**GBT MEMO 196**

# Fitting Models to Simulated Rangefinder Data

Don Wells*

April 6, 1999

**Abstract**

Simulated rangefinder data is fitted to estimate rangefinder coordinates, zero points and backprism offsets, and to estimate coordinates of target retroreflectors. The translation and tilt of trusses with retroreflectors attached are estimated from rangefinder data, for the cases of differential backup-structure pointing corrections and subreflector pose determination. The differential pointing technique is advocated for Phase I implementation early in 2000. The simulation code shown here executes faster than the data acquisition process being simulated, so this code is a candidate for production use.

## Contents



*mailto:dwells@nrao.edu

# 1 Introduction

A total of 20 laser rangefinders [Par97, PPB92, PP90, Cre98, Par96, PPB95] have been built to be used in the control system of the Green Bank Telescope. They are to be allocated as follows:

- 12 of them are to be mounted on stable piers, arranged in a ring with 120 m radius, at about the height of the pintle bearing, in order to provide a ground reference for pointing the beam of the telescope
- 4 of them are to be mounted around the Gregorian feedroom [Wel98c] in order to measure the primary mirror surface and to measure the position of the subreflector
- 2 of them are to be mounted on the lower part of the feedarm [Wel98c] in order to provide the lateral measurement of the primary mirror
- 2 of them are spares

In the Fall of 1998, during discussions of rangefinder data processing strategies for generating pointing corrections, it was pointed out that no least-squares model-fitting software for processing such data had yet been built. The author decided to construct some proof-of-concept implementations in the form of simulations. The simulations consist of datasets generated by Perl [WCS96] code and fitted by Gaussfit [JFMM88] models; they are a proof-of-concept because if real rangefinder data were substituted for the simulated data, the models would produce analogous answers for the real GBT. As we will see, these model-fitting programs have turned out to execute faster than the real-time data acquisition which they are simulating, and so they are not just proofs-of-concept, but could actually be practical for production use.

Model fitting has a fundamental advantage over surveyor-style algorithms for rangefinder data: it can incorporate appropriate mathematics for modelling index of refraction variations (e.g. empirical lapse rates), thermal expansion effects and motion of targets. For example, the model shown in Figure 5 (p.17) solves for thermal expansion and incorporates the angular velocity of the target truss structure.

There has been some prior fitting of models to real (not simulated) rangefinder data. Cotton [Cot97] conducted experiments in which four prototype rangefinders measured ranges to three retroreflectors mounted on the backup structure of the 140 foot telescope [Par96]. Cotton's report discusses prototype data acquisition code which could probably still operate, because it is believed that there have been no significant protocol changes for the rangefinders since his work. Cotton coded his analysis implementation in Fortran 77, calling ODRPACK [BBRS92] subroutines to fit geometric models to the range measurements.

A Fortran program which fits simulated noisy range data for the case of four or more GBT feedarm rangefinders measuring the XYZ coordinates of a GBT primary mirror retroreflector prism was built by F. Schwab some years ago. This work was not published, but Schwab [Sch90] did publish the error analysis for the problem.

Goldman has used the STAR*NET[1] surveyor's reduction program to simulate determination of the subreflector position and orientation; his simulations are similar to the author's Gaussfit simulations discussed in Section 4.3. He used the same technology to analyze the case of retrospheres [Gol96] to be mounted beneath the elevation bearing housings of the GBT [Gol97]. Goldman [citation?] previously simulated the use of the "triplet" retroreflectors around the edge of the primary mirror to aid in transferring ground coordinate system to the feedarm rangefinders.

[1] Starplus Software, Inc., Oakland, CA

## 2 The "plane" problem

The GBT will have 12 rangefinders mounted on stable piers, arranged in a ring with 120 m radius, at about the height of the pintle bearing. The piers have been installed to such accuracy that the rangefinders can be regarded as being in a plane, and 2-D regressions can be used instead of 3-D for modeling inter-rangefinder data. Currently 7 of the 12 rangefinders are operational. Of the $n(n-1) = 42$ ranges which could be measured between 7 rangefinders, only 32 are currently unblocked.

Ranges measured by the rangefinders contain zero-point offsets. Each rangefinder has a prism mounted in its baseplate which can be observed by orienting the steering mirror at 45° to the baseplate. The distance between the steering mirror and the reflection point of the baseplate prism can be determined, so that ranges to this zero-point prism can be subtracted from all other ranges to correct the zero point offset so that ranges will be referred to the axis-intersection point in the surface of the steering mirror.

Each rangefinder has a retroreflector prism mounted on the back of its steering mirror; these will be referred to as 'backprisms'. If the mirror is oriented so that its backprism points toward another rangefinder, that rangefinder can measure the distance to the first rangefinder; the range must be corrected for the distance between the surface of the mirror and the reflection point of the back-prism.

In practice there are several other details about the rangefinders which must be considered. The ranges are really phase measurements, and there is a cycle count ambiguity, with period about 10 cm. The geometry of each retroreflector prism has a manufacturing error which can be calibrated in the laboratory to produce the "glass correction". The piers have Kelvin mounts which permit interchanging rangefinders, and so the vector offset between the Kelvin mount and steering mirror of a rangefinder should be calibrated. These operational details have been ignored in these simulations.

Three models will be simulated in this section: (1) solving for rangefinders positions from range data, (2) same as (1), but with unknown zero-points and (3) same as (2), but with unknown back-prism calibrations as well. The second and third simulations are provided to demonstrate that these calibration values can be inferred from internal evidence in datasets if necessary; in practice it is preferable to determine these numbers by precision measurements in the laboratory so that the full statistical weight of the data can be brought to bear on the determination of the parameters required for GBT operations.

## 2.1 Relative 2-D positions of rangefinders in a plane

In this first simulation 7 rangefinders range on each other, and we assume that all corrections have been applied to the data so that the ranges represent true steering-mirror to steering-mirror distances. The 7 instruments (ZY110, 111, 112, 101, 102, 103 and 104 as identified in Figure 2 on p.4 of [Wel98c]) are assumed to be 120000 mm from the pintle bearing, spaced by $\frac{360}{12} = 30°$ in azimuth, with ZY101 5° East of North. Only 32 combinations of distance between them have been computed (the ones which were unblocked as of 1999-03-16[2]) and Gaussian noise[3] with amplitude 0.1 mm is added to the distances to produce the input data table:

| Simulated range data (`rfsPlane1`) | | | | | |
|---|---|---|---|---|---|
| `r1` | `r2` | `range` mm | $\sigma^2_{\texttt{range}}$ | `noise` mm | $\Delta_{\texttt{range}}$ mm |
| 110 | 111 | 62116.49 | 0.01 | -0.08 | 0.00 |
| 110 | 112 | 119999.90 | 0.01 | -0.10 | 0.00 |
| 110 | 101 | 169705.59 | 0.01 | -0.04 | -0.04 |
| 111 | 110 | 62116.53 | 0.01 | -0.04 | 0.04 |
| 111 | 112 | 62116.49 | 0.01 | -0.08 | 0.01 |
| 111 | 101 | 119999.96 | 0.01 | -0.04 | -0.02 |
| 111 | 102 | 169705.64 | 0.01 | 0.01 | 0.01 |
| 112 | 110 | 119999.81 | 0.01 | -0.19 | -0.08 |
| 112 | 111 | 62116.49 | 0.01 | -0.08 | 0.01 |
| 112 | 101 | 62116.53 | 0.01 | -0.04 | -0.12 |
| 112 | 102 | 120000.09 | 0.01 | 0.09 | 0.02 |
| 112 | 103 | 169705.88 | 0.01 | 0.25 | 0.12 |
| 112 | 104 | 207846.11 | 0.01 | 0.01 | 0.02 |
| 101 | 110 | 169705.71 | 0.01 | 0.08 | 0.08 |
| 101 | 111 | 120000.08 | 0.01 | 0.08 | 0.11 |
| 101 | 112 | 62116.56 | 0.01 | -0.01 | -0.08 |
| 101 | 102 | 62116.45 | 0.01 | -0.12 | -0.09 |
| 101 | 103 | 120000.06 | 0.01 | 0.06 | 0.04 |
| 101 | 104 | 169705.57 | 0.01 | -0.06 | -0.01 |
| 102 | 111 | 169705.53 | 0.01 | -0.10 | -0.10 |
| 102 | 112 | 120000.20 | 0.01 | 0.20 | 0.14 |
| 102 | 101 | 62116.54 | 0.01 | -0.03 | 0.00 |
| 102 | 103 | 62116.60 | 0.01 | 0.03 | -0.02 |
| 102 | 104 | 120000.09 | 0.01 | 0.09 | 0.03 |
| 103 | 112 | 169705.68 | 0.01 | 0.06 | -0.08 |
| 103 | 101 | 119999.97 | 0.01 | -0.03 | -0.06 |
| 103 | 102 | 62116.60 | 0.01 | 0.03 | -0.02 |
| 103 | 104 | 62116.66 | 0.01 | 0.09 | -0.00 |
| 104 | 112 | 207846.05 | 0.01 | -0.05 | -0.04 |
| 104 | 101 | 169705.61 | 0.01 | -0.01 | 0.03 |
| 104 | 102 | 120000.05 | 0.01 | 0.05 | -0.00 |
| 104 | 103 | 62116.64 | 0.01 | 0.07 | -0.02 |

These range data are fitted using the Gaussfit model shown in Figure 1, in order to solve for the 14 X and Y corrections `prd[ranger,axis]` to the a priori coordinates of the 7 rangefinders `pr[ranger,axis]`.[4] The $\Delta_{\texttt{range}}$ column above gives the range residuals from the model fit, and they appear to be consistent with the 0.1 mm RMS noise (column `noise` above) which was added to the distances computed for the simulated dataset.[5]

[2] R.Creager private communication

[3] See the fragment of Perl source code on p.29 for the algorithm which generates the noise for the simulated dataset.

[4] Solving for corrections rather than for the coordinates themselves is called an 'adjustment' in surveyor parlance.

