AST/RO Technical Memorandum #33
13 October 1998

RESULTS FROM TESTING OF THE AST/RO AND SPIFI SECONDARY MIRRORS

Henry H. Hsieh
hhsieh@cfa.harvard.edu
Smithsonian Astrophysical Observatory; 60 Garden St. MS 78; Cambridge, MA 02138

This memorandum describes the testing, and subsequent data analysis and surface-fitting, of the surfaces of the secondary mirrors for the Antarctic Submillimeter Telescope and Remote Observatory (AST/RO, ASTR-502-P1) and the South Pole Imaging Fabry-Perot Interferometer (SPIFI, ASTR-1026-P2). The mirrors, their equations, and their placement within the AST/RO telescope system are described in another memorandum, *Equations of the Secondary Mirrors for AST/RO*, by Antony A. Stark, Jingquan Cheng, Richard Chamberlin, and Greg Wright, dated 2 April 1992, revised 9 March 1994.

The mirrors were delivered to Boston University where their surfaces were tested by David Marzocchi (karldm@bu.edu, (617) 353-5635), using a coordinate measuring machine (CMM), which has a supposed accuracy of $\pm$ 2 μm. However, in the week during which the measurements were taken, the BU machine shop reported calibration difficulties with the CMM which interfered with its precision, so the actual uncertainty of the measurements is likely somewhat higher. These measurements are tabulated in Appendices 1 and 2.

Each mirror has four measuring balls (described in the aforementioned memorandum by Stark *et al.*) which were labeled Spheres 1-4 at the machine shop. The x axis, as defined by Marzocchi, was set half-way between Sphere 1 and Sphere 2, with increasing x going along the Sphere 3 to Sphere 2 direction. The y axis was set half-way between Sphere 2 and Sphere 3, with increasing y going along the Sphere 1 to Sphere 2 direction. The z axis was set perpendicular to the plane of the measuring balls, with increasing z coming up and out of the mirror surface. Upon preliminary analysis of the data, it was discovered that the $z = 0$ plane was not set along the plane of the centers of the measuring balls as requested, but along the upper flat surface of the mirrors themselves upon which the measuring balls rested. Therefore, in addition to the mirror depth calculations to be described now, which are heavily based on and only slightly adapted from those shown in Stark *et al.*, 1994, a 7.938 mm (0.3125 in) translation of the surface coordinates in the positive z direction was required.

Both mirrors are sections of prolate spheroids that can be described as follows where the coordinate system (x'', y'', z'') is chosen so that the equation of the secondary surface is as simple as can be:

$$\left(\frac{x''}{b}\right)^2 + \left(\frac{y''}{a}\right)^2 + \left(\frac{z''}{b}\right)^2 = 1 \tag{1}$$

This double prime coordinate system is different for the two mirrors. These coordinates are related to the local coordinate system of the mirrors (x', y', z') by translations x_0, y_0, and z_0 along x'', y'', and z'', respectively, followed by rotations through an angle θ_z about the z'' axis, an angle θ_x about the x'' axis, and by -90° about the new x axis:

$$\begin{aligned}\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x'' - x_0 \\ y'' - y_0 \\ z'' - z_0 \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 \\ \sin\theta_x \, \sin\theta_z & -\sin\theta_x \, \cos\theta_z & -\cos\theta_x \\ -\cos\theta_x \, \sin\theta_z & \cos\theta_x \, \cos\theta_z & -\sin\theta_x \end{pmatrix} \begin{pmatrix} x'' - x_0 \\ y'' - y_0 \\ z'' - z_0 \end{pmatrix}\end{aligned} \tag{2}$$

The inverse transform is

$$\begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \begin{pmatrix} \cos\theta_z & \sin\theta_x \, \sin\theta_z & -\cos\theta_x \, \sin\theta_z \\ \sin\theta_z & -\sin\theta_x \, \cos\theta_z & \cos\theta_x \, \cos\theta_z \\ 0 & -\cos\theta_x & -\sin\theta_x \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \tag{3}$$

Using this relation, Equation 1 can be rewritten in the form

$$(\alpha z' + \beta)^2 + (\gamma z' + \delta)^2 + (\epsilon z' + \kappa)^2 = 1 \tag{4}$$

where

$$\begin{aligned} \alpha &= \frac{\cos\theta_x \, \cos\theta_z}{a} & \delta &= \frac{x' \cos\theta_z + y' \sin\theta_x \, \sin\theta_z + x_0}{b} \\ \beta &= \frac{x' \sin\theta_z - y' \sin\theta_x \, \cos\theta_z + y_0}{a} & \epsilon &= \frac{-\sin\theta_x}{b} \\ \gamma &= \frac{-\cos\theta_x \, \sin\theta_z}{b} & \kappa &= \frac{-y' \cos\theta_x + z_0}{b} \end{aligned}$$

Equation 4 can be solved for z':

$$z' = \frac{-(\alpha\beta + \gamma\delta + \epsilon\kappa) \pm \sqrt{(\alpha\beta + \gamma\delta + \epsilon\kappa)^2 - (\alpha^2 + \gamma^2 + \epsilon^2)(\beta^2 + \delta^2 + \kappa^2 - 1)}}{(\alpha^2 + \gamma^2 + \epsilon^2)} \tag{5}$$

This is the desired expression for the depth of the mirror's surface in the local coordinate system, and is the formula that is used in the attached computer programs whenever it was necessary to compute a z depth from input x and y coordinates. The originally specified parameters for the two mirrors are shown in Table 1.

Using this formula, the measured x and y coordinates of the surfaces were used to calculate the desired corresponding z coordinates. These calculated z values were then checked against the actual measured z values. For the AST/RO mirror, an average deviation of 147.0 μm was found, with an RMS of 169.3 μm. For the SPIFI mirror, an average deviation of 653.0 μm was found, with an RMS of 657.0 μm. These numbers reflect the fit using the originally specified parameters. The attached C program `stats.c` was used to make these calculations for given sets of ellipse parameters.

