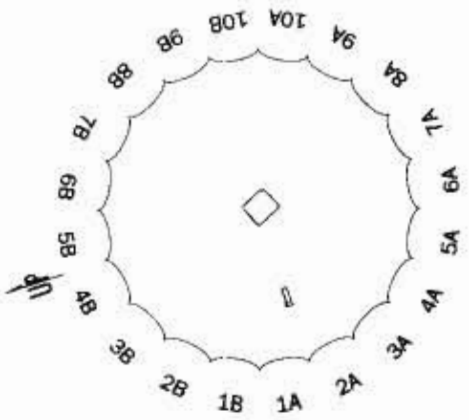
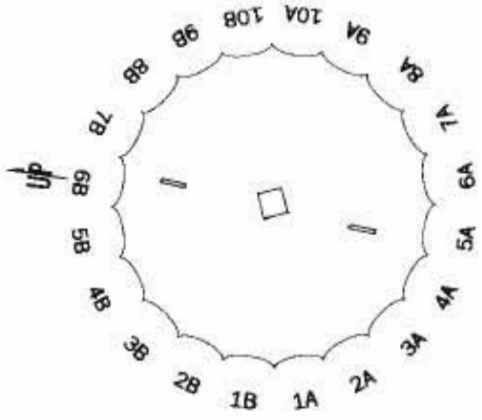
Drums E – Top Surfaces

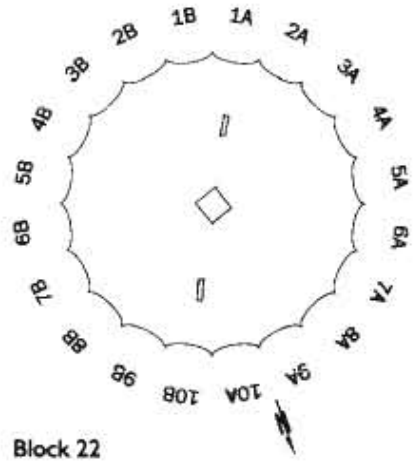


Block 497

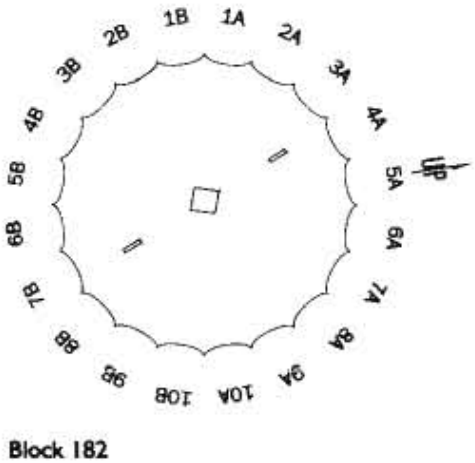


Block 533

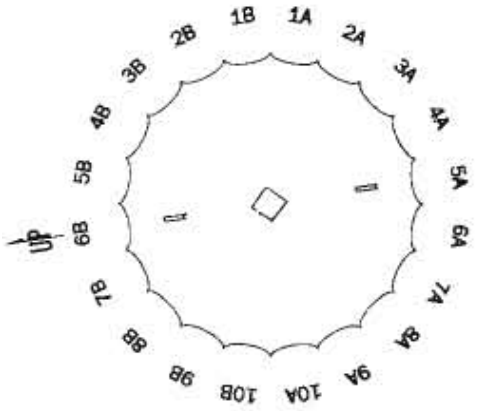
Drums F – Bottom Surfaces



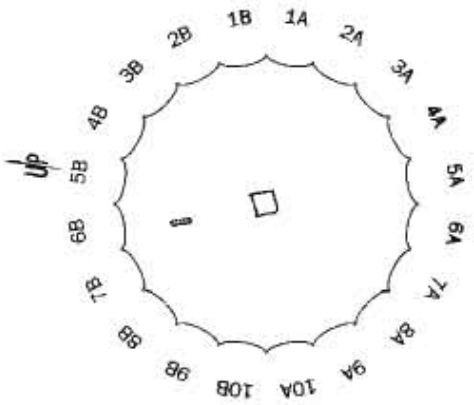
Block 22



Block 182



Block 544



Block 542

Matching Drums

Pairs of drums that could on the basis of measurements match with each other are listed in the following. The schematic drawings of empolion cuttings and dowel holes of the drum pair have been checked: the information is typographically coded in the list.

Reverse – Match according to 1:10 drawing
Bold, gray background – Possible match according to 1:25 schematic drawings
Gray background – Possible match according to 1:25 schematic drawings, but one or both surface drawings incomplete (e.g. only one dowel hole)
Bold – Close, but not exact match
Normal – No match
Italics – No drawing of lower or upper face

A:8	B:24	A:51	B:563	B:36	C:9	B:498	C:15	*C:27*	*D:90*
A:8	*B:561*	A:93	B:6	B:36	C:15	*B:498*	*C:46*	*C:27*	*D:415*
A:21	B:3	A:93	B:36	B:36	C:27	*B:498*	*C:94*	C:46	D:7
A:21	B:6	*A:93*	*B:79*	*B:36*	*C:46*	B:498	C:363	C:46	D:35
A:21	B:36	A:93	B:91	B:36	C:92	*B:498*	*C:809*	*C:46*	*D:90*
A:21	B:45	A:93	B:395	*B:36*	*C:94*	B:529	C:9	*C:46*	*D:415*
A:21	*B:79*	A:93	B:498	B:36	C:135	B:529	C:15	C:46	D:492
A:21	B:91	A:564	B:3	B:36	C:363	**B:529**	**C:27**	C:92	D:5
A:21	B:395	A:564	B:6	B:36	C:506	*B:529*	*C:46*	C:92	D:7
A:21	B:498	A:564	B:24	D:45	C:9	B:529	C:92	C:92	D:33
A:21	B:529	A:564	B:36	B:45	C:15	*B:529*	*C:94*	C:92	D:35
A:21	*B:561*	A:564	B:45	B:45	C:27	B:529	C:135	*C:92*	*D:90*
A:21	B:563	A:564	B:395	*B:45*	*C:46*	B:529	C:363	C:92	*D:415*
A:47	B:3	A:564	B:498	B:45	C:92	B:529	C:506	C:92	D:492
A:47	B:6	A:564	B:529	*B:45*	*C:94*	B:561	C:9	C:94	D:5
A:47	**B:36**	*A:564*	*B:561*	B:45	C:363	B:561	C:15	C:94	D:7
A:47	B:45	**A:564**	**B:563**	*B:45*	*C:809*	B:561	C:27	C:94	D:33
A:47	B:91			*B:79*	*C:9*	*B:561*	*C:46*	**C:94**	**D:35**
A:47	B:395	B:3	C:9	*B:79*	*C:15*	B:561	C:92	C:94	D:80
A:47	B:498	B:3	C:15	*B:79*	*C:27*	*B:561*	*C:94*	*C:94*	*D:90*
A:47	B:529	B:3	C:27	*B:79*	*C:92*	B:561	C:135	*C:94*	*D:415*
A:47	*B:561*	*B:3*	*C:46*	*B:79*	*C:94*	B:561	C:363	C:135	D:7
A:47	B:563	B:3	C:92	*B:79*	*C:135*	B:561	C:506	C:135	D:33
A:48	B:3	*B:3*	*C:94*	*B:79*	*C:363*	B:563	C:9	C:135	D:35
A:48	B:6	B:3	C:135	*B:79*	*C:506*	B:563	C:15	*C:135*	*D:90*
A:48	B:24	B:3	C:363	B:91	C:9	B:563	C:27	*C:135*	*D:415*
A:48	B:36	*B:3*	*C:809*	B:91	C:15	*B:561*	*C:46*	C:135	D:492
A:48	B:45	B:6	C:9	B:91	C:27	B:563	C:92	C:506	D:5
A:48	B:91	B:6	C:15	*B:91*	*C:46*	B:563	C:94	C:506	D:33
A:48	B:395	B:6	C:27	B:91	C:92	B:563	C:135	C:506	D:35
A:48	B:498	B:6	C:92	*B:91*	*C:94*	B:563	C:363	C:506	D:80
A:48	B:529	*B:6*	*C:94*	B:91	C:135	B:563	C:506	*C:506*	*D:90*
A:48	*B:561*	B:6	C:135	B:91	C:363			*C:506*	*D:415*
A:48	B:563	B:6	C:363	B:91	C:506	C:9	D:7	C:809	D:5
A:51	B:3	B:6	C:506	B:395	C:9	C:9	D:35	**C:809**	**D:80**
A:51	B:6	B:24	C:9	B:395	C:15	*C:9*	*D:90*	*C:809*	*D:90*
A:51	B:36	**B:24**	**C:15**	**B:395**	**C:27**	*C:9*	*D:415*		
A:51	B:45	B:24	C:27	*B:395*	*C:46*	C:9	D:492	D:5	E:20
A:51	*B:79*	*B:24*	*C:46*	B:395	C:92	C:15	D:5	D:5	E:115
A:51	B:91	B:24	C:92	*B:395*	*C:94*	C:15	D:80	D:5	E:401
A:51	B:395	*B:24*	*C:94*	B:395	C:135	*C:27*	*D:5*	D:5	E:454
A:51	B:498	B:24	C:135	B:395	C:363	*C:27*	*D:33*	D:5	E:497
A:51	B:529	B:24	C:363	B:395	C:506	*C:27*	*D:35*	D:5	E:533
A:51	*B:561*	B:24	C:506	B:498	C:9	*C:27*	*D:80*	D:7	E:20

D:7 E:115
D:7 E:401
D:7 E:454
D:7 E:497
D:33 E:88
D:33 E:401
D:33 E:533
D:35 E:20
D:35 E:115
D:35 E:401
D:35 E:454
D:35 E:497

D:80 E:20
D:80 E:115
D:80 E:401
D:80 E:454
D:80 E:497
D:80 E:533
D:90 E:88
D:90 E:115
D:90 E:401
D:90 E:497
D:90 E:533
D:415 E:88

D:415 E:115
D:415 E:401
D:415 E:497
D:415 E:533
D:492 E:20
D:492 E:115
D:492 E:454
D:492 E:497

E:20 F:22
E:20 F:89
E:20 F:182

E:20 F:507
E:20 F:544
E:88 F:22
E:88 F:89
E:88 F:542
E:88 F:544
E:115 F:22
E:115 F:89
E:115 F:182
E:115 F:507
E:401 F:22
E:401 F:89

F:401 F:542
E:401 F:544
E:454 F:22
E:454 F:89
E:454 F:182
E:454 F:544
E:497 F:22
E:497 F:89
E:497 F:542
E:497 F:544
E:533 F:542

Appendix B: Capitals

Measurements taken between preserved surfaces underlined.
For general abbreviations, see p. iii.
For abbreviations used for capitals, see also Fig. 12 (p. 33).
Trachelion height includes the height of the relieving edge.
All photographs by J.P.

C	Co-ordinates of the block
$DiamEch_{max}$	Maximum diameter of the echinus
$DiamEch_L$	Lower diameter of the echinus
$DiamAnn_L$	Lower diameter of the annulets
$Diam_A$	Diameter at the arrises
Diam	Diameter at the bottom of flutes

26. Capital. Abacus top and bottom surfaces largely preserved and partially 1 vertical abacus surface. No echinus profile. Greatest remaining abacus dimensions: c. 1.20 x c. 1.19 m. Lower surface with an empolion hole (0.13 x 0.13 m), upper with 4 dowel holes. Pres. c. 3/4.
H: 0.588.
C: On broken surface, 0.04 m S of the edge of the 45° surface. X: 39.82 Y: 0.93 Z: 0.57

Block 28.

28. Capital fragment. Something left of the surface attaching it to the column with remains of an empolion hole, 5 flutes. Full profile of the echinus, part of one side of the abacus. Pres. c. 2/5.
H: 0.589. AbH: 0.244. FlW: 0.189.
C: Empolion. X: 42.16 Y: 0.79 Z: -0.03

57. Capital fragment. Echinus and annulet profile preserved, abacus slightly on 1 side. Pres. c. 1/10.
H: 0.45.
C: On abacus at the SW side. X: 19.58 Y: 14.54 Z: -1.25

Block 86.

69. Capital fragment. Only abacus top accessible. Pry mark and dowel hole fit a capital. Surface ca. 1.30 × 0.75. Pres. c. 2/5.
H: 0.609.
C: On W dowel hole. X: 27.18 Y: 19.07 Z: -1.24

86. Capital. About half preserved, but no empolion on the bottom surface. Trachelion with 7 flutes. Pres. c. 1/2.
EchH: 0.160. AnnH: 0.047. TrachH: 0.140. FlW: 0.189–0.190 (2 flutes).
C: Highest point. X: 30.35 Y: 21.04 Z: -1.08

109. Capital. No abacus vertical profile preserved. Full height probably preserved, bottom against the ground. Pres. c. 1/2. 1.40 × 0.95 × c. 0.55 m.
C: On top of abacus, W side, 0.50 m from the N side. X: 55.91 Y: -1.46 Z: -1.15

133. Capital. Abacus fragmentarily, otherwise full profile preserved. 3 pry marks, 1 dowel hole on abacus top. Pres. c. 4/5.
H: 0.597. AbH: 0.243. EchH: 0.167. AnnH: 0.046. TrachH: 0.140. FlW: 0.187–0.188 (5 flutes). AbW: ca. 1.624. $\mathrm{DiamEch}_{max}$: 1.588. $\mathrm{DiamEch}_{L}$: 1.288. $\mathrm{DiamAnn}_{L}$: 1.234. Diam_{A}: 1.196. Diam: 1.148.
C: Highest point. X: 46.53 Y: -7.37 Z: -0.60

143. Capital fragment. Small part of the echinus profile and annulets preserved. Pres. c. 5%.
H. 0.588.
C: Highest point. X: 41.35 Y: -7.58 Z: -1.00

276. Capital. No abacus vertical surface preserved. Total profile preserved, but not measurable due to conglomerate block next to the capital. Upside down. Pres. c. 9/10.
H: 0.593. Diam_{A}: 1.206. Diam: 1.151. FlW: 0.189–0.191 (20 flutes).
C: Empolion. X: 31.28 Y: -21.88 Z: -0.48

Block 133.

Block 276.

320. Capital fragment. Corner of abacus and part of echinus preserved. Pres. dimensions of the abacus 1.12 × 0.49. Pres. c. 1/8.
H: c. 0.48.
C: SE corner. X: 31.82 Y: -11.49 Z: -0.88

Block 514.

Block 516.

340. Pronaos capital. Dugas Pl. 57. Pres. c. 4/5. FlW. c. 0.165.
C: Empolion. X: 26.73 Y: -16.43 Z: -0.66

384. Capital. Only small part of the profile with annulets preserved. Bottom with empolion. Max. Pres. dimensions c. 1.35 × 0.98. Pres. c. 3/5. H: 0.588.
C: Empolion. X: 7.19 Y: -14.76 Z: -0.72

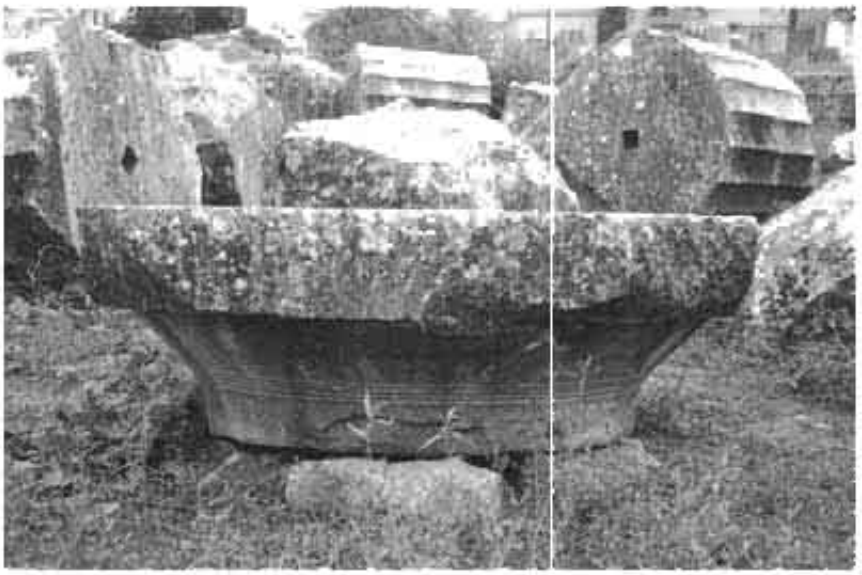
Block 520.

