**Particle not in v (or u) element or particle jumped over v (or u) element**

Elizabeth North, 3/14/14

These errors can occur when there are single-grid-width channels in the hydrodynamic model which appear to be parallel to the grid when grid cells are rectilinear (no curvature). The LTRANS boundary maker creates a boundary but there may not be a v or u element inside the boundary so the particle can be inside the model domain but not within an element, resulting in the errors above.

I took the following steps to manually correct the model by forcing the creation of elements by setting the mask value of nodes to water when elements were being generated. Here are the steps:

1. I ran the model with 100,000 particles distributed evenly throughout the model domain to find to locations where the model was having problems. I ran with ErrorFlag = 3 in LTRANS.data so that particles would stop moving and return an error in the ErrorLog.txt file. The ErrorLog.txt lists the particle numbers. I added code in LTRANS.f90 to print out the x/y location of the particle numbers that were having problems (see ‘!ewn grid’ in code for my additions to LTRANS v.2b):

OPEN(210,FILE='ErrorLog.txt',POSITION='APPEND')

SELECT CASE (ele\_err)

CASE(4)

write(210,21) n,ix(3)

CASE(5)

write(210,22) n,ix(3)

CASE(6)

write(210,23) n,ix(3)

write(210,\*) par(n,pX),par(n,pY),par(n,pZ) !ewn grid

END SELECT

2. I visualized the locations of the model boundary (xybounds.bln created by LTRANS when BoundaryBLNs = .TRUE. in LTRANS.data) and made files with the following information: v-mask node locations and values (v\_mask.csv), the v\_element locations (v\_elements.bln), and the node numbers of the v\_elements (v\_element\_node\_numbers.csv). The .bln files are used to create basemaps in Surfer. The code that created these files is in the section of the hydrodynamic module that is pasted below. Search for the name of the file to see it. See Fig. 1 and 2 to see location of missing elements.

3. I set the mask value to 1 at a node that was part of the missing element, let LTRANS create the elements, then set the mask value back to zero. The node number was read off of the plot created in set #2. The code simply looked like this:

v\_mask(124975)=1 !ewn grid

See more in the hydrodynamic module pasted below.

4. Change the number of wet elements in the parameter module so that dimensions of the v and u elements include the new elements:

u\_elements = wetU + 2 ! u elements with at least one corner with mask\_u=1 !ewn grid

v\_elements = wetV + 25 ! v elements with at least one corner with mask\_v=1 !ewn grid

This code in the hydrodynamic module helped me make sure they were the same:

write(\*,\*) 'u\_elements', u\_elements !ewn grid

write(\*,\*) 'count with adjusted nodes',count !ewn grid

Note: setting nodes that were land to water to create the elements, and then setting them back to land, will not influence current velocities because u and v velocities are multiplied by the mask values (u\_m, v\_m) when they are read in so will be set to zero.

Section of Hydrodynamic model that was changed (search for !ewn grid to see changes):

! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CREATE ELEMENTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!Store Mask Values to multiply by...

m\_r = mask\_rho

m\_u = mask\_u

m\_v = mask\_v

! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

! ~ 4B. Prepare Elements (i.e., assign ID numbers to rectangular grids) ~

! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

write(\*,\*) 'create elements'

! Convert rho nodes lon/lat to x/y coordinates

do j=1,uj

do i=1,vi

x\_rho(i,j) = lon2x(lon\_rho(i,j),lat\_rho(i,j))

y\_rho(i,j) = lat2y(lat\_rho(i,j))

enddo

enddo

open(1122,FILE='u\_mask.csv',STATUS='REPLACE') !ewn grid

! Convert u nodes lon/lat to x/y coordinates

do j=1,uj

do i=1,ui

x\_u(i,j) = lon2x(lon\_u(i,j),lat\_u(i,j))

y\_u(i,j) = lat2y(lat\_u(i,j))

write(1122,\*) x\_u(i,j),',',y\_u(i,j),',',mask\_u(i,j) !ewn grid

enddo

enddo

close(1122) !ewn grid

open(1112,FILE='v\_mask.csv',STATUS='REPLACE') !ewn grid

! Convert v nodes lon/lat to x/y coordinates

do j=1,vj

do i=1,vi

x\_v(i,j) = lon2x(lon\_v(i,j),lat\_v(i,j))

y\_v(i,j) = lat2y(lat\_v(i,j))

write(1112,\*) x\_v(i,j),',',y\_v(i,j),',',mask\_v(i,j) !ewn grid

enddo

enddo

close(1112) !ewn grid

! Assign mask values to rho nodes

count = 0

do j=1,uj

do i=1,vi

count = count + 1 !move to next node number

!cycles through each variable replacing the vi,uj part with count

! essentially giving it node numbers

rho\_mask(count) = mask\_rho(i,j)

