A Microcomputer Program for the Calculation of a Trawlnet Section Taper

DAVID K. MARTIN and CONRAD W. RECKSIEK

Introduction

In this paper we describe a computer program which enables the user to find the correct taper of a trawlnet belly or extension section, given the dimensions, in meshes, of wide end, narrow end, and depth.

Our purpose in this report is fourfold:

1) To illustrate how general principles of web shaping and assembly, as described in the companion article, "Shaping and assembling webbing" (Recksiek, 1983), can be expressed as functional computer program algorithms;

2) to describe the program logic, through the flowchart medium, so the readers, who may so desire, can create their own code in the computer language of their choice according to their particular hardware constraints;

3) to discuss the logic in a general context so the readers may expand the program described for other requirements, e.g., reckoning wing section tapers; and

4) to report in tabular format, actual code in a version of BASIC programming language used in an inexpensive microcomputer system.

We will first describe the microcomputer system we used. We will then illustrate the use of our program by showing how some exemplary problems are solved with the system. The figures consist of flowcharts of the main program and its subroutines, which are outlined in an accompanying text discussion. Finally, we present the actual program code as Table 1.

The notation and specialized terms used in this paper will be the same as those of Recksiek (1983). (Variables in this paper are always written in upper case since the system we used cannot display lower case characters.)

David K. Martin is with the College of Resource Development, University of Rhode Island, Kingston, RI 02881, and Conrad W. Recksiek is Associate Professor, Department of Fisheries, Aquaculture, and Pathology, University of Rhode Island, Kingston, RI 02881.

The Microcomputer System

We chose a readily available, low-cost microcomputing system which, exclusive of the television monitor, can be purchased for under $200. We used a Timex-Sinclair TS 1000[1] having 1 K bytes of random access memory (RAM). Plugged into the small (17×17 cm) keyboard/computing hardware package is an expandable memory of 16 K bytes. (The expandable 16 K RAM pack is required for the application being described in this report.) An inexpensive cassette tape recorder is interfaced with the machine and the programs are stored on cassette tape. The system's television monitor can display 31 columns by 21 rows of characters.

Program Function: Some Examples

After turning on the machine and loading the program, the user is prompted for the dimensions of the piece of webbing. When these have been entered, the machine performs the various calculations. Then, selected variables along with the tapering sequence are displayed on the monitor screen. The first example is taken from Figure 8(a) of "Shaping and assembling webbing" (Recksiek, 1983):

ENTER WIDE END	prompt line.
16	user entered 16; screen clears and,
ENTER NARROW END	second prompt line.
8	user entered 8; screen clears and,
ENTER DEPTH	third prompt line.
9	user entered 9; screen clears and,
WE=16 NE=8 DEPTH=9	output lines begin....

TAPER IS STEPS/POINTS

TSTEPS=13 P=5 S=3 R=3 U=2 (RPRI=3 UPRI=2 F=1)

TAPER EQUAL TO: 3 2 3 2 3	final output line.

The next example is from Figure

[1]Mention of trade names or commercial firms does not imply endorsement by the National Marine Fisheries Service, NOAA.

8(b) of Recksiek (1983). Here, note that the user has input 10 for the depth, N. The user also has input T = 15 and K = 8. A belly section of these dimensions cannot be symmetric, i.e., the taper cannot be the same on both sides. The program executes a test for symmetry. We structured the program to revise the input depth in this circumstance so that dimensions of a symmetric piece could be displayed:

ENTER WIDE END	prompt line.
15	user entered 15; screen clears and,
ENTER NARROW END	second prompt line.
8	user entered 8; screen clears and,
ENTER DEPTH	third prompt line.
10	user entered 10; screen clears and,
WE=15 NE=8 DEPTH=9.5	output lines begin showing revised DEPTH;
SUBTRACTED 1/2 MESH FROM DEPTH	message signaling change . . . .