[5] In this simulation, whose noise is generated numerically, we can be certain that the range residuals have a "normal" (Gaussian) distribution, and that standard least-squares (a.k.a. "$L_2$-norm") is the appropriate technique for fitting models.

```
/* rfsPlane1Model.gf --- N rangefinders in a plane

   This model does a LS fit of inter-rangefinder measurements between
   N rangefinders in a plane without zero point errors and
   without backprism errors.

   The sum of the corrections to rangefinder coordinates is constrained
   to zero, and rotation of the solution is constrained to zero.

   This rfsPlane1Model.gf was generated Mon Apr  5 17:40:00 EDT 1999
*/

constant        pr[ranger,axis];
parameter       prd[ranger,axis];
data            r1, r2;
observation     range;

main()
{
        variable i, sum, rad, naxes=2;
        variable nr, lr1[30], prdsum[2], found;

        nr = 0; prdsum[0] = 0; prdsum[1]=0;
        while (import()) {
              sum = 0.0; for (i = 0; i < naxes; i = i + 1)
                  sum = sum + ((pr[r1,i]+prd[r1,i]) -
                               (pr[r2,i]+prd[r2,i]))^2;
              rad = sqrt(sum);
              export(range - rad);

              /* accumulate sums of prd[,] over rangers: */
              found = 0; for (i = 0; i < nr; i = i + 1) {
                  if (r1 == lr1[i]) { found = 1; }
              }
              if (found == 0) {
                  lr1[nr] = r1;
                  nr = nr + 1;
                  for (i = 0; i < naxes; i = i + 1)
                      prdsum[i] = prdsum[i] + prd[r1,i];
              }
        }
        for (i = 0; i < naxes; i = i + 1)
                exportconstraint(prdsum[i]);     /* delta-XY translation */
        exportconstraint(prd[110,1]-prd[104,1]); /* 110-->104  rotation */
}
```

Figure 1: Gaussfit model for the rangefinders-in-a-plane case

The model shown in Figure 1 solves for the 14 parameters `prd[ranger,axis]`. The independent variables ("`data`" declaration in Figure 1) of the regression are rangefinder indices `r1` and `r2`, and the dependent variable ("`observation`") is `range`. The `while` loop reads the data table shown above by calling function `import()`, which sets `r1`, `r2` and `range`. The algorithm sums the squares of the coordinate differences between the two rangefinders and takes the square root to get the computed range. It then calls function `export()` with the difference between the observed and computed ranges; this is the equation-of-condition formed from the observation, and Gaussfit will adjust the `prd[,]` parameters until the sum of the squares of the these O-C differences is minimized.[6]

It is necessary to supply three constraints on the 14 unknowns `prd[,]`, because the input range data is consistent with any arbitrary translation and rotation of the set of rangefinder coordinates. The implementation shown here constrains translation by forming the sums of the `prd[,axis]` values and then calling function `exportconstraint()` to assure that the means of the coordinate corrections will be zero.[7] It constrains rotation by calling function `exportconstraint()` with the difference between the corrections to the Y-coordinates of the two rangefinders which are farthest apart in X.[8] Even with these constraints, there are still four possible X and Y mirror-symmetry solutions to the coordinate corrections; Gaussfit delivers the one which is near the initial parameter values. In practice the absolute positions of certain monuments and the true azimuth between several monuments will be known by geodetic calibrations, and these values can be used as the necessary constraints instead of the arbitrary constraints used in this simulation.

The parameter values produced by this fit are the `prd` column in the table below:

| Rangefinder coordinates (`rfsPlane1`) | | | | | |
|---|---|---|---|---|---|
| `ranger` | `axis` | `pr` mm | `prq` mm | `prd` mm | $\sigma_{\texttt{prd}}$ mm |
| 110 | 0 | -119543.36 | 0.0 | -0.00 | 0.14 |
| 110 | 1 | 10459.69 | -1.0 | -0.21 | 0.05 |
| 111 | 0 | -98298.25 | 0.0 | -0.13 | 0.05 |
| 111 | 1 | 68829.17 | 0.0 | 0.75 | 0.06 |
| 112 | 0 | -50714.19 | 0.0 | -0.41 | 0.02 |
| 112 | 1 | 108756.93 | 0.0 | 0.94 | 0.07 |
| 101 | 0 | 10460.69 | -2.0 | -2.39 | 0.03 |
| 101 | 1 | 119543.36 | 0.0 | 1.25 | 0.07 |
| 102 | 0 | 68827.17 | 2.0 | 1.71 | 0.03 |
| 102 | 1 | 98303.25 | -5.0 | -3.37 | 0.05 |
| 103 | 0 | 108755.93 | 1.0 | 1.06 | 0.07 |
| 103 | 1 | 50715.19 | -1.0 | 0.86 | 0.06 |
| 104 | 0 | 119543.36 | 0.0 | 0.17 | 0.12 |
| 104 | 1 | -10456.69 | -2.0 | -0.21 | 0.05 |

Note that `prd[104,1]` has the same value as `prd[110,1]`; this is the rotation constraint on the solution. The $\sigma_{\texttt{prd}}$ values are the formal uncertainties of the correction parameters; they are fairly uniform and symmetrical across the array of rangefinders (e.g., the X coordinate formal errors of both ZY104 and ZY110 are larger,

However, real datasets often contain wild "outlier" values. In the case of the rangefinders and their cycle ambiguity, it is entirely possible for ranges to be wrong by about 10 cm. Gaussfit is able to cope with these non-Gaussian-statistics situations, by enabling one of its "robust estimation" ("Huber-type") modes, which automatically pay less attention than would least-squares to observations with large residuals; see [JFMM88, Section 2.4 "Robust Estimation"] for the details and environment file options.

[6] In this section the author concentrates on the least squares model language; in Section 4.3 (p.22) the author illustrates the shell command, the environment file and multiple parameter files for a Gaussfit modeling problem.

[7] Early versions of this code set the X and Y coordinate of the first rangefinder to nominal values. This worked but biased the distributions of residuals and formal errors.

[8] A more elegant, but more complicated, technique for constraining rotation of the solution would be to compute the net moment of the X-Y correction vectors about the centroid of the distribution of rangefinders, and export that quantity as the constraint.

exactly what one would expect after inspecting the geometry). In general, the formal errors of the parameters are comparable to or smaller than the noise ($100\mu$m) in the range data.

The `prq[ranger,axis]` values above are perturbations which were applied to the `pr[,]` values so that we could simulate solving for unknown coordinates. I.e., ranges were computed with true values, but the starting values used for the model fit were perturbed by random numbers. The perturbations are Gaussian with $\sigma$ of 2 mm, and they have been truncated to integers. The reader can see that the `prq` and `prd` columns are correlated, but the X and Y values are offset because means of the `prd` are constrained to zero, and that depends on whatever happens to be the mean of the `prq` values. Presumably the `prd[,]` and `prq[,]` values are related by a 3-term linear transformation (the author has not checked this), representing the fundamental uncertainty about absolute translation and rotation which is inherent in surveying. I.e., we are able to solve for 14 numbers which can reproduce the observed ranges with RMS of order $100\mu$m (see the $\Delta_{\texttt{range}}$ column of the table on p.4), but the real information content of the 14 numbers is only their relative values.

The fourteen parameters which have been estimated by fitting this least-squares model to simulated range data are related. The logic is inescapable: if the coordinates of one rangefinder are perturbed, range data connecting that rangefinder to another rangefinder will imply that the coordinates of the second rangefinder must also be perturbed. Range data connecting the second rangefinder to a third rangefinder will imply that it too must be perturbed. Therefore it is not surprising that each of the fourteen parameters is correlated with almost every other parameter:

| | prd 110,1 | prd 104,1 | prd 111,0 | prd 101,0 | prd 112,0 | prd 102,0 | prd 103,1 | prd 101,1 | prd 112,1 | prd 102,1 | prd 103,0 | prd 104,0 | prd 111,1 | prd 110,0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| prd[110,1] | 100 | 100 | -34 | -41 | 34 | -8 | 49 | -79 | -80 | -24 | 62 | 91 | -12 | -91 |
| prd[104,1] | | 100 | -34 | -41 | 34 | -8 | 49 | -79 | -80 | -24 | 62 | 91 | -12 | -91 |
| prd[111,0] | | | 100 | 7 | -38 | 32 | -25 | 47 | 43 | 50 | -48 | -52 | -65 | 27 |
| prd[101,0] | | | | 100 | -17 | -5 | -46 | 41 | 44 | -5 | -45 | -43 | 22 | 42 |
| prd[112,0] | | | | | 100 | -9 | 12 | -31 | -36 | -17 | 24 | 34 | 23 | -38 |
| prd[102,0] | | | | | | 100 | -17 | 2 | 5 | 57 | -23 | -18 | -25 | -5 |
| prd[103,1] | | | | | | | 100 | -59 | -66 | -12 | 82 | 48 | -26 | -62 |
| prd[101,1] | | | | | | | | 100 | 66 | 11 | -69 | -79 | -8 | 81 |
| prd[112,1] | | | | | | | | | 100 | 6 | -71 | -79 | 1 | 82 |
| prd[102,1] | | | | | | | | | | 100 | -26 | -30 | -50 | 9 |
| prd[103,0] | | | | | | | | | | | 100 | 59 | -1 | -72 |
| prd[104,0] | | | | | | | | | | | | 100 | 6 | -87 |
| prd[111,1] | | | | | | | | | | | | | 100 | 18 |
| prd[110,0] | | | | | | | | | | | | | | 100 |

Note that the correlation of `prd[110,1]` with `prd[104,1]` is 100%; this is simply the rotation constraint on the two Y coordinates. However, this constraint causes the two X coordinates `prd[110,0]` and `prd[104,0]` to be about 90% correlated with the Y coordinates and to be almost 90% correlated with each other. Experience shows that correlations less than about 80% are not serious difficulties in this type of least squares problem. If all 12 rangefinders were available, so that there would be multiple independent chains of ranges from station to station, many of these correlations would be significantly reduced.

The simulation in this section (determination of rangefinder positions with zero points and backprism offsets known) is one of the real problems which we will want to solve in the GBT. However, sometimes we may wish to reduce datasets even though the rangefinder calibration is incomplete. In the next two sections we will demonstrate how the Gaussfit model can be extended to solve such problems in this case of rangefinders-in-a-plane.

## 2.2 Relative 2-D positions and zero points of rangefinders

Suppose that the zero point calibrations of the rangefinders are not known; is it still possible to solve for the rangefinder positions from range data? The answer is *yes*. We simply define 7 more parameters `zero[ranger]`, and subtract them from the ranges computed in the model, as shown in Figure 2. The total number of parameters is $2 \times 7 - 3 + 7 = 18$; as long as we have more equations of condition than parameters,

```
/* rfsPlane2Model.gf --- N rangefinders+zero_points in a plane

   This model does a LS fit of inter-rangefinder measurements between
   N rangefinders in a plane with N zero point errors and
   without backprism errors.

   The sum of the corrections to rangefinder coordinates is constrained
   to zero, and rotation of the solution is constrained to zero.

   This rfsPlane2Model.gf was generated Mon Apr  5 17:40:01 EDT 1999
*/

constant        pr[ranger,axis];
parameter       prd[ranger,axis], zero[ranger];
data            r1, r2;
observation     range;

main()
{
        variable i, sum, rad, naxes=2;
        variable nr, lr1[30], prdsum[2], found;

        nr = 0; prdsum[0] = 0; prdsum[1]=0;
        while (import()) {
              sum = 0.0; for (i = 0; i < naxes; i = i + 1)
                  sum = sum + ((pr[r1,i]+prd[r1,i]) -
                               (pr[r2,i]+prd[r2,i]))^2;
              rad = sqrt(sum) - zero[r1];
              export(range - rad);

              /* accumulate sums of prd[,] over rangers: */
              found = 0; for (i = 0; i < nr; i = i + 1) {
                  if (r1 == lr1[i]) { found = 1; }
              }
              if (found == 0) {
                  lr1[nr] = r1;
                  nr = nr + 1;
                  for (i = 0; i < naxes; i = i + 1)
                      prdsum[i] = prdsum[i] + prd[r1,i];
              }
        }
        for (i = 0; i < naxes; i = i + 1)
                exportconstraint(prdsum[i]);     /* delta-XY translation */
        exportconstraint(prd[110,1]-prd[104,1]); /* 110-->104  rotation */
}
```