rho\_angle(count) = angle(i,j)

enddo

enddo

open(1123,FILE='u\_node\_numbers.csv',STATUS='REPLACE') !ewn grid

! Assign mask values to u nodes

count = 0

do j=1,uj

do i=1,ui

count = count + 1 !move to next node number

!cycles through each variable replacing the vi,uj part with count

! essentially giving it node numbers

u\_mask(count) = mask\_u(i,j)

write(1123,\*) x\_u(i,j),',',y\_u(i,j),',',count,',',u\_mask(count) !ewn grid

enddo

enddo

close(1123) !ewn grid

open(1113,FILE='v\_node\_numbers.csv',STATUS='REPLACE') !ewn grid

! Assign mask values to v nodes

count = 0

do j=1,vj

do i=1,vi

count = count + 1 !move to next node number

!cycles through each variable replacing the vi,uj part with count

! essentially giving it node numbers

v\_mask(count) = mask\_v(i,j)

write(1113,\*) x\_v(i,j),',',y\_v(i,j),',',count,',',v\_mask(count) !ewn grid

enddo

enddo

close(1113) !ewn grid

! Create matrix that contains the node numbers for each rho element

count = 0

do j=1,uj-1 !z2v3.2

do i=1,vi-1

count = count + 1

r\_ele(1,count) = i + (j-1)\*vi

r\_ele(2,count) = i + 1 + (j-1)\*vi

r\_ele(3,count) = i + 1 + j\*vi

r\_ele(4,count) = i + j\*vi

enddo

enddo

! Create matrix that contains the node numbers for each u element

count = 0

do j=1,uj-1 !z2v3.2

do i=1,ui-1

count = count + 1

u\_ele(1,count) = i + (j-1)\*ui

u\_ele(2,count) = i + 1 + (j-1)\*ui

u\_ele(3,count) = i + 1 + j\*ui

u\_ele(4,count) = i + j\*ui

enddo

enddo

! Create matrix that contains the node numbers for each v element

count = 0

do j=1,vj-1 !z2v3.2

do i=1,vi-1

count = count + 1

v\_ele(1,count) = i + (j-1)\*vi

v\_ele(2,count) = i + 1 + (j-1)\*vi

v\_ele(3,count) = i + 1 + j\*vi

v\_ele(4,count) = i + j\*vi

enddo

enddo

! Create matrix that contains only the rho elements that contain a node

! whose mask value = 1 (i.e., it has at least one water point).

count = 0

do i=1,max\_rho\_elements

inele = 0

!using the mask determine if any of the nodes for the current

! element are inbounds, if so set inele to 1

if( rho\_mask(r\_ele(1,i)) .EQ. 1) inele=1

if( rho\_mask(r\_ele(2,i)) .EQ. 1) inele=1

if( rho\_mask(r\_ele(3,i)) .EQ. 1) inele=1

if( rho\_mask(r\_ele(4,i)) .EQ. 1) inele=1 !z2v3.2

!if inele = 1 then at least one of the three nodes for this element

! are in bounds.

if( inele .EQ. 1 ) then

count = count + 1

!create array of elements that contain at least one node in bounds

RE(1,count) = r\_ele(1,i)

RE(2,count) = r\_ele(2,i)

RE(3,count) = r\_ele(3,i)

RE(4,count) = r\_ele(4,i) !z2v3.2

endif

enddo

u\_mask(40750)=1 !ewn grid

! Create matrix that contains only the u elements that contain a node

! whose mask value = 1 (i.e., it has at least one water point).

count = 0

do i=1,max\_u\_elements

inele = 0

!using the mask determine if any of the nodes for the current

! element are inbounds, if so set inele to 1

if( u\_mask(u\_ele(1,i)) .EQ. 1) inele=1

if( u\_mask(u\_ele(2,i)) .EQ. 1) inele=1

if( u\_mask(u\_ele(3,i)) .EQ. 1) inele=1

if( u\_mask(u\_ele(4,i)) .EQ. 1) inele=1 !z2v3.2

!if inele = 1 then at least one of the three nodes for this element

! are in bounds.

if( inele .EQ. 1 ) then

count = count + 1

!create array of elements that contain at least one node in bounds

UE(1,count) = u\_ele(1,i)

UE(2,count) = u\_ele(2,i)

UE(3,count) = u\_ele(3,i)