501. Capital. All corners of abacus broken, otherwise complete. See Fig. 13 on p. 36 for drawing (Dugas *et al.* 1924, pl. 35; measurements slightly different, the ones adopted from this plate are in *italics* in the list below). Abacus top straight, no angle for horizontal curvature adjustment. Pres. c. 1/1.
H: 0.590. AbH: 0.247 (S face, 0.246 on E and N). EchH: 0.161. AnnH: 0.046. TrachH: 0.136. FlW: 0.190.
AbW: 1.610 (NS axis, 1.615 EW). $DiamEch_{max}$: 1.590. $DiamEch_{L}$: 1.302. $DiamAnn_{L}$: 1.246. $Diam_{A}$: *1.209*. Diam: *1.158*.
C: SW corner. X: -30.77 Y: -6.40 Z: -0.82

514. Capital. Abacus vertical faces completely broken, otherwise almost complete. Empolion cutting 0.105 x 0.11. Pres. c. 4/5.
EchH: 0.159. AnnH: 0.044. TrachH: 0.139. FlW: 0188–0.191 (12 flutes).
$DiamEch_{max}$: 1.599. $DiamEch_{L}$: 1.307. $DiamAnn_{L}$: 1.253. $Diam_{A}$: 1.209. Diam: 1.155.
C: Empolion. X: -23.96 Y: -0.70 Z: -1.00

516. Capital. No abacus corners preserved. Pres. c. 1/2.
H: 0.592 (E side, 0.595 on S). AbH: 0.250 (E side, 0.246 on S). EchH: 0.159. AnnH: 0.047. TrachH: 0.136. FlW: 0.190.
C: Empolion. X: -19.25 Y: -1.76 Z: -0.83

520. Capital. Broken on 3 sides, one with full profile. 2 prymarks and 1 dowel hole. Pres. c. 1/2.
H: 0.602. AbH: 0.251. EchH: 0.165. AnnH: 0.047. TrachH: 0.139. FlW: 0.190.
C: E of the W prymark. X: -16.44 Y: -0.33 Z: -0.75



Block 539.

Block 562.

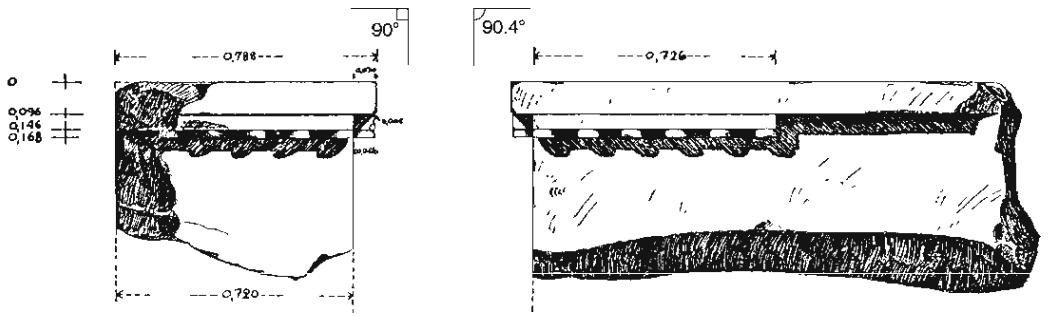
539. Capital. Almost complete. Abacus top with 3 pry marks and 2 dowel holes. Top surface straight, no angle for adjustment of horizontal curvature. Pres. c. 1/1.
H: 0.609. AbH: 0.243. EchH: 0.160. AnnH: 0.050. TrachH: 0.139. FlW: 0.189–0.191 (4 flutes).
AbW: 1.615 (NS axis, 1.609 EW). $DiamEch_{max}$: 1.599. $DiamEch_L$: 1.313. $DiamAnn_L$: 1.255.
Diam: 1.165. C: S of the S pry mark. X: -26.39 Y: 2.29 Z: -0.88

562. Capital. From the corner: band at the edge goes over corner, dowels not parallel but at a straight angle to each other. One corner of abacus largely broken, otherwise almost complete. See also Fig. 14 (Dugas *et al.* 1924, pl. 35; measurements adopted from this plate are in *italics* in the list below), profile in Fig. 12 on p. 33. Abacus top surface faces N. Pres. c. 9/10.
H: 0.590 (top, 0.589 W, 0.591 E). AbH: 0.248 (top, 0.246 W, 0.247 E). EchH: 0.158. AnnH: 0.046. TrachH: 0.138. FlW: 0.189–0.190 (2 flutes).
AbW: *1.616* (top to bottom, 1.609 EW). $DiamEch_{max}$: 1.604. $DiamEch_L$: 1.312. $DiamAnn_L$: 1.254. $Diam_A$: *1.213*. Diam: ca. *1.160*
C: SW corner of the top side of abacus. X: -14.34 Y: 13.60 Z: -0.29

Appendix C: Architrave and Frieze Blocks

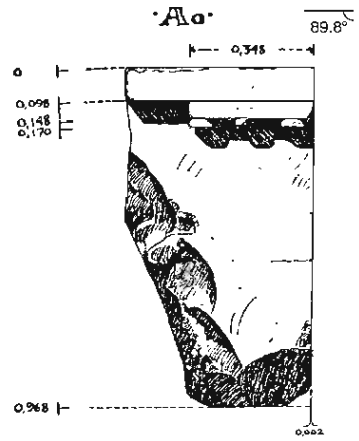
Catalogue of Architrave and Frieze Blocks Diagnostic of Horizontal Curvature

Measurements taken between preserved surfaces underlined.
For general abbreviations see p. iii.
Drawings of blocks 503 and 531 by P. Pakkanen (1995), and of blocks 1, 159, 431, 489, and 534 by M. Clemmensen (1912). Angle measurements added on these by J. P.
C Co-ordinates of the block



1. Architrave block, from corner. Dugas *et al.* 1924, pl. 38. Block adjusted for horizontal curvature: the angle between the N lateral surface below the taenia and regula and the top surface of the block is 90.4° (3 mm in 0.47 m). The other vertical face (W) is at a straight angle to the top of the block. Photographs of the angle measuring procedure on the next page.
W: 0.786. L: 1.568. Taenia H: 0.093 (at the corner), 0.096 (at 0.50 from corner).
C: Dowel hole, W-most. X: -12.84 Y: 12.07 Z: 0.13

84. Frieze block fragment. Upper part of a triglyph with a small trace of the metope. Metope taenia slightly preserved. Anathyrosis on the lateral surface. Dowel holes on the top. Angle between top and lateral surfaces 89.8° (2 mm in 0.47 m), adjusted for horizontal curvature.
H: c. 0.82. W: c. 0.86 (on triglyph). L: 0.82. Triglyph W: 0.71. Metope taenia H: 0.11.
C: On W side, 0.18 m from upper surface and 0.04 m from lateral side. X: 32.33 Y: 20.46 Z: -1.26

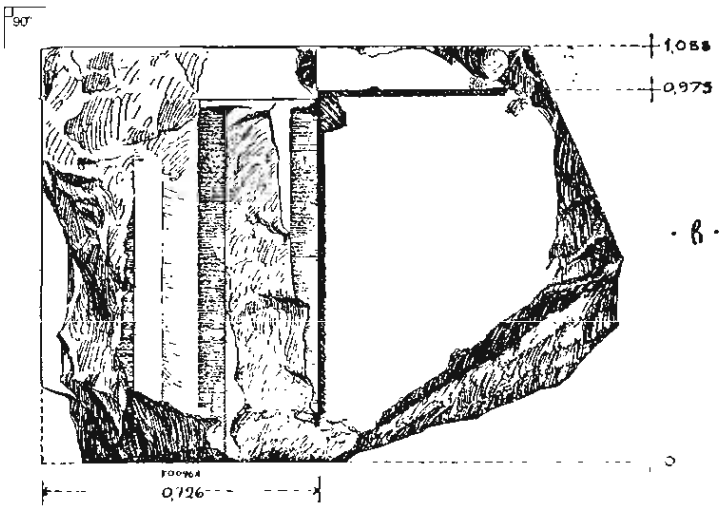


159. Architrave block. Dugas *et al.* 1924, pl. 39 A (preserved bottom surface only 0.145 m long, not 0.20 as in the drawing). Adjusted for horizontal curvature: angle between top and lateral surfaces 89.8° (3 mm in 0.715 m).
C: SW corner. X: 43.10 Y: -16.10 Z: -0.45

1. Architrave block, from corner. Measuring the angle. The line drawn on the metal square (right) enhanced. (Photographs by P. Pakkanen, 1995.)

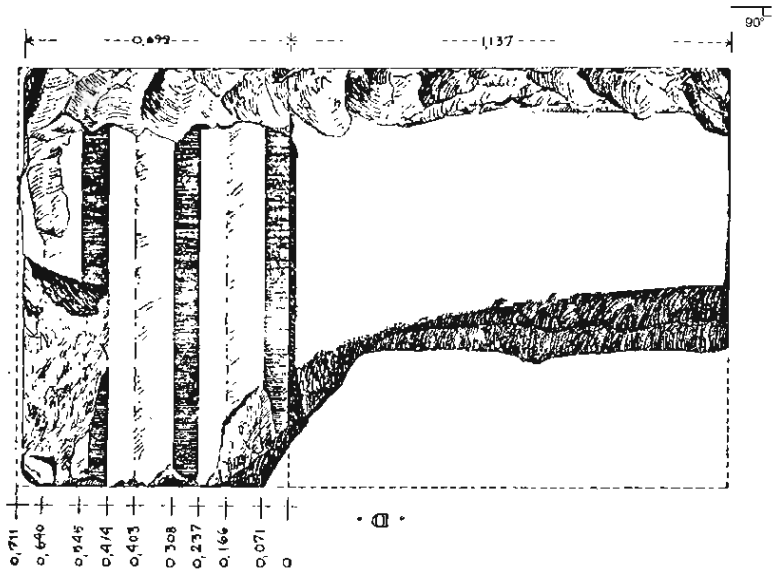
329. Architrave block. Exterior upper edge broken, not possible to determine whether inner or exterior architrave. Lateral surface with anathyrosis preserved. Top with 1 dowel hole, 1 cutting for clamp and 1 pry mark. Angle between lateral and top surfaces is 90.8° (6.5 mm in 0.47 m). Angle between bottom and lateral surfaces cannot be directly measured, but from height measurements it can be calculated as 89.4°.
H (on the front of the block): 0.969. W: 0.700. L: 1.58.
C: N end. X: 30.46 Y: -13.73 Z: -0.19

362. Frieze block. Angle between top surface and lateral triglyph face 90°.
H: c. 0.72. W: c. 0.96 (on metope). L: 1.774.
C: Highest point, 0.08 m from N end. X: 16.89 Y: -15.74 Z: -0.09

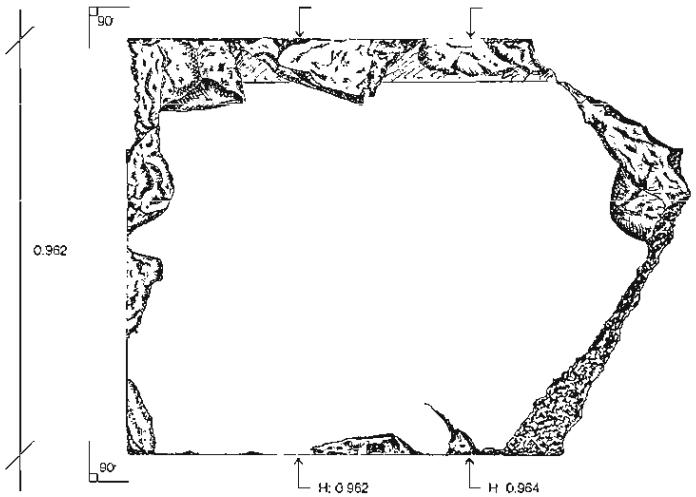


431. Frieze block from the corner. Dugas *et al.* 1924, pl. 43. Angle between the short side triglyph and top surface 90°.
C: NW corner. X: -9.68 Y: -14.24 Z: -0.26

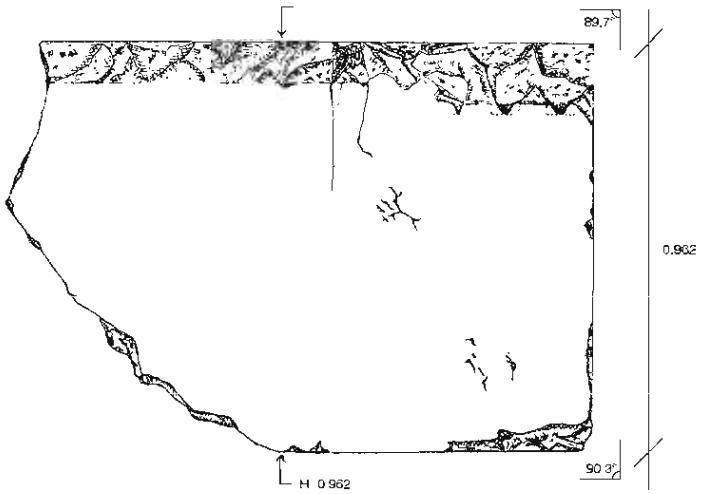
482. Inner architrave block. Top surface with 1 dowel hole, 2 cuttings for clamps, and 1 pry mark. Back and lateral surfaces with anathyrosis. Angle between the lateral anathyrosis rim and top surface 90°. Most probably matching with exterior architrave 503 (clamp cuttings, angle at the corner).
H (at the back): 0.961. W: 0.705. L: 1.23.
C: W cutting for clamp. X: -25.27 Y: -12.66 Z: -0.67



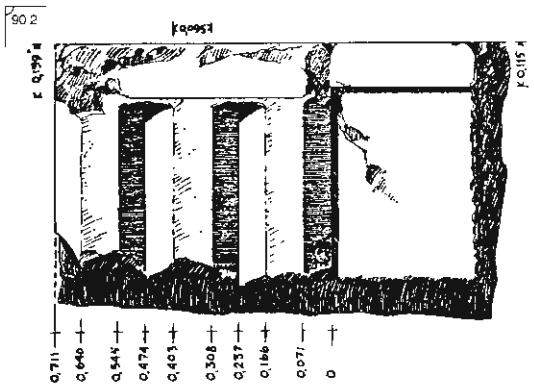
489. Frieze block. Dugas *et al.* 1924, pl. 41. The only measurable angle 90° (top corner of the metope). Top surface straight. No adjustment for horizontal curvature.
L (from metope edge to anathyrosis face): 1.815. L (from metope edge to side of the triglyph): 1.826.
C: S corner. X: -25.84 Y: -11.60 Z: -0.38



503. Architrave block. Taenia almost completely broken off. Top, front and bottom smooth, preserved lateral and back surfaces with anathyrosis. Angles between top and lateral surfaces and between lateral and bottom surfaces both 90°, but bottom surface is not straight (height of the block slightly varying). On the bottom a groove marking the edge of the abacus at 0.812-0.820 m from the end of the block (goes in 0.315 m from the face of the block, then disappears).
H: 0.962 (at 0.40 from the lateral surface of the block), 0.964 (at 0.81). W: 0.719. L: 1.32.
Taenia H: 0.090.
C: E corner. X: -28.01 Y: -4.53 Z: -0.80



531. Architrave block. Traces of 3 guttae and taenia. Top, front and bottom surfaces smooth, lateral and back surfaces with anathyrosis rim. Angle between bottom surface and lateral side 90.2° (3 mm in 0.76 m). Top surface edge broken, so the angle cannot be directly measured, but on the basis of the height measurements it is 89.8°.
H: 0.962 (right end of the block), 0.962 (at 0.72 in from the end). W: 0.720. L: 1.31.
Taenia H: 0.093.
C: SW corner. X: -19.97 Y: 3.20 Z: -0.73



534. Frieze block. Dugas *et al.* 1924, pl. 42. Angle between the frieze lateral surface and the top of the block 90.2° (2 mm in 0.470 m). Adjusted for horizontal curvature.
C: SW corner. X: -25.55 Y: 0.63 Z: -0.90

794. Frieze block fragment. Metope taenia preserved. Angle between top surface and lateral metope surface 89.7° (4 mm in 0.82 m).
H: c. 0.71. W: c. 0.89. L: c. 1.11. Metope taenia H: 0.112.
C: NW corner. X: 12.32 Y: 23.19 Z: -0.93

Appendix D: Capital and Column Measurements Used in Architectural Comparison

General abbreviations used in the appendix are listed on p. iii.
All measurements in meters.

Table D1. Capital measurements.

	CapH	AbW	$Diam_A$	AbH	EchH
Bassai, t. of Apollo (type A)	0.534	1.229	0.927	0.204	0.172
Bassai, t. of Apollo (type B)	0.534	1.180	0.900	0.204	0.172
Bassai, t. of Apollo (type C)	0.501	1.172	0.900	0.190	0.153
Argive Heraion, second t. of Hera	0.560–565	1.369	c. 1.01	0.228–234	0.169
Delphi, tholos	0.353	0.893	0.671	0.142	0.097
Epidauros, t. of Asklepios	0.304	0.811	0.606	0.122	0.083
Delphi, 4th cent. t. Apollo	0.725	1.910	1.384	0.31	0.175
Delphi, 4th cent. t. Athena	0.362	0.967	0.725	0.143	0.095
Epidauros, tholos	0.38	1.02	0.75–0.77	0.159	0.105
Megalopolis, Thersilion	0.385	1.05	0.80	0.16	0.10
Nemea, t. of Zeus	0.624	1.76	1.307	0.250	0.166
Stratos, t. of Zeus	0.505	1.36	1.00	0.202	0.136
Olympia, Metroon	0.345	0.890	0.65	0.14	0.096

Table D2. Column measurements.

