The parameter stacking is sometimes performed immediately after reading the normal equation system. But usually some type of parameter transformation precedes the stacking. A typical example is the reduction of the number of parameters according to the following figure:

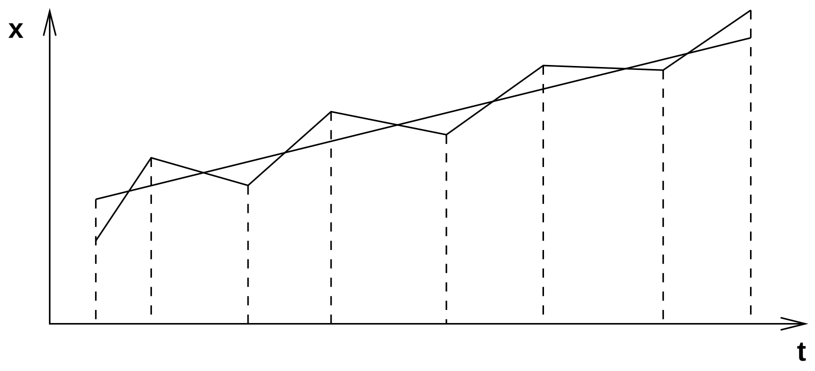


**Figure 19.5:** Reducing the number of parameters.

The operation is performed in two steps: first, the validity interval is changed according to Section 19.2.5 and then the stacking is performed.

### 19.2.8 Constraining of the Parameters

In the *Bernese GPS Software* the constraining of the parameters is performed by adding fictitious observations as constraints. The old ADDNEQ program used the constraining technique rather extensively for almost every parameter manipulation. Because of numerical problems in this approach, ADDNEQ2 is based on parameter transformation instead of constraining technique wherever possible. However, there are still quite a few situations when the constraining technique has to be used or when a transformation is too complicated. We will discuss several such cases in this section.

The constraining of the parameters in the *Bernese GPS Software* is done by introducing fictitious observations in the form

$$\mathbf{Hx} - \mathbf{h} = \mathbf{0} \, . \tag{19.17}$$

The corresponding weight matrix (of constraints) is denoted by $\mathbf{P}_h$ and it is a diagonal matrix of the form

$$\mathbf{P}_h = \text{diag}\left(\frac{\sigma_0^2}{\sigma_i^2}\right) \ , \tag{19.18}$$

where the values of $\sigma_i$ are specified by the user. Denoting the normal equation matrix of the original (unconstrained) system by **N** and the right-hand side vector of the original system by **b**, the new (constrained) system receives the form

$$(\mathbf{N} + \mathbf{H}^T\mathbf{P}_h\mathbf{H})\ \mathbf{x} = \mathbf{b} + \mathbf{H}^T\mathbf{P}_h\mathbf{h} \, . \tag{19.19}$$

#### 19.2.8.1 Constraining Parameters to Their A Priori Values

Any parameter may be constrained to its a priori value using the fictitious observation in the form

$$X_i - X_i^0 = 0 \quad \text{or, which is equivalent} \quad x_i = 0 \, . \tag{19.20}$$

Such a fictitious observation has the weight

$$p_i = \frac{\sigma_0^2}{\sigma_i^2} \ , \tag{19.21}$$

where both, $\sigma_0^2$ (a priori variance of unit weight) and $\sigma_i^2$, are input parameters of the program. This type of constraining is often used for station coordinates and it is “almost mandatory” for $UT1 - UTC$ estimates at the starting epoch (see Chapter 14).

#### 19.2.8.2 Constraining Ellipsoidal Coordinates

The situation is slightly more complicated when ellipsoidal coordinates are to be constrained. The reason for constraining ellipsoidal coordinates rather than rectangular coordinates might be, e.g., different accuracy of the a priori horizontal and vertical positions. The ellipsoidal coordinates may be fixed using the fictitious observation

$$\mathbf{H}\,\mathbf{x} = \mathbf{0} \, , \tag{19.22}$$

where **H** is the Jacobi matrix of the transformation between a rectangular and ellipsoidal coordinate system:

$$\mathbf{H} = \begin{pmatrix} \frac{\partial\varphi}{\partial X} & \frac{\partial\varphi}{\partial Y} & \frac{\partial\varphi}{\partial Z} \\ \frac{\partial\lambda}{\partial X} & \frac{\partial\lambda}{\partial Y} & \frac{\partial\lambda}{\partial Z} \\ \frac{\partial h_{ell}}{\partial X} & \frac{\partial h_{ell}}{\partial Y} & \frac{\partial h_{ell}}{\partial Z} \end{pmatrix} \, . \tag{19.23}$$

For the computation of the matrix **H** the spherical approximation is sufficient.

#### 19.2.8.3 Relative Constraints of Station Coordinates and Velocities

It is also possible to constrain two parameters with respect to each other using the fictitious observation

$$x_i - x_j = 0 \, . \tag{19.24}$$

Ellipsoidal coordinates of two stations may be constrained as well:

$$\mathbf{H}_i\,\mathbf{x}_i - \mathbf{H}_j\,\mathbf{x}_j = \mathbf{0}\;. \tag{19.25}$$

Velocities of two stations may be handled in this way, too. We have the possibility to constrain the horizontal or the vertical component only. We use the following fictitious observation:

$$\mathbf{H}_i\,(\mathbf{x}_i(t_2) - \mathbf{x}_i(t_1)) - \mathbf{H}_j\,(\mathbf{x}_j(t_2) - \mathbf{x}_j(t_1)) = \mathbf{0}\;. \tag{19.26}$$

In this latter equation $t_1$ and $t_2$ are the boundaries of the time interval. Because of small station velocities, the Jacobian matrices $\mathbf{H}_i$ and $\mathbf{H}_j$ are (almost) time independent.

Relative constraining of station velocities may be used, e.g., if the velocity of a large lithospheric plate (with many GPS stations) is estimated.

### 19.2.8.4 Free Network Conditions

GPS is in principle an interferometric technique. Therefore it is in general not possible to estimate the absolute position of all stations. Some of them (at least one) have to be kept fixed on their a priori positions or so-called free network conditions have to be introduced.

Free network conditions are based on the assumption that there are two reference frames:

(1) a priori reference frame, and
(2) the reference frame of the resulting coordinates.

Both reference frames are related to each other by the 7-parameter transformation:

$$\begin{pmatrix} \widetilde{X}_i \\ \widetilde{Y}_i \\ \widetilde{Z}_i \end{pmatrix} = (1+\mu)\begin{pmatrix} 1 & \gamma & -\beta \\ -\gamma & 1 & \alpha \\ \beta & -\alpha & 1 \end{pmatrix}\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{pmatrix}\;. \tag{19.27}$$

(The 7-parameter transformation may be written in this linearized form, because only small rotations $\alpha, \beta, \gamma$ are considered). The idea of free network conditions is based on the requirement that some of these seven parameters (computed using the Helmert method) are set equal to zero.

Equation (19.27) may be rewritten as

$$\begin{pmatrix} \widetilde{X}_i \\ \widetilde{Y}_i \\ \widetilde{Z}_i \end{pmatrix} = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 & -Z_i & Y_i & X_i \\ 0 & 1 & 0 & Z_i & 0 & -X_i & Y_i \\ 0 & 0 & 1 & -Y_i & X_i & 0 & Z_i \end{pmatrix}\begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \\ \alpha \\ \beta \\ \gamma \\ \mu \end{pmatrix}\;, \tag{19.28}$$

or, in vector notation:

$$\widetilde{\mathbf{X}}_i = \mathbf{X}_i + \mathbf{B}_i\,\boldsymbol{\xi}\;. \tag{19.29}$$

Let us introduce the vectors $\widetilde{\mathbf{X}}$, $\mathbf{X}$, and the matrix $\mathbf{B}$ by

$$\widetilde{\mathbf{X}} = \begin{pmatrix} \widetilde{\mathbf{X}}_1 \\ \widetilde{\mathbf{X}}_2 \\ \vdots \end{pmatrix} \quad , \quad \mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \end{pmatrix} \quad , \quad \mathbf{B} = \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \end{pmatrix} \quad .$$

If we want to compute the parameters, we can use the following "observation equation":

$$\mathbf{v} = \mathbf{B}\,\boldsymbol{\xi} - (\widetilde{\mathbf{X}} - \mathbf{X}) \,, \tag{19.30}$$

which results into the following system of normal equations:

$$\mathbf{B}^T\mathbf{B}\,\boldsymbol{\xi} = \mathbf{B}^T\,(\widetilde{\mathbf{X}} - \mathbf{X}) \,. \tag{19.31}$$

In our case the expression $\widetilde{\mathbf{X}} - \mathbf{X}$ is the difference between the estimated and the a priori value:

$$\widetilde{\mathbf{X}} - \mathbf{X} = \mathbf{x} \,. \tag{19.32}$$

The parameters of the 7-parameter transformation are given by

$$\boldsymbol{\xi} = (\mathbf{B}^T\mathbf{B})^{-1}\,\mathbf{B}^T\,\mathbf{x} \,. \tag{19.33}$$

The free network condition thus may be imposed by adding the following fictitious observation

$$\mathbf{H}\,\mathbf{x} = \mathbf{0} \,, \tag{19.34}$$

where

$$\mathbf{H} = (\mathbf{B}^T\mathbf{B})^{-1}\,\mathbf{B}^T \,. \tag{19.35}$$

Because of the regularity of the matrix $\mathbf{B}^T\mathbf{B}$ it is even possible to use

$$\mathbf{H} = \mathbf{B}^T \,. \tag{19.36}$$

It is not always necessary to constrain all seven parameters $\boldsymbol{\xi}$. The number of singularities (that have to be removed using the constraints) depends on the solution type. In global solutions, it is usually sufficient to constrain the rotation parameters only (using the corresponding part of $\mathbf{H}$). This is the minimal constraint for the GPS network.

#### 19.2.8.5 Continuity Condition (SINEX)

Usually we require the continuity on the interval boundaries:

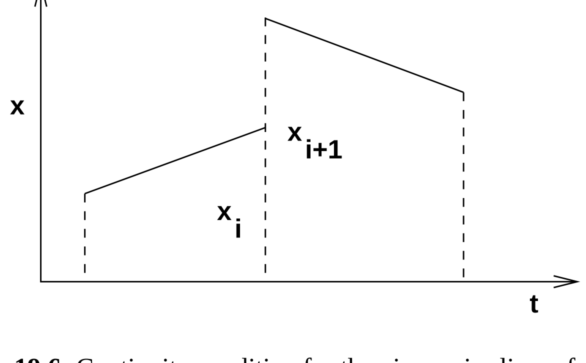


**Figure 19.6:** Continuity condition for the piece-wise linear function.

If we parameterize the piece-wise linear function by offsets only, this condition may be assured by the correct parameter transformation (stacking). However, for SINEX output we are compelled to use drifts (see Figure 19.2). Of course, it might be possible to formulate the corresponding condition for offset and drifts. We prefer a different approach:

(1) We use offsets only. With this parameterization the continuity is automatically assured.
(2) We expand our NEQ system according to Section 19.2.9. We thus obtain a singular system.
(3) We introduce the fictitious observations in the simple form

$$x_i - x_{i+1} = 0 \ . \tag{19.37}$$

These constraints remove the singularity from the system.

(4) We transform the (constrained) system according to Equation (19.1).

### 19.2.9 Expansion of the Normal Equation System

Adding new parameters to the system of normal equations leads to a so-called expansion of the normal equation system. The procedure may be described by the parameter transformation

$$\mathbf{x} = \mathbf{C}\, \widetilde{\mathbf{x}} \ , \tag{19.38}$$

where $\mathbf{x}$ is the original, $\widetilde{\mathbf{x}}$ the expanded vector of parameters and $\mathbf{C}$ is given by

$$\mathbf{C} = \begin{pmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \end{pmatrix} \ . \tag{19.39}$$

Obviously, after performing the transformation of the NEQ system according to the equation (19.2), the resulting normal equation matrix $\widetilde{\mathbf{N}}$ becomes singular:

$$\begin{pmatrix} \mathbf{N} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \widetilde{\mathbf{x}} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \ . \tag{19.40}$$

This operation is required at least in four cases:

(1) Estimation of station velocities.
(2) Addition of stochastic pulses in long-arc combination (see Chapter 8).
(3) Change of parameter description for SINEX.
(4) Helmert parameter estimation.

#### 19.2.9.1 Estimation of Station Velocities

Station coordinates in a terrestrial reference frame are very stable in time. The program GPSEST processes data stemming from time intervals not exceeding a few days (typically only one day). It is therefore reasonable and sufficient to model each station coordinate by a single value only. However, the analyses based on the combining of normal equations may cover time intervals of several months or even years. For such applications it is necessary to take station velocities into account. The problem is solved in the following way: after reading the NEQ file, the system is expanded (station coordinate parameters are added) according to a scheme illustrated in Figure 19.7.

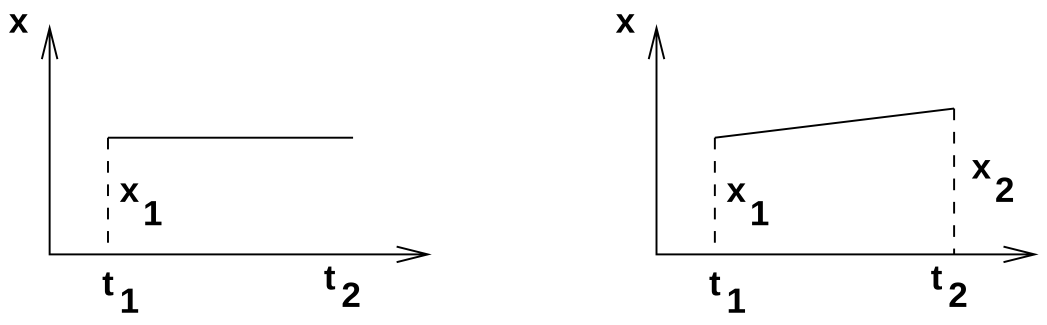


**Figure 19.7:** Addition of the new coordinate parameter.

The resulting system of normal equations is singular. But it is possible now to change the validity interval according to Section 19.2.5. The length of the new interval covers the entire analyzed period (e.g., one year). The resulting NEQ system may then be stacked together with the other systems (transformed in the same way — see Section 19.2.7). Stacking of many NEQ systems referring to different epochs removes the singularity.

### 19.2.9.2 Changing the Parameter Description for SINEX.

As mentioned, we consistently try to use only offset parameters for (piece-wise) linear functions. However, the SINEX standard (see Section 7.3) requires offsets and drifts for Earth rotation parameters.

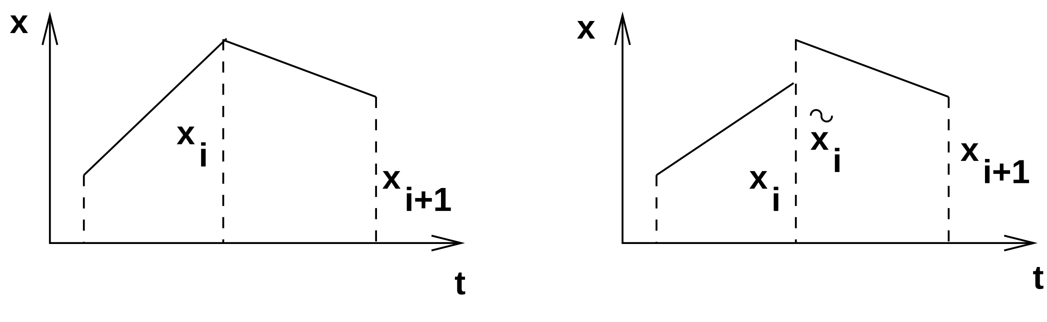


**Figure 19.8:** NEQ system expansion for SINEX.

In Section 19.1.2 we gave the formulae for the transformation between these two approaches. If we want to change the parameterization from offsets only to offsets and drifts, we have to expand the NEQ system according to Figure 19.8.

After the expansion the pairs of offsets are transformed into offsets and drifts. The resulting NEQ system is, of course, singular. The singularity is removed if continuity at the interval boundaries is imposed.

## 19.3 Input Options of the Program ADDNEQ2

In this section the input options of the program ADDNEQ2 are described briefly. It has to be said that only a limited set of options is available through the menu. The reason is, that in ADDNEQ2 the basic operations described in previous sections (e.g, constraining, binning etc.) can be applied specifically to each parameter (really parameter, not only parameter type).

The menu system provides a reasonable subset of available options. However, the experts may use all the options by editing the input `N`-file directly (ADDNEQ2 uses only the `N`-file and no `F-` or `I`-files).

In the Panel 4.8.3 the user specifies the names of input and output files. It should be mentioned, that the files from the `X:/GEN` directory have to be specified here, too. The menu program of ADDNEQ2 does not use the selection from Menu 0.3.1 . The station crux file specified here has to be in the new format (see Section 24.4.13). This format differs from the format used by the programs GPSEST and ADDNEQ.

```
 4.8.3  |     COMBINATION OF NORMAL EQUATION SYSTEMS (NEW ADDNEQ2 PROGRAM)

  CAMPAIGN                      >          <    (blank for selection list)

General Input File Names (including extensions, if any)

  GENERAL CONSTANTS             > CONST.       <  (blank:  sel.list)
  LOCAL GEODETIC DATUM          > DATUM.       <  (blank:  sel.list)
  STATION CRUX FILE             > STACRUX.NEW  <  (blank:  sel.list)
  SATELLITE PROBLEMS            > SAT_1996.CRX <  (blank:  sel.list)
  SATELLITE INFORMATION         > SATELLIT.TTT <  (blank:  sel.list)
  PHASE CENTER ECCENTRICITIES   > PHAS_IGS.01  <  (blank:  sel.list)
  POLE INFORMATION              > C04_1996.ERP <  (blank:  sel.list)
  SINEX HEADER FILE             > SINEX.       <  (blank:  sel.list)

Campaign-specific Input Files (names without extensions)

  A PRIORI COORDINATES          > ITRF0696 < (blank:  sel.list)
  A PRIORI VELOCITIES           > NO       < (NO: not used, blank: sel.list)
  A PRIORI COVARIANCES          > NO       < (NO: not used, blank: sel.list)

Output Files (names without extensions)

  NORMAL EQUATIONS              > NO       < (NO: not used, blank: sel.list)
  STATION COORDINATES           > TESTNEW  < (NO: not used, blank: sel.list)
  VELOCITIES                    > NO       < (NO: not used, blank: sel.list)
  SINEX                         > NO       < (NO: not used, blank: sel.list)
  TROPOSPHERE CORRECTIONS       > NO       < (NO: not used, blank: sel.list)
  TROPOS. SINEX OUTPUT          > NO       < (NO: not used, blank: sel.list)
  EARTH ROTATION PARAMETERS     > NO       < (NO: not used, blank: sel.list)
  IERS FORMAT ERP OUTPUT        > NO       < (NO: not used, blank: sel.list)
  INDV. SOL. PLOT FILE          > NO       < (NO: not used, blank: sel.list)
  PROGRAM OUTPUT FILE           > TESTNEW  < (blank: sel.list)

Auxiliary File Names (including extensions)

  ERROR MESSAGE FILE            > NO           <     (NO: sysout used)
  SCRATCH FILE                  > ADDNEQ2.SCR  <
```

The a priori coordinate file is mandatory. All station coordinates are transformed to these new a priori values at the very beginning of the ADDNEQ2 run. If the coordinate parameters are constrained, these new a priori values are used. The same is true for the station velocities. However, if you omit the velocity file, zero velocity will be assume per default.

```
4.8.3-1       ADDNEQ2: SELECTION OF NEQ INPUT FILES

Number of parameters in combined NEQ File
does not exceed > 1000 <   (500, 1000, 2000, 3000)

Input Normal Equation Files     > TEST     <

Orbital Elements                >          <
```

ADDNEQ2 allocates the memory dynamically, there are no fixed dimensions in the Fortran 90 code. However, it would be difficult for the program to figure out the necessary dimension of the resulting combined normal equation system (it would be necessary to go through all normal equation input files). Therefore the program requires the maximal number of parameters in the resulting normal equation system as an input parameter in Panel 4.8.3–1 . Specifying the value greater than necessary does not do any harm (assuming, of course, that your machine has enough memory).

In the second editable field of the Panel 4.8.3–1 the input normal equation files have to be specified. These files may stem from the previous runs of the ADDNEQ2 or from the GPSEST run. The program GPSEST stores the normal equations automatically in both, old (`*.NEQ`), and new (`*.NQ0`) formats. There is a program NEQ2NQ0 that transforms the old normal equation format into the new format. This program is, however, not accessible via the menu system.

In the third editable field of the Panel 4.8.3–1 the user has the opportunity to specify the output files containing the orbital elements (the result of the orbit estimation). If normal equation files of different days are combined (e.g. to compute long arcs), a different orbital element file has to be saved for each day. In this case, the wildcard characters (either % or ?) have to be used in the orbital elements file name. The logic is the same here as in the program ADDNEQ — see Chapter 18.

The names of standard orbit files and the radiation pressure files (that are required for orbit determination) are stored in the normal equation files. For user's convenience the file names are listed in the scratch panel that follows the panel Panel 4.8.3–1.

```
  NEQ-file        STD-file        RPR-file        ELE-file

> EB296165 <    >          <    >          <    >          <
```

Remember that the normal equation files, standard orbit files, and the radiation pressure files are input files. Orbital element files are output files.

```
4.8.3-2  ADDNEQ2: PARAMETER REQUESTS:

Input Number of Intervals for Each Parameter Type

               0 ... remains as it is (in input NEQ-file)
  greater than 0 ... (piece-wise) linear or (piece-wise) constant function

  Station Coordinates > 0  <
  Tropospheric Delay  > 0  <
  x-Pole Coordinate   > 0  <
  y-Pole Coordinate   > 0  <
  UT1-UTC             > 0  <
  Nutation Term Eps   > 0  <
  Nutation Term Psi   > 0  <
```

In the Panel 4.8.3–2 the modeling of time-dependent parameters is handled. Specifying a value $n > 0$ means that the entire time interval covered by the parameter type will be divided into $n$ subintervals. In each of these subintervals, the parameter will be modeled by a piece-wise constant (in case of tropospheric delay) or piece-wise linear function. For piece-wise linear functions the continuity at subinterval borders will be demanded (we think this is a reasonable behavior, however the experts may change it in `N`-file).

```
4.8.3-3 | ADDNEQ2: PARAMETER PRE-ELIMINATION

  0 ... do not pre-eliminate
  1 ... pre-eliminate before stacking
  2 ... pre-eliminate after stacking
  3 ... (trop. only) first and third file in 3-days solution

  Station Coordinates       > 0 <   (0, 1, 2)
  Orbital Parameters        > 0 <   (0, 1, 2)
  Tropospheric Delay        > 0 <   (0, 1, 2, 3)
  x-Pole Coordinate         > 0 <   (0, 1, 2)
  y-Pole Coordinate         > 0 <   (0, 1, 2)
  UT1-UTC                   > 0 <   (0, 1, 2)
  Nutation Term Eps         > 0 <   (0, 1, 2)
  Nutation Term Psi         > 0 <   (0, 1, 2)
  Stochastic Orbits         > 0 <   (0, 1, 2)
  Sat. Antenna Offsets      > 0 <   (0, 1, 2)
  Center of Mass            > 0 <   (0, 1, 2)
```

The Panel 4.8.3–3 defines which parameters will be pre-eliminated and when (see Section 18.2.2). Again, it is possible to refine the selection much more in the `N`-file.

```
4.8.3-4A | ADDNEQ2: CONSTRAINING OF PARAMETERS

STATION COORDINATES: station selection

> $FIRST       <  (blank, ALL, NONE, SPECIAL_FILE, $FIRST)

default sigma:    > 0.0001 <       selection file:  > IGSSIG   <

free network:     > NO  <       (YES, NO)
```

The Panel 4.8.3–4A and the Panel 4.8.3–4B are almost identical. The first one handles the constraining of station coordinates, the second handles the constraining of station velocities. The default sigma is applied for all selected stations or for stations listed in the selection file without station-specific sigma values. The selection file is used, if “`SPECIAL_FILE`” is specified in the first editable field. Remember that only selected stations contribute to the free network condition (i.e., you have to select the stations for free network condition either directly or using the selection file). Currently the free network condition is implemented as a minimal constraint (3 rotations). You can change this behavior easily in subroutine `FREENET`.

```
4.8.3-4C | ADDNEQ2: CONSTRAINING OF PARAMETERS

ORBITS:
Osculating elements:

major axis   eccentricity inclination
>        <  >        <  >        <
asc. node    perigee      arg. of lat.
>        <  >        <  >        <

Dynamical Parameters:

D0 (direct)  Y0 (Y bias)  X0
>        <  >        <  >        <
Dcos         Ycos         Xcos
> 0.1E-11 <  > 0.1E-11 <  >        <
Dsin         Ysin         Xsin
> 0.1E-11 <  > 0.1E-11 <  >        <

Stochastic Parameters:

radial       along track  out of plane
> 0.1E-05 <  > 0.1E-04 <  > 0.1E-05 <

EARTH ORIENTATION PARAMETERS

x Pole       y Pole       UT  first
>        <  >        <  > 0.0001  <
UT  other    Nut. Eps     Nut. Psi
>        <  > 0.001   <  > 0.001   <

TROPOSPHERIC DELAY        > 5.0     <

SAT. ANTENNA OFFSETS      > 0.0001  <

CENTER OF MASS            > 0.0001  <
```

In the Panel 4.8.3–4C the user defines the constraints for the remaining parameter types. The sigmas should be given in the following units:

| | | |
|---|---|---|
| 1 m | : | major axis, tropospheric delay, satellite antenna offsets, center of mass, |
| 1 " | : | inclination, ascending node, perigee, argument of latitude, |
| 1 m.s$^{-2}$ | : | dynamical and stochastic parameters, |
| 0.001 " | : | Earth orientation parameters with exception of UT, |
| 0.001 s | : | UT. |

```
4.8.3-5 | ADDNEQ2: REMAINING OPTIONS

Title (will be written into the output file)

>                                                                          <

Add Stochastic Orbit Parameters on NEQ Boundaries         > NO  <  (YES, NO)

Long-arc Combination                                      > NO  <  (YES, NO)

Time Interval for Storing EOPs (in hours)                 > 12 <

Block the Retrograde Terms in X and Y Polar Wobble Series > NO  <  (YES, NO)

Compute and compare individual solutions                  > NO  <  (YES, NO)

Reference Epoch for the Coordinates (YYYY MM DD or blank) >            <
```

Several more options have to be set in Panel 4.8.3–5. These options are similar to those of the old program ADDNEQ. We refer to Chapter 18 for their description. If the reference epoch for the (output) coordinates is left blank, the program will use the middle of the time interval covered by observations. The reference epoch is important only if the station velocities are computed — the coordinates stored in the coordinate output file will be computed for this epoch.

