CIVIL ENGINEERS
POCKET COMPUTER MONTHLY

GCP PROGRAM - RECTANGULAR RESERVOIR (PC-2)

In our August issue we deal with a problem which, at least in our experience, can be time consuming - - the sizing of a rectangular reservoir.

Despite what you may have thought, the exact volume of such a "prismoid" is not found using average end areas. Simpson's Rule allows an exact solution using the following formula:

$$V = H/6 (A_0 + 4A_1 + A_2)$$

where:
H = Height, or water depth, ft.
A_0 = Base Area, $ft.^2$
A_1 = Area at mid-height, $ft.^2$
A_2 = Area of top surface, $ft.^2$

In our approach to the problem, we will assume that the designer wishes to investigate all parameters, namely, surface dimensions, water depth, volume, base dimensions, and side slopes. We have therefore written four programs so that these parameters may be juggled until the right combination of dimensions, depth, and slopes is found so as to achieve the desired volume.

PROGRAM "J"

Given the water depth, base dimensions and four side slopes, the program solves for volume in cubic feet and gallons, and the dimensions of the upper surface. See sketch.

PROGRAM "K"

Given the water depth, surface dimensions, and four side slopes, the program solves for volume in cubic feet and gallons, and the dimensions of the base.

PROGRAM "L"

Given the volume in either cubic feet or gallons, the base dimensions, and four side slopes, the program solves for the water depth (nearest 1/10 foot) and the surface dimensions.

PROGRAM "M"

Given the volume in either cubic feet or gallons, the upper surface dimensions, and the side slopes, the program solves for the water depth (nearest 1/10 foot) and the base dimensions.

In order to use these four programs within the GCP (PC-2) format, the GCP procedure is altered slightly.

In using GCP ordinarily, as you will recall, the initial run is started by pressing DEF A. The program is titled, then each input and output parameter is labelled and the first problem is solved. Successive runs are made by pressing DEF C so as to activate the program again; the use of DEF C will allow rapid solution of problems with full documentation.

A variation in GCP procedure has, however, been introduced in this case, as follows:

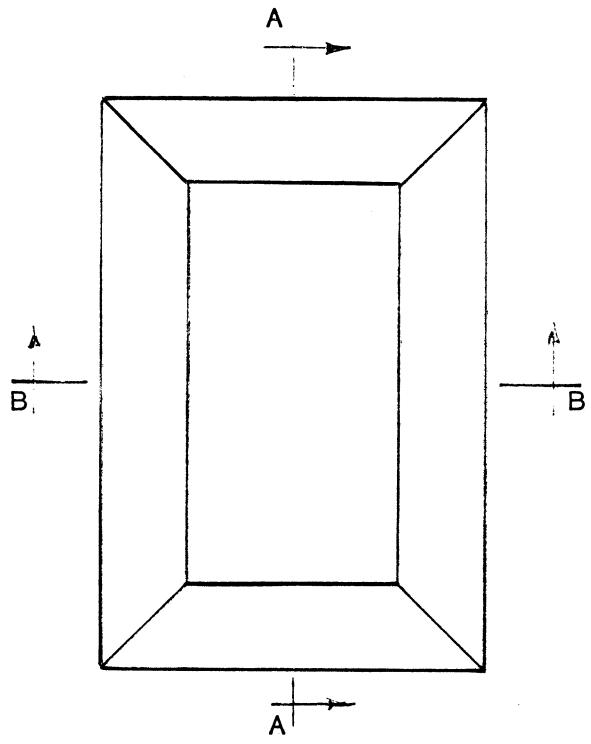
> Press DEF A, then label the input parameters; when the program asks for values of the input parameters, simply press ENTER; then label the output parameters. When that task is completed, press either DEF J, DEF K, DEF L, or DEF M so as to start the desired program. Successive runs of that same program may be accomplished by pressing DEF C.

Note that the list of input parameters is identical to the list of output parameters, except that there are eleven inputs and only eight outputs.

