GBT Memo No. 37

ERROR SENSITIVITY OF GBT LASER METROLOGY

FRED SCHWAB

February 27, 1990

I. INTRODUCTION

Currently it is envisioned that the information needed for active surface control and pointing of the Green Bank Telescope will be derived from distance measurements obtained by laser ranging. Laser rangefinders attached to solid ground would monitor the positions of other rangefinders attached to the feed support structure, and these, in turn, would monitor the deformation and relative motion of the primary reflector and the position and orientation of the subreflector. This concept is described in a recent memo by John Payne, *Pointing and Surface Control of GBT* (GBT Memo No. 36), and in an earlier memo by John W. Findlay, *Notes on Measuring Distances* (GBT Memo No. 24), where alternative possibilities are considered as well.

Findlay comments in his memo that "The tasks of deciding the layout of a ranging system and the computing needed to derive the information of the surface and sub-reflector locations can be attacked by computation." Here I want to take a first step in that direction, by considering the dependence of the potential accuracy of the determination of the location of a point upon the geometry of the rangefinding setup.

II. METHOD OF ANALYSIS

Suppose that we have some number n of rangefinders, located at different, known, positions $\{P_i = (x_i, y_i, z_i) \,|\, i = 1, \ldots, n\}$ and that we wish to determine the position (x, y, z) of a point P, based on n measured distances, $\tilde{d}_1, \ldots, \tilde{d}_n$ (one per rangefinder), each of r.m.s. accuracy σ. The actual distances are given by $d_i(x, y, z) = |P_i - P| = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$. In order to design competently a surface metrology scheme, it would be helpful to know the accuracy to which the coordinates of a point P are determinable, for every P near the nominal design paraboloid, as a 'function' (so to speak) of the geometrical arrangement of the rangefinders. [The nominal design paraboloid is given by $z(x, y) = f(r) = \frac{r^2}{4c}$, where c (= 60 meters) is the focal length and $r \equiv \sqrt{x^2 + y^2}$.]

Payne proposes that three rangefinders be used to locate positions on the primary reflector. Clearly, at least three are required (in the absence of any prior information—e.g., that lateral displacements of points on the surface are negligible). But my analysis is formulated for the case of arbitrarily many rangefinders, just in case that a number greater than three is settled upon.

Given the n measurements $\tilde{d}_i$, and assuming normally distributed errors, a maximum-likelihood estimate of the coordinates of P is obtained by solving for

those coordinates $(\tilde{x}, \tilde{y}, \tilde{z})$ which minimize the expression

$$S(x,y,z) = \frac{1}{2}\sum_{i=1}^{n}\left(d_i(x,y,z) - \tilde{d}_i\right)^2 . \tag{1}$$

This solution can be obtained by solving for a zero of the gradient, ∇S, of S:

$$\nabla S = \begin{pmatrix} \frac{\partial S}{\partial x} \\ \frac{\partial S}{\partial y} \\ \frac{\partial S}{\partial z} \end{pmatrix} . \tag{2}$$

For present purposes (i.e., until the telescope is built), the method of solving for this zero is unimportant, since all we are interested in now is a statistical analysis of the error that is incurred, assuming that the zero has been located by some means or other. The gradient of the gradient of S (the so-called *Hessian matrix* of S) is given by

$$H = \begin{pmatrix} \frac{\partial^2 S}{\partial x^2} & \frac{\partial^2 S}{\partial x\,\partial y} & \frac{\partial^2 S}{\partial x\,\partial z} \\ \frac{\partial^2 S}{\partial x\,\partial y} & \frac{\partial^2 S}{\partial y^2} & \frac{\partial^2 S}{\partial y\,\partial z} \\ \frac{\partial^2 S}{\partial x\,\partial z} & \frac{\partial^2 S}{\partial y\,\partial z} & \frac{\partial^2 S}{\partial z^2} \end{pmatrix} . \tag{3}$$

Estimates of the variances and covariances of the solution $(\tilde{x}, \tilde{y}, \tilde{z})$ can be obtained by evaluating H somewhere in the neighborhood of the solution, inverting, and scaling by σ^2; i.e., the variance–covariance matrix, to good approximation, is given by

$$V \approx \sigma^2 H^{-1} . \tag{4}$$

The standard errors of the estimate of (x, y, z) are $\sigma_x = \sqrt{V_{11}}$, $\sigma_y = \sqrt{V_{22}}$, and $\sigma_z = \sqrt{V_{33}}$. Off-diagonal elements of V represent the covariances of the estimated parameters, and correlation coefficients can be obtained by dividing each element V_{ij} of V by the product of the ith and jth standard errors.

Solving for distance from paraboloid, rather than for Cartesian coordinates. Instead of wanting to know the Cartesian coordinates of P, we might be primarily interested in knowing the distance δ between P and the nominal design paraboloid. The surface normal $\mathbf{N}(x, y)$ equals $\frac{(-x,-y,2c)}{\sqrt{r^2+4c^2}}$, and so the Cartesian coordinates of a point a distance δ away from the nominal design surface are given by

$$\begin{aligned} \mathbf{P}(\delta, x_0, y_0) &= (x_0, y_0, \frac{r_0^2}{4c}) + \delta \mathbf{N}(x_0, y_0) \\ &= (x_0, y_0, \frac{r_0^2}{4c}) + \delta \frac{(-x_0, -y_0, 2c)}{\sqrt{r_0^2 + 4c^2}} , \end{aligned} \tag{5}$$

with δ measured along the surface normal intersecting the surface at $(x_0, y_0, f(r_0))$. In this case, the distance to the ith rangefinder is

$$d_i(\delta, x_0, y_0) = \left|\mathbf{P}(\delta, x_0, y_0) - (x_i, y_i, z_i)\right| , \tag{6}$$

where $|\mathbf{v}| \equiv \sqrt{\mathbf{v} \cdot \mathbf{v}}$. Having redefined the functional dependence of the d_i, we can rewrite Equation 1 as

$$S(\delta, x_0, y_0) = \frac{1}{2} \sum_{i=1}^{n} \left(d_i(\delta, x_0, y_0) - \tilde{d}_i \right)^2 \tag{7}$$

and now proceed as before, but with different solution parameters.

Calculation of the Hessian matrix. Denoting by p and q any two (not necessarily distinct) elements of the set of possible solution parameters (either $\{x, y, z\}$ or $\{\delta, x_0, y_0\}$, depending on whether Eq. 1 or Eq. 7 is selected), we see, from Equation 3, that H is composed of elements of the form

$$\frac{\partial^2 S}{\partial p \, \partial q} = \sum_{i=1}^{n} \left((d_i - \tilde{d}_i) \frac{\partial^2 d_i}{\partial p \, \partial q} + \frac{\partial d_i}{\partial p} \frac{\partial d_i}{\partial q} \right) . \tag{8}$$

If the r.m.s. error σ of the rangefinding measurements is sufficiently small, then an adequate approximation to H is obtained by ignoring the second-order derivatives. (With $\sigma = 50$ μm we are, I believe, well within this regime; 50 μm is the r.m.s. rangefinding accuracy that is realistically achievable, according to Payne.) When simulating the measurement scheme we may evaluate the needed partial derivatives at the exact solution of the estimation problem, since the exact solution is known. In practice, of course, we would evaluate the Hessian, and its inverse, at the calculated solution, since in that case the true answer would be unknown.