	ColH	ShaftH	$Diam_{LA}$	$Diam_L$	$Diam_{UA}$	$Diam_U$	FlW_L	FlW_U	Ent_{max}	EntH
A.	5.959	5.425–58	1.112	1.041	0.900	0.853	0.173	0.140	–	–
B.	7.10–43	6.60–92	c. 1.308	1.226	c. 1.011	0.965	0.205	0.158		
C.	5.93	5.58	0.868	0.812	0.671	0.642	0.136	0.105	0.005	3.0
D.	9.35	8.99	1.716		1.384	1.286	0.268	0.217		
E.	5.282	4.92	0.893	0.818	0.725	0.669	0.140	0.113	0.004	2.6
F.	6.9 / 7.5	6.5 / 7.1	0.998	0.944	0.772 / 0.750	0.740 / 0.718	0.156	0.121 / 0.117	0.01	3.4
G.	9.544–80	8.952–77	1.55	1.46	1.21	1.15	0.24	0.19	0.011	4.3–4.7
H.	3.86	3.64	0.556		0.458		0.087	0.72	0.008	2.0
I.	10.33	9.70	1.63	1.52	1.307	1.245	0.255	0.204	0.013	4.6
J.	7.9?	7.4?	1.31	1.22	1.00	0.96	0.205	0.156		

A. Bassai, t. of Apollo (not frontal)
B. Argive Heraion, Second t. of Hera
C. Delphi, tholos
D. Delphi, 4th cent. t. Apollo
E. Delphi, 4th cent. t. Athena
F. Epidauros, tholos (11/12 drums)
G. Tegea, t. of Athena Alea
H. Delphi, treas. of Kyrene
I. Nemea, t. of Zeus
J. Stratos, t. of Zeus

Table D3. Sources of measurements.

	Sources
Bassai, t. of Apollo	Cooper 1992, pls. 20 and 40. Cooper 1996, 184, 229–230.
Argive Heraion, 2nd t. of Hera	Pfaff 1992, 123–125, 130–131.; see also n. 51 on p. 73.
Delphi, tholos	Charbonneaux—Gottlob 1925, 4–5, pl. 4; Amandry—Bousquet 1940–41, 125 n. 2 (ColH, $Diam_{LA}$). FlWs calculated. For entasis, see Pakkanen 1997, 324–326.
Epidauros, t. of Asklepios	Roux 1961, 93 and 410–411. Only CapH given, rest calculated from tables on pp. 410–411 and checked by measuring from fig. 16.
Delphi, 4th cent. t. Apollo	Courby 1927, 17, figs. 11, 16, 17. AbH and EchH measured from fig. 17 and checked from Coulton 1979, tables 18 and 19. On ColH, DiamLA, and entasis, see Ducoux 1940–41, 267.
Delphi, 4th cent. t. Athena	Michaud 1977, 31–36. FlWs calculated. For entasis, see Pakkanen 1997, 326.
Epidauros, tholos	Roux 1961, 138–140 and 410–411. AbW, AbH, and EchH calculated from tables on pp. 410–411 and checked by measuring from fig. 16. For ColH, ShaftH, $Diam_{UA}$, and entasis, see Pakkanen 1997, 327–329. FlWs calculated.
Tegea, t. of Athena Alea	New dimensions.
Delphi, treasury of Kyrene	Bousquet 1952, 46–48; For entasis, see Pakkanen 1997, 332–334.
Megalopolis, Thersilion	Gardner *et al.* 1892, fig. 18. AbW and AbH given, others measured from fig. 18; CapH and EchH checked from Coulton 1979, tables 16 and 19.
Nemea, t. of Zeus	Hill 1966, 9–10, pls. 13 and 27. EchH measured from pl. 27. For entasis of the pronaos column, see Pakkanen 1997, 334–336.
Stratos, t. of Zeus	Courby—Picard 1924, 25–29. FlWs calculated.
Olympia, Metroon	Adler *et al.* 1892, 37, pl. 26. CapH measured from pl. 26 and checked from Coulton 1979, table 16.

Appendix E: Computer Programs

The programs used in the analyses have been written especially for the purposes of shaft analysis, and they have been implemented on top of MS-DOS program Survo 84C.[1] Survo is an open system which provides very good tools for graphics, report generating, statistical analysis, and database management, and it also supports extensions made by the user. Both sucros (Survo or super macros) and additional modules written in C language have been used.[2] The output of the programs is stored in Survo data files.[3] In the following, a short description of the programs is given, and full program listings can be found on pp. E4–30.

1. Computer-intensive Statistics

A. Bootstrap-*t* method

The sucro program `bootstrap-t_fp.tut` is used in Section V.2.B. (pp. 53–54) to calculate the 95% bootstrap-*t* confidence interval on the basis of the preserved column drums at Tegea. The sucro code is listed on pp. E4–E5, and an example how the program is used in connection with the drums at Tegea is given on lines 106–125 of p. E5. The parameters of the program (line 115) must include the name of the data list in the edit field (X, on lines 107–113), the name of the data file for output (BT001.SVO), the number of *t*-values produced (5000), and the size of the population, or, in this case, the number of column drums originally in the building (216). The results are printed on lines 117–125.

B. Monte Carlo Test for Evaluating Bootstrap Method

The sucro program `strapeva.tut` is used in Section V.2.C. (pp. 54–55) to test the validity of bootstrap-*t* method. The sucro code is listed on pages E6–7, and an example how the program is used to simulate the temple colonnade at Tegea and to test the accuracy of bootstrap confidence intervals is given on lines 88–123 of p. E7. The program uses C module `!simul.exe` (lines 38–41; see also pp. E2 and E10–22), and the parameters listed on lines 91–112 are needed by the module, even though `strapeva.tut` uses only the drum height data listed on lines 114–121 (Position 0 corresponds to *A* drums, 1 to *B* drums, etc.) in the simulation. The parameters of the program line (line 123) must include the name of the data in the edit field (DRUMS, on lines 114–121), the number of repetitions for each height (8), the starting height of the simulation (8.76 m), the distance between the heights (0.02 m), the maximum shaft height (8.98 m), and the name of the data file for output (STRAPEVA.SVO). The results stored in the output file (see lines 23–30) are the lower limit of the confidence interval (*CImin,* in meters), the upper limit of the confidence interval (*CImax*), shaft height used in the simulation (*ColH*), the value of the lower bootstrap *t*-value (*t1*), the value of the higher bootstrap *t*-value (*t2*), and the mean and standard deviation of the simulated sample (*mean* and *std*).

---

[1] I have compiled the two C modules listed in App. E for Survo 84C, and they are not currently compatible with the new 32-bit version of the program, Survo 98.

[2] On sucros see Mustonen 1992, 399–443, and on programming Survo in C see Mustonen 1989.

[3] On data files, see Mustonen 1992, 75–130.

C. Non-random Data

The sucro program `simuhght.tut` is used in Section V.2.D. (pp. 55–56) to simulate the effect of non-randomness of the column drum data on the shaft height distribution. The sucro code is listed on pp. E8–9, and an example is given on lines 93–128 of p. E9. The parameters listed on lines 96–117 are used by the C module `!simul.exe`. The degree of randomness of the data is simulated by changing the amount of columns used by the module. The number of columns is given on lines 102–103: six columns on the front and eight on the sides corresponds to a total of 24 columns ($2 \times 6 + 2 \times 6 = 24$). The drum height data is listed on lines 119–126. The parameters of the program line (line 128) must include the name of the data in the edit field (DRUMS, on lines 114–121), the number of repetitions for each height (24), the starting height of the simulation (8.76 m), the distance between the heights (0.02 m), the maximum shaft height (8.98 m), and the name of the data file for output (SIMUHT24.SVO). The results stored in the output file (see lines 24–32) are the number of the simulation (*Nro*), the mean of the simulated sample (*Height*), the lower limit of the confidence interval (*CIMin,* in meters), the upper limit of the confidence interval (*CIMax*), shaft height used in the simulation (*OrHght*), difference between the simulated shaft height and the original (*HDiff* = *Height* – *OrHght*), the confidence interval width (*CIW*), and whether the original shaft height is within the simulated confidence interval or not (*OK*).

2. Shaft Profile

A. Colonnade Simulation

The C module `!simul.exe` can be used to simulate construction of a colonnade, the process of its destruction and its reconstruction by a scholar (see pp. 54–55). The example on pp. E23–24 presents how the program can be used to build a file of the possible shaft combinations based on the preserved drums at Tegea (see Section V.3.A, p. 62). It is also used by the two programs described above in Sections 1.B and 1.C. The program code is printed on pp. E10–22.

Then most important programmer defined functions are listed on lines 696–917: they map the possible shaft combinations. The function `first_path_all()` takes the first bottom drum and looks for a matching second level drum: the diameter ranges of the two drums have to be overlapping. When this is found, the drum data is recorded, and a search for a next level drum is started. This pattern is repeated until a matching top level drum is found. If a dead-end is reached before the top drums, the program goes back to the next lower level drum and starts the search again. All the complete possible column combinations are recorded into a text file: the program keeps track of the individual drum heights and margins, and besides the total height also the height margins are recorded. After this the program returns to other top level drums and tries to look for a new match, and when all top drums have been mapped and the data of the new combinations written into the text file, the program goes back to the level below etc. until all the possible shaft combinations with this particular bottom drum have been found. Then the procedure is repeated for the next bottom drum until all the conceivable ways to combine the drums have been discovered.

In the example the parameters input to the program are listed and explained on lines 2–23 on p. E23. The Tegea drum data is listed on lines 27–75, and the data file TEGEADR.SVO is created from the text file produced by the program on lines 77–137. The data file includes the *x* and *y* co-ordinates of the shaft profile as well as the measurement margins.

B. Acceptable Shaft Profiles and Maximum Entasis

The sucro program `shaft-maxent.tut` is used in Section V.3.B. (pp. 62–66) to determine the number of acceptable shaft combinations within measurement accuracy. The code of

the sucro and programs called by it are listed on pp. E25–30. The example given on lines 361–364 (p. E30) is used to produce the data in Table 9 on p. 65. The parameters of the program line (line 364) must include the name of the data file with co-ordinate data (TEGEADR2.SVO—the data file includes the shafts within the height range 8.952–8.977 from data file tegeadr.svo; on the latter file, see p. E2), the identification number the first used record (1), the identification number the last record (1,678), the minimum amount of maximum entasis (0.009 m), the maximum amount of maximum entasis (0.013 m), the minimum proportional height of the entasis (0.40), the maximum proportional height of the entasis (0.60), and the measurement accuracy (0.0015 m). The results stored are in data file SHAFTFIT.SVO (see lines 28–32 on p. E25): it includes the height of maximum entasis (*EntH*), the amount of maximum entasis (*MaxEnt*), and the number of shaft combinations in each category (*N*). It is the responsibility of the user to save the data file under a new name before reactivating the sucro `shaft-maxent.tut`. The other programs used in the analysis, `shaft-curve.tut` and `!lsqmat.exe`, are listed on lines 61–358 (the programs are nested so that `shaft-maxent.tut` calls the sucro `shaft-curve.tut` which in turn accesses the module `!lsqmat.exe`).

1.A. Bootstrap-*t* method

```
 31  | SURVO 84C EDITOR Sat Dec 12 22:33:07 1998        D:\COMMON\ 140 100 C
  1 *BOOTSTRP-T_FP.TUT
  2 *
  3 *tutsave bootstrp-t_fp
  4 / Sucro bootstrp-t_fp.tut by Jari Pakkanen (Nov 16 1998)
  5 / for calculating confidence interval of a finite population
  6 / using bootstrap-t method.
  7 / def Wdata=W1 Wfile=W2 Wrep=W3 WN=W4 Wi=W5 Wlin=W6 Wcol=W7 Wmmean=W8
  8 / def Wsstd=W9 Wn=W10 Wt025=W11 Wt975=W12 Wlim1=W13 Wlim2=W14
  9 *{tempo -1}{init}{save cursor Wlin,Wcol}
 10 - if Wdata '<>' ? then goto A
 11 *{line start}{d}{erase}{erase}DATA X: 1.465 1.469 1.472 1.473 1.474 1.
 12 *466 1.470 1.474 1.472 1.464{R}
 13 *{erase}{erase}1.472 1.465 1.469 1.472 1.473 1.474 1.466 1.470 1.474 1
 14 *.472 1.464 END{R}
 15 *{R}
 16 *{erase}{erase}Activating sucro{R}
 17 *{erase}{erase}/BOOTSTRP-T_FP <data>,<file>,<repetitions>,<pop. size>{R}
 18 *{erase}{erase}calculates bootstrap-t values from edit field data <dat
 19 *a> given in the{R}
 20 *{erase}{erase}same format as data set X above. T-values are stored in
 21 * file <file>.{R}
 22 *{erase}{erase}{goto End}
 23 /
 24 + A: {Wi=0}{R}
 25 *■SCRATCH /{act}{home}STAT {print Wdata},CUR+1{act}{ins}{search}N>{R}
 26 *{r2} {save word Wn}{search}mean={R}
 27 *{r5} {save word Wmmean}{search}stddev={R}
 28 *{r7} {save word Wsstd}{ins}{jump Wlin,Wlin,1,1}{R}
 29 *■SCRATCH /{act}{home}FILE CREATE HELPFL04{R}
 30 *FIELDS:{R}
 31 *   1 NA_    8 T         (#####.####){R}
 32 *END{R}
 33 *{u4}{act}
 34 / Starting loop.
 35 + Loop: {Wi=Wi+1}{jump Wlin,Wlin,1,1}{R}
 36 *■SCRATCH /{act}{home}FILE CREATE HELPFL01{R}
 37 *FIELDS:{R}
 38 *   1 NA_    4 N         (####){R}
 39 *   2 NA_    8 {print Wdata}         (####.###){R}
 40 *END{R}
 41 *{u5}{act}{line end}{1}7{act}{home}■SCRATCH /{act}{home}FILE COPY ()
 42 *{print Wdata},HELPFL01{act}{R}
 43 *VAR N=ORDER TO HELPFL01{act}{R}
 44 *FILE INIT HELPFL02,{print Wn}{act}{R}
 45 *INSEED=SEED OUTSEED=SEED{R}
 46 *VAR N=int({print Wn}*rand(1))+1 TO HELPFL02{act}{R}
 47 *.{copy}{R}
 48 *{R}
 49 *FILE SORT HELPFL02 BY N TO HELPFL03{act}{R}
 50 *.{copy}{R}
 51 *{R}
 52 *VARS={print Wdata} MATCH=N MODE=2{R}
 53 *FILE COPY HELPFL01,HELPFL03{act}{R}
 54 *.{copy}{R}
 55 *{R}
 56 *VAR {print Wdata}=if({print Wdata}=MISSING}then{{print Wdata}[-1]}els
 57 *e{{print Wdata}} TO HELPFL03{act}{R}
 58 *.{copy}{R}
 59 *{R}
 60 *STAT HELPFL03,CUR+10 / VARS={print Wdata}{act}{R}
 61 *.{copy}{R}
 62 *{R}
 63 *{R}
 64 *DATA TDATAXXX{R}
 65 *T{R}
 66 *{R}
 67 *{R}
```

```
 31  1 SURVO 84C EDITOR Sat Dec 12 22:39:39 1998          D:\COLMON\ 140 100 C
 68 *FILE COPY TDATAXXX,HELPFL04{R}
 69 *{R}
 70 *T={mean-{print Wsmean}}/{stddev/sqrt({print Wn})}{R}
 71 *{u5}T={act}{home}{del2}{R}
 72 *{R}
 73 *{act}
 74 - if Wi < Wrep then goto Loop
 75 *{jump Wlin,Wlin,1,1}{R}
 76 *■SCRATCH /{act}{home}FILE SORT HELPFL04 BY T TO {print Wfile}{act}{R}
 77 *VAR Nro=ORDER TO {print Wfile}{act}{R}
 78 *int({print Wrep}*0.025)={act}{save line Wt975}{R}
 79 - if Wt975 > 0 then goto Cont
 80 *{Wt975=1}
 81 + Cont: int({print Wrep}*0.975)+1={act}{save line Wt025}{R}
 82 *IND=Nro,{print Wt975} VARS=T{R}
 83 *FILE LOAD {print Wfile}{act}{R}
 84 *{d2}{next word}{save word Wlim1}{R}
 85 *.{copy}{R}
 86 *{R}
 87 *IND=Nro,{print Wt025} VARS=T{R}
 88 *FILE LOAD {print Wfile}{act}{R}
 89 *{d2}{next word}{save word Wlim2}{R}
 90 *{jump Wlin,Wlin,1,1}{R}
 91 *■SCRATCH /{act}{R}
 92 *Bootstrap t-values   t1={print Wlim1}  {obs. {print Wt975} in {}
 93 *{print Wfile}.svo}{R}
 94 *                     t2={print Wlim2}  {obs  {print Wt025} in {}
 95 *{print Wfile}.svo}{R}
 96 *Sample statistics:   n={print Wn}  N={print WN}  mean={print Wsmean}
 97 *{R}
 98 *                     stddev={print Wsstd}{R}
 99 *Estimated standard error: E=stddev/sqrt(n)*sqrt(1-n/N){R}
100 *Lower limit of the Confidence interval:{R}
101 *                     mean-t2*E={act}{R}
102 *Upper limit of the Confidence interval:{R}
103 *                     mean-t1*E={act}{R}
104 + End: {jump Wlin,Wlin,1,1}{tempo +1}{end}
105 *
106 *EXAMPLE:
107 *DATA X:
108 *1.465 1.469 1.472 1.473 1.474 1.466 1.470 1.474 1.472 1.484 1.472 1.481
109 *1.476 1.478 1.482 1.469 1.474 1.484 1.473 1.477 1.478 1.321 1.399 1.413
110 *1.444 1.457 1.479 1.498 1.500 1.510 1.561 1.643 1.668 1.415 1.447 1.448
111 *1.480 1.493 1.511 1.514 1.658 1.708 1.347 1.356 1.368 1.392 1.398 1.411
112 *1.438 1.515 1.522 1.580 1.662 1.320 1.331 1.349 1.479 1.484 1.500 1.631
113 *END
114 *
115 */BOOTSTRP-T_FP X,BT001,5000,216
116 *
117 *Bootstrap t-values:  t1=-2.1863  (obs  125 in BT001.svo)
118 *                     t2=1.8455  (obs. 4876 in BT001.svo)
119 *Sample statistics.   n=60  N=216  mean=1.476383
120 *                     stddev=0.082748
121 *Estimated standard error: E=stddev/sqrt(n)*sqrt(1-n/N)
122 *Lower limit of the Confidence interval:
123 *                     mean-t2*E=1.4596285045566
124 *Upper limit of the Confidence interval:
125 *                     mean-t1*E=1.4962314710852
```