Figure 2: Gaussfit model for the rangefinders-in-a-plane+zeroes

least squares problems of this type can be solved.[9] In this case we have almost twice as many observations as unknowns. Of course in solving for zero points we are spreading the weight of the data more thinly to cover more unknowns, and so the formal errors of the rangefinder coordinate parameters are somewhat larger:

| Rangefinder coordinates (`rfsPlane2`) | | | | | |
|---|---|---|---|---|---|
| `ranger` | `axis` | `pr` mm | `prq` mm | `prd` mm | $\sigma_{\texttt{prd}}$ mm |
| 110 | 0 | -119543.36 | 0.0 | -0.44 | 0.24 |
| 110 | 1 | 10455.69 | 3.0 | 1.33 | 0.15 |
| 111 | 0 | -98297.25 | -1.0 | -1.79 | 0.09 |
| 111 | 1 | 68829.17 | 0.0 | -1.46 | 0.10 |
| 112 | 0 | -50716.19 | 2.0 | 1.02 | 0.06 |
| 112 | 1 | 108755.93 | 1.0 | 0.04 | 0.16 |
| 101 | 0 | 10457.69 | 1.0 | 0.04 | 0.05 |
| 101 | 1 | 119543.36 | 0.0 | -1.04 | 0.16 |
| 102 | 0 | 68827.17 | 2.0 | 1.25 | 0.07 |
| 102 | 1 | 98297.25 | 1.0 | 0.36 | 0.10 |
| 103 | 0 | 108754.93 | 2.0 | 1.48 | 0.11 |
| 103 | 1 | 50714.19 | 0.0 | -0.58 | 0.10 |
| 104 | 0 | 119544.36 | -1.0 | -1.56 | 0.21 |
| 104 | 1 | -10460.69 | 2.0 | 1.33 | 0.15 |

As before, the `prq` values are perturbations of the a priori coordinates by quantized 2 mm RMS noise; although the `prq` and `prd` columns are correlated, the X and Y values are offset because means of the `prd` are constrained to zero, and that depends on whatever happens to be the mean of the `prq` values. The formal errors of the 7 zero point parameters `zero[ranger]` are comparable to the 100$\mu$m noise in the range data:

| Zero points (`rfsPlane2`) | | | |
|---|---|---|---|
| `ranger` | `zeroq` mm | `zero` mm | $\sigma_{\texttt{zero}}$ mm |
| 110 | 0.00 | 0.14 | 0.12 |
| 111 | 1.00 | 1.07 | 0.10 |
| 112 | 2.00 | 2.25 | 0.10 |
| 101 | 5.00 | 5.11 | 0.09 |
| 102 | 0.00 | 0.17 | 0.09 |
| 103 | 0.00 | 0.19 | 0.10 |
| 104 | -3.00 | -2.88 | 0.11 |

The `zeroq` values are the quantized 2 mm RMS perturbations of zero point which were used in the computation of the simulated range data. Note that the least-squares model has recovered these zero points with good accuracy.

## 2.3 Relative 2-D positions and zero points and backprism offsets

Suppose that, in addition to not knowing the zero point prism calibrations, we don't even know the backprism offsets; can this problem be solved? Again the answer is *yes*, except that numerical experiments have shown that we must impose one more constraint on the solution. The explanation for this is that the same ranges will be observed if an arbitrary constant is added to all zero points and subtracted from all backprism offsets.

[9] The exceptions to this assertion are the pathological cases; for the "plane" problem being considered here the obvious pathological case would be the seven rangefinders almost in a line, so that positions orthogonal to the line would be indeterminate.

```
/* rfsPlane3Model.gf --- N rangefinders+zero_points+backprisms in a plane

   This model does a LS fit of inter-rangefinder measurements between
   N rangefinders in a plane with N zero point errors and
   with N backprism errors.

   The sum of the corrections to rangefinder coordinates is constrained
   to zero, and rotation of the solution is constrained to zero.
   The sum of the backprism corrections is constrained to zero.

   This rfsPlane3Model.gf was generated Mon Apr  5 17:40:02 EDT 1999
*/

constant        pr[ranger,axis];
parameter       prd[ranger,axis], zero[ranger], back[ranger];
data            r1, r2;
observation     range;

main()
{
        variable i, sum, rad, naxes=2;
        variable nr, lr1[30], prdsum[2], found;
        variable backsum, zerosum;

        nr = 0; prdsum[0] = 0; prdsum[1]=0;
        backsum = 0; zerosum = 0;
        while (import()) {
              sum = 0.0; for (i = 0; i < naxes; i = i + 1)
                  sum = sum + ((pr[r1,i]+prd[r1,i]) -
                               (pr[r2,i]+prd[r2,i]))^2;
              rad = sqrt(sum) - zero[r1] - back[r2];
              export(range - rad);

              /* accumulate sums of prd[,] over rangers: */
              found = 0; for (i = 0; i < nr; i = i + 1) {
                  if (r1 == lr1[i]) { found = 1; }
              }
              if (found == 0) {
                  lr1[nr] = r1;
                  nr = nr + 1;
                  for (i = 0; i < naxes; i = i + 1)
                      prdsum[i] = prdsum[i] + prd[r1,i];
                  backsum = backsum + back[r1];
              }
        }
        for (i = 0; i < naxes; i = i + 1)
                exportconstraint(prdsum[i]);     /* delta-XY translation */
        exportconstraint(prd[110,1]-prd[104,1]); /* 110-->104  rotation */
        exportconstraint(backsum);
}
```

Figure 3: Gaussfit model for rangefinders+zeroes+backprisms

In Figure 3 we see that the zero point and backprism parameters are subtracted from the computed ranges, and that the sum of the backprism offsets is constrained to zero. We now have $2 \times 7 - 3 + 7 + 7 - 1 = 24$ parameters, and even with the blocked lines-of-sight the current array of 7 rangefinders has more equations-of-condition than parameters. However, the uncertainties in the parameters get rather large for such a dataset. Therefore, to make this simulation more interesting, we will do it on the assumption that we have all 12 rangefinders and that they can all see each other, so that we have $12 \times 11 = 132$ equations of condition for the 24 parameters. We get:

| Rangefinder coordinates (`rfsPlane3`) | | | | | |
|---|---|---|---|---|---|
| `ranger` | `axis` | `pr` mm | `prq` mm | `prd` mm | $\sigma_{\texttt{prd}}$ mm |
| 101 | 0 | 10457.69 | 1.0 | 1.33 | 0.04 |
| 101 | 1 | 119543.36 | 0.0 | 0.01 | 0.09 |
| 102 | 0 | 68828.17 | 1.0 | 1.40 | 0.06 |
| 102 | 1 | 98299.25 | -1.0 | -0.92 | 0.08 |
| 103 | 0 | 108755.93 | 1.0 | 1.47 | 0.08 |
| 103 | 1 | 50714.19 | 0.0 | 0.10 | 0.06 |
| 104 | 0 | 119545.36 | -2.0 | -1.55 | 0.09 |
| 104 | 1 | -10458.69 | 0.0 | -0.01 | 0.03 |
| 105 | 0 | 98298.25 | 0.0 | 0.48 | 0.07 |
| 105 | 1 | -68830.17 | 1.0 | 0.98 | 0.06 |
| 106 | 0 | 50717.19 | -3.0 | -2.61 | 0.05 |
| 106 | 1 | -108753.93 | -3.0 | -2.95 | 0.08 |
| 107 | 0 | -10457.69 | -1.0 | -0.61 | 0.04 |
| 107 | 1 | -119543.36 | 0.0 | -0.04 | 0.09 |
| 108 | 0 | -68830.17 | 1.0 | 1.38 | 0.06 |
| 108 | 1 | -98299.25 | 1.0 | 0.92 | 0.08 |
| 109 | 0 | -108755.93 | -1.0 | -0.67 | 0.08 |
| 109 | 1 | -50717.19 | 3.0 | 2.99 | 0.06 |
| 110 | 0 | -119543.36 | 0.0 | 0.52 | 0.09 |
| 110 | 1 | 10458.69 | 0.0 | -0.01 | 0.03 |
| 111 | 0 | -98298.25 | 0.0 | 0.46 | 0.07 |
| 111 | 1 | 68829.17 | 0.0 | -0.06 | 0.06 |
| 112 | 0 | -50712.19 | -2.0 | -1.60 | 0.05 |
| 112 | 1 | 108757.93 | -1.0 | -1.02 | 0.08 |

Note that the formal errors of the coordinate corrections are significantly less (often $2-3\times$) than the noise in the simulated range data; this is due to the averaging that occurs with more than $5\times$ as many observations as unknowns. The X and Y values in the `prq` and `prd` columns are offset, as in the two previous simulations, because means of the `prd` are constrained to zero, and that depends on whatever happens to be the mean of the random `prq` values.

The simulated range data were perturbed by zero points and backprism offsets with quantized 2 mm RMS, as in the previous section:

| Zero points (`rfsPlane3`) | | | | | | |
|---|---|---|---|---|---|---|
| `ranger` | `zeroq` mm | `zero` mm | `backq` mm | `back` mm | $\sigma_{\texttt{back}}$ mm | $\sigma_{\texttt{zero}}$ mm |
| 101 | 0.00 | 0.01 | 0.00 | -0.01 | 0.07 | 0.08 |
| 102 | 0.00 | 0.05 | -2.00 | -2.02 | 0.07 | 0.08 |
| 103 | 0.00 | 0.11 | 0.00 | 0.05 | 0.07 | 0.08 |
| 104 | 0.00 | 0.09 | 3.00 | 3.05 | 0.07 | 0.08 |
| 105 | 0.00 | 0.07 | 0.00 | 0.04 | 0.07 | 0.08 |
| 106 | 2.00 | 1.98 | -4.00 | -4.09 | 0.07 | 0.08 |
| 107 | 0.00 | 0.01 | 0.00 | 0.03 | 0.07 | 0.08 |
| 108 | 0.00 | 0.08 | 3.00 | 3.06 | 0.07 | 0.08 |
| 109 | 0.00 | 0.07 | 2.00 | 2.07 | 0.07 | 0.08 |
| 110 | 0.00 | -0.02 | -1.00 | -1.07 | 0.07 | 0.08 |
| 111 | 2.00 | 1.91 | -2.00 | -2.07 | 0.07 | 0.08 |
| 112 | -1.00 | -1.00 | 1.00 | 0.94 | 0.07 | 0.08 |

The regression has recovered the zero point and backprism offsets with good accuracy and with formal error comparable to the range noise.

In summary, the rangefinders-in-a-plane problem is completely soluble, even including unknown zero points and backprism offsets. The author has presented these simulations of the zero point and backprism solutions mainly as an intellectual exercise, because these unknowns are susceptible to fairly straightforward laboratory calibration, whereas independent field measurement of numerous 60 to 240 m distances between rangefinders with $10^{-6}$ relative precision would be difficult and tedious. However, it will be interesting to see if the laboratory calibrations of zero points and backprism offsets can be confirmed using these regression techniques.

```
/* rfsCubes1Model.gf --- Model for N known rangefinders and M unknown cubes

   This model does a LS fit of measurements between an array of N
   rangefinders with known positions and M retroreflector targets.

   This rfsCubes1Model.gf was generated Wed Mar 31 10:15:18 EST 1999
*/

constant        pr[ranger,axis];
parameter       pc[cube,axis];
data            r, c;
observation     range;

main()
{
        variable i, sum, rad, naxes=3;

        while (import()) {
              sum = 0.0; for (i = 0; i < naxes; i = i + 1)
                  sum = sum + (pc[c,i] - pr[r,i])^2;
              rad = sqrt(sum);
              export(range - rad);
        }
}
```

Figure 4: Gaussfit model for known rangefinders and unknown retroreflectors

# 3 The "cube" problem

Suppose that we have determined the locations of the ground-based rangefinders by inter-rangefinder measurements as discussed in Section 2.1. We can now lock those rangefinder coordinates (by using the "`constant pr[ranger,axis]`" declaration in Figure 4), and use ranges measured from them to determine the positions of various retroreflector cubes, *if* we have also calibrated the relative Z coordinates of the rangefinders with the water-level system[10][Pel94]. This regression (LS model fit) is simple, and is shown in Figure 4.