An attempt was then made to find a best-fit ellipse. The multidimensional downhill simplex "amoeba" minimization algorithm from *Numerical Recipes in C*, by William Press *et al.*, was used to minimize the chi-square of the fit of the ellipse to the measured surface coordinates. The parameters in question were x_0, y_0, z_0, θ_x, θ_z, a, and b. The chi-square computation was done by the attached C program `chi_sq.c`. The minimization was done by `fit_surface.c`, which references `amoeba.c` which was taken from *Numerical Recipes*. It was found, though, that three initial parameters of `fit_surface.c` itself significantly influenced the final fit. Therefore, a second "outer" round of minimization was applied to the results of `fit_surface.c` by the `best_fit.c` program. The parameters in question in this outer minimization were the fractional tolerance `ftol` which controls the termination of the inner amoeba routine, the initialization factor `length` which was used by the inner minimization to initialize the simplex for non-zero parameters, and the arbitrary simplex initialization value `zero_init` for parameters with inital values of zero (z_0 and θ_x) that could not be "initialized away" from zero simply by multiplying by `length`. The best-fit values obtained through this process are shown in Table 1.

Table 1: Original & Best-Fit Mirror Surface Specifications

	AST/RO (*Original*)	**AST/RO** (*Best-Fit*)		**SPIFI** (*Original*)	**SPIFI** (*Best-Fit*)
x_0	-79.3750 mm	-78.8636 mm	x_0	-81.3300 mm	-82.7762 mm
y_0	1820.9390 mm	1820.9413 mm	y_0	494.8500 mm	495.4400 mm
z_0	0.0000 mm	0.0041 mm	z_0	0.0000 mm	0.0047 mm
θ_x	0.0000 $^\circ$	-0.0074 $^\circ$	θ_x	0.0000 $^\circ$	-0.0021 $^\circ$
θ_z	27.2208 $^\circ$	27.2663 $^\circ$	θ_z	31.3950 $^\circ$	31.6245 $^\circ$
a	1891.0990 mm	1891.0191 mm	a	570.0170 mm	570.4816 mm
b	606.8110 mm	604.4725 mm	b	321.2500 mm	322.0508 mm

The average deviation associated with the best-fit ellipse for the AST/RO mirror is -0.3 μm with an RMS of 4.3 μm, which is below the maximum acceptable RMS of 5.0 μm given in the original specifications for the mirrors. This mirror is good enough to be used in AST/RO as it is. However, the best-fit ellipse for the SPIFI mirrors has an associated RMS of 32.6 μm with an average deviation of -0.7 μm. Attached to this memorandum is a rough contour plot done in Microsoft Excel displaying the deviations as functions of y-position. It appears to indicate a systematic deformity across the entire surface of the mirror. There is a possibility that this simply indicates the poor calibration of CMM (despite BU's assurances that the machine was fully operational and reliable at the time of the measurements) and the mirror is in fact a usable spheroid section in its present condition, although an additional round of measurements will be necessary to verify this and also to ascertain the parameters of the best-fit spheroid. It is also possible that no good spheroid fit exists for the surface in which case the SPIFI mirror may have to be re-machined.

Since the AST/RO mirror meets specifications, it is now left to determine the positions of the four measuring balls so that the mirror can be positioned properly within the AST/RO telescope system. There are three criteria which fix the position of the secondary mirror:

(1) Both the near and far focii of the secondary mirror as well as the mirror center lie in the y=0 plane.

(2) The near focus of the secondary mirror coincides with the focus of the primary mirror, defined to be at (0 mm, 0 mm, 1200 mm) in the telescope coordinate system.

(3) The center of the primary mirror at (1150 mm, 0 mm, 275.521 mm), the near focus of the secondary mirror at (0 mm, 0 mm, 1200 mm), and the center of the secondary mirror are colinear. The "center" of the secondary mirror is defined as the point at which a ray passing through the near focus can strike the mirror, be reflected through a half angle of 24.8594°, and then finally pass through the far focus.

The first two criteria can be met by applying the same transformations on the local coordinates of the measuring balls that were used to translate the measured surface coordinates of the mirrors into the ellipse system, and then applying two additional transformations to shift from the ellipse system to the telescope system. The labels for the y and z axes need to be transposed and a translation along the new z axis is required to align the primary focus and the secondary near focus.

$$\begin{pmatrix} x''_{ellipse} \\ y''_{ellipse} \\ z''_{ellipse} \end{pmatrix} = \begin{pmatrix} \cos\theta_z & \sin\theta_x \sin\theta_z & -\cos\theta_x \sin\theta_z \\ \sin\theta_z & -\sin\theta_x \cos\theta_z & \cos\theta_x \cos\theta_z \\ 0 & -\cos\theta_x & -\sin\theta_x \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \tag{6}$$

$$\begin{pmatrix} x''_{telescope} \\ y''_{telescope} \\ z''_{telescope} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x''_{ellipse} \\ y''_{ellipse} \\ z''_{ellipse} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -(c - \mathrm{f}_1) \end{pmatrix} \tag{7}$$

where f_1 is the focal length of primary mirror ($f_1 = 1200$ mm) and $c = \sqrt{a^2 - b^2}$. The original local coordinates of the measuring balls are taken to be (±78.740 mm,±119.380 mm, 0 mm).

To satisfy the third criterion, first the coordinates in the ellipse system of the center of the secondary mirror must be determined. Then, this point must be translated into the telescope system. Next one must determine and compare the angles of the vectors running from the primary center to the primary focus and from the primary focus to the center of the secondary mirror. Finally, any discrepancy must be remedied by a compensatory rotation about an axis parallel to the y axis and passing through the primary focus.