For the minimization problem defined by Equation 1, with d_i defined as in the first paragraph of this section, the required partial derivatives are $\frac{\partial d_i}{\partial x} = (x - x_i)/d_i$, $\frac{\partial d_i}{\partial y} = (y - y_i)/d_i$, and $\frac{\partial d_i}{\partial z} = (z - z_i)/d_i$. Analytic expressions for the derivatives $\frac{\partial d_i}{\partial \delta}$, $\frac{\partial d_i}{\partial x_0}$, and $\frac{\partial d_i}{\partial y_0}$ required for the problem defined by Equations 6 and 7 are straightforward, but tedious, to calculate (I used the algebraic manipulation program MACSYMA on my Sun workstation). Since they are quite messy, I do not include them here (they can be ferreted out of the program listing given in the Appendix).

All the foregoing analysis is standard fare in nonlinear parameter estimation. Textbook treatments of nonlinear least-squares methods, for example, follow a parallel development.

III. Results

In this section I present the results of an analysis of three possible configurations of those rangefinders whose task would be to survey the primary reflector. These configurations are illustrated in Figure 1. I assume the r.m.s. accuracy σ of each rangefinding measurement to be 50 μm, as in Payne's and Findlay's memoranda. All results of this section are based on calculations performed by a simple Fortran program whose listing is given in the Appendix.

In practice, the positions of the rangefinders would not be known to exact precision. Approximate positions would be established by measurements obtained by other, ground-based rangefinders anchored to solid foundations. In this preliminary analysis I have not taken into account the uncertainties in the positions of the rangefinders, but my method of analysis could be extended to do so.

Figures 2 and 3 show contour plots of the estimated errors for the configuration (A, B, B') proposed in Payne's memorandum.[1] There are obvious problems (i.e., singularities, except in determining the y-coordinates) along the line of intersection of the paraboloid with the plane in which the three rangefinders lie. Over a region of perhaps two-thirds the area of the primary reflector, σ_δ, the error in determining the distance of a point from the paraboloid, would be less than approximately 100 μm—something close to our goal. However, the errors along the line of singularity, and possibly those toward the far edge of the dish, would be excessive.

Figures 4 and 5 illustrate the improvement that would be achieved by adding a fourth rangefinder at an intermediate position (labeled D in Fig. 1) on the main feed arm. The improvement is dramatic in the areas adjacent to the previous lines of singularity; elsewhere the improvement is minor, but not insignificant.

Figures 6 and 7 show the effect of moving the rangefinders that would be located on the feed arm support legs (at B and B') to lower positions (C and C') on those legs. The result, I believe, would represent a net improvement. The plots of σ_δ and σ_z show significant improvement near the far edge of the dish, where it was most needed. Errors in the determination of δ and z are, however, slightly increased near the vertex of the paraboloid. Figure 8 is an enlarged plot of σ_δ for this configuration (A, C, C', D).

IV. Conclusions

If the rangefinders for surface metrology are to be located on the main feed arm and on the feed arm support legs, then there evidently would be a distinct advantage in using more than three rangefinders, so that they would not all have to lie within a plane that cuts the reflector. The performance of a 4-rangefinder system looks much superior to that of a 3-rangefinder system. The 4-rangefinder configuration (A, C, C', D) would apparently achieve our goal of 100 μm accuracy for surface setting, given range measurements good to 50 μm.

[1] Greg Morris provided the coordinates of the rangefinder locations shown in Figure 3 of Payne's memorandum. It might be useful for others to be aware that the structural engineering group, in their computer-aided design (CAD) work for the GBT, measure distances in inches, in a rectangular (x', y', z')-coordinate system whose origin is near the middle of the backup structure. The (x', z')-plane faces the backup structure, and the y'-axis is directed skyward. Coordinate calculations provided by the CAD programs can be converted to the vertex-centered coordinate system of my memorandum via:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{(meters)}} = 0.0254 \left[\begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x' \\ z' \\ y' \end{pmatrix}_{\text{(inches)}} + \begin{pmatrix} 2278.5315 \\ 0 \\ 549.4567 \end{pmatrix} \right],$$

where $\theta = \arctan(9/20)$.

In Section III of this memo I concentrated on the problem of surface metrology. However, the methodology for error analysis developed in Section II would apply also to our other major problem—pointing.

I have outlined here two solution schemes.[2] In considering the surface metrology problem, I believe that the error estimates σ_δ are of primary interest, because the surface actuators will act in directions approximately normal to the surface. For the pointing problem, the errors $(\sigma_x, \sigma_y, \sigma_z)$ in all the Cartesian components of the observed point P would be of interest.

I have assumed that the r.m.s. accuracy σ of the rangefinding measurements would be independent of distance; of course that will not be the case in practice. Combined with the systematic increase of σ_δ with radial distance r (see Fig. 8), this could be a source of difficulty. (And do we know the consequences of random surface errors whose r.m.s. levels have a systematically varying spatial dependence?)

Besides providing the error estimates that were discussed above, the program given in the Appendix also provides estimates of the error of each parameter assuming prior knowledge of one or both of the other parameters. For example, it prints $\sigma_{\delta|(x_0,y_0)}$, the error in estimating δ if x_0 and y_0 are (precisely) known *a priori*. I may later work on the problem of estimating σ_δ for the case in which there is some *approximate* prior knowledge of x_0 and y_0. It might be that structural inhibitions of lateral motions of the surface would help us out, because they would translate into constraints on x_0 and y_0. (Because of this, the singularity problem evident in Figures 2 and 3, for the 3-rangefinder configuration, might, in reality, vanish.)

---

[2]The errors in estimating δ, x_0, and y_0 could have been obtained by first finding the covariance matrix of the estimates of the coordinates x, y, and z of P, and then applying the usual formula for error propagation (Hald, *Statistical Theory with Engineering Applications*, p. 118):

$$V(f(x,y,z)) \approx \left(\frac{\partial f}{\partial x}\right)^2 V(x) + \left(\frac{\partial f}{\partial y}\right)^2 V(y) + \left(\frac{\partial f}{\partial z}\right)^2 V(z) + 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}V(x,y) + 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial z}V(x,z) + 2\frac{\partial f}{\partial y}\frac{\partial f}{\partial z}V(y,z).$$

I chose instead to re-parametrize, and solve directly for (δ, x_0, y_0) and for estimates of $(\sigma_\delta, \sigma_{x_0}, \sigma_{y_0})$. The contour plots of σ_{x_0} and σ_{y_0} differ slightly from those of σ_x and σ_y because of the slight difference in the definitions of the parameters (x_0, y_0) and (x, y). Supposing that one is interested only in the single parameter δ, one might ask why it is necessary to estimate two additional parameters. The answer is that it is not possible to measure the distance of a point from the paraboloid without also knowing, at least implicitly, along which surface normal that distance should be measured; this implies a knowledge of all our parameters: x_0, y_0, x, y, and z, in addition to δ.