[10]If the water-level is not available, the procedure for determining the relative Z coordinates is to observe a number of retroreflectors, in the manner discussed in this present section, and allow the Z coordinates of the rangefinders to be parameters.

For this simulation we use 7 rangefinders and 2 retroreflectors at the approximate locations of the retrospheres [Gol96] to be mounted beneath the elevation bearing housings of the GBT [Gol97], about 48 m above the level of the rangefinders:

| Rangefinder coordinates (`rfsCubes1`) | | |
|---|---|---|
| `ranger` | `axis` | `pr` mm |
| 110 | 0 | -119543.36 |
| 110 | 1 | 10458.69 |
| 110 | 2 | 0.00 |
| 111 | 0 | -98298.25 |
| 111 | 1 | 68829.17 |
| 111 | 2 | 0.00 |
| 112 | 0 | -50714.19 |
| 112 | 1 | 108756.93 |
| 112 | 2 | 0.00 |
| 101 | 0 | 10458.69 |
| 101 | 1 | 119543.36 |
| 101 | 2 | 0.00 |
| 102 | 0 | 68829.17 |
| 102 | 1 | 98298.25 |
| 102 | 2 | 0.00 |
| 103 | 0 | 108756.93 |
| 103 | 1 | 50714.19 |
| 103 | 2 | 0.00 |
| 104 | 0 | 119543.36 |
| 104 | 1 | -10458.69 |
| 104 | 2 | 0.00 |

The simulated data assume that all 7 rangefinders can see both retroreflectors:

| Simulated ranges (`rfsCubes1`) | | | | |
|---|---|---|---|---|
| `r` | `c` | `range` mm | $\sigma^2_{\texttt{range}}$ | $\Delta_{\texttt{range}}$ mm |
| 110 | 1 | 129512.40 | 0.01 | 0.08 |
| 110 | 2 | 133153.20 | 0.01 | -0.07 |
| 111 | 1 | 118763.24 | 0.01 | -0.07 |
| 111 | 2 | 142823.40 | 0.01 | 0.04 |
| 112 | 1 | 110811.63 | 0.01 | -0.01 |
| 112 | 2 | 149077.30 | 0.01 | 0.07 |
| 101 | 1 | 108563.57 | 0.01 | -0.07 |
| 101 | 2 | 150722.20 | 0.01 | 0.02 |
| 102 | 1 | 112948.81 | 0.01 | 0.15 |
| 102 | 2 | 147464.66 | 0.01 | -0.07 |
| 103 | 1 | 122200.45 | 0.01 | -0.04 |
| 103 | 2 | 139893.78 | 0.01 | -0.08 |
| 104 | 1 | 133153.26 | 0.01 | -0.04 |
| 104 | 2 | 129512.45 | 0.01 | 0.09 |

Fitting this model to these data produces estimates of the X-Y-Z coordinates of the retroreflectors:

| Retro coordinates (`rfsCubes1`) | | | |
|---|---|---|---|
| `cube` | `axis` | `pc` mm | $\sigma_{\mathtt{pc}}$ mm |
| 1 | 0 | -0.03 | 0.05 |
| 1 | 1 | 22860.15 | 0.10 |
| 1 | 2 | 48260.07 | 0.13 |
| 2 | 0 | -0.01 | 0.06 |
| 2 | 1 | -22860.13 | 0.11 |
| 2 | 2 | 48260.11 | 0.22 |

Note that the formal errors of determination of the retroreflector coordinates are roughly comparable to the sigma of the ranges (100$\mu$m). In this simulation we have allowed all of the 7 rangefinders to range on the two cubes; in reality the $\pm 65°$ conical (130° full angle, $0.577 \times 2\pi$ steradian) visibility capability of the retrospheres [Gol96] would prevent many of these paths from producing ranges. The formal errors would be larger if the impossible ranges were eliminated from the dataset. Goldman [Gol97, App.II,p.16] discusses the idea of two retrospheres back-to-back with their centers precisely 0.25 m apart, which would double the azimuth coverage of the retrospheres. A slightly more complex Gaussfit model could simultaneously solve for the rangefinder coordinates from inter-rangefinder measurements and solve for the retroreflector coordinates from rangefinder-to-cube data. The input data would consist of ranges measured from rangefinder $i$ to target $j$, and an `if`-statement would test $j$ to decide which formula to use for computed ranges. The model could incorporate the two 0.25 m distances as a constraint on the coordinates of the four retrospheres. Goldman [Gol97, p.17] has done a STAR*NET simulation equivalent to this in his "ELBEBALL" analysis.

This simulation has only six parameters. The correlations between them are zero or small, with two exceptions:

| | `pc 1,0` | `pc 2,1` | `pc 2,0` | `pc 1,1` | `pc 1,2` | `pc 2,2` |
|---|---|---|---|---|---|---|
| `pc[1,0]` | 100 | | | 11 | 11 | |
| `pc[2,1]` | | 100 | 14 | | | 86 |
| `pc[2,0]` | | | 100 | | | 16 |
| `pc[1,1]` | | | | 100 | 73 | |
| `pc[1,2]` | | | | | 100 | |
| `pc[2,2]` | | | | | | 100 |

The two correlations of order 75% are due to solving for Y and Z of the cubes using ranges measured only from rangefinders with $Y \geq 0$ (which makes the major axis of the error ellipsoid of the solution point roughly midway between the Y and Z axes); if we had ranges from the rest of the "ring of fire", as Goldman assumed for ELBEBALL, these two correlations would be essentially eliminated.

# 4 The "truss" problem

The GBT is a set of interconnected truss structures. A truss is a collection of nodes (joints) which are joined by rods; force vectors associated with the rods are assumed to be collinear with the rods (i.e., no bending moments). The forces are assumed to change the rod lengths according to Hooke's Law of springs but, for the purposes of the present discussion, we will ignore the distortions of the GBT trusses due to gravity and wind and thermal gradients, and will regard them as rigid.

We will assume that several retroreflectors are attached to different nodes of a truss. The relative positions of these retroreflectors can be determined with the procedure simulated in Section 3. In the simulations discussed in this section, several rangefinders measure distances to several such retroreflectors. The relative coordinates of both the rangefinders and the retroreflectors are locked (except for a temperature term), and we solve for parameters describing the translation and tilt of the truss relative to the ensemble of rangefinders.

## 4.1 The static truss

Consider the GBT with its elevation and azimuth locked, but with a set of retroreflectors mounted around the edges of the primary reflector backup structure [BUS]. The ground-based rangefinders can measure ranges to these retroreflectors to determine their relative coordinates, which we can lock. Now we can measure those ranges again and ask how the set of retroreflectors has translated and tilted with respect to its former position. *This is not a theoretical problem of academic interest only–it is precisely the problem to be solved in the differential pointing technique which has been suggested frequently as an interim operational mode for the GBT.*

For this simulation we will assume 12 rangefinders are available, and that 6 retrospheres are mounted around the edge of the BUS at 60° intervals.[11] The edge of the BUS is approximated as a circle of 50 m radius in a plane located 63.5 m above the rangefinders. The model shown in Figure 5 holds the positions of the rangefinders (`pr[]`) and the relative positions of the retrospheres (`pc[]`) constant, and solves for the translations and tilts of the BUS. It also solves for a temperature change parameter: the relative coordinates of the retroreflectors on the truss will scale linearly with temperature due to thermal expansion/contraction, and we can solve for the scale factor from range data. In fact, essentially *all* regressions on the GBT trusses *must* solve/compensate for temperature scale factors; the factors are of order $10^{-5}$ per degree Centigrade, and this is one millimeter per degree in a 100 meter structure, 10× larger than the rangefinder RMS.

The data file (72 equations of condition) for the static truss simulation is shown below; note that the `time` values are all zero:

[11] In fact the presence of the GBT feedarm will prevent us from putting the retrospheres at some of the ideal points.

```
/* rfsTruss1Model.gf --- truss orientation from known rangefinders & cubes

   This model does a LS fit of truss orientation parameters to
   measurements between an array of N rangefinders with known
   positions and M retroreflector targets with known positions in the
   truss coordinate system.

   The third Euler angle (truss rotation) is constrained to zero.

   This rfsTruss1Model.gf was generated Wed Mar 31 10:15:19 EST 1999
*/

constant        pr[ranger,axisr], pc[cube,axisc], coefficient;
constant        eulervelocity[axisp];
parameter       translate[axisp], euler[axisp], temperature;
data            time, r, c;
observation     range;

main()
{
        variable i, cp[3], sum, euclid, naxes=3;

        while (import()) {
            for (i = 0; i < naxes; i = i + 1)
                cp[i] = pc[c,i] * expansion(temperature,coefficient);
            about_z(cp, (euler[2]+eulervelocity[2]*time));
            about_x(cp, (euler[1]+eulervelocity[1]*time));
            about_z(cp, (euler[0]+eulervelocity[0]*time));
            sum = 0.0; for (i = 0; i < naxes; i = i + 1) {
                cp[i] = cp[i] + translate[i];
                sum = sum + (cp[i] - pr[r,i])^2;
            }
            euclid = sqrt(sum);
            export(range - euclid);
        }
        exportconstraint(euler[2]);
}
about_z(v, a) {
    variable temp;
    temp = +v[0]*cos(a) +v[1]*sin(a);
    v[1] = -v[0]*sin(a) +v[1]*cos(a);
    v[0] = temp;
}
about_x(v, a) {
    variable temp;
    temp = +v[1]*cos(a) +v[2]*sin(a);
    v[2] = -v[1]*sin(a) +v[2]*cos(a);
    v[1] = temp;
}
expansion(temp, tempcoeff) {
    return (1 + temp * tempcoeff);
}
```

Figure 5: Gaussfit model for translation, tilt and temperature of a (moving) truss

| Range data (`rfsTruss1`) | | | | | |
|---|---|---|---|---|---|
| `time` s | `r` | `c` | `range` mm | $\sigma^2_{\texttt{range}}$ mm$^2$ | $\Delta_{\texttt{range}}$ mm |
| 0.00 | 101 | 901 | 99395.94 | 0.01 | 0.19 |
| 0.00 | 101 | 902 | 111947.00 | 0.01 | -0.07 |
| 0.00 | 101 | 903 | 153573.84 | 0.01 | 0.06 |
| 0.00 | 101 | 904 | 178843.46 | 0.01 | -0.13 |
| 0.00 | 101 | 905 | 171266.90 | 0.01 | -0.15 |
| 0.00 | 101 | 906 | 135202.25 | 0.01 | 0.02 |
| 0.00 | 102 | 901 | 117033.44 | 0.01 | 0.02 |
| 0.00 | 102 | 902 | 101398.19 | 0.01 | -0.02 |
| 0.00 | 102 | 903 | 132352.07 | 0.01 | 0.07 |
| 0.00 | 102 | 904 | 167832.48 | 0.01 | 0.03 |
| *(Some lines omitted to fit page)* | | | | | |
| 0.00 | 109 | 904 | 139473.29 | 0.01 | -0.19 |
| 0.00 | 109 | 905 | 94828.97 | 0.01 | -0.09 |
| 0.00 | 109 | 906 | 102332.67 | 0.01 | 0.02 |
| 0.00 | 110 | 901 | 127511.83 | 0.01 | 0.07 |
| 0.00 | 110 | 902 | 170497.78 | 0.01 | 0.06 |
| 0.00 | 110 | 903 | 183691.22 | 0.01 | -0.08 |
| 0.00 | 110 | 904 | 160016.54 | 0.01 | -0.02 |
| 0.00 | 110 | 905 | 113115.17 | 0.01 | -0.15 |
| 0.00 | 110 | 906 | 90122.40 | 0.01 | -0.03 |
| 0.00 | 111 | 901 | 106578.47 | 0.01 | 0.17 |
| 0.00 | 111 | 902 | 152903.13 | 0.01 | 0.13 |
| 0.00 | 111 | 903 | 181528.72 | 0.01 | 0.02 |
| 0.00 | 111 | 904 | 174658.66 | 0.01 | 0.10 |
| 0.00 | 111 | 905 | 135960.20 | 0.01 | -0.14 |
| 0.00 | 111 | 906 | 94402.70 | 0.01 | -0.04 |
| 0.00 | 112 | 901 | 94993.76 | 0.01 | 0.13 |
| 0.00 | 112 | 902 | 131628.99 | 0.01 | -0.21 |
| 0.00 | 112 | 903 | 170981.78 | 0.01 | 0.11 |
| 0.00 | 112 | 904 | 181220.17 | 0.01 | -0.01 |
| 0.00 | 112 | 905 | 156647.20 | 0.01 | 0.16 |
| 0.00 | 112 | 906 | 112382.39 | 0.01 | -0.06 |

The variation of index of refraction of air with temperature is approximately $10^{-6}$, about 10× less than metallic thermal coefficients. The index near the ground will be measured easily if the true distance between any pair of rangefinders is known, and it would be easy to incorporate an index parameter in the Gaussfit model and true distance(s) in a data file.