Consider the cross-section of the ellipse lying in the x-y plane. If a ray passing through the near focus is to strike the mirror center and then pass through the far focus, reflecting through a half-angle of 24.8594°, then the vectors from an arbitrary point (x, y) on the ellipse drawn to each focus at $(0\ mm, \pm c)$ must form an angle twice as large as the specified half-angle of reflection, or 49.7188°.

$$\begin{aligned} \vec{v} &= (0-x)\vec{\imath} + (-c-y)\vec{\jmath} & \vec{w} &= (0-x)\vec{\imath} + (c-y)\vec{\jmath} \\ \|\vec{v}\| &= \sqrt{x^2 + (-c-y)^2} & \|\vec{w}\| &= \sqrt{x^2 + (c-y)^2} \end{aligned} \tag{8}$$

$$\vec{v} \cdot \vec{w} = \|\vec{v}\|\ \|\vec{w}\| \cos 49.7188 \tag{9}$$

Combining the Equations 1 and 9 and solving for x and y gives the secondary mirror center position as (-95.2515 mm, 0 mm, 1867.6556 mm) in the ellipse coordinate system. The translation (by subtracting $(c - f_1)$ from the y coordinate) gives the position as (-95.2515 mm, 0 mm, 1276.5565 mm).

Now, one must consider the angles of the vectors running from the primary mirror center to the primary focus and from the primary focus to the secondary mirror center.

$$\begin{aligned} \phi_1 &= \tan^{-1} \frac{1200mm - 275.521mm}{0mm - 1150mm} \\ &= -38.7956 \\ \phi_2 &= \tan^{-1} \frac{1276.5565mm - 1200mm}{95.2515mm - 0mm} \\ &= -38.78999 \end{aligned} \tag{10}$$

There is a discrepancy of 0.00571°. The most straight-forward way to compensate for this is to translate the primary focus to the origin, carrying all the other points with it, apply the necessary compensatory rotation about the y axis, and then translate the primary focus back to its original position at (0 mm, 0 mm, 1200 mm).

$$\begin{pmatrix} x_{rotated} \\ y_{rotated} \\ z_{rotated} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x_{unrotated} \\ y_{unrotated} \\ z_{unrotated} - 1200mm \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1200mm \end{pmatrix} \tag{11}$$

The new positions for the measuring balls are given in Table 2. All coordinates are given in millimeters. It should also be noted that this rotation changes the position of the secondary mirror's far focus from (0.0000 mm, 0.0000 mm, -1791.8053 mm) to (-0.2983 mm, 0.0000 mm, -1791.8053 mm).

Table 2: Measuring Ball Positions for the AST/RO Secondary Mirror

Local Coordinates	**In Telescope System** *For Original Parameters*	**In Telescope System** *For Best-fit Paramaters*
(78.7400, 119.3800, 0.0000)	(-9.3554, -119.3800, 1265.8573)	(-8.8862, -119.3760, 1265.2218)
(78.7400, -119.3800, 0.0000)	(-9.3554, 119.3800, 1265.8573)	(-8.8721, 119.3841, 1265.2943)
(-78.7400, 119.3800, 0.0000)	(-149.3946, -119.3800, 1193.8226)	(-148.8609, -119.3759, 1193.0620)
(-78.7400, -119.3800, 0.0000)	(-149.3946, 119.3800, 1193.8226)	(-148.8467, 119.3841, 1193.0346)

REFERENCES

Ingalls, J.G. 1997, "`align`": *A Computer Program for the Alignment of the AST/RO Primary-Secondary Mirror System*, unpublished

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. 1992, *Numerical Recipes* (2nd edition), (New York: Cambridge University Press), 408-412

Stark, A.A., August 1998 - October 1998, ongoing personal conversations

Stark, A.A., Cheng, J., Chamberlin, R., Wright, G. 1994, *Equations of the Secondary Mirrors for AST/RO*, AST/RO Technical Memorandum #11

Stark, A.A., Chamberlin, R.A., Ingalls, J.G., Cheng, J., Wright, G. 1997, *Optical and mechanical design of the Antarctic Submillimeter Telescope and Remote Observatory*, Rev. Sci. Inst., 68 (5), 2200

Appendix 1: Coordinate Measurements for the AST/RO Secondary Mirror Surface

All measurements in millimeters

x	y	z
-81.428	-104.268	-14.106
-78.952	70.298	-27.190
-78.796	81.414	-23.688
-78.570	97.288	-17.734
-62.476	-110.886	-15.004
-62.114	-85.496	-25.874
-61.798	-63.272	-33.036
-61.436	-37.884	-38.592
-61.034	-9.324	-41.542
-60.654	17.656	-41.140
-60.266	44.640	-37.646
-59.706	84.316	-26.766
-59.304	112.884	-14.544
-40.274	-111.202	-18.062
-39.932	-87.394	-28.698
-39.572	-62.006	-37.144
-39.212	-36.612	-42.696
-38.854	-11.226	-45.400
-38.460	16.510	-45.132
-38.022	47.494	-40.844
-37.636	74.476	-33.612
-37.298	98.288	-24.454
-37.052	115.742	-16.026
-18.238	-124.216	-12.428
-17.898	-100.404	-24.952
-17.628	-81.354	-32.962
-17.310	-59.144	-40.118
-16.976	-35.338	-45.246
-16.436	2.748	-48.088
-15.894	40.832	-44.364
-15.534	66.230	-38.184
-15.172	91.624	-28.954
-14.838	115.424	-17.434
-0.788	-124.464	-12.032
-0.580	-110.172	-20.148
-0.244	-86.372	-31.314
0.098	-62.560	-39.676
0.522	-32.412	-46.366
0.994	0.910	-48.806
1.490	35.828	-45.820
1.896	64.400	-39.102
2.280	91.388	-29.112
2.592	113.594	-18.134

x	y	z
2.728	123.122	-12.620
19.862	-123.172	-10.910
20.202	-99.360	-24.154
20.472	-80.314	-32.578
22.276	-64.462	-38.032
22.636	-39.064	-44.402
23.038	-10.512	-47.798
23.468	19.644	-47.090
23.848	46.620	-42.716
24.256	75.184	-34.140
24.640	102.174	-22.168
24.930	122.794	-10.362
46.986	-112.438	-12.116
47.280	-91.806	-23.160
47.730	-60.066	-35.562
48.044	-37.844	-41.076
48.470	-7.698	-44.486
48.874	20.866	-43.448
49.258	47.852	-38.622
49.620	73.244	-30.596
49.870	90.706	-23.040
50.130	108.972	-13.302
67.778	-101.620	-11.918
67.932	-90.516	-17.758
68.178	-73.050	-25.476
68.468	-52.422	-32.382
68.854	-25.442	-37.890
69.302	6.292	-39.374
69.710	34.858	-36.106
70.070	60.254	-29.476
70.316	77.718	-22.822
70.630	99.938	-11.782
80.636	-90.690	-12.702
80.906	-71.638	-21.334
81.308	-43.078	-30.304
81.738	-12.928	-34.796
82.142	15.644	-34.430
82.574	45.804	-29.120
83.042	79.126	-17.228
90.742	-47.972	-25.088
91.126	-20.996	-30.196
91.532	7.568	-31.096
91.922	34.544	-27.696