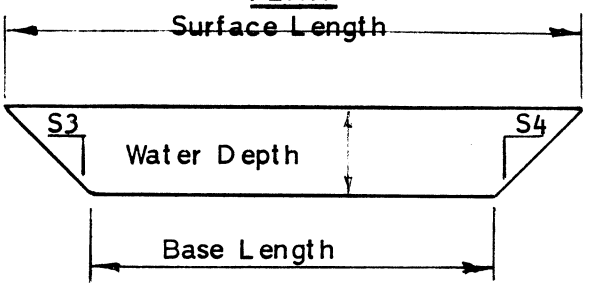
In entering input data, enter "zero" for each unknown value. The side slopes are always "known"; none of the programs solves for side slopes.

If, during a rerun of a problem, only one parameter (such as water depth) is being changed, you may press DEF C, enter the new value of the parameter (e.g. depth); then when the next value is requested,

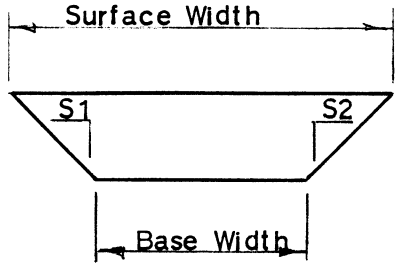
press only ENTER; the remaining values will all be retained as in the prior problem. This saves time.



PLAN



SECTION A-A



SECTION B-B

PROGRAM LISTING/PC-2 (GCP)
RECTANGULAR RESERVOIR

```
 95   ON X GOTO 100,200,300,400
 96   "J" X=1:GOTO "C"
 97   "K" X=2:GOTO "C"
 98   "L" X=3:GOTO "C"
 99   "M" X=4:GOTO "C"
100   U=L*M:F=L+I*(P+Q):G=M+I*(R+S)
110   V=(L+I/2*(P+Q))*(M+I/2*(R+S))
120   W=F*G:A=I:B=A/6*(U+4*V+W):
      C=7.481*B:D=L:E=M
130   GOTO 1203
200   U=N*O:D=N-I*(P+Q):E=O-I*(R+S)
210   V=(N-I/2*(P+Q))*(O-I/2*(R+S))
220   W=D*E:A=I:B=A/6*(U+4*V+W):
      C=7.481*B:F=N:G=O
230   GOTO 1203
300   Z=0:I=0:IF J=0 LET J=K/7.481
310   I=I+1:GOTO 330
320   I=I+.1
330   U=L*M:F=L+I*(P+Q):G=M+I*(R+S)
340   V=(L+I/2*(P+Q))*(M+I/2*(R+S))
350   W=F*G:A=I:B=A/6*(U+4*V+W):
      C=7.481*B:D=L:E=M
360   IF B=J THEN 1203
365   IF Z=1 AND B<J THEN 320
370   IF Z=1 THEN 1203
380   IF B>J LET I=I-1:Z=1:GOTO 320
390   GOTO 310
400   Z=0:I=0:IF J=0 LET J=K/7.481
410   I=I+1:GOTO 430
420   I=I+.1
430   U=N*O:D=N-I*(P+Q):E=O-I*(R+S)
440   V=(N-I/2*(P+Q))*(O-I/2*(R+S))
450   W=D*E:A=I:B=A/6*(U+4*V+W):
      C=7.481*B:F=N:G=O
460   IF B=J THEN 1203
465   IF Z=1 AND B<J THEN 420
470   IF Z=1 THEN 1203
480   IF B>J LET I=I-1:Z=1:GOTO 420
490   GOTO 410
```

Input Parameters:		Output Parameters:
I =	Water Depth, Ft.	= A
J =	Volume, Cubic Feet	= B
K =	Volume, Gallons	= C
L =	Base Width, ft.	= D
M =	Base Length, ft.	= E
N =	Surface Width, ft.	= F
O =	Surface Length, ft.	= G
P =	Slope S1	
Q =	Slope S2	
R =	Slope S3	
S =	Slope S4	

WORKED-OUT EXAMPLES

A rectangular lagoon of six foot maximum water depth, 3:1 side slopes, and seven million gallon capacity is required. The width is to be approximately 2/3 of the length.

Using program K, try a lagoon with a surface area of 350' x 500' and 3:1 side slopes.