An example of the output of the ADDNEQ2 run is given in the Figure 19.9. The input and output files are listed at the top of the log file followed by the most important statistical informations (a posteriori rms error, number of parameters etc.). In the remaining part of the log file the estimates of various parameter types are given. In the Figure 19.9 only station coordinates and troposphere parameters are present. In case of station coordinates, there are the a priori values, resulting values and the corresponding rms errors in both, Cartesian and ellipsoidal coordinate systems. In case of troposphere parameters the station name, the direction (“`U`” stands for “up”), time interval, the estimated value, and the rms error are given. Remember that no information can be extracted from this output file using the GPSXTR program.

```
ADDNEQ2 output file created on 25-NOV-00 20:09

DOCU42_1 BPE EXAMPLE: TEST OF NEW ADDNEQ2 PROGRAM

Input and Output Files:
------------

CONST    X:/GEN/CONST.
DATUM    X:/GEN/DATUM.
STACRUX  X:/GEN/STACRUX.NEW
SATCRUX  X:/GEN/SAT_1996.CRX
SATELL   X:/GEN/SATELLIT.TTT
PHASECC  X:/GEN/PHAS_IGS.01
POLE     X:/GEN/R3_96165.ERP
SINEXIN  X:/GEN/SINEX.
COORD    P:/DOCU42_1/STA/ITRF0696.CRD
VELAPR
COVCOMI
NEQOUT   P:/DOCU42_1/OUT/EB296165.NQ0
COORDRS  P:/DOCU42_1/STA/EB296165.CRD
VELORS
SINEXRS
TROPSAV
TROPSNX
POLERS
IERSPOL
PLOTRS
SYSOUT   P:/DOCU42_1/OUT/EB296165.OUT
SYSERR
AUXFIL   U:/WORK/ADDNEQ2.SCR

List of NEQ and orbit files:
--------------

BRK01650.NQ0
JOWT1650.NQ0
KOON1650.NQ0
KOWT1650.NQ0
KOZI1650.NQ0

Results:
----

A posteriori RMS of Unit Weight: 0.0010 m

Number of Parameters:
-----------
Coordinates and Velocities      18
Orbits                           0
Troposphere                     72
EOPs                             0
Stoch. Orb.                      0
Sat. Ant.                        0
Center of Mass                   0


Coordinates:
------

BRUS 13101M004       4027893.8308               50 47 52.141298
                      307045.7093                4 21 33.182863
                     4919475.0352                      149.6582

                     4027893.8307 +- 0.0013     50 47 52.141060 +- 0.000046
                      307045.7117 +- 0.0009      4 21 33.182987 +- 0.000045
                     4919475.0236 +- 0.0015            149.6493 +- 0.0014


Troposphere Parameters:
------------
BRUS 13101M004    U  1996-06-12 23:59  1996-06-13  2:00    2.37811 +- 0.00240
BRUS 13101M004    U  1996-06-13  2:00  1996-06-13  4:00    2.37959 +- 0.00191
```

**Figure 19.9:** Output of the program ADDNEQ2.

# 20. Data Simulation and Variance-Covariance Studies

## 20.1 Principles of Data Simulation

The simulation part of the *Bernese GPS Software* includes the program GPSSIM. GPSSIM may be used to generate simulated observations which may then be processed by standard processing programs of the *Bernese GPS Software*. The capability of GPSSIM is limited to the generation of GPS data and this is why GPSSIM does *not* allow to generate synthetic GLONASS, or even "mixed" data.

You cannot start this program using the menu system. It is necessary to prepare (using an ASCII editor) the input files (`GPSSIMI.INP, GPSSIMN.INP, GPSSIMF.INP`) and start the program manually – see Chapter 3. You will find the examples for the input files in the directory `X:/INX`. According to the input specifications (see below) the following output files may be generated:

- code observations – code zero-difference (header and observation) files in the Bernese format (binary),
- phase observations – phase zero-difference (header and observation) files in the Bernese format (binary), and
- meteo files (in Bernese ASCII format – see Chapter 24).

In one program run, observations for *one session* are generated. With several program runs you may generate an entire simulated campaign.

The following input files define the scope for your observation scenario:

- the standard orbit file (`*.STD`) is used to compute satellite positions,
- the broadcast orbit file (`*.BRD`) is used only to apply satellite clock corrections, and
- the coordinate file (`*.CRD`) gives the station positions.

The broadcast orbit file and the standard orbit file should cover (roughly) the same time interval. Observations may be generated only in the time span covered by the standard orbit file, and only for stations in the coordinate file.

GPSSIM may be used for two different purposes:

Pre-Analysis: Given the satellite scenario, the network and the statistical a priori information, and the systematic errors, you may study the expected quality of the results depending on the processing strategy and options used (e.g., ambiguities free or fixed).

Research-Oriented Analyses: Questions concerning

- cycle slip repair (introduce gaps, etc.),
- ambiguity resolution strategies for long baselines,
- orbit determination,
- etc.

may be studied.

If the program system is used for pre-analysis studies, the user will not be interested to go through the preprocessing part. Therefore, the simulation program writes the true receiver clock information into the observation files and it includes the correct ambiguity parameters into the phase files. This information is preserved by program SNGDIF, the only program that has to be used before program GPSEST, if simulated data are processed by the *Bernese GPS Software* Version 4.2. It is then possible to make at once tests with the "ambiguity-fixed" mode of program GPSEST (assuming that you were able to resolve the ambiguities). By ignoring the ambiguity information in the files, GPSEST also allows you to make tests with the ambiguity-free mode. If you solve, e.g., for ambiguity parameters and store them in the single-difference file headers, you know the true answer (all ambiguities should be zero). Because GPSEST internally works with double differences, identical ambiguity values for all ambiguities of an ambiguity cluster (see Chapter 15) is a correct answer, too. Of course you may use all programs of the *Bernese GPS Software* with simulated observations. Some analysts may want, e.g., to remove cycle slips introduced by the simulation program with program MAUPRP. You may also be interested in studying the effect of using lower polynomial degrees in program CODSPP than in program GPSSIM, etc.

It is very easy to use the program GPSSIM for the present and past GPS satellite configuration. You simply start the simulation with a broadcast file corresponding to the time interval of interest. It is a good idea, but not mandatory, to use a broadcast file which was recorded at the same time of the year your new campaign will take place (the daily observation session will then take place at about the same time of the day). Then, by using programs BRDTAB and ORBGEN, you generate your standard orbit file. The next step consists of defining a ground truth by generating a coordinate file containing all stations of the campaign (not only of one session).

## 20.2 Models Used for Data Simulation

### 20.2.1 Stochastic Properties

In the `I`-file (`GPSSIMI.INP` – see below), you may specify the following rms errors:

- rms for code observations (separately for $L_1$ and $L_2$),
- rms for phase observations (separately for $L_1$ and $L_2$),
- rms errors for meteorological conditions (pressure, temperature, and humidity).

Each simulated measurement will contain a normally distributed random error with the rms specified in the input file for the measurement type considered. This is true for phase, code and meteo measurements. The phase and code measurements generated will not contain any meteo errors (we assume that “natura (resp. troposphere) non facit saltas”). In addition to these random errors, you may assign *systematic* errors to the troposphere measurements by specifying maximum biases for temperature, pressure and humidity measurements. The program then generates constant biases for the session but different for each station using a random number generator: it assigns to each station and to each measurement type a bias which is a random number uniformly distributed in the interval

$$I = \langle -B_{max}, +B_{max} \rangle,$$

where $B_{max}$ is the maximum bias you specified in the input file for the measurement type considered. There is one more stochastic quantity, the random part of the free electron content of the ionosphere, which has to be mentioned. Since this is closely related to the discussion of the ionosphere model used, we will deal with this effect in the next section.

The random numbers are generated with the subroutines

| | |
|---|---|
| – NORMAL | generating a normally distributed random number (given its mean and variance), |
| – RANDU | generating a uniformly distributed random number in the interval $\langle 0, 1 \rangle$. |

NORMAL calls RANDU several times. RANDU updates a positive `INTEGER*4` random variable which has to be initialized. It would have been possible to initialize this number in a “random way” (using, e.g., the last digits of the system clock reading), but it seemed advisable to have this initialization under control by initializing this random number in the input file. We suggest that you use positive four-digit integers for initialization. If you re-run the program with the same input specifications and the same initial random number, the same random errors will occur.

### 20.2.2 Deterministic Models

#### Satellite Orbits, Clocks, Coordinates

We mentioned already that the satellite positions are computed using a standard orbit, the satellite clocks using a broadcast file, the station coordinates using the file with station coordinates. Therefore, the model for satellite orbits is the complex force model underlying program ORBGEN, the model for satellite clocks is purely deterministic (polynomial of degree 2 defined in the message). Moreover the user has to define the receiver clock behavior (by a polynomial) in a file of the following kind:

```
RECEIVER CLOCKS AS POLYNOMIALS OF DEGREE N-1
POLYNOMIAL COEFFICIENTS IN SEC, SEC/DAY, SEC/DAY**2,...
 STATION         N      A0        A1        A2        A3        A4        A5
ZIMMERWALD       6 +0.016017-0.000123+0.000222-0.002000-0.012345+0.000555
CHASSERAL        6 -0.003002+0.001234-0.000000+0.010000-0.034556-0.023476
GENEROSO         6 +0.020000-0.010000+0.010000-0.005045-0.000345-0.123456
TITLIS           6 -0.013000+0.020000-0.001234+0.001000-0.006000-0.500000
```

If you are not interested in receiver clocks, just state N=0 for each station.

### Troposphere

Tropospheric refraction may be neglected or it may be modeled by the

- Saastamoinen Model,
- Hopfield model (2 versions).

These models ask for (ground) meteo values for each station. These values are generated using the reference height and the temperature, pressure, and humidity at this reference height, as specified in the constant file. This allows you, e.g., to simulate data with a standard atmosphere different from that used by program GPSEST by using different constant files in programs GPSSIM and GPSEST.

### Ionosphere

The "ionosphere truth" (deterministic part) is defined by the subroutine IONOS. The model underlying is the single layer model for the electron content (height of the layer=350 km). The number of electrons in the layer E is a function of local time only:

$$\begin{aligned} T_{loc} &= UT + \lambda \\ E &= \begin{cases} E_0 & \text{for } \ T_{loc} \in \langle 20^h, 8^h \rangle \\ E_0 + E_1 \, \cos\left(\dfrac{T_{loc} - 14}{12}\,\pi\right) & \text{for } \ T_{loc} \in \langle 8^h, 20^h \rangle \end{cases} \end{aligned}$$

where : $E_0$ is the night time electron content,
$E_1$ is the day time variation of the ionosphere.

$E_0, E_1$ are input parameters of program GPSSIM. No azimuth dependence is modeled here. In addition to this regular part an irregular part may be superposed: at epoch $i$ the following term is added

$$\delta E_i = (E_{i1} \, \cos a + E_{i2} \, \sin a)\; d/200$$

| where: | | |
|---|---|---|
| | $\delta E_i$ | is the random part of the electron content in the atmosphere, |
| | $d$ | is the distance in km between the station considered and the ionosphere reference station (specified in the input file), |
| | $a$ | is the azimuth of the station considered from the ionosphere reference station, |
| | $E_{ik}\ , k = 1, 2$ | is the result of the following random walk: $E_{1k} = 0 \;\;,\quad \text{var}(E_{i+1,k} - E_{ik}) = I_i \cdot \text{var}_{1min}\ ,$ where $I_i$ is the time between observations $i, i+1$ in minutes, $\text{var}_{1min}$ is the variance of the difference of the electron contents at time $t$ and $t+1$ minute. $\text{var}_{1min}$ is an input variable of program GPSSIM. |

This model certainly is a crude simplification for the random behavior of the true ionosphere. It shares, however, important aspects with the real situation: (1) the irregular contribution increases linearly with the baseline length, for distances $> 200$ km this distance dependence “stops”, (2) baselines which are close to each other geometrically will show similar random contributions. The strength of the irregular contribution may be tuned by an input parameter.

### Cycle Slips

If you like preprocessing and its secrets, you may introduce cycle slips in the simulation program. It is then really hard to tell whether you are looking at real or at simulated observations . . .

## 20.3 Essential Input Files

Let us briefly present the basic input files for GPSSIM, and add some comments to the input options.

### Option Input File (`I`-File):

- Campaign name and title line will show up in the output files. This helps to identify simulated files.
- Program SNGDIF will assume simulated data, if the receiver type starts with “`SIMULA`”. Therefore the string “`SIMULA`” is essential in the input part “`RECEIVER TYPE`”.
- In the troposphere model, you may wish to assume identical biases for all station, you may even consider to skip the random number generator for the bias part by using the maximum biases. The troposphere biases actually applied are given in the program output.
- In the scenario input section, you may specify a minimum elevation and the data sampling rate. For pre-analysis purposes we often use a very low rate (4 to 6 minutes) in order to save disk space. For other applications (e.g., investigations concerning minimum session lengths which allow you to resolve ambiguities) you may use short session lengths and high data rates (e.g., 3 sec).
- In the input section “cycle slips”, you may specify how many cycle slips you wish to introduce into each file of the session. The program assumes that a slip may occur with the same probability at each epoch and for each satellite and that its size is uniformly distributed in the interval $I = \langle -slip_{max}, +slip_{max} \rangle$.
- You may wish to apply identical slips in $L_1$ and $L_2$. These are particularly hard to discover. It might be of interest to look at the quality of results if, let us say, $n$ slips of size 1 remain undiscovered in the data for $n = 1, 2, \ldots$ .

```
GPSSIM: OPTION INPUT FILE                                      21-APR-90 13:02
-------------------------------------------------------------------------------

(REMARK: YES=1,NO=0 ; 2 EMPTY LINES AFTER EVERY INPUT GROUP)

CAMPAIGN NAME:
-------------
      ****************

--> : TESTV42

TITLE LINE:
----------
      *****************************************************

--> : SIMULATED DATA TO TEST VERSION 4.2
RECEIVER TYPE:
-------------
             *********

--> : SIMULA

SESSION DEFINITION:
------------------
             FROM                 TO
       YYYY MM DD HH MM    YYYY MM DD HH MM

--> :  1995 06 16 10 30    1995 06 16 14 00

                                                        ****

SESSION NUMBER                                  --> :   167
FILE NUMBER (SAME SESSION, SAME STATION)        --> :     1
TIME BETWEEN SUBSEQUENT OBSERVATIONS (SEC)      --> :   120
MIN. ELEVATION (DEGREES)                        --> :    20

TROPOSPHERIC MODEL:
------------------                                      *

NO METEO                            =  0
SAASTAMOINEN                        =  1
MODIFIED HOPFIELD (REMONDI)         =  2
SIMPLIFIED HOPFIELD (DAVIDSON)      =  3        --> :  1

                                                        ******

RMS ERROR FOR PRESSURE MEASUREMENT              --> :   2.0    MBAR
RMS ERROR FOR TEMPERATURE MEASUREMENT           --> :   1.0    DEG C
RMS ERROR FOR HUMIDITY MEASUREMENT              --> :   5.0    %

MAXIMUM PRESSURE BIAS FOR ONE STATION           --> :   3.0    MBAR
MAXIMUM TEMPERATURE BIAS FOR ONE STATION        --> :   2.0    DEG C
MAXIMUM HUMIDITY BIAS FOR ONE STATION           --> :  10.0    %

                                                        *

SAME BIAS FOR ALL STATIONS                      --> :  0
MAX.BIASES = ACTUAL BIASES FOR ALL STAT.        --> :  0
```

```
IONOSPHERIC MODEL:
-----------------                                       ***

NIGHT TIME ELECTRON NUMBER (IN 10**16)           --> :   20
DAY   TIME ELECTRON NUMBER (IN 10**16)           --> :   30
VARIANCE OF IRREG. CHANGE OF IONOSPHERE CONTENT
  IN 1 MIN. AT 200 KM FROM REF.SITE (IN 10**15)  --> :    1
                                                        ****************
REFERENCE SITE                                   --> :  ZIMMERWALD

STATISTICAL INFORMATION:
-----------------------              **.***

A PRIORI SIGMA CODE L1         --> :   1.000 M
A PRIORI SIGMA CODE L2         --> :   1.000 M
A PRIORI SIGMA PHASE L1        --> :   0.003 M
A PRIORI SIGMA PHASE L2        --> :   0.003 M

                                     ******

INITIAL INTEGER RANDOM NUMBER  --> :    7778

INTRODUCE CYCLE SLIPS:
---------------------                ******

NUMBER OF SLIPS PER FILE       --> :       0
MAXIMUM SIZE OF SLIP           --> :     100

SAME SLIPS IN L1 AND L2 ?      --> :       0
```

Session Definition File (F-File):

The file is truncated to the right: the definition of phase header and observation files is missing. The file defines code/phase header/obs file names (external) and meteo file names. It also relates the file to the station (name) with the additional information at the bottom of the file.

```
GPSSIM: OUTPUT FILE NAMES FOR CODE, PHASE, AND METEO
-------------------------------------------------------------------------..
CODE HEADER FILE NAME (OUTPUT)   CODE OBSERVATION FILE NAME (OUT) PHASE ..
******************************** ******************************** ******..

X:/CAMPNAM/OBS/CHAS1671.CZH      X:/CAMPNAM/OBS/CHAS1671.CZO      X:/CAM..
X:/CAMPNAM/OBS/GENE1671.CZH      X:/CAMPNAM/OBS/GENE1671.CZO      X:/CAM..
X:/CAMPNAM/OBS/TITL1671.CZH      X:/CAMPNAM/OBS/TITL1671.CZO      X:/CAM..
X:/CAMPNAM/OBS/ZIMM1671.CZH      X:/CAMPNAM/OBS/ZIMM1671.CZO      X:/CAM..


METEO FILE NAME (OUTPUT)
********************************

X:/CAMPNAM/ATM/CHAS1671.MET
X:/CAMPNAM/ATM/GENE1671.MET
X:/CAMPNAM/ATM/TITL1671.MET
X:/CAMPNAM/ATM/ZIMM1671.MET

STATION NAME      REC.UNIT  ANTENNA  OPERATOR NAME
****************    ****     ****    ****************

CHASSERAL              1      101    OPERATOR 1
GENEROSO               2      202    OPERATOR 2
```

- The information on which types of observations should be created is located in the receiver file (see Chapter 24).
- If at least one of the meteo error parameters (rms or max. bias) is different from zero, the program will produce meteo measurement files. The names of these files have to be defined in the `F`-file above.

# 21. Services

In this chapter the options of Menu 5, as described below, will be discussed. The topics “POLE” and “COORDINATES” are described in Chapters 14 and 11, respectively, and will therefore not be discussed here.

```
 5       |                 SERVICES: OPTION MENU
---------+------------------------------------------------------------------
         |
---------+------------------------------------------------------------------
         |
   1     |  OBSERVATIONS    : Browse, Edit , Update and Graphic Obs. Files
   2     |  FILE HEADERS    : Change the Content of Obs. File Headers
   3 ..  |  RESIDUALS       : Browse and Check Residuals
   4 ..  |  COORDINATES     : Compare Coord. Files, Helmert Transformation
   5 ..  |  POLE            : Update and Extract Pole Information
   6 ..  |  EXTRACTIONS     : Generation of output summaries
   7 ..  |  BINARY <-> ASCII: Conversion Between Binary and ASCII Files
   8     |  DELETE FILES    : Delete GPS Specific Files
   9     |  JOB             : Processing of Job Output
```

## 21.1 Observations and File Headers

Since Menu 5.1 is frequently used, one may also invoke it by typing “OBS” after the prompt “Enter Selection”. This panel is used to manipulate the code and phase zero and single difference observation files. The menu program used here is called SERVOBS.

```
 5.1     |                 SERVICES: OBSERVATIONS
---------+------------------------------------------------------------------

   B - Browse Observation File         M - Mark Observations or Satellites
   E - Edit Observation File           D - Delete Observation File
   G - Graphic of Observations         C - Create File Table
   2 - Split Observation File          A - Add Files to the File Table
   H - Edit Header File only           R - Reorder Files in File Table
   X - Exit

   Option:               >   <         (blank: Select option in file list)

   CAMPAIGN           > IGSA     <     (blank for selection list)

 Input File:
   MEASUREMENT TYPE     > PHASE  <     (CODE, PHASE, BOTH /options C,A,R/)
   DIFFERENCES          > ZERO   <     (ZERO or SINGLE)
   OBSERVATION FILE    > %%%%$D11 <    (blank for selection list)
```

In the upper half of this panel the options are listed with a short description.

The create ("C") or add ("A") options are used to either create a new observation file list from scratch or to update an existing list file list. Usually, the observation file list will always be up-to-date because the two programs generating observation files, RXOBV3 and SNGDIF, update the observation file list automatically. However, if files are copied or deleted manually, the observation file list is *not* updated, which is why these two options exist. One should keep in mind that the list of files shown when selecting an option is based on the observation file list files and *not* on the actually available observation files. Individual observation file lists exist for the four different categories of observation files: zero-difference phase, zero-difference code, single-difference phase, and single-difference code. The observation file lists are discussed in Chapter 24. Apart from creating or updating observation file lists, we may browse ("B") or edit ("E") the observation files. Browsing a file does not allow to alter the file, while editing does. Editing an observation file is only rarely necessary.

Occasionally, one might want to change some information in the file header if it contains erroneous information. For that purpose the "H" option should be used since it is much quicker only to edit the header and not the entire file. Manually editing the observation file should be done cautiously. One should, e.g., not remove any satellites from the header file, nor change the number of satellites and the number of frequencies. Also, do not change the ambiguity cluster information. One may change observation flags and remove observation epochs.

Program CHGHED, Menu 5.2, allows to change the file header automatically. This is useful if numerous header files have to be changed. Erroneous information may be stored in the file header if the RINEX files contained wrong information, or if the information in the translation tables used with RXOBV3 were in error. The "old values" in the input area of the program CHGHED must be exactly identical with the entries in the observation file headers. Otherwise no change will be done. Therefore, after running program CHGHED be sure to check the output of the program for the changes performed in the observation header files.

