01333C

PROGRAM DESCRIPTION I

Program Title: Non-Linear Curve Fit Using Simplex Function Minimization

Contributor's Name: Kenneth Butterfield

Address: 2020 23rd Street, Apt. C

City: Los Alamos State/Country: New Mexico Zip Code: 87544

This program is used to fit a set of experimental data to an arbitrary user supplied function. The function used can be nonlinear, and have up to three free parameters (variables). The program will prompt the user for all required data. The experimental data is stored in (x,y) pairs. Initial values (guesses), and step sizes for the free parameters are also required. The better the initial guess the quicker the program will reach a good fit.

The method used performs a least squared error fit of the theoretical curve to the experimental data. This is accomplished using the simplex function minimization of Nelder and Mead (ref 1). The function that is minimized is the square root of the sum of the errors at each data point. The simplex method is used because it is relatively fast at converging toward a minimum even when the initial guess for the free parameters is not very good. Many other data fitting methods can have problems finding the region of the minimum unless they are started close to the minimum. The main requirement for the simplex method is the storage of an N+1 by N+1 array to store the simplex, where N is the number of free parameters.

Necessary Accessories: At least two extra memory modules (with program separation).

Operating Limits and Warnings: The more memory, the better.

Reference(s): "A Simplex Method for Function Minimization"
J.A. Nelder and R. Mead
Comput. J. vol. 7, 308 (1965)

This program requires a large amount of memory for data and program storage. A large part of the memory is used to support generality. Both the number of data points, and the size of the simplex can vary in size. In fact, one third of the program does nothing more then initialize the data structure, and prompt for input. These sections of the program could be loaded and ran separately from the main program loop to allow more memory space for data storage.

Another resource the program uses freely is time. One pass through the main loop may take several seconds. In order to calculate the sum of the squared errors, the theoretical function must be evaluated at each of the data points. This must be done an average of two or three times per pass. For a function with three free parameters it may take 150-200 evaluations to reach a minimum.

Note: For many problems the program will reach the region of the minimum quickly (20 iterations), and then take many more iterations to zero in on the final result. Therefore the program is written to display the latest calculated value for the RMS error. The user can monitor this value to determine how the program is doing. If it is fluctuating rapidly from large to small values the program is searching for the region of a minimum. Once such a region is found the value displayed will more or less continually decrease until the required errror is reached. It is possible, however, to have theoretical functions that are either very slow in converging, or that can 'fool' the program into finding a 'local' minimum. For example of such functions see reference one. The user should be aware that the program can find spurious results. There is no guarantee that the best minimum has been found. One method of checking the results is to start the program with a different set of initial values for the free parameters. Approximately the same final values should be found for a 'good' minimum. Another good check is to plot the experimental and theoretical data on the same graph.

DATA STRUCTURE

A simplex is a set of N+1 points in an N dimensional space. For example, one free parameter has a simplex composed of two points. Each point is a value of the free parameter and the value of the RMS error for that parameter.

$$pi = \begin{bmatrix} Vi \\ Pi \end{bmatrix} \qquad \begin{matrix} \text{RMS error} \\ \text{parameter value} \end{matrix}$$

the simplex for one free parameter is

$$s = \begin{bmatrix} p0 \\ p1 \end{bmatrix} = \begin{bmatrix} V0 & P0 \\ V1 & P1 \end{bmatrix}$$

The geometric representation for this simplex is a line segment.

For two free parameters there will be three points, and the geometric repr[redacted]

For three free parameters the simplex is

$$s = \begin{bmatrix} p0 \\ p1 \\ p2 \\ p3 \end{bmatrix} = \begin{bmatrix} V0 & P01 & P02 & P03 \\ V1 & P11 & P12 & P13 \\ V2 & P21 & P22 & P23 \\ V3 & P31 & P32 & P33 \end{bmatrix}$$

Graphically, this is a tetrahedron in a three dimensional space.