TRIAL #1

Enter

Water Depth	= 6 ft.
Vol, CF	= 0
Vol, Gal	= 0
Base Width	= 0
Base Length	= 0
Surface Width	= 350 ft.
Surface Length	= 500 ft.
S1	= 3
S2	= 3
S3	= 3
S4	= 3

Output

Vol, CF	= 960792
Vol, Gal	= 7187685
Base Width	= 314
Base Length	= 464

```
**RECT RESERVOIR
**   **  **   **

WATER DPTH, FT =
                 6
VOL, CF =
                 0
VOL, GAL =
                 0
BASE WDTH, FT =
                 0
BASE LGTH, FT =
                 0
SURF W, FT =
               350
SURF L, FT =
               500
S1 =
                 3
S2 =
                 3
S3 =
                 3
S4 =
                 3
```

```
COMP. RESULTS:

WATER DPTH, FT =
                 6
VOL, CF =
           960792
VOL, GAL =
        7187684.952
BASE W, FT =
               314
BASE L, FT =
               464
SURF W, FT =
               350
SURF L, FT =
               500
```

TRIAL #2

Using Program M the program found that the water depth was 5.9 for surface dimensions of 350 and 500 feet. However, the reservoir was 1 percent oversized.

```
**RECT RESERVOIR
**   **  **   **

COMP. RESULTS:

WATER DPTH, FT =
               5.9
VOL, CF =
         946199.048
VOL, GAL =
        7078515.078
BASE W, FT =
             314.6
BASE L, FT =
             464.6
SURF W, FT =
               350
SURF L, FT =
               500
```

TRIAL #3

Using program J, a water depth of 5.8 ft. and base dimensions of 315 and 465 ft., the volume is 6,962,000 gal. This 1/2 percent undersize of volume can be readily made up by a slight increase in depth; a rerun of Program J with a water depth of 5.85 ft. yields a volume of 7,030,000.

```
**RECT RESERVOIR
**   **  **   **

COMP. RESULTS:

WATER DPTH,FT =
                5.8
VOL,CF =
         930613.944
VOL,GAL =
        6961922.915
BASE W,FT =
                315
BASE L,FT =
                465
SURF W,FT =
              349.8
SURF L,FT =
              499.8
**RECT RESERVOIR
**   **  **   **

COMP. RESULTS:

WATER DPTH,FT =
               5.85
VOL,CF =
        939361.8195
VOL,GAL =
        7027365.772
BASE W,FT =
                315
BASE L,FT =
                465
SURF W,FT =
              350.1
SURF L,FT =
              500.1
```

Since the lagoon will vary in depth throughout the year, the relationship between volume and water depth can now be readily investigated using repeated runs of Program J. As stated, for each run, enter the new value of depth, then when the next parameter is asked for, press ENTER to achieve a new run.

Reminder - The listing and discussion of GCP programs, with and without printer/plotter, was provided in our January, 1983 issue. Our mailing of the February issue included a set of instructions to aid you in the use of GCP.

Note This publication is solely for educational and experimental purposes. CECOM makes no representation or warranty of any kind, expressed or implied, concerning the accuracy, completeness, suitability or utility of any program, information, or process provided in this publication. We do not assume any liability for any direct, indirect, incidental or consequential damages relating to the use or application of the programs or information contained herein. We suggest that the user prove each program by making a trial run on a test problem for which answers are known.

HELPFUL HINTS FOR PC-2

STATUS

The STATUS function is useful as follows:

Type STATUS Ø and press ENTER to determine the number of remaining bytes of memory. However, memory assigned to array variables is not accounted for, nor memory assigned to 2-character variables.

STATUS 1 reports the number of bytes of memory your program has used up so far. Again it does not account for 2-character or array variables.

STATUS 2 provides the address where user's program ends.

STATUS 3 reports the address of the bottom of the stack which includes 2-character variables and array elements.

STATUS 4 provides the last complete line number which the program executed.

In order to find the address where user program storage begins, type STATUS 2-STATUS 1.

In order to determine the amount of memory left after accounting for all program, 2-character variables, and array variables, enter STATUS 3- STATUS 2.