The graphics option ("G") produces similar results as program RNXGRA: it produces a pseudo-graphic of each selected observation file. These pseudo-graphics may be generated for both, zero and single difference files. Another option is to split up the observation files into two parts ("2"). This is useful if you have data from a campaign with different start and end times. It will also be useful if, e.g., the antenna height was changed during the observation time span. However, we recommend to split up data files on the RINEX level rather than after the conversion to the Bernese format. Programs CCRINEXN and CCRINEXO are available for this purpose, see Chapter 7.

The mark option ("M") is used to mark data in the observation files. This option may be used to "throw away" bad satellites or to remove bad data points using an "edit file". The use of an edit file will be discussed later in this chapter. The format of the edit file is discussed in Chapter 24. If no "edit file" is specified, a set of satellites and a time interval (as time window or epochs) can be selected in the panel to mark, delete, or unmark the observations manually.

The delete ("D") option may be used to delete files. The files will be deleted and removed from the observation file list. In Menu 0.1 you specify whether the deletion should be done with confirmation or not.

The reorder ("R") option may be used to change the file order. By default, we use the session number as first criterion to order the files followed by the station name and, finally, by the file sequence number. With the reorder option you may specify a different ordering.

Finally, you exit this menu panel using the "X" option. You may also use "X" to exit when you are in an observation file selection list. Pressing "Q" when in an observation file selection list will put you back to the services panel.

## 21.2 Residuals

In Menu 5.3, three different options to work with residual files are available. One may wish to look at the residuals as numbers printed on screen or written to a file, to screen the residuals for outliers or to look at the residuals graphically.

```
 5.3  |                  RESIDUALS: OPTION MENU
------+-------------------------------------------------------------
      |
------+-------------------------------------------------------------
      |
  1   |   BROWSE RESID.   : Browse Residuals of GPSEST,MAUPRP,CODSPP
  2   |   CHECK RESIDUALS : Check Residuals for Outliers
  3   |   GRAPHIC TOOL    : Display Residuals with GT
      |
```

The "browse residuals" option allows you to inspect the residual files from either CODSPP, MAUPRP, or GPSEST using program REDISP. The residuals will either be displayed onto your computer screen or written into a file. The latter option is useful if you run into problems during processing. It will help you to identify a station causing problems. However, going through the data of a large number of stations "manually" is quite cumbersome. Therefore, the residuals may also be screened automatically with the "check residual" option in Menu 5.3.2.

The "check residual" option will run program RESRMS. This program screens the residuals for outliers. Outliers are observations with residuals exceeding a threshold defined by the user. RESRMS creates a summary file and a so-called "edit file" (see Chapter 24 for a description of the format of the edit file). The edit file contains the points identified as outliers in the residual file(s). The file may be used with the SERVOBS program, Menu 5.1 option "M", to remove these points from the actual observation files.

The "graphic tool", GT, is currently only available on some UNIX systems, and on the VAX/VMS platform. It has been developed by UNAVCO and is freely available. For more information you may inspect the URL `http://www.unavco.ucar.edu/software/visualization/gt.html` or contact Chris Rocken at UNAVCO (e-mail `rocken@ucar.edu`). The graphic tool will give a (X-Windows) plot of the residuals and allows manual deletion of points, correction of cycle slips, and the setting up of new ambiguity parameters. These manual editing requests are also handled by means of an "edit file" and have to be applied to the observation files using the program SERVOBS.

## 21.3 Extractions

There are several extraction programs capable of summarizing the output files of the most important/critical programs of the *Bernese GPS Software*. They are very useful for automated processes (BPE), because they will summarize the relevant information and allow the user to verify easily whether the processing was successful or not. Extraction programs are available for programs CODSPP, ORBGEN, MAUPRP, GPSEST, and ADDNEQ. It does not work with the program ADDNEQ2 in the Version 4.2 of the *Bernese GPS Software*.

```
 5.6                       EXTRACTIONS


  1    CODSPP OUTPUT    : CODSPP output summary              (CODXTR)
  2    DEFSTD OUTPUT    : DEFSTD output summary         (DEFXTR/DEFXTP)
  3    MAUPRP OUTPUT    : MAUPRP output summary              (MPRXTR)
  5    GPSEST/ADDNEQ OUT: General summary (several formats)    (GPSXTR)
  7    ORBIT WEIGHTS    : Change accuracy codes in precise orb.(PREWEI)
```

## CODXTR

CODXTR, the extraction program for program CODSPP, produces a summary of all CODSPP output in the following form:

```
CODSPP EXTRACTION
-----------------
  23   4  GUAM 50501M002    OUT  19  96-08-06  6:26:30  96-08-06  7:07:30    83
   3   6  USUD 21729S007    OUT  19  96-08-06  7:02:30  96-08-06  7:59:30   115
   7   7  ALBH 40129M003    OUT  19  96-08-06  6:29:00  96-08-06  8:00:00   183
   8   8  ALGO 40104M002    OUT  19  96-08-06  6:39:30  96-08-06  8:00:00   162
  14  10  CHUR              OUT  19  96-08-06  6:26:30  96-08-06  8:00:00   188
  21  12  DRAO 40105M002    OUT  19  96-08-06  6:27:30  96-08-06  8:00:00   186
  23  13  FAIR 40408M001B   OUT  19  96-08-06  6:26:30  96-08-06  6:56:00    60
  24  13  FAIR 40408M001B   OUT  19  96-08-06  6:57:00  96-08-06  7:59:30   126
  26  14  FLIN              OUT  19  96-08-06  6:32:30  96-08-06  7:59:30   175
  28  15  GODE 40451M123    OUT  19  96-08-06  6:40:30  96-08-06  7:59:30   159
   4   2  KELY 43005M001    OUT  19  96-08-06  6:58:30  96-08-06  7:59:30   123
   7   3  NLIB 40465M001    OUT  19  96-08-06  6:26:30  96-08-06  7:59:30   187
  11   4  MDO1 40442M012    OUT  19  96-08-06  6:27:30  96-08-06  7:59:30   185
  14   5  PIE1 40456M001    OUT  19  96-08-06  6:27:30  96-08-06  7:59:30   185
  16   6  QUIN 40433M004    OUT  19  96-08-06  6:28:00  96-08-06  7:59:30   184
  19   7  RCM5 40499S018B   OUT  19  96-08-06  7:11:00  96-08-06  7:59:30    98
  22   8  THU1 43001M001    OUT  19  96-08-06  6:35:00  96-08-06  7:59:30   170
  23   9  WES2 40440S020    OUT  19  96-08-06  6:44:00  96-08-06  7:59:30   152
  28  10  WHIT              OUT  19  96-08-06  6:27:30  96-08-06  7:59:30   185
  30  11  YELL 40127M003B   OUT  19  96-08-06  6:27:30  96-08-06  6:56:00    58
  31  11  YELL 40127M003B   OUT  19  96-08-06  6:57:00  96-08-06  7:59:30   126

 61 FILES, MAX. RMS:   39.28 m for station: YAR1 50107M004
           MAX. BAD:   12.63 % FOR STATION: ONSA 10402M004
```

The summary file will contain at least two lines, namely the last two lines in the above example. These lines of the summary give the number of code files that were processed, the maximum rms, and the station for which this maximum occurred. The rms value should be around 25 meters, under SA conditions. Values up to 300 meters are still acceptable, although they deserve special attention. The rms value is of the order of about 5 m since SA was switched off in May 2000. In the second line, the maximum percentage of bad observations in a code zero-difference file is given together with the name of the station for which this occurred.

Furthermore, it shows one line for each station–satellite combination where more than 50 points are considered outliers. This is the case in the above example. The first two numbers on these “outlier” lines are not important, they reflect some internal CODSPP numbering. Following these two numbers is the station name, the outlier identification, and the satellite PRN number. The outlier identification will usually be “`OUT`”. The line is ended with the start and end epochs of the outlier interval and the number of observations in this interval.

If a specific station shows up often, something is probably wrong with its data. If a certain satellite shows up frequently, there is something wrong with the satellite. In the above example, satellite 19 was marked for many sites. As the marked time intervals are during the day (UT) and not only towards the end of the day, the outliers are not caused by a maneuver. In fact, on this particular day satellite 19 was scheduled for maintenance. Usually the satellite clock is reset in such cases. The outlier detection of CODSPP handles such problems and no action is required by the user.

### DEFXTR and DEFXTP

For program ORBGEN the extraction programs DEFXTR and DEFXTP may be used. We recommend to use only the latter of the two. The difference between the programs has to be seen in number and the format of the output files generated. We discuss only the program DEFXTP and the two output files associated with it, the normal summary and the weekly summary.

```
ORBGEN.L97      # Sat.:  25 , # Eclipsing  9 , Max. Rms.:  0.38 for sat.: 10
(DOY: 219)      Eclips. Sat. :    3   7  10  14  16  21  23  28  31
                Min in eclips:   24  23  11  28  30  20  30  13  22
                Rms          :   37  13  38  14  31  11  11  11   9
```

The normal summary contains 4 lines. The first line contains the output file name, the number of satellites, the number of eclipsing satellites, the maximum rms of the fit, and the satellite associated with this rms. The second line gives the (middle) day of year for the arc and a list of the PRN numbers for the eclipsing satellites. The third line gives the time interval (minutes) of the eclipse, the last line gives the rms (in cm) of the fit for the eclipsing satellites. Usually, the rms of eclipsing satellites is larger than the rms of non-eclipsing satellites.

```
219   6   4278   4   3   4   7   3 31256   5 21   4   4 11   6   4 35   4   7   4   5   6   3   5
```

The weekly summary produces just one line per ORBGEN output and contains the (middle) day of the year for the arc and the rms for all satellites. The satellite numbers are not printed but they are listed in numerical order. This output, if always appended to one file to give a long time series, is useful for generating plots.

### MPRXTR

The extraction program MPRXTR, for program MAUPRP, produces a summary of the following type:

```
SUMMARY OF THE MAUPRP OUTPUT FILE
*********************************

SESS FIL OK?  ST1  ST2 L(KM) #OBS.  RMS   DX    DY    DZ   #SL #DL #MA  MAXL3      MIN. SLIP
--------------------------------------------------------------------------------------------
2191    1 OK   NCRO SKOU 1879  4819 0.012 0.001-0.020-0.034   8 940  85  0.030             85
2191    2 OK   NCHU NFLI  658  9422 0.018 0.009-0.017 0.057  24 486 108  0.011         283610
2191    3 OK   NCHU NTHU 2202  8485 0.012-0.006 0.004-0.039  25 882  95  0.014         283610
  .    .  .     .    .    .     .   .     .     .     .      .   .   .   .               .
  .    .  .     .    .    .     .   .     .     .     .      .   .   .   .               .
  .    .  .     .    .    .     .   .     .     .     .      .   .   .   .               .
2191   67 OK   EWTR EZIM  476  9820 0.007 0.032-0.003 0.054  31 907  49  0.013        1821677
2191   68 OK   EWTZ EZIM  476  8612 0.009 0.036-0.005 0.033   2 369  61  0.015        1990187
2191   69 OK   AYAR JISC 5937  1964 0.007 0.003 0.031 0.058  24 499  52  0.004        6384009
-------------------------------------------------------------------------------
Tot:  69                2228  6997 0.008                   295 575  70
```

The summary file contains one line per baseline processed. The line starts with the session, a file sequence number and the "OK" flag. If the baseline was not successfully cleaned the flag will be different from "OK" and the baseline should be removed. The extraction program will also (try) to identify which of the two stations of the baseline was causing the problems. Apart from the summary file, MPRXTR may also create a deletion list file, see Chapter 24, and a new baseline definition file. The "bad" single difference file(s) and, optionally, the bad zero-difference file(s) will be listed in the deletion list file. This file may then be used by program DELFIL, Panel 5.8, to delete these files. If a zero-difference file was removed a new baseline should be created to make sure that the network is complete. MPRXTR defines a new baseline which is listed in the baseline definition file. This file in turn may then be used by program SNGDIF, Panel 4.3, to actually create the new baseline. Make sure that you do not forget to run MAUPRP for the newly created baseline. Note that the deletion list file and the new baseline definition file are mainly intended to be used in an automated processing.

The next items in the line, after the OK flag, are the first 4 characters of the station names and the length of the baseline. Thereafter some information concerning the triple-difference solution is given: the number of observations, rms, baseline change in the Cartesian X, Y, and Z components. The next three fields contain the number of corrected cycle slips, the number of deleted points, and the number of multiple ambiguities. Finally, the maximum residual for the ionosphere-free linear combination is given followed by the smallest corrected cycle slip.

The last line of the summary gives the number of files, the mean baseline length of all baselines, the mean number of observations, the mean triple-difference rms, the mean number of corrected cycle slips, the mean number of deleted points, and the mean number of multiple ambiguities. See Chapter 10 for more information about the program MAUPRP.

## GPSXTR

Program GPSXTR consists of four different extraction routines for specific purposes. First, there is the "general" extraction which works for both programs GPSEST and ADDNEQ. The other three extraction routines access GPSEST output files only. One is a coordinate summary based on a baseline-wise processing (one baseline per run). The second creates statistics for the baseline-wise "QIF" ambiguity resolution runs, and the third generates statistics for ionosphere estimation runs.

Included below is an example for the "general" extraction output.

```
ADDNEQ.L98      Rms:  2.9 , # fil.:  69 , # obs.: 144888 , # par.:  2873
(DOY:   0)                  Max. correction in a:  -0.13 +- 0.01 for sat.:  7
                            Pole correction (middle of interval)
                                       in x(mas):  -2.90 +- 0.119
                                       in y(mas):   0.83 +- 0.101
                                  in t(sec*1/D6):  52.40 +- 5.05
```

The first line contains the output file name, the rms (mm) of the solution, the number of single difference files, the number of observations used, and the number of estimated parameters. The second line should first contain the day of year, but this is not always correct as can be seen in the above example. If orbit estimation was performed, the maximum change in the semi-major axis (“a”) and the corresponding satellite are listed as well. The last four lines summarize the ERP estimates if ERPs were estimated and no separate pole output file was generated.

Apart from this summary, two one line summaries (“campaign format” and “weekly summary”) may be generated containing the same information in a different format. These one line summary files are very convenient to generate plots. Furthermore, a pole output summary, if pole estimation was performed, may be generated.

At CODE, we usually check the processing quality by processing all baselines one by one with GPSEST. In this case GPSXTR should be used to generate a coordinate summary of the following type:

```
BASELINE  #OBS.  #AMB  RMS(MM)  L(KM)  DHT(M)    DH(MM) +-    DN(MM) +-    DE(MM) +-    DL(MM) +-
------------------------------------------------------------------------------------------------------
ABDR2181  1989    45     1.9   302.      510.1     5.0  2.5     0.3  0.4    -1.1  0.9    -0.8  0.9
ABQU2181  1632    47     2.0   956.     1074.0    -7.4  4.0    -0.3  1.0     1.8  1.4     0.1  1.1
ABWH2181  1506    47     2.0  1562.     1395.6    -5.4  3.8     4.5  1.3    -0.2  1.5     3.2  1.4
```

For each run of program GPSEST, this file contains the name of the single difference file, the number of observations, the number of ambiguities, the a posteriori rms, the length of the baseline, the height difference between the two points and the change of the baseline coordinates in height, north, east, and length with respect to the a priori coordinates together with their associated formal uncertainties.

When using the “QIF” strategy (baseline by baseline) to resolve the ambiguities one may generate a special output summary and, in addition, a file which lists the fractional parts of all the resolved ambiguities.

```
File       Length   #Amb  RMS0   Max/RMS L5 Amb  Max/RMS L3 Amb   #Amb  RMS0   #Amb Resolved
           (km)           (mm)   (L5 Cycles)     (L3 Cycles)            (mm)   (%)
---------------------------------------------------------------------------------------------
ABDR2191    301.8    102   1.7    0.250  0.058    0.090  0.032      10   2.1    90.2
ABQU2191    956.0    100   2.4    0.440  0.166    0.099  0.035      12   2.8    88.0
ALGD2191    776.5    102   2.6    0.405  0.107    0.093  0.034      14   2.9    86.3
   .           .       .    .       .      .        .      .         .    .       .
   .           .       .    .       .      .        .      .         .    .       .
   .           .       .    .       .      .        .      .         .    .       .
TAUS2191    987.2     76   2.6    0.440  0.161    0.100  0.040      12   3.3    84.2
WRZI2191    475.9     80   2.7    0.307  0.061    0.099  0.031      10   3.0    87.5
WZZI2191    475.9     78   2.9    0.357  0.068    0.098  0.030      10   3.1    87.2
---------------------------------------------------------------------------------------------
Tot:  42    912.5   4084   2.8    0.499  0.130    0.100  0.037     766   3.2    81.2
```

For each run of program GPSEST, this summary contains the name of the single difference file, the length of the baseline, the number of ambiguities (L1 and L2) before resolving any ambiguity and the rms before resolving any ambiguity, the maximum fractional part and the rms for all resolved ambiguity fractional parts for both, the L5 and L3 linear combination, the number of unresolved ambiguities, the rms after ambiguity resolution. The percentage of resolved ambiguities concludes the line.

In the last line of the summary the number of baselines used, the mean length of the baselines, the total number of ambiguities before resolving, the mean rms before resolving, the maximal fractional part and rms of all ambiguities and all baselines for both, the L5 and L3 linear combination, the total number of ambiguities after resolution, the mean rms after resolution and the percentage of fixed ambiguities.

If program GPSEST was used to estimate an ionosphere model a separate ionosphere summary output can be generated.

## 21.4 Conversions

The observation files in the Bernese format are binary files and as such, they cannot be looked at directly and cannot be transferred between different operating systems (e.g., VMS to UNIX). Therefore, two conversion programs are made available: OBSFMT, to convert binary observation files to ASCII and the second, FMTOBS, to convert the ASCII observation files back to binary. These programs are also used when editing or browsing files in Panel 5.1

Similar conversion programs exist for the so-called standard orbit files, Chapter 24. These are mainly used to transfer standard orbits to a different operating system. There is not much sense in editing standard orbit files.

## 21.5 Delete Files

The program DELFIL, Menu 5.8, is used to delete files stored in the campaign directories. This option was implemented for the BPE, but may also be used manually. A so-called deletion file, see Chapter 24, may be specified containing names of files to be deleted (including the use of wildcards and “$” variables). You may also specify which file types should be deleted and select specific files of this type from a selection list.

# 22. Bernese Processing Engine (BPE)

## 22.1 Introduction

In the last few years more and more permanent networks have been established in many different regions of the world. The amount of data collected by such networks day by day calls for a highly automated GPS data processing. Large campaigns are organized with a few tens to a few hundreds of occupied sites. For this type of tasks the *Bernese Processing Engine* (BPE) has been developed allowing a very automated GPS data processing. It is used at the CODE Analysis Center of the IGS for the routine processing of the global IGS network since 1995. The BPE was also implemented at GSI (Geographical Survey Institute) in Japan to process the nation-wide Japanese network consisting of 1'200 receivers on a daily basis.

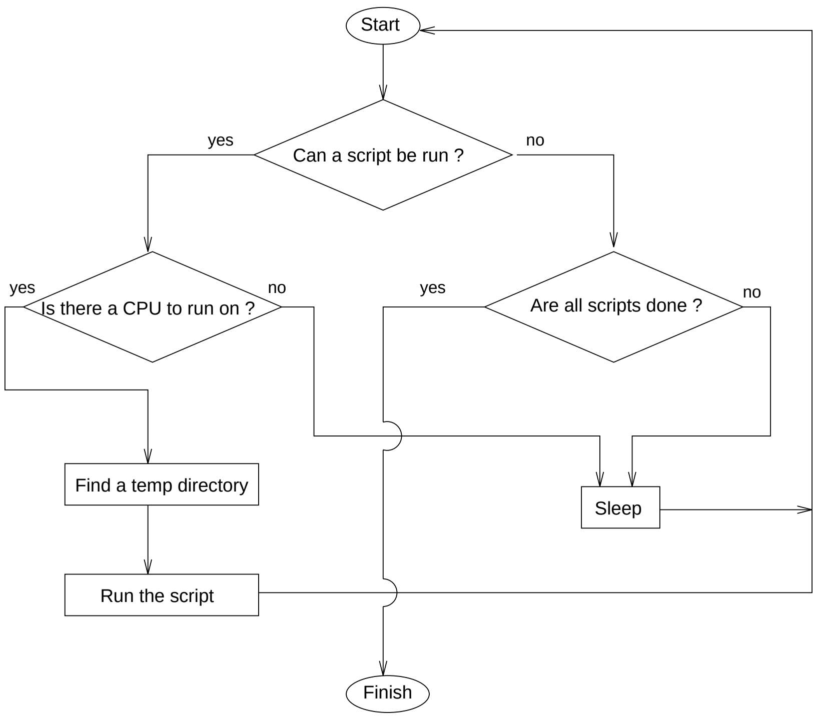


**Figure 22.1:** Process Control Script flow chart.

The BPE is a system of programs, shell scripts, and control files designed to run and control Bernese GPS programs in an automated fashion. At the heart of the BPE is the Process Control Script (PCS). This script is responsible for starting and monitoring all the processes that are run by the BPE. The PCS can be given a list of various tasks to perform along with information about the interdependency of these tasks, such as which tasks need to wait for other tasks to finish. The PCS will then start executing each task that is ready to run. If a task cannot be run because it has to wait for another one to finish, the PCS will wait for the dependent tasks to finish. The PCS is also able to run more than one task at a time on several different CPUs, and can even divide a single task across different CPUs. A flow chart of what the PCS does is given in Figure 22.1.

The BPE provides a sturdy framework for writing shell scripts that can use the many different programs available in the *Bernese GPS Software*. It takes care of setting up environment variables, creating temporary work directories, error handling, logging, and interaction with the Bernese menu system. The actual body of a script that is run by the BPE can be very simple, such as calling one Bernese program, or quite complicated.

## 22.2 Getting Started

### 22.2.1 LOADGPS

As with the Bernese menu system, the BPE is started with the LOADGPS script. There are, however, some additional requirements on the LOADGPS script in order for the BPE to function correctly as listed below.

UNIX Version:

- The `SHELL` variable in LOADGPS must be set to a Bourne (`sh`) compatible shell. Shells such as the Bourne-Again shell (`bash`) can be used as long as they can interpret all standard `sh` commands.
- The LOADGPS script must have execute permissions on all hosts that are to be used by the BPE. The BPE executes scripts by issuing a remote shell command to start the LOADGPS script. The location of the LOADGPS script will be taken from the content of the variable $LOADGPS as defined on the system where the BPE is running. This means that the location of the LOADGPS script must be identical (or must be made identical by links) on all hosts.

VMS Version:

- The call of the command file `LOADGPS` has to be included in the user's `LOGIN.COM` (located in the user's home directory) to make sure all symbols and logicals are correctly set.

### 22.2.2 LOADGPS Command Line Arguments

UNIX Version:

Usually, the LOADGPS script is run without command line arguments. On UNIX systems, the user may want to specify a command line argument in two cases: first, if one of the directory variables in the LOADGPS script has been changed, second, if the user wants to run a script automatically after the LOADGPS script starts. The second option is useful if the BPE should start automatically on some sort of schedule. The two options available are shown below. Only one of the options may be specified at a time.

`NEW` : This option will cause the LOADGPS script to re-set all directory links in the user directory structure to the values specified in the LOADGPS script. This option should not be used if the user is currently running the BPE because programs and scripts using links that are re-set by this option may crash.

THIS MUST BE DONE AFTER DIRECTORY VARIABLES ARE CHANGED IN LOADGPS!

`RUN_SCRIPT` : If this option is used, then LOADGPS will run the script named by the next argument. The script must be in the user's work area (`$U/WORK`).

Example: `LOADGPS RUN_SCRIPT START_PCF`

The above example will run the script `START_PCF` in the `$U/WORK` area after the LOADGPS script has finished.

VMS and DOS Version:

The command file `LOADGPS` is always run *without* command line arguments.

### 22.2.3 Getting Around

Once the LOADGPS script has been run, the user can use the directory variables that are defined in the script to navigate quickly through the directory trees. For example, to get to the work area, you can simply type:

```
erde[jjohnson]:V42-$ cd $U/WORK    (UNIX)
$   set default U:[WORK]    (VMS)
```

In the next sections we will not always include both the UNIX *and* the VMS directory and file names. VMS users should, e.g., replace `$U/WORK` with `U:[WORK]`, `$P/CAMP/OUT` with `P:[CAMP.OUT]`, `$T/AUTO_TMP` with `T:[AUTO_TMP]`, etc. DOS users will use `U:\WORK`, `P:\CAMP\OUT`, and `T:\AUTO_TMP`, accordingly.