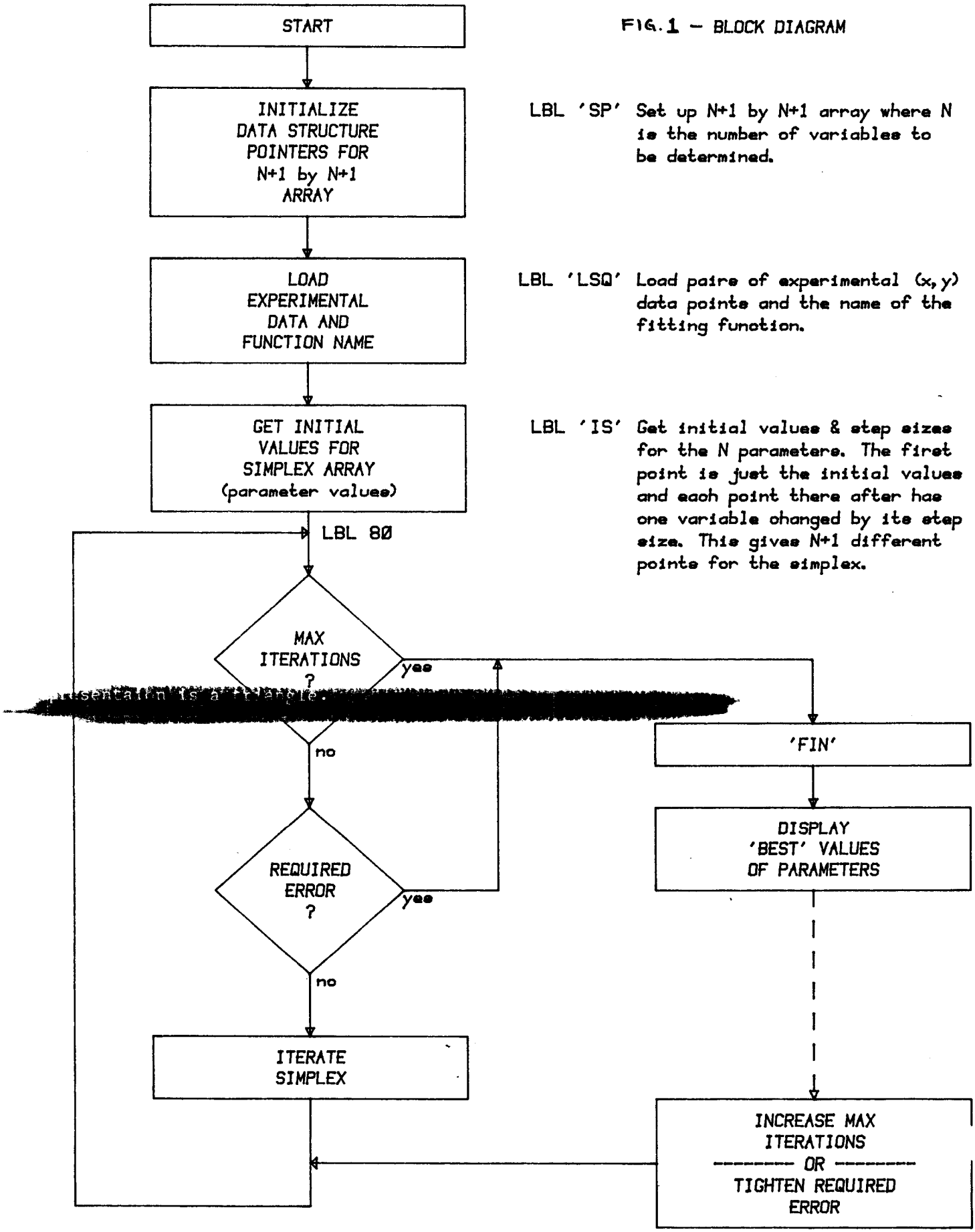


FIG. 1 – BLOCK DIAGRAM

LBL 'SP' Set up N+1 by N+1 array where N is the number of variables to be determined.

LBL 'LSQ' Load pairs of experimental (x, y) data points and the name of the fitting function.

LBL 'IS' Get initial values & step sizes for the N parameters. The first point is just the initial values and each point there after has one variable changed by its step size. This gives N+1 different points for the simplex.

PROGRAM FLOW

A block diagram for the overall program flow is given in figure one. The first step in the program is to define the data pointers used in addressing the (N+1) by (N+1) simplex, and the (x,y) data pairs of the experimental data array. This is done in the routine 'SP' (Set Pointers). The program will prompt for N, the number of free parameters. This value is used to determine the size of the simplex. 'SP' then calculates the various pointers, offsets, and constants needed by the main program. This data resides in the first 25 data registers.

The next step is to input the experimental data (routine 'LSQ'). the program will prompt for the number of data points and for each of the x and y values of the data. The present value stored will be displayed as part of the prompt. If this value is correct, pressing R/S will keep the value without having to re-enter anything. This feature makes it possible to restart 'SIMP' either to check the data or to change selected values.

The last initialization is done in routine 'IS' (Initialize Simplex). The program prompts for the initial values and the step sizes of the free parameters. This information is used to form the initial simplex and each point in the simplex is eval[redacted]

For three parameters, the initial simplex has the form

$$S = \begin{bmatrix} V0 & IV1 & IV2 & IV3 \\ V1 & IV1+SS1 & IV2 & IV3 \\ V2 & IV1 & IV2+SS2 & IV3 \\ V3 & IV1 & IV2 & IV3+SS3 \end{bmatrix}$$

The main loop of the program starts at 'LBL 80'. An expanded flow chart for the main loop is given in figure two. The main goal of the algorithm is to use the present points in the simplex to find a direction to move. A point is calculated along this direction. The point is evaluated and copied into the simplex if it is better then the worst value presently in the simplex.

For example, in two dimensions a simplex is a triangle. The program will determine the high and low points (PH,PL), and form a new point P* by reflecting PH across the center of mass (PBAR) of the remaining points. Another point P** will be calculated in this same direction. Its position will depend upon the success of P*. If P* is lower then PH, P** will be further along, otherwise P** will be stepped closer in. Which ever of P* or P** is the lowest will be compared to PH, and if it is an impprovement copied into the simplex. Should it turn out that neither P* nor P** is better then PH, the whole simplex is contracted toward PL. The main advantage of this method is that the largest possible step in an intelligent direction is taken. Thus the method moves rapidly toward better function values. The shape of the simplex adapts itself to the local terrain, enlarging while searching for the region of the minimum, and elongating when entering narrow valleys.

