BNL-104758-2014-TECH

AGS/AD/Tech Note No. 342;BNL-104758-2014-IR

A METHOD FOR PREPARING INPUT FILES FOR THE POISSON CODES

A. Kponou

August 1990

Collider Accelerator Department

Brookhaven National Laboratory

U.S. Department of Energy

USDOE Office of Science (SC)

DISCLAIMER

Accelerator Division
Alternating Gradient Synchrotron Department
BROOKHAVEN NATIONAL LABORATORY
Upton, New York 11973

Accelerator Division
Technical Note

AGS/AD/Tech. Note No. 342

A METHOD FOR PREPARING INPUT FILES FOR THE POISSON CODES

A. Kponou

August 22, 1990
Corrigendum, November 19, 1991

ABSTRACT

A method is suggested for preparing input files for AUTOMESH, the first of the POISSON programs. Boundary points used to define the regions in a problem are given labels which are associated with the coordinates of the points. The '$PO' lines are then written with the labels instead of the coordinates. A suitable program then reads the file, replaces the labels with the coordinates, and prepares an output file in the standard format.

INTRODUCTION

At present, preparing an input file for AUTOMESH, the first of the POISSON program, requires that the coordinates of boundary points of the different regions of the problem be entered as often as these points occur. Moreover, if one or more boundary points are changed--changing the shape of the regions--the coordinates must be changed wherever these points occur.

In the approach suggested here, a unique label is associated with each boundary point, and the coordinates of the points are given at the same time. A computer program then reads the file and substitutes the coordinates of a point wherever the label of that point is encountered--in the lines beginning with '$PO'. An output file, without the label definitions, is saved for use as the input to AUTOMESH.

The advantages of this approach are:

i. The coordinates of a point are entered only once.

ii. Any change in the coordinates of a point is made at only one location--the label definition for that point.

An input file can therefore be prepared and modified much faster, and the chances for typographical errors are significantly reduced.

IMPLEMENTATION

REXX (IBM 3090 and PC)

The scheme was first tried out with the REXX language on the IBM 3090. A program was written by H. Berry of the Brookhaven Computing & Communications Division (CCD) to do the conversion. It was later modified to run on a PC using Personal REXX,[1] the PC implementation of the language. Listings of both versions of the program are given in Appendices A and B. An example of an input file is given in Figure 1.

The labels have the root, "POINT", and a number which REXX uses as an array index. Lines beginning with "$PO" are searched for a label, which is then replaced with its associated coordinates. Lines not starting with "POINT" or "$PO", except for the delimiter $$, are copied to the output file, which is also reproduced in Figure 1.

The output file was successfully processed by AUTOMESH on the VAX Cluster at CCD.

DCL (VAX/VMS)

DCL on the VAX also has similar string manipulation capabilities. A program in this language was written by J. Bennett of CCD. It is listed in Appendix C.

DISCUSSION

The scheme proposed for simplifying preparation of the input file for AUTOMESH has been successfully implemented on three classes of widely used computers, without modifying AUTOMESH in any way. A FORTRAN program to do the same thing appeared to be very complicated to write; manipulating character strings is not one of FORTRAN's strengths.

EXEC (IBM 3090), COM (VAX), and BATCH (PC) files can be written for "single pass" operation, where AUTOMESH is executed immediately after the file has been converted to standard format. Passing the input and output file names as parameters in the program execution command gives complete flexibility in naming them.

Readers familiar with the code MAD[2] will recognize a similarity between the work described here and the way a beam line is set up there. The idea for this work came to me after I started using MAD.

ACKNOWLEDGMENTS

At the outset, I considered using FORTRAN, then EDT Macros on the VAX, but felt there had to be a "neater" way. H. Berry demonstrated a REXX solution in under an hour. Later, J. Bennett did the same with DCL. F. DeVito and C. Rohrig (a summer student) of CCD also graciously helped.