Appendix 2: Coordinate Measurements for the SPIFI Secondary Mirror Surface

All measurements in millimeters

x	y	z
-71.154	-92.766	-18.846
-70.740	-75.250	-25.882
-70.314	-57.008	-31.602
-69.794	-34.794	-36.422
-69.350	-15.750	-38.732
-68.850	5.688	-39.354
-68.462	22.336	-38.384
-68.128	36.612	-36.542
-67.796	50.898	-33.716
-67.280	73.110	-27.456
-66.580	103.258	-14.976
-49.178	-103.542	-18.190
-48.730	-84.510	-26.982
-48.168	-60.706	-35.346
-47.680	-40.082	-40.340
-47.198	-19.450	-43.302
-46.610	5.938	-44.208
-46.132	26.574	-42.730
-45.504	53.550	-37.756
-44.838	82.106	-28.602
-44.362	102.738	-19.350
-25.372	-104.108	-20.722
-24.630	-72.366	-34.788
-23.962	-43.804	-42.942
-23.182	-10.480	-47.376
-22.330	26.016	-46.096
-21.556	59.340	-39.266
-20.818	91.076	-27.478
-20.228	116.472	-14.022
4.480	-117.482	-13.140
4.964	-96.868	-24.954
5.336	-80.994	-32.326
5.948	-54.902	-41.418