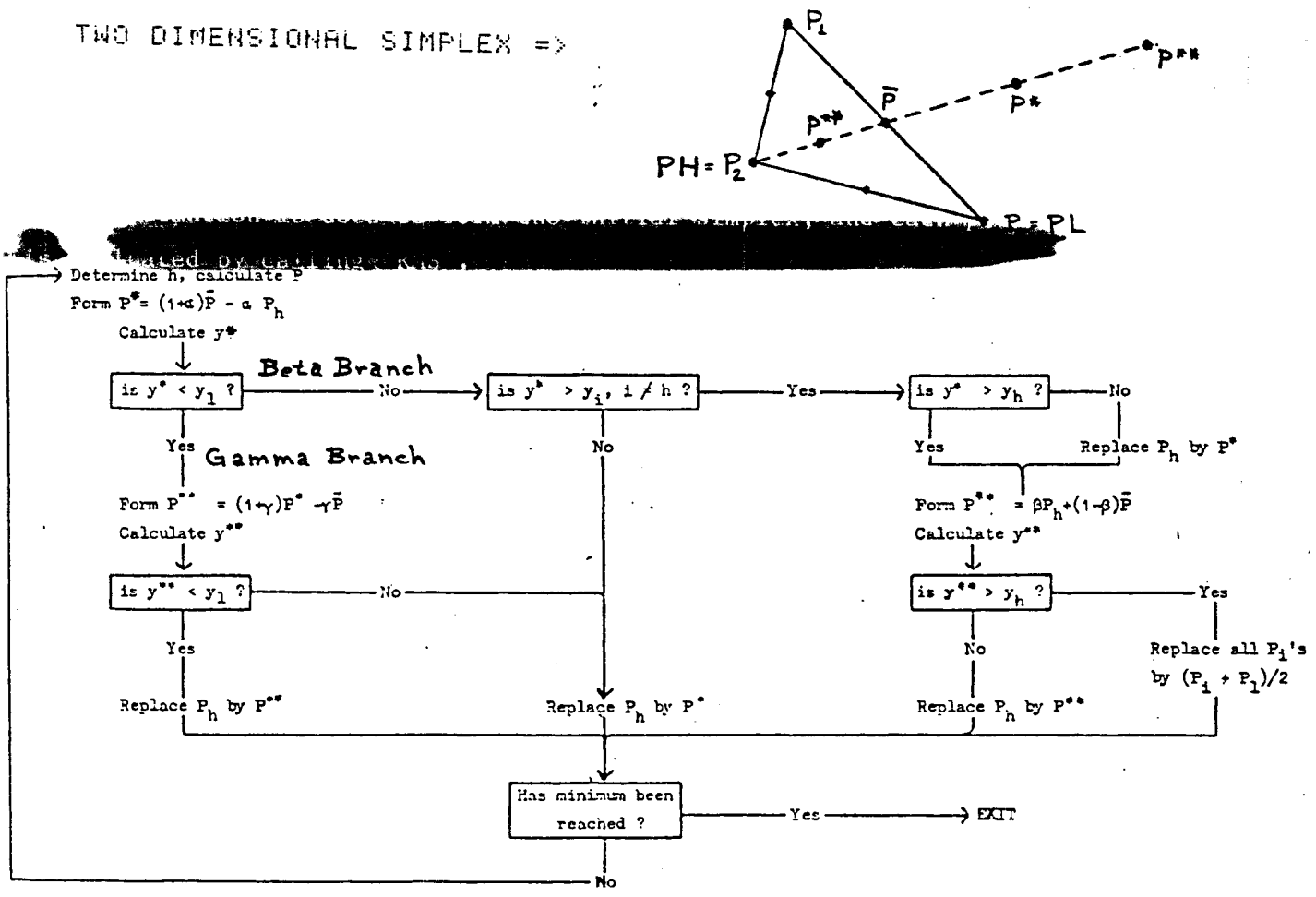


Fig.2 - Flow Diagram, Main Loop

PROGRAM DESCRIPTION II

Sample Problem (Sketch if Desired)

Fit F(x) to the data given.

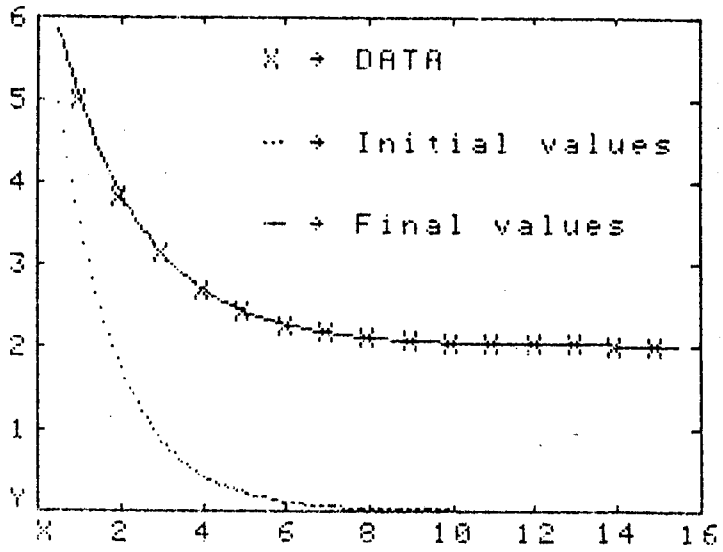
F(x)=A*exp(-B*x)+C

Initial values:

A = 7.0
B = 0.7
C = 0.0

Final values:

A = 4.91
B = 0.493
C = 2.00



SAMPLE PROBLEM

Fit the data listed below to the expression A*exp(-B*X)+C. This expression is non-linear, and can not be linearized by taking logarithms. The data could have come from an experiment, with an unknown background level. The program was r[redacted]on a[redacted]

X	Y
1	5
2	3.82
3	3.14
4	2.67
5	2.41
6	2.25
7	2.15
8	2.09
9	2.05
10	2.03
11	2.02
12	2.01
13	2.01
14	2.0
15	2.0

Initial values were

val1 = 7 step = .1 = A

val2 = .7 step = .1 = B

val3 = 0 step = 3. = C

Final values were

val1 = 4.91

val2 = .493

val3 = 2.00

Time to run was 55 minutes

It took 113 evaluations.

PROGRAM DESCRIPTION II

Step One Enter function into calculator.

on input x register contains value to evaluate

on output x register contains function value

```
REG 01 = val1 = A
REG 02 = val2 = B
REG 03 = val3 = C

LBL 'EX
RCL 02
CHS
*
E↑X
RCL 01
+
RTN
```

Step Two Run program (calculator display is underlined)

display	input	comments
N?	3 R/S	number of free parameters
NAME?	EX R/S	name of function to fit to
DATA N	15 R/S	number of data points
ERR	.001 R/S	required error (difference in value vh-vl)
MAX EVAL	100 R/S	maximum tries at fitting data before stopping
X1::0.00	1 R/S	data entry
Y1::0.00	5 R/S	
X2::0.00	2 R/S	
Y2::0.00	3.82 R/S	
X3::0.00	3 R/S	
Y3::0.00	3.14 R/S	
X4::0.00	4 R/S	
Y4::0.00	2.67 R/S	
X5::0.00	5 R/S	
Y5::0.00	2.41 R/S	

PROGRAM DESCRIPTION II

X6::0.00	6 R/S
Y6::0.00	2.25 R/S
X7::0.00	7 R/S
Y7::0.00	2.15 R/S
X8::0.00	8 R/S
Y8::0.00	2.09 R/S
X9::0.00	9 R/S
Y9::0.00	2.05 R/S
X10::0.00	10 R/S
Y10::0.00	2.03 R/S
X11::0.00	11 R/S
Y11::0.00	2.02 R/S
X12::0.00	12 R/S
Y12::0.00	2.01 R/S
X13::0.00	13 R/S
Y13::0.00	2.01 R/S
X14::0.00	14 R/S
Y14::0.00	2.00 R/S
X15::0.00	15 R/S
Y15	[redacted]
VAL1	[redacted] R/S initial guesses for free parameters
STEP	.1 R/S
VAL2	.7 R/S
STEP	.1 R/S
VAL3	0 R/S
STEP	3 R/S

*** END OF INPUT ***

If data editing is wanted stop input before entering the free parameters and restart the program. This time, the values for the data points are the values entered earlier. If the value is correct R/S will re-enter that value. If a change is needed, enter the new value and then R/S. This is a good way to verify data and doesn't add very much complexity to the program.

PROGRAM DESCRIPTION II

Step Three Improvement of result

After displaying 100 values representing the value of the RMS error at each iteration the program beeps and displays 'DONE'. It should be noted that flag 2 is set indicating that the maximum number of iterations has been reached. Entering R/S will allow viewing the best point to data.

DONE	R/S	
P0.::.02947	R/S	RMS error of fit at this point
P1.::4.90975	R/S	val1 = A
P2.::.49163	R/S	val2 = B
P3.::1.99480	R/S	val3 = C
MAX=200		the last R/S forced an improvement
		Since the program had counted out,
		more evaluations were allowed.
		The program is again searching
		for the minimum RMS error.

After 13 more evaluations the program again stops. This time flag 2 is not [redacted] ed. To view the results, press R/S.

DONE	R/S	
P0.::.02943	R/S	this has not changed much
P1.::4.91623	R/S	
P2.::.4925	R/S	
P3.::1.99484		

At this point if R/S is entered the program will divide the required error by 100 and continue searching. Since it appears that we have a good minimum it is time to stop. The total time to solve the problem was 55 minutes.

Calculator Requirements

This problem required 200 registers to run. There were 50 registers for the simplex, 2*15=30 registers for the data, and 120 registers for the program. For machines without this much memory the routines 'SP', 'LSQ', and 'IS' can be loaded into memory separately from 'SIMP'. Run each routine (in the above order), and then load 'SIMP' into the same memory space. The first few lines of 'SIMP' will need to be modified to remove the calls to the missing routines.

