genmap issue

Friday, September 23, 2016

I initially started with a `ExodusII` file exported from `Trelis`. Then I converted it to nek format and wrote to `.rea` file with the following format (sample):

```
 ***** BOUNDARY CONDITIONS *****
 ***** FLUID  BOUNDARY CONDITIONS *****
E             1  38751.000000      3.0000000000      0.0000000000      0.0000000000      0.0000000000
E             1  2.0000000000      4.0000000000      0.0000000000      0.0000000000      0.0000000000
E             1  51.000000000      1.0000000000      0.0000000000      0.0000000000      0.0000000000
W             1  0.0000000000      0.0000000000      0.0000000000      0.0000000000      0.0000000000
P             1  38001.000000      6.0000000000      0.0000000000      0.0000000000      0.0000000000
E             1  401.00000000      5.0000000000      0.0000000000      0.0000000000      0.0000000000
E             2  38752.000000      3.0000000000      0.0000000000      0.0000000000      0.0000000000
E             2  3.0000000000      4.0000000000      0.0000000000      0.0000000000      0.0000000000
E             2  52.000000000      1.0000000000      0.0000000000      0.0000000000      0.0000000000
E             2  1.0000000000      2.0000000000      0.0000000000      0.0000000000      0.0000000000
P             2  38002.000000      6.0000000000      0.0000000000      0.0000000000      0.0000000000
E             2  402.00000000      5.0000000000      0.0000000000      0.0000000000      0.0000000000
E             3  38753.000000      3.0000000000      0.0000000000      0.0000000000      0.0000000000
E             3  4.0000000000      4.0000000000      0.0000000000      0.0000000000      0.0000000000
E             3  53.000000000      1.0000000000      0.0000000000      0.0000000000      0.0000000000
E             3  2.0000000000      2.0000000000      0.0000000000      0.0000000000      0.0000000000
P             3  38003.000000      6.0000000000      0.0000000000      0.0000000000      0.0000000000
E             3  403.00000000      5.0000000000      0.0000000000      0.0000000000      0.0000000000
E             4  38754.000000      3.0000000000      0.0000000000      0.0000000000      0.0000000000
E             4  5.0000000000      4.0000000000      0.0000000000      0.0000000000      0.0000000000
--
```

Then I ran `reatore2` to generate `.re2` file and I didn't find any errors. Below shows a runtime message from the terminal:

```
 Input old (source) file name:
case3d
 Input new (output) file name:
turbl

Start converting ...

read:  **MESH DATA** 6 lines are X,Y,Z;X,Y,Z. Columns corners 1-4;5-8
     2457600      2457600 3 nelt/nelv/ndim
read:   ***** CURVED SIDE DATA *****
read:   ***** BOUNDARY CONDITIONS *****
read:   ***** FLUID  BOUNDARY CONDITIONS *****
           1      159104  Number of bcs
 done
```