Table 1.— BASIC symbolic language code for computer program designed to generate a number sequence which represents a webbing taper for a trapezoidal section.

```
1000 REM "BELLY"
1005 REM MAIN PROGRAM FOR SQUARES, BELLIES
     AND EXTENSIONS
1010 REM
1020 PRINT AT 10,8; "ENTER WIDE END"
1030 INPUT T
1035 REM "CLS" MEANS CLEAR SCREEN ONLY
1040 CLS
1050 PRINT AT 10,7; "ENTER NARROW END"
1060 INPUT K
1070 CLS
1080 PRINT AT 10,10; "ENTER DEPTH"
1090 INPUT N
1100 CLS
1110 LET M = ((T - K)/2) + 1
1120 IF M = INT M THEN GO TO 1150
1130 IF N = INT N THEN GO TO 1160
1140 GO TO 1180
1150 IF N = INT N THEN GO TO 1180
1160 LET N = N - (0.5)
1170 PRINT AT 3,1; "SUBTRACTED 1/2 MESH FROM
     DEPTH"
1180 PRINT AT 1,2; "WE="; T; "NE="; K;
     "DEPTH="; N
1190 IF N = M THEN GO TO 1250
1200 IF N > M THEN GO TO 1270
1210 PRINT AT 5,5; "TAPER IS STEPS/MESHES"
1220 LET M2 = N
1230 LET N2 = M
1240 GO TO 1300
1250 PRINT AT 5,4; "TAPER IS A STRAIGHT BAR"
1260 GO TO 1280
1270 PRINT AT 5,5; "TAPER IS STEPS/POINTS"
1280 LET M2 = M
1290 LET N2 = N
1300 LET TSTEPS = N2 + M2 - 1
1310 LET P = N2 - M2 + 1
1320 LET S = (INT(TSTEPS/P)) + 1
1330 LET R = TSTEPS - ((S - 1) * P)
1340 LET U = P - R
1350 PRINT AT 7,1; "TSTEPS ="; TSTEPS; "P ="; P;
     "S ="; S; "R ="; R; "U ="; U
1360 LET AVALUE = S
1370 LET BVALUE = S - 1
1371 REM
1372 REM CALL RENAME SUBROUTINE
1373 REM
1380 GOSUB 4000
1390 LET UPRIME = U
1400 LET RPRIME = R
1401 REM
1402 REM CALL FACTOR SUBROUTINE
1403 REM
1410 GOSUB 5000
1420 PRINT AT 8,4; "(RPRI = "; RPRIME; "UPRI =";
     UPRIME; "F ="; F; ")"
1421 REM
1422 REM CALL ORDER SUBROUTINE
1423 REM
1430 GOSUB 6000
1435 REM CODE FROM HERE TO STATEMENT 1550 IS
     USED TO OUTPUT ARRAY S
1440 PRINT AT 13,0; "TAPER EQUAL TO:"
1450 LET H = 1
1460 LET V = 14
1470 FOR L = 1 TO KF
1480 PRINT AT V, H; SEQ(L)
1490 LET H = H + 2
1500 IF H = 31 THEN GO TO 1520
1510 GO TO 1540
1520 LET H = 1
1530 LET V = V + 1
1540 NEXT L
1550 STOP
4000 REM
4002 REM SUBROUTINE RENAME
4004 REM
4010 IF U < R THEN GO TO 4080
4020 LET X = R
4030 LET R = U
4040 LET U = X
4050 LET X = AVALUE
4060 LET AVALUE = BVALUE
4070 LET BVALUE = X
4080 RETURN
4090 REM
5000 REM SUBROUTINE FACTOR
5001 REM
5010 LET F = 1
5020 FOR I = 2 TO U
5030 IF UPRIME <= 1 THEN GO TO 5110
5040 IF (UPRIME/I) - INT(UPRIME/I) >< 0 THEN GO
     TO 5100
5050 IF (RPRIME/I) - INT(RPRIME/I) >< 0 THEN GO
     TO 5100
5060 LET UPRIME = UPRIME/I
5070 LET RPRIME = RPRIME/I
5080 LET F = F * I
5090 GO TO 5030
5100 NEXT I
5110 RETURN
5120 REM
6000 REM SUBROUTINE ORDER
6001 REM
6010 DIM A(200)
6020 DIM SEQ(300)
6030 LET INDEX = 0
6040 LET MF = UPRIME/RPRIME
6050 LET LASTR = RPRIME - 1
6060 FOR I = 1 TO LASTR
6070 IF INT(MF*I) = INT (MF*(I-1)) THEN GO TO
     6100
6080 LET INDEX = INDEX + 1
6090 LET A(INDEX) = BVALUE
6100 LET INDEX = INDEX + 1
6110 LET A(INDEX) = AVALUE
6120 NEXT I
6130 IF UPRIME = 0 THEN GO TO 6160
6140 LET INDEX = INDEX + 1
6150 LET A(INDEX) = BVALUE
6160 LET INDEX = INDEX + 1
6170 LET A(INDEX) = AVALUE
6180 LET KF = 0
6190 FOR I = 1 TO F
6200 FOR J = 1 TO INDEX
6210 LET KF = KF + 1
6220 LET SEQ (KF) = A (J)
6230 NEXT J
6240 NEXT I
6250 RETURN
```