Program Modifications

There are several possible modifications that may improve the program in certain problems. For the sample problem the x data values were consecutive integers and didn't really need to be stored. For other problems a different type of fit may be required, such as a chi squared minimization. In another type of problem the minimum value of an analytic function may be needed. To facilitate changes of these natures I have listed the routines and registers available.

Routines

LSQ	LSQ defines the data storage structure
RMS	RMS uses the data stored by LSQ and calls the user function. It also calculates the sum of the squared errors.

Registers

24	'rms' contains name of function to minimize
25	'user name' contains name of function to fit to
5,6,7	available for counters, ect
9	start of data offset
10	data counter, used to initialize loops
21	required error
22	maximum allowed iterations
0,1,2,3	point evaluated by 'RMS'. 0 is used to return the value.

USER INSTRUCTIONS

STEP	INSTRUCTIONS
1	Adjust the number of data registers to allow for storage of the simplex, and the experimental data. See the registers and status section to determine this size.
2	Enter the program from cards, or by the light wand.
3	Enter the function to be used in the data fit. This function should get its argument from the x register, and return its value in the x register. The values for the free parameters should be obtained from registers 1, 2, and 3 (see sample problem)
4	[redacted] Run the program. It will prompt for all required data (see sample problem).
5	When the program stops it will display the RMS error, and the free parameters one at a time. At this point the option exists to try to improve the result. This is done by simply continuing the program (see sample problem).

01333C

PROGRAM LISTING

☐ 67 ☐ 97 ☐ 41C

STEP/ LINE	KEY ENTRY	KEY CODE (67/97 only)	COMMENTS
1	LBL 'SIMP'		
2	XEQ 'SP'		SET POINTERS
3	XEQ 'LSQ'		LOAD EXPERIMENTAL DATA
4	XEQ 'IS'		INITIALIZE SIMPLEX
5	LBL 80		MAIN LOOP
6	FS? 02		MAXIMUM # ITERATIONS?
7	XEQ 'FIN'		DISPLAY RESULTS
8	XEQ 20		GET HI,LO POINTS
9	FS?C 01		
10	XEQ 30		DISPLAY CURRENT VALUES
11	XEQ 21		CALC CURRENT ERROR
12	RCL 21		GET REQUIRED ERROR
13	X>Y?		END CHECK
14	XEQ 'FIN'		
15	XEQ 22		FIND PBAR
16	RCL 17		FIND P*
17	RCL 20		P*=2PBAR-PH
18	-1		
19	ENTER		
20	2		
21	XEQ 23		DO CALC.
22	0		
23	RCL 18		
24	XEQ 25		COPY P* TO PH
25	RCL IND 18		
26	RCL IND 16		
27	X>Y?		VL>V*?
28	GTO 81		GAMMA BRANCH
29	RCL 11		(V*<VI?)I<>H
30	STO 05		
31	LBL 82		
32	RCL 05		
33	INT		
34	RCL 17		
35	INT		
36	X=Y?		I=H?
37	GTO 79		THEN SKIP
38	RCL IND 18		ELSE
39	RCL IND 05		
40	X>Y?		VI>V*?
41	GTO 84		REPLACE PH BY P* AND LOOP
42	LBL 79		
43	ISG 05		CHECK ALL POINTS
44	GTO 82		
45	RCL IND 18		BETA BRANCH
46	RCL IND 17		
47	X>Y?		VH>V*?
48	XEQ 83		PH=P*
49	RCL 17		
50	RCL 20		
51	.5		
52	ENTER		
53	XEQ 23		P** = .5*(PBAR+PH)
54	RCL IND 17		
55	RCL 00		
56	X>Y?		V**>VH?
57	GTO 26		SHRINK SIMPLEX
58	LBL 85		PH=P** AND LOOP
59	0		
60	RCL 17		
61	XEQ 25		COPY P** TO PH
62	GTO 80		LOOP TO MAIN LOOP
63	LBL 84		PH=P* AND LOOP
64	XEQ 83		
65	GTO 80		
66	LBL 83		PH=P* AND RETURN
67	RCL 18		POINTER TO P*
68	RCL 17		POINTER TO PH
69	XEQ 25		COPY P* TO PH
70	RTN		
71	LBL 81		GAMMA BANCH
72	RCL 17		P**=3*PBAR-2*PH
73	RCL 20		
74	-2		
75	ENTER		
76	3		
77	XEQ 23		
78	RCL 00		
79	RCL IND		
80			V**?
81	GTO 85		PH=P**
82	GTO 84		PH=P*
83	LBL 08		FORM POINTER
84	INT		
85	RCL 12		J INIT 1
86	*		
87	RCL 13		J INIT 2
88	+		
89	RTN		
90	LBL 25		COPY POINTX = POINTY
91	XEQ 08		
92	STO 05		TO POINTER
93	RDN		
94	XEQ 08		
95	STO 06		FROM POINTER
96	LBL 99		COPY LOOP
97	RCL IND 06		
98	STO IND 05		
99	ISG 05		
100	RDN		
101	ISG 06		
102	GTO 99		LOOP BACK
103	RTN		
104	LBL 23		NEW POINT
105	STO 14		INPUT PARAMS=
106	RDN		X=A

Note: Refer to the PPC OWNER'S HANDBOOK AND PROGRAMMING GUIDE for specific information on keystrokes. The Function Index is found at the very back of the Handbook. Refer to Appendix E in the OWNER'S HANDBOOK AND PROGRAMMING GUIDE for card reader use.