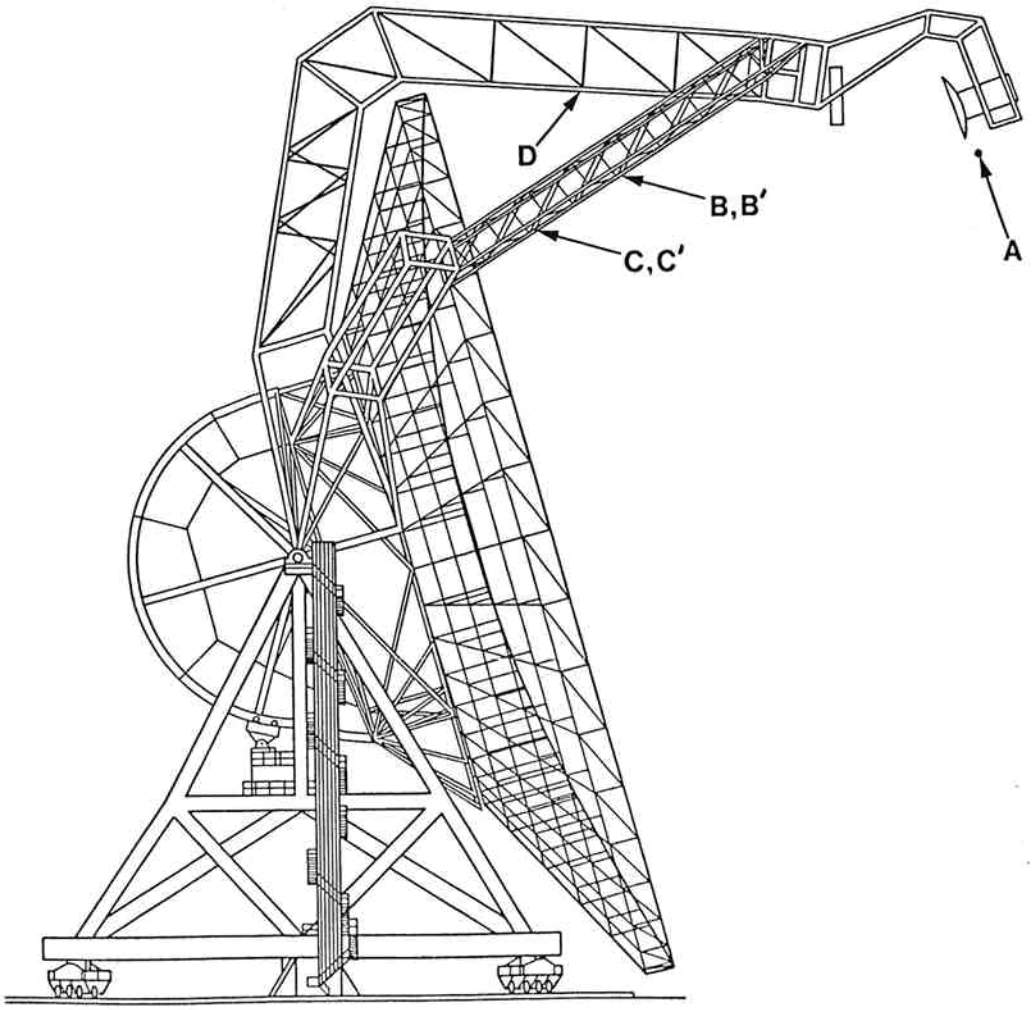

Figure 1. Possible locations for the rangefinders that would be used to survey the primary reflector. Points A and D are on the main feed arm, B and C are on the nearer feed arm support leg, and B' and C' are on the opposite leg. Payne's memo proposed locating rangefinders at A, B, and B'. In this memo, I consider three cases: (1) Payne's arrangement (A, B, B'); (2) the same arrangement, augmented by one additional rangefinder, — (A, B, B', D); and (3) the configuration (A, C, C', D).

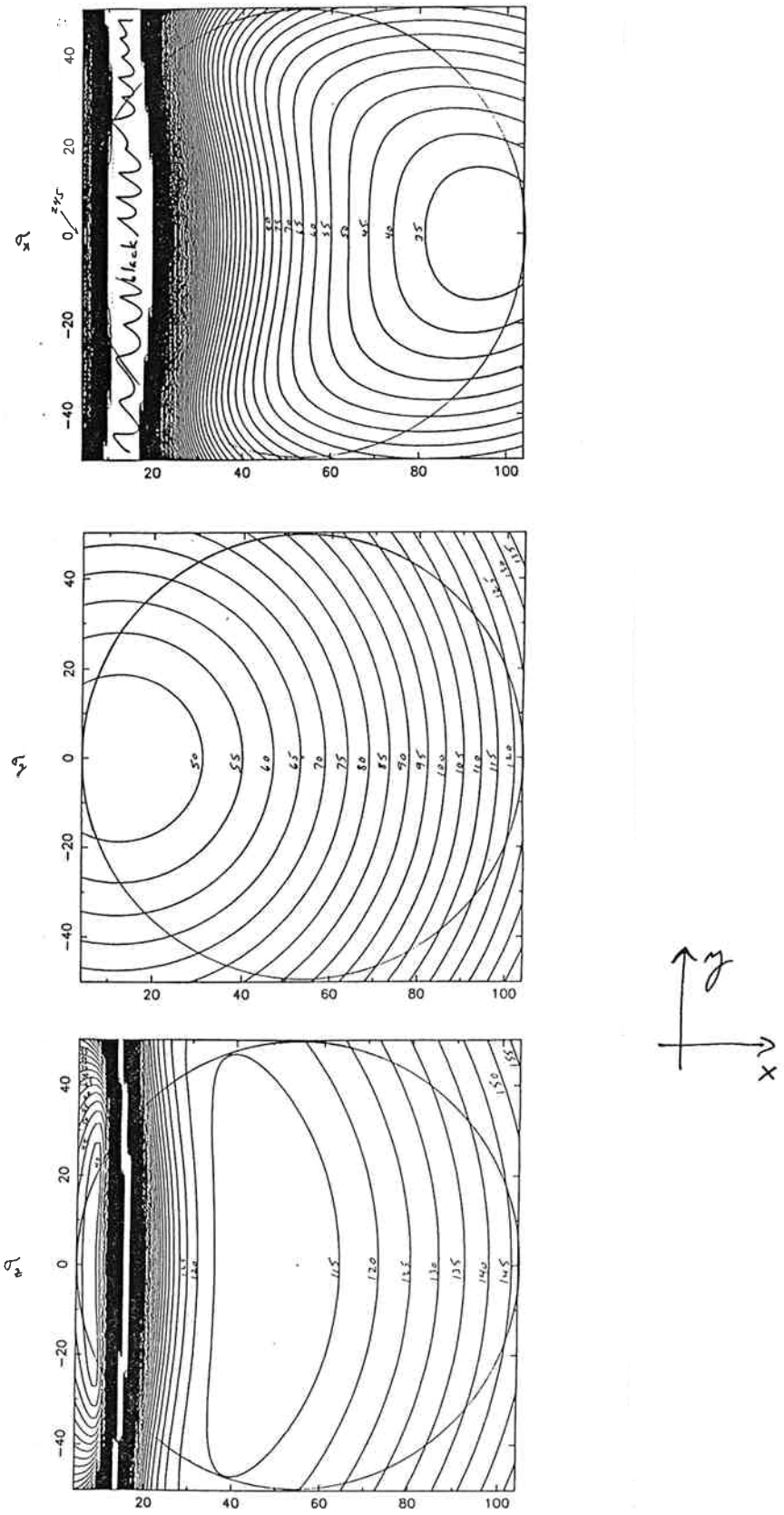

Figure 2. Contour plots of the standard errors σ_x, σ_y, and σ_z, as functions of (x, y), for the configuration (A, B, B') of three rangefinders. The circles represent the (x, y)-projection of the rim of the reflector. The contour interval is 5 μm.