x	y	z
6.526	-30.228	-46.716
7.230	-0.080	-48.998
7.860	26.898	-47.170
8.560	57.052	-40.726
9.260	87.198	-29.412
9.920	115.768	-13.810
31.626	-110.180	-14.294
32.146	-87.968	-26.516
33.076	-48.310	-41.006
33.596	-26.094	-45.312
34.152	-2.292	-47.016
34.854	27.856	-44.884
35.444	53.246	-39.312
35.978	76.174	-31.170
36.314	90.660	-24.428
36.716	108.078	-14.552
49.404	-96.314	-18.392
50.152	-64.580	-32.752
50.744	-39.196	-39.920
51.262	-16.976	-43.226
51.892	10.004	-43.600
52.668	43.318	-38.532
53.370	73.468	-28.462
53.704	87.752	-21.748
70.024	-96.796	-11.142
70.544	-74.576	-22.488
71.290	-42.852	-33.186
71.844	-19.046	-37.260
72.622	14.280	-37.446
73.290	42.838	-32.474
74.100	77.748	-19.638
74.322	87.276	-14.742

```
/*  stats.c  */
/*  9/30/98 by Henry Hsieh  */
/*  hhsieh@cfa.harvard.edu  */
/*  hsieh@fas.harvard.edu   */
/*  Reads in coordinates of secondary mirror surfaces for AST/RO and SPIFI
    and returns statistical data (average, variance, rms, chi square) of
    the deviations of the measured z-values from the calculated z-values
    given the previously obtained best-fit spheroid parameters.    */
#include <math.h>
#define INCHES  0
#define MM      1
#define PI      3.141592654
#define SIGMA   0.002
#define MIRROR_NAME "SPIFI Mirror"
#define MIRROR "mirror1.txt"
#define MIRROR_NUM 1
#define OUTPUT "out1-original.txt"
#define FIXED 0
#define ORIGINAL 1
double zcalc(double x_zcalc, double y_zcalc, int units,
              double theta_x_zcalc, double theta_z_zcalc,
              double x0_zcalc, double y0_zcalc, double z0_zcalc,
              double a_zcalc, double b_zcalc);
double input[100][3];
int num_pts;
double theta_x, theta_z, x0, yy0, z0, a, b;
void main(void)
{
  double x, y, sum_deviations, sum_squares, avg_deviation, temp_x, temp_y;
  double rms, deviation, chi_square, variance, meas_z, calc_z, temp_z;
  double output[100][4];
  char *p;
  int i, j, k;
  FILE *mirror, *out;
  /* Parameter initialization for SPIFI mirror */
  if(MIRROR_NUM == 1) num_pts = 65;
  /* Parameter initialization for AST/RO mirror */
  else if(MIRROR_NUM == 2) num_pts = 86;
  else {
    printf("Mirror file name not recognized.\n");
    exit(1);
  }
  if((mirror = fopen(MIRROR, "r"))==NULL) {
    printf("Cannot open ");
    printf(MIRROR);
    printf("\n");
    exit(1);
  }
  if((out = fopen(OUTPUT, "w"))==NULL) {
    printf("Cannot open ");
    printf(OUTPUT);
    printf("\n");
    exit(1);
  }
  /* Read coordinate data into an array */
  for(i=0;i<num_pts;i++) {
    for(j=0;j<3;j++) {
      fscanf(mirror, "%lf", &input[i][j]);
    }
  }
/* for AST/RO mirror - original ellipse */
  if(MIRROR_NUM == 2 && ORIGINAL) {
    fprintf(out, "AST/RO - original ellipse\n");
    theta_x = 0.0000;
    theta_z = 27.2208;
    x0 = -79.375;
    yy0 = 1820.939;
```

```
    z0 = 0.000;
    a = 1891.099000;
    b = 606.811000;
  }
/* for AST/RO mirror - fixed ellipse */
  if(MIRROR_NUM == 2 && FIXED && !ORIGINAL) {
    fprintf(out, "AST/RO - fixed ellipse\n");
    theta_x = 0.001679;
    theta_z = 27.266211;
    x0 = -79.529643;
    yy0 = 1821.043169;
    z0 = 0.002431;
    a = 1891.099000;
    b = 606.811000;
  }
/* for AST/RO mirror - free ellipse */
  if(MIRROR_NUM == 2 && !FIXED && !ORIGINAL) {
    fprintf(out, "AST/RO - free ellipse\n");
    theta_x = -0.007413;
    theta_z = 27.266307;
    x0 = -78.863572;
    yy0 = 1820.941340;
    z0 = 0.004084;
    a = 1891.019111;
    b = 604.472526;
  }
/* for SPIFI mirror - original ellipse */
  if(MIRROR_NUM == 1 && ORIGINAL) {
    fprintf(out, "SPIFI - original ellipse\n");
    theta_x = 0.0000;
    theta_z = 31.3950;
    x0 = -81.330;
    yy0 = 494.850;
    z0 = 0.000;
    a = 570.017000;
    b = 321.250000;
  }
/* for SPIFI mirror - fixed ellipse */
  if(MIRROR_NUM == 1 && FIXED && !ORIGINAL) {
    fprintf(out, "SPIFI - fixed ellipse\n");
    theta_x = 0.010114;
    theta_z = 32.079538;
    x0 = -83.863039;
    yy0 = 494.141865;
    z0 = 0.003844;
    a = 570.017000;
    b = 321.250000;
  }
/* for SPIFI mirror - free ellipse */
  if(MIRROR_NUM == 1 && !FIXED && !ORIGINAL) {
    fprintf(out, "SPIFI - free ellipse\n");
    theta_x = -0.002106;
    theta_z = 31.624470;
    x0 = -82.776193;
    yy0 = 495.440030;
    z0 = 0.004675;
    a = 570.481604;
    b = 322.050779;
  }
  sum_deviations = 0;
  sum_squares = 0;
  chi_square = 0;
/* Sort by ascending x-values */
  for(i=0;i<num_pts-1;i++) {
    for(j=i+1;j<num_pts;j++) {
      if(input[i][1]>input[j][1]) {
        temp_x = input[i][1];
        input[i][1] = input[j][1];
        input[j][1] = temp_x;
```

```
        temp_y = input[i][0];
        input[i][0] = input[j][0];
        input[j][0] = temp_y;
        temp_z = input[i][2];
        input[i][2] = input[j][2];
        input[j][2] = temp_z;
      }
    }
  }
/* Calculate accepted values for Z and determine average deviation */
  fprintf(out, "Excel data:    x, y, deviation\n\n");
  for(i=0;i<num_pts;i++) {
    x            = input[i][1]/25.4;
    y            = input[i][0]/25.4;
    meas_z       = input[i][2];
    output[i][0] = meas_z;
    fprintf(out, "%lf\t%lf\t",input[i][1],input[i][0]);
    calc_z = -(zcalc(x,y,INCHES,theta_x,theta_z,
                  x0,yy0,z0,a,b)-0.3125+0.3125)*25.4;
    output[i][1] = calc_z;
    /* deviation */
    deviation = meas_z - calc_z;
    output[i][2] = deviation;