But when I ran `genmap` to generate `.map` file, I received the following message:

```
Input .rea / .re2 name:
turbl
 reading turbl.rea
Input mesh tolerance (default 0.2):
NOTE: smaller is better, but generous is more forgiving for bad meshes.

 using default value
 reading mesh data ...
 start locglob_lexico:           8     2457600    19660800  0.20000000000000001
 locglob:           1           1    19660800
 locglob:           2         513    19660800
 locglob:           3       26163    19660800
 locglob:           1     2537811    19660800
 locglob:           2     2537811    19660800
 locglob:           3     2537811    19660800
 locglob:           1     2537811    19660800
 locglob:           2     2537811    19660800
 locglob:           3     2537811    19660800
 done locglob_lexico:     2537811     2537811    19660800           8
 start periodic vtx:     2457600     2537811
       38600 5  5  3  0.00000000E+00     1000 shift
       77200 5  5  3  0.00000000E+00     2000 shift
      153400 6  5  3  0.00000000E+00     3000 shift
      192000 6  5  3  0.00000000E+00     4000 shift
      230600 5  5  3  0.00000000E+00     5000 shift
      269200 5  5  3  0.00000000E+00     6000 shift
      345400 6  5  3  0.00000000E+00     7000 shift
```

```
 384000 6  5  3  0.00000000E+00     8000 shift
 422600 5  5  3  0.00000000E+00     9000 shift
 461200 5  5  3  0.00000000E+00    10000 shift
 537400 6  5  3  0.00000000E+00    11000 shift
 576000 6  5  3  0.00000000E+00    12000 shift
 614600 5  5  3  0.00000000E+00    13000 shift
 653200 5  5  3  0.00000000E+00    14000 shift
 729400 6  5  3  0.00000000E+00    15000 shift
 768000 6  5  3  0.00000000E+00    16000 shift
 806600 5  5  3  0.00000000E+00    17000 shift
 845200 5  5  3  0.00000000E+00    18000 shift
 921400 6  5  3  0.00000000E+00    19000 shift
 960000 6  5  3  0.00000000E+00    20000 shift
 998600 5  5  3  0.00000000E+00    21000 shift
1037200 5  5  3  0.00000000E+00    22000 shift
1113400 6  5  3  0.00000000E+00    23000 shift
1152000 6  5  3  0.00000000E+00    24000 shift
1190600 5  5  3  0.00000000E+00    25000 shift
1229200 5  5  3  0.00000000E+00    26000 shift
1305400 6  5  3  0.00000000E+00    27000 shift
1344000 6  5  3  0.00000000E+00    28000 shift
1382600 5  5  3  0.00000000E+00    29000 shift
1421200 5  5  3  0.00000000E+00    30000 shift
1497400 6  5  3  0.00000000E+00    31000 shift
1536000 6  5  3  0.00000000E+00    32000 shift
1574600 5  5  3  0.00000000E+00    33000 shift
1613200 5  5  3  0.00000000E+00    34000 shift
1689400 6  5  3  0.00000000E+00    35000 shift
1728000 6  5  3  0.00000000E+00    36000 shift
1766600 5  5  3  0.00000000E+00    37000 shift
1805200 5  5  3  0.00000000E+00    38000 shift
1881400 6  5  3  0.00000000E+00    39000 shift
1920000 6  5  3  0.00000000E+00    40000 shift
1958600 5  5  3  0.00000000E+00    41000 shift
1997200 5  5  3  0.00000000E+00    42000 shift
2073400 6  5  3  0.00000000E+00    43000 shift

abort: PERIODIC MISMATCH 2:
    2073601  5 P    ie
    2111600  6 E    je
  2111600.0009832908        0.0000000000000000       bc


          8  quit
```

First thing I noticed was the weird number `2111600.0009832908` which was suppose to have zeros after the floating point. So I searched for the actual input periodic boundary conditions correspond to the element `2073601` in the `.rea` file and I found that (see attachment `pbc-rea.output`):

```
E       2073601  2112351.0000      3.0000000000      0.0000000000      0.0000000000      0.0000000000
E       2073601  2073602.0000      4.0000000000      0.0000000000      0.0000000000      0.0000000000
E       2073601  2073651.0000      1.0000000000      0.0000000000      0.0000000000      0.0000000000
W       2073601  0.0000000000      0.0000000000      0.0000000000      0.0000000000      0.0000000000
P       2073601  2111601.0000      6.0000000000      0.0000000000      0.0000000000      0.0000000000
E       2073601  2074001.0000      5.0000000000      0.0000000000      0.0000000000      0.0000000000
```

So, it seems the correct element id for this periodic boundary condition is actually `2111601`! I double-checked that in the `.rea` file again and it showed:

```
E       2111601  2150351.0000      3.0000000000      0.0000000000      0.0000000000      0.0000000000
E       2111601  2111602.0000      4.0000000000      0.0000000000      0.0000000000      0.0000000000
E       2111601  2111651.0000      1.0000000000      0.0000000000      0.0000000000      0.0000000000
W       2111601  0.0000000000      0.0000000000      0.0000000000      0.0000000000      0.0000000000
E       2111601  2111201.0000      6.0000000000      0.0000000000      0.0000000000      0.0000000000
P       2111601  2073601.0000      5.0000000000      0.0000000000      0.0000000000      0.0000000000
```

Up to this point, the outputs agree with each other in the `.rea` file. It implies that the weird number `2111600.0009832908` was indeed causing the boundary condition mismatch. Because the boundary conditions was written as binary data by `reatore2` (I assume ASCII I/O in `reatore2` was not causing the problem) and read in by `genmap`, I wonder if the mismatch was due to the binary I/O.
So, I looked into `genmap.f` and found the subroutine `buf_to_bc` that is used to assign the binary data to the variable `bc(5,6,lelt)`.
I uncommented out the write statement trying to output the actual bc data stored in `buf(30)` and it printed (see attachment `pbc-genmap-with-error.output`):

```
 2073599   6  P  2035599.00047395      5.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073600   2  ON       0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073600   6  P  2035600.00047395      5.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073601   4  W        0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073601   5  P  2111600.00098329      6.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073602   5  P  2111602.00098329      6.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073603   5  P  2111602.00098329      6.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073604   5  P  2111604.00098329      6.00000000      0.00000000      0.00000000      0.00000000  CBC
--
 2111501   4  W        0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
 2111550   2  ON       0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
 2111551   4  W        0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
 2111600   2  ON       0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
 2111601   4  W        0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
 2111601   6  P  2073601.00048280      5.00000000      0.00000000      0.00000000      0.00000000  CBC
 2111602   6  P  2073602.00048280      5.00000000      0.00000000      0.00000000      0.00000000  CBC
```