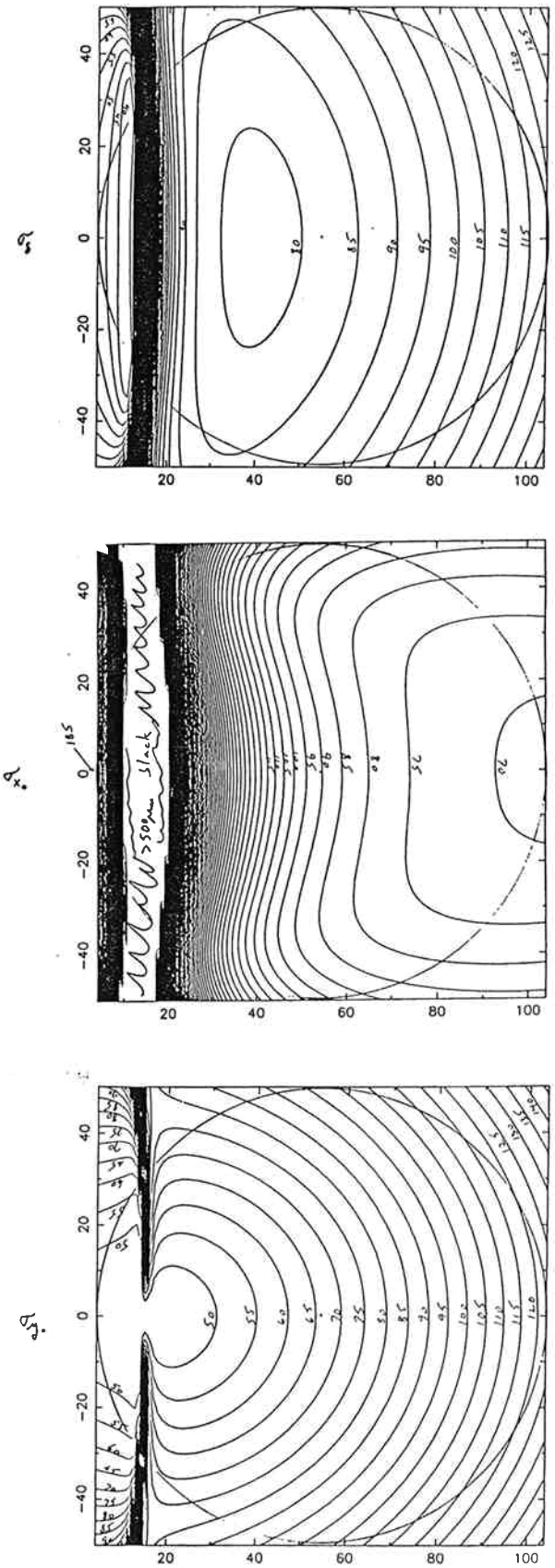

Figure 3. Contour plots of the standard errors σ_δ, σ_{x_0}, and σ_{y_0}, as functions of (x, y), for the configuration (A, B, B') of three rangefinders. The contour interval is 5 μm.

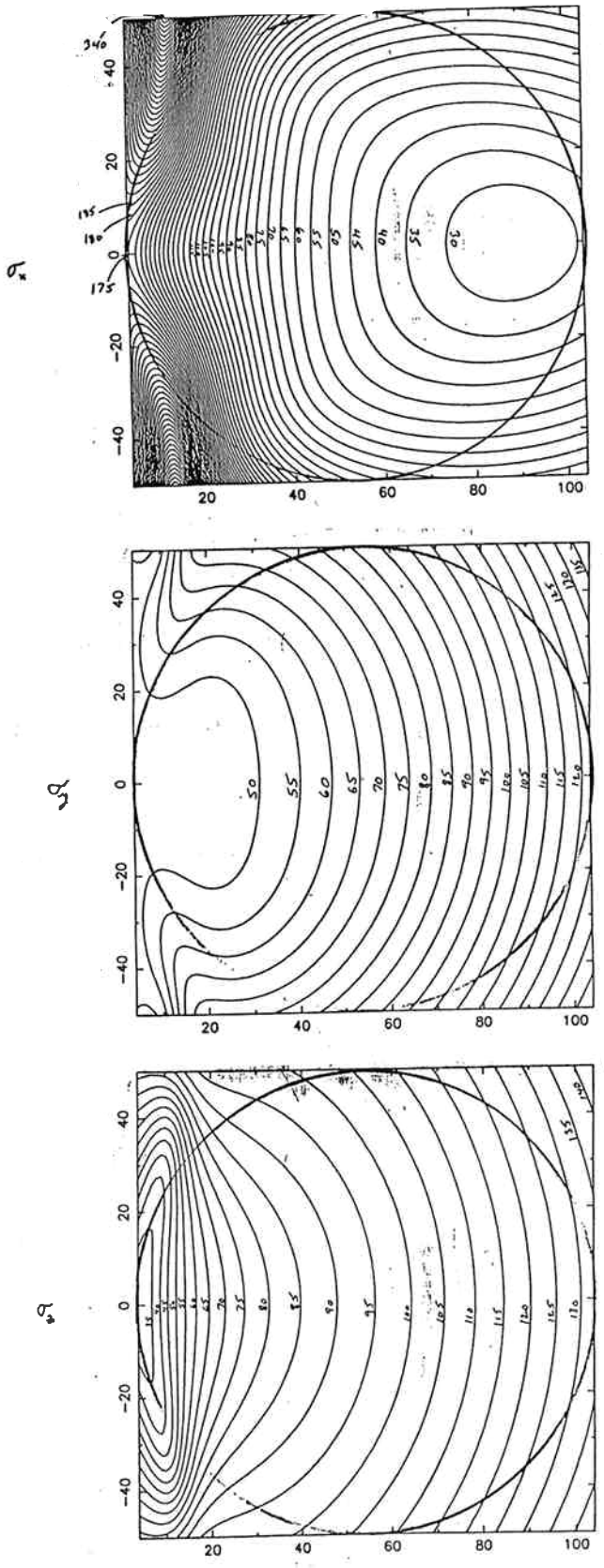

Figure 4. Contour plots of the standard errors σ_x, σ_y, and σ_z, as functions of (x, y), for the configuration (A, B, B', D). This configuration is like that of Figures 2 and 3, but augmented by one additional rangefinder on the main feed arm. The contour interval is, again, 5 μm.