REFERENCES

1. Personal REXX, Mansfield Software Group, Inc., Box 532, Storrs, CT 06268.

2. The MAD Program, F.C. Iselin, E. Keil, J. Niederer, and J-M. Veuillen, CERN/LEP-TH/87-60.

```
POINT1 X=0.,Y=0.
POINT2 X0=0.,Y0=0.,R=4.,THETA=0.
POINT3 X=6.6,Y=0.
POINT4 X0=0.,Y0=0.,R=6.6,THETA=30.
POINT5 R=4.,THETA=30.
POINT6 X0=0.,Y0=0.,R=1.2,THETA=30.
POINT7 R=1.3416,THETA=3.4349
POINT8 R=4.,THETA=21.4692
POINT9 R=2.2361,THETA=3.4349
POINT10 R=4.,THETA=15.9638
 $$
 ENTRANCE MK3 - a test of rexx 002
 $REG NREG=8, NPOINT=7, XMAX=6.6, YMAX=3.3,
  DX=.07, DY=.07$
 $PO POINT1 $
 $PO NT=1,POINT2 $
 $PO POINT3 $
 $PO NT=2, POINT4 $
 $PO NT=1, POINT5 $
 $PO POINT6 $
 $PO POINT1 $
 $REG IREG=2, NPOINT=5, MAT=2$
 $PO POINT2 $
 $PO POINT3 $
 $PO NT=2, POINT4 $
 $PO NT=1, POINT5 $
 $PO NT=2, POINT2 $
 $REG IREG=3, NPOINT=5, MAT=6$
 $PO POINT5 $
 $PO NT=1,POINT6 $
 $PO POINT7 $
 $PO POINT8 $
 $PO NT=2, X0=0., Y0=0., POINT5 $
 $REG IREG=4, NPOINT=5, MAT=7$
 $PO POINT7 $
 $PO NT=1, POINT9 $
 $PO NT=1, POINT10 $
 $PO NT=2, X0=0, Y0=0., POINT8 $
 $PO NT=1, POINT7 $
 $REG IREG=5, MAT=1, NPOINT=7$
 $PO POINT1 $
 $PO NT=1, POINT6 $
 $PO POINT7 $
 $PO POINT9 $
 $PO POINT10 $
 $PO NT=2, X0=0., Y0=0., POINT2 $
 $PO NT=1, POINT1 $
 $REG IREG=6, NPOINT=2, MAT=1, IBOUND=0$
 $PO POINT1 $
 $PO POINT4 $
 $REG IREG=7, NPOINT=2, MAT=1, IBOUND=0$
 $PO POINT3 $
 $PO NT=2, POINT4 $
 $REG IREG=8, NPOINT=2, CUR=100., MAT=1$
 $PO POINT5 $
 $PO POINT8 $
```

```
ENTRANCE MK3 - A TEST OF REXX 002
$REG NREG=8, NPOINT=7, XMAX=6.6, YMAX=3.3,
DX=.07, DY=.07$
$PO  X=0.,Y=0.  $
$PO NT=1, X0=0.,Y0=0.,R=4.,THETA=0.  $
$PO  X=6.6,Y=0.  $
$PO NT=2,  X0=0.,Y0=0.,R=6.6,THETA=30.  $
$PO NT=1,  R=4.,THETA=30.  $
$PO  X0=0.,Y0=0.,R=1.2,THETA=30.  $
$PO  X=0.,Y=0.  $
$REG IREG=2, NPOINT=5, MAT=2$
$PO  X0=0.,Y0=0.,R=4.,THETA=0.  $
$PO  X=6.6,Y=0.  $
$PO NT=2,  X0=0.,Y0=0.,R=6.6,THETA=30.  $
$PO NT=1,  R=4.,THETA=30.  $
$PO NT=2,  X0=0.,Y0=0.,R=4.,THETA=0.  $
$REG IREG=3, NPOINT=5, MAT=6$
$PO  R=4.,THETA=30.  $
$PO NT=1, X0=0.,Y0=0.,R=1.2,THETA=30.  $
$PO  R=1.3416,THETA=3.4349  $
$PO  R=4.,THETA=21.4692  $
$PO NT=2, X0=0., Y0=0.,  R=4.,THETA=30.  $
$REG IREG=4, NPOINT=5, MAT=7$
$PO  R=1.3416,THETA=3.4349  $
$PO NT=1,  R=2.2361,THETA=3.4349  $
$PO NT=1,  R=4.,THETA=15.9638  $
$PO NT=2, X0=0, Y0=0.,  R=4.,THETA=21.4692  $
$PO NT=1,  R=1.3416,THETA=3.4349  $
$REG IREG=5, MAT=1, NPOINT=7$
$PO  X=0.,Y=0.  $
$PO NT=1,  X0=0.,Y0=0.,R=1.2,THETA=30.  $
$PO  R=1.3416,THETA=3.4349  $
$PO  R=2.2361,THETA=3.4349  $
$PO  R=4.,THETA=15.9638  $
$PO NT=2, X0=0., Y0=0.,  X0=0.,Y0=0.,R=4.,THETA=0.  $
$PO NT=1,  X=0.,Y=0.  $
$REG IREG=6, NPOINT=2, MAT=1, IBOUND=0$
$PO  X=0.,Y=0.  $
$PO  X0=0.,Y0=0.,R=6.6,THETA=30.  $
$REG IREG=7, NPOINT=2, MAT=1, IBOUND=0$
$PO  X=6.6,Y=0.  $
$PO NT=2,  X0=0.,Y0=0.,R=6.6,THETA=30.  $
$REG IREG=8, NPOINT=2, CUR=100., MAT=1$
$PO  R=4.,THETA=30.  $
$PO  R=4.,THETA=21.4692  $
```