It looks like for every periodic boundary condition, the first number is erroneously represented. After some experiment, I finally identified that one statement in the subroutine that is producing the error:

```
3243 c-----------------------------------------------------------------------
3244       subroutine buf_to_bc(cbl,bl,buf,nelt)   ! version 1 of binary reader
3245
3246       include 'SIZE'
3247
3248       character*3 cbl(6,nelt)
3249       real*8      bl(5,6,nelt)
3250       integer     buf(30)
3251
3252       integer e,eg,f
3253
3254       if(wdsizi.eq.8) then
3255          call copyi4(e,buf(1),1) !1-2
3256          call copyi4(f,buf(3),1) !3-4
3257          call copy  (bl(1,f,e),buf(5),5) !5--14
3258          call chcopy(cbl( f,e),buf(15),3)!15-16
3259          if(nelt.ge.1000000.and.cbl(f,e).eq.'P  ')
3260      $   call copyi4(bl(1,f,e),buf(5),1) !Integer assign connecting P element
3261       else
3262          e  = buf(1)
3263          f  = buf(2)
3264          call copy48r ( bl(1,f,e),buf(3),5)
3265          call chcopy  (cbl(  f,e),buf(8),3)
3266          if(nelt.ge.1000000.and.cbl(f,e).eq.'P  ')
3267      $     bl(1,f,e) = buf(3) ! Integer assign of connecting periodic element
3268       endif
3269
3270
3271       write(7,1) e,f,cbl(f,e),(bl(k,f,e),k=1,5),' CBC'
3272    1  format(i8,i4,2x,a3,5f16.8,1x,a4)
3273
3274       return
3275       end
3276
```

and the subroutine `copyi4` is found in `core/math.f`:

```
 423 c-----------------------------------------------------------------------
 424       subroutine copyi4(a,b,n)
 425       integer a(1)
 426       real    b(1)
 427
 428       do i=1,n
 429          a(i)=b(i)
 430       enddo
 431
 432       return
 433       end
```

In my opinion, the error might be due to the incorrect interpretation of word size (8 byte real vs 4 byte real) during the data type conversion when passing variables from `buf_to_bc` to `copyi4`.
If I commented out this statement, recompiled and ran `genmap` again, then I got (see attachment `pbc-genmap-without-error.output`):

```
 2073599   6  P  2035599.00000000      5.00000000      0.00000000      0.00000000      0.00000000  CBC
 2073600   2  ON       0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
```

```
2073600   6  P  2035600.00000000      5.00000000      0.00000000      0.00000000      0.00000000  CBC
2073601   4  W        0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
2073601   5  P  2111601.00000000      6.00000000      0.00000000      0.00000000      0.00000000  CBC
2073602   5  P  2111602.00000000      6.00000000      0.00000000      0.00000000      0.00000000  CBC
2073603   5  P  2111603.00000000      6.00000000      0.00000000      0.00000000      0.00000000  CBC
2073604   5  P  2111604.00000000      6.00000000      0.00000000      0.00000000      0.00000000  CBC
--
2111551   4  W        0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
2111600   2  ON       0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
2111601   4  W        0.00000000      0.00000000      0.00000000      0.00000000      0.00000000  CBC
2111601   6  P  2073601.00000000      5.00000000      0.00000000      0.00000000      0.00000000  CBC
2111602   6  P  2073602.00000000      5.00000000      0.00000000      0.00000000      0.00000000  CBC
2111603   6  P  2073603.00000000      5.00000000      0.00000000      0.00000000      0.00000000  CBC
2111604   6  P  2073604.00000000      5.00000000      0.00000000      0.00000000      0.00000000  CBC
```

After what I had done, the two boundary conditions matched and the program proceeded without problems!

So, this is what I have found so far in this problem but I'm still not very sure why the variables were interpreted in that way.

Hope this workflow helps you and if you know the reason why it happens, I would be very happy to learn about that.