1.B. Monte Carlo Test for Evaluating Bootstrap Method

```
 48  1 SURVO 84C EDITOR Sat Dec 12 23:05:49 1998          D:\COLMON\ 140 100 C
  1 *STRAPEVA.TUT
  2 *
  3 *TUTLOAD STRAPEVA
  4 / Sucro strapeva.tut by Jari Pakkanen (Nov 18 1998)
  5 / for evaluating the accuracy of bootstrap confidence intervals of
  6 / column shafts
  7 *{tempo -1}{init}
  8 - if W1 '<>' ? then goto A
  9 *{line start}{d}{erase}{erase}Activating sucro{R}
 10 *{erase}{erase}/STRAPEVA <data>,<reps>,<minh>,<step>,<maxh>,<file>{R}
 11 *{erase}{erase}simulates temple colonnades and tests the accuracy of b
 12 *ootstrap CIs.{R}
 13 *{erase}{erase}and stores the results in data file <file>.svo.{R}
 14 *{erase}{erase}Specifications must be given as in the example edit fie
 15 *ld {goto End}
 16 / def Wdata=W1 Wrep=W2 Wmin=W3 Wstep=W4 Wmax=W5 Wfile=W6
 17 / def Whelp1=W7 Wlin2=W8 Wcol2=W9 Wi=W10 Wj=W11 Wk=W12
 18 / def Wfline=W13 Wlline=W14 Wpresdr=W15 WCImin=W16 WCImax=W17
 19 / def Wmeanh=W18 Wlin=W19 Wcol=W20 Wcoldl=W21 Wcoldu=W22 Wcolh=W23
 20 / def Wlow=W24 Whigh=W25
 21 + A: {save cursor Wlin,Wcol}{Wj=Wmin}{R}
 22 *■SCRATCH /{act}{home}FILE CREATE {print Wfile}{R}
 23 *FIELDS:{R}
 24 *    1 NA_    4 CImin      (## ###){R}
 25 *    2 NA_    4 CImax      (## ###){R}
 26 *    3 NA_    4 ColH       (##.###){R}
 27 *    4 NA_    4 t1         (## ###){R}
 28 *    5 NA_    4 t2         (## ###){R}
 29 *    6 NA_    4 mean       (##.###){R}
 30 *    7 NA_    4 std        (##.###){R}
 31 *END{R}
 32 *{u10}{act}@SCRATCH /{act}
 33 + MainLoop: {jump 1,1,1,1}{search}ColH={R}
 34 *{find =}{r}{print Wj}    {Wi=0}
 35 /
 36 / Starting loop:
 37 + Loop: {Wi=Wi+1}
 38 / Calling SIMUL.EXE
 39 *{jump 1,1.1,1}{search}Mode={R}
 40 *{find =}{r}2{jump Wlin,Wlin,1,1}{d}@SCRATCH /{act}{home}SIMUL {}
 41 *{write Wdata},CUR+3.0,70,-100{act}{jump Wlin,Wlin,1,1}{d4}{erase}Col {}
 42 *Dr    DiamL    MaN1  MaP1    DiamU    MaN2  MaP2  Height    MaN3  MaP3{R}
 43 *{save cursor Wlin2,Wcol2}{Wfline=Wlin2}{u}{pre}{d}{pre}{d}{u}
 44 *{save cursor Wlin2,Wcol2}{Wlline=Wlin2}{jump 1,1.1,1}{search}PresDr=
 45 *{R}
 46 *{find =}{r}{ins} {ins}{save word Wpresdr}{1}{del}
 47 /
 48 / Selecting the preserved drums:
 49 *{Wk=0}
 50 + Preserved: {jump Wlin,Wlin,1,1}{d2}{erase}int({print Wfline}+(
 51 *{print Wlline}+1-{print Wfline})*rnd(0))={act}{save line Wlin2}
 52 *{jump Wlin2,Wlin2,1,1}{save char Whelp1}
 53 - if Whelp1 '=' * then goto Preserved
 54 **{Wk=Wk+1}
 55 - if Wk < Wpresdr then goto Preserved
 56 *{jump Wlin,Wlin,1,1}{d5}
 57 + Del: {save char Whelp1}{next word}{save cursor Wlin2,Wcol2}
 58 *{line start}
 59 - if Wcol2 = 1 then goto Cont
 60 - if Whelp1 '=' * then goto Del2
 61 *{del line}{goto Del}
 62 + Del2: {d}{goto Del}
 63 + Cont: {jump Wlin,Wlin,1,1}{d2}{erase}{d}DATA X:{R}
 64 *{erase}DELETE 50{home}{act}{r7}{act}{del line}{home}{u}{pre}{d}{pre}
 65 *{d}END{R}
 66 *.{copy}{R}
 67 *{R}
```

```
 48  1 SURVO 84C EDITOR Sat Dec 12 23:06:30 1998          D:\COLMON\ 140 100 C
 68 *DATA DR(R)
 69 *CImin CImax ColN t1 t2 mean std(R)
 70 *(R)
 71 *(R)
 72 *(save stack helpstck)/BTSTRP_P X,BTSTRE01,1000,216(tempo +1)(act)
 73 *(tempo -1)(load stack helpstck)(d8)(find =)(r)(save line WCImin)(R)
 74 *(d)(find =)(r)(save line WCImax)(R)
 75 *(ul3)(print WCImin) (print WCImax) (print Wj) t1=(act)(l3)(del3)
 76 *(line end) t2=(act)(l3)(del3)(line end) mean=(act)(l5)(del5)
 77 *(line end) stddev=(act)(l7)(del7)(R)
 78 *SCRATCH /(act)(R)
 79 *SAVEP C:\E\RESULTS(act)(home)(erase)(d)FILE COPY DR.(print Wfile).SVO
 80 *(act)
 81 /
 82 - if Wi < Wmp then goto Loop
 83 *(Wj=Wj+Wstep)
 84 - if Wj <= Wmax then goto MainLoop
 85 ( End: (jump Wlin,Wlin,1,1)(tempo +1)(end)
 86 *
 87 *
 88 *EXAMPLE.
 89 *All dimensions in meters (Parameters needed for SIMUL EXE. only
 90 *                          height data used by STRAPEVA.TUT.)
 91 *ColDiamL=1.455       Lower diameter of the column between flutes
 92 *DiamVar=0.005        Range of lower diameters (plus and minus)
 93 *ColDiamU=1.15        Upper diameter of the column between flutes
 94 *ColH=8.92           Column height
 95 *MaxEnt=0.01         Maximum entasis
 96 *MaxEntH=4.50        Height where the maximum entasis is
 97 *ColNFr=6            Number of columns on front
 98 *ColNS=14            Number of columns on side
 99 *DrN=6               Number of drums in one column
100 *PresDr=60           Number of preserved drums
101 *MinMarg=0.003       Minimum margin for measurements
102 *MaxMarg=0.003       Maximum margin for measurements
103 *Search=ALL          Place of possibly matching drums (ALL or number of
104 *                    adjacent columns where to look for)
105 *Mode=2              0 = Create, select, print and match
106 *                    1 = Create, select and print
107 *                    2 = Create and print
108 *                    3 = Read drum data from the edit field and match
109 *Zcoord=0            0 = No Printing of the drum Z coordinate
110 *                    1 = Print the drum Z coordinate
111 *Profile=0          0 = No printing of shaft profile coordinates
112 *                    1 = Print the shaft profile coordinates
113 *
114 *DATA DRUMS
115 * Pos       MinH       MaxH
116 *   0       1.46       1.48
117 *   1       1.46       1.49
118 *   2       1.30       1.73
119 *   3       1.30       1.73
120 *   4       1.30       1.73
121 *   5       1.30       1.73
122 *
123 */STRAPEVA DRUMS,0,0.76,0.02,0 98,STRAPEVA
```

I.C. Simulating Non-random Data

```
 37  1 SURVO 84C EDITOR Sat Dec 12 23:22:26 1998          D:\COLMON\ 140 100 C
  1 *SIMUHGHT.TUT
  2 *
  3 *TUTLOAD SIMUHGHT
  4 / Sucro simuhght.tut by Jari Pakkanen (Nov 25 1998)
  5 / for calculating simulated classical confidence
  6 / intervals of shaft height
  7 *{tempo -1}{init}
  8 - if W1 '<>' ? then goto A
  9 *{line start}{d}{erase}{erase}Activating sucro{R}
 10 *{erase}{erase}/SIMUHGHT <data>,<reps>,<minh>,<step>,<maxh>,<file>{R}
 11 *{erase}{erase}simulates temple colonnades and defines a classicalconf
 12 *idence {R}
 13 *{erase}{erase}interval for the mean shaft height. The results are sto
 14 *red in data file{R}
 15 *{erase}{erase}<file>.svo  Specifications must be given as in the exam
 16 *ple edit field.{goto End}
 17 / def Wdata=W1 Wrep=W2 Wmin=W3 Wstep=W4 Wmax=W5 Wfile=W6
 18 / def Whelp1=W7 Wlin2=W8 Wcol2=W9 Wi=W10 Wj=W11 Wk=W12
 19 / def Wfline=W13 Wlline=W14 Wpresdr=W15 Wlin=W17 Wcol=W18
 20 / def Wt025=W19 Wt975=W20 Wlim1=W21 Wlim2=W22
 21 / def WCImin=W23 WCImax=W24 Wmeanh=W25 Wcolh=W26
 22 + A: {save cursor Wlin,Wcol}{Wj=Wmin}{R}
 23 *■SCRATCH /{act}{home}FILE CREATE {print Wfile}{R}
 24 *FIELDS {R}
 25 *    1 NA_    4 Nro       (######){R}
 26 *    2 NA_    4 Height    (##.###){R}
 27 *    3 NA_    4 CIMin     (## ###){R}
 28 *    4 NA_    4 CIMax     (## ###){R}
 29 *    5 NA_    4 OrHght    (##.###){R}
 30 *    6 NA_    4 HDiff     (##.###) Height-OrHght{R}
 31 *    7 NA_    4 CIW       (## ###) CIMax-CIMin{R}
 32 *    8 NA_    1 OK        (#)      Height within CI{R}
 33 *END{R}
 34 *{u11}{act}@SCRATCH /{act}
 35 + MainLoop: {jump 1,1,1,1}{search}ColH=(R)
 36 *{find =}{r}{print W}}    (Wi=0)
 37 /
 38 / Starting loop:
 39 + Loop: {Wi=Wi+1}
 40 / Calling SIMUL.EXE
 41 *{jump 1,1,1,1}{search}Mode=(R)
 42 *{find =}{r}2{jump Wlin,Wlin,1,1}{d}@SCRATCH /{act}{home}SIMUL {}
 43 *{write Wdata},CUR+3,0,70,-100{act}{jump Wlin,Wlin,1,1}{d4}{erase}Col {}
 44 *Dr   DiamL   MaN1  MaP1   DiamU   MaN2  MaP2  Height   MaN3  MaP3{R}
 45 *{save cursor Wlin2,Wcol2}{Wfline=Wlin2}{u}{pre}{d}{pre}{d}{u}
 46 *{save cursor Wlin2,Wcol2}{Wlline=Wlin2}{jump 1,1,1,1}{search}PresDr=
 47 *{R}
 48 *{find =}{r}{ins} {ins}{save word Wpresdr}{1}{del}
 49 /
 50 / Selecting the preserved drums:
 51 *{Wk=0}
 52 + Preserved: {jump Wlin,Wlin,1,1}{d2}{erase}int({print Wfline}+(
 53 *{print Wlline}+1-{print Wfline})*rnd(0)}=(act}{save line Wlin2}
 54 *{jump Wlin2,Wlin2,1,1}{save char Whelp1}
 55 - if Whelp1 '=' * then goto Preserved
 56 ++{Wk=Wk+1}
 57 - if Wk < Wpresdr then goto Preserved
 58 *{jump Wlin,Wlin,1,1}{d5}
 59 + Del: {save char Whelp1}{next word}{save cursor Wlin2,Wcol2}
 60 *{line start}
 61 - if Wcol2 = 1 then goto Cont
 62 - if Whelp1 '=' * then goto Del2
 63 *{del line}{goto Del}
 64 + Del2: {d}{goto Del}
 65 + Cont: {jump Wlin,Wlin,1,1}{d2}{erase}{d}DATA SIMULATX:(R)
 66 *{d}{block}{block}{pre}{d}{pre}{d}{block}{erase}{R}
 67 *{d}STAT SIMULATX.CUR+12 / VARS=Height {act}{R}
```

```
 37  1 SURVO 84C EDITOR Sat Dec 12 23:23:10 1998          D:\COLMON\ 140 100 C
 68 *E=stddev/sqrt(N)*sqrt(1-N/216){R}
 69 *L=E*2.001{R}
 70 *L1=mean-L  6*L1={act}{ins} {save word WCImin}{R}
 71 *L2=mean+L  6*L2={act} {save word WCImax}{R}
 72 *6*mean={act} {save word Wmeanh}{ins}{R}
 73 *.{copy}{R}
 74 *{R}
 75 *DATA DR{R}
 76 *Nro Height CIMin CIMax GrHght HDiff CIW OK{R}
 77 *{print Wi} {print Wmeanh} {print WCImin}{13} {print WCImax}{13} {}
 78 *{print Wj}{Whelp1=Wmeanh-Wj} {print Whelp1}{Whelp1=WCImax-WCImin} {}
 79 *{print Whelp1}{13}{erase}{Whelp1=1}
 80 - if Wj < WCImin then goto !OK
 81 - if Wj > WCImax then goto !OK else goto OK
 82 + !OK: {Whelp1=0}
 83 + OK:  {print Whelp1}{R}
 84 *{R}
 85 *FILE COPY DR,{print Wfile}{act}
 86 /
 87 - if Wi < Wrep then goto Loop
 88 *{Wj=Wj+Wstep}
 89 - if Wj <= Wmax then goto MainLoop
 90 + End: {jump Wlin,Wlin,1,1}{tempo +1}{end}
 91 *
 92 *
 93 *EXAMPLE:
 94 *All dimensions in meters (Parameters needed for SIMUL.EXE: only
 95 *                          height data used by SIMUHGHT.TUT.)
 96 *ColDiamL=1.455       Lower diameter of the column between flutes
 97 *DiamVar=0.005        Range of lower diameters (plus and minus)
 98 *ColDiamU=1.15        Upper diameter of the column between flutes
 99 *ColH=8 98           Column height
100 *MaxEnt=0.01          Maximum entasis
101 *MaxEntH=4.50         Height where the maximum entasis is
102 *ColNFr=6             Number of columns on front
103 *ColNS=8              Number of columns on side
104 *DrN=6                Number of drums in one column
105 *PresDr=60            Number of preserved drums
106 *MinMarg=0.003        Minimum margin for measurements
107 *MaxMarg=0.003        Maximum margin for measurements
108 *Search=ALL           Place of possibly matching drums (ALL or number of
109 *                     adjacent columns where to look for)
110 *Mode=2               0 = Create, select, print and match
111 *                     1 = Create, select and print
112 *                     2 = Create and print
113 *                     3 = Read drum data from the edit field and match
114 *Zcoord=0             0 = No Printing of the drum Z coordinate
115 *                     1 = Print the drum Z coordinate
116 *Profile=0           0 = No printing of shaft profile coordinates
117 *                     1 = Print the shaft profile coordinates
118 *
119 *DATA DRUMS
120 * Pos      MinH      MaxH
121 *   0      1.46      1.48
122 *   1      1.46      1.49
123 *   2      1.30      1.73
124 *   3      1.30      1.73
125 *   4      1.30      1.73
126 *   5      1.30      1.73
127 *
128 */SIMUHGHT DRUMS,24,8.76,0.02,8 98,SIMUHT24
```