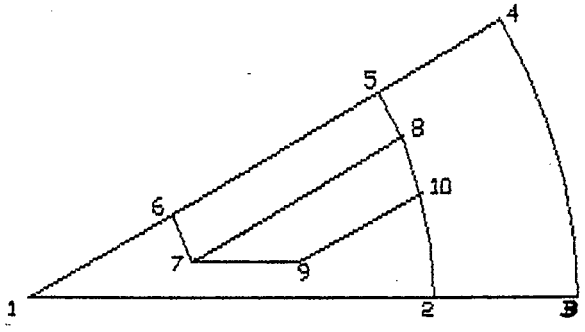

Figure 1. A sample AUTOMESH input file (left) and the corresponding standard AUTOMESH input file (right).

Appendix A

```
/* AUTOMESH file conversion with REXX by H. Berry  7/6/90 */
Say ''
Say 'Enter input and output filenames - ft = DATA assumed '
Say ''
Parse Pull filein fileout .
Trace n
count = 0
newcount = 0
filein = filein ' DATA A'
fileout = fileout ' DATA A'
If Fexist(fileout) Then 'ERASE 'fileout

'EXECIO * DISKR' filein '(STEM LINE.'
If rc ^= 0 Then Exit rc

n = line.0
Do i = 1 To n
   Parse Upper Var line.i word1 remain
   remain = Strip(remain, 'Both', ' ')
      If Substr(word1,1,5) = 'POINT' Then Do
      count = Substr(word1,6,3)
      count = Strip(count,'T', ' ')
      point.count = word1
      value.count = remain
      End
   Else If word1 = '$PO' Then Do
      newcount = newcount + 1
      Parse Var remain before 'POINT' kount . '$'
      newline.newcount = ' ' word1 before value.kount ' $'
      End
   Else If word1 = '$$' Then Nop
   Else Do
      newcount = newcount + 1
      newline.newcount = ' ' word1 remain
      End
End

Do i = 1 To newcount
   'EXECIO 1 DISKW' fileout '(VAR NEWLINE.'i
End
Exit
```