STEP/ LINE	KEY ENTRY	KEY CODE (67/97 only)	COMMENTS
107	STO 15		Y=B
108	RDN		Z=POINT1
109	XEQ 08		T=POINT2
110	STO 05		
111	RDN		
112	XEQ 08		
113	STO 06		
114	RCL 13		ON OUTPUT THE
115	STO 07		NEW POINT P
116	LBL 98		P=A*P1+B*P2
117	RCL IND 05		WHERE P IS
			STORED
118	RCL 14		IN REG 1,2,3
119	*		AND THE VAL AT
120	RCL IND 06		P IS IN REG 0
121	RCL 15		
122	*		
123	+		
124	STO IND 07		
125	ISG 07		
126	RDN		
127	ISG 06		
128	RDN		
129	ISG 05		
130	GTO 98		
131	XEQ IND 24		EVALATE P
132	RTN		
133	LBL 20		FIND HI,LO
			POINTS
134	RCL 11		FOR I=P1 TO PN
135	STO 05		
136	STO 16		SET PL,PH=P1
137	RDN		
138	ISG 05		
139	LBL 95		COMPARE LOOP
140	RCL IND 05		
141	RCL IND 16		
142	X>Y?		VL>VI?
143	XEQ 97		SET PL=PI
144	RDN		
145	RCL IND 17		
146	X<=Y?		VH<=VI?
147	XEQ 96		SET PH=PI
148	ISG 05		
149	GTO 95		LOOP
150	RTN		
151	LBL 96		SET PH=PI
152	RCL 05		
153	STO 17		
154	RTN		
155	LBL 97		SET PL=PI
156	RCL 05		
157	STO 16		
158	RDN		
159	RTN		
160	LBL 22		CALCULATE PBAR
161	RCL 20		
162	STO 07		
163	0		
164	LBL 91		PBAR SET TO 0
165	STO IND 07		
166	ISG 07		
167	GTO 91		
168	RCL 11		
169	STO 05		
170	LBL 94		PBAR=AVG OF ALL
171	RCL 20		POINTS EXCEPT
172	STO 07		PH
173	RCL 17		
174	INT		
175	RCL 05		
176	INT		
177	X=Y?		
178	GTO 93		
179	RCL 12		
180	*		
181	RCL 13		
182	+		
183	STO 06		
184	LBL 92		ADD IN PI
185	RCL IND 06		
186	ST+ IND 07		
187	ISG 07		
188	RDN		
189	ISG 06		
190	GTO 92		
191	LBL 93		
192	ISG 05		
193	GTO 94		DO NEXT POINT
194	RCL 20		
195	STO 07		
196	RCL 04		
197	RCL 20		
198	ST/ IND 07		NOW DIVIDE BY N
199	ISG 07		
200	GTO 90		
201	RTN		
202	LBL 26		SHRINK SIMPLEX
203	RCL 11		
204	STO 08		
205	LBL 89		ALL POINTS
206	RCL 16		ARE AVG OF
207	INT		POINT AND PL
208	RCL 08		
209	INT		
210	X=Y?		CAN'T SHRINK PL
211	GTO 88		
212	.5		
213	ENTER		
214	XEQ 23		
215	0		
216	RCL 08		
217	XEQ 25		
218	LBL 88		
219	ISG 08		
220	GTO 89		
221	GTO 80		GO TO MAIN LOOP
222	LBL 21		ERROR
			CALCULATION
223	RCL IND 17		ERROR=VH-VL

Note: Refer to HP-41C OWNER'S HANDBOOK AND PROGRAMMING GUIDE for specific information on keystrokes. The Function Index is found at the very back of the Handbook. Refer to Appendix E in the HP-67/97 OWNER'S HANDBOOK AND PROGRAMMING GUIDE for exact keystrokes.