2.A. Colonnade simulation

```
  1  1 SURVO 98  Sat Dec 12 23:51:44 1998          C:\COLMON\    1000    80 0
  1 *!SIMUL.EXE
  2 *
  3 */* !SIMUL.C Oct 10th 1996/Jari Pakkanen */
  4 *
  5 *#include <stdio.h>
  6 *#include <stdlib.h>
  7 *#include <conio.h>
  8 *#include <malloc.h>
  9 *#include <math.h>
 10 *#include <time.h>
 11 *#include "survo.h"
 12 *#include "survoext.h"
 13 *#include "survodat.h"
 14 *
 15 *#define DRMAX 14      /* Max number of drums in one column              */
 16 *#define COLMAX 50     /* Max number of columns in the building          */
 17 *#define EPS 0.0001    /* Epsilon value                                  */
 18 *
 19 *FILE *fpt;           /* Filepointer                                    */
 20 *SURVO_DATA d;
 21 *
 22 *double Pos[DRMAX],   /* Position of drum range                         */
 23 *double MinH[DRMAX],  /* Minumum height of drum range                   */
 24 *double MaxH[DRMAX],  /* Maximum height of drum range                   */
 25 *double ColDiaml,     /* Lower diameter of the column between flutes    */
 26 *double DiamVar,      /* Range of lower diameters (plus and minus)      */
 27 *double ColDiamU,     /* Upper Diameter of the column between flutes    */
 28 *double ColH;         /* Column height                                  */
 29 *int    ColNFr,       /* Number of columns on front                     */
 30 *int    ColNS;        /* Number of columns on side                      */
 31 *int    DrN,          /* Number of drums in one column                  */
 32 *int    PresDr;       /* Number of preserved drums                      */
 33 *double MinMarg,      /* Minimum margin for measurements                */
 34 *double MaxMarg,      /* Maximum margin for measurements                */
 35 *char   *Search,      /* Place of possibly matching drums               */
 36 *int    search,       /* Number of adjacent drums in search             */
 37 *int    Mode;         /* Program mode                                   */
 38 *int    Zcoord;       /* Print drum Z coordinate                        */
 39 *int    Profile;      /* Print shaft profile coordinates                */
 40 *double aa;           /* Constant of the entasis parabola               */
 41 *double bb;           /* Coefficent of x of the entasis parabola        */
 42 *double cc;           /* Coefficent of x^2 of the entasis parabola      */
 43 *int    ColN;         /* Number of columns                              */
 44 *char   txtname[LLENGTH];  /* Name of the output txt-file               */
 45 *int    dc[DRMAX];    /* Drum counter for matching drums                */
 46 *
 47 *struct drum          /* drum information                               */
 48 *  { int    pres,     /* drum preserved (0=not pres. 1=preserved)       */
 49 *    double zcoord,   /* Z coordinate of the bottom of the drum         */
 50 *    double diaml;    /* lower diameter between flutes                  */
 51 *    double diamu;    /* upper diameter between flutes                  */
 52 *    double height,   /* height of the drum                             */
 53 *    double diamlnmar, /* neg. measurement margin for lower diameter    */
 54 *    double diamlpmar; /* pos. measurement margin for lower diameter    */
 55 *    double diamunmar, /* neg. measurement margin for upper diameter    */
 56 *    double diamupmar, /* pos. measurement margin for upper diameter    */
 57 *    double heightnmar; /* neg  measurement margin for height           */
 58 *    double heightpmar; /* pos. measurement margin for height           */
 59 *  },
 60 *
 61 *struct colonnade     /* colonnade information                          */
 62 *  { struct drum dr[DRMAX],
 63 *  } col[COLMAX];
 64 *
 65 *int i,j,k,match,minsearch,missing_level,maxsearch,results_line,
 66 *long l;
 67 *double adj,dd,diff,height,help_i,help_j,random,root1,root2,
 68 *char line[LLENGTH],
 69 *time_t start;
 70 *
```

```
  1  1 SURVO 98  Sat Dec 12 23:53:18 1998          D:\COLMGN\    1000    80 C
 71 *main(argc,argv)
 72 *int argc; char *argv[];
 73 *   {
 74 *   if (argc==1) return;
 75 *   s_init(argv[1]);
 76 *   if (g<3)
 77 *      {
 78 *      sur_print("\nUsage: SIMUL data,output line,<coeff  a>,
                     <coeff  b(x)>,<coeff  c(x^2)>");
 79 *      WAIT; return;
 80 *      }
 81 *   results_line=0;
 82 *   results_line=edline2(word[2],1,1);
 83 *   if (results_line==0) return;
 84 *   i=data_open(word[1],&d); if (i<0) return;
 85 *   i=sp_init(r1+r-1); if (i<0) return;
 86 *   i=mark(&d); if (i<0) return;
 87 *
 88 *   /* Finding spefications */
 89 *   i=spfind("Mode");
 90 *   if (i>=0)
 91 *      {
 92 *      Mode=atoi(spb[i]);
 93 *      if (Mode<0 || Mode>3)
 94 *         {
 95 *         sprintf(sbuf,"\nError in specification Mode");
 96 *         sur_print(sbuf); WAIT; return;
 97 *         }
 98 *      }
 99 *   else
100 *      {
101 *      sprintf(sbuf,"\nError in specification Mode");
102 *      sur_print(sbuf); WAIT; return;
103 *      }
104 *   i=spfind("Zcoord");
105 *   if (i>=0)
106 *      {
107 *      Zcoord=atoi(spb[i]);
108 *      if (Zcoord<0 || Zcoord>1)
109 *         {
110 *         sprintf(sbuf,"\nError in specification Zcoord");
111 *         sur_print(sbuf); WAIT; return;
112 *         }
113 *      }
114 *   else
115 *      {
116 *      sprintf(sbuf,"\nError in specification Zcoord");
117 *      sur_print(sbuf); WAIT; return;
118 *      }
119 *   i=spfind("Profile");
120 *   if (i>=0)
121 *      {
122 *      Profile=atoi(spb[i]);
123 *      if (Profile<0 || Profile>1)
124 *         {
125 *         sprintf(sbuf,"\nError in specification Profile");
126 *         sur_print(sbuf); WAIT; return;
127 *         }
128 *      }
129 *   else
130 *      {
131 *      sprintf(sbuf,"\nError in specification Profile");
132 *      sur_print(sbuf); WAIT; return;
133 *      }
134 *   i=spfind("ColNFr");
135 *   if (i>=0)
136 *      {
137 *      ColNFr=atoi(spb[i]);
138 *      if (ColNFr<1 || ColNFr>12)
139 *         {
140 *         sprintf(sbuf,"\nError in specification ColNFr");
141 *         sur_print(sbuf); WAIT; return;
142 *         }
```

```
  1  1 SURVO 98  Sun Dec 13 00:08:51 1998          D:\COLMON\    1000    80 C
143 *       }
144 *    else
145 *       {
146 *       sprintf(sbuf,"\nError in specification ColNFr");
147 *       sur_print(sbuf); WAIT; return;
148 *       }
149 *    i=spfind("ColNS");
150 *    if (i>=0)
151 *       {
152 *       ColNS=atoi(spb[i]);
153 *       if (ColNS<1 || ColNS>20)
154 *          {
155 *          sprintf(sbuf,"\nError in specification ColNS");
156 *          sur_print(sbuf); WAIT; return;
157 *          }
158 *       }
159 *    else
160 *       {
161 *       sprintf(sbuf,"\nError in specification ColNS");
162 *       sur_print(sbuf); WAIT; return;
163 *       }
164 *    i=spfind("DrN");
165 *    if (i>=0)
166 *       {
167 *       DrN=atoi(spb[i]);
168 *       if (DrN<1 || DrN>DRMAX)
169 *          {
170 *          sprintf(sbuf,"\nError in specification DrN");
171 *          sur_print(sbuf); WAIT; return;
172 *          }
173 *       }
174 *    else
175 *       {
176 *       sprintf(sbuf,"\nError in specification DrN");
177 *       sur_print(sbuf); WAIT; return;
178 *       }
179 *    i=spfind("Search");
180 *    if (i>=0)
181 *       Search=spb[i];
182 *    else
183 *       {
184 *       sprintf(sbuf,"\nError in specification Search");
185 *       sur_print(sbuf); WAIT; return;
186 *       }
187 *    if (Mode<3)
188 *       {
189 *       i=spfind("ColDiamL");
190 *       if (i>=0)
191 *          {
192 *          ColDiamL=atof(spb[i]);
193 *          if (ColDiamL<0 || ColDiamL>5)
194 *             {
195 *             sprintf(sbuf,"\nError in specification ColDiamL");
196 *             sur_print(sbuf); WAIT; return;
197 *             }
198 *          }
199 *       else
200 *          {
201 *          sprintf(sbuf,"\nError in specification ColDiamL");
202 *          sur_print(sbuf); WAIT; return;
203 *          }
204 *       i=spfind("DiamVar");
205 *       if (i>=0)
206 *          {
207 *          DiamVar=atof(spb[i]);
208 *          if (DiamVar<0 || DiamVar>1)
209 *             {
210 *             sprintf(sbuf,"\nError in specification DiamVar");
211 *             sur_print(sbuf); WAIT; return;
212 *             }
213 *          }
214 *       else
215 *          {
```

```
  1  1 SURVO 98  Sun Dec 13 00:10:01 1998          D:\COLMON\   1000   80 C
216 *          sprintf(sbuf,"\nError in specification DiamVar");
217 *          sur_print(sbuf); WAIT; return;
218 *          }
219 *       i=spfind("ColDiamU");
220 *       if (i>=0)
221 *          {
222 *          ColDiamU=atof(spb[i]);
223 *          if (ColDiamU<0 || ColDiamU>5)
224 *             {
225 *             sprintf(sbuf,"\nError in specification ColDiamU");
226 *             sur_print(sbuf); WAIT; return;
227 *             }
228 *          }
229 *       else
230 *          {
231 *          sprintf(sbuf,"\nError in specification ColDiamU");
232 *          sur_print(sbuf); WAIT; return;
233 *          }
234 *       i=spfind("ColH");
235 *       if (i>=0)
236 *          {
237 *          ColH=atof(spb[i]);
238 *          if (ColH<0 || ColH>20)
239 *             {
240 *             sprintf(sbuf,"\nError in specification ColH");
241 *             sur_print(sbuf); WAIT; return;
242 *             }
243 *          }
244 *       else
245 *          {
246 *          sprintf(sbuf,"\nError in specification ColH");
247 *          sur_print(sbuf); WAIT; return;
248 *          }
249 *       i=spfind("PresDr");
250 *       if (i>=0)
251 *          {
252 *          PresDr=atol(spb[i]);
253 *          if (PresDr<0 || PresDr>(2*ColNFr+2*(ColNS-2))*DRMAX)
254 *             {
255 *             sprintf(sbuf,"\nError in specification PresDr");
256 *             sur_print(sbuf); WAIT; return;
257 *             }
258 *          }
259 *       else
260 *          {
261 *          sprintf(sbuf,"\nError in specification PresDr");
262 *          sur_print(sbuf); WAIT; return;
263 *          }
264 *       i=spfind("MinMarg");
265 *       if (i>=0)
266 *          {
267 *          MinMarg=atof(spb[i]);
268 *          if (MinMarg<0 || MinMarg>1)
269 *             {
270 *             sprintf(sbuf,"\nError in specification MinMarg");
271 *             sur_print(sbuf); WAIT; return;
272 *             }
273 *          }
274 *       else
275 *          {
276 *          sprintf(sbuf,"\nError in specification MinMarg");
277 *          sur_print(sbuf); WAIT; return;
278 *          }
279 *       i=spfind("MaxMarg");
280 *       if (i>=0)
281 *          {
282 *          MaxMarg=atof(spb[i]);
283 *          if (MaxMarg<0 || MaxMarg>1)
284 *             {
285 *             sprintf(sbuf,"\nError in specification MaxMarg");
286 *             sur_print(sbuf); WAIT; return;
287 *             }
288 *          }
```

```
  1  1 SURVO 98  Sun Dec 13 00:11:16 1998          D:\COLMON\    1000    80 C
288 *          }
289 *       else
290 *          {
291 *          sprintf(sbuf,"\nError in specification MaxMarg");
292 *          sur_print(sbuf); WAIT; return;
293 *          }
294 *       /* Coefficents of the entasis parabola */
295 *       aa=atof(word[3]);
296 *       bb=atof(word[4]);
297 *       cc=atof(word[5]);
298 *       }
299 *
300 *    /* Reading data from the edit field */
301 *    ColN=2*ColNFr+2*(ColNS-2)-1;
302 *    if (Mode<3)
303 *       {
304 *       i=0;
305 *       for (l=d.l1; l<=d.l2; ++l)
306 *          {
307 *          data_load(&d,l,d.v[0],&Pos[i]);
308 *          data_load(&d,l,d.v[1],&MinH[i]);
309 *          data_load(&d,l,d.v[2],&MaxH[i]);
310 *          ++i;
311 *          }
312 *       }
313 *    else
314 *       {
315 *       for (i=0; i<=ColN; ++i)
316 *          for (j=0; j<DrN; ++j)
317 *             col[i].dr[j].pres=0;
318 *       for (l=d.l1; l<=d.l2; ++l)
319 *          {
320 *          data_load(&d,l,d.v[0],&help_i);
321 *          i=(int)(help_i);
322 *          data_load(&d,l,d.v[1],&help_j);
323 *          j=(int)(help_j);
324 *          data_load(&d,l,d.v[2],&col[i].dr[j].diaml);
325 *          data_load(&d,l,d.v[3],&col[i].dr[j].diamlnmar);
326 *          data_load(&d,l,d.v[4],&col[i].dr[j].diamlpmar);
327 *          data_load(&d,l,d.v[5],&col[i].dr[j].diamu);
328 *          data_load(&d,l,d.v[6],&col[i].dr[j].diamunmar);
329 *          data_load(&d,l,d.v[7],&col[i].dr[j].diamupmar);
330 *          data_load(&d,l,d.v[8],&col[i].dr[j].height);
331 *          data_load(&d,l,d.v[9],&col[i].dr[j].heightnmar);
332 *          data_load(&d,l,d.v[10],&col[i].dr[j].heightpmar);
333 *          if (Zcoord==1)
334 *             data_load(&d,l,d.v[11],&col[i].dr[j].zcoord);
335 *          col[i].dr[j].pres=1;
336 *          }
337 *       }
338 *    data_close(&d);
339 *
340 *    /* CREATING THE COLONNADE */
341 *
342 *    if (Mode<3)
343 *       {
344 *       srand(time(&start));
345 *       for (i=0; i<=ColN; ++i)
346 *          {
347 *          height=0;
348 *          for (j=0; j<DrN; ++j)
349 *             {
350 *             random=rand()/32768.0;
351 *             col[i].dr[j].height=MinH[j]+(MaxH[j]-MinH[j])*random;
352 *             height=height+col[i].dr[j].height;
353 *             }
354 *          /* Adjusting the drum heights */
355 *          diff=ColH-height;
356 *          if (fabs(diff)>MaxMarg)
357 *             do {
358 *                j=(int)((DrN-1)*(rand()/32768.0));
359 *                if (diff<0)
360 *                   if (col[i].dr[j].height+diff>=MinH[j])
361 *                      {
```

```
  1  1 SURVO 98  Sun Dec 13 00:11:59 1998          D:\COLMON\    1000    80 C
362 *                      col[i].dr[j].height=col[i].dr[j].height+diff;
363 *                      height=height+diff;
364 *                      }
365 *                   else
366 *                      {
367 *                      height=height-col[i].dr[j].height;
368 *                      adj=0.33*(col[i].dr[j].height-MinH[j]);
369 *                      col[i].dr[j].height=col[i].dr[j].height-adj;
370 *                      height=height+col[i].dr[j].height;
371 *                      }
372 *                else
373 *                   if (col[i].dr[j].height+diff<=MaxH[j])
374 *                      {
375 *                      col[i].dr[j].height=col[i].dr[j].height+diff;
376 *                      height=height+diff;
377 *                      }
378 *                   else