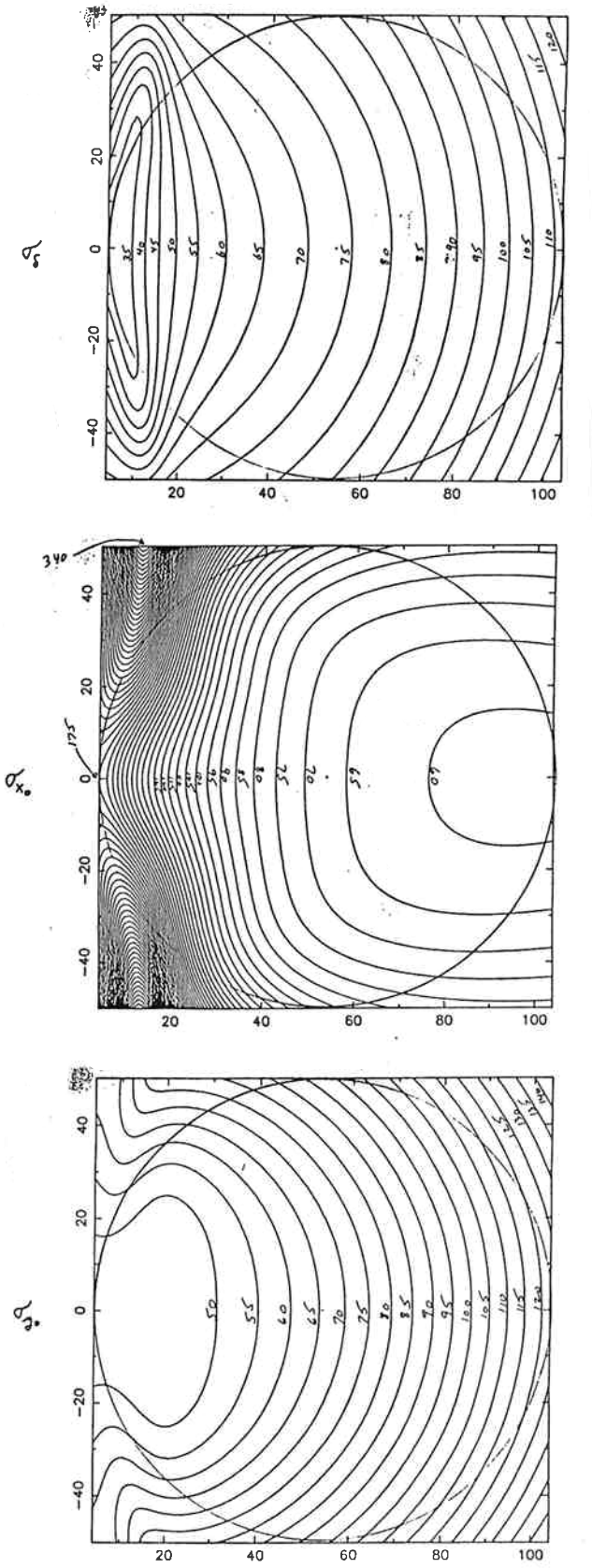

Figure 5. Contour plots of the standard errors σ_δ, σ_{x_0}, and σ_{y_0}, as functions of (x, y), for the configuration (A, B, B', D). The contour interval is 5 μm.

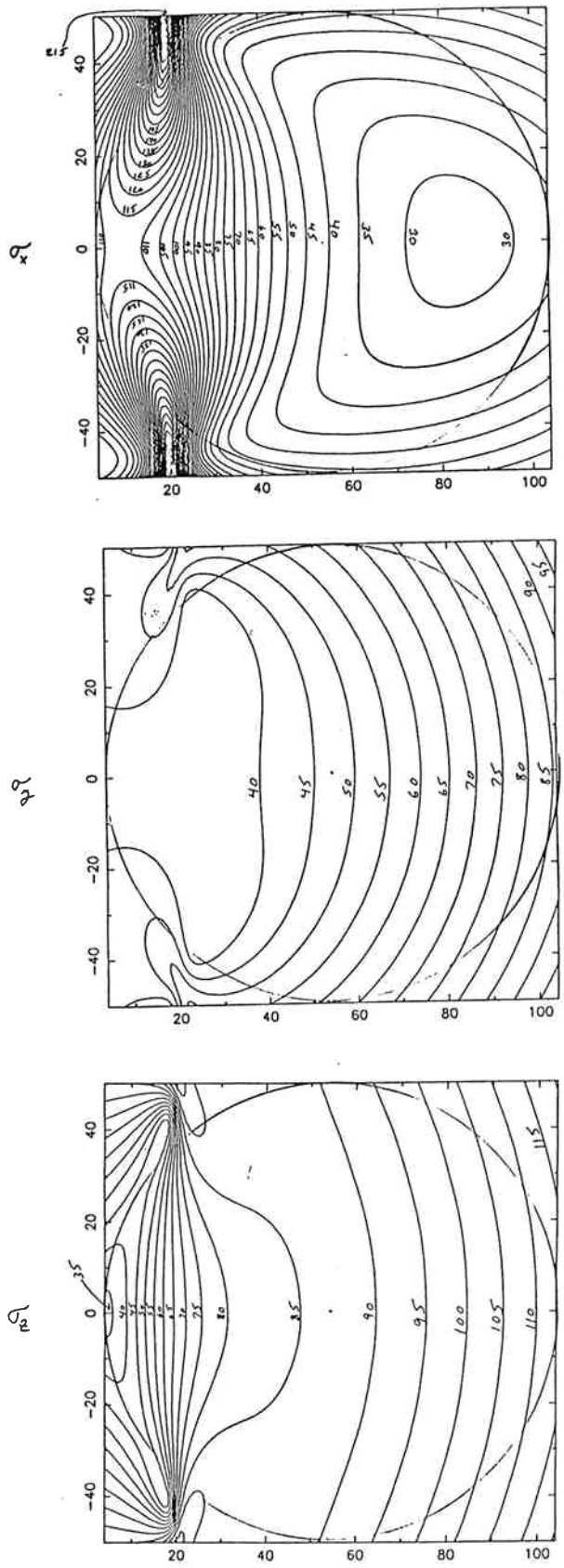

Figure 6. Contour plots of the standard errors σ_x, σ_y, and σ_z, as functions of (x, y), for the configuration (A, C, C', D). This configuration is like that of Figures 4 and 5, but with lower positions of the rangefinders on the feed arm support legs. The contour interval is 5 μm.