    fprintf(out,"%lf\n",deviation);
    sum_deviations += deviation;
    sum_squares += (deviation*deviation);
  }
  avg_deviation = sum_deviations / num_pts;
  variance = sum_squares / num_pts;
  rms = sqrt(variance);
  chi_square = sum_squares / (SIGMA*SIGMA);
  printf("\n");
  printf("Theta(X): %lf\t Average Deviation:    %lf\n", theta_x, avg_deviation);
  printf("Theta(Z): %lf\t Variance:    %lf\n", theta_z, variance);
  printf("x0:              %lf\t RMS Diff of Deviations:    %lf\n", x0, rms);
  printf("y0:              %lf\t Chi Square:    %lf\n", yy0, chi_square);
  printf("z0:              %lf\n", z0);
  printf("a:                %lf\n", a);
  printf("b:                %lf\n\n", b);
  printf("\n");
  fprintf(out,"\n");
  fprintf(out,"Theta(X): %lf\t Average Deviation:    %lf\n", theta_x, avg_deviation);
  fprintf(out,"Theta(Z): %lf\t Variance:    %lf\n", theta_z, variance);
  fprintf(out,"x0:    %lf\t RMS Diff of Deviations:    %lf\n", x0, rms);
  fprintf(out,"y0:    %lf\t Chi Square:    %lf\n", yy0, chi_square);
  fprintf(out,"z0:              %lf\n", z0);
  fprintf(out,"a:                %lf\n", a);
  fprintf(out,"b:                %lf\n\n", b);
  fclose(mirror);
  fclose(out);
}
double zcalc(x_zcalc, y_zcalc, units, theta_x_zcalc, theta_z_zcalc,
                x0_zcalc, y0_zcalc, z0_zcalc, a_zcalc, b_zcalc)
  double x_zcalc, y_zcalc, theta_x_zcalc, theta_z_zcalc, x0_zcalc, y0_zcalc,
        z0_zcalc, a_zcalc, b_zcalc;
  int units;
{
      double z, alpha, beta, gamma, delta, epsilon, kappa;
      double ctheta_x, stheta_x, ctheta_z, stheta_z, temp1, temp2;
      if (units == INCHES)  { x_zcalc *= 25.4; y_zcalc *= 25.4;}
      ctheta_x = cos(theta_x_zcalc * PI /180.);
      stheta_x = sin(theta_x_zcalc * PI /180.);
      ctheta_z = cos(theta_z_zcalc * PI /180.);
```

```
        stheta_z = sin(theta_z_zcalc * PI /180.);
        alpha = ctheta_x * ctheta_z / a_zcalc;
        beta = (x_zcalc*stheta_z - y_zcalc*stheta_x*ctheta_z
                   + y0_zcalc) / a_zcalc;
        gamma = - ctheta_x * stheta_z / b_zcalc;
        delta = (x_zcalc*ctheta_z + y_zcalc*stheta_x*stheta_z
                   + x0_zcalc)/b_zcalc;
        epsilon = - stheta_x / b_zcalc;
        kappa = (-y_zcalc*ctheta_x + z0_zcalc) / b_zcalc;
        temp1 = alpha*beta + gamma*delta + epsilon*kappa;
        temp2 = alpha*alpha + gamma*gamma + epsilon*epsilon;
        z = ( - temp1 + sqrt(temp1*temp1 - temp2*
             (beta*beta + delta*delta + kappa*kappa - 1.0)))/temp2;
        if (units == INCHES) z /= 25.4;
        return (z);
}
```

```
/*  best_fit.c  */
/*  9/30/98 by Henry Hsieh  */
/*  hhsieh@cfa.harvard.edu  */
/*  hsieh@fas.harvard.edu   */
/*  Requires fit_surface.c, chi_sq.c, amoeba.c, nrutil.c, and nr.h          */
/*  cc best_fit.c fit_surface.c chi_sq.c amoeba.c nrutil.c -o best_fit -lm  */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include "nr.h"
#include "nrutil.h"
#define NDIM 3
#define FIT_LENGTH 0.93
double length, zero_init, ftol_fit;
void call_amoebaB(void);
void main(void)
{
  /* Parameter initialization */
  length = 1.01;
  zero_init = 0.01;
  ftol_fit = 0.1;
  call_amoebaB();
}
void call_amoebaB()
{
  double **p;
  double y[NDIM + 2];
  double ftol, temp;
  double funkB();
  int nfunk;
  int i, j;
  void amoeba(), printpB();
  p = dmatrix(1,NDIM+1,1,NDIM);
  /* Initial parameters */
  printf("LENGTH:     %lf\n", length);
  printf("ZERO_INIT:  %lf\n", zero_init);
  printf("FTOL:       %lf\n", ftol_fit);
  ftol = 0.00001;
  /* Initializing the simplex */
  for(i=1;i<NDIM+2;i++) {
    p[i][1] = length;
    p[i][2] = zero_init;
    p[i][3] = ftol_fit;
  }
  /* Setting up the actual simplex */
  for(i=1;i<NDIM+1;i++) {
    p[i+1][i] *= FIT_LENGTH;
  }
  printpB(p);
  for(i=1;i<NDIM+2;i++) {
    y[i] = funkB(p[i]);
  }
  /* The Amoeba */
  amoeba(p,y,NDIM,ftol,funkB,&nfunk);
  printpB(p);
  free_dmatrix(p,1,NDIM+1,1,NDIM);
}
double funkB(x)
double x[];
{
  double length, zero_init, ftol_fit;
  double fit_surface(double lengthA, double zero_initA, double ftol_fitA);
  length = x[1];
  zero_init = x[2];
```

```
  ftol_fit = x[3];
  return ( fit_surface(length, zero_init, ftol_fit) );
}
void printpB(p)
double **p;
{
  printf("\nLENGTH:     %8.3lf  %8.3lf  %8.3lf  %8.3lf",
          p[1][1], p[2][1], p[3][1], p[4][1]);
  printf("\nZERO_INIT:  %8.3lf  %8.3lf  %8.3lf  %8.3lf",
          p[1][2], p[2][2], p[3][2], p[4][2]);
  printf("\nFTOL:       %8.3lf  %8.3lf  %8.3lf  %8.3lf",
          p[1][3], p[2][3], p[3][3], p[4][3]);
  printf("\n\n");
}
```

```
/*  fit_surface.c  */
/*  9/30/98 by Henry Hsieh   */
/*  hhsieh@cfa.harvard.edu   */
/*  hsieh@fas.harvard.edu    */
/*  Required for best_fit.c  */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include "nr.h"
#include "nrutil.h"
#define INCHES  0
#define MM      1
#define PI      3.141592654
#define MIRROR_NAME "SPIFI Mirror"
#define MIRROR "mirror1.txt"
#define MIRROR_NUM 1
#define OUTPUT "out1.txt"
#define NDIM_FIT 7
double input_coords[100][3];
int numpts;
double theta_x_mirror, theta_z_mirror, x0_mirror, y0_mirror, z0_mirror;
double a_mirror, b_mirror;
double theta_x_final, theta_z_final, x0_final, y0_final, z0_final;
double a_final, b_final;
double length, zero_init, ftol_fit;
void call_amoebaA(void);