TAPER IS STEPS/POINTS

TSTEPS=13 P=6 S=3 R=1 U=5
(RPRI=1 UPRI=5 F=1)

TAPER EQUAL TO: 2 2 2 2 3 2	final output line.

The program has the capability to deal with pieces having all bar edges or jib cut edges. The following example illustrates output for a piece having an all bar edge. The input lines have been omitted for the sake of brevity:

WE=18 NE=7 DEPTH=6.5	output lines begin . . . .

TAPER IS A STRAIGHT BAR

TSTEPS=12 P=1 S=13 R=0 U=1
(RPRI=0 UPRI=1 F=1)

TAPER EQUAL TO: 12	final output line.

In the next example, the program

identifies the taper as being a jib cut and the message, TAPER IS STEPS/MESHES, is displayed:

WE=18 NE=4 DEPTH=6 — output lines begin....

TAPER IS STEPS/MESHES

TSTEPS=13 P=3 S=5 R=1 U=2 (RPRI=1 UPRI=2 F=1)

TAPER EQUAL TO: 4 5 4 — final output line.

Program Logic: A Flowchart

The logic used in the microcomputer program is portrayed in the figures. An expression of this logic in BASIC language code is presented in Table 1. This program code, when loaded into a Timex-Sinclair TS 1000 microcomputer system, produces the monitor displays described in the previous section.

Our intention in this section is to explain the flowchart so the prospective users can adapt the logic to serve their own ends. Basically, the logic parallels the theoretical development of Recksiek (1983). Readers are encouraged to thoroughly familiarize themselves with that article, particularly the introductory sections and the section, "Squares, Bellies, and Extensions."

As a matter of interest, we would like to mention that our preliminary coding of the flowcharted logic was accomplished in FORTRAN symbolic programming language on a mainframe computer. Our goal in that work, which is presently in progress, is to use that system's pen plotter to draw finished net sections. For the purpose of the application described in this article, we wrote the BASIC code directly from our FORTRAN code listing.

The program is structured as a main program module with three subroutines. We developed this structure with the potential user/programmer in mind. The main program can be easily modified to do other tasks, e.g., calculating wing tapers. This particular main program does the simple arithmetic of the "belly top formula" (equation (11.1) of Recksiek, 1983) and the general tapering equations.

The subroutines perform tasks which must be done by any program doing tapering calculations. These subroutines are called at critical points in the main program. In the following sections of this article, we summarize the functions of the various program modules.