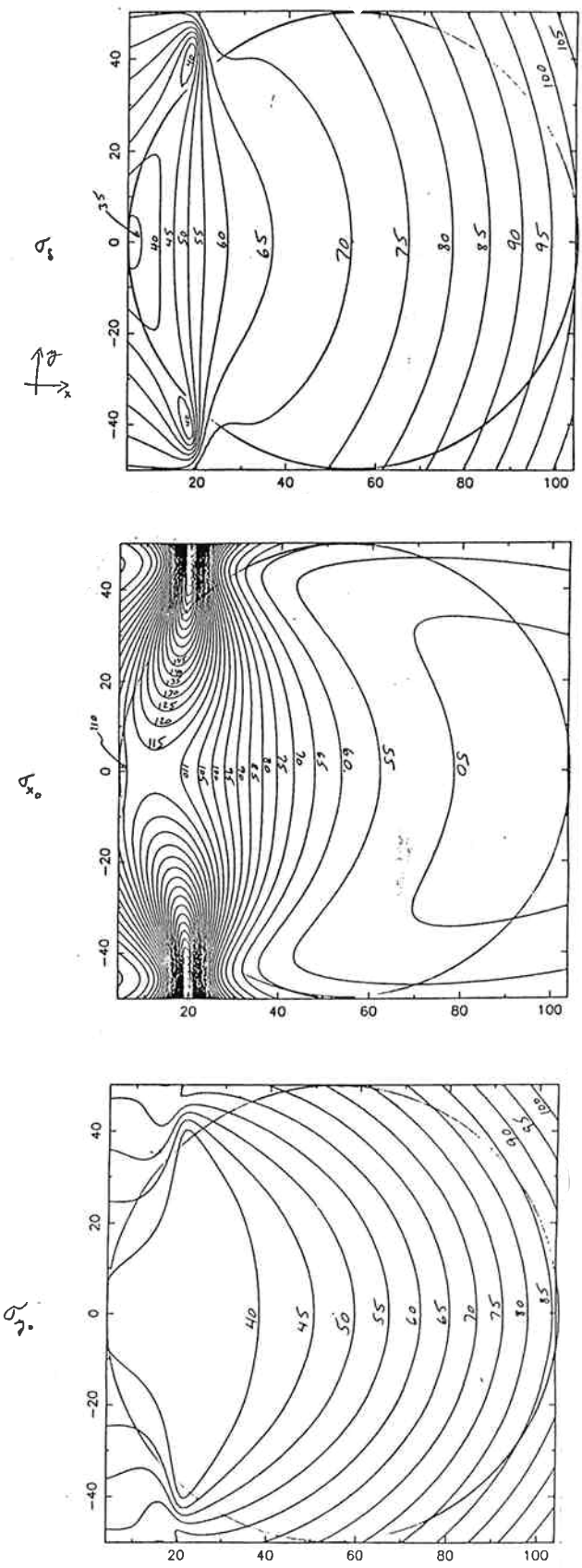

Figure 7. Contour plots of the standard errors σ_δ, σ_{x_0}, and σ_{y_0}, as functions of (x, y), for the configuration (A, C, C', D). The contour interval is 5 μm.

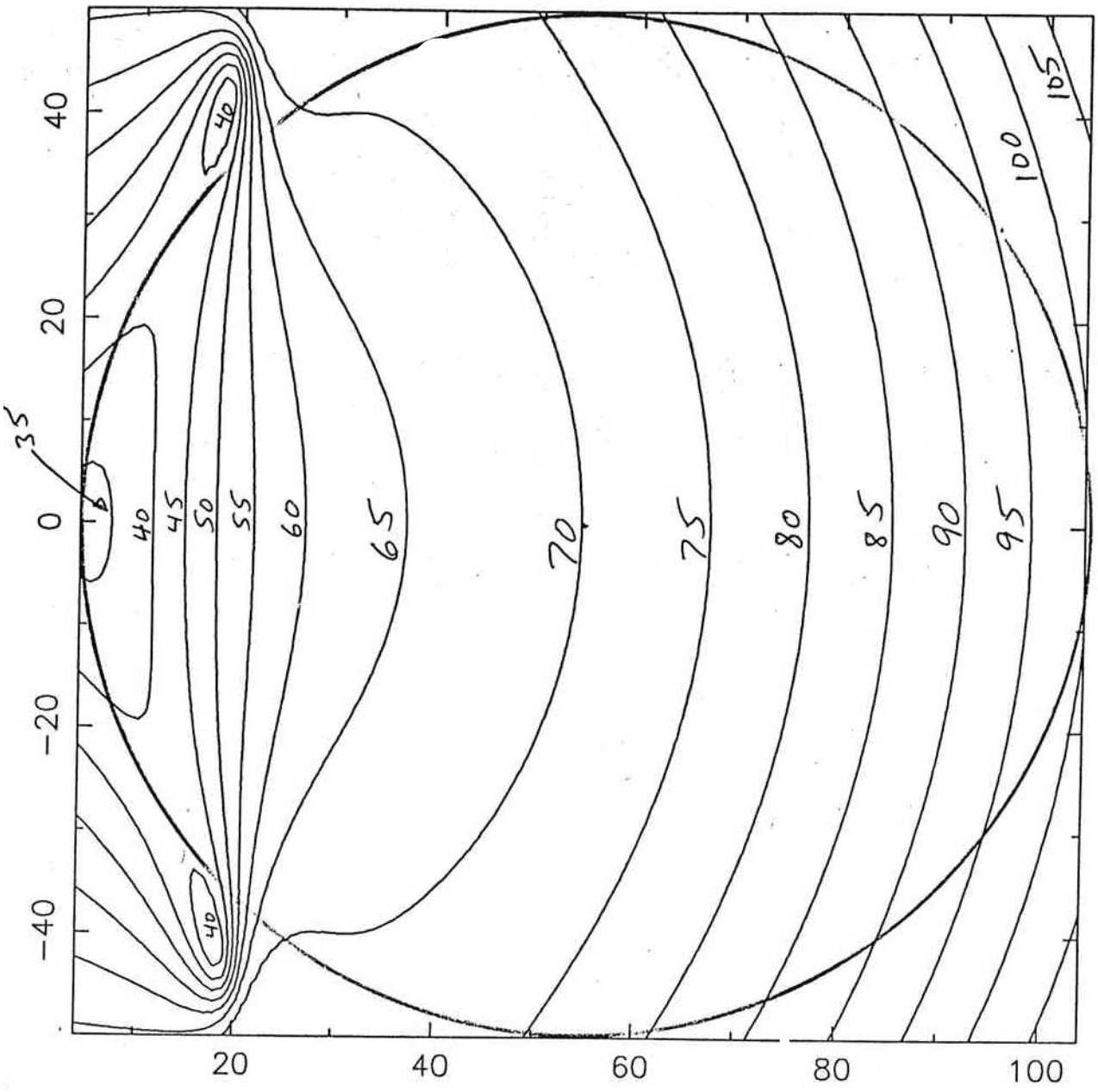

Figure 8. An enlarged contour plot of the standard error σ_δ, as a function of (x, y), for the configuration (A, C, C', D).