double chi_sq(int num_pts, double input[100][3], double theta_x, double theta_z,
                  double x0, double y0, double z0, double a, double b);
double fit_surface(double lengthA, double zero_initA, double ftol_fitA)
{
  int i, j, k;
  FILE *mirror;
  char *p;
  length = lengthA;
  zero_init = zero_initA;
  ftol_fit = ftol_fitA;
  /* Parameter initialization for SPIFI mirror */
  if(MIRROR_NUM == 1) numpts = 65;
  /* Parameter initialization for AST/RO mirror */
  else if(MIRROR_NUM == 2) numpts = 86;
  else {
    printf("Mirror file name not recognized.\n");
    exit(1);
  }
  if((mirror = fopen(MIRROR, "r"))==NULL) {
    printf("Cannot open ");
    printf(MIRROR);
    printf("\n");
    exit(1);
  }
  /* Read coordinate data into an array */
  for(i=0;i<numpts;i++) {
    for(j=0;j<3;j++) {
      fscanf(mirror, "%lf", &input_coords[i][j]);
    }
  }
  /* Coordinate check */
  printf("%lf    %lf    %lf\n", input_coords[0][0], input_coords[0][1],
    input_coords[0][2]);
  /* Parameter initialization for SPIFI mirror */
  if(MIRROR_NUM == 1) {
    theta_x_mirror = 0.0;
    theta_z_mirror = 31.395;
    x0_mirror = -81.33;
    y0_mirror = 494.850;
    z0_mirror = 0.0;
```

```
    a_mirror = 570.017;
    b_mirror = 321.250;
  }
  /* Parameter initialization for AST/RO mirror */
  else if(MIRROR_NUM == 2) {
    theta_x_mirror = 0.0;
    theta_z_mirror = 27.2208;
    x0_mirror = -79.375;
    y0_mirror = 1820.939;
    z0_mirror = 0.0;
    a_mirror = 1891.099;
    b_mirror = 606.811;
  }
  else {
    printf("Mirror file name not recognized.\n");
    exit(1);
  }
  call_amoebaA();
  fclose(mirror);
  return ( chi_sq(numpts, input_coords, theta_x_final, theta_z_final,
             x0_final, y0_final, z0_final, a_final, b_final) );
}
void call_amoebaA()
{
  double **p;
  double small[NDIM_FIT + 1];
  double y[NDIM_FIT + 2];
  double ftol, temp, smallest;
  double funkA();
  int nfunk;
  int i, j;
  void amoeba(), printpA();
  p = dmatrix(1,NDIM_FIT+1,1,NDIM_FIT);
  /* Initial Parameters */
  printf("Theta(X):  %lf\n", theta_x_mirror);
  printf("Theta(Z):  %lf\n", theta_z_mirror);
  printf("x0:                %lf\n", x0_mirror);
  printf("z0:                %lf\n", z0_mirror);
  printf("a:                  %lf\n", a_mirror);
  printf("b:                  %lf\n", b_mirror);
  ftol = ftol_fit;
  /* Initializing the simplex */
  for(i=1;i<NDIM_FIT+2;i++) {
    p[i][1] = theta_x_mirror;
    p[i][2] = theta_z_mirror;
    p[i][3] = x0_mirror;
    p[i][4] = y0_mirror;
    p[i][5] = z0_mirror;
    p[i][6] = a_mirror;
    p[i][7] = b_mirror;
  }
  /* Setting up the actual simplex */
  for(i=1;i<NDIM_FIT+1;i++) {
    p[i+1][i] *= length;
  }
  /* Setting up the simplex for the zero values */
  p[2][1] = zero_init;
  p[6][5] = zero_init;
  printpA(p);
  for(i=1;i<NDIM_FIT+2;i++) {
    y[i] = funkA(p[i]);
  }
  /* The Amoeba */
  amoeba(p,y,NDIM_FIT,ftol,funkA,&nfunk);
  printpA(p);
```

```
  free_dmatrix(p,1,NDIM_FIT+1,1,NDIM_FIT);
  smallest = 10000;
  for(i=1;i<NDIM_FIT+1;i++) {
    if( (chi_sq(numpts, input_coords, p[i][1], p[i][2], p[i][3],
          p[i][4], p[i][5], p[i][6], p[i][7]))<smallest) {
      theta_x_final = p[i][1];
      theta_z_final = p[i][2];
      x0_final = p[i][3];
      y0_final = p[i][4];
      z0_final = p[i][5];
      a_final = p[i][6];
      b_final = p[i][7];
    }
  }
}
double funkA(x)
double x[];
{
  double theta_x_mirror, theta_z_mirror, x0_mirror, y0_mirror, z0_mirror;
  theta_x_mirror = x[1];
  theta_z_mirror = x[2];
  x0_mirror = x[3];
  y0_mirror = x[4];
  z0_mirror = x[5];
  a_mirror = x[6];
  b_mirror = x[7];
  return ( chi_sq(numpts, input_coords, theta_x_mirror, theta_z_mirror,
             x0_mirror, y0_mirror, z0_mirror, a_mirror, b_mirror) );
}
void printpA(p)
double **p;
{
  printf("\nLENGTH:  %5.4lf       ZERO_INIT:  %5.4lf\n", length, zero_init);
  printf("\nTheta(X):  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf",
             p[1][1], p[2][1], p[3][1], p[4][1], p[5][1], p[6][1]);
  printf("  %-8.3lf  %-8.3lf", p[7][1], p[8][1]);
  printf("\nTheta(Z):  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf",
             p[1][2], p[2][2], p[3][2], p[4][2], p[5][2], p[6][2]);
  printf("  %-8.3lf  %-8.3lf", p[7][2], p[8][2]);
  printf("\nx0:                %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf",
             p[1][3], p[2][3], p[3][3], p[4][3], p[5][3], p[6][3]);
  printf("  %-8.3lf  %-8.3lf", p[7][3], p[8][3]);
  printf("\ny0:                %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf",
             p[1][4], p[2][4], p[3][4], p[4][4], p[5][4], p[6][4]);
  printf("  %-8.3lf  %-8.3lf", p[7][4], p[8][4]);
  printf("\nz0:                %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf",
             p[1][5], p[2][5], p[3][5], p[4][5], p[5][5], p[6][5]);
  printf("  %-8.3lf  %-8.3lf", p[7][5], p[8][5]);
  printf("\na:              %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf",
             p[1][6], p[2][6], p[3][6], p[4][6], p[5][6], p[6][6]);
  printf("  %-8.3lf  %-8.3lf", p[7][6], p[8][6]);
  printf("\nb:              %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf  %-8.3lf",
             p[1][7], p[2][7], p[3][7], p[4][7], p[5][7], p[6][7]);
  printf("  %-8.3lf  %-8.3lf", p[7][7], p[8][7]);
  printf("\n\n");
}
```

```
/*  chi_sq.c  */
/*  9/30/98 by Henry Hsieh   */