The simulation assumes that each of the 12 rangefinders can see all 6 retroreflectors, and that the BUS is stationary (the `eulervelocity[]` values are zero). The model constrains the third Euler angle ("twisting" of the BUS) to zero; i.e, we are only solving for the Azimuth and Elevation of the BUS. A rangefinder can measure two ranges per second, so the 6 retroreflectors can be measured by the 12 rangefinders in 3 seconds, yielding 72 equations of condition. The results for the simulation are:

| Translations & tilts of the truss (`rfsTruss1`) | | | | | |
|---|---|---|---|---|---|
| `axisp` | `translate` mm | `euler` radian | `eulervelocity` rad/s | $\sigma_{\texttt{euler}}$ radian | $\sigma_{\texttt{translate}}$ mm |
| 0 | 0.02 | 0.400000 | 0.000000 | 0.000000 | 0.02 |
| 1 | -0.02 | -0.200000 | 0.000000 | 0.000001 | 0.02 |
| 2 | 63500.13 | 0.000000 | 0.000000 | 0.000000 | 0.03 |

Note that the formal error of the translation parameters is more than 3× smaller than the assumed sigma of the rangefinders (100$\mu$m); this is due to the averaging of 72 measurements by the regression. The formal

errors of the tilt parameters are $1\mu r$ (0.2 arcsec) or less. This high precision of angular determination is due to simple geometric considerations: the radius to the retroreflectors is about 50000 mm and the range to them is determined to 0.1 mm, so individual range measurements imply the BUS orientation to one part in 500000, which is two microradians, or 0.4 arcsec, and averaging of measurements improves the precision.

The `euler[0]` parameter is the determination of the change of azimuth of the BUS relative to the input relative coordinates of the retroreflectors (0.4 radian = 22.9° in this case), and the `euler[1]` parameter is the change of elevation (−11.5°) from horizontal (the "birdbath" position). *The conclusion is that 12 rangefinders are more than adequate to determine the orientation of the BUS to microradian precision from only a few seconds of range measurements.*

The simulation shown here computed the simulated range data with the ambient temperature changed by 0.20°C and it used the thermal coefficient for steel. The model solves for a temperature parameter:

| Temperature (`rfsTruss1`) | | |
|---|---|---|
| `coefficient` °C$^{-1}$ | `temperature` °C | $\sigma_{\texttt{temperature}}$ °C |
| 1.2e-05 | 0.20 | 0.03 |

We see that the simulated temperature change has been recovered accurately, with a formal error well under 0.1°C. The high precision of this temperature estimate is due to the 100 m of steel between the retroreflectors: with $10^5$mm and $1.2 \times 10^{-5}$C$^{-1}$ we have about one millimeter expansion per degree, so that rangefinder precision of 0.1 mm implies temperature precision of 0.1°C, and averaging of rangefinder data further improves the precision of the estimate.

The seven parameters of this model are almost perfectly uncorrelated, due mostly to our assumption that we have all 12 rangefinders (symmetric distribution):

| | `euler [2]` | `euler [0]` | `euler [1]` | `translate [0]` | `translate [1]` | `translate [2]` | `temperature` |
|---|---|---|---|---|---|---|---|
| `euler[2]` | | | | | | | |
| `euler[0]` | | 100 | | 3 | -1 | | |
| `euler[1]` | | | 100 | 4 | 10 | 13 | -6 |
| `translate[0]` | | | | 100 | | | |
| `translate[1]` | | | | | 100 | | 1 |
| `translate[2]` | | | | | | 100 | -17 |
| `temperature` | | | | | | | 100 |

This simulation (`rfsTruss1`) corresponds to the situation of the GBT during much of 1999: the telescope is immobile, not under the control of NRAO, but retroreflectors can be mounted on the BUS and measured by the 7 or more rangefinders which we have mounted around the telescope. It should be possible to measure changes of the tilt parameters as a function of time, and to show that they correspond to solar heating and wind forces. Such a demonstration would be a strong proof-of-concept for our goal of determining pointing corrections from rangefinder data.

## 4.2 The dynamic truss (the GBT differential pointing case)

The operational GBT will not have `eulervelocity[]` of zero, it will be moving. We will know the angular velocities with high accuracy, indeed with better accuracy than we could measure the velocities. A real determination of pointing error while tracking astronomical sources requires that rangefinders measure ranges to retroreflectors over an interval of time during which the ranges and tilts change substantially. It turns out that for the least-squares modeling technique which we are simulating the dynamic truss problem is no harder than the stationary truss problem. In fact, we can use exactly the same model (see Figure 5).

For our simulation of the GBT dynamic truss problem we have the same rangefinders and retroreflectors as in Section 4.1, but we have `eulervelocity[0]=eulervelocity[1]=0.001` (one milliradian per second, about 13× sidereal rate, in both Azimuth and Elevation). The 12 rangefinders are assumed to measure the 6 retroreflectors at random times during an interval of ±1.5s; i.e., no two ranges to the moving retroreflectors are made at the same instant (we are *not* trilaterating). Range data is assumed to be corrected for motion during integration using the precepts discussed by Goldman [Gol98a]. The simulated range data are:

| Range data (`rfsTruss2`) | | | | | |
|---|---|---|---|---|---|
| `time` s | `r` | `c` | `range` mm | $\sigma^2_{\texttt{range}}$ mm$^2$ | $\Delta_{\texttt{range}}$ mm |
| -1.30 | 101 | 901 | 99365.23 | 0.01 | -0.04 |
| -1.16 | 101 | 902 | 112015.59 | 0.01 | 0.02 |
| -1.18 | 101 | 903 | 153642.34 | 0.01 | -0.06 |
| -0.08 | 101 | 904 | 178844.72 | 0.01 | 0.11 |
| -0.62 | 101 | 905 | 171243.22 | 0.01 | 0.15 |
| -0.96 | 101 | 906 | 135138.43 | 0.01 | -0.14 |
| 0.60 | 102 | 901 | 117058.18 | 0.01 | -0.00 |
| -0.18 | 102 | 902 | 101405.78 | 0.01 | 0.04 |
| 0.26 | 102 | 903 | 132334.62 | 0.01 | 0.04 |
| 1.35 | 102 | 904 | 167794.06 | 0.01 | 0.11 |
| *(Some lines omitted to fit page)* | | | | | |
| 0.36 | 109 | 904 | 139488.77 | 0.01 | -0.08 |
| 1.45 | 109 | 905 | 94890.53 | 0.01 | -0.05 |
| -0.44 | 109 | 906 | 102341.25 | 0.01 | 0.26 |
| -0.00 | 110 | 901 | 127511.52 | 0.01 | -0.23 |
| 0.25 | 110 | 902 | 170488.14 | 0.01 | 0.11 |
| -0.55 | 110 | 903 | 183694.03 | 0.01 | -0.10 |
| 0.36 | 110 | 904 | 160029.13 | 0.01 | 0.11 |
| 0.98 | 110 | 905 | 113171.51 | 0.01 | 0.14 |
| 1.35 | 110 | 906 | 90136.94 | 0.01 | -0.04 |
| 0.61 | 111 | 901 | 106557.36 | 0.01 | -0.17 |
| -1.40 | 111 | 902 | 152975.58 | 0.01 | -0.02 |
| -0.71 | 111 | 903 | 181545.77 | 0.01 | 0.10 |
| 1.10 | 111 | 904 | 174681.35 | 0.01 | -0.05 |
| -0.52 | 111 | 905 | 135929.64 | 0.01 | -0.12 |
| 0.52 | 111 | 906 | 94426.92 | 0.01 | 0.13 |
| 0.34 | 112 | 901 | 94991.04 | 0.01 | 0.01 |
| -0.19 | 112 | 902 | 131640.33 | 0.01 | -0.13 |
| -0.34 | 112 | 903 | 170996.07 | 0.01 | 0.09 |
| -1.23 | 112 | 904 | 181215.11 | 0.01 | -0.06 |
| 1.19 | 112 | 905 | 156707.69 | 0.01 | 0.06 |
| -0.63 | 112 | 906 | 112342.05 | 0.01 | -0.12 |

The translation and tilt parameter results for the truss are essentially identical to those in the `rfsTruss1` case:

| Translations & tilts of the truss (`rfsTruss2`) | | | | | |
|---|---|---|---|---|---|
| `axisp` | `translate` mm | `euler` radian | `eulervelocity` rad/s | $\sigma_{\texttt{euler}}$ radian | $\sigma_{\texttt{translate}}$ mm |
| 0 | -0.03 | 0.400000 | 0.001000 | 0.000000 | 0.02 |
| 1 | -0.01 | -0.200000 | 0.001000 | 0.000001 | 0.02 |
| 2 | 63500.09 | 0.000000 | 0.000000 | 0.000000 | 0.03 |

Note that these values are for `time` of zero, the midpoint of the three second measurement interval. We are able to estimate the orientation of the BUS with precision about 0.2 arcsecond even while the BUS is moving

a total of about 600 arcsec in each axis. Furthermore, just as before, we can estimate the temperature of the BUS at the same time:

| Temperature (`rfsTruss2`) | | |
|---|---|---|
| `coefficient` $^\circ\mathrm{C}^{-1}$ | `temperature` $^\circ\mathrm{C}$ | $\sigma_{\texttt{temperature}}$ $^\circ\mathrm{C}$ |
| 1.2e-05 | 0.16 | 0.03 |

The `rfsTruss2` correlation matrix will not be shown, because it is *identical* to the matrix for `rfsTruss1` shown in the previous section.

*This simulation is a strong argument in favor of implementing a differential pointing solution for the GBT early in 2000, at or soon after 'first light'.* We are near to having the rangefinders and retroreflectors in place and operational, and this simulation shows that we have model-fitting software nearly ready to process the data. There are a number of practical details which must be addressed in an operational implementation, such as scheduling the rangefinders and making the interprocess communication robust, and so an early start on developing a prototype of this concept is desirable.