(1)

```
      program ranger

c  Program to estimate the errors in laser ranging metrology
c  of the GBT surface, based on the geometry of the rangefinders.

      implicit real*8 (a-h,o-z)
c  nmax is maximum number of rangefinders.
      parameter (nmax=10)
      real*8 x(nmax),y(nmax),z(nmax),d(nmax)
      real*8 h(3,3),hi(3,3),s(3),hh(2,2)
      real*4 f1(101,101),f2(101,101),f3(101,101)

c  R.m.s. error of each range determination (microns):
      sigma=50.0d0
c  Focal length (meters):
      c=60d0

c  Define number of rangefinders and their locations (Cartesian
c  coordinates, in meters):
      n=4

c  Position near the top of the feed arm:
      x(1)= 0.714d0
      y(1)= 0.d0
      z(1)-66.795d0

c  Positions on the feed arm support legs:
c  Upper position (as in Payne's memo):
c     x(2)=10.181d0
c     y(2)=28.973d0
c     z(2)=24.095d0
c  Lower position:
      x(2)=18.047d0
      y(2)=41.180d0
      z(2)=14.851d0
c  Other leg:
      x(3)=x(2)
      y(3)=-y(2)
      z(3)=z(2)

c  Added position on lower part of feed arm:
      x(4)= 1.000d0
      y(4)= 0.0d0
      z(4)=18.200d0
      do i=1,n
         print *,'i,x(i),y(i),z(i)-',i,x(i),y(i),z(i)
      end do

1     print *,'Choose one:'
      print *,'    1  ---  Solve for errors in x, y, and z'
      print *,'    2  ---  Solve for errors in delta, x0, and y0'
      print *,'    3  ---  Plot errors in x, y, and z'
      print *,'    4  ---  Plot errors in delta, x0, and y0'
      read *,iopt
      if (iopt.lt.1.or.iopt.gt.4) go to 1
      if (iopt.gt.2) go to 3
2     if (iopt.eq.1) then
        print *,'Type x,y (in meters)'
        read *,x0,y0
        z0=(x0**2+y0**2)/(4d0*c)
```

(2)

```
        do i=1,n
           d(i)=sqrt((x(i)-x0)**2+(y(i)-y0)**2+(z(i)-z0)**2)
        end do
      else
        print *,'Type x0,y0,delta (meters,meters,microns)'
        read *,x0,y0,delta
        delta=delta*1d-6
        a=sqrt(x0**2+y0**2+4d0*c**2)
        z0=(x0**2+y0**2)/(4d0*c)
        x00=x0*(1d0-delta/a)
        y00=y0*(1d0-delta/a)
        z00=z0+delta*2d0*c/a
        do i=1,n
           d(i)=sqrt((x(i)-x00)**2+(y(i)-y00)**2+(z(i)-z00)**2)
        end do
      end if
      print *,'x,y,z=',x0,y0,z0
      do i=1,n
         print *,'i,d(i)=',i,d(i)
      end do

c  Form Hessian matrix:
      h11=0d0
      h12=0d0
      h13=0d0
      h22=0d0
      h23=0d0
      h33=0d0
      do i=1,n
         xi=x(i)
         yi=y(i)
         zi=z(i)
         di=d(i)
         if (iopt.eq.1) then
            p1=(x0-xi)/di
            p2=(y0-yi)/di
            p3=(z0-zi)/di
         else
            r02=x0**2+y0**2
            p1=
     *(4*C*(-ZI+R02/C/4.0D0+2*C*DELTA/A)/A-2*Y0*(-YI-DELTA*Y0/A+Y0)/A-2*
     1   X0*(-XI-DELTA*X0/A+X0)/A)/SQRT((-ZI+R02/C/4.0D0+2*C*DELTA/A)**2
     2   +(-YI-DELTA*Y0/A+Y0)**2+(-XI-DELTA*X0/A+X0)**2)/2.0D0
            p2=
     *(2*(X0/C/2.0D0-2*A**(-3)*C*DELTA*X0)*(-ZI+R02/C/4.0D0
     1   +2*C*DELTA/A)+2*A**(-3)*DELTA*X0*Y0*(-YI-DELTA*Y0/
     2   A+Y0)+2*(A**(-3)*DELTA*X0**2-DELTA/A+1)*(-XI-DELTA
     3   *X0/A+X0))/SQRT((-ZI+R02/C/4.0D0+2*C*DELTA/A)**2+(-YI-DELTA*Y0/
     4   A+Y0)**2+(-XI-DELTA*X0/A+X0)**2)/2.0D0
            p3=
     *(2*(Y0/C/2.0D0-2*A**(-3)*C*DELTA*Y0)*(-ZI+R02/C/4.0D0
     1   +2*C*DELTA/A)+2*(A**(-3)*DELTA*Y0**2-DELTA/A+1)*(-
     2   YI-DELTA*Y0/A+Y0)+2*A**(-3)*DELTA*X0*(-XI-DELTA*X0
     3   /A+X0)*Y0)/SQRT((-ZI+R02/C/4.0D0+2*C*DELTA/A)**2+(-YI-DELTA*Y0/
     4   A+Y0)**2+(-XI-DELTA*X0/A+X0)**2)/2.0D0
         end if
         h11=h11+p1**2
         h22=h22+p2**2
         h33=h33+p3**2
         h12=h12+p1*p2
```

(3)

```
         h13=h13+p1*p3
         h23=h23+p2*p3
      end do
      h(1,1)=h11
      h(1,2)=h12
      h(1,3)=h13
      h(2,2)=h22
      h(2,3)=h23
      h(3,3)=h33
      h(2,1)=h(1,2)
      h(3,1)=h(1,3)
      h(3,2)=h(2,3)
      print *,'H='
      do i=1,3
         print *,h(i,1),h(i,2),h(i,3)
      end do

c  Invert H, and calculate errors (dlinrg is a matrix inversion routine
c  from the IMSL Library; anything else would work just as well):
      call dlinrg(3,h,3,hi,3)
      print *,'H inverse='
      do i=1,3
         print *,hi(i,1),hi(i,2),hi(i,3)
      end do
      print *,'r.m.s. errors:'
      if (iopt.eq.1) then
         print *,'sigma(x)=',sqrt(hi(1,1))*sigma
         print *,'sigma(y)=',sqrt(hi(2,2))*sigma
         print *,'sigma(z)=',sqrt(hi(3,3))*sigma
         hh(1,1)=h(2,2)
         hh(1,2)=h(2,3)
         hh(2,2)=h(3,3)
         hh(2,1)=hh(1,2)
         call dlinrg(2,hh,2,hh,2)
         print *,'sigma(y|x)=',sqrt(hh(1,1))*sigma
         print *,'sigma(z|x)=',sqrt(hh(2,2))*sigma
         hh(1,1)=h(1,1)
         hh(1,2)=h(1,3)
         hh(2,2)=h(3,3)
         hh(2,1)=hh(1,2)
         call dlinrg(2,hh,2,hh,2)
         print *,'sigma(x|y)=',sqrt(hh(1,1))*sigma
         print *,'sigma(z|y)=',sqrt(hh(2,2))*sigma
         hh(1,1)=h(1,1)
         hh(1,2)=h(1,2)
         hh(2,2)=h(2,2)
         hh(2,1)=hh(1,2)
         call dlinrg(2,hh,2,hh,2)
         print *,'sigma(x|z)=',sqrt(hh(1,1))*sigma
         print *,'sigma(y|z)=',sqrt(hh(2,2))*sigma
         print *,'sigma(x|y,z)=',1d0/sqrt(h(1,1))*sigma
         print *,'sigma(y|x,z)=',1d0/sqrt(h(2,2))*sigma
         print *,'sigma(z|x,y)=',1d0/sqrt(h(3,3))*sigma
      else
         print *,'sigma(delta)=',sqrt(hi(1,1))*sigma
         print *,'sigma(x)=',sqrt(hi(2,2))*sigma
         print *,'sigma(y)=',sqrt(hi(3,3))*sigma
         hh(1,1)=h(2,2)
         hh(1,2)=h(2,3)
         hh(2,2)=h(3,3)
```