379 *                      {
380 *                      height=height-col[i].dr[j].height;
381 *                      adj=0.33*(MaxH[j]-col[i].dr[j].height);
382 *                      col[i].dr[j].height=col[i].dr[j].height+adj;
383 *                      height=height+col[i].dr[j].height;
384 *                      }
385 *             diff=ColH-height;
386 *             } while (fabs(diff)>MaxMarg);
387 *          }
388 *       /* Rounding heights to millimeters */
389 *       for (i=0; i<=ColN; ++i)
390 *          for (j=0; j<DrN; ++j)
391 *             {
392 *             dd=1000*(col[i].dr[j].height)+0.5;
393 *             col[i].dr[j].height=(int)(dd)/1000.0;
394 *             }
395 *       /* Bottom drum diameters */
396 *       for (i=0; i<=ColN; ++i)
397 *          {
398 *          random=rand()/32768.0;
399 *          dd=1000*(ColDiamL-DiamVar+2*DiamVar*random)+0.5;
400 *          col[i].dr[0].diaml=(int)(dd)/1000.0;
401 *          col[i].dr[0].xcoord=0.0;
402 *          height=col[i].dr[0].height;
403 *          root1=(-bb+sqrt(bb*bb-(4*cc*(aa-height))))/(2*cc);
404 *          root2=(-bb-sqrt(bb*bb-(4*cc*(aa-height))))/(2*cc);
405 *          if (fabs(root1)<=fabs(root2)) diff=root1;
406 *              else diff=root2;
407 *          dd=1000*(col[i].dr[0].diaml-2*diff)+0.5;
408 *          col[i].dr[0].diamu=(int)(dd)/1000.0;
409 *          /* Measurement margins */
410 *          random=rand()/32768.0;
411 *          dd=1000*(MinMarg+(MaxMarg-MinMarg)*random)+0.5;
412 *          col[i].dr[0].heightpmar=(int)(dd)/1000.0;
413 *          col[i].dr[0].heightmmar=-col[i].dr[0].heightpmar;
414 *          random=rand()/32768.0;
415 *          dd=1000*(MinMarg+(MaxMarg-MinMarg)*random)+0.5;
416 *          col[i].dr[0].diamlpmar=(int)(dd)/1000.0;
417 *          col[i].dr[0].diamlmmar=-col[i].dr[0].diamlpmar;
418 *          random=rand()/32768.0;
419 *          dd=1000*(MinMarg+(MaxMarg-MinMarg)*random)+0.5;
420 *          col[i].dr[0].diamupmar=(int)(dd)/1000.0;
421 *          col[i].dr[0].diammmar=-col[i].dr[0].diamupmar;
422 *          }
423 *       /* Diameters of other drums */
424 *       for (i=0; i<=ColN; ++i)
425 *          {
426 *          height=col[i].dr[0].height;
427 *          for (j=1; j<DrN; ++j)
428 *             {
429 *             col[i].dr[j].xcoord=height;
430 *             height=height+col[i].dr[j].height;
431 *             root1=(-bb+sqrt(bb*bb-(4*cc*(aa-height))))/(2*cc);
432 *             root2=(-bb-sqrt(bb*bb-(4*cc*(aa-height))))/(2*cc);
433 *             if (fabs(root1)<=fabs(root2)) diff=root1;
434 *                else diff=root2;
```

```
  1  1 SURVO 98  Sun Dec 13 00:12:45 1998          D:\COLMON\    1000    80 0
435 *              col[i].dr[j].diaml=col[i].dr[j-1].diamu;
436 *              dd=1000*(col[i].dr[0].diaml-2*diff)+0.5;
437 *              col[i].dr[j].diamu=(int)(dd)/1000.0;
438 *              /* Measurement margins */
439 *              random=rand()/32768.0;
440 *              dd=1000*(MinMarg+(MaxMarg-MinMarg)*random)+0.5;
441 *              col[i].dr[j].heightpmar=(int)(dd)/1000.0;
442 *              col[i].dr[j].heightnmar=-col[i].dr[j].heightpmar;
443 *              random=rand()/32768.0;
444 *              dd=1000*(MinMarg+(MaxMarg-MinMarg)*random)+0.5;
445 *              col[i].dr[j].diamlpmar=(int)(dd)/1000.0;
446 *              col[i].dr[j].diamlnmar=-col[i].dr[j].diamlpmar;
447 *              random=rand()/32768.0;
448 *              dd=1000*(MinMarg+(MaxMarg-MinMarg)*random)+0.5;
449 *              col[i].dr[j].diamupmar=(int)(dd)/1000.0;
450 *              col[i].dr[j].diamunmar=-col[i].dr[j].diamupmar;
451 *              }
452 *           }
453 *        }
454 *
455 *     /* SELECTING PRESERVED DRUMS */
456 *
457 *     if (Mode<2)
458 *        {
459 *        for (i=0; i<=ColN; ++i)
460 *           for (j=0; j<DrN; ++j)
461 *              col[i].dr[j].pres=0;
462 *        k=0;
463 *        do {
464 *           i=(int)((ColN+1)*(rand()/32768.0));
465 *           j=(int)((DrN)*(rand()/32768.0));
466 *           if (col[i].dr[j].pres==0)
467 *              {
468 *              col[i].dr[j].pres=1;
469 *              k=++k;
470 *              }
471 *           } while (k<PresDr);
472 *        }
473 *     else
474 *        if (Mode==2)
475 *           for (i=0; i<=ColN; ++i)
476 *              for (j=0; j<DrN; ++j)
477 *                 col[i].dr[j].pres=1;
478 *
479 *     /* PAIRS OF TRULY MATCHING DRUMS */
480 *
481 *     if (Mode<2)
482 *        {
483 *        k=0;
484 *        for (i=0; i<=ColN; ++i)
485 *           for (j=0; j<DrN-1; ++j)
486 *              if (col[i].dr[j].pres==1 && col[i].dr[j+1].pres==1)
487 *                 ++k;
488 *        }
489 *
490 *     /* OUTPUT OF DATA TO EDIT FIELD */
491 *
492 *     output_open(eout);
493 *     if (Mode<3)
494 *        {
495 *        if (Mode<2)
496 *           {
497 *           sprintf(line,"Number of truly matching pairs: %i", k);
498 *           print_line(line);
499 *           }
500 *        if (Zcoord==0)
501 *        strcpy(line,"Col Dr   DiamL   -Mar   Mar   DiamU   -Mar   Mar
                      Height   -Mar   Mar");
502 *        else
503 *          strcpy(line,
504 *          "Col Dr   DiamL   -Mar   Mar   DiamU   -Mar   Mar   Height
             Height   -Mar   Mar   Zcoord");
505 *        print_line(line);
```

```
  1  1 SURVO 98  Sun Dec 13 00:35:50 1998          D:\COLMGN\    1000    80 C
506 *       for (j=0; j<DrN; ++j)
507 *          for (i=0; i<=ColN; ++i)
508 *             if (col[i].dr[j].pres==1)
509 *                if (Zcoord==0)
510 *                   {
511 *                   sprintf(line,
512 *                       "%3i %2i %7.3f %6.3f %5.3f %7.3f %6.3f %5.3f
                        %7.3f %6.3f %5.3f",
513 *                       i, j, col[i].dr[j].diaml, col[i].dr[j].diamlnmar,
514 *                       col[i].dr[j].diamlpmar, col[i].dr[j].diamu,
515 *                       col[i].dr[j].diamunmar, col[i].dr[j].diamupmar,
516 *                       col[i].dr[j].height, col[i].dr[j].heightnmar,
517 *                       col[i].dr[j].heightpmar);
518 *                   print_line(line);
519 *                   }
520 *                else
521 *                   {
522 *                   sprintf(line,
523 *                       "%3i %2i %7.3f %6.3f %5.3f %7.3f %6.3f %5.3f
                        %7.3f %6.3f %5.3f %6.3f",
524 *                       i, j, col[i].dr[j].diaml, col[i].dr[j].diamlnmar,
525 *                       col[i].dr[j].diamlpmar, col[i].dr[j].diamu,
526 *                       col[i].dr[j].diamunmar, col[i].dr[j].diamupmar,
527 *                       col[i].dr[j].height, col[i].dr[j].heightnmar,
528 *                       col[i].dr[j].heightpmar, col[i].dr[j].zcoord);
529 *                   print_line(line);
530 *                   }
531 *       }
532 *
533 *    /* MATCHING DRUMS */
534 *
535 *    if (Mode==0 || Mode==3)
536 *       {
537 *       sprintf(txtname,"%s%s.txt",edisk, word[1]);
538 *       fpt=fopen(txtname,"w");
539 *       missing_level=0;
540 *       for (j=0; j<=DrN-1; ++j)
541 *          {
542 *          k=0;
543 *          for (i=0; i<=ColN; ++i)
544 *             if (col[i].dr[j].pres==1) ++k;
545 *          if (k==0) missing_level=1;
546 *          }
547 *       if (!missing_level)
548 *          {
549 *          strcpy(line,Search);
550 *          i=strlen(line);
551 *          if (i<3) /* Limited search */
552 *             {
553 *             search=atoi(Search);
554 *             i=-1;
555 *             do ++i; while (col[i].dr[0].pres==0 && i<=ColN);
556 *             if (i<=ColN)
557 *                {
558 *                maxsearch=i+search;
559 *                if (maxsearch>ColN) maxsearch=maxsearch-ColN-1;
560 *                minsearch=i-search;
561 *                if (minsearch<0) minsearch=ColN+minsearch+1;
562 *                k=first_path_lim_search();
563 *                if (k==0)
564 *                   {
565 *                   strcpy(line,"No matching complete columns ");
566 *                   print_line(line);
567 *                   }
568 *                else
569 *                   {
570 *                   print_to_file();
571 *                   do {
572 *                      k=find_path_lim_search();
573 *                      if (k!=0) print_to_file();
574 *                      } while (k!=0);
575 *                   }
576 *                }
577 *             else
578 *                {
```

```
  1  1 SURVO 98  Sun Dec 13 00:38:31 1998          D:\COLMON\    1000    80 C
579 *                strcpy(line,"No matching complete columns.");
580 *                print_line(line);
581 *                }
582 *             }
583 *          else /* Search all */
584 *             {
585 *             i=-1;
586 *             do ++i; while (col[i].dr[0].pres==0 && i<=ColN);
587 *             if (i<=ColN)
588 *                {
589 *                minsearch=0;
590 *                k=first_path_all();
591 *                if (k==0)
592 *                   {
593 *                   strcpy(line,"No matching complete columns.");
594 *                   print_line(line);
595 *                   }
596 *                else
597 *                   {
598 *                   print_to_file();
599 *                   do {
600 *                      k=find_path_all();
601 *                      if (k!=0) print_to_file();
602 *                      } while (k!=0);
603 *                   }
604 *                }
605 *             else
606 *                {
607 *                strcpy(line,"No matching complete columns.");
608 *                print_line(line);
609 *                }
610 *             }
611 *          }
612 *       else
613 *          {
614 *          strcpy(line,"No matching complete columns ");
615 *          print_line(line);
616 *          }
617 *       fclose(fpt);
618 *       }
619 *    output_close(eout);
620 *    }
621 *
622 *print_line(line)
623 *char *line;
624 *    {
625 *    output_line(line,eout,results_line);
626 *    if (results_line) ++results_line;
627 *    }
628 *
629 *print_to_file()
630 *    {
631 *    int i;
632 *    double height,height_neg_mar,height_pos_mar,prof_coor;
633 *    char line2[LLENGTH];
634 *
635 *    if (Profile==1)
636 *       {
637 *       fprintf(fpt,"%i,0,", dc[0]);
638 *       height=col[dc[0]].dr[0].height;
639 *       height_neg_mar=col[dc[0]].dr[0].heightnmar;
640 *       height_pos_mar=col[dc[0]].dr[0].heightpmar;
641 *       prof_coor=col[dc[0]].dr[0].diaml/2-(col[dc[0]].dr[0].diamu
                    +col[dc[1]].dr[1].diaml)/4;
642 *       fprintf(fpt,"%f,%f,", prof_coor,height);
643 *       for (i=1; i<DrN-1; ++i)
644 *          {
645 *          fprintf(fpt,"%i,%i,", dc[i], i);
646 *          height=height+col[dc[i]].dr[i].height;
647 *          height_neg_mar=height_neg_mar+col[dc[i]].dr[i].heightnmar;
648 *          height_pos_mar=height_pos_mar+col[dc[i]].dr[i].heightpmar;
649 *          prof_coor=col[dc[0]].dr[0].diaml/2-(col[dc[i]].dr[i].diamu
                       +col[dc[1]].dr[1].diaml)/4;
```

```
  1  1 SURVQ 98  Sun Dec 13 00:37:42 1998          D:\COLMGN\    1000    80 C
650 *          fprintf(fpt,"%f,%f,", prof_coor,height);
651 *          }
652 *       i=DrN-1;
653 *       fprintf(fpt,"%i,%i,", dc[i], i);
654 *       height=height+col[dc[i]].dr[i].height;
655 *       height_neg_mar=height_neg_mar+col[dc[i]].dr[i].heightnmar;
656 *       height_pos_mar=height_pos_mar+col[dc[i]].dr[i].heightpmar;
657 *       prof_coor=(col[dc[0]].dr[0].diaml-col[dc[i]].dr[i].diamu)/2;
658 *       fprintf(fpt,"%f,%f,%f,%f\n", prof_coor,height,height_neg_mar,
                 height_pos_mar);
659 *       }
660 *    else
661 *       {
662 *       fprintf(fpt,"%i,0,", dc[0]);
663 *       height=col[dc[0]].dr[0].height;
664 *       height_neg_mar=col[dc[0]].dr[0].heightnmar;
665 *       height_pos_mar=col[dc[0]].dr[0].heightpmar;
666 *       for (i=1; i<DrN-1; ++i)
667 *          {
668 *          fprintf(fpt,"%i,%i,", dc[i], i);
669 *          height=height+col[dc[i]].dr[i].height;
670 *          height_neg_mar=height_neg_mar+col[dc[i]].dr[i].heightnmar;
671 *          height_pos_mar=height_pos_mar+col[dc[i]].dr[i].heightpmar;
672 *          }
673 *       i=DrN-1;
674 *       fprintf(fpt,"%i,%i,", dc[i], i);
675 *       height=height+col[dc[i]].dr[i].height;
676 *       height_neg_mar=height_neg_mar+col[dc[i]].dr[i].heightnmar;
677 *       height_pos_mar=height_pos_mar+col[dc[i]].dr[i].heightpmar;
678 *       fprintf(fpt,"%f,%f,%f\n", height,height_neg_mar,height_pos_mar);
679 *       }
680 *    }
681 *
682 */* FINDING THE FIRST PATH IN LIMITED SEARCH */
683 *
684 *first_path_lim_search()
685 *    {
686 *    /* NOT IMPLEMENTED */
687 *    }
688 *
689 */* FINDING THE NEXT PATH IN LIMITED SEARCH */
690 *
691 *find_path_lim_search()
692 *    {
693 *    /* NOT IMPLEMENTED */
694 *    }
695 *
696 */* FINDING THE FIRST PATH IN SEARCHING ALL POSSIBILITIES */
697 *
698 *first_path_all()
699 *    {
700 *    int column,dead_end,i0,i1,j0,k0,more_drums,no_match;
701 *    double maxlower,maxupper,minlower,minupper;
702 *
703 *    column=0;
704 *    dc[0]=i;
705 *    for (j0=1; j0<DrN; ++j0)
706 *       dc[j0]=-1;
707 *    do {
708 *       i0=i; j0=0; dead_end=0;
709 *       do {
710 *          i1=minsearch; no_match=1;
711 *          do {
712 *             if (col[i0].dr[j0].pres==1 && col[i1].dr[j0+1].pres==1)
713 *                {
714 *                minlower=col[i0].dr[j0].diamu+col[i0].dr[j0].diamunmar;
715 *                maxlower=col[i0].dr[j0].diamu+col[i0].dr[j0].diamupmar;
716 *                minupper=col[i1].dr[j0+1].diaml
                            +col[i1].dr[j0+1].diamlnmar;
717 *                maxupper=col[i1].dr[j0+1].diaml
                            +col[i1].dr[j0+1].diamlpmar;
718 *                if ((minlower-maxupper)>EPS || EPS<(minupper-maxlower))
719 *                   ++i1;
```

```
  1  1 SURVO 98  Sun Dec 13 00:38:33 1998          D:\COLMON\    1000    80 C
720 *                  else
721 *                     {
722 *                     no_match=0;
723 *                     dc[j0+1]=i1;
724 *                     i0=i1;
724 *                     i0=i1;
725 *                     }
726 *                  }
727 *               else ++i1;
728 *               } while (no_match && i1<=ColN);
729 *            if (!no_match)
730 *               {
731 *               ++j0; minsearch=0;