In order to determine the line number where branching to a subroutine occurred insert the following at the beginning of each subroutine:

```
Z=STATUS 4:LPRINT Z
```

This command will allow you to track all branching or jumps during program execution.

Suppose you wish to insert a subroutine which RETURNS to another line (as contrasted with GOSUB which always RETURNS to the same line). Wherever such a "subroutine" is called in the program something like the following could be used instead of GOSUB.

```
100 IF Z=1 GOTO 1000
105 GOTO 120
110 . . . .(this is the line to which
    the program returns after
    branching to the subroutine.
120 . . . .
1000 X=STATUS 4+10
1010 . . . . .(operation desired in this
    subroutine)
1020 GOTO X
```

Essentially this use of STATUS 4 allows a RETURN to a line which bears a constant numerical relationship, plus or minus, to the line containing the GOTO command.

Other variations will occur to you as you consider the power of STATUS 4.

MERGE

Where more than one program has been MERGED the CSAVE command will save the entire memory, and will not respect any attempt to compartmentalize the program memory relative to mass storage to tape.

The first, second and subsequent "MERGE" programs can be activated by the use of one of the definable characters found in the second and third row of the PC-2 keyboard, e.g., GOTO "A", despite whether it exists in the first or a subsequent program. It should exist only once, however.

In proving this we went through the following procedure:

With the PC-2's memory cleared, we first typed the following line:

```
10 "B" LPRINT "SECOND PROGRAM":
  GOTO "A"
ENTER
```

Then we stored this program on tape (we called it "B-MERGE"). Then we typed: NEW ENTER to clear the PC-2. Then:

```
10  "A" LPRINT "FIRST PROGRAM":
    GOTO "B"
ENTER
```

Then we cued up the tape properly and typed: MERGE ENTER. The program title "B-MERGE" appeared on the display, then the prompt, and the tape machine stopped. We checked by scrolling at this point; both lines 1Ø were in the memory. We then pressed DEF A. The printer typed:

FIRST PROGRAM
SECOND PROGRAM
FIRST PROGRAM

etc.

We pressed BREAK (ON) to stop this action.

We obtained the same response by typing RUN ENTER.

Then we pressed DEF B and got the same result. However, the order was reversed.

SECOND PROGRAM
FIRST PROGRAM
SECOND PROGRAM

etc.

Again we pressed BREAK.

Then we typed LList and the two Lines 10 were typed out.

We then attempted to edit the first program without success; only the second program could be edited. In fact when we displayed line 1Ø (the first program), edited it, and pressed ENTER, line 1Ø in the second program was destroyed. It was replaced by the edited version of line 1Ø of the first program; line 1Ø of the first program remained intact (unedited)

We then restored the two lines 1Ø as before, then displayed the first program. Using ► we moved the cursor to the right a few places but did not edit, simply pressed "ENTER". As before, the first program displaced the second.

We then restored the original condition once more; however, we assigned the second program to line 2Ø. Upon moving the cursor to the right in line 1Ø (first program), as before, and pressing ENTER, we found we now had two lines 1Ø and one line 2Ø.

Generally, therefore, we had discovered how to move lines from one side of the "partition" to the other. Having moved the line across this partition, we could edit it normally.

After clearing PC-2's memory we then proceeded to enter.

```
3Ø "C" LPRINT "THIRD PROGRAM":
   GOTO "A"
```

and recorded same on tape.

Then the same procedure was used for:

```
2Ø "B" LPRINT "SECOND PROGRAM":
   GOTO"C"
```

and recorded on tape.

Then we cleared memory and entered:

```
1Ø "A" LPRINT "FIRST PROGRAM":
   GOTO "B"
```

Then we merged the B program, then the C program.

Upon entering either DEF A, or RUN ENTER we obtained on the printer:

FIRST PROGRAM
SECOND PROGRAM
THIRD PROGRAM
FIRST PROGRAM

etc.

We found that we could activate any of the three programs by pressing DEF A, DEF B, or DEF C.

We were able to SAVE these three programs to tape. Then, after clearing the computer's memory, we re-loaded them. We found we could LLIST, and we could edit in a limited sense. However, editing

seemed to follow a set of rules with which we were not familiar. For example line numbers could be changed but reorientation of lines might not occur. We could not delete lines by typing the line number, ENTER. Editing seemed therefore uncertain and chancy. At this point we stopped experimenting. As a general rule, using MERGE, we would avoid editing.