The most important directories to know about are listed below:

`$U/WORK` This is the user's main work area. This is where the Bernese menu system will write temporary files when the menu system is run interactively. Also, this is the directory where scripts that start up the BPE are normally kept. This directory is usually the starting point for all BPE and Bernese runs.

| | |
|---|---|
| `$P/CAMP` | This directory is the top of a directory tree that contains all campaign-specific data of the campaign named `CAMP`. This includes raw and processed data, orbit data, and station information. The `P` variable (logical on VMS) is used here as an example, it could be any valid campaign variable set in the `LOADGPS` script. |
| `$P/CAMP/OUT` | One of the directories under the campaign directory tree is the `OUT` directory which holds all the output from Bernese and BPE programs. |
| `$T/AUTO_TMP` | The BPE creates a number of temporary files in this directory. It also stores log files (e.g., from remote shells) in this directory. |
| `$T/AUTO0001` | Under the `T` directory tree, there are a number of temporary directory structures that are created to run BPE scripts in. The names of these directories are `AUTOnnnn` where `nnnn` is a number. The BPE is able to run more than one script at a time and there will be at least one of these directories for each script the BPE is running. If there are no available directories when the BPE needs to run a script, then it will create a new one with a number one higher than the highest numbered `AUTO` directory. |

## 22.2.4 The CPU File (PCFCTL.CPU)

The BPE is capable of running on a single host with one CPU, or on several hosts with multiple CPUs. In order to start jobs, the BPE must know which hosts it can use to run jobs on and how many jobs it can run on each host. This is all controlled by the file `PCFCTL.CPU` which is located in the `$U/WORK` directory. When the BPE is first installed, an example `PCFCTL.CPU` file is copied into `$U/WORK`. For example:

```
# Process control CPU information
CPU      TRUE_NAME                      SPEED    NCPU SF   IDLE MAXJ JOBS
8******* 30**************************** 8******* 4*** 4*** 4*** 4*** 4***
CPU1     sakic1                         FAST        4  100  100    2    0
CPUQIF   sakic2                         FAST        4  100  100    2    0
```

Before you can start the BPE, you must edit the `PCFCTL.CPU` file so that it contains a valid CPU. Notice that the same CPU may be used twice. The fields of the `PCFCTL.CPU` file are:

| | |
|---|---|
| `CPU` ... | This is the 'nickname' of the CPU that may be used in Process Control Files (PCF) to reference the CPU. The limit is 8 characters. |
| `TRUE_NAME` ... | This is the name of the host that is to be used in the `rsh` resp. `remsh` command. It must be a valid host name and can be up to 30 characters long. |
| `SPEED` ... | Valid entries in this columns are "FAST", "SLOW", and "" (nothing). These keywords may be used in the Process Control Files (PCF) to reference a CPU with a special characteristic. The field width is 8. |
| `NCPU` ... | The number of CPUs in the host. This value is not used by the BPE at this time and is reserved for future use. The field width is 4. |
| `SF` ... | Reserved for future use, place the value 100 here. The field width is 4. |
| `IDLE` ... | Reserved for future use, place the value 100 here. The field width is 4. |

| | | |
|---|---|---|
| `MAXJ` | ... | The maximum number of jobs the BPE is allowed to run on the CPU at a time. If the host is a single CPU computer, this number should be set to 2 or 3. If the host is a multi-CPU computer, then this number can be set higher. The field width is 4. |
| `JOBS` | ... | This is the number of jobs that the BPE is currently running on the host. This value is maintained by the BPE and should be set to zero before starting any BPE jobs. The field width is 4. |

The `PCFCTL.CPU` file is used by the BPE during run time in order to keep track of the number of running jobs on each host. This means that the `PCFCTL.CPU` should not be edited while the BPE is running. If the JOBS field in this file is not zero when the BPE is started — this can happen when a BPE was finished by an error — the number of possible BPE-jobs per CPU will decrease. In the worst case the number in JOBS is equal to MAXJ when the BPE is started. In this case, the BPE will not start any jobs at all because the maximum number of running jobs has already been reached on all hosts. There is a script named `CLEAN` in the directory `$X/EXE` that can be used to make sure the CPU file is cleared before starting the BPE:

```
erde[jjohnson]:V42-$ . CLEAN     (UNIX)
         $    @X:[EXE]CLEAN     (VMS)
```

The clean script should NOT be run if the BPE is running.

## 22.2.5 System Administration Notes

The BPE can be set up to run on a simple stand alone system or on a complicated network of computers. In the case of a stand alone system, there are only a few system administration details that are important:

<u>UNIX Version:</u>

- The user must have permission to execute a remote shell (`rsh` on most UNIX systems, `remsh` on HP-UNIX). To test if this is possible, the user can type the simple command “`rsh hostname ls`”. If the user gets a response such as “`permission denied`”, then he may need to consult his system administrator to enable remote shell capabilities. A common solution to this problem is to have a `.rhosts` file in the user’s home directory that grants himself permission to execute remote shells. If your system names are, e.g., “`sakic1`” and “`sakic2`” and your login on both machines is “`jjohnson`”, then the two entries in the .rhosts file may look like this:

```
sakic1 jjohnson
sakic2 jjohnson
```

- All required system environment variables must be set when the remote shell command is executed. Since the remote shell command is non-interactive, a different set of start up scripts may be started than when the user logs in with an interactive account. This can be confusing since a command may work for a user when the test is done in a login shell, but fails when the BPE issues the command from a script in a remote shell. A common problem is that the path to a command will not be found in a remote shell. Depending on what shell is actually used by the remote shell command you may have to define all start up variables in one of the login files (e.g., `.profile`, `.login`, `.cshrc`, ...). Another way to fix this problem is to place all start up variables required for the BPE into a file named `.profile.bernese` such as:

```
#
# setup command required for bernese
#
path=${path}:/u/aiub/BERN42/GPS42/EXE
#
DOT_PROFILE_BERNESE=TRUE
export DOT_PROFILE_BERNESE
```

and then add the following to the beginning of the LOADGPS script:

```
#
# make sure .profile.bernese has been executed
#
if [ "${DOT_PROFILE_BERNESE}" != "TRUE" ]
then
    . .profile.bernese
fi
```

The BPE has the ability to execute scripts on many different computers connected together by a network. There are some important restrictions for this configuration to consider:

UNIX Version:

- A variable $LOADGPS is defined in the LOADGPS script. It must contain the full path information to the LOADGPS script. This variable will be used to start the LOADGPS script on all hosts. Therefore the location of the LOADGPS script must be identical on all hosts that are to be utilized. If the script LOADGPS is located at "/home/jjohnson" the variable must be

  ```
  set LOADGPS = /home/jjohnson/LOADGPS
  ```

  and the LOADGPS script must be available on all hosts at the same location "/home/jjohnson".

- There must be at least one common directory structure that is mounted on each of the hosts to be utilized to store campaign data (i.e. `$P` and `$Q`), temporary data (`$T`), and the user's work area (`$U`). The Bernese program and executable area (`$C`) may be in a directory local to each host if desired in order to minimize network traffic. On some systems, the exact path to mounted directories may be different on each system, which means that variables set in the LOADGPS may be host-specific. In the case where the user has the same home directory on all hosts, it is suggested that a separate LOADGPS file be maintained for each host and given the name LOADGPS.host. Then the LOADGPS script should be replaced with a simple script such as:

```
#!/bin/sh
#
# find out which host we are on
#
host=`/bin/hostname`
#
# execute the proper LOADGPS
#
if [ -f LOADGPS.${host} ]
then
    . LOADGPS.${host}
else
    echo LOADGPS.${host} not found
fi
```

- The Bernese and BPE programs may be compiled using shared libraries. This means that when a BPE program is executed, it must have access to these shared libraries. It is important to make sure that shared libraries are installed on all computers that are to run GPS programs. If you get errors when trying to run the programs that indicate that shared libraries cannot be found, you should consult your system administrator in order to correct the problem.

VMS Version:

The machines to be used by the BPE have to form a VMS cluster. The disks containing the directories for the campaign data (e.g., logical `P:`), the temporary area (`T:`), the user work area (`U:`), and the source code area (`C:`) have to be accessible from each of the machines.

## 22.3 Directory Structure

There are 4 directory structures in the Bernese/BPE system. Each of these directory structures can be located on different disks and the location of these directories are defined in the LOADGPS script. Three of the structures are fixed and generally do not change. These are the user's area (`$U`), program area (`$C`), and the temporary directory (`$T`). The fourth directory type is the campaign directory structure (`$P`, etc.) — see also Chapters 23 and 24. In addition to the directory structures themselves, there are two other items to be aware of. These are the directory variables (`$U`, `$T`, `$C`, `$P`, etc.) and directory links (links are not necessary and not present for VMS). The directory variables, which are defined in the LOADGPS script, are used by the user and BPE scripts to navigate to the base of each directory tree. On UNIX systems the links are found in the `WORK` directories under `$U` and the `$T/AUTOnnnn` directories. Directory links are used by the Bernese programs to find the location of directory structures. All of the Bernese programs expect to be started in a `WORK` directory that has these links set.

### 22.3.1 Directory Trees

An overview of the various directory structures used by the BPE is given in Figure 23.1 and Figure 24.1. You should keep in mind that *on UNIX systems* there are links defined (e.g., named `U:`) for each of these directory structures in the work area `$U/WORK` (pointing, e.g., at the directory `$U`).

Notice that the directory link `U:` is recursive, i.e. that it points to an ancestor directory. This can cause problems if the user tries to copy the directory structure `$U` with the recursive option, for example:

```
erde[jjohnson]:V42-$ cd $U
erde[jjohnson]:V42-$ cp -r * ../BACKUP
```

will eventually fill up the disk. This is because the copy command follows links and will forever create directories in the following way (assuming `$U` is `/home/user/GPSUSER`): `/home/user/BACKUP/WORK/WORK/WORK/WORK/WORK ...`

The `tar` command is safe, however, because it does not follow links and will instead copy the link information to the tar file. Thus, to safely back up the `$U` area, the user could do the following:

```
erde[jjohnson]:V42-$ cd $U
erde[jjohnson]:V42-$ tar cvf ../GPSUSER.BACKUP .
```

The `$T` directory structure deserves some additional comments. Notice that the `$T/AUTO0001` directory may have the `U:` notation in the BPE environment. This can be confusing because the `$U` directory carries the same notation.

The `AUTOnnnn` directories are used by the BPE to run scripts. Each script that the BPE runs requires its own work area because the Bernese programs expect to write temporary files in the `$U` area (`$U/INP`, `$U/PAN`, and `$U/WORK`). It also expects to find the input parameters in panel files in `$U/PAN`. So, in order to run more than one script at a time, it is necessary to have different directory structures for each script. When the BPE runs a script, it finds an available `AUTOnnnn` directory structure to run in and then re-sets the `$U` variable to that `AUTOnnnn` area. All the panel files in the `AUTOnnnn/PAN` area are deleted and new panel files for the specific script to be run are copied over. On UNIX systems the BPE also sets the links in the `AUTOnnnn/WORK` area. Thus, as far as the Bernese programs know, they are running in the standard `$U/WORK` area. So, the `$U` and `U:` directories are specific to each work area. The `U:` link in `AUTO0001/WORK` points to `AUTO0001`, `U:` in `AUTO0002/WORK` points to `AUTO0002` and so on. The user will never see the changing of `$U` to one of the temporary directories as this is all handled internally by the BPE. The `$U` variable defined at the Bourne shell prompt will always be the same.

The reason for this `AUTOnnnn` structure and the different `$U` variables is that each BPE script can have a user (work) area to run Bernese programs without overwriting input/output files from other scripts.

## 22.4 The Process Control Script (PCS)

The Process Control Script, or PCS, is the outer most script of the BPE. The script lives in the `$X/EXE` directory, which will be in the user's path after the LOADGPS script is run. The PCS script expects to be run from the user's `$U/WORK` area and takes as its basic parameters the name of a Process Control File (PCF), the session to run, and the year of the session. In general the user will start the PCS using the Menu 6.4.1 which allows him to specify all the parameters passed to the PCS script in input panels (see Section 22.7.5).

### 22.4.1 How the PCS Works

The main function of the PCS is to execute the scripts contained in a Process Control File (PCF). The PCF, which is covered in more detail in a later section (see Section 22.5), is a file containing a list of scripts that are to be run. In addition, the PCF contains information about what input options to use for these scripts and the order in which to run the scripts. Most importantly, the PCF file tells the PCS script which scripts must be completed before starting any given script. The steps the PCS takes are outlined below:

**(1)** Validate input parameters. The PCS will check to make sure that:

- Session and year are defined and all parameters are valid.
- The PCF file is in the `$U/PCF` directory.
- All scripts in the PCF file are in `$U/SCRIPT`.
- All options directories in the PCF file are in `$U/OPT`.

If any of the above tests fail, the PCS will display an error message and terminate.

**(2)** Clear all protocol and standard-output log files. The PCS uses protocol files in order to track the status of the scripts in the PCF that are to be run. It uses these files to tell if they have been started and to tell when a script has finished. These protocol files are stored in `$P/CAMP/OUT`. All standard output from the scripts running in the BPE are written to log files. These log files are located in `$T/AUTO_TMP`. They have the same names as the protocol files.

**(3)** Check for errors in the protocol files. The PCS will scan all the protocol files generated for errors. If an error is found, the PCS will print an error message and terminate. Any remote jobs that have been started and have not finished will still run, but new scripts will not be started.

**(4)** See if there is a script that can be run. The PCS goes through each script in the PCF file and checks to see if it has been started. If it has not been started, then it checks to see if the scripts that it needs to wait for have finished. If these scripts have finished, then the PCS will start the script and go to step 6. If the script has been started, or if the scripts it needs to wait for have not finished yet, then the PCS will move on to the next script until it either finds a script that is ready to run — in which case we go to step 6 — or there are no more scripts to check. If the PCS cannot find a script to run and not all the scripts have finished, then it goes to step 5. If all scripts have finished, then the PCS terminates.

**(5)** Sleep and then go back to step 3.

**(6)** Check the `PCFCTL.CPU` file (located in `$U/WORK`) for a computer to run the script on. The `PCFCTL.CPU` file specifies which computers the BPE is allowed to run scripts on and how many scripts at a time it can run on these computers. The PCS will look for an available computer and if it does not find one, it will go to step 5 (sleep a while, then check again). If a computer is found, the PCS increments the count for the number of jobs running on the computer in the `PCFCTL.CPU` file and then continues with the next step.

**(7)** Run a script. The PCS comes to this step if it has found a script it can run (step 4) and there is an available computer to run the script on (step 6). The PCS will generate a header and tail script and copy them into the `$T/AUTO_TMP` directory. The header and tail scripts are attached to the beginning and end of the script to run. These scripts take care of the book-keeping required by the BPE, such as writing messages to the protocol file, and set up variables that define the parameters (such as the year and session) that are required by the script. The names of these files will be `HDRnnnnn` and `TL_nnnnn`, where `nnnnn` is a unique ID number that the PCS automatically generates. The PCS then generates a script that will be executed by an `rsh` command (UNIX) or a `submit` command (VMS) that will start the script in the PCF file. This file will be named `PCFnnnnn.RSH` (`PCFnnnnn.COM` on VMS) and will be copied over to the `$T/AUTO_TMP` directory. Finally, the PCS will start up the script by issuing an `rsh/submit` command. The `rsh/submit` command will run the LOADGPS script with special options that specify that the script `PCFnnnnn.RSH/PCFnnnnn.COM` should be run after the Bernese environment has been loaded. After the `rsh/submit` command has been issued, the PCS will go back to step 3.

The PCS process finishes once all scripts have been started and have finished, or if an error is found in one of the protocol files.

The error handling in the PCS requires that the name of the error message file is `U:/WORK/ERROR.MSG`. This name is hardwired in the PCS but not in the menu system of the *Bernese GPS Software*. Please check the data panel Panel 0.3–1 in all `U:/OPT` option directories. The last line in this panel must be:

```
 0.3-1                     DEFAULTS: GENERAL DATASET NAMES
.
.
 Auxiliary Files (Scratch Files) > U:/WORK                 <    > SCR <
 Error Message File (Full Name): > U:/WORK/ERROR.MSG            <
```

If the PCS detects an error in one of the scripts it will stop and not start new jobs. However, all scripts previously started will continue their work on the remote hosts. They will update their protocol files, and if you restart the PCS before all remote jobs have been terminated (or killed) the restarted PCS will be seriously confused. Therefore, make sure that no BPE jobs are running when you restart the PCS (using appropriate commands of your system). Issue the `CLEAN` command before restarting the BPE (see below).

### 22.4.2 Format of Protocol File

There are two files that will be generated for each script that is started. The log file containing the standard output of the script will be generated in the `$T/AUTO_TMP` directory. The protocol file will be in the `$P/CAMP/OUT` directory and is used to report the status of the script (errors, completion status, etc.). The name of these two files is identical and has the following format:

`TTYYSSSS.PID` or `TTYYSSSS.PID_SUB`

Where:

| | |
|---|---|
| `TT` ... | Task id, 2 characters, has to be used only, if more than one BPE runs using the same campaign and session(s) (default task id is `00`). |
| `YY` ... | Two digit year. |

| | | |
|---|---|---|
| `SSSS` | ... | The session number. |
| `PID` | ... | The three digit process id of the script (defined in the PCF file). |
| `SUB` | ... | The sub process id of the script. Parallel scripts can be run a number of times. Each separate run of a parallel script is given a sub process id, starting with 001 and incrementing by 1 each subsequent run. The `_SUB` is only present for parallel scripts. |

To see if a script has finished, the PCS will look through the protocol file in `$P/CAMP/OUT`. Here is an example protocol file from the campaign `OUT` directory:

```
erde[jjohnson]:V42-\$ cd $P/NOAA_FSL/OUT
erde[jjohnson]:V42-\$ cat 00950170.001
PROTOCOL FILE FOR BPE SCRIPT
----------------------------

SCRIPT NAME         : FTP_RNX
YEAR                : 95
SESSION             : 0170
CAMPAIGN            : NOAA_FSL
CAMPAIGN PATH       : P:/
OPTION DIRECTORY    : FTP_NOAA
PROCESS ID          : 001
SUB PROCESS ID      : 001
CPU                 : CPU1
PATH TO WORK AREA   : /u/erde1/rocken/GPSDATA/AUTO0001/

 DATE      TIME     STA PROGRAM  MESSAGE
----------------------------------------
 31-MAR-95 21:07:21 MSG FTP_RNX  PROCESS STARTED
 31-MAR-95 21:07:26 MSG FTPRNX   PROGRAM STARTED
 31-MAR-95 21:08:21 MSG FTPRNX   PROGRAM ENDED
 31-MAR-95 21:08:23 MSG FTP_RNX  PROCESS ENDED
---------------------------------------
```

The protocol file first displays the value of the variables that were set in the header script (`HDRnnnnn`). Below is a table of the items displayed:

| | | |
|---|---|---|
| `SCRIPT NAME` | ... | This is the name of the script that was executed. |
| `YEAR` | ... | The two digit year. |
| `SESSION` | ... | The four character session. |
| `CAMPAIGN` | ... | Name of the campaign. |
| `CAMPAIGN PATH` | ... | The path to the campaign. For UNIX systems this shows the link name. For the above example, the campaign path is given as `P:/` which corresponds to the `$P` variable set in the LOADGPS script. |
| `OPTION DIRECTORY` | ... | This is the name of the option directory used for input panel files. The option directories are located in the `$U/OPT` area. |
| `PROCESS ID` | ... | This is the 3 digit process id. The process id for a script is assigned in the PCF file. |
| `SUB PROCESS ID` | ... | The sub process id only has a meaning for parallel scripts. For parallel scripts, this indicates which run of the script the protocol file is for. For example, if a parallel script is split into 10 separate runs, then there will be 10 protocol files with sub process ids from 001 to 010. |

| | |
|---|---|
| `CPU` . . . | This is the name of the computer (or the batch queue on VMS) that the script was run on. This is the abbreviated name. The full name of the computer (batch queue) can be found by looking at the `PCFCTL.CPU` file in `$U/WORK`. |
| `PATH TO WORK AREA` . . . | This shows which `AUTOnnnn` directory was used as a temporary work area for the script. |

Below the display of the variables, a chronology of which programs were run is shown. There may also be some additional warnings and/or error messages. The first two columns show the date and time the message was added to the protocol file. The third column shows the type of message. The possibilities are `MSG` (a simple message), `WAR` (a warning), and `ERR` (a fatal error). The fourth column shows the program or script the message comes from and the final column shows the message. The first and last message in a protocol file will be from the script being run, the first being `PROCESS STARTED` and the final `PROCESS ENDED`. The PCS looks for the message `PROCESS ENDED` to determine if a script has finished.

When the PCS runs a script, it pipes all the output from the `rsh` command (UNIX) or from the submitted job (VMS) in the log file in `$T/AUTO_TMP`. For example, here is the log file `00950730.001` on a UNIX system:

```
HEADER: copying PAN files from /u3/jjohnson/GPSUSER/OPT/FTP_BERN
U:/WORK/FTPORB.COM started at : Wed Mar 29 12:10:39 MST 1995
will get cod07922.eph
Interactive mode off.
Local directory now /u3/jjohnson/GPSDATA/JJ_TEST/ORB
start
cod07922.eph
U:/WORK/FTPORB.COM   ended at : Wed Mar 29 12:10:54 MST 1995
```

The log file will contain any output from the script that is not explicitly directed to the protocol file. Most importantly, if a shell/command file were to have a hard crash, such as from a program crashing with a segmentation fault, then the error message would appear in the log file rather than in the protocol file. This is because scripts have to explicitly write messages to the protocol file, and if the script itself crashes, there is no way for it to send a message to the protocol file. An extended log file is generated if the PCS is running in the debug mode (which may be specified in Panel 6.4.1–1.4).

## 22.5 The Process Control File (PCF)

The Process Control File (PCF) defines which scripts the Process Control Script (PCS) should run. In addition to listing the scripts to run, the PCF defines which scripts must wait for other scripts before they can be run and defines parameters that are to be passed to the scripts.

### 22.5.1 Linear PCFs

The basic PCF is linear. This means that each listed script is executed only once. It is still possible that processing will happen in parallel by having two different scripts running at the same time.

Each script, however, is only run one time. For example, it may be possible to run an ftp script to get precise orbits at the same time a script is running to get IGS RINEX data.

A very simple PCF file is shown below (`GET_ORB.PCF`):

```
#
# Procedure Control File (PCF)
# All comment lines start with a #
# Comments:
#   GET_ORB.PCF -> a PCF file that will retrieve COD orbits from CODE
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
001 FTP_ORB  FTP_BERN          any      1
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
#
# PCF Variables
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
#
# That's it
#
```

This is a simple PCF file that contains only one script to run. This PCF file will run a script to retrieve CODE precise orbits from the IGS Analysis Center CODE. Any line that starts with a # is a comment and comments can appear at any location in the PCF. The first two non-comment lines define the fields for the first section of the PCF. These two lines must be present. After the two header lines, the scripts that are to be run are listed. A description of the different fields is given below:

`PID` ... Each script in the PCF is assigned a unique Process Identification number. It is up to the author of the script to assign PID numbers to the script and the only restriction is that they must be unique 3 digit numbers and they must increase from script to script (this is required, but not checked in the source code!). For a short PCF, usually, the first script is given the process id 001, which is incremented by one for each subsequent script. For longer PCF's we suggest to make sections with PIDs following each other. There may be unused numbers between sections in order to make it easier to add a new script into a section of the PCF file in future.

`SCRIPT` ... This is the name of the script to run. The script must be located in `$U/SCRIPT`.

`OPT_DIR` ... This is the name of the option directory that will be used to get panel files from. The option directories must be in `$U/OPT`.

`CAMPAIGN` ... This is the name of the campaign to process. Normally, you will leave this field blank to be able to process any campaign using this PCF. If you specify a campaign name here it will take precedence over the campaign selected in Menu 6.4.1 when starting the BPE run.

`CPU` . . . This is the name of the computer/queue that the script should be run on. This can be the 8 character name specified in the `PCFCTL.CPU` file (found in `$U/WORK`) or `ANY`. If the keyword `ANY` is used, then the script will be run on the first available computer found. You may also specify `FAST` or `SLOW`. In this case a computer with the keyword `FAST` or `SLOW` respectively in the `PCFCTL.CPU` in column `SPEED`, will be selected. Still another possibility is `IDLE` which means that a computer/queue is chosen where no other BPE script is running yet. The computer/queue given in the PCF file takes precedence over the `CPU` selection in Menu 6.4.1 .