Main Program

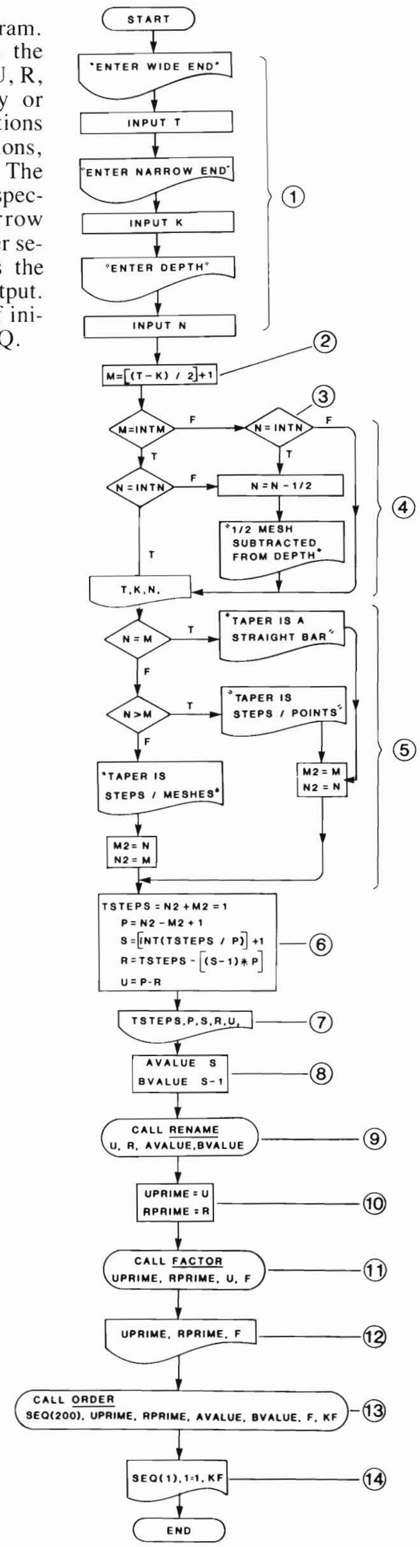
The main program calculates the various tapering values U, R, S, F for trawlnet belly or extension pieces upon input of the dimensions. The latter, expressed in numbers of meshes, are, respectively, N, K, T for "depth, narrow end, wide end." A number sequence which represents the taper is generated and displayed as output. This sequence consists of initial elements of an array, SEQ. The flowchart of this routine is presented in Figure 1. Referring to that figure, the following features are of interest:

1) Prompts and inputs of T, K, N.

2) Calculate horizontal mesh distance M as a function of T and K. This is equation (11.1), the "belly top formula" of Recksiek (1983).

3) Test of N being a whole number. (Other programming languages differentiating between real and integer variables may require some additional arithmetic.)

Figure 1. — Main program. This program calculates the various tapering values, U, R, S, F for trawlnet belly or extension-shaped net sections upon input of the dimensions, in numbers of meshes. The latter are N, K, T or, respectively, the "depth, narrow end, wide end." A number sequence which represents the taper is displayed as output. This sequence consists of initial elements of array SEQ.



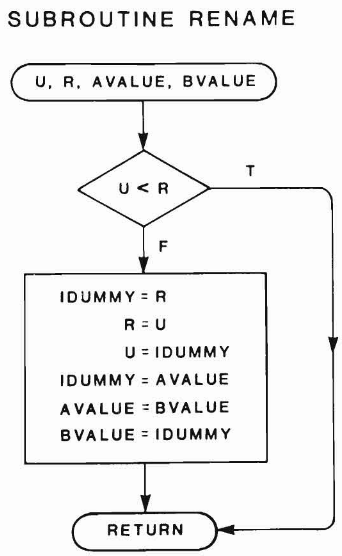


Figure 2. — Subroutine RENAME. Arguments: U, R, AVALUE, BVALUE. The subroutines FACTOR and ORDER require that arguments derived from U and R must be ordered one greater than or equal to the other. If the arguments are incorrectly ordered, the subroutine reverses their values.