☐ 67 ☐ 97 ☐ 41C

STEP/ LINE	KEY ENTRY	KEY CODE (67/97 only)	COMMENTS
224	RCL IND 16		
225	-		
226	RTN		
227	LBL 40		DISPLAY ROUTINE
228	FIX 0		
229	ARCL Y		
230	'F::'		
231	FIX 5		
232	ARCL X		
233	RTN		
234	LBL 'FIN'		DISPLAY FINAL
235	BEEP		RESULTS AND
236	'DONE'		THEN ALLOW MORE
237	PROMPT		ITERATIONS
238	LBL 30		TO IMPROVE
239	RCL 16		RESULTS
240	INT		
241	RCL 12		
242	*		
243	RCL 13		
244	STO 06		
245	+		
246	STO 05		
247	LBL 31		
248	'P'		
249	RCL 06		
250	INT		
251	RCL IND 05		
252	XEQ 40		
253	PROMPT		
254	ISG 06		
255	ENTER		
256	[illegible]		SET TO
257	GTO 31		SIMPLEX
258	FS? [illegible]		CHECK TYPE OF
			END
259	GTO 32		
260	100		IMPROVE BY
261	ST/ 21		TIGHTNING
262	'R.E.='		REQUIRED
263	ARCL 21		ERROR
264	AVIEW		
265	RTN		
266	LBL 32		IMPROVE BY
267	RCL 22		ALLOWING MORE
268	INT		ITERATIONS
269	RCL 19		
270	*		
271	ST+ 22		
272	'MAX='		
273	ARCL 22		
274	AVIEW		
275	RTN		
276	LBL 'RMS'		FUNCTION
277	RCL 10		MINIMIZED
278	STO 07		BY SIMPLEX
279	0		
280	STO 00		
281	LBL 41		FIND RMS ERROR
282	RCL IND 07		OF FIT
283	XEQ IND 25		TO DATA
284	ISG 07		
285	RCL IND 07		
286	-		
287	X^2		
288	ST+ 00		
289	ISG 07		
290	GTO 41		
291	RCL 00		
292	SQRT		
293	STO 00		
294	ISG 22		
295	GTO 03		
296	SF 02		
297	LBL 03		
298	CLA		
299	RCL 22		DISPLAY VALUE
300	RCL 00		WHILE FORMING
301	XEQ 40		NEXT POINT
302	AVIEW		
303	RTN		
304	LBL 'SP'		SET POINTERS
305	.00001		
306	STO 14		
307	.001		
308	STO 13		
309	STO 19		
310	1		
311	+		
312	STO 12		JINIT 1
313	26		
314	[illegible]		SET TO
			SIMPLEX
315	*		
316	STO 11		
317	RCL 04		
318	'N?'		GET NUMBER
319	PROMPT		OF VARIABLES
320	STO 04		
321	ST* 13		JINIT 2
322	1		
323	+		
324	STO 15		
325	ST* 14		
326	X^2		
327	STO 05		
328	1		
329	-		
330	RCL 19		
331	*		
332	ST+ 11		
333	RCL 14		
334	ST+ 11		
335	RCL 05		
336	RCL 23		
337	+		
338	STO 20		
339	STO 18		
340	STO 09		
341	RCL 15		

PROGRAM LISTING

☐ 67 ☐ 97 ☐ 41C

STEP/ LINE	KEY ENTRY	KEY CODE (67/97 only)	COMMENTS
342	ST+ 20		
343	ST+ 09		
344	ST+ 09		DATA OFFSET
345	RCL 12		
346	ST* 18		P* POINTER
347	ST* 20		PBAR POINTER
348	RCL 13		
349	ST+ 18		
350	ST+ 20		
351	RTN		
352	LBL 'IS'		FORM INITIAL SIMPLEX
353	RCL 13		
354	1		
355	+		
356	STO 06		
357	LBL 61		INITIAL SIMP.
358	RCL 06		IS FORMED
359	INT		FROM INITIAL
360	RCL 11		VALUES AND
361	+		STEP SIZES
362	STO 05		
363	RCL 06		ONE SS. IS
364	RCL IND 05		ADDED TO IT'S
365	'VAL'		VALUE PER POINT
366	XEQ 40		EXCEPT P1
367	PROMPT		WHICH HAS ONLY
368	LBL 62		VALUES.
369	STO IND 05		
370	ISG 05		
371	GTO 62		
372	RCL 15		
373	RCL 06		
374	*		
375	RCL 09		
376	+		
377	RCL 06		
378	+		
379	'STEP'		GET STEP SIZE I
380	PROMPT		
381	ST+ IND Y		ADD TO PROPER
382	ISG 06		POINT IN SIMPLEX
383	GTO 61		
384	RCL 11		
385	STO 08		
386	LBL 63		EVALUATE INTIAL
387	RCL 08		SIMPLEX TO GET
388	0		VALUES VI
389	XEQ 25		
390	XEQ IND 24		
391	RCL 00		
392	STO IND 08		
393	ISG 08		
394	GTO 63		
395	RTN		
396	LBL 'LSQ'		LOAD EXPERIMENTAL DATA FOR LEAST SQUARE
397	ADV		
398	'FCN'		EVALUATION TO
399	PROMPT		'FCN'
400	ASTO 25		
401	AOFF		
402	'RMS'		
403	ASTO 24		
404	'DATA N'		NUM. OF DATA
405	PROMPT		POINTS
406	STO 00		
407	.002		
408	*		
409	STO 10		DATA INIT
410	RCL 09		
411	RCL 12		
412	*		
413	ST+ 10		
414	'ERR'		REQUIRED ERROR
415	PROMPT		
416	STO 21		
417	RCL 19		
418	ST- 10		
419	ST* 00		
420	'MAX EVAL'		MAX ALLOWED
421	PROMPT		ITERATIONS OF
422	*		MAIN LOOP
423	STO 22		
424	1.1		
425	ST+ 00		
426	RCL 10		
427	STO 05		
428	LBL 51		
429	AVIEW		GET X DATA
430	RCL 05		
431	RCL IND 05		
432	XEQ 40		SHOW PRESENT
433	PROMPT		VALUE
434	STO IND 05		
435	ISG 05		
436	'Y'		SAME FOR Y
437	RCL 00		DATA
438	RCL IND 05		
439	XEQ 40		
440	PROMPT		
441	STO IND 05		
442	ISG 00		
443	ISG 05		
444	GTO 51		
445	RTN		
446	END		

Note: Refer to HP-41C OWNER'S HANDBOOK AND PROGRAMMING GUIDE for specific information on keystrokes. The Function Index is found at the very back of the Handbook. Refer to Appendix E in the OWNER'S HANDBOOK AND PROGRAMMING GUIDE for exact keystrokes.