`P` . . . This is the priority of the script. The number can range from 1 to 9. On UNIX systems a priority of 1 will cause the script to be run with the highest possible priority and a value of 9 will cause the script to be run with the lowest possible priority (`nice` command). On VMS different batch queues have to be selected for jobs that should run with different priorities.

`WAIT FOR` . . . This field specifies which scripts must have finished before the script can be started. Up to nine scripts to be waited for can be specified.

The next section of the PCF file starts with two more header lines. If these additional parameters are not required for a script, then this section can be left empty. If additional parameters are required, then they will be listed in this section. The meaning of the different fields is listed below:

`PID` . . . This is the PID of the script that the parameters belong to. This `PID` must be listed in the first section of the PCF file.

`USER` . . . This field is reserved for future use and is not used at this time.

`PASSWORD` . . . This field is reserved for future use and is not used at this time.

`PARAMn` . . . Up to six parameters at a time can be specified to each script. If, as an example, the value of `PARAM1` is "get_erp", a corresponding variable `$PARAM1` is defined in the corresponding script and has the value "get_erp". The same script may in this way be used for slightly different tasks in different PCF scripts (e.g., get orbits with Earth orientation parameters or get orbits only). There are some parameters that have a special meaning for all scripts and are listed below:

> `SKIP` . . . If the keyword `SKIP` is passed to the script as `PARAM1`, then the script will not execute. It will write a message to its protocol file that it was skipped and then exit.
>
> `PARALLEL` . . . This keyword is used for parallel scripts, which will be covered in detail in Section 22.5.2.
>
> `$<extension>` . . . This is used to generate unique file names used by parallel scripts. The PCS will automatically expand the `$` into a unique number and add the extension to it. For example, `$tmp1` would be expanded to a file name `019283.tmp1` (the 019283 will be different for different runs).

The third and final section of the PCF defines any special variables for the PCF. If the PCS is started from the Bernese menu system then a special panel will be created and presented to the user based on this information. The key words are:

| | |
|---|---|
| `VARIABLE` ... | This is the name of the variable that will be set in all scripts run by the PCF. These variables must start with the two characters `V_` and may then have up to six more characters. There are 8 special variables that can be set that correspond to variables in Panel 1.5.1 (see also the corresponding help panel and Chapter 3.8). |

`V_O` ... Sets the `$O` variable.
`V_U` ... Sets the `$U` variable.
`V_V` ... Sets the `$V` variable.
`V_W` ... Sets the `$W` variable.
`V_X` ... Sets the `$X` variable.
`V_Z` ... Sets the `$Z` variable.
`V_PLUS`, `V_MINUS` ... These two variables are used together to specify a range in the following Panel 1.5.1 variables:

| | |
|---|---|
| `$JJ1` ... | 4-character year parameter `4-YEAR1` |
| `$SS1` ... | 4-character session parameter `4-SESS1` |
| `$GW1` ... | 4-character session parameter `4-WEEK1` |
| `$JRD1` ... | 5-character session parameter `5-SESS1` |
| `$GDY1` ... | 5-character session parameter `5-WDAY1` |
| `$JRSS1` ... | 6-character session parameter `6-SESS1` |

The six variables `V_x`, if defined in the PCF file, will be available in all scripts and the `$x` in all panels. For example, if the user were to set `V_O` to the value `XI`, then `$O` in a Bernese panel would be replaced with `XI` and the variable `$V_O` (in a UNIX script) resp. the symbol `V_O` (in a VMS command file) would be `XI`. The `V_PLUS` and `V_MINUS` variables will be concatenated together and appended to the session variables listed above.

| | |
|---|---|
| `DESCRIPTION` ... | This field can be up to 40 characters and defines the description the variable will be given. |
| `DEFAULT` ... | This defines the default value the variable will have. |
| `LENGTH` ... | This tells the Bernese menu system how many characters to allow for the variable. The maximum is 16. |

NOTE:

The length of the eight special `V_x` variables above must be 2, because they are declared as 2-character variables in the Bernese menu system. All other `V_x`-variables are available within the script as parameters (e.g., `V_TEST` as `$TEST` for UNIX, `%TEST%` for DOS, and TEST for VMS) but cannot be used within the panels as $-variables (see below).

EXAMPLE:

The following excerpt from a PCF would generate the following Panel 6.4.1–1.3 when starting the BPE with Menu 6.4.1:

```
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_O      EUROCLUS ORBIT FILE NAME                 R3                2
V_X      AMBIGUITY FREE  RESULTS                  EG                2
V_Z      AMBIGUITY FIXED RESULTS                  EQ                2
V_U      U ....                                   UU                2
V_V      V ....                                   VV                2
V_W      W ....                                   WW                2
V_PLUS   PLUS  DAYS                               +0                2
V_MINUS  MINUS DAYS                               -0                2
```

```
6.4.1-1.3 |          BPE SESSION PROCESSING:  SPECIAL PARAMETERS

  Special Parameter Setting:
    EUROCLUS ORBIT FILE NAME                "O"        > R3 <
    AMBIGUITY FREE  RESULTS                 "X"        > EG <
    AMBIGUITY FIXED RESULTS                 "Z"        > EQ <
    U ....                                  "U"        > UU <
    V ....                                  "V"        > VV <
    W ....                                  "W"        > WW <
    PLUS  DAYS                              "PLUS"     > +0 <
    MINUS DAYS                              "MINUS"    > -0 <

  Control Process:
    SLEEP TIME    > 0      <    (in seconds, 0: default value used)
```

### 22.5.2 Parallel PCFs

Parallel PCFs are slightly more complicated than linear PCF files. They have the advantage, however, that they can split up a single task into multiple tasks, each of which can be executed on a separate computer. For example, to run the single point positioning using the code data (CODSPP) on all observation files, a linear PCF script could be written that would simply loop over all code observation files for the specified year and session and run the CODSPP program once for all these files. The program would process one code observation file after another in a linear fashion. But the computations of the different stations is independent. Therefore, with parallel scripts, it is possible to divide up the code observation files into groups (also called “clusters”) and have different computers work on different groups of these files. Improvements in speed can be achieved even by running two groups of code observation files on the same computer: while one group performs disk I/O tasks, the other can use the CPU and vice versa. The only requirement for a parallelization is that the analysis program can handle these clusters independently from each other. More examples for possible parallel processing tasks are the transfer of the data from RINEX into Bernese format (RXOBV3), the screening of the single difference phase files (MAUPRP), or the ambiguity resolution (GPSEST).

Parallel scripts require two scripts that work together in conjunction. The first script is the file script and its job is to either (a) create a list of files (e.g., observation files) to be processed one-by-one in parallel or (b) to divide up a list of files that are to be processed in groups. The second script is the script that actually does the work and will be executed once for each individual file in the file list in case (a) and once for each group of files generated by the file script in case (b).

Below is an excerpt from the example PCF file given in `$X/PCF` that shows the parallel script files for running the program GPSEST, using the QIF ambiguity resolution strategy on each individual baseline of a session:

```
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
:
018 GPSQIFAP EURO_QIF          any      1 017
019 GPSQIF_P EURO_QIF          any      1 018
:
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
018                       $tmp1
019                       PARALLEL $tmp1
:
```

The first section defines the option directories to use for the scripts and the order in which they must be run. In this case, the GPSQIFAP script must wait until script 017 has finished before it can start. Since the GPSQIFAP script only creates a list of baselines to be processed in parallel, it does not need any input options and instead of the option directory EURO_QIF an (almost) empty option directory (usually called NO_OPTS) could be specified. This option directory NO_OPTS is a directory only containing some panels that have to be present in any option directory (e.g. the default panels DAT0xxxx.PAN, the panel DAT151__.PAN, etc.) in the $U/OPT area. The GPSQIF_P script is the one that will do the work and the name of an option directory is passed to it that contains panel and option files that are needed in order to run GPSEST.

The bottom section of the PCF defines the parameters that are passed into the different scripts. The parameter $tmp1 is passed into GPSQIFAP as PARAM1. The $tmp1 variable is automatically expanded by the PCS into a file name that has a unique base name and the extension .tmp1 (01298.tmp1 for example). The GPSQIFAP script will write into the file $tmp1 one line for each baseline to be processed. The lines that GPSQIFAP writes into the $tmp1 file will be used by the PCS to generate parameters for the parallel script GPSQIF_P.

The keyword PARALLEL is specified for the GPSQIF_P script followed by the file name $tmp1. When the PCS detects the PARALLEL keyword, it will read the file specified in PARAM2, which is $tmp1 in this case, and read and remove the first line from this file. This line is then used to generate the PARAMn parameters that are actually passed into the GPSQIF_P script: the first item in the line will be passed to the script as PARAM1, the second as PARAM2, etc. The PCS keeps track of how many times it has executed the GPSQIF_P script and passes this number to the script as the subprocess id. This process of reading and removing the first line from the $tmp1 file continues until the $tmp1 file is empty, in which case the PCS will mark the script as being started. Once all of the scripts started by the PCS have finished, the script will be marked as done, and any scripts that have been waiting for the GPSQIF_P script to finish will then be able to run.

The protocol and log files for the GPSQIF_P will have an additional number added to the normal extension. For example, the first time the GPSQIF_P script is run, its protocol file name will be (assuming we are running session 1650 of the year 1996) 00961650.019_001 (in $P/CAMP/OUT) and 00961650.019_001 for the log file (in $T/AUTO_TMP).

If you want to process files in N groups (e.g., running program CODSPP with groups of files, i.e. running it once for each group of code observation files) the preparatory script has to generate N files (one for each group, let's call them group files) containing each a list of the files belonging to this group and one file (the $tmp1 file) containing the names of the N group files (one per line). The PCS then passes the name of the group file to the script to be run using the parameter PARAM1. The parallel script (the script that runs, e.g., CODSPP) may then copy the group file to the "SELECTED" file (e.g., $U/WORK/CDZFILE.SEL for CODSPP), the file where the menu system saves the list of files selected the last time by the user (see Section 3.4.3). By setting the option “SELECTED” in the panel, where the names of the files to be processed have to be selected (e.g., `DAT42___.PAN` for CODSPP), exactly the files contained in the group file will be processed.

A detailed description of the way how to organize the parallel processing in the scripts will be given in Section 22.6.5.

## 22.6 Running a PCS

We will now go through the process of running a simple PCS.

### 22.6.1 An Example PCF

The PCF file that we will use for this example is `T_PCF.PCF`:

```
#
# Process Control File
# Comments:
# Any line that starts with a # is a comment line
#
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT
*** ******** ******** ******** ******** * *** *** *** *** *** *** *** *** ***
001 T_SCRIPT NO_OPTS           any      1
002 T_SCRIPT NO_OPTS           any      1 001
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5
*** ************ ******** ******** ******** ******** ******** ********
#
# PCF Variables
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_0      TEST SETTING V_0                         HI                2
```

This PCF will run the script `T_SCRIPT` (which must be in `$U/SCRIPT`) twice. Since we have a wait PID for the second `T_SCRIPT`, it must wait for the first one to finish before it can start. If we did not have a wait PID for `PID 002`, then the BPE would start the second `T_SCRIPT` script right after the first one was started.

The option directory for the script `T_SCRIPT` has been selected to be `NO_OPTS`. In this option directory we only need to have a few basic panels that are required by the Bernese menu system. `T_SCRIPT` does not run any Bernese programs and, in fact, will only echo some messages. At this point, we only want to show the flow of the BPE.

Here is the script `T_SCRIPT` for the UNIX case (for VMS please have a look at the scripts in `X:[USERSCPT]`):

```
#!/bin/sh
#
# T_SCRIPT
# ========
#
# An example script to show the BPE functioning
# --------------------------------------------
#
# sh script file written by bds
# ----------------------------
# functions used by shell
do_rm( ) {
    if [ "$1" ]
    then
        if test -f `echo $1 | tr ' ' '\012' | head -1`
        then
            eval rm $1
        fi
    fi
}
#
toupper( ) {
    eval $1=`echo $2 | tr '[a-z]' '[A-Z]'`
    eval export $1
}
#
seterr( ) {
    if [ $? = 0 ]
    then
        ERRSTAT=OK
    else
        ERRSTAT=ERR
    fi
}
#
# ----------------------------
# SHELL STARTS HERE
# ----------------------------
#
# SHELL VARIABLES:
# ----------------
#
# YEAR    :  Year of the session to be processed (2 digits)
# YR_4    :  Year of the session to be processed (4 digits)
# SESSION :  Session number (4 characters)
# CAMPAIGN:  Campaign name
# CAMP_PTH:  Campaign path
# CAMP_DRV:  Drive letter for campaign (i.e. P)
# OPT_DIR :  Directory for panels
# PID     :  Process identification number (3 digits)
# SUB_PID :  Subprocess id (3 digits)
# PRT_FILE:  Protocol file name including path
# SCRIPT  :  Name of script
# TASKID  :  Task id of script, usually 00
# PRIORITY:  Priority of the script
# CPU     :  CPU the script is running on
# DAYYEAR :  Julian day of the year
# DAY     :  Day of the Month
# MONTH   :  Month, 1=JAN, 12=DEC
# GPSWEEK :  GPS week
# DAYWEEK :  Day of the week, 0=SUN, 6=SAT
# V_X     :  X Variable in DAT151__.PAN
# V_O     :  O Variable in DAT151__.PAN
# V_Z     :  Z Variable in DAT151__.PAN
# V_PLUS  :  Plus variable in DAT151__.PAN
# V_MINUS :  Minus variable in DAT151__.PAN
# V_x     :  User variable
# PARAMx  :  Script specific parameter, x is 1 thru 9
# U       :  Directory path to U:
#
# See if this shell is being run from the PCS
# script, or directly for testing.  If testing,
# the header script is not passed in as %1%
#
if [ "$1" = "" ]
then
#
#   set variables for testing here
#
    TEST_START_DIR=‘pwd‘
    cd $U/WORK
    TESTING="YES"
    export TESTING
else
    TESTING="NO"
    export TESTING
    . "$1"
    seterr
fi
#
# START THE MENU SYSTEM IN NON-INTERACTIVE MODE
# -------------------------------------------
. $X/SCRIPT/BEG_MENU
seterr
#
# SET VARIABLES IN DAT151__.PAN
# ----------------------------
. $X/SCRIPT/SET_SESS
seterr
#
# T_SCRIPT BODY
# -------------
#
# This is a simple test script that does nothing.
# It starts HERE
#
echo T_SCRIPT: Starting
echo SESSION IS ${SESSION}
echo V_O IS ${V_O}
echo T_SCRIPT: Ending
#
# It ends HERE
#
# --------------
# end the script
# --------------
. $X/SCRIPT/END_MENU
seterr
if [ "$TESTING" = "YES" ]
then
    cd "$TEST_START_DIR"
else
    . $X/SCRIPT/DO_TAIL
    seterr
fi
```

At a first glance this script looks complicated. However, most of the script is a ’skeleton’ required by the BPE (all scripts have these start and end sections). What the script actually does is between the lines:

```
# It starts HERE
# It ends HERE
```

So, all this script does is echo some information (no GPS program is run). The output of the echo commands will be captured in the log files.

Scripts written for the BPE have to be standard Bourne shell scripts under UNIX, DCL command files under VMS. Notice that most of the above script consists of comments. The line

```
. "$1"     (UNIX)
$ @'P1' 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'     (VMS)
```

executes the header script that the PCS generated (the PCS will pass the name of the header script as first parameter named `$1` under UNIX, `P1` under VMS). Then two scripts (`BEG_MENU` and `SET_SESS`) are run that will put the Bernese menu system into the non-interactive mode and set the campaign and session (which are defined in the header script). The last two lines in the script put the Bernese menu system back into interactive mode (script `END_MENU`) and run the script `DO_TAIL` that does some finishing tasks, such as writing the final messages into the protocol file. The `DO_TAIL` script will also execute the tail script that is generated by the PCS.

To start the PCF from the menu system, we select Menu 6.4.1 :

```
6.4.1                    BPE: SESSION PROCESSING

   CAMPAIGN              > TST_CAMP <    (blank for selection list)

  Job Identification:
   JOB CHARACTER            >   <        (blank, or A..Z, 0..9)

 Input Files:
   PROCESS CONTROL FILE  > T_PCF    <    (blank for selection list)
```

We are using a fictitious campaign with the name `TST_CAMP` to demonstrate the menu steps. Next the following panel is displayed:

```
6.4.1-1           BPE SESSION PROCESSING: INPUT OPTIONS

  Sessions Information:
    SESSION (START)      > 1110 <
    YEAR (START)         > 1996 <
    NUMBER OF SESSIONS   > 1    <      (if negative: processing backwards)

  Task Identification:
    TASK IDENTIFICATION  > 00 <        (blank: 00)

  CPU/QUEUE Specification:
    CPU / BATCH QUEUE    > NO       <  (NO, or blank for selection list)

  Special Options:
    SPECIAL PARAMETERS   > NEW  <      (OLD.. NEW.. or ASIS)
    SKIP PROCESSES       > NO   <      (YES..  NO,  or ASIS)
    REMOTE SUBMIT        > NO   <      (YES..  NO,  or ASIS)
    DEBUGGING OPTIONS    > NO   <      (YES..  NO,  or ASIS)
```

We are thus processing the session 1110 of the year 1996. Then the Panel 6.4.1–1.3 is displayed:

```
6.4.1-1.3          BPE SESSION PROCESSING:  SPECIAL PARAMETERS

   Special Parameter Setting:
     TEST SETTING V_0                         "0"       > HI <

   Control Process:
     SLEEP TIME    > 0      <    (in seconds, 0: default value used)
```

Here is where we are prompted for the `V_0` parameter. After we accept this panel, the BPE starts to run (whether the BPE runs in the foreground or background depends on the setting of the `JOB CLASS` option in Menu 0.1). First the PCS will read the PCF file specified and check to make sure that the option directories of all scripts are present. It will also do some basic error checking to make sure that the PCF file is consistent. Then the PCS will remove any protocol files for the year and session in the campaign OUT directory (`$P/TST_CAMP/OUT` in this case) and log files in `$T/AUTO_TMP`. The PCS will then check the status of the scripts and decide if a script needs to be run. Since the PCS just started, the scripts in the PCF file will show up as not being started or finished:

```
U:/WORK/PCS.COM started at : Wed Sep  4 19:45:15 MDT 1996
 PID STATUS IN PRCNXT
PID 001 T_SCRIPT NO_OPTS  STARTED = F ENDED = F
PID 002 T_SCRIPT NO_OPTS  STARTED = F ENDED = F
 WILL RUN PID  1
```

The status of which scripts are being run is listed. The displayed process list includes the PID of each script, the option directory for the script, whether or not the script has been started and whether or not the script has finished. There are three possible flags for `STARTED`: `F` (false) means that the script has not been started, `P` (partial) means that the script has been partially started (only for parallel scripts), and `T` (true) means that the script has been started. For `ENDED`, there are only two possibilities: `F` if the script has not finished and `T` if the script has finished. The above output tells us that no script has been started (or has finished). After the BPE starts the first script, we will see the lines change to:

```
PID 001 T_SCRIPT NO_OPTS  STARTED = T ENDED = F <
PID 002 T_SCRIPT NO_OPTS  STARTED = F ENDED = F
```

Note that any script that is running (`STARTED = T ENDED = F`) will have a “<” symbol to the right to allow the user to quickly see which script is running.

Since the second script in the PCF may only be started after the first one has finished, there is nothing to do for the PCS but to wait for it to finish. Thus the PCS sleeps for 30 seconds (default value). When it wakes up, it will again check the status of the scripts.

After the PCS has finished, we can take a look at the log files which will be in `$T/AUTO_TMP`:

```
sakic[jjohnson]:V42-$ cd $T/AUTO\_TMP
sakic[jjohnson]:V42-$ ls -lt
total 6
-rw-r--r--   1 jjohnson users         847 Sep  4 19:46 PCFLOCK.NUM
-rw-r--r--   1 jjohnson users        1206 Sep  4 19:46 PCFLOCK.CPU
-rw-r--r--   1 jjohnson users         478 Sep  4 19:46 PCFLOCK.ATO
-rw-r--r--   1 jjohnson users         128 Sep  4 19:46 00961110.002
-rw-r--r--   1 jjohnson users         128 Sep  4 19:45 00961110.001
lrwxrwxrwx   1 jjohnson users          24 Aug 21 10:24 P: -> /home/jjo...
```

The files that start with `PCFLOCK` are temporary files used by the BPE. The two files that have the extension `.001` resp. `.002` are the log files and contain all the output generated by the script that would normally go to the screen. Since the `T_SCRIPT` echos information, we will find what the script echoed in these files. This is very useful for the debugging of scripts. If the script itself were to have an error, then this error would appear in the log file and not in the protocol output file (see below).

Here is the content of `00961110.002`:

```
sakic[jjohnson]:V42-$ cat 00961110.002
HEADER: copying PAN files from /home/jjohnson/GPSUSER/OPT/NO_OPTS
T_SCRIPT: Starting
SESSION IS 1110
V_O IS HI
T_SCRIPT: Ending
```

If we go to the campaign `OUT` directory, we will see the protocol output files:

```
sakic[jjohnson]:V42-$ cd $P/TST_CAMP/OUT
sakic[jjohnson]:V42-$ ls -lt
total 2
-rw-r--r--   1 jjohnson users         584 Sep  4 19:46 00961110.002
-rw-r--r--   1 jjohnson users         584 Sep  4 19:45 00961110.001
```

These files contain information that is written by the basic BPE scripts:

```
sakic[jjohnson]:V42-$ cat 00961110.002
PROTOCOL FILE FOR BPE SCRIPT
----------------------------

SCRIPT NAME         : T_SCRIPT
YEAR                : 96
SESSION             : 1110
CAMPAIGN            : TST_CAMP
CAMPAIGN PATH       : P:/
OPTION DIRECTORY    : NO_OPTS
PROCESS ID          : 002
SUB PROCESS ID      : 002
CPU                 : CPU1
PATH TO WORK AREA   : /home/jjohnson/GPSTEMP/AUTO0001

 DATE      TIME     STA PROGRAM  MESSAGE
----------------------------------------
 04-SEP-96 19:46:00 MSG T_SCRIPT PROCESS STARTED
 04-SEP-96 19:46:03 MSG T_SCRIPT PROCESS ENDED
----------------------------------------
sakic[jjohnson]:V42-$
```

From the time tags of the start and end messages, we see that the script only took 3 seconds. When the last line of the protocol file is written (the `PROCESS ENDED` message), the PCS will know that the script has finished.

The `PATH TO WORK AREA` tells us where the BPE has set up a temporary work area to run BPE programs. Normally the Bernese menu system runs in the `$U/WORK` area and expects to find program input panels and other input files in directories under `$U`. Since the BPE can run more than one script and thus more than one Bernese program at a time, each script must have its own area for input files and panels so that programs running at the same time do not overwrite panels that are in use. The first thing the BPE does when it starts a script is to look for a free temporary directory with the name `$T/AUTOnnnn` where `nnnn` is a number. First it starts with `0001` and increments this number until it finds a free area. If it cannot find one, then it creates a new one. For the simple PCF file `T_PCF.PCF`, there is only one script running at a time, so there will only be one `AUTOnnnn` directory required, since the second run of `T_SCRIPT` can safely use the same area. From the protocol file we can see that the BPE used the directory `$T/AUTO0001`.