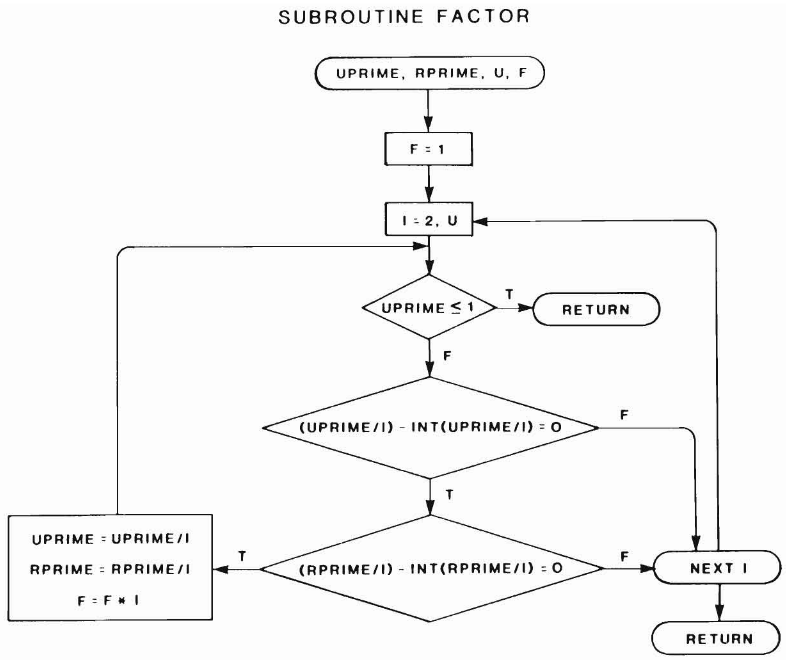


Figure 3.—Subroutine FACTOR. Arguments: UPRIME, RPRIME, U, F. This subroutine determines F, RPRIME, and UPRIME such that U = F × UPRIME and R = F × RPRIME. That is, F is the largest common factor of U and R.

4) Test of symmetry. If M contains a half mesh, then N must contain a half mesh. If M has no half mesh, then N must contain no half mesh. In other words, M and N must both be either whole numbers or both whole numbers plus a half mesh. In this logic, if the condition (both whole or both not whole) cannot be met, N is "adjusted" by subtraction of 0.5. As was illustrated earlier in the second problem example, the user is informed of this having taken place. This is a test of symmetry in that the tapers on both side edges of the piece must be the same.

5) The taper is a straight bar when N = M; it is a body cut, designated as "STEPS/POINTS," when N > M; it is a jib cut, designated as "STEPS/MESHES," when N < M.

6) Fundamental arithmetic. Calculation of TSTEPS (for total steps), P, S, R, U based upon principles embodied in equations (1.3) through (6.6) of Recksiek (1983).

7) Display of variables and calculated values.

8) AVALUE and BVALUE defined. These are arguments used in subroutines RENAME and ORDER. The subroutines FACTOR and ORDER require that arguments derived from U and R must be ordered one greater than or equal to the other. (The true values of U, R, and S are no longer of interest to the user by the time subroutine RENAME is called; hence they can be renamed to satisfy order requirements of the subroutines.)

9) Subroutine RENAME checks values of its arguments for correct relative size. Values are reversed, or "renamed," if necessary.

10) Arguments for subroutine FACTOR are defined.

11) Subroutine FACTOR determines F such that F is a factor common to U and R and that U = F×UPRIME and R = F×RPRIME.

12) Display of calculated values.

13) Subroutine ORDER determines the actual number sequence of the taper. Argument KF is the total number of nonzero elements of array SEQ, i.e., there are KF numbers in the sequence.

14) Display of sequence array SEQ.

Subroutine RENAME

Arguments: U, R, AVALUE, BVALUE. The subroutines FACTOR and ORDER require that arguments derived from U and R must be ordered one greater than or equal to the other. If the arguments are incorrectly ordered, this subroutine reverses their values (Fig. 2).

Subroutine FACTOR

Arguments: UPRIME, RPRIME, U, F. This subroutine determines F, RPRIME, and UPRIME such that U = F×UPRIME and R = F×RPRIME. That is, F is the largest common factor of U and R (Fig. 3).

Subroutine ORDER

Arguments: SEQ, UPRIME, RPRIME, AVALUE, BVALUE, F, KF. This subroutine generates KF nonzero elements of array SEQ such that each element represents one number in a tapering sequence. The reader is referred to Recksiek (1983) where equations (10.1) through (10.5) are discussed. That discussion includes a reference to the paper's Figure 7. This subroutine essentially performs the task embodied in that figure. Referring to Figure 4 in this paper, the following features are of interest:

1) Slope MF is determined by division. (This slope can be likened to examples illustrated in Figure 7 of Recksiek, 1983.)

2) Loop determines sequence, expressed as elements of primary array A, over the open interval UPRIME, RPRIME (but not over the closed interval UPRIME, RPRIME (see next step 3)). INDEX is the counter.