How could the `euler[0]` and `euler[1]` results be used for differential pointing corrections? The key technique is the act of observing a source with known celestial coordinates and empirically finding the position of the GBT which will maximize detected flux. The difference between this position and the position computed by M&C's "Commanded Track" is a zero point correction which M&C should add to subsequent Commanded Track computations to make the GBT point at target coordinates accurately. The correction will be valid for a limited period of time and over a limited patch of sky around the calibrator source. At the moment of such a local pointing offset observation the program which is fitting the moving truss model to the rangefinder data should note the difference between the `euler[i]` and Commanded Track, and should subsequently subtract this difference from the `euler[i]` (this difference is the offset between the orientation of the coordinate system of the retroreflectors and the orientation of the beam formed by the GBT). Subsequently any difference between the corrected `euler[i]` and the corrected Commanded Track is an indication that the BUS no longer has the orientation of the open-loop Commanded Track. I.e., it is an indication of thermal or wind loading changes. This pointing difference should be sent to M&C and added into the Commanded Track calculation. Of course, as soon as M&C adds this correction the telescope will move so that our next set of rangefinder data will produce corrected `euler[i]` which will agree with Commanded Track; obviously we need to *integrate* the differences between the corrected `euler[i]` and Commanded Track, and send the integral to M&C. This will be a closed-loop servo which will correct Commanded Track for changes in thermal gradients and wind loading. The accuracy of this servo will gradually degrade as time passes and as the telescope moves away from the Az/El where it was initialized by the local-zero observation, but the servo will lengthen the time period when the local-zero observation is good enough, thereby improving observational efficiency.

The accuracy and longevity of the differential correction can be improved by independently monitoring the elevation axle retroreflectors and inclinometers to detect thermal and wind loading changes in the alidade; such changes in the alidade can be converted to corrections to the traditional model pointing coefficients associated with the alidade, and can be transmitted to M&C so that it can compute Commanded Track with the revised model. The major correction of this type is the elevation axle collimation error (one alidade tower expanding more than the other). This technique will further improve the longevity of the differential pointing correction servo.

When the closed-loop surface control becomes operational it will correct another source of degradation of the differential pointing servo: higher-order thermal gradient and wind loading distortions of the BUS structure. It will also convert the differential system into an absolute system by servoing the primary mirror into a specified shape expressed in a coordinate system anchored in the "triplet" retroreflectors around the rim of the BUS, the same retroreflectors used by the differential pointing technique.[12] The concept of starting out

[12] The alternative reference frame, the feedarm, is an inferior choice for two reasons: (1) the flexibility of the feedarm makes

with a differential pointing correction and gradually evolving into a full absolute system is very attractive because it means that a portion of the Phase III capability can be delivered during Phase I, even before the open-loop surface servo is operational.

### 4.3 Fitting position & orientation of subreflector (with input files & $4\times$ $\sigma$-test)

Six rangefinders will be mounted on the GBT feedarm. Six retroreflector prisms will be mounted on the surface of the subreflector. This is another key truss measurement problem for the GBT. In this simulation we will use the rangefinder and retroreflector coordinates which are tabulated by Goldman [Gol98b]. "Twist" (`euler[2]`) is not locked to zero in this model, unlike the tipping structure case. All input files plus the shell command will be shown for this simulation, as an aid for others who might wish to use Gaussfit for LS modeling. The shell command to execute Gaussfit under Unix has this form:

`gaussfit.Linux` *[ModelFile] [EnvironmentFile]*

In the case of this simulation the command is

```
gaussfit.Linux rfsTruss3Model.gf rfsTruss3Env.gf >rfsTruss3Stdout.gf
```

The use of `gf` as a file type is not required, it is merely the author's convention. The environment file tells Gaussfit where to find parameter and data files, tells where to send the results, and specifies certain operational parameters (e.g. the iteration limit for the nonlinear LS solver):

[`rfsTruss3Env.gf`]

```
param1  =       'rfsTruss3Param1.gf'
param2  =       'rfsTruss3Param2.gf'
param3  =       'rfsTruss3Param3.gf'
param4  =       'rfsTruss3Param4.gf'
data1   =       'rfsTruss3Data.gf'
results =       'rfsTruss3Results.out'
iters   =        5
prmat   =        1
prvar   =        1
END
```

The parameter and data files are TAB-delimited tables of ASCII text, with the column labels specified on the first line. Gaussfit reads the parameter files and matches the column labels with the variables in `constant` and `parameter` declarations in the model file. The model used in this simulation is shown in Figure 6; it is identical to Figure 5 except that `euler[2]` is not constrained.

The data (observation) file `rfsTruss3Data.gf` is:[13]

it an inherently unstable object to use as the fundamental reference, and (2) the feedarm choice leads to an "all or nothing" situation in which no pointing correction at all is available until both closed-loop focus tracking and closed-loop surface control are operational.

[13] In this report these tables have been transformed to LaTeX `tabular` environments by a Perl function whose calling sequence is: `print_table(gfile, caption, editlist, style)`. When Gaussfit executes it adds columns to those in the input file, and these new column labels have certain forms which can be matched by regular expression patterns to enable transformation of the labels into LaTeX mathematical symbols. In the case of the data file, the $\Delta$ column has been added by Gaussfit. The table files do not contains units declarations, but they do declare datatypes (always "`double`") in their second lines.

```
/* rfsTruss3Model.gf --- truss orientation from known rangefinders & cubes

   This model does a LS fit of truss orientation parameters to
   measurements between an array of N rangefinders with known
   positions and M retroreflector targets with known positions in the
   truss coordinate system.

   This rfsTruss3Model.gf was generated Wed Mar 31 10:15:28 EST 1999
*/

constant        pr[ranger,axisr], pc[cube,axisc], coefficient;
constant        eulervelocity[axisp];
parameter       translate[axisp], euler[axisp], temperature;
data            time, r, c;
observation     range;

main()
{
        variable i, cp[3], sum, euclid, naxes=3;

        while (import()) {
            for (i = 0; i < naxes; i = i + 1)
                cp[i] = pc[c,i] * expansion(temperature,coefficient);
            about_z(cp, (euler[2]+eulervelocity[2]*time));
            about_x(cp, (euler[1]+eulervelocity[1]*time));
            about_z(cp, (euler[0]+eulervelocity[0]*time));
            sum = 0.0; for (i = 0; i < naxes; i = i + 1) {
                cp[i] = cp[i] + translate[i];
                sum = sum + (cp[i] - pr[r,i])^2;
            }
            euclid = sqrt(sum);
            export(range - euclid);
        }
}
about_z(v, a) {
    variable temp;
    temp = +v[0]*cos(a) +v[1]*sin(a);
    v[1] = -v[0]*sin(a) +v[1]*cos(a);
    v[0] = temp;
}
about_x(v, a) {
    variable temp;
    temp = +v[1]*cos(a) +v[2]*sin(a);
    v[2] = -v[1]*sin(a) +v[2]*cos(a);
    v[1] = temp;
}
expansion(temp, tempcoeff) {
    return (1 + temp * tempcoeff);
}
```

Figure 6: Gaussfit model for the subreflector orientation case

| Range data (`rfsTruss3`) | | | | | |
|---|---|---|---|---|---|
| `time` s | `r` | `c` | `range` mm | $\sigma^2_{\texttt{range}}$ mm$^2$ | $\Delta_{\texttt{range}}$ mm |
| 0.00 | 118 | 601 | 13045.59 | 0.01 | 0.01 |
| 0.00 | 118 | 602 | 13168.62 | 0.01 | 0.00 |
| 0.00 | 118 | 603 | 15049.66 | 0.01 | -0.03 |
| 0.00 | 118 | 604 | 14861.65 | 0.01 | 0.14 |
| 0.00 | 118 | 605 | 16309.04 | 0.01 | -0.00 |
| 0.00 | 118 | 606 | 15662.77 | 0.01 | -0.08 |
| 0.00 | 117 | 601 | 13094.31 | 0.01 | -0.10 |
| 0.00 | 117 | 602 | 15041.69 | 0.01 | -0.02 |
| 0.00 | 117 | 603 | 13165.68 | 0.01 | 0.04 |
| 0.00 | 117 | 604 | 16307.06 | 0.01 | -0.08 |
| 0.00 | 117 | 605 | 14863.97 | 0.01 | -0.09 |
| 0.00 | 117 | 606 | 15816.34 | 0.01 | 0.07 |
| 0.00 | 114 | 601 | 46007.81 | 0.01 | 0.14 |
| 0.00 | 114 | 602 | 45929.90 | 0.01 | 0.07 |
| 0.00 | 114 | 603 | 48156.62 | 0.01 | -0.02 |
| 0.00 | 114 | 604 | 47670.27 | 0.01 | -0.05 |
| 0.00 | 114 | 605 | 49503.05 | 0.01 | 0.00 |
| 0.00 | 114 | 606 | 48837.96 | 0.01 | -0.07 |
| 0.00 | 113 | 601 | 46062.32 | 0.01 | -0.13 |
| 0.00 | 113 | 602 | 48149.33 | 0.01 | 0.23 |
| 0.00 | 113 | 603 | 45924.02 | 0.01 | -0.02 |
| 0.00 | 113 | 604 | 49502.52 | 0.01 | 0.03 |
| 0.00 | 113 | 605 | 47671.41 | 0.01 | -0.06 |
| 0.00 | 113 | 606 | 49032.93 | 0.01 | -0.09 |
| 0.00 | 116 | 601 | 21621.01 | 0.01 | -0.02 |
| 0.00 | 116 | 602 | 20919.48 | 0.01 | -0.09 |
| 0.00 | 116 | 603 | 23328.74 | 0.01 | 0.06 |
| 0.00 | 116 | 604 | 21994.94 | 0.01 | -0.02 |
| 0.00 | 116 | 605 | 23966.56 | 0.01 | 0.11 |
| 0.00 | 116 | 606 | 23212.83 | 0.01 | -0.15 |
| 0.00 | 115 | 601 | 21680.47 | 0.01 | 0.05 |
| 0.00 | 115 | 602 | 23325.81 | 0.01 | -0.15 |
| 0.00 | 115 | 603 | 20907.33 | 0.01 | 0.07 |
| 0.00 | 115 | 604 | 23970.33 | 0.01 | 0.15 |
| 0.00 | 115 | 605 | 21991.57 | 0.01 | 0.04 |
| 0.00 | 115 | 606 | 23421.71 | 0.01 | 0.05 |

The environment file specifies four different parameter files to be read by Gaussfit. The first of these, `rfsTruss3Param1.gf`, specifies the feedarm rangefinder coordinates (which are declared `constant` in the model):

| Rangefinder coordinates (`rfsTruss3`) | | |
|---|---|---|
| `ranger` | `axisr` | `pr` mm |
| 118 | 0 | -3848.20 |
| 118 | 1 | 58671.10 |
| 118 | 2 | 102596.00 |
| 117 | 0 | 3848.20 |
| 117 | 1 | 58671.10 |
| 117 | 2 | 102596.00 |
| 114 | 0 | -15205.20 |
| 114 | 1 | 58891.10 |
| 114 | 2 | 71322.00 |
| 113 | 0 | 15205.20 |
| 113 | 1 | 58891.10 |
| 113 | 2 | 71322.00 |
| 116 | 0 | -7754.00 |
| 116 | 1 | 53722.10 |
| 116 | 2 | 96169.00 |
| 115 | 0 | 7754.00 |
| 115 | 1 | 53722.10 |
| 115 | 2 | 96169.00 |