Appendix B

```
/* AUTOMESH file conversion with Personal REXX */
say ''
say 'Enter input and output filenames - .dat assumed'
say ''
parse pull filein fileout .
count = 0
newcount = 0
trace = n
filein=filein '.dat'
fileout=fileout '.dat'
/* erase output file if it exists */
Call dosdel(fileout)
n=1
/* read filein one line at a time                          */
do while lines(filein) > 0
   line.n=linein(filein)
   if line.n='' then leave
   n=n+1
end
n=n-1
DO i = 1 TO n
   PARSE UPPER VAR line.i word1 remain
   remain = STRIP(remain, 'Both', ' ')
   IF SUBSTR(WORD1,1,5)='POINT' THEN DO
      count = SUBSTR(WORD1,6,3)
      count = STRIP(count,'T',' ')
      point.count=word1
      value.count=remain
   END
 /* find which label to substitute for                      */
   ELSE IF word1 = '$PO' THEN DO
      newcount = newcount + 1
      PARSE VAR remain before 'POINT' kount . '$'
 /*   the new $po line                                      */
      newline.newcount = ' ' word1 before value.kount ' $'
   END
   ELSE IF word1 = '$$' THEN NOP  /* A delimiter-don't copy*/
   ELSE DO
 /*   this is not a $po line. copy as is                    */
       newcount = newcount + 1
       newline.newcount = ' ' word1 remain
   END

END
DO i = 1 TO newcount  /* send lines to output file */
   Call lineout fileout, newline.i
END
   Call lineout filein
   Call lineout fileout
EXIT
```

Appendix C

```
$!**************************************************************************
$!****DCL Version by J. Bennett,BNL  7/30/90 ******************************
$!**************************************************************************
$ set noverify
$ set noon
$!
$ on CONTROL_Y then goto Cleanup
$ on ERROR then goto Cleanup
$!
$ open/read infil "p1".dat
$ create "p2".dat
$ open/append outfil "p2".dat
$ cnt=0 $Loop1:
$ cnt=cnt+1
$ read infil data
$ if f$locate("POINT",data).eq.f$length(data) then goto Eof1
$ point"cnt"=data-"POINT""CNT""
$ goto Loop1 $Eof1:
$ max=cnt-1 $Loop2:
$ read/end_of_file=Cleanup infil data
$ cnt=0 $ Inner_loop_1:
$ cnt=cnt+1
$   if cnt.gt.max then goto wrt_new_data
$ offset=f$locate("POINT""cnt" ",data)
$   if offset.eq.f$length(data) then goto Inner_loop_1
$ first=f$extract(0,offset,data)
$ last=f$extract(offset,100,data)-"POINT""cnt""
$ data=first+point"cnt"+last
$Wrt_new_data:
$ write outfil data
$ goto Loop2
$Cleanup:
$ set noverify
$ if f$trnlnm("infil").nes."" then close infil
$ if f$trnlnm("outfil").nes."" then close outfil
$ EXIT
```

The VAX program given in Appendix C has a bug. It incorrectly substitutes for labels with indices of 10 or higher. For example, when cnt=1, the lexical f$locate correctly locates the string POINT1 in the label POINT10, and the coordinates for POINT1 - in the string point1 - are substituted for POINT10 in the $PO line. The following program solves the problem. Use of the f$element lexical function has also simplified and shortened the program. All data must start in column two or higher, and for the $PO cards, there should be at least one blank space on either side of the label string.

```
$ set noverify
$ set noon
$!
$ on CONTROL_Y then goto Cleanup
$ on ERROR then goto Cleanup
$!
$ open/read infil 'p1'.dat
$ create 'p2'.dat
$ open/append outfil 'p2'.dat
$ cnt=0
$Loop1:
$ cnt=cnt+1
$ read infil data
$ first=f$element(1," ",data)
$ if first.eqs."$$" then goto Subst
$ point'cnt'=f$element(2," ",data)
$ goto Loop1
$Subst:
$ read/end_of_file=Cleanup infil data
$ first=f$element(1," ",data)
$ if first.nes."$PO" then goto wrt_new_data
$ second=f$element(2," ",data)
$ count=second-"POINT"
$ data=" "+first+" "+point'count'+" "+f$element(3," ",data)
$Wrt_new_data:
$ write outfil data
$ goto Subst
$Cleanup:
$ set noverify
$ if f$trnlnm("infil").nes."" then close infil
$ if f$trnlnm("outfil").nes."" then close outfil
$ EXIT
```