If we go to this directory (`$T/AUTO0001/PAN`), we can see the panel files that were used. We reproduce the panel `DAT151__.PAN` here:

```
 1.5.1  |    PROCESSING: FILENAME PARAMETERS FOR AUTOMATIC PROCESSING

Station Parameters:
  $STATION1 >                   <      $STATION2 >                 <
  ($i will be set to 2-char station abbrev, $STi to 4-char abbrev)

4-character Parameters:
  $CD1      >                <       $CD2      >                <
  $CD3      >                <       $CD4      >                <

3-character Parameters:
  $D1       > 111         <          $D2       >            <
  $D3       >             <          $D4       >            <

2-character Parameters:
  $M   > 04       <        $O   > HI       <        $T   > 20       <
  $U   >          <        $V   >          <        $W   >          <
  $X   >          <        $Y   > 96       <        $Z   >          <

6-character Session Parameters (+ - allowed):
  $JRSS1    > 961110 +0 -0        <  $JRSS2    > 961110               <
  $JRSS3    >                     <  $JRSS4    >                      <
  $JRS+1    > 961120              <  $JRS-1    > 961100               <
  $JRS+2    > 961130              <  $JRS-2    > 961090               <

5-character Session Parameters (+ - allowed):
  $JRD1     > 96111 +0 -0         <  $JRD2     > 96111                <
  $JRD3     >                     <  $JRD4     >                      <
  $JD+1     > 96112               <  $JD-1     > 96110                <
  $JD+2     > 96113               <  $JD-2     > 96109                <

  $GDY1     > 08496 +0 -0         <  $GDY2     > 08496                <
  $GD+1     > 08500               <  $GD-1     > 08495                <
  $GD+2     > 08501               <  $GD-2     > 08494                <

4-character Session Parameters (+ - allowed):
  $SS1      > 1110 +0 -0     <       $SS2      > 1110           <
  $SS3      >                <       $SS4      >                <
  $S+1      > 1120           <       $S-1      > 1100           <
  $S+2      > 1130           <       $S-2      > 1090           <

  $GW1      > 0849 +0 -0     <       $GW2      > 0849           <
  $G+1      > 0850           <       $G-1      > 0849           <
  $G+2      > 0850           <       $G-2      > 0849           <

4-character Year Parameters (+ - allowed):
  $JJ1      > 1996 +0 -0     <       $JJ2      > 1110           <
  $JJ3      >                <       $JJ4      >                <
  $J+1      > 1996           <       $J-1      > 1996           <
  $J+2      > 1996           <       $J-2      > 1996           <
```

Notice that the BPE automatically filled in many of the `DAT151__.PAN` $-variables and the `$O` variable. All these variables may be used in the option panels.

## 22.6.2 Running Bernese Programs in BPE Scripts

In order to do something useful, scripts that are run by the BPE must be able to run Bernese programs. To do this, the user must run a special script `$X/SCRIPT/RUN_PGMS` and set the variable `PGMNAM`, where `PGMNAM` is the name of the Bernese GPS program to be run. For example, the following two lines would execute the Bernese program SNGDIF:

```
PGMNAM="SNGDIF"
. $X/SCRIPT/RUN_PGMS
```

Note that the `RUN_PGMS` script must be "sourced" (command ".") and not executed as a separate shell.

For the VMS version the corresponding lines would be:

```
$ PGMNAM == "SNGDIF"
$ @X:[SCRIPT]RUN_PGMS
```

All the panels that are required by the program to be executed must be contained in the option directory that is specified in the PCF along with the script that is running the program. For example, if you want to run the program SNGDIF, you could have an option directory named:

`$U/OPT/SDIF_NRM`

(`SDIF_NRM` could stand for SNGDIF with NoRMal settings). In this directory all the panels that are used by SNGDIF would need to be present and filled out with all options that are needed.

Then in the PCF we would have:

```
:
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
:
007 SNGDIF   SDIF_NRM          any      1 006
:
```

The names of the Bernese GPS programs may be found by looking at the settings in Menu 0.2, e.g., for the processing programs in Menu 0.2.4:

```
 0.2-4                 DEFAULTS: PROCESSING PROGRAM NAMES

    Preprocessing:
      CODE PREPROCESSING         > CODCHK <
      SINGLE POINT POSITIONING   > CODSPP <
      SINGLE DIFFERENCE FILES    > SNGDIF <
      OLD PHASE PREPROCESSING    > OBSTS1 <
      NEW PHASE PREPROCESSING    > MAUPRP <

    Processing:
      PARAMETER ESTIMATION       > GPSEST <
      IONOSPHERE ESTIMATION      > IONEST <
      ADD NORMAL EQUATIONS       > ADDNEQ <
      ADD NORMAL EQUATIONS       > ADDNEQ2 <

    Path to the Programs         >  XG:/                    <
```

For most of the Bernese programs, the `PGMNAM` variable is set to the name of the GPS program. The name of the corresponding menu program — used to prepare the run of the GPS program—is then obtained by adding "`_P`" to the program name (e.g., CODSPP $\longrightarrow$ CODSPP_P). There are several programs, however, where the name of the menu program is different from the name of the GPS program after adding "`_P`":

```
PGMNAM="CCRINEXO" - Runs menu program CCRNXO_P and GPS program CCRINEXO
PGMNAM="CCRINEXN" - Runs menu program CCRNXN_P and GPS program CCRINEXN
PGMNAM="CCRINEXG" - Runs menu program CCRNXG_P and GPS program CCRINEXG
PGMNAM="CCPREORB" - Runs menu program CCPREN_P and GPS program CCPREORB
PGMNAM="PRETAB"   - Runs menu program BRDTAB_P and GPS program PRETAB
PGMNAM="SATMRK"   - Runs menu program SERVOBS
PGMNAM="ADDNQ2"   - Runs menu program ADDNQ2_P and GPS program ADDNEQ2
```

### 22.6.3 The Panel Files

Before running the BPE all the options in the panel directories used by the BPE scripts have to be set correctly. For example if we want to run the program SNGDIF, we have to fill out all panels that are used by SNGDIF. The SNGDIF program is started using Menu 4.3 which means that all panels required by the SNGDIF program start with DAT43xxx.PAN. (Use the command "=S" after starting the Bernese menu system with "G" and enter the menu program name (e.g., SNGDIF_P) to obtain the menu and submenu options for a specific program).

If we want to use the option directory SDIF_NRM to hold options for SNGDIF, then we have to make a directory $U/OPT/SDIF_NRM and put the basic panels for the menu system (e.g., DAT0*.PAN, DAT151__.PAN, ...) and the DAT43* panels in this directory. We can save some work by letting the Bernese menu system do this for us. If we have a PCF written that uses scripts and option directories corresponding to these scripts, then we may use Menu 6.1 to create the option directories and automatically copy all needed panels there. Menu 6.1 may also be used then to modify the options in these panels.

For example, add these lines to the simple script T_SCRIPT (in the T_SCRIPT BODY section):

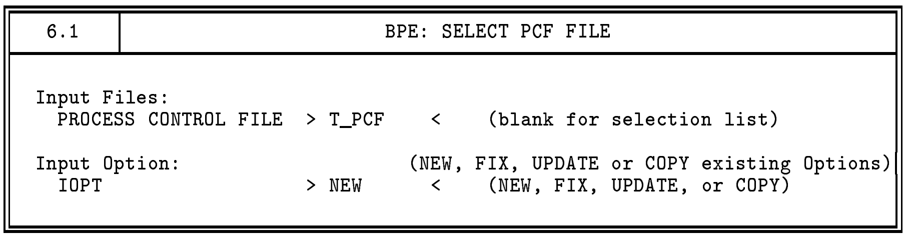
```
PGMNAM="SNGDIF"
. $X/SCRIPT/RUN_PGMS
```

and then change T_PCF.PCF to be:

```
#
# Process Control File
# Comments:
# Any line that starts with a # is a comment line
#
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT
*** ******** ******** ******** ******** * *** *** *** *** *** *** *** *** ***
001 T_SCRIPT SDIF_NRM          any      1
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5
*** ************ ******** ******** ******** ******** ******** ********
#
# PCF Variables
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_0      TEST SETTING V_0                         HI                2
```

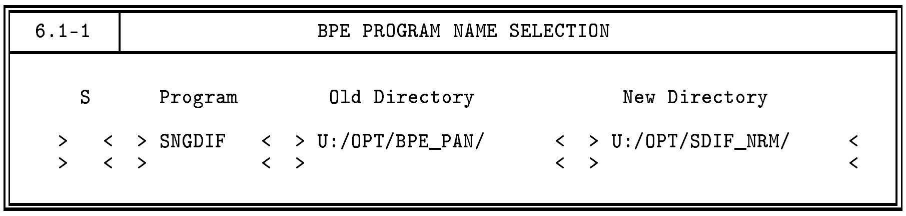
Now we have a PCF that uses a script that actually calls a Bernese program. Note that it is not very useful, but it will demonstrate how to setup option panels for BPE scripts.

Assuming that `$U/OPT/SDIF_NRM` does not exist yet, we can create a new option directory using Menu 6.1 :



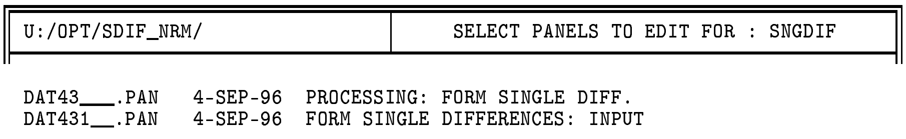
```
 6.1   |                    BPE: SELECT PCF FILE

 Input Files:
   PROCESS CONTROL FILE  > T_PCF    <    (blank for selection list)

 Input Option:                    (NEW, FIX, UPDATE or COPY existing Options)
   IOPT                  > NEW      <    (NEW, FIX, UPDATE, or COPY)
```

Then all scripts in the PCF will be searched to see if they run any Bernese programs. In our simple case we get:



```
 6.1-1 |                 BPE PROGRAM NAME SELECTION

     S      Program        Old Directory             New Directory

   >   <  > SNGDIF   <  > U:/OPT/BPE_PAN/      <  > U:/OPT/SDIF_NRM/     <
   >   <  >          <  >                      <  >                      <
```

This is telling us that the program SNGDIF is being run using the option directory `$U/OPT/SDIF_NRM`. The directory `$U/OPT/BPE_PAN` (under Old Directory) is the directory that will be used to get the first version of the panels from. In the normal software distribution the option directory `$U/OPT/BPE_PAN` does not exist. The user may create it and copy all the panels in his `$U/PAN` directory to the directory `BPE_PAN` or he may change the directory name given in the Panel 6.1–1 under "Old Directory" to, e.g., `$U/PAN`. If we select SNGDIF (by putting an "`S`" in the first data field) then the directory `$U/OPT/SDIF_NRM` will be created for us and panels copied into that directory. After this we are prompted with the list of panels that go with SNGDIF:



```
U:/OPT/SDIF_NRM/                 |    SELECT PANELS TO EDIT FOR : SNGDIF

DAT43___.PAN   4-SEP-96  PROCESSING: FORM SINGLE DIFF.
DAT431__.PAN   4-SEP-96  FORM SINGLE DIFFERENCES: INPUT
```

We can edit any of these panels by selecting them (placing an "`S`" in the most left column). If we select both, first the panel `DAT43___.PAN`, then the panel `DAT431__.PAN` will be displayed to us for editing.

The basic panels of the menu system are — of course — also copied into the option directory `$U/OPT/SDIF_NRM`. The most important of them is the panel for setting the general dataset names Panel 0.3.1 . It will be used by most of the Bernese programs, but it is not displayed by the Bernese menu system when it is located in the option directory. Therefore an external editor has to be used to check and modify the entries in this panel:

```
edit $U/OPT/SDIF_NRM/DAT031__.PAN
```

Please be careful when editing the panel manually: *never change the format of the panels!*

### 22.6.4 Panel Variables

In order to write generic scripts that will, for example, work with any given session, there must be some mechanism to automatically supply Bernese panels with information that will change. To do this, the BPE uses panel variables.

One of the first scripts that is executed in a BPE script is `SET_SESS`. This script will make entries into the Panel 1.5.1 and will set variables for the session being processed. Shown below is the panel with values that are automatically filled in by the BPE (for session 1110, day of year 111, year 1996 in this example):

```
  1.5.1        PROCESSING: FILENAME PARAMETERS FOR AUTOMATIC PROCESSING                  KEYWORDS

    Station Parameters:
      $STATION1 >                  <      $STATION2 >                   <       STATION1 STATION2
      ($i will be set to 2-char station abbrev, $STi to 4-char abbrev)

    4-character Parameters:
      $CD1      >                <        $CD2      >                 <         CODE1    CODE2
      $CD3      >                <        $CD4      >                 <         CODE3    CODE4

    3-character Parameters:
      $D1       > 111         <           $D2       >              <            SESSION1 SESSION2
      $D3       >             <           $D4       >              <            SESSION3 SESSION4

    2-character Parameters:
      $M  > 04        <        $O  > HI        <       $T  > 20        <      2CHAR-M  2CHAR-O  2CHAR-T
      $U  >           <        $V  >           <       $W  >           <      2CHAR-U  2CHAR-V  2CHAR-W
      $X  >           <        $Y  > 96        <       $Z  >           <      2CHAR-X  2CHAR-Y  2CHAR-Z

    6-character Session Parameters (+ - allowed):
      $JRSS1    > 961110 +0 -0         <  $JRSS2    > 961110                <   6-SESS1  6-SESS2
      $JRSS3    >                      <  $JRSS4    >                       <   6-SESS3  6-SESS4
      $JRS+1    > 961120               <  $JRS-1    > 961100                <   6-SES+1  6-SES-1
      $JRS+2    > 961130               <  $JRS-2    > 961090                <   6-SES+2  6-SES-2

    5-character Session Parameters (+ - allowed):
      $JRD1     > 96111 +0 -0          <  $JRD2     > 96111                 <   5-SESS1  5-SESS2
      $JRD3     >                      <  $JRD4     >                       <   5-SESS3  5-SESS4
      $JD+1     > 96112                <  $JD-1     > 96110                 <   5-SES+1  5-SES-1
      $JD+2     > 96113                <  $JD-2     > 96109                 <   5-SES+2  5-SES-2

      $GDY1     > 08496 +0 -0          <  $GDY2     > 08496                 <   5-WDAY1  5-WDAY2
      $GD+1     > 08500                <  $GD-1     > 08495                 <   5-WDY+1  5-WDY-1
      $GD+2     > 08501                <  $GD-2     > 08494                 <   5-WDY+2  5-WDY-2

    4-character Session Parameters (+ - allowed):
      $SS1      > 1110 +0 -0     <        $SS2      > 1110            <         4-SESS1  4-SESS2
      $SS3      >                <        $SS4      >                 <         4-SESS3  4-SESS4
      $S+1      > 1120           <        $S-1      > 1100            <         4-SES+1  4-SES-1
      $S+2      > 1130           <        $S-2      > 1090            <         4-SES+2  4-SES-2

      $GW1      > 0849 +0 -0     <        $GW2      > 0849            <         4-WEEK1  4-WEEK2
      $G+1      > 0850           <        $G-1      > 0849            <         4-WEK+1  4-WEK-1
      $G+2      > 0850           <        $G-2      > 0849            <         4-WEK+2  4-WEK-2

    4-character Year Parameters (+ - allowed):
      $JJ1      > 1996 +0 -0     <        $JJ2      > 1110            <         4-YEAR1  4-YEAR2
      $JJ3      >                <        $JJ4      >                 <         4-YEAR3  4-YEAR4
      $J+1      > 1996           <        $J-1      > 1996            <         4-YEA+1  4-YEA-1
      $J+2      > 1996           <        $J-2      > 1996            <         4-YEA+2  4-YEA-2
```

The `SET_SESS` script does not set *all* the values in `DAT151__.PAN`, but only those that are likely to be used by most of the scripts. This variable panel `DAT151__.PAN` is simply a means of setting some generic values to be passed into other Bernese panels. The names of the variables are shown in the main body of the panel and to the right (after column 80) the keyword for each variable is given. The values of the panel variables are shown in-between the angle brackets. In the above panel, only the PCF variable `V_O` (`$O`) has been set. Note that the variables `$T`, `$M`, `$Y` contain the day, month, and the two-digit year, respectively.

To use $-variables within Bernese panels, the variable name is used, `$CD1` for example. The keywords are used together with the script named `PUTKEYWE` to set the values in panel `DAT151__.PAN`.

For example, the `$CD1` variable is set using the keyword `CODE1`. The script `PUTKEYWE` is covered in the Section 22.8.3.

Some sets of variables have the annotation "`(+ - allowed)`". This option allows to specify a range of sessions. For example, it may be desirable to select files from more than just one session in a specific program run. A gliding comparison of coordinate files from, e.g., the last 7 sessions using the program COMPAR is a possible example. In this case you may use, e.g., "`1110 +0 -6`" for the value of `$SS1` etc. When the ± option is used the BPE will correctly adjust the day of the year and the year, accounting for transitions between the current and next or previous year. Remember that the format of a 4-digit session is to have the day of the year as the first three digits followed by the 1-character session identification within the day. To make use of the ± option you have to specify the variables `V_PLUS` and `V_MINUS` in the PCF file:

```
...
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_O      EUROCLUS ORBIT FILE NAME                 HI                2
V_PLUS   PLUS  DAYS                               +0                2
V_MINUS  MINUS DAYS                               -0                2
```

When starting the BPE using Menu 6.4.1 these variables will also be displayed to you (in Panel 6.4.1–1.3) and you may change their values. The variables `$JJ1`, `$JRSS1`, `$JRD1`, `$GDY1`, `$SS1`, and `$GW1` will then contain the ± values (e.g., "`961110 +1 -2` for `$JRSS1`, if `V_PLUS=1`, `V_MINUS=2`), whereas the variables `$JRSS2`, `$JRD2`, ... will NOT include the ± parameters (e.g., "`961110`" only for `$JRSS2`).

The definition of ranges of sessions may generate some strange behaviour if more than one session per day is defined in the session definition panel ( Panel 1.3–2 ). We therefore recommend to define only one session per day (see example below) if you want to make use of session ranges.

```
 1.3-2                  CAMPAIGNS: SESSION DEFINITION

 SESSION NUMBER          START DATE                    END DATE

     nnnn           yy mm dd      hh mm ss       yy mm dd      hh mm ss

   > ???0 <       >          < > 00 00 00 <   >          < > 23 59 59 <
```

If any of the special variables are set (`V_O`, `V_PLUS`, etc...) then the `SET_SESS` script will also set the appropriate variables in `DAT151__.PAN`.

## 22.6.5 Parallel Scripts

In this section we describe how to implement parallel processing into the scripts. The examples are given in UNIX Bourne shell. The VMS user are referred to the example in `X:[USERSCPT]`.

The example PCF contains a parallel script for the solution of the phase ambiguities using QIF. The corresponding parts in the PCF are shown below:

```
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
:
018 GPSQIFAP EURO_QIF          any      1 017
019 GPSQIF_P EURO_QIF          any      1 018
:
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
018                       $tmp1
019                       PARALLEL $tmp1
:
```

The script `GPSQIFAP` in the first step (`PID 018`) writes all files to be processed into a temporary file (see Section 22.5.2) whose name is available within the script as `$PARAM1`. The part of the script where the file is written looks like

```
#
# SHELL SCRIPT-NAME BODY
# ----------------------
#
# Clean up old output files
# -------------------------
   rm $CAMP_PTH$CAMPAIGN/OUT/*$SESSION.OUT
#
# Get the list of phase single difference files
# ---------------------------------------------
   filspec=""$CAMP_PTH""$CAMPAIGN"/OBS/????"$SESSION".PSH"
   export filspec
#
# Write all phase single difference files into a temp. file
# ---------------------------------------------------------
   for file in  $filspec
   do
     echo "$file" >> "$T/AUTO_TMP/$PARAM1"
   done
#
# --------------
# end the script
# --------------
```

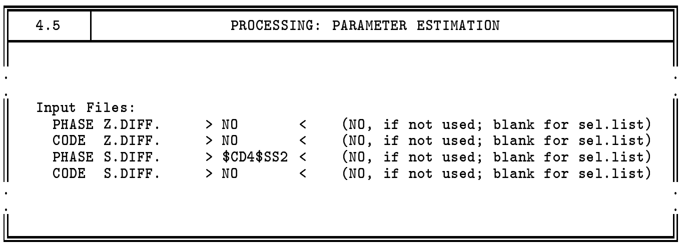
The PCS reads the temporary file and starts the script `GPSQIF_P` (`PID 019`) for each line in the file (i.e., for each single difference observation file). The line is routed as `$PARAM1` into the script. The parallel script `GPSQIF_P` puts the name of the single difference observation file (e.g., `P:/DOCU42_1/OBS/KOWT1650.PZH`) into the input panel of the program GPSEST and runs it to solve the ambiguities for this baseline. The body of the script `GPSQIF_P` is:

```
#
# SHELL SCRIPT-NAME BODY
# ----------------------
#
# Define the name of the program
#
   PGMNAM="GPSEST"
   export PGMNAM
#
# Get two temporary file names using the process ID ($$)
# ------------------------------------------------------
   tmpid=$$
   export tmpid
#
# 1st temp. file is named "tmpfil1"
# ---------------------------------
   tmpfil1="tmp1"$tmpid".txt"
   export tmpfil1
#
# 2nd temp. file is named "tmpfil2"
# ---------------------------------
   tmpfil2="tmp3"$tmpid".txt"
   export tmpfil2
#
# Process the baseline (given by PARAM1)
# ---------------------------------------
    echo processing "$PARAM1"
#
# Use the program "PRSLIN" to extract the name of the single diff. file
# (example: PARAM1=P:/DOCU42_1/OBS/KOWT1650.PZH)
# ---------------------------------------------------------------------
    echo "$PARAM1" > "$tmpfil1"
    $X/EXE/RBPE PRSLIN "$tmpfil1" "$tmpfil2"
    seterr
#
# The variable "file_r" contains the file name only
# (example: file_r=KOWT1650)
# -------------------------------------------------
    file_r=‘cat "$tmpfil2"‘
    export file_r
    echo file_r is "$file_r"
#
# The variable "file_r4" contains the baseline ID only
# (example: file_r4=KOWT)
# ----------------------------------------------------
    file_r4=‘echo "$file_r" | cut -c1-4‘
    export file_r4
    echo file_r4 is "$file_r4"
#
# Use the program "PUTKEYWE" to put the value of the variable
# "$file_r4" into the panel "DAT151__PAN" and make it available
# as variable "$CD4" in the Bernese input panels.
# -------------------------------------------------------------
    pp1="$U/PAN/DAT151__.PAN"
    export pp1
    pp2="CODE4"
    export pp2
    pp3=""$file_r4""
    export pp3
    . $X/SCRIPT/PUTKEYWE
    seterr
#
# Use the program "RUN_PGMS" to run the menu and main program of GPSEST
# ---------------------------------------------------------------------
    . $X/SCRIPT/RUN_PGMS
    seterr
#
# remove the two temp. files
# --------------------------
    do_rm "$tmpfil1"
    do_rm "$tmpfil2"
#
# -------------
# end the script
# -------------
```

The programs PRSLIN and PUTKEYWE are described in the Section 22.8. After running the program PUTKEYWE the panel `$U/PAN/DAT151__.PAN` has the following new entry (Please keep in mind that this panel is located in the `$T/AUTOxxxx/`-environment of the script):

```
 1.5.1  |     PROCESSING: FILENAME PARAMETERS FOR AUTOMATIC PROCESSING     | KEYWORDS
        |                                                                  |
.                                                                          .
.                                                                          .
   4-character Parameters:
     $CD1      >                 <       $CD2      >                 <      CODE1    CODE2
     $CD3      >                 <       $CD4      > KOWT            <      CODE3    CODE4
.                                                                          .
.                                                                          .
```

The variable `$CD4` may be used in the panels of GPSEST to select the baseline for processing in Panel 4.5 as well as to name the program output ( Panel 4.5.0 ), or to put the baseline ID into the title line ( Panel 4.5.1 ). As an example the selection of the baseline in Panel 4.5 (located in the option directory `$U/OPT/EURO_QIF`) is shown below:



```
4.5                    PROCESSING: PARAMETER ESTIMATION

Input Files:
  PHASE Z.DIFF.     > NO       <    (NO, if not used; blank for sel.list)
  CODE  Z.DIFF.     > NO       <    (NO, if not used; blank for sel.list)
  PHASE S.DIFF.     > $CD4$SS2 <    (NO, if not used; blank for sel.list)
  CODE  S.DIFF.     > NO       <    (NO, if not used; blank for sel.list)
```

It is not always best to process each baseline (or station) separately. In some cases it might be preferable to form groups of baselines (or stations). We give a simple example of how to implement this grouping into a parallelization script. The program CODSPP may run station by station because the single point positioning using the code observations is only station dependent. It is, therefore, possible to use the script presented above for a station-wise parallelization. We show in the following how to set-up the parallel processing in groups of five stations. The parallelization is performed in the usual way (not part of the BPE example):

```
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
:
005 CODSPPAP EUROCLUS          any      1 004
006 CODSPP_P EUROCLUS          any      1 003 005
:
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
005                       $tmp1
006                       PARALLEL $tmp1
:
```

The script `CODSPPAP` (`PID 005`) defines the groups of stations. It writes the names of the code observation files belonging to one group into “cluster-files” whose names are written into the file references by `$tmp1`. The code of this part of the script `CODSPPAP` may look like:

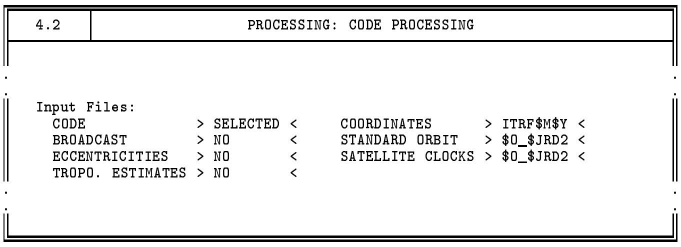
```
#
# SHELL SCRIPT-NAME BODY
# ----------------------
#
# Clean up old output files
# -------------------------
   rm $CAMP_PTH$CAMPAIGN/OUT/CODSPP.L??
   rm -f "$CAMP_PTH""$CAMPAIGN"/OUT/"$YEAR""$DAYYEAR"*.BPE
#
# Get two temporary file names using the process ID ($$)
# -------------------------------------------------------
   tmpid=$$
   export tmpid
#
# 1st temp. file is named "tmpfil1"
# ---------------------------------
   tmpfil1="tmp1"$tmpid".txt"
   export tmpfil1
#
# 2nd temp. file is named "tmpfil2"
# ---------------------------------
   tmpfil2="tmp3"$tmpid".txt"
   export tmpfil2
#
# Set MAXFIL = maximum number of files per cluster
# ------------------------------------------------
   MAXFIL="5"
   export MAXFIL
#
# Init a counter to count the stations per cluster
# ------------------------------------------------
   COUNT="0"
   export COUNT
#
# Init a counter to count the clusters
# ------------------------------------
   CLUST="1"
   export CLUST
#
# Get the list of phase single difference files
# ---------------------------------------------
   filspec=""$CAMP_PTH""$CAMPAIGN"/OBS/????"$SESSION".CZH"
   export filspec
#
# Loop over all Baselines
# -----------------------
   for CZHfil in  $filspec
   do
#
# Get the filename without path and extension
# --------------------------------------------
    echo "$CZHfil" > "$TMPFIL1"
    $X/EXE/RBPE PRSLIN "$TMPFIL1" "$TMPFIL2"
    seterr
    FILNAM=`cat "$TMPFIL2"`
    export FILNAM
#
# Add the file name to a cluster file
# -----------------------------------
    echo "$FILNAM" >> "$CAMP_PTH""$CAMPAIGN"/OUT/"$YEAR""$DAYYEAR""$CLUST".BPE
#
# If the first station was written in a cluster file it must be put
# into the list of jobs in "PARAM1"
# -----------------------------------------------------------------
    if [ "$COUNT" -eq 1 ]
    then
        echo "$CAMP_PTH""$CAMPAIGN"/OUT/"$YEAR""$DAYYEAR""$CLUST".BPE >> "$T/AUTO_TMP/$PARAM1"
    fi
#
# Check the number of files per cluster
# -------------------------------------
    if [ "$COUNT" -ge "$MAXFIL" ]
    then
#
# Start a new cluster
# -------------------
        CLUST=`expr "$CLUST" + 1`
        export CLUST
#
# Reset the counter of stations for the new cluster
# -------------------------------------------------
        COUNT="1"
        export COUNT
    else
#
# A further station was added into the cluster
# --------------------------------------------
        COUNT=`expr "$COUNT" + 1`
        export COUNT
    fi
#
# remove the two temp. files
# --------------------------
    do_rm "$tmpfil1"
    do_rm "$tmpfil2"
#
# -------------
# end the script
# -------------
```

The PCS reads the content of the temporary file referenced by `$tmp1`. Each of the cluster file names

(`*.BPE`) will be routed as `$PARAM1` into the parallel script `CODSPP_P` which copies it to the selection list name and executes the program CODSPP:

```
#
# SHELL SCRIPT-NAME BODY
# ----------------------
#
# Define the name of the program
#
  PGMNAM="CODSPP"
  export PGMNAM
#
# Process the cluster (given by PARAM1)
# ------------------------------------
    echo processing "$PARAM1"
#
# Generate "SELECTED" file
# ------------------------
    rm -f U:/WORK/CDZFILE.SEL
    cp "$PARAM1" U:/WORK/CDZFILE.SEL
#
# Use the program "RUN_PGMS" to run the menu and main program of GPSEST
# ---------------------------------------------------------------------
    . $X/SCRIPT/RUN_PGMS
    seterr
#
# --------------
# end the script
# --------------
```

In order to force CODSPP to use the selection file `U:/WORK/CDZFILE.SEL` the option "SELECTED" must be used for the code observation file in Panel 4.2 in the directory `$U/OPT/EUROCLUS`:



```
4.2                    PROCESSING: CODE PROCESSING

Input Files:
  CODE            > SELECTED <     COORDINATES      > ITRF$M$Y <
  BROADCAST       > NO       <     STANDARD ORBIT   > $O_$JRD2 <
  ECCENTRICITIES  > NO       <     SATELLITE CLOCKS > $O_$JRD2 <
  TROPO. ESTIMATES > NO      <
```

### 22.6.6 The Clean Script

In the `$X/EXE` directory there is a simple script named `CLEAN`. This script can be used to delete any left-over temporary files that the PCS may have created on previous runs and will make sure that the `PCFCTL.CPU` is initialized to show that no jobs are running. The clean script can only be run if no PCS's are running. If it is run while a PCS is running, the PCS will crash since temporary files that it is using will be deleted. To run the clean script, the user on a UNIX platform can use the Bourne shell dot command:

```
sakic[jjohnson]:V4.2-$ . CLEAN
```

On a VMS systems the corresponding command is:

```
@X:[EXE]CLEAN
```

### 22.6.7 Starting PCS From the Shell or System Prompt

As seen previously, the user can start the BPE using Menu 6.4.1. In some cases it may be desirable to start the BPE from a shell or shell script (UNIX) or from the system prompt (VMS). One example would be if you would like to run a BPE job every day at a given time. You could then set up a cron job that starts a script. Here is an example of starting the BPE from a shell:

First we change directory to the `$U/WORK` area, and then we start the PCS specifying the PCF file (here `GET_ORB.PCF`), the desired campaign (`NEW_CAMP`), the year (`95`), the session (`0710`), and possibly other parameters.

UNIX Version:

```
erde[jjohnson]:V42-$ cd $U/WORK
erde[jjohnson]:V42-$ PCS GET_ORB CAMPAIGN NEW_CAMP YEAR 95 SES 0710
```

VMS Version:

```
$ SET DEFAULT U:[WORK]
$ @X:[EXE]PCS.COM GET_ORB CAMPAIGN NEW_CAMP YEAR 95 SES 0101
```

Please be aware of the fact that PCF variables (variables defined in the third part of the PCF file) will not automatically be set when using the command above. You have to explicitly add these parameters when starting the PCS (e.g., by adding “`V_PLUS +2 V_MINUS -1`” to the PCS command above). On VMS the maximum number of parameters to be passed to a DCL command file is limited to 8. To overcome this limitation you may specify a file as one of the parameters. The file may then contain as many additional parameters as necessary. This might look as follows (this procedure is also used by the menu system on both, UNIX and VMS systems):

```
$ @X:[EXE]PCS.COM GET_ORB YEAR 95 SES 0101 + U:[INP]PCS.INP
```

The “+” indicates that the name of a file will follow containing additional input parameters for the PCS. The file `U:[INP]PCS.INP` in this case might have the following content:

```
CAMPAIGN
NEW_CAMP
V_O
HI
V_PLUS
+2
V_MINUS
-1
```

Further important keywords given either in the input file or in the command line are

| | | |
|---|---|---|
| `TASKID or TID` | : | The first two characters of the protocol as well as the log file are `00` by default. You may change them to another 2-character string using this keyword. |
| `YEAR or YR` | : | Year of the actual session to be processed. |
| `CAMPAIGN or CAMP` | : | Campaign to be processed. |
| `SESSION or SES` | : | Session to be processed |
| `PID` | : | To restart the PCS at a particular PID (e.g., after an error occured) use `PID xxx` where `xxx` is the PID of the first script to be executed. |
| `SKIP` | : | Add the PID or a list of PIDs to this keyword to skip the corresponding scripts for this run. |
| `CPU_FILE` | : | Name of the CPU file without extension ".CPU", if it is different from "PCFCTL.CPU". |
| `V_X, V_O, V_Z, V_U, V_V, V_W, V_PLUS, V_MINUS, V_START, V_END` | : | Content of variables which may be used in the scripts. |
| `V_xxxxxx` | : | Additional variables may be defined freely. The length of the name of such variables is limited to 8 characters; the length of the content is limited to 16 characters. The defaults values may be specified in the third section of the PCF. These variables cannot be used in the panels. The values of such variables may, however, be put into the panels using the `PUTKEYWE` command (see below). |

## 22.7 BPE Menu Items

We can access the BPE menu items through Menu 6 :

```
 6       |                  BPE:  OPTION MENU
S:Y C:0  |

   0     |  PANEL UPDATE    : Update Panels for New Release
   1     |  PANEL EDITING   : Prepare Option Panels for BPE
   2     |  PREPARE RINEX   : Prepare RINEX Files for BPE
   3 ..  |  SPECIAL FILES   : Prepare Special Files for BPE
   4 ..  |  BPE PROCESSING  : BPE Processing (Session or Campaign)
   5 ..  |  BPE SERVICES    : BPE Service Programs

            Enter Selection :
```

The menu items under this option allow the user to deal with panel files, PCF files, and to start BPE runs.

### 22.7.1 PANEL UPDATE

This option ( Menu 6.0 ) can be used to update panels in one or more option directories. This is useful when the panels for the Bernese programs are changed and new keywords (input fields) are added to them (e.g., with a new release of the software or due to changes by the user). In this case, new panels will be created that retain the settings for pre-existing parameters with new parameters filled in from a master set of panels. This menu item is not thought to be used to change panel options for the BPE. It should only be used if menu programs and input panels have changed. To modify BPE options you should use Menu 6.1 . The first menu that appears under the `PANEL UPDATE` option is shown below:

```
 6.0     |                  BPE: PANEL UPDATING

 Master Panels:
   PANEL DIRECTORY  > X:/PAN/              <
   MASTER PANEL     >          <               (blank for selection list)

 Panels to be Updated:
   PANEL DIRECTORY  > U:/PAN/              <   (NO or full directory name)
     or
   DIRECTORY LIST   > NO       <               (NO, blank for sel. list)

 Update Options:
   UPDATE/COPY      > UPDATE   <        (UPDATE or COPY)
   EXISTING/ALL     > EXISTING <        (EXISTING panels only, ALL panels)
```

Due to the fact that the BPE creates and uses a lot of panel option directories, this tool is essential to maintain the hundreds of panels. Therefore you also have the possibility to update an entire list of panel directories in one run. An example of such a list of directories may be found in `$X/INX` with the name `EXAMPLE.UPD`. A detailed description of the options above may be found in the help panel `$X/HLP/DAT60___.HLP`.

### 22.7.2 PANEL EDITING

Below the panel for `PANEL EDITING` ( Menu 6.1 )is shown:

```
 6.1                      BPE: SELECT PCF FILE

Input Files:
  PROCESS CONTROL FILE  >          <    (blank for selection list)

Input Option:                   (NEW, FIX, UPDATE or COPY existing Options)
  IOPT                  > FIX      <    (NEW, FIX, UPDATE, or COPY)
```

This menu item is the main tool to handle the BPE option panels. It allows the user to create new panel directories and to modify the options set in the individual panels in a user-friendly way. In addition to the name of the PCF file for which new panels have to be created or for which options in existing panels have to be modified, the user selects one of the following options: `NEW`, `FIX`, `UPDATE`, or `COPY`. The `IOPT` options will be explained below. Once a PCF file has been selected, all the scripts specified in the PCF will search for Bernese GPS programs that are to be run and the program names along with the option directories to be used with each of the programs will be extracted. Below is an example of what the display might look like after a PCF file has been selected:

```
6.1-1                 BPE PROGRAM NAME SELECTION

    S      Program        Old Directory           New Directory

  >   <  > FTPRNX   <  > U:/OPT/FTP_OPTS/     <  > U:/OPT/FTP_OPTS/     <
  >   <  > FTPORB   <  > U:/OPT/FTP_OPTS/     <  > U:/OPT/FTP_OPTS/     <
  >   <  > CCRNXN   <  > U:/OPT/CCRX__ON/     <  > U:/OPT/CCRX__ON/     <
  >   <  > CCRNXO   <  > U:/OPT/CCRX__ON/     <  > U:/OPT/CCRX__ON/     <
  >   <  > CRDRNX   <  > U:/OPT/CRDRNX_1/     <  > U:/OPT/CRDRNX_1/     <
  >   <  > FTPRNX   <  > U:/OPT/GET_MISS/     <  > U:/OPT/GET_MISS/     <

U:/PAN/DAT611__.PAN                                               REPLACE
```

Depending on how large the screen is, the user may see more lines at a time than shown above. If not all lines are fitting on the screen, the missing lines can be viewed my moving the cursor to the bottom line and then pressing the down arrow key. The lines will then scroll up the screen until the end of the list is reached. In order to select individual lines, the user places an "S" in the first column of each line that is desired. After all lines have been selected, the user types the continuation character (see Section 3.3.2).

The first column is used to select panels to edit. The second column shows the name of the Bernese program. The third column shows a source option directory and the last column shows a destination directory. In the above example, the old and new directories are set to be the same directories. The use of the old and new directory will be explained in the following sections.

When a program is selected, all panels that are related to the program will be presented to the user. For example, if the `CCRNXN` line were selected, the user would be presented with the following:

```
U:/OPT/CCRX__ON/               SELECT PANELS TO EDIT FOR : CCRNXN

DAT2562_.PAN  29-MAR-95  Concatenate RINEX Navigation Files  (Main Data Panel)
DAT25621.PAN  29-MAR-95  Concatenate RINEX Navigation Files: Input 1
```

All the panels that pertain to the CCRNXN program are displayed. The CCRNXN program corresponds to Menu 2.5.6.2 . The menu program will offer the user all panels that start with DAT2562, since all of these panels will contain options for the program CCRNXN. In this case there are only two panels offered. For more complicated programs (e.g., GPSEST), more panel names will be displayed.

You may now select each panel you wish to edit by placing an “S” in the left most column. All selected panels will then be offered to the user for editing, one after another. Once all panels have been edited, the complete list of panels will be shown again. At this point the user can either select more panels to edit, or exit by typing “Q” in the first column.

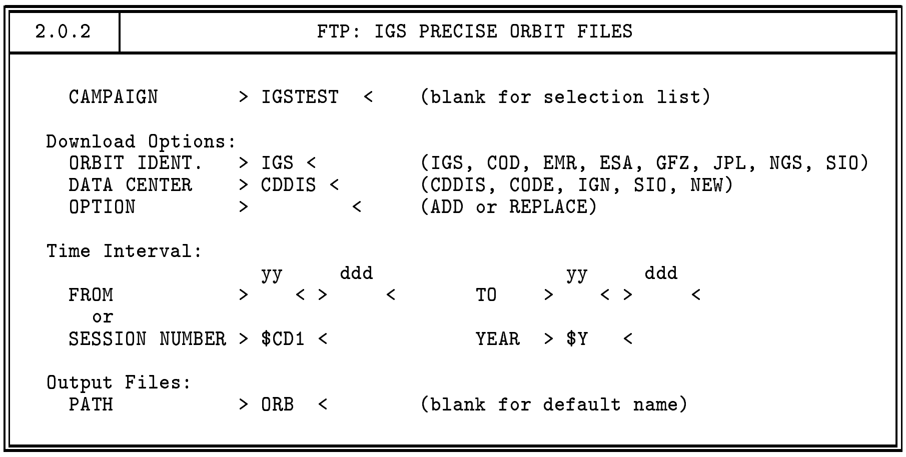
#### 22.7.2.1 Panel Editing FIX Option

This option is used if the user wants to edit the options in existing panels. For this option, the Old Directory column will be the same as the New Directory column. If the directory name in the Old Directory column is changed, then the panels there will be copied to the New Directory before the panels are edited. Usually, however, the Old Directory column is not used with the FIX option.

If the options in one of the *default panels* have to be changed (e.g., the pole file in the Panel 0.3.1 ) the user has to edit the corresponding panel “manually” and cannot make use of Menu 6.1 (default panels are not displayed there).

#### 22.7.2.2 Panel Editing UPDATE Option

When this option is selected, the panels in the New Directory will be updated with the panels in the Old Directory before panels are edited. The Old Directory column will be filled in with the value U:/OPT/BPE_PAN. Updating a panel means that any new fields in the source panel (Old Directory) are added to the target panel, but any selections existing in the target panel are left unchanged. For example, suppose a new DATA CENTER was added for the IGS precise orbit panel DAT202__.PAN:



```
 2.0.2  |                FTP: IGS PRECISE ORBIT FILES

   CAMPAIGN       > IGSTEST  <    (blank for selection list)

 Download Options:
   ORBIT IDENT.   > IGS <         (IGS, COD, EMR, ESA, GFZ, JPL, NGS, SIO)
   DATA CENTER    > CDDIS <       (CDDIS, CODE, IGN, SIO, NEW)
   OPTION         >        <      (ADD or REPLACE)

 Time Interval:
                    yy     ddd                 yy     ddd
   FROM           >    < >     <       TO    >    < >     <
     or
   SESSION NUMBER > $CD1 <             YEAR  > $Y   <

 Output Files:
   PATH           > ORB  <        (blank for default name)
```

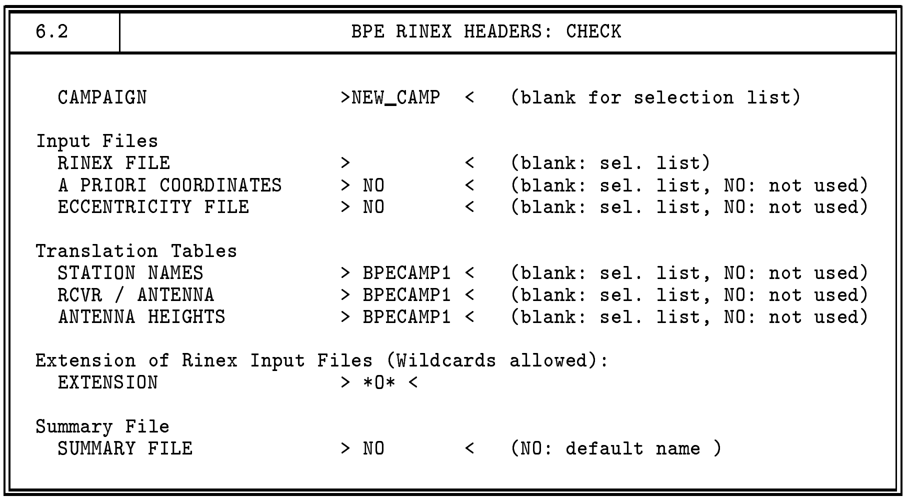
Then the panel in the New Directory would be updated to show NEW as one of the options for the DATA CENTER field, but the actual value in this field would be left unchanged. If a completely new field is added to the panel, say OPTION2 under OPTION, then this new line would be added to the panel in New Directory, and whatever selection existed in the source panel would be copied over since there would be no pre-existing options in the target panel. This update of panels may be performed in a more general way using Menu 6.0 .

#### 22.7.2.3 Panel Editing COPY Option

This option will just copy the panels in the Old Directory over to New Directory. The Old Directory column is filled in with `U:/OPT/BPE_PAN` initially, but may be edited. After panels are copied, they are not offered for editing. Pre-existing panels (together with their option settings) in the New Directory column are overwritten.

### 22.7.3 PREPARE RINEX

The `PREPARE RINEX` option is presently *only working on UNIX systems*. It allows the user to quickly look at RINEX observation files to check such parameters as antenna heights, receiver and antenna names, etc. The first panel that appears after selecting Menu 6.2 is:



```
6.2                     BPE RINEX HEADERS: CHECK

  CAMPAIGN                 >NEW_CAMP  <   (blank for selection list)

Input Files
  RINEX FILE               >          <   (blank: sel. list)
  A PRIORI COORDINATES     > NO       <   (blank: sel. list, NO: not used)
  ECCENTRICITY FILE        > NO       <   (blank: sel. list, NO: not used)

Translation Tables
  STATION NAMES            > BPECAMP1 <   (blank: sel. list, NO: not used)
  RCVR / ANTENNA           > BPECAMP1 <   (blank: sel. list, NO: not used)
  ANTENNA HEIGHTS          > BPECAMP1 <   (blank: sel. list, NO: not used)

Extension of Rinex Input Files (Wildcards allowed):
  EXTENSION                > *O* <

Summary File
  SUMMARY FILE             > NO       <   (NO: default name )
```

The parameters for this panel are described in the corresponding help panel.

After the above panel has been filled out, the user will be presented with a selection list of all RINEX files that were found given the specified parameters. For each RINEX file selected, header information will be extracted and presented to the user in a panel such as the following:

```
 6.2-1                     BPE RINEX HEADER CHECKING
               (Edit Old OR Enter New To Change Entries and Select)
                 (If New Filename Changes are applied to New File)
             (If New Stationname is shown Station is in Translation Table !!)

   S   Old Filename  Old Station Name  Old Ant Height       Receiver
   S   New Filename  New Station Name  New Ant Height       Antenna
                                             [m]

 > S <> FORT0900 <> FORTALEZA           <>  0.6430 <> ROGUE SNR-8000       <
 >   <>          <>                     <>         <> DORNE MARGOLIN T     <
 > S <> NLIB0900 <> NLIB                <>  0.1630 <> ROGUE SNR-8000       <
 >   <>          <>                     <>         <> DORNE MARGOLIN T     <
 > S <> PIE10900 <> PIETOWN             <>  0.1630 <> ROGUE SNR-8000       <
 >   <>          <>                     <>         <> DORNE MARGOLIN T     <
 > S <> ZIMM0900 <> ZIMM                <>  0.0000 <> TRIMBLE 4000SSE      <
 >   <>          <>                     <>         <> 4000ST L1/L2 GEOD    <
 >   <>          <>                     <>         <>                      <
```

The data extracted from the RINEX files will appear in pairs of lines. The first column allows the user to select a pair to be updated. The second column shows the name of the RINEX file, the third the station name in the RINEX file, the fourth column the antenna height, and the fifth column the receiver/antenna pair. If a translation name is found for the station name, then it will appear on the second line of the pair. The same is true for the antenna height. If the receiver/antenna pair is not found in the receiver/antenna translation table, then the character “R” will appear in the first column of the second line of the pair. The user has the option of editing any of the fields of the pair, although values in the second line for parameters two, three, and four will take precedence over changes in the parameters on the first line of the pair. An “S” appearing in the first column of either line of the pair will cause the RINEX file to be updated if either something was changed in the first line or there are new values in the second line. Note that all pairs are selected by default. If you do not want to change a RINEX file, then you must remove the “S” (replacing it with a blank) in the first column for the pair.

### 22.7.4 SPECIAL FILES

There are two menu options (Menu 6.3.1 and Menu 6.3.2) to prepare special files for the BPE processing. Both options are not fully implemented at present and should not be used. The special files that would be generated by the two options may as well be created just using an ordinary editor. The special file containing a list of stations for which RINEX files should be downloaded by ftp in connection with Menu 2.0.1 (presently only working on UNIX) just contains a list of the 4-character identifiers of the IGS sites you are interested in. An example of such an FTP station file is available in the directory `$X/INX` under the name `EXAMPLE.FTP`. It may be specified in Panel 2.0.1 for the input field “`STATION LIST`”.

The second special file of importance for the BPE processing is the list of stations to be fixed or constrained in the programs GPSEST or ADDNEQ. More details on the format of this file type may be found in Chapter 24. To use such a file to fix station coordinates you have to specify “`SPECIAL_FILE`” in the Panel 4.5–1 (GPSEST) or Panel 4.8.1–1 (ADDNEQ) for the station(s) to be fixed and the name of the “fixed station” file in the Panel 4.5–1.5 or Panel 4.8.1–1.5, respectively. To *constrain* the station coordinates (the recommended procedure) you need to specify “`SPECIAL_FILE`” in Panel 4.5–2.4.B or Panel 4.8.1–1.7, respectively (again the name of the “fixed station” file is specified in Panel 4.5–1.5 or Panel 4.8.1–1.5). If the “fixed station” file contains a priori sigma values the corresponding station will be constrained, if no a priori sigmas are specified, the corresponding station will be fixed (see Chapter 24).

A third special file type is used in connection with the estimation of troposphere zenith delays for individual stations (see Chapter 12). A special file may be used to constrain the troposphere estimates of individual sites (see `$X/INX/EXAMPLE.SIG` for an example and Chapter 24) in the program GPSEST. You have to set the option "`SPECIAL_FILE`" in Panel 4.5–2.4.0 and the name of the special troposphere sigma file in Panel 4.5–1.6 to activate this special option.