3) Last values of array A are determined for UPRIME and RPRIME. The highest and final value of INDEX is determined here. This program determines which of the two comes first in cut-and-dry fashion as shown in the flowchart. A person would make an "aesthetic" choice based, more or less, on the look of the taper. While this touch can probably be programmed, we thought the extra complication did not warrant the effort.

4) INDEX elements of array A are reproduced F times to produce the required sequence KF = F×INDEX elements of array SEQ.

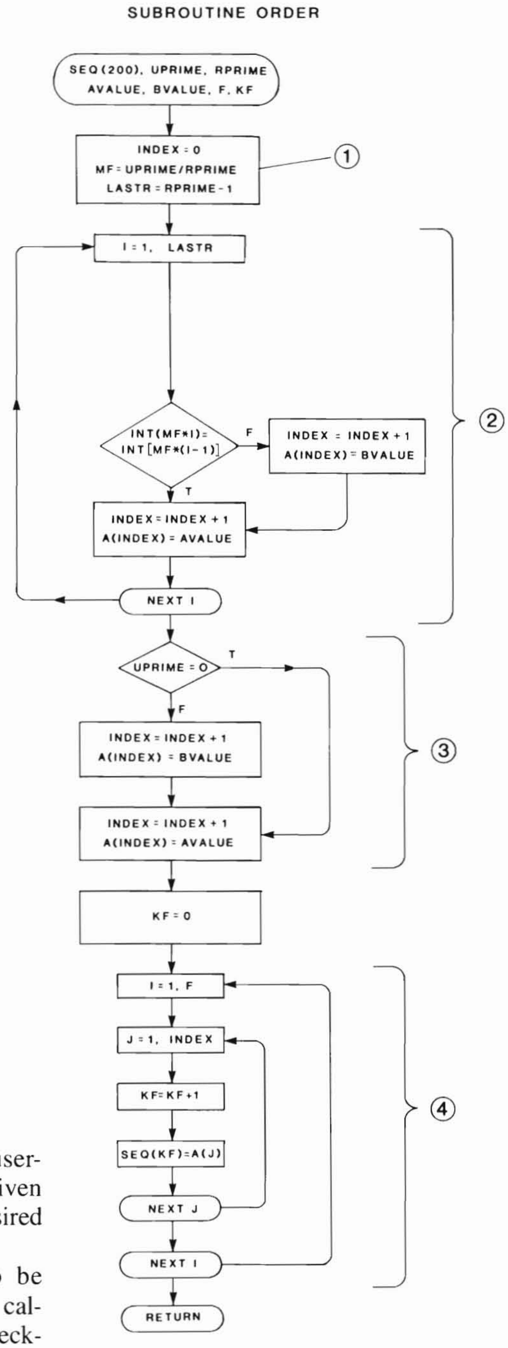


Figure 4.—Subroutine ORDER. Arguments: SEQ, UPRIME, RPRIME, AVALUE, BVALUE, F, KF. This subroutine generates KF nonzero elements of array SEQ such that each element represents one number in a tapering sequence.

Conclusions

The program described in this paper can be easily modified for other tasks. As mentioned earlier, the point of modification would be in the main program. For instance, the main program can be easily modified to reckon wing tapers.

The main program may be expanded to perform jobs other than, or in addition to, finding tapering sequences. For example, the user may wish to consider all dimensions of the piece, i.e., depth, wide end, narrow end, and taper, as being potential unknowns. On user-specified option, the program, given three dimensions, could find the desired but unknown fourth dimension.

Other capabilities could also be added. For example, hanging ratio calculations could be incorporated to reckon actual dimensions (for hanging the web onto headrope and fishing line, etc.). Or, twine weight parameters could be entered to estimate the amount of material actually required to construct the piece.

Acknowledgments

We thank Marion McHugh for preparing the drawings. This research was supported by the Office of Sea Grant (National Oceanic and Atmospheric Administration, NA81AA-D-00073). This publication is Contribution number 2142 of the Rhode Island Experiment Station.

Literature Cited

Recksiek, C. W. 1983. Shaping and assembling webbing. Mar. Fish. Rev. 45(10-11-12):26-41.