The second parameter file, `rfsTruss3Param2.gf`, specifies the subreflector retroreflector coordinates (which are also declared `constant` in the model):

| Retroreflector coordinates (`rfsTruss3`) | | |
|---|---|---|
| `cube` | `axisc` | `pc` mm |
| 601 | 0 | -101.59 |
| 601 | 1 | 2144.70 |
| 601 | 2 | -1850.95 |
| 602 | 0 | -3443.00 |
| 602 | 1 | 86.72 |
| 602 | 2 | -818.94 |
| 603 | 0 | 3443.00 |
| 603 | 1 | 86.72 |
| 603 | 2 | -818.94 |
| 604 | 0 | -2926.66 |
| 604 | 1 | -2579.31 |
| 604 | 2 | 819.21 |
| 605 | 0 | 2926.66 |
| 605 | 1 | -2579.31 |
| 605 | 2 | 819.21 |
| 606 | 0 | -315.07 |
| 606 | 1 | -2143.69 |
| 606 | 2 | 1284.61 |

The third parameter file, `rfsTruss3Param3.gf`, specifies the subreflector translation and tilt parameters which are declared `parameters` in the model and the angular velocities, which are declared `constant` in the model:

| Translations & tilts of the truss (rfsTruss3) | | | | | |
|---|---|---|---|---|---|
| axisp | translate mm | euler radian | eulervelocity rad/s | $\sigma_{\texttt{euler}}$ radian | $\sigma_{\texttt{translate}}$ mm |
| 0 | 9.99 | -0.000729 | 0.000000 | 0.003533 | 0.06 |
| 1 | 59721.63 | 0.001998 | 0.000000 | 0.000015 | 0.15 |
| 2 | 116526.95 | 0.004772 | 0.000000 | 0.003538 | 0.03 |

Finally, the fourth parameter file, `rfsTruss3Param4.gf`, specifies the expansion coefficient which is declared `constant` and the temperature which is a `parameter`:

| Temperature (rfsTruss3) | | |
|---|---|---|
| coefficient $^\circ\mathrm{C}^{-1}$ | temperature $^\circ$C | $\sigma_{\texttt{temperature}}$ $^\circ$C |
| 2.3e-05 | 2.39 | 0.80 |

The coefficient used in this case is appropriate for the *aluminum* (not steel) backup truss of the GBT subreflector.[14] The temperature change used to generate this simulated dataset was 2°C; we see that it has been recovered with accuracy consistent with the formal error of nearly a degree. This formal error of temperature determination is more than 10× larger than the formal error of determining the effective temperature of the primary mirror BUS simply because the baseline is about 14× shorter and because the number of equations of condition is halved (36 versus 72), but the disadvantage is somewhat offset by the 2× larger thermal coefficient of aluminum.

The data and parameter files are used by Gaussfit for both input and output; this feature is both a blessing and a curse. It is a blessing in that residual and formal error columns are intimately associated with the data values and parameters to which they apply (which makes it possible to produce elegant tabular output in this memo with minimum effort). It is a curse during development of models, when bugs in the code cause crashes and result in NaNs (IEEE Not-a-Number floating point values) being written to parameter files, which prevents them from being input files on the next test run. Even the environment file is modified by Gaussfit: the overall $\sigma$ of the fit is added to it. In the present simulation application the author generates fresh copies of all of these files in the Perl driver program for each execution.

Examining the third parameter file, we see that the XYZ position of the subreflector is determined with formal error $\sigma$ about 160$\mu$m. The orientation is specified by three Euler angles; the dataset was generated using the three values `euler[0]=0.001`, `euler[1]=0.002` and `euler[2]=0.003`. Because the second angle is nearly zero (only two milliradians), the first and third angles are strongly correlated (solution is invariant if a constant is added to one and subtracted from the other), and so their formal errors are about 3 mr, larger than their values. Future production versions of this algorithm should be changed to solve for the tilts in the peculiar convention used by the subreflector actuators [Wel98b], for which these fictitious correlations will not occur. For the purposes of this proof-of-concept simulation it is the second Euler angle which is most interesting: its formal error is about 15 microradians, or about 3 arcseconds.

Examination of file `rfsTruss3Results.out` (specified in the environment file) shows that the least-squares fit of this model is almost linear, that it converges "quadratically", already reaching high enough precision on its second iteration.

The `prmat=1` setting in the environment file causes Gaussfit to write a table of the correlations between the seven parameters of this simulation into a file whose name is derived from the results file name. In this simulation the file is `rfsTruss3Results.out.corr`:

[14] In a production implementation of this model it would be appropriate to also solve for the (different?) temperature of the steel truss connecting the six rangefinders.

| | euler [1] | euler [0] | euler [2] | translate [2] | translate [0] | translate [1] | temperature |
|---|---|---|---|---|---|---|---|
| euler[1] | 100 | | | -52 | -1 | 44 | -60 |
| euler[0] | | 100 | -99 | -1 | 21 | | |
| euler[2] | | | 100 | 1 | -21 | | |
| translate[2] | | | | 100 | | -69 | 2 |
| translate[0] | | | | | 100 | | 2 |
| translate[1] | | | | | | 100 | 6 |
| temperature | | | | | | | 100 |

With the exception of the bogus -99% correlation of `euler[0]` and `euler[2]`, these correlations are typical of those encountered in complicated models, which indicates that this is a well-posed problem.

Note that formal errors of parameters can be improved by extra observations, *if* the observational errors have Gaussian (normal) distributions. *For normal distributions, the formal errors can be halved by measuring the data set four times.*[15] This fact is very important for determination of rangefinder locations and determination of relative truss coordinates of retroreflectors, and for determining the calibration of subreflector focus-tracking [Wel98a], because these are not real-time problems and so we can afford to acquire arbitrarily large datasets. To illustrate the effect on formal errors of repeated observations, the author ran the `rfsTruss3` simulation again with each observation repeated four times (with independent Gaussian noise) to demonstrate that formal errors are approximately halved (the reader should compare with the earlier version of this table):

| Translations & tilts of the truss (`rfsTruss3`) | | | | | |
|---|---|---|---|---|---|
| axisp | translate mm | euler radian | eulervelocity rad/s | $\sigma_{\texttt{euler}}$ radian | $\sigma_{\texttt{translate}}$ mm |
| 0 | 10.01 | 0.000821 | 0.000000 | 0.001928 | 0.03 |
| 1 | 59721.38 | 0.001984 | 0.000000 | 0.000008 | 0.08 |
| 2 | 116527.03 | 0.003181 | 0.000000 | 0.001931 | 0.01 |

In [Gol98b], Goldman has reported results from three STAR*NET simulations whose input files are named `SUBREF3.DAT`, `SUBREF5.DAT` and `SUBREF7.DAT`. The first, `SUBREF3`, analyzes the geometry in a ground-referenced coordinate system. The latter two cases involve feedarm rangefinders measuring the subreflector retroreflectors in a local coordinate system. The `rfsTruss3` results discussed in this section correspond fairly well to Goldman's simulation `SUBREF5`; both of these simulations assume that the rangefinder locations are known with negligible error. Goldman's simulation solves for the locations of the six retroreflectors, but their relative locations are constrained. He is able to infer the angular orientation of the subreflector from the retroreflector locations. The author's model solves for the three rotation parameters directly. In addition, as discussed above, the model also solves for a differential temperature correction for the subreflector backup structure. Because of the differences between the assumptions of the simulations and the software technologies employed, only general comparison between the results is possible. For `SUBREF5`, Goldman gets XYZ (translation) RMSes of [85,175,45]$\mu$m, while `rfsTruss3` gets about [60,150,30]. Goldman infers orientation uncertainties of order 3 to 5 arcsec from the XYZ RMSes and the 3 to 4 meter baselines. The orientations computed by `rfsTruss3` are harder to interpret because they are Euler angles, and the well-known ambiguity for zero-tilt applies in this case as noted above. However, the key result which comes out is that the second Euler angle (the elevation-like tilt) has formal error about 9$\mu$r (about 1.8 arcsec), somewhat smaller than for `SUBREF5`. The pros and cons of the different assumptions used in Goldman's three `SUBREF` simulations and in `rfsTruss3` are debatable, of course, but the overall conclusions of the two studies are more or less consistent in saying that translation uncertainties of order 150$\mu$m and tilt uncertainties of order 2 arcsec appear to be obtainable from the rangefinder technology.

[15] This assertion assumes that range residuals are independent, i.e. not correlated in time or space; this assumption is true only for short enough time intervals and distances smaller than the typical convection cell size. Production implementations will need to solve for variations of the index of refraction.

The first focal point of the subreflector is roughly five meters from the origin of the subreflector system, so RMS orientation $10\mu r$ (2 arcsec) is equivalent to about $5 \times 10^6 \times 10 \times 10^{-6} \approx 50\mu m$ RMS at the first focal point, somewhat smaller than the translation RMS for the subreflector. So, the position of the first focal point relative to the Gregorian feedroom can be inferred to RMS precision of order $150\mu m$. This sort of RMS is $\frac{\lambda}{20}$ focus error at 3 mm wavelength. It is also an RMS pointing error of roughly $0.15/60000 \approx 2.5\mu r$ $\approx 0.5$ arcsec. *The* `SUBREF5` *and* `rfsTruss3` *simulations thus demonstrate that we should be able to position the subreflector accurately relative to the Gregorian feedroom by using the six feedarm rangefinders to measure the six retroreflectors mounted on the subreflector.*

# 5 Implementation and operational issues

In this section the performance of the simulation software will be reviewed, and other sources of model-fitting software will be discussed. The algorithm for computing Gaussian noise is displayed.

## 5.1 Perl & Gaussfit performance, other software options

The reader may be wondering whether interpreted code could ever be fast enough for real-time use. In fact, the author assumed at the start of this project that it would be purely a demonstration project, that Perl and Gaussfit would be used only because they are easier and quicker to program, and that the algorithms would be recoded in Fortran using `DQED` [HK87] or `ODRPACK` [BBRS92] called from C code for production purposes, analogous to Cotton's [Cot97] implementations . The first versions of these simulations were developed on a Sun "IPX" workstation and, as expected, were not fast enough for real-time use. However, when the code was tried on a state-of-the-art Pentium-II machine, it was instantly apparent that this conclusion would need to be revised. E.g., a 450 MHz system says:[16]

| One execution of `rfsTruss2` took 2 secs ( 0.10 usr 0.01 sys + 0.29 cusr 0.08 csys = 0.48 cpu). |
|---|

The wallclock execution time is quantized to one second, and includes a number of overheads of the Perl system and the execution of Gaussfit which could be eliminated or reduced in a production implementation:

- The Gaussfit dataset, parameter and environment files are being computed in this time and written to disk as ASCII text.
- Gaussfit is being spawned as a subprocess under Perl
- Output ASCII text files produced by Gaussfit are opened, parsed, and reformatted into LaTeX tables for reproduction in this memo.
- Unix command "`ls -lg`" is spawned as a subprocess to produce a directory listing of the files associated with the simulation.

*It is clear that even with these overheads the implementation is significantly faster than the three seconds required for the rangefinders to acquire the 72 ranges which are being fitted.* Considering the desirability of invoking M&C classes to handle operator interface and logging functions, one could certainly argue that coding this application in Perl would make it harder to integrate into the long-term operational system. However, it might be that an experimental prototype system could be implemented more quickly and easily in Perl.