(4)

```
         hh(2,1)=hh(1,2)
         call dlinrg(2,hh,2,hh,2)
         print *,'sigma(x|delta)=',sqrt(hh(1,1))*sigma
         print *,'sigma(y|delta)=',sqrt(hh(2,2))*sigma
         hh(1,1)=h(1,1)
         hh(1,2)=h(1,3)
         hh(2,2)=h(3,3)
         hh(2,1)=hh(1,2)
         call dlinrg(2,hh,2,hh,2)
         print *,'sigma(delta|x)=',sqrt(hh(1,1))*sigma
         print *,'sigma(y|x)=',sqrt(hh(2,2))*sigma
         hh(1,1)=h(1,1)
         hh(1,2)=h(1,2)
         hh(2,2)=h(2,2)
         hh(2,1)=hh(1,2)
         call dlinrg(2,hh,2,hh,2)
         print *,'sigma(delta|y)=',sqrt(hh(1,1))*sigma
         print *,'sigma(x|y)=',sqrt(hh(2,2))*sigma
         print *,'sigma(delta|x,y)=',1d0/sqrt(h(1,1))*sigma
      print *,'sigma(x|delta,y)=',1d0/sqrt(h(2,2))*sigma
      print *,'sigma(y|delta,x)=',1d0/sqrt(h(3,3))*sigma
   end if
   print *,'correlation matrix:'
   s(1)=sqrt(hi(1,1))
   s(2)=sqrt(hi(2,2))
   s(3)=sqrt(hi(3,3))
   do i=1,3
      do j=1,3
         hi(i,j)=hi(i,j)/s(i)/s(j)
      end do
   end do
   do i=1,3
      print *,hi(i,1),hi(i,2),hi(i,3)
   end do
   go to 2
```

```
c  Generate contour plots over a 101 x 101 grid, with a grid spacing
c  of one meter in x and one meter in y:

3     np=101
      il=0
      do ix=4,104
         il=il+1
         i2=0
         do iy=-50,50
            i2=i2+1
            x0=ix
            y0=iy
            delta=0d0
            z0=(x0**2+y0**2)/(4d0*c)
            do i=1,n
               d(i)=sqrt((x(i)-x0)**2+(y(i)-y0)**2+(z(i)-z0)**2)
            end do
c  Form Hessian matrix:
            h11=0d0
            h12=0d0
            h13=0d0
            h22=0d0
```

(5)

```
            h23=0d0
            h33=0d0
            do i=1,n
               xi=x(i)
               yi=y(i)
               zi=z(i)
               di=d(i)
               if (iopt.eq.3) then
                  p1=(x0-xi)/di
                  p2=(y0-yi)/di
                  p3=(z0-zi)/di
               else
                  a=sqrt(x0**2+y0**2+4d0*c**2)
                  r02=x0**2+y0**2
                  p1=
     *(4*C*(-ZI+R02/C/4.0D0+2*C*DELTA/A)/A-2*Y0*(-YI-DELTA*Y0/A+Y0)/A-2*
     1   X0*(-XI-DELTA*X0/A+X0)/A)/SQRT((-ZI+R02/C/4.0D0+2*C*DELTA/A)**2
     2   +(-YI-DELTA*Y0/A+Y0)**2+(-XI-DELTA*X0/A+X0)**2)/2.0D0
                  p2=
     *(2*(X0/C/2.0D0-2*A**(-3)*C*DELTA*X0)*(-ZI+R02/C/4.0D0
     1   +2*C*DELTA/A)+2*A**(-3)*DELTA*X0*Y0*(-YI-DELTA*Y0/
     2   A+Y0)+2*(A**(-3)*DELTA*X0**2-DELTA/A+1)*(-XI-DELTA
     3   *X0/A+X0))/SQRT((-ZI+R02/C/4.0D0+2*C*DELTA/A)**2+(-YI-DELTA*Y0/
     4   A+Y0)**2+(-XI-DELTA*X0/A+X0)**2)/2.0D0
                  p3=
     *(2*(Y0/C/2.0D0-2*A**(-3)*C*DELTA*Y0)*(-ZI+R02/C/4.0D0
     1   +2*C*DELTA/A)+2*(A**(-3)*DELTA*Y0**2-DELTA/A+1)*(-
     2   YI-DELTA*Y0/A+Y0)+2*A**(-3)*DELTA*X0*(-XI-DELTA*X0
     3   /A+X0)*Y0)/SQRT((-ZI+R02/C/4.0D0+2*C*DELTA/A)**2+(-YI-DELTA*Y0/
     4   A+Y0)**2+(-XI-DELTA*X0/A+X0)**2)/2.0D0
               end if
               h11=h11+p1**2
               h22=h22+p2**2
               h33=h33+p3**2
               h12=h12+p1*p2
               h13=h13+p1*p3
               h23=h23+p2*p3
            end do
            h(1,1)=h11
            h(1,2)=h12
            h(1,3)=h13
            h(2,2)=h22
            h(2,3)=h23
            h(3,3)=h33
            h(2,1)=h(1,2)
            h(3,1)=h(1,3)
            h(3,2)=h(2,3)
c  Invert H, and store standard errors in array (truncate at 500 microns,
c  so not too many contours are displayed at singularities):
            call dlinrg(3,h,3,hi,3)
            f1(i1,i2)=min(500.,sngl(sqrt(hi(1,1))*sigma))
            f2(i1,i2)=min(500.,sngl(sqrt(hi(2,2))*sigma))
            f3(i1,i2)=min(500.,sngl(sqrt(hi(3,3))*sigma))
         end do
      end do
      call plott(f1,np)
      call plott(f2,np)
      call plott(f3,np)
      stop
```

(6)

```
      end


      subroutine plott(fc,np)
c  Generates a contour plot using routines from the Caltech graphics
c  package, PGPLOT.
      real*4 fc(np,np),alev(500),tr(6),fmin,fmax
      dx=100./(np-1)
      dy=100./(np-1)
      fmin=fc(1,1)
      fmax=fc(1,1)
      do i=1,np
         x=4.+(i-1)*dx
         do j=1,np
            y=-50.+(i-1)*dy
            fmin=min(fc(i,j),fmin)
            fmax=max(fc(i,j),fmax)
         end do
      end do
      print *,'fmin,fmax=',fmin,fmax

      xc=5.
      ii=fmin/xc
      jj=fmax/xc+1
      nlev=jj-ii+1
      alev(1)=ii*xc
      do i=2,nlev
         alev(i)=alev(i-1)+xc
      end do
      print *,'nlev=',nlev
      print *,'alev(1)=',alev(1)
      print *,'alev(nlev)=',alev(nlev)

      call pgbegin(0,'/QMS',1,1)
      call pgenv(4.,104.,-50.,50.,1,0)
      tr(1)=4.-dx
      tr(2)=dx
      tr(3)=0.0
      tr(4)=-50.-dy
      tr(5)=0.0
      tr(6)=dy
      call pgsci(1)
      call pgcont(fc,np,np,1,np,1,np,alev,nlev,tr)
      call pgend
      return
      end
```