732 *               if (j0==DrN-1) column=1;
733 *               }
734 *            else
735 *               {
736 *               if (j0==0) dead_end=1;
737 *               if (j0>0 && j0<DrN-1)
738 *                  {
739 *                  more_drums=0; k0=dc[j0];
740 *                  do ++k0; while (col[k0].dr[j0].pres==0 && k0<=ColN);
741 *                  if (k0<=ColN) more_drums=1;
742 *                  if (more_drums)
743 *                     {
744 *                     minsearch=k0;
745 *                     if (j0>1) dc[j0]=-1;
746 *                     if (j0>0) --j0;
747 *                     i0=dc[j0];
748 *                     }
749 *                  else
750 *                     {
751 *                     do {
752 *                        if (j0>1) dc[j0]=-1;
753 *                        --j0; k0=dc[j0];
754 *                        do ++k0; while (col[k0].dr[j0].pres==0 &&
                                          k0<=ColN);
755 *                        if (k0<=ColN) more_drums=1;
756 *                        } while (!more_drums && j0>1);
757 *                     if (j0>1) dc[j0]=-1;
758 *                     if (j0>0) --j0;
759 *                     i0=dc[j0];
760 *                     if (more_drums) minsearch=k0;
761 *                     else minsearch=0;
762 *                     }
763 *                  if (j0<1)
764 *                     {
765 *                     i0=dc[0];
766 *                     k0=dc[1];
767 *                     do ++k0; while (col[k0].dr[1].pres==0 && k0<=ColN);
768 *                     if (k0>ColN) dead_end=1;
769 *                     else minsearch=k0;
770 *                     }
771 *                  }
772 *               }
773 *            } while (j0<DrN-1 && !dead_end);
774 *         if (column==0)
775 *            {
776 *            do ++i; while (col[i].dr[0].pres==0 && i<=ColN);
777 *            minsearch=0;
778 *            dc[0]=i;
779 *            for (j0=1; j0<DrN; ++j0)
780 *               dc[j0]=-1;
781 *            }
782 *         } while (column==0 && i<=ColN);
783 *      return(column);
784 *      }
785 *
786 */* FINDING THE NEXT PATH IN SEARCHING ALL POSSIBILITIES */
787 *
788 *find_path_all()
789 *      {
790 *      int column,dead_end,i0,i1,j0,k0,more_drums,no_match;
791 *      double maxlower,maxupper,minlower,minupper;
```

```
  1  1 SURVO 98  Sun Dec 13 00:39:09 1998          D:\COLMON\    1000    80 C
792 *
793 *    column=0; j0=DrN-1; more_drums=0; dead_end=0;
794 *    k0=dc[j0];
795 *    do ++k0; while (col[k0].dr[j0].pres==0 && k0<=ColN);
796 *    if (k0<=ColN) more_drums=1;
797 *    if (more_drums)
798 *       {
799 *       minsearch=k0;
800 *       if (j0>1) dc[j0]=-1;
801 *       if (j0>0) --j0;
802 *       i0=dc[j0];
803 *       }
804 *    else
805 *       {
806 *       do {
807 *          if (j0>1) dc[j0]=-1;
808 *          --j0; k0=dc[j0];
809 *          do ++k0; while (col[k0].dr[j0].pres==0 && k0<=ColN);
810 *          if (k0<=ColN) more_drums=1;
811 *          } while (!more_drums && j0>1);
812 *       if (j0>1) dc[j0]=-1;
813 *       if (j0>0) --j0;
814 *       i0=dc[j0];
815 *       if (more_drums) minsearch=k0;
816 *       else minsearch=0;
817 *       }
818 *    if (j0<1)
819 *       {
820 *       i0=dc[0];
821 *       k0=dc[1];
822 *       do ++k0; while (col[k0].dr[1].pres==0 && k0<=ColN);
823 *       if (k0>ColN)
824 *          {
825 *          do ++i; while (col[i].dr[0].pres==0 && i<=ColN);
826 *          if (i>ColN) dead_end=1;
827 *          else
828 *             {
829 *             minsearch=0;
830 *             dc[0]=i;
831 *             for (j0=1; j0<DrN; ++j0)
832 *                dc[j0]=-1;
833 *             i0=i; j0=0;
834 *             }
835 *          }
836 *       else minsearch=k0;
837 *       }
838 *    if (!dead_end)
839 *       do {
840 *          dead_end=0;
841 *          do {
842 *             i1=minsearch; no_match=1;
843 *             do {
844 *                if (col[i0].dr[j0].pres==1 && col[i1].dr[j0+1].pres==1)
845 *                   {
846 *                   minlower=col[i0].dr[j0].diamu
                                 +col[i0].dr[j0].diamunmar;
847 *                   maxlower=col[i0].dr[j0].diamu
                                 +col[i0].dr[j0].diamupmar;
848 *                   minupper=col[i1].dr[j0+1].diaml
                                 +col[i1].dr[j0+1].diamlnmar;
849 *                   maxupper=col[i1].dr[j0+1].diaml
                                 +col[i1].dr[j0+1].diamlpmar;
850 *                   if ((minlower-maxupper)>EPS || EPS<(minupper
                                                          -maxlower))
851 *                      ++i1;
852 *                   else
853 *                      {
854 *                      no_match=0;
855 *                      dc[j0+1]=i1;
856 *                      i0=i1;
857 *                      }
858 *                   }
```

```
  1  1 SURVO 98  Sun Dec 13 00:39:51 1998          D:\COLMON\    1000    80 C
859 *                else ++i1;
860 *                } while (no_match && i1<=ColN);
861 *             if (!no_match)
862 *                {
863 *                ++j0; minsearch=0;
864 *                if (j0==DrN-1) column=1;
865 *                }
866 *             else
867 *                {
868 *                if (j0==0) dead_end=1;
869 *                if (j0>0 && j0<DrN-1)
870 *                   {
871 *                   more_drums=0; k0=dc[j0];
872 *                   do ++k0; while (col[k0].dr[j0].pres==0 && k0<=ColN);
873 *                   if (k0<=ColN) more_drums=1;
874 *                   if (more_drums)
875 *                      {
876 *                      minsearch=k0;
877 *                      if (j0>1) dc[j0]=-1;
878 *                      if (j0>0) --j0;
879 *                      i0=dc[j0]
880 *                      }
881 *                   else
882 *                      {
883 *                      do {
884 *                         if (j0>1) dc[j0]=-1;
885 *                         --j0; k0=dc[j0];
886 *                         do ++k0; while (col[k0].dr[j0].pres==0 &&
                                             k0<=ColN);
887 *                         if (k0<=ColN) more_drums=1;
888 *                         } while (!more_drums && j0>1);
889 *                      if (j0>1) dc[j0]=-1;
890 *                      if (j0>0) --j0;
891 *                      i0=dc[j0];
892 *                      if (more_drums) minsearch=k0;
893 *                      else minsearch=0;
894 *                      }
895 *                   if (j0<1)
896 *                      {
897 *                      i0=dc[0];
898 *                      k0=dc[1];
899 *                      do ++k0; while (col[k0].dr[1].pres==0 &&
                                          k0<=ColN);
900 *                      if (k0>ColN) dead_end=1;
901 *                      else minsearch=k0;
902 *                      }
903 *                   }
904 *                }
905 *             } while (j0<DrN-1 && !dead_end);
906 *          if (column==0)
907 *             {
908 *             do ++i; while (col[i].dr[0].pres==0 && i<=ColN);
909 *             minsearch=0;
910 *             dc[0]=i;
911 *             for (j0=1; j0<DrN; ++j0)
912 *                dc[j0]=-1;
913 *             }
914 *          i0=i; j0=0;
915 *          } while (column==0 && i<=ColN);
916 *    return(column);
917 *    }
918 *
```

```
 20  1 SURVO 84C EDITOR Sun Dec 13 19:27:12 1998          D:\COLMON\ 200 100 C
  1 *EXAMPLE:
  2 *ColDiamL=1.455        Lower diameter of the column between flutes
  3 *DiamVar=0.005         Range of lower diameters (plus and minus)
  4 *ColDiamU=1.15         Upper diameter of the column between flutes
  5 *ColH=8.97             Column height
  6 *MaxEnt=0.01           Maximum entasis
  7 *MaxEntH=4.49          Height where the maximum entasis is
  8 *ColNFr=6              Number of columns on front
  9 *ColNS=14              Number of columns on side
 10 *DrN=6                 Number of drums in one column
 11 *PresDr=50             Number of preserved drums
 12 *MinMarg=0.002         Minimum margin for measurements
 13 *MaxMarg=0.003         Maximum margin for measurements
 14 *Search=ALL            Place of possibly matching drums (ALL or number of
 15 *                      adjacent columns where to look for)
 16 *Mode=3                0 = Create, select, print and match
 17 *                      1 = Create, select and print
 18 *                      2 = Create and print
 19 *                      3 = Read drum data from the edit field and match
 20 *Zcoord=0              0 = No Printing of the drum Z coordinate
 21 *                      1 = Print the drum Z coordinate
 22 *Profile=1             0 = No printing of shaft profile coordinates
 23 *                      1 = Print the shaft profile coordinates
 24 *
 25 *SIMUL TEGEADR.CUR+1
 26 *
 27 *DATA TEGEADR
 28 *Col Dr   DiamL    MaN1  MaP1    DiamU    MaN2  MaP2  Height   MaN3  MaP3
 29 A  1  0   1.456 -0.004 0.004   1.412 -0.002 0.002   1.465 -0.009 0.009
 30 *  2  0   1.453 -0.004 0.004   1.421 -0.004 0.004   1.469 -0.007 0.005
 31 *  3  0   1.459 -0.004 0.003   1.420 -0.002 0.001   1.472 -0.003 0.003
 32 *  5  0   1.458 -0.003 0.003   1.422 -0.002 0.001   1.474 -0.002 0.002
 33 *  7  0   1.455 -0.003 0.003   1.416 -0.003 0.003   1.472 -0.003 0.002
 34 *  1  1   1.419 -0.001 0.002   1.373 -0.004 0.003   1.464 -0.001 0.002
 35 *  2  1   1.421 -0.002 0.002   1.380 -0.003 0.003   1.472 -0.003 0.002
 36 *  3  1   1.414 -0.002 0.002   1.376 -0.004 0.004   1.481 -0.004 0.004
 37 *  4  1   1.423 -0.004 0.004   1.375 -0.004 0.004   1.476 -0.005 0.005
 38 *  5  1   1.418 -0.002 0.002   1.370 -0.005 0.005   1.479 -0.004 0.004
 39 *  6  1   1.426 -0.003 0.003   1.379 -0.003 0.003   1.482 -0.003 0.003
 40 *  7  1   1.423 -0.003 0.003   1.377 -0.003 0.003   1.469 -0.003 0.003
 41 *  8  1   1.421 -0.003 0.003   1.377 -0.003 0.003   1.474 -0.003 0.003
 42 *  9  1   1.420 -0.003 0.003   1.370 -0.003 0.003   1.484 -0.003 0.002
 43 * 10  1   1.418 -0.002 0.002   1.376 -0.002 0.002   1.473 -0.003 0.002
 44 * 11  1   1.417 -0.003 0.003   1.374 -0.003 0.003   1.477 -0.002 0.002
 45 * 12  1   1.418 -0.002 0.002   1.377 -0.003 0.003   1.478 -0.002 0.002
 46 *  1  2   1.375 -0.003 0.003   1.322 -0.002 0.003   1.468 -0.004 0.003
 47 *  2  2   1.375 -0.003 0.003   1.337 -0.004 0.003   1.399 -0.005 0.005
 48 *  3  2   1.377 -0.002 0.001   1.332 -0.004 0.004   1.464 -0.003 0.002
 49 *  4  2   1.371 -0.003 0.003   1.322 -0.003 0.003   1.479 -0.004 0.003
 50 *  5  2   1.378 -0.003 0.003   1.325 -0.003 0.003   1.643 -0.001 0.001
 51 *  6  2   1.374 -0.003 0.003   1.328 -0.003 0.003   1.413 -0.003 0.002
 52 *  7  2   1.378 -0.002 0.002   1.323 -0.003 0.003   1.498 -0.002 0.002
 53 *  8  2   1.375 -0.003 0.003   1.338 -0.003 0.002   1.321 -0.003 0.002
 54 *  9  2   1.379 -0.002 0.001   1.329 -0.002 0.001   1.510 -0.002 0.002
 55 * 10  2   1.365 -0.005 0.005   1.332 -0.002 0.002   1.457 -0.003 0.002
 56 *  1  3   1.331 -0.003 0.002   1.270 -0.003 0.002   1.658 -0.003 0.002
 57 *  2  3   1.323 -0.002 0.003   1.267 -0.004 0.004   1.514 -0.002 0.002
 58 *  3  3   1.328 -0.002 0.001   1.280 -0.004 0.004   1.480 -0.002 0.001
 59 *  4  3   1.326 -0.004 0.003   1.269 -0.003 0.002   1.493 -0.002 0.002
 60 *  5  3   1.331 -0.002 0.002   1.271 -0.003 0.002   1.708 -0.002 0.001
 61 *  7  3   1.326 -0.003 0.003   1.274 -0.003 0.003   1.447 -0.003 0.003
 62 *  8  3   1.321 -0.003 0.003   1.268 -0.002 0.002   1.448 -0.002 0.002
 63 *  1  4   1.264 -0.004 0.004   1.212 -0.003 0.003   1.382 -0.010 0.010
 64 *  2  4   1.279 -0.002 0.002   1.216 -0.003 0.002   1.662 -0.002 0.001
 65 *  3  4   1.272 -0.003 0.002   1.210 -0.002 0.002   1.589 -0.002 0.001
 66 *  4  4   1.274 -0.003 0.003   1.216 -0.002 0.002   1.411 -0.003 0.003
 67 *  5  4   1.268 -0.003 0.002   1.212 -0.001 0.001   1.368 -0.001 0.001
 68 *  6  4   1.268 -0.003 0.003   1.218 -0.003 0.003   1.347 -0.003 0.003
 69 *  7  4   1.274 -0.002 0.002   1.223 -0.002 0.002   1.356 -0.002 0.002
 70 *  1  5   1.214 -0.003 0.003   1.151 -0.004 0.003   1.320 -0.003 0.003
```

```
 20  1 SURVO 84C EDITOR Sun Dec 13 19:28:09 1998          D:\COLMON\ 200 100 C
 71 *  2  5   1.215 -0.003 0.003   1.158 -0.003 0.003   1.331 -0.005 0.005
 72 *  3  5   1.209 -0.003 0.003   1.156 -0.002 0.001   1.479 -0.001 0.001
 73 *  4  5   1.206 -0.004 0.004   1.155 -0.003 0.003   1.349 -0.006 0.004
 74 *  5  5   1.220 -0.002 0.002   1.154 -0.003 0.003   1.500 -0.003 0.005
 75 B  6  5   1.215 -0.002 0.002   1.158 -0.003 0.003   1.484 -0.005 0.004
 76 *
 77 *FILE CREATE TEGEADR
 78 *FIELDS:
 79 *    1 NA_    1 CA         (##)
 80 *    2 NA_    1 DA         (##)
 81 *    3 NA_    4 XA         (## ###)
 82 *    4 NA_    4 YA         (##.###)
 83 *    5 NA_    1 CB         (##)
 84 *    6 NA_    1 DB         (##)
 85 *    7 NA_    4 XB         (##.###)
 86 *    8 NA_    4 YB         (##.###)
 87 *    9 NA_    1 CC         (##)
 88 *   10 NA_    1 DC         (##)
 89 *   11 NA_    4 XC         (##.###)
 90 *   12 NA_    4 YC         (##.###)
 91 *   13 NA_    1 CD         (##)
 92 *   14 NA_    1 DD         (##)
 93 *   15 NA_    4 XD         (##.###)
 94 *   16 NA_    4 YD         (##.###)
 95 *   17 NA_    1 CE         (##)
 96 *   18 NA_    1 DE         (##)
 97 *   19 NA_    4 XE         (##.###)
 98 *   20 NA_    4 YE         (##.###)
 99 *   21 NA_    1 CF         (##)
100 *   22 NA_    1 DF         (##)
101 *   23 NA_    4 XF         (## ###)
102 *   24 NA_    4 Height     (##.###)
103 *   25 NA_    4 NegMarg    (## ###)
104 *   26 NA_    4 PosMarg    (#.###)
105 *   27 NA_    4 Hmin       (##.###)
106 *   28 NA_    4 Hmax       (##.###)
107 *END
108 *
109 *FILE SAVE TEGEADR.TXT,TEGEADR.SVO
110 *FIELDS
111 * 1  CA          ,
112 * 2  DA          ,
113 * 3  XA          ,
114 * 4  YA          ,
115 * 5  CB          ,
116 * 6  DB          ,
117 * 7  XB          ,
118 * 8  YB          ,
119 * 9  CC          ,
120 *10  DC          ,
121 *11  XC          ,
122 *12  YC          ,
123 *13  CD          ,
124 *14  DD          ,
125 *15  XD          ,
126 *16  YD          ,
127 *17  CE          ,
128 *18  DE          ,
129 *19  XE          ,
130 *20  YE          ,
131 *21  CF          ,
132 *22  DF          ,
133 *23  XF          ,
134 *24  Height      ,
135 *25  NegMarg     ,
136 *26  PosMarg    LF
137 *END
138 *
139 *FILE SHOW TEGEADR
```