We would always use DEF (key letter) to run programs, and to move across the partitions.

GCP PROGRAM - RECTANGULAR RESERVOIR - COMPLETE LISTING (PC-1)

The following program for PC-1 is in the GCP format but was necessarily modified because of shortage of memory. To start the program, press SHIFT A in the DEF Mode, then when the "prompt" appears, press either SHIFT J, SHIFT K, SHIFT L, OR SHIFT M, depending on which program is desired (see page 1). Then enter all eleven input parameters (I through S). Enter zeros for the unknown values.

See Page 2 for diagrams and parameter list.

```
 5   "J"X$="J":GOTO "C"
 10  "K"X$="K":GOTO "C"
 15  "L"X$="L":GOTO "C"
 20  "M"X$="M":GOTO "C"
 25  "A" CLEAR
 30  I$="I":J$="J":K$="K"
 31  L$="L":M$="M":N$="N":O$="O":
     P$="P":Q$="Q":R$="R":S$="S"
 35  FOR A=9 TO 19
 40  A$(A+26)=A$(A):NEXT A:END
 69  "C"FOR A=9 TO 19:A$(A)=
     A$(A+26): NEXT A
 70  FOR A=9 TO 19:PAUSE "INPUT":
     PAUSE A$(A);".....";A$(A);".
     .....";A$(A):INPUT A(A):NEXT A
 96  IF X$="J" THEN "V"
 97  IF X$="K" THEN "X"
 98  IF X$="L" THEN "Z"
 99  IF X$="M" THEN "="
100  "V"U=LM:F=L+I*(P+Q):G=M+I*(R+S):
     V=(L+I/2*(P+Q))*(M+I/2*(R+S))
120  W=FG:A=I:B=A/6*(U+4*V+W):C=
     7.481*B:D=L:E=M
130  GOTO 501
200  "X"U=NO:D=N-I*(P+Q):E=O-I*(R+S):
     V=(N-I/2*(P+Q))*(O-I/2*(R+S))
220  W=DE:A=I:B=A/6*(U+4*V+W):
     C=7.481*B:F=N:G=O
230  GOTO 501
300  "Z"Z=0:I=0:IF J=0 LET J=K/7.481
310  I=I+1:GOTO 330
320  I=I+.1
330  U=LM:F=L+I*(P+Q):G=M+I*(R+S):
     V=(L+I/2*(P+Q))*(M+I/2*(R+S))
350  W=FG:A=I:B=A/6*(U+4*V+W):
     C=7.481*B:D=L:E=M
360  IF B=J THEN 501
365  IF Z=1 IF B<J THEN 320
370  IF Z=1 THEN 501
380  IF B>J LET I=I-1:Z=1:GOTO 320
390  GOTO 310
400  "="Z=0:I=0:IF J=0 LET J=K/7.481
410  I=I+1:GOTO 430
420  I=I+.1
430  U=NO:D=N-I*(P+Q):E=O-I*(R+S):
     V=(N-I/2*(P+Q))*(O-I/2*(R+S))
450  W=DE:A=I:B=A/6*(U+4*V+W):
     C=7.481*B:F=N:G=O
460  IF B=J THEN 501
465  IF Z=1 IF B<J THEN 420
470  IF Z=1 THEN 501
480  IF B>J LET I=I-1:Z=1:GOTO 420
490  GOTO 410
501  PRINT "PROGRAM ";X$:FOR
     Y=1 TO 7:PRINT A(Y):NEXT
     Y:PRINT P:PRINT Q: PRINT
     R: PRINT S
502  PRINT " ":PRINT " ":PRINT
     " ":END
```

The printout will include parameters A through G, and P through S.

A = Water Depth, ft.
B = Volume, Cubic Feet
C = Volume, Gallons
D = Base Width, ft.
E = Base Length, ft.
F = Surface Width, ft.
G = Surface Length, ft.
P = Slope S1
Q = Slope S2
R = Slope S3
S = Slope S4