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

PROGRAM STORAGE

The program uses 120 registers when packed, and 123 registers when loaded by the light wand. This can be reduced by dividing the program into two parts. The first part initializes the data structure, and is then removed. Then the main program is loaded to find the fit. Aditional program space is needed to store the theoretical funtion used in the least squares data fit.

DATA STORAGE

FLAG USED	flag 01	set indicates maximum number of iterations has been reached clear indicates required error has been reached flag is valid only at program end.
	flag 02	

REGISTER	USE	STORAGE ONE DIM	TWO DIM	THREE DIM
0	[redacted]			
1	[redacted]			
2	[redacted] param 2 temporary point			
3	[redacted] param 3			
4	N number of free params.			
5	I temporary counters			
6	J			
7	K			
8	L			
9	Data offset	34	41	50
10	Data init			
11	Iinit	26.029	26.034	26.041
12	Jinit1	1.001	1.001	1.001
13	Jinit2	.001	.002	.003
14	temp A			
15	temp B			
16	PL pointer to 'best' point			
17	PH pointer to 'worst' point			
18	P* pointer	30.031	35.037	42.045
19	.001	.001	.001	.001
20	PBAR pointer	32.033	38.040	46.049
21	required error			
22	maximum iterations			
23	simplex offset	26	26	26
24	type of fit	RMS	RMS	RMS
25	function name			

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

REGISTER USE	STORAGE ONE DIM	TWO DIM	THREE DIM
		SIMPLEX STORAGE	
26	V0	V0	V0
	P0	P01	P01
	V1	P02	P02
	P1	V1	P03
	V*	P11	V1
	P*	P12	P11
	VBAR	V2	P12
33	PBAR	P21	P13
		P22	V2
		V*	P21
		P*1	P22
		P*2	P23
		VBAR	V3
		PBAR1	P31
40		PBAR2	P32
			P33
			V*
			P*1
[redacted]			
			VBAR
			PBAR1
			PBAR2
49			PBAR3
		DATA POINTS	
34 OR 41 OR 50			
	X1	X1	X1
.	Y1	Y1	Y1
.	X2	X2	X2
.	Y2	Y2	Y2
.	.	.	.
.	.	.	.
.	.	.	.
.	XM	XM	XM
.	YM	YM	YM

ROW 1 (1 : 3)

ROW 2 (3 : 7)

ROW 3 (7 : 11)

ROW 4 (11 : 15)

ROW 5 (16 : 23)

ROW 6 (23 : 29)

ROW 7 (30 : 38)

ROW 8 (38 : 44)

ROW 9 (44 : 50)

ROW 10 (51 : 57)

ROW 11 (58 : 63)

ROW 12 (64 : 69)

ROW 13 (69 : 76)

ROW 14 (77 : 82)

ROW 15 (83 : 92)

ROW 16 (93 : 99)

ROW 17 (100 : 108)

ROW 18 (109 : 116)

ROW 19 (117 : 125)

ROW 20 (126 : 133)

ROW 21 (133 : 140)

ROW 22 (141 : 147)

ROW 23 (148 : 155)

ROW 24 (155 : 164)

ROW 25 (164 : 171)

ROW 26 (171 : 180)

ROW 27 (181 : 189)

ROW 28 (189 : 194)

ROW 29 (195 : 202)

ROW 30 (202 : 211)

ROW 31 (211 : 218)

ROW 32 (218 : 223)

ROW 33 (224 : 230)

ROW 34 (230 : 234)

ROW 35 (235 : 241)

ROW 36 (242 : 251)

ROW 37 (252 : 258)

ROW 38 (258 : 262)

ROW 39 (262 : 269)

ROW 40 (270 : 276)

ROW 41 (276 : 282)

ROW 42 (282 : 289)

ROW 43 (290 : 297)

ROW 44 (298 : 304)

ROW 45 (304 : 307)

ROW 46 (308 : 317)

ROW 47 (318 : 326)

ROW 48 (327 : 336)

ROW 49 (336 : 344)

ROW 50 (344 : 352)

ROW 51 (352 : 359)

ROW 52 (360 : 366)

ROW 53 (367 : 374)

ROW 54 (375 : 381)

ROW 55 (382 : 389)

ROW 56 (389 : 396)

ROW 57 (396 : 400)

ROW 58 (400 : 404)

ROW 59 (404 : 412)

ROW 60 (413 : 418)

ROW 61 (419 : 422)

ROW 62 (423 : 429)

ROW 63 (430 : 436)

ROW 64 (437 : 443)

ROW 65 (444 : 446)