2.B. Acceptable Shaft Profiles and Maximum Entasis

```
  1  1 SURVO 84C EDITOR Sun Dec 13 19:35:10 1998          D:\COLMON\ 400 100 C
  1 *SHAFT-MAXENT.TUT
  2 *
  3 *tutload shaft-maxent
  4 / Sucro shaft-maxent.tut by Jari Pakkanen (Mar 31 1997)
  5 / for finding the place of maximum entasis in a column shaft
  6 *{tempo -1}{init}
  7 - if W1 '<>' ? then goto A
  8 *{line start}{d}{erase}{erase}Activating sucro{R}
  9 *{erase}{erase}/SHAFT-MAXENT <data>,<ID1>,<ID2>,<emin>,<emax>,<ehmin>,
 10 *<ehmax><acc>{R}
 11 *{erase}{erase}determines whether shaft profile fits into the defined {}
 12 *area.{R}
 13 *{erase}{erase}ID1 is the lower limit of the shaft ID, ID2 the upper. {}
 14 *emin and emax{R}
 15 *{erase}{erase}are the centre of the minimum and maximum entases in m.
 16 * ehmin and ehmax{R}
 17 *{erase}{erase}give the proportional height of the minimum and maximum
 18 * entasis and {R}
 19 *{erase}{erase}acc defines the width of the area in m. The number of f
 20 *itting shaft {R}
 21 *{erase}{erase}profiles for each case are stored in data file SHAFTFIT
 22 * SVD.
 23 / def Wdata=W1 Wid1=W2 Wid2=W3 Wentmin=W4 Wentmax=W5 Wenthmin=W6
 24 / def Wenthmax=W7 Waccu=W8 Whelp1=W9 Wi=W10 Wj=W11 Wlin=W12 Wcol=W13
 25 /
 26 + A: {save cursor Wlin,Wcol}{R}
 27 *@SCRATCH /{act}{home}FILE CREATE SHAPTFIT{R}
 28 *FIELD$:{R}
 29 *   1 NA_    4 EntH      (#.##){R}
 30 *   2 NA_    4 MaxEnt    (#.###){R}
 31 *   3 NA_    4 N         (#####){R}
 32 *END{R}
 33 *{u6}{act}{Wj=Wenthmin}
 34 + MainLoop: {jump Wlin,Wlin,1,1}{R}
 35 *■SCRATCH /{act}{home}{Wi=Wentmin}
 36 /
 37 / Starting loop:
 38 + Loop: {jump Wlin,Wlin,1,1}{R}
 39 *■SCRATCH /{act}{home}
 40 / Calling SHAFT-CURVE.TUT
 41 *{save stack helpstck}/SHAFT-CURVE {print Wdata},{print Wid1},
 42 *{print Wid2},{Whelp1=Wi+Waccu}{print Whelp1},{Whelp1=Wi-Waccu}
 43 *{print Whelp1},{print Wj},{print Waccu}{tempo +1}{act}{tempo -1}
 44 *{load stack helpstck}{R}
 45 *■SCRATCH /{act}{home}IND=OK,1{R}
 46 *STAT {print Wdata},CUR+1 / VARS=Height{act}{R}
 47 *{find =}{r}{save line Whelp1}{jump Wlin,Wlin,1,1}{R}
 48 *■SCRATCH /{act}{home}DATA FITTING{R}
 49 *EntH MaxEnt N{R}
 50 *{print Wj} {print Wi} {print Whelp1}{R}
 51 *{d}VAR OK=0 TO {print Wdata}{act}{home}{erase}SAVEP C:\R\RESULTS{act}
 52 *{home}{erase}FILE COPY FITTING,SHAFTFIT{act}
 53 /
 54 *{Wi=Wi+0.001}
 55 - if Wi <= Wentmax then goto Loop
 56 *{Wj=Wj+0.01}
 57 - if Wj <= Wenthmax then goto MainLoop
 58 + End: {jump Wlin,Wlin,1,1}{tempo +1}{end}
 59 *
 60 *
 61 *SHAFT-CURVE.TUT
 62 *
 63 *tutload shaft-curve
 64 / Sucro shaft-curve.tut by Jari Pakkanen (Mar 31 1997)
 65 / for determining whether shaft profile fits into the defined area
 66 *{tempo -1}{init}
 67 - if W1 '<>' ? then goto A
```

```
  1  1 SUHVO 84C EDITOR Sun Dec 13 19:36:31 1998          D:\COLMON\ 400 100 C
 67 - if W1 '<>' ? then goto A
 68 *|line start||d||erase||erase|Activating sucro{R}
 69 *|erase||erase|/SHAFT-CURVE <data>,<ID1>,<ID2>,<maxent>,<minent>,<anth
 70 *>,<accuracy>{R}
 71 *|erase||erase|determines whether shaft profile fits into the defined ||
 72 *area.|R|
 73 *|erase||erase|ID1 is the lower limit of the shaft ID, ID2 the upper, ||
 74 *maxent and |R|
 75 *|erase||erase|minent are the maximum and minimum entases in m. anth g
 76 *ives the |R|
 77 *|erase||erase|proportional height of the maximum entasis and accuracy
 78 * in m defines |R|
 79 *|erase||erase|the width of the area (plus-minus at the top and bottom
 80 *). |R|
 81 *|goto End|
 82 / def Wdata=W1 Wid1=W2 Wid2=W3 Wmaxent=W4 Wminent=W5 Wmaxenth=W6
 83 / def Waccu=W7 Wcold1=W8 Wcoldu=W9 Wcolh=W10 Whelp1=W11 Wxtop=W12
 84 / def Wm=W13 Whord=W14 Wverd=W15 Wxent=W16 Wlin=W17 Wenth=W18 Wal=W19
 85 / def Wb1=W20 Wc1=W21 Wlin2=W22 Wlin3=W23 Wa2=W24 Wb2=W25 Wc2=W26
 86 / def Wc2=W27 Wid=W28 Wx=W29 Wy=W30 Wok=W31 Wy1=W32 Wy2=W33
 87 + A: |R|
 88 *■SCRATCH /|act||home| |copy||R|
 89 *|R|
 90 *|save cursor Wlin,Whelp1||Wid=Wid1|
 91 + Loop: |jump Wlin.Wlin.1.1||erase|INU=Nro.|print Wid|{R}
 92 *MASK=--AA--AA--AA--AA--AA--AA----------------|R|
 93 *FILE LOAD |print Wdata||act||R|
 94 *|d2||save cursor Wlin2,Whelp1||line end||1||save word Wcolh||18|
 95 *|save word Wxtop|
 96 /
 97 / Determining the coordinates of maximum entasis:
 98 / 1. Slope Wm of the straight line from bottom of the column to the top:
 99 *|Wm=Wcolh/Wxtop|
100 /
101 / 2. Horizontal distance from maxent-point to straight line
102 *|Whelp1=Wm*Wm||Whelp1=1/Whelp1||Whelp1=Whelp1+1|
103 *|Whord=Wmaxent/Whelp1|
104 /
105 / 3. Vertical distance from maxent-point to straight line:
106 *|Whelp1=1/Wm||Wverd=Whelp1*Whord|
107 /
108 / 4  X-coordinate of the maxent-point:
109 *|Wenth=Wmaxenth*Wcolh||Whelp1=Wenth-Wverd||Whelp1=Whelp1/Wm|
110 *|Wxent=Whelp1-Whord|
111 /
112 / Calling LSQMAT.EXE:
113 *|R|
114 *|line start|.|copy||R|
115 *|R|
116 *|save cursor Wlin3,Whelp1|@SCRATCH /|act||home|DATA $HAFT.|R|
117 *   -|write Waccu|   0|R|
118 *   -|write Waccu|   0|R|
119 *   |write Wxent|  |write Wenth||R|
120 *   |Whelp1=Wxtop-Waccu||write Whelp1|  |write Wcolh||R|
121 *   |write Whelp1|  |write Wcolh|  END|R|
122 *|d|LSQMAT SHAFT,2,CUR+1|act||R|
123 *|d12|MAT SAVE A|act||R|
124 *MAT SAVE B|act||R|
125 *MAT SOLVE X FROM A*X=B|act||R|
126 *MAT LOAD X,CUR+1|act||R|
127 *|d3||next word||save word Wa1||R|
128 *|next word||save word Wb1||R|
129 *|next word||save word Wc1|
130 /
131 / Determining the coordinates of minimum entasis:
132 / 2. Horizontal distance from minent-point to straight line:
133 *|Whelp1=Wm*Wm||Whelp1=1/Whelp1||Whelp1=Whelp1+1|
134 *|Whord=Wminent/Whelp1|
135 /
136 / 3. Vertical distance from minent-point to straight line:
137 *|Whelp1=1/Wm||Wverd=Whelp1*Whord|
138 /
139 / 4  X-coordinate of the minent-point:
```

```
  1  1 SURVO 84C EDITOR Sun Dec 13 19:37:30 1998        D:\COLMON\ 400 100 C
140 *{Wenth=Wmaxenth*Wcolh}{Whelp1=Wenth-Wverd}{Whelp1=Whelp1/Wm}
141 *{Wxent=Whelp1-Whord}
142 /
143 / Calling LSQMAT.EXE
144 *{jump Wlin3,Wlin3,1,1}@SCRATCH /{act}{home}DATA SHAFT:{R}
145 *   {write Waccu}   0{R}
146 *   {write Waccu}   0{R}
147 *   {write Wxent}  {write Wenth}{R}
148 *   {Whelp1=Wxtop+Waccu}{write Whelp1}  {write Wcolh}{R}
149 *   {write Whelp1}  {write Wcolh}  END{R}
150 *{d}LSQMAT SHAFT,2,CUR+1{act}{R}
151 *{d12}MAT SAVE A{act}{R}
152 *MAT SAVE B{act}{R}
153 *MAT SOLVE X FROM A*X=B{act}{R}
154 *MAT LOAD X,CUR+1{act}{R}
155 *{d3}{next word}{save word Wa2}{R}
156 *{next word}{save word Wb2}{R}
157 *{next word}{save word Wc2}{R}
158 *.{copy}{R}
159 *{R}
160 *{Wok=0}
161 + Data: {jump Wlin2,Wlin2,1,1}{next word}{save word Wx}{next word}
162 *{save word Wy}{home}{del14}
163 - if Wx '=' Wxtop then goto Check
164 *{pre}{d}{pre}{d}({write Wa1}}+({write Wb1}}*{write Wx}+{{write Wc1}}*
165 *{write Wx}^2={act}{save line Wy1}{home}{erase}{{write Wa2}}+{
166 *{write Wb2}}*{write Wx}+{{write Wc2}}*{write Wx}^2={act}
167 *{save line Wy2}{home}{erase}
168 - if Wy > Wy1 then goto Copy
169   if Wy < Wy2 then goto Copy
170 *{goto Data}
171 + Check: {pre}{d}{pre}{d}({write Wa1}}+({write Wb1}}*{write Wx}+{
172 *{write Wc1}}*{write Wx}^2={act}{save line Wy1}{home}{erase}{
173 *{write Wa2}}+({write Wb2}}*{write Wx}+({write Wc2}}*{write Wx}^2=
174 *{act}{save line Wy2}{home}{erase}
175   if Wy > Wy1 then goto Copy
176 - if Wy < Wy2 then goto Copy
177 *{Wok=1}
178 + Copy: FILE COPY APU1,{print Wdata}{R}
179 *MATCH=Nro{R}
180 *DATA APU1{R}
181 *Nro OK a1 b1 c1 a2 b2 c2{R}
182 *{write Wid} {write Wok} {write Wa1} {write Wb1} {write Wc1} {}
183 *{write Wa2} {write Wb2} {write Wc2}{R}
184 *{u5}{act}{Wid=Wid+1}
185 - if Wid > Wid2 then goto Jump
186 *{goto Loop}
187 + Jump: {jump Wlin,Wlin,1,1}{u2}
188 + End: {tempo +1}{end}
189 *
190 *
191 *!LSQMAT.EXE
192 *
193 *loadp c:\c6\!lsqmat.c
194 */* 'LSQMAT.C 29.5.1995/Jari Pakkanen */
195 *
196 *#include <stdio.h>
197 *#include <stdlib.h>
198 *#include <conio.h>
199 *#include <malloc.h>
200 *#include <math.h>
201 *#include "survo.h"
202 *#include "survoext.h"
203 *#include "survodat.h"
204 *
205 *#define MAX 50        /* Maximum number of coordinates                 */
206 *#define DEG 4         /* Max size of matrix (for 3rd degree function) */
207 *
208 *SURVO_DATA d;
209 *
210 *double XC[MAX];       /* X coordinate data                             */
211 *double YC[MAX];       /* Y coordinate data                             */
```

```
  1  1 SUNVO 84C EDITOR Sun Dec 13 19:38:12 1998          D:\COLMON\ 400 100 C
212 *double MA[DEG][DEG]; /* Matrix A                                        */
213 *double MB[DEG];      /* Matrix B                                        */
214 *double SumX;         /* Sum of X's                                      */
215 *double SumX2;        /* Sum of X's^2                                    */
216 *double SumX3;        /* Sum of X's^3                                    */
217 *double SumX4;        /* Sum of X's^4                                    */
218 *double SumX5;        /* Sum of X's^5                                    */
219 *double SumX6;        /* Sum of X's^6                                    */
220 *double SumY;         /* Sum of Y's                                      */
221 *double SumXY;        /* Sum of X*Y's                                    */
222 *double SumX2Y;       /* Sum of X^2*Y's                                  */
223 *double SumX3Y;       /* Sum of X^3*Y's                                  */
224 *int i,j,degree,results_line
225 *char line[LLENGTH];
226 *char elem[32];
227 *
228 *main(argc,argv)
229 *int argc; char *argv[];
230 *    {
231 *    if (argc==1) return;
232 *    s_init(argv[1]);
233 *    if (g<3)
234 *        {
235 *        sur_print("\nUsage: LSQMAT <data>,<degree>,<output_line>");
236 *        WAIT; return;
237 *        }
238 *    results_line=0;
239 *    if (g>3)
240 *         {
241 *         results_line=edline2(word[3],1,1);
242 *         if (results_line==0) return;
243 *         }
244 *    i=data_open(word[1],&d); if (i<0) return;
245 *    i=mask(&d); if (i<0) return;
246 *    j=0;
247 *    for (i=0; i<d.n; i+=2)
248 *        {
249 *        data_load(&d,d.ll,i,&XC[j]);
250 *        data_load(&d,d.ll,i+1,&YC[j]);
251 *        ++j;
252 *        }
253 *
254 *    /* CALCULATING THE LEAST-SQUARE MATRICES */
255 *
256 *    /* 2nd and 3rd degree functions */
257 *
258 *    SumX=0; SumX2=0; SumX3=0; SumX4=0; SumX5=0; SumX6=0;
259 *    SumY=0; SumXY=0; SumX2Y=0; SumX3Y=0;
260 *    for (i=0; i<=d.n; ++i)
261 *        {
262 *        SumX=SumX+XC[i];
263 *        SumX2=SumX2+XC[i]*XC[i];
264 *        SumX3=SumX3+XC[i]*XC[i]*XC[i];
265 *        SumX4=SumX4+XC[i]*XC[i]*XC[i]*XC[i];
266 *        SumY=SumY+YC[i];
267 *        SumXY=SumXY+XC[i]*YC[i];
268 *        SumX2Y=SumX2Y+XC[i]*XC[i]*YC[i];
269 *        }
270 *    degreeauto(word[2]);
271 *    if (degree==3)  /* 3rd degree function */
272 *        for (i=0; i<=d.n; ++i)
273 *            {
274 *            SumX5=SumX5+XC[i]*XC[i]*XC[i]*XC[i]*XC[i];
275 *            SumX6=SumX6+XC[i]*XC[i]*XC[i]*XC[i]*XC[i]*XC[i];
276 *            SumX3Y=SumX3Y+XC[i]*XC[i]*XC[i]*YC[i];
277 *            }
278 *    MA[0][0]=d.n/2;
279 *    MA[0][1]=SumX; MA[1][0]=SumX;
280 *    MA[0][2]=SumX2; MA[1][1]=SumX2; MA[2][0]=SumX2;
281 *    MA[1][2]=SumX3; MA[2][1]=SumX3;
282 *    MA[2][2]=SumX4;
283 *    MB[0]=SumY;
```

```
  1   1 SURVO 84C EDITOR Sun Dec 13 19:39:16 1998          D:\COLMON\ 400 100 C
284 *     MB[1]=SumXY;
285 *     MB[2]=SumX2Y;
286 *     if (degree==3)  /* 3rd degree function */
287 *         {
288 *         MA[0][3]=SumX3; MA[3][0]=SumX3;
289 *         MA[1][3]=SumX4; MA[3][1]=SumX4;
290 *         MA[2][3]=SumX5; MA[3][2]=SumX5;
291 *         MA[3][3]=SumX6;
292 *         MB[3]=SumX3Y;
293 *         }
294 *     data_close(&d);
295 *
296 *     /* OUTPUT OF MATRICES TO EDIT FIELD */
297 *
298 *     output_open(eout);
299 *     strcpy(line,"MATRIX A");
300 *     print_line(line);
301 *     if (degree==2)
302 *         strcpy(line,"///  0            1            2");
303 *     else
304 *         strcpy(line,"///  0            1            2            3");
305 *     print_line(line);
306 *     if (degree==2)
307 *         for (i=0; i<=2; ++i)
308 *             {
309 *             sprintf(line,"%d  ",i);
310 *             for (j=0; j<=2; ++j)
311 *                 {
312 *                 fnconv(MA[i][j],accuracy+6,elem);
313 *                 strncat(line,elem,accuracy+6);
314 *                 }
315 *             print_line(line);
316 *             }
317 *     else
318 *         for (i=0; i<=3; ++i)
319 *             {
320 *             sprintf(line,"%d  ",i);
321 *             for (j=0; j<=3; ++j)
322 *                 {
323 *                 fnconv(MA[i][j],accuracy+6,elem);
324 *                 strncat(line,elem,accuracy+6);
325 *                 }
326 *             print_line(line);
327 *             }
328 *     strcpy(line," ");
329 *     print_line(line);
330 *     strcpy(line,"MATRIX B");
331 *     print_line(line);
332 *     strcpy(line,"///  0");
333 *     print_line(line);
334 *     for (i=0; i<=2; ++i)
335 *         {
336 *         sprintf(line,"%d  ",i);
337 *         fnconv(MB[i],accuracy+6,elem);
338 *         strncat(line,elem,accuracy+6);
339 *         print_line(line);
340 *         }
341 *     if (degree==3)  /* 3rd degree function */
342 *         {
343 *         strcpy(line,"3  ");
344 *         fnconv(MB[3],accuracy+6,elem);
345 *         strncat(line,elem,accuracy+6);
346 *         print_line(line);
347 *         }
348 *     strcpy(line," ");
349 *     print_line(line);
350 *     output_close(eout);
351 *     }
352 *
353 *print_line(line)
354 *char *line;
355 *     {
```

```
  1   1 SURVO 84C EDITOR Sun Dec 13 19:40:49 1998         D:\COMMON\ 400 100 C
356 *     output_line(line,sout,results_line)'
357 *     if (results_line) ++results_line.
358 *     )
359 *
360 *
361 *EXAMPLE.
362 *TEGEADR2 SVO. MaxEnt=0.009-0.013. EntH=0.40-0.60
363 *
364 */SHAFT-MAXENT tegeadr2,1,1678,0.009,0.013,0.40,0.60,0.0015
365 *
```

Index