### 22.7.5 BPE PROCESSING

To start BPE runs in a comfortable way you may use Menu 6.4.1. It allows you to set up a BPE run to process one or N consecutive sessions. The preparation of a BPE run with the menu system follows exactly the same principles as the preparation of e.g., a GPSEST run. The menu system generates — according to your selection of options — two script files (UNIX: `$U/WORK/PCS.COM` and `$U/WORK/PCS.CTL`, VMS: `U:[WORK]PCS.CO2` and `U:[WORK]PCS.CTL`, see also Chapter 3) and an option input file (`$U/INP/PCS.INP`). Whether the PCS is running in the foreground or background and whether you can schedule the BPE run for a specific time depends on the setting of the "`JOB CLASS`" option in Menu 0.1. You may also use the command "`SJ PCS`" to directly start a BPE run that was previously prepared using the menu system (see Chapter 3).

If you intend to run several PCS (BPE runs) at the same time you may want to give the PCS a higher priority compared to the BPE scripts started by the PCS. This will make sure that the process control script gets the necessary CPU to efficiently control the status of the various BPE scripts instead of having to compete with all the other scripts running. The batch queue (priority) used by the PCS is defined in Menu 0.1, option "`JOB CLASS`".

When you want to run more than one PCS at the same time you have to choose a different job identification character in Panel 6.4.1, option `JOB CHARACTER`, for each BPE run. When two or more such PCS are processing *data of the same session* you also have to specify another `TASK IDENTIFICATION` in Panel 6.4.1–1 for each PCS (e.g., "AA" instead of "00"). The task identification is used to uniquely name the protocol and log files generated by the various scripts specified in the PCF file in the directories `$P/CAMP/OUT` and `$T/AUTO_TMP`, respectively.

```
6.4.1-1                 BPE SESSION PROCESSING: INPUT OPTIONS

    Sessions Information:
      SESSION (START)      > 1110 <
      YEAR (START)         > 1996 <
      NUMBER OF SESSIONS   > 1    <      (if negative: processing backwards)

    Task Identification:
      TASK IDENTIFICATION  > AA <        (blank: 00)

    CPU/QUEUE Specification:
      CPU / BATCH QUEUE    > NO       <  (NO, or blank for selection list)

    Special Options:
      SPECIAL PARAMETERS   > NEW  <      (OLD.. NEW.. or ASIS)
      SKIP PROCESSES       > NO   <      (YES..  NO,  or ASIS)
      REMOTE SUBMIT        > NO   <      (YES..  NO,  or ASIS)
      DEBUGGING OPTIONS    > NO   <      (YES..  NO,  or ASIS)
```

The most important additional input options are:

| | |
|---|---|
| `SESSION (START)` : | ID of the first session to be processed. |
| `YEAR (START)` : | Year of the first session to be processed. |
| `NUMBER OF SESSIONS` : | Specify the number of sessions to be processed in a single BPE run. |
| `SKIP PROCESSES` : | Define the first script to be processed as well as the PIDs of the scripts to be skipped. |
| `SPECIAL PARAMETERS` : | Redefine the variables and ranges of the PCF for this BPE run. |
| `DEBUGGING OPTIONS` : | For debugging purposes the output on the screen while running the PCS and the print out of the scripts into the log files may be more detailed. In addition the autopurge of all temporary files written by the PCS may be prevented. |

For more details we refer to the corresponding help panels.

### 22.7.6 BPE SERVICES

The two menu items Menu 6.5.1 and Menu 6.5.2 are handy tools if you are interested in setting up a BPE run, where new stations (e.g., in a growing permanent network or at the very beginning of the processing of a new campaign) are automatically included in the processing. In this case the most important part is, that of good a priori values for the *geocentric* station coordinates of the new sites have to be known. Especially for the station(s) you would like to fix or heavily constrain in the final solutions good geocentric coordinates in the ITRF (see Chapter 11) must be available.

Menu 6.5.2 is used to check the existence of good a priori coordinates in the user-specified coordinate file for each station given in a list of RINEX files (typically the RINEX files of one session). If there are no sites in the session with good a priori coordinate information, the program CRDRNX (Menu 6.5.2) will generate a list of nearby IGS sites for which RINEX files should be downloaded. These additional RINEX files from IGS sites may then be used to form baselines and to determine a set of coordinates in the ITRF for all sites with coordinates of yet insufficient quality. The list of IGS sites may be used in connection with Menu 2.0.1 to download the RINEX data automatically.

After having transferred all RINEX files of a session (including the additional IGS sites) into the Bernese observation file format, the program CRDCHK (Menu 6.5.1) may be used to generate a list of baselines suited to improve the a priori coordinates of the new sites. This list of baselines may finally be used as input into a script that does a baseline-wise processing from program SNGDIF up to program GPSEST in order to obtain good a priori coordinates.

## 22.8 BPE Scripts

The main function of the BPE is to run scripts. These scripts are written in the script language native to the platform the BPE is running on. For UNIX, BPE scripts are written using the Bourne shell (`sh`), for VMS they are written using the Digital Command Language (DCL). This section will describe how these scripts operate.

### 22.8.1 Skeleton Script

Although there is no real restriction on what is contained in a script that is run by the BPE, there are a few tasks that must be performed in any case. Even if the script does not run any Bernese program(s), the following skeleton structure should be used. This is because the script must make entries into the protocol file in order to let the BPE know that the script has finished. The basic skeleton for a UNIX script is shown below (comment lines start with the # character):

```
# The first task for the script is to execute the header script.  The
# name of the header script will be passed in as the first parameter.
# The header script will set variables and make entries to the protocol
# file.  It will also change to the temporary working directory.
#
if [ "$1" = "" ]
then
#
#   set variables for testing here
#
    TEST_START_DIR=`pwd`
    cd $U/WORK
    TESTING="YES"
    export TESTING
else
    TESTING="NO"
    export TESTING
    . "$1"
    seterr
fi
#
# Now we set up the Bernese menu system to run in non-interactive mode.
. $X/SCRIPT/BEG_MENU
#
# Set standard variables in the DAT151__.PAN.
. $X/SCRIPT/SET_SESS
#
BODY OF THE SCRIPT GOES HERE
#
# The Bernese menu system is taken out of interactive mode.
. $X/SCRIPT/END_MENU
#
# Terminate the script and update the protocol file.
if [ "$TESTING" = "YES" ]
then
    cd "$TEST_START_DIR"
else
    . $X/SCRIPT/DO_TAIL
    seterr
fi
```

For the VMS system the corresponding skeleton script looks as follows:

```
$! The first task for the script is to execute the header script.  The
$! name of the header script will be passed in as the first parameter.
$! The header script will set variables and make entries to the protocol
$! file.  It will also change to the temporary working directory.
$!
$   @'P1' 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'
$!
$! Now we set up the Bernese menu system to run in non-interactive mode.
$   @X:[SCRIPT]BEG_MENU 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'
$!
$! Set standard variables in the DAT151__.PAN.
$   @X:[SCRIPT]SET_SESS 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'
$!
BODY OF THE COMMAND FILE GOES HERE
$!
$! The Bernese menu system is taken out of interactive mode.
$   @X:[SCRIPT]END_MENU 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'
$!
$! Terminate the command file and update the protocol file.
$   @X:[SCRIPT]DO_TAIL 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'
```

A list of variables which are set by the PCS and are available within the scripts follows below:

| | | |
|---|---|---|
| `CAMPAIGN` | : | Name of the campaign. |
| `CAMP_PTH` | : | Path of the campaign (e.g., `DOCU42_1`). |
| `CAMP_DRV` | : | Drive letter of the campaign directory (simulated by links on UNIX systems — e.g., `P:`). |
| `OPT_DIR` | : | Option directory containing the panels with the input parameters. |
| `PID` | : | Process ID of this script in the PCF in the format `xxx` for normal and `xxx_yyy` for parallel scripts. |
| `SUB_PID` | : | Sub-PID of a parallel script. |
| `PRT_FILE` | : | Name of the protocol and log file. |
| `SCRIPT` | : | Name of the script file. |
| `TASKID` | : | Task ID (first two characters of the protocol file). |
| `PRIORITY` | : | Priority level specified in the PCF for this script. |
| `CPU` | : | Nick name of the CPU (taken from the file `$U/WORK/PCFCTL.CPU` where the script is running. |
| `YEAR` | : | Year of the day being processed (2 characters). |
| `YR_4` | : | Year of the day being processed (4 characters). |
| `SESSION` | : | ID of session being processed (4 characters). |
| `DAYYEAR` | : | Day of year of the day being processed. |
| `DAY` | : | Day of month of the day being processed. |
| `MONTH` | : | Month of the day being processed. |
| `GPSWEEK` | : | GPS week of the day being processed. |
| `DAYWEEK` | : | Day of the GPS week of the day being processed. |
| `YEARM3 ...YEARM1, YEARP1 ...YEARP3` | : | Year for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed (2-character). |
| `YR_4M3 ...YR_4P3` | : | Year for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed (4-character). |

| | | |
|---|---|---|
| SESSM3 ...SESSP3 | : | Session ID for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed (4-character). |
| DAYYM3 ...DAYYP3 | : | Day of year for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed. |
| MONTM3 ...MONTP3 | : | Month for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed. |
| DAYMM3 ...DAYMP3 | : | Day of month for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed. |
| GPSWM3 ...GPSWP3 | : | GPS week for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed. |
| DAYWM3 ...DAYWP3 | : | Day of GPS week for the days $-3 \ldots -1$ resp. $+1 \ldots +3$ around the day being processed. |
| V_X, V_O, V_Z, V_U, V_V, V_W, V_PLUS, V_MINUS, V_START, V_END | : | Input variable specified when starting the PCS. |
| V_xxxxxxxx | : | Additional user-specified PCF variables. |
| CPUFIL | : | Name of the CPU file (usually PCFCTL.CPU). |
| PARAM1 ...PARAM6 | : | Parameters given in the PCF file for this script. |

### 22.8.2 The RUN_PGMS Script

To run a Bernese program from within a script/command file, the RUN_PGMS script must be used. This script is kept in the $X/SCRIPT area. To use the script, the user sets an environment variable named PGMNAM to the name of the program to be run, and then starts the RUN_PGMS script. Below is an example for running the Bernese program GPSEST:

<u>UNIX Version:</u>

```
PGMNAM="GPSEST"
. $X/SCRIPT/RUN_PGMS
```

<u>VMS Version:</u>

```
$ PGMNAM == "GPSEST"
$ @X:[SCRIPT]RUN_PGMS
```

The user then has to use Menu 6.1 to correctly set any input fields in the panels (in the panel directory to be used by the script) that may be required for program GPSEST (all panels named DAT45*.PAN).

### 22.8.3 The PUTKEYWE Script

When a script needs to update a value in a panel, it must use the PUTKEYWE script, which is in the $X/SCRIPT area. This script will replace data input fields in Bernese panels. Each panel input field is referenced by a keyword. The keywords are given on the far right side of the panel after column 80. Below is an example of using the PUTKEYWE script:

UNIX Version:

```
pp1=U:/PAN/DAT43___.PAN
pp2=STRATEGY
pp3=MANUAL
. $X/SCRIPT/PUTKEYWE
```

VMS Version:

```
$  pp1 == "U:[PAN]DAT43___.PAN"
$  pp2 == "STRATEGY"
$  pp3 == "MANUAL"
$  @X:[SCRIPT]PUTKEYWE
```

Three environment variables must be set before calling the PUTKEYWE script:

| | | |
|---|---|---|
| pp1 | – | The full path and name of the panel to update. On UNIX systems the linked directory name must be used, i.e. U:. |
| pp2 | – | Name of the keyword to update. |
| pp3 | – | Value to be placed in the data input field referenced by the keyword in pp2. If there is a space contained in the value, then double or single quotes must surround the value under UNIX. |

After running the RUN_PGMS script a variable $JOBNUM is available. It contains the number of the program output, e.g., if the program CODSPP generated the output file CODSPP.L12 the variable $JOBNUM is set to “12”.

## 22.9 BPE Special Programs

There are some utility programs available in the `$XB` directory that are run outside of the Bernese menu system. The use of these programs is described in the following sections.

### 22.9.1 GPSWIND

This program is used to set the time window for a session in the GPSEST panel `DAT4512_.PAN`. It will read from standard input the campaign name, the year and session (each on a separate line) and will update the panel `U:/PAN/DAT4512_.PAN` with the time window defined in the session definition file (`$P/TST_CAMP/DATPAN/DAT132__.PAN`, see Menu 1.3). Here is the UNIX example:

```
erde[jjohnson]:V42-$ cd $U/WORK
erde[jjohnson]:V42-$ cat U:/PAN/DAT4512_.PAN
```

```
 4.5-1.2  |            PARAMETER ESTIMATION: OBSERVATION WINDOWS

                   START DATE                    END DATE

              yy mm dd     hh mm ss        yy mm dd     hh mm ss

            >          < >          <    >          < >          <
```

```
erde[jjohnson]:V42-$ cat P:/TST_CAMP/DATPAN/DAT132__.PAN
```

```
  1.3-2   |                 CAMPAIGNS: SESSION DEFINITION

 SESSION NUMBER          START DATE                    END DATE

     nnnn           yy mm dd     hh mm ss        yy mm dd     hh mm ss

    > ???0 <       >          < > 00 00 00 <   >          < > 23 59 59 <
```

```
erde[jjohnson]:V42-$ echo TST_CAMP > tmp.inp
erde[jjohnson]:V42-$ echo 95 >> tmp.inp
erde[jjohnson]:V42-$ echo 2120 >> tmp.inp
erde[jjohnson]:V42-$ XB:/GPSWIND < tmp.inp
erde[jjohnson]:V42-$ cat U:/PAN/DAT4512_.PAN
```

```
 4.5-1.2  |            PARAMETER ESTIMATION: OBSERVATION WINDOWS

                   START DATE                    END DATE

              yy mm dd     hh mm ss        yy mm dd     hh mm ss

            > 95 07 31 < > 00 00 00 <   > 95 07 31 < > 23 59 59 <
```

### 22.9.2 PRSLIN

The PRSLIN program in `$XB` may be used to get the root name of a file without the extension. This is useful to remove the path and extension from a file name. For example on a UNIX platform:

```
sakic[jjohnson]:V42-$ cd $U/WORK
sakic[jjohnson]:V42-$ echo P:/TST_CAMP/OUT/00961110.002 > inp
sakic[jjohnson]:V42-$ $XB/PRSLIN < inp > out
sakic[jjohnson]:V42-$ basename=‘cat out‘
sakic[jjohnson]:V42-$ echo $basename
00961110
sakic[jjohnson]:V42-$
```

### 22.9.3 PRSLINF

This program is similar to `PRSLIN` except that the extension of the file name is also returned. For example:

```
sakic[jjohnson]:V42-$ cd $U/WORK
sakic[jjohnson]:V42-$ echo P:/TST_CAMP/OUT/00961110.002 > inp
sakic[jjohnson]:V42-$ $XB/PRSLINF < inp > out
sakic[jjohnson]:V42-$ basename=‘cat out‘
sakic[jjohnson]:V42-$ echo $basename
00961110.002
sakic[jjohnson]:V42-$
```

## 22.10 BPE Example

Together with the *Bernese GPS Software* Version 4.2 a BPE example is distributed to allow the user to gain insight into the working of the BPE.

As part of the installation procedure the example PCF file `DOCU42_1.PCF` as well as the corresponding example scripts and option directories are copied from the general `$X` area to the user-specific `$U` area (e.g., `$X/PCF/DOCU42_1.PCF` is copied to `$U/PCF/DOCU42_1.PCF`). After a successful installation (including the BPE part) it should be possible for the user to start the processing of the example campaign `DOCU42_1` (see Chapter 4) with the PCF `DOCU42_1.PCF` right away. The data files for the example campaign `DOCU42_1` are available through anonymous ftp at the AIUB (see Chapter 4). Please read the file `README.TXT`, available in `ftp://ftp.unibe.ch/aiub/BSWUSER/EXAMPLES/`, carefully before starting with the transfer of the example files. You should create a campaign ( Menu 1.1 ) and the corresponding campaign directory and sub-directories ( Menu 1.2 ) according to the steps explained in Chapter 4 before downloading the BPE example data.

The example BPE is deduced from our EUREF processing procedure and shows an elaborate sequence of tasks which might look very complex at first sight. The main results of the BPE are three separate solutions computed with free and fixed ambiguities and with different elevation cut-off angle. The reader may get some ideas on how to construct BPEs for his own applications. He is encouraged to browse through the individual scripts located in `$U/SCRIPT`.

The example PCF file is included here and the most important steps are described below:

```
#
# Procedure Control File (PCF)
# All comment lines start with a #
# Comments: Example PCF to be used to process the example campaign DOCU42_1
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
001 DOCU_COP EUROCLUS          any      1
002 PRETAB   EUROCLUS          any      1 001
003 ORBGEN   EUROCLUS          any      1 002
004 RXOBV3   EUROCLUS          any      1 001
005 CODCHK   EUROCLUS          any      1 004
006 CODSPP   EUROCLUS          any      1 003 005
007 SNGDIF   EUROCLUS          any      1 006
008 MAUPRP   EUROCLUS          any      1 007
009 GPSEDT   EUROCLUS          any      1 008
010 ADDNEQ   EUROCLUS          any      1 009
011 COMPAR   EUROCLUS          any      1 010
012 GPSBAS   EURO_BAS          any      1 011
013 ADDNEQ   EURO_BAS          any      1 012
014 ADDNEQ2  EURO_BAS          any      1 013
015 COMPAR   EURO_BAS          any      1 014
016 GPSEST   EURO_IOF          any      1 015
017 GPSEST   EUROFREE          any      1 016
018 COMPAR   EUROFREE          any      1 017
019 GPSQIFAL EURO_QIF          any      1 018
020 GPSQIF_P EURO_QIF          CPUQIF   1 019
021 QIFXTR   EURO_QIF          any      1 020
022 GPSEST   EURO_IOX          any      1 021
023 GPSEST   EURO_FIX          any      1 022
024 COMPAR   EURO_FIX          any      1 023
025 GPSEST   EURO_FXB          any      1 024
026 COMPAR   EURO_FXB          any      1 025
027 DOCU_DEL EUROCLUS          any      1 026
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
005                       SKIP
#14                       SKIP
019                       $tmp1
020                       PARALLEL $tmp1
#
# That's it
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_O      ORBIT AND ERP INPUT FILE NAME            R3                2
V_X      BASELINE, AMB.FREE., 10DEG WGT           EO                2
V_Z      BASELINE, AMB.FREE., 10DEG WGT, RESRMS   EB                2
V_U      NETWORK, AMB.FREE, 15DEG                 EN                2
V_V      NETWORK, AMB.FIX., 15DEG                 EF                2
V_W      NETWORK, AMB.FIX., 10DEG WGT.            EW                2
V_PLUS   PLUS  DAYS                               +0                2
V_MINUS  MINUS DAYS                               -0                2
```

The first section lists the scripts to be executed paired with the option directories to be used with these scripts. Following is a section showing the parameters to be passed to the scripts and finally there is a section defining the variables used in all scripts called by the PCF. All of the scripts shown in the PCF are located in `$U/SCRIPT` while the option directories are located in `$U/OPT`.

Notice that a CPU is explicitely specified for the script GPSQIF_P at PID 020. This will be the host this script will execute on. An entry in the $U/WORK/PCFCTL.CPU file must exist with this CPU name, for example:

```
# Process control CPU information
CPU      TRUE_NAME                        SPEED    NCPU SF   IDLE MAXJ JOBS
8******* 30****************************** 8******* 4*** 4*** 4*** 4*** 4***
CPU1     sakic                            FAST        1  100  100    2    0
CPUQIF   sakic                            FAST        1  100  100    2    0
```

In the above case, CPUQIF and CPU1 use the same host and specifying CPUQIF specifically in the PCF file seems to have no advantage. In fact, CPUQIF could be changed to any in the PCF file and the entry CPUQIF could be taken out of PCFCTL.CPU for the example CPU file shown above. However, there are cases where it may be desirable to specify a specific CPU for a script when more than one host is available for processing and one of the hosts is faster than the others. The GPSQIF_P script is very CPU intensive and we may want to make sure that the script is run on a fast CPU.

After copying a few files the processing starts very straight forward with the generation of a standard orbit (PRETAB and ORBGEN). Then the code and phase pre-processing is performed (RXOBV3, CODCHK (this program — PID 005 — is skipped, see second area of the PCF), CODSPP, SNGDIF, MAUPRP).

This brings us to PID 009. This step is called GPSEDT, which means that the GPSEST residuals are checked in order to remove bad phase double-difference observations. For this procedure the programs GPSEST, RESRMS, and SERVOBS are used: GPSEST is run in a baseline mode saving *both* the double-difference $L_3$ residuals in residual files and the normal equations in NEQ files. The residual files are checked by program RESRMS followed by the program SERVOBS, which marks the detected outliers in the single difference files according to the edit file generated by RESRMS. The NEQ files are used with program ADDNEQ (PID 010) to get a network solution based on "uncleaned" (no RESRMS) data. This solution is labeled "E0_".

The script at PID 011 runs program COMPAR to extract a short solution summary. Together with the summaries from the other COMPAR runs it will be used by the last script of the PCF, DOCU_DEL, to generate a protocol file.

Now we are at PID 012 with the script called GPSBAS. Here the single difference files are processed baseline by baseline saving the NEQ files. These NEQs are combined with ADDNEQ to give a "cleaned" (after RESRMS) network solution. This solution is labeled "EB_".

For comparison the normal equations (extension NQ0) from the same GPSEST run are combined at PID 014 using the new ADDNEQ2. The solution is labeled "EB2". Those users for which ADDNEQ2 (Fortran 90) is not available may skip this script by removing the comment character in the second section of the PCF example.

At PID 016 we use GPSEST to estimate an ionosphere model to be used for the QIF ambiguity fixing in PID 020.

PID 017 is a GPSEST run using all single-difference files in one run with "CORRECT" correlations (see Panel 4.5–2) to generate an ambiguity-free network solution. This solution is labeled "EN_".

PID 019/020 are examples for a parallel processing. The ambiguity fixing using the QIF strategy with the ionosphere model from PID 016 is done baseline-wise and more than one baseline may be processed in parallel (depending on the entries in the CPU file $U/WORK/PCFCTL.CPU).

At PID 022 a final ionosphere model is computed, as in PID 016, with ambiguities introduced.

The GPSEST solutions of PID 023 and 025 are similar to those of PID 017, except that the ambiguities are now fixed (PID 023). The elevation limit in the other solution PID 025 is set to $10^o$ and elevation-dependent weighting is activated. The (PID 023 solution is labeled "EF_", the PID 025 solution "EW_".

For those programs, where extraction programs are available, the extraction program is used (CODXTR, MPRXTR, GPSXTR, RESRMS summary) right after the program has been run. E.g., in the MAUPRP script not only MAUPRP but also the extraction program MPRXTR is run. The same is true for the programs CODSPP, GPSEST, and ADDNEQ. All these extraction files are concatenated in the DOCU_DEL script (PID 027) to form a "nice" protocol/overview of the processing. Two example protocols are available in the OUT directory of the example campaign DOCU42_1.

The following PCF variables are used for solution identifiers:

| | |
|---|---|
| V_O ... | gives the first two characters of the standard orbit to be used. The orbit file name should be of the form $O_$Y$D1. The first script (DOCU_COP) can be used to rename any input orbit file name to the naming convention given above. Currently the DOCU_COP script does practically nothing (only deleting some files). |
| V_X ... | (EO above) is used to label the results from the first network solution in baseline mode before RESRMS has been run. |
| V_Z ... | (EB above) is used to label ambiguity-free results based on "cleaned" (after RESRMS). The (hard-coded) "_" is used to identify the solution using the program ADDNEQ. The EB2 solution is an analogue solution using the Fortran 90 program ADDNEQ2. |
| V_U ... | (EN above) labels the ambiguity free network solution computed with an elevation cut-off angle of $15^o$. |
| V_V ... | (EF above) labels the ambiguity fixed network solution computed with an elevation cut-off angle of $15^o$. |
| V_W ... | (EW above) labels the ambiguity fixed network solution computed with an elevation cut-off angle of $10^o$ and with elevation-dependent weighting. |

The example PCF file covers quite a lot of programs and different applications of these programs. It shows an example of the parallel capabilities of the BPE (PIDs 019/020) and also covers the screening of residual files for outliers, the fixing of ambiguities, the use of baseline-wise and correct correlation, and different elevation cut-off angles.