/*  hhsieh@cfa.harvard.edu   */
/*  hsieh@fas.harvard.edu    */
/*  Required for best_fit.c  */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define INCHES  0
#define MM      1
#define PI      3.141592654
#define SIGMA   0.010
double zcalc(double x_zcalc, double y_zcalc, int units,
              double theta_x_zcalc, double theta_z_zcalc, double x0_zcalc,
              double y0_zcalc, double z0_zcalc, double a_zcalc, double b_zcalc);
double chi_sq(int num_pts, double input[100][3], double theta_x, double
                  theta_z, double x0, double y0, double z0, double a, double b)
{
  double x, y, sum_deviations, sum_squares, avg_deviation;
  double deviation, rms, chi_square, variance, calc_z, meas_z;
  double output[100][4];
  int i, j, k;
  sum_deviations = 0;
  sum_squares = 0;
  for(i=0;i<num_pts;i++) {
    x            = input[i][1]/25.4;
    y            = input[i][0]/25.4;
    meas_z = input[i][2];
    output[i][0] = meas_z;
    calc_z = -(zcalc(x,y,INCHES,theta_x,theta_z,x0,y0,z0,a,b)-0.3125+0.3125)*25.4;
    output[i][1] = calc_z;
    deviation = calc_z - meas_z;
    output[i][2] = deviation;
    sum_deviations += deviation;
    sum_squares += (deviation*deviation);
  }
  avg_deviation = sum_deviations / num_pts;
  variance = sum_squares / num_pts;
  rms = sqrt(variance);
  chi_square = sum_squares / (SIGMA*SIGMA);
  printf("\n");
  printf("Theta(X):  %lf\t Average Deviation:    %lf\n",theta_x, avg_deviation);
  printf("Theta(Z):  %lf\t Variance:    %lf\n", theta_z, variance);
  printf("x0:    %lf\t RMS Diff of Deviations:    %lf\n", x0, rms);
  printf("y0:    %lf\t Chi Square:    %lf\n", y0, chi_square);
  printf("z0:    %lf\n", z0);
  printf("a:      %lf\n", a);
  printf("b:      %lf\n", b);
  printf("Chi Square:    %lf\n", chi_square);
  return(chi_square);
}
double zcalc(x_zcalc, y_zcalc, units, theta_x_zcalc, theta_z_zcalc,
                x0_zcalc, y0_zcalc, z0_zcalc, a_zcalc, b_zcalc)
  double x_zcalc, y_zcalc, theta_x_zcalc, theta_z_zcalc
  double x0_zcalc, y0_zcalc, z0_zcalc, a_zcalc, b_zcalc;
  int units;
{
       double z, alpha, beta, gamma, delta, epsilon, kappa;
       double ctheta_x, stheta_x, ctheta_z, stheta_z, temp1, temp2;
       if (units == INCHES)  { x_zcalc *= 25.4; y_zcalc *= 25.4;}
       ctheta_x = cos(theta_x_zcalc * PI /180.);
       stheta_x = sin(theta_x_zcalc * PI /180.);
       ctheta_z = cos(theta_z_zcalc * PI /180.);
       stheta_z = sin(theta_z_zcalc * PI /180.);
       alpha = ctheta_x * ctheta_z / a_zcalc;
```

```
        beta = (x_zcalc*stheta_z - y_zcalc*stheta_x*ctheta_z
                   + y0_zcalc) / a_zcalc;
        gamma = - ctheta_x * stheta_z / b_zcalc;
        delta = (x_zcalc*ctheta_z + y_zcalc*stheta_x*stheta_z
                   + x0_zcalc)/b_zcalc;
        epsilon = - stheta_x / b_zcalc;
        kappa = (-y_zcalc*ctheta_x + z0_zcalc) / b_zcalc;
        temp1 = alpha*beta + gamma*delta + epsilon*kappa;
        temp2 = alpha*alpha + gamma*gamma + epsilon*epsilon;
        z = ( - temp1 + sqrt(temp1*temp1 - temp2*
             (beta*beta + delta*delta + kappa*kappa - 1.0)))/temp2;
        if (units == INCHES) z /= 25.4;
        return (z);
}
```

```
/* amoeba.c */
#include <math.h>
#define NMAX 50000
#define ALPHA 1.0
#define BETA 0.5
#define GAMMA 2.0
#define GET_PSUM for (j=1;j<=ndim;j++) { for (i=1,sum=0.0;i<=mpts;i++)\
                  sum += p[i][j]; psum[j]=sum;}
void amoeba(p,y,ndim,ftol,funk,nfunk)
double **p,y[],ftol,(*funk)();
int ndim,*nfunk;
{
  int i,j,ilo,ihi,inhi,mpts=ndim+1;
  double ytry,ysave,sum,rtol,amotry(),*psum,*dvector();
  void nrerror(),free_dvector();
  psum=dvector(1,ndim);
  *nfunk=0;
  GET_PSUM
  for (;;) {
    ilo=1;
    ihi = y[1]>y[2] ?  (inhi=2,1) :  (inhi=1,2);
    for (i=1;i<=mpts;i++) {
      if (y[i] < y[ilo]) ilo=i;
      if (y[i] > y[ihi]) {
        inhi=ihi;
        ihi=i;
      } else if (y[i] > y[inhi])
        if (i != ihi) inhi=i;
    }
    rtol=2.0*fabs(y[ihi]-y[ilo])/(fabs(y[ihi])+fabs(y[ilo]));
    if (rtol < ftol) break;
    if (*nfunk >= NMAX) break;
/*
    if (*nfunk >= NMAX) nrerror("Too many iterations in AMOEBA");
*/
    ytry=amotry(p,y,psum,ndim,funk,ihi,nfunk,-ALPHA);
    if (ytry <= y[ilo])
      ytry=amotry(p,y,psum,ndim,funk,ihi,nfunk,GAMMA);
    else if (ytry >= y[inhi]) {
      ysave=y[ihi];
      ytry=amotry(p,y,psum,ndim,funk,ihi,nfunk,BETA);
      if (ytry >= ysave) {
        for (i=1;i<=mpts;i++) {
          if (i != ilo) {
            for (j=1;j<=ndim;j++) {
              psum[j]=0.5*(p[i][j]+p[ilo][j]);
              p[i][j]=psum[j];
            }
            y[i]=(*funk)(psum);
          }
        }
        *nfunk += ndim;
        GET_PSUM
      }
    }
  }
  free_dvector(psum,1,ndim);
}
double amotry(p,y,psum,ndim,funk,ihi,nfunk,fac)
double **p,*y,*psum,(*funk)(),fac;
int ndim,ihi,*nfunk;
{
  int j;
  double fac1,fac2,ytry,*ptry,*dvector();
  void nrerror(),free_dvector();
  ptry=dvector(1,ndim);
```

```
  fac1=(1.0-fac)/ndim;
  fac2=fac1-fac;
  for (j=1;j<=ndim;j++) ptry[j]=psum[j]*fac1-p[ihi][j]*fac2;
  ytry=(*funk)(ptry);
  ++(*nfunk);
  if (ytry < y[ihi]) {
    y[ihi]=ytry;
    for (j=1;j<=ndim;j++) {
      psum[j] += ptry[j]-p[ihi][j];
      p[ihi][j]=ptry[j];
    }
  }
  free_dvector(ptry,1,ndim);
  return ytry;
}
#undef ALPHA
#undef BETA
#undef GAMMA
#undef NMAX
```