If this Perl program were to be expanded into a real-time application, it would need to talk over the network. This is not a problem, Perl can do that, in several different ways. The most obvious approach would be to

[16] This Gateway GP6-450 workstation is roughly 15× faster than the Sun IPX it replaced in October 1998.

code the necessary RPC interface routines in C and invoke the C functions from Perl. This would make it possible to send commands to the rangefinders, get their data back, received "Commanded Track" and other information from the M&C system and send computed pointing correction results to M&C.

Perl is a portable language, and versions are available under Windows 9x and NT. Gaussfit is also able to run under 9x and NT. The author wondered whether the Perl functions which cause Gaussfit to run under the Perl program would be available under NT; inspection of textbook-example Perl code [Joh96, Chapter 6, "Launching Applications"] demonstrates that it would probably be possible to make these simulation scripts run under both Unix and Windows NT. In general, we can conclude that variations on the theme presented here are candidates to be used as standalone test tools in the native environment of the rangefinder systems, independent of the M&C system.

The author expects that the Fortran subroutine package `DQED` [HK87] could be substituted for Gaussfit in all of the model-fitting applications discussed in this memo, because it supports nonlinear function fitting with linear constraints on parameters. `DQED` is used in the NRAO VLA Sky Survey [NVSS][17] to compute source model parameters for the survey catalog, where its nonlinear constraint capability is used to prevent producing source width parameter estimates smaller than the restoring beam width.[18]

The Fortran subroutine package `ODRPACK` [BBRS92] can substitute for Gaussfit for much of the functionality of these simulation codes. However, `ODRPACK` does not support linear equality constraints, and so it is unclear whether it can support the models in this report which call Gaussfit's `exportconstraint()` function.

## 5.2 Function `normal(sigma)`

The function subroutine `normal()`, which is in library `rfsPerlSubs.pl`, is used by the simulation programs to add Gaussian noise to computed ranges. The Perl [WCS96] source is:

```
# -=-=-=-=-=-=-=-=-=-=- normal(sig) -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# Function normal(sig) implements the polar method for normal deviates,
# which is Algorithm P in Section 3.4.1 (Numerical Distributions)
# of Vol.2 (Seminumerical Algorithms) of Knuth's 'The Art of Computer
# Programming' (1st Edition, 1969):
sub normal {
    my ($sig, $s, $v1, $v2, $x1);
    $sig = $_[0];
    $s = 2.0;
    while ($s >= 1.0) {
        $v1 = 2.0 * rand() - 1.0;
        $v2 = 2.0 * rand() - 1.0;
        $s = $v1*$v1 + $v2*$v2;
    };
    $x1 = $v1 * sqrt(-2.0 * log($s) / $s);
    return ($x1 * $sig);
}
```

This algorithm computes normal deviates by an *exact* transformation of the uniform distribution produced by Perl library function `rand()`. For readers who are not familiar with Perl, this fragment of code will indicate many of the syntactic differences between Perl and Fortran or C for numerical work. An interesting fact about this code is that it appears that the Perl library function `rand()` is initialized with a "seed" which is unique for each execution of Perl. I.e., a *different* simulation occurs each time this code is executed.

[17] `http://info.cv.nrao.edu/~jcondon/nvss.html`
[18] B. Cotton, private communication

# References

[BBRS92] Paul T. Boggs, Richard H. Byrd, Janet E. Rogers, and Robert B. Schnabel. *User's Reference Guide for ODRPACK Version 2.01—Software for Weighted Orthogonal Distance Regression.* National Institute of Standards and Technology, Gaithersburg, MD 20899, June 1992. ODRPACK is a software package for *weighted orthogonal distance regression*, i.e., for finding the set of parameters that minimize the sum of the weighted orthogonal distances from a set of observations to the curve or surface determined by the parameters. It can also be used to solve the nonlinear ordinary least squares problem. This is report NISTIR 92-483. Also see [BBS87] and [BDBS89]. See `http://netlib.att.com/magic/netlib_find?db=0&pat=odrpack` or `ftp://netlib.att.com/netlib/odrpack/`.

[BBS87] Paul T. Boggs, Richard H. Byrd, and Robert B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal distance regression. *SIAM Journal on Scientific and Statistical Computing*, 8(6):1052–1078, November 1987.

[BDBS89] P. T. Boggs, J. R. Donaldson, R. H. Byrd, and R. B. Snabel. Algorithm 676, ODRPACK – software for weighted orthogonal distance regression. *ACM Transactions On Mathematical Software*, 15:348–364, 1989.

[Cot97] W. D. Cotton. Status of pointing tests on the 140 ft using the GBT metrology system. GBT Memo 169, NRAO, October 1997. "..rate.. widely space targets.. 1 per second. Short term.. RMS 60$\mu$.. independent of range.. residuals [to] backup structure.. 120$\mu$.. on a muggy August afternoon.".

[Cre98] Ramón E. Creager. The Green Bank Telescope laser metrology system ZIY version 2.5 software interface reference manual. GBT Memo 189, National Radio Astronomy Observatory, Green Bank, WV 24944, September 1998. This GBT Memo contains only the table of contents of the manual; complete copies can be obtained by requesting GBT Archive L0480 from `scurry@nrao.edu`. An earlier version of this manual was published in January 1994 as GBT Memo 111.

[Gol96] M. A. Goldman. Ball retroreflector optics. GBT Memo 148, NRAO, March 1996. "..design and optical properties of ball retroreflectors are discussed. Ray tracing computations are given for.. reflectors to be used at the GBT.." (note that an erratum to p.6 was issued).

[Gol97] M. A. Goldman. Laser distance measurements to the elevation bearing support weldments. GBT Memo 164, National Radio Astronomy Observatory, March 1997. "..for any alidade azimuth, at least two ranging lines of sight are available on the right side of the alidade and two or more on the left side, to the bottom surface of each weldment.. the horizontal angle of the line between the [bearings] could be measured withing 1 arcsecond, and the vertical tilt of this line to 2 arcseconds.. ball retroreflectors.. would have to be viewed near the limits of their response range to return signal to both right and left side rangefinders..".

[Gol98a] M. A. Goldman. Dynamical rangefinder measurements. GBT Memo 188, NRAO, June 1998. "..problem of.. adjusting the measured ranges to give the coordinates of the target point's trajectory versus time.. velocity and acceleration.. determined.. by the commanded telescope pointing schedule.. Methods of range interpolation are presented. Range rate corrections to measured range distance are given".

[Gol98b] M. A. Goldman. A summary of STAR*NET code simulations of subreflector position measurements using the GBT laser metrology system. GBT Archive A0118, NRAO, July 1998.

[Han86] R. J. Hanson. Least squares with bounds and linear constraints. *SIAM J. Sci. Stat. Comput.*, 7(3):826–834, July 1986.

[HK87] R. J. Hanson and F. T. Krogh. Subroutine DQED. Fortran subprogram `DQED()` solves the constrained nonlinear least squares problem. The user provides simple upper and/or lower bounds on parameters, linear constraints on parameters and evaluation code for the functions. The comments of `DQED` cite [Han86]. See: `http://www.netlib.org/opt/dqed.f`, February 1987.

[JFMM88] William H. Jefferys, Michael J. Fitzpatrick, Barbara E. McArthur, and James E. McCartney. *User's Manual—GaussFit: A system for least squares and robust estimation.* The University of Texas at Austin, 1.0-12/2/88 edition, December 1988. Gaussfit is a program which supports a full-featured programming language in which models can be built to solve generalized least squares (both linear and nonlinear) and robust estimation problems. Derivatives are computed analytically, linear constraints and orthogonal distance regression are supported and errors in independent variables and correlated observations are handled correctly. See also `ftp://clyde.as.utexas.edu/pub/gaussfit/` and `http://clyde.as.utexas.edu/Gaussfit.html`.

[Joh96] Eric F. Johnson. *Cross-Platform Perl.* M & T Books, New York, 1996. ISBN=1-55851-483-X.

[Par96] David H. Parker. A status report on the GBT laser demonstration at the 140-foot telescope. GBT Memo 157, NRAO, August 1996.

[Par97] David H. Parker. The Green Bank Telescope laser metrology R&D project. GBT Memo 166, National Radio Astronomy Observatory, April 1997. This report is a comprehensive history of the development of the GBT laser rangefinders.

[Pel94] Pellissier. Pellissier Model H5 portable hydrostatic level/tiltmeter. GBT Memo 116, NRAO, November[?] 1994. "..water tube level.. insulated and temperature stabilized.. range of ±25mm.. motor driven probe.. capable of measuring the Earth tide effect with an overall accuracy of ±3 microns..".

[PP90] John Payne and David Parker. The laser ranging system for the GBT. GBT Memo 57, National Radio Astronomy Observatory, July 1990. Describes early prototype rangefinder, discusses group refractive index $n_g$.

[PPB92] J. M. Payne, D. Parker, and R. Bradley. A rangefinder with fast multiple range capability. GBT Memo 73, National Radio Astronomy Observatory, 1992. This report was published in June 1992 issue of *Rev. Sci. Instr.*, pp.3311–3316, which was reprinted in SPIE Milestone Series Volume MS115.

[PPB95] John M. Payne, David H. Parker, and Richard F. Bradley. Optical electronic distance measuring apparatus with movable mirror. GBT Memo 140, NRAO, October 1995. This is a copy of US Patent 5,455,670 filed May 27, 1993, and issued October 3, 1995.

[Sch90] Fred Schwab. Error sensitivity of GBT laser metrology. GBT Memo 37, NRAO, February 1990. The errors of rangefinder measurements of the primary mirror discussed in this memo are estimated from the Hessian matrix, not from a least-squares model fit. Contour maps of formal errors are displayed for various possible rangefinder configurations.

[WCS96] Larry Wall, Tom Christianson, and Randal L. Schwartz. *Programming Perl.* O'Reilly & Associates, Sebastopol, CA, second edition, 1996. QA76.73.P47W34, ISBN 1-56592-149-6.

[Wel98a] Don Wells. GBT Gregorian focus tracking in C. GBT Memo 183, NRAO, June 1998. The GBT Gregorian subreflector, an off-axis portion of an ellipsoid, plus the feedroom with the feedhorn move relative to the prime focal point of the main paraboloidal mirror. The subreflector must be maneuvered relative to the prime focal point and the feedhorn to maintain nearly stigmatic imaging (maximum gain, minimum sidelobes). An algorithm is described which computes the required actuator motions as a function of elevation. Raytracing analysis shows that GBT wavefronts produced by this optical prescription exhibit *no* focus error, spherical abberation or coma; their only abberation is astigmatism, with amplitude 0.4 mm for $E = 0°$ and $E = 90°$.

[Wel98b] Don Wells. GBT subreflector actuator functions in C. GBT Memo 175, NRAO, January 1998. Two ANSI-C functions are described: `srDisplacementToLength()`, which accepts displacements (three linear, three angular) from the "home" position of the subreflector and computes six actuator lengths, and `srLengthToDisplacement()`, which performs an iterative numerical inversion of function `srDisplacementToLength()` (i.e., it accepts actuator lengths and produces displacements). These two functions will be used in the GBT Monitor and Control software to transform Gregorian focus tracking outputs into actuator commands. Partial derivatives of the actuator lengths *wrt* translation and tilt are tabulated.

[Wel98c] Don Wells. Visualizing the geometry of the GBT metrology systems. GBT Memo 177, NRAO, February 1998. This report presents an atlas of images of the GBT metrology systems rendered with NRAO's visualization workstation hardware using the "AVS" software system. These images illustrate geometric relationships between the metrology sensors and the structure of the GBT; they also illustrate methods for using visualization technology to study complicated geometry.