UE(4,count) = u\_ele(4,i) !z2v3.2

endif

enddo

write(\*,\*) 'u\_elements', u\_elements !ewn grid

write(\*,\*) 'count with adjusted nodes',count !ewn grid

u\_mask(40750)=0 !ewn grid

v\_mask(124975)=1 !ewn grid

v\_mask(14823)=1 !ewn grid

v\_mask(23861)=1 !ewn grid

v\_mask(23864)=1 !ewn grid

v\_mask(61075)=1 !ewn grid

v\_mask(64966)=1 !ewn grid

v\_mask(98039)=1 !ewn grid

v\_mask(98040)=1 !ewn grid

v\_mask(98041)=1 !ewn grid

v\_mask(98042)=1 !ewn grid

v\_mask(98046)=1 !ewn grid

v\_mask(108558)=1 !ewn grid

v\_mask(129954)=1 !ewn grid

v\_mask(129454)=1 !ewn grid

v\_mask(45484)=1 !ewn grid

! Create matrix that contains only the v elements that contain a node

! whose mask value = 1 (i.e., it has at least one water point).

count = 0

do i=1,max\_v\_elements

inele = 0

!using the mask determine if any of the nodes for the current

! element are inbounds, if so set inele to 1

if( v\_mask(v\_ele(1,i)) .EQ. 1) inele=1

if( v\_mask(v\_ele(2,i)) .EQ. 1) inele=1

if( v\_mask(v\_ele(3,i)) .EQ. 1) inele=1

if( v\_mask(v\_ele(4,i)) .EQ. 1) inele=1 !z2v3.2

!if inele = 1 then at least one of the three nodes for this element

! are in bounds.

if( inele .EQ. 1 ) then

count = count + 1

!create array of elements that contain at least one node in bounds

VE(1,count) = v\_ele(1,i)

VE(2,count) = v\_ele(2,i)

VE(3,count) = v\_ele(3,i)

VE(4,count) = v\_ele(4,i) !z2v3.2

endif

enddo

write(\*,\*) 'v\_elements', v\_elements !ewn grid

write(\*,\*) 'count with adjusted nodes',count !ewn grid

v\_mask(124975)=0 !ewn grid

v\_mask(14823)=0 !ewn grid

v\_mask(23861)=0 !ewn grid

v\_mask(23864)=0 !ewn grid

v\_mask(61075)=0 !ewn grid

v\_mask(64966)=0 !ewn grid

v\_mask(98039)=0 !ewn grid

v\_mask(98040)=0 !ewn grid

v\_mask(98041)=0 !ewn grid

v\_mask(98042)=0 !ewn grid

v\_mask(98046)=0 !ewn grid

v\_mask(108558)=0 !ewn grid

v\_mask(129954)=0 !ewn grid

v\_mask(129454)=0 !ewn grid

v\_mask(45484)=0 !ewn grid

! Create matrices of x/y for rho nodes and depth values in rho node number

! format

count = 0

do j=1,uj

do i=1,vi

count = count + 1 !move to next node number

!cycles through each variable replacing the vi,uj part with count

! essentially giving it node numbers

rx(count) = x\_rho(i,j)

ry(count) = y\_rho(i,j)

depth(count) = romdepth(i,j)

enddo

enddo

! Create matrices of x/y values for u nodes in u node number format

count = 0

do j=1,uj

do i=1,ui

count = count + 1 !move to next node number

!cycles through each variable replacing the ui,uj part with count

! essentially giving it node numbers

ux(count) = x\_u(i,j)

uy(count) = y\_u(i,j)

enddo

enddo

! Create matrices of x/y values for v nodes in v node number format

count = 0

do j=1,vj

do i=1,vi

count = count + 1 !move to next node number

!cycles through each variable replacing the vi,vj part with count

! essentially giving it node numbers

vx(count) = x\_v(i,j)

vy(count) = y\_v(i,j)

enddo

enddo

! Create matrices of x/y node values for each rho, u, and v element

do j=1,rho\_elements

do i=1,4 !z2v3.3

r\_ele\_x(i,j) = rx(RE(i,j))

r\_ele\_y(i,j) = ry(RE(i,j))

enddo

enddo

! do j=1,u\_elements

! do i=1,4 !z2v3.3

! u\_ele\_x(i,j) = ux(UE(i,j))

! u\_ele\_y(i,j) = uy(UE(i,j))

! enddo

! enddo

open(2223,FILE='u\_elements.bln',STATUS='REPLACE') !ewn grid 3/5/14

4443 format(I2,',',I2) !ewn grid

do j=1,u\_elements

write(2223,4443) 5,0 !ewn grid

do i=1,4

u\_ele\_x(i,j) = ux(UE(i,j))

u\_ele\_y(i,j) = uy(UE(i,j))

write(2223,\*) u\_ele\_x(i,j), u\_ele\_y(i,j) !ewn grid

if(i.EQ.4) write(2223,\*) u\_ele\_x(1,j), u\_ele\_y(1,j) !ewn grid

enddo

enddo

close(2223) !ewn grid

! do j=1,v\_elements

! do i=1,4 !z2v3.3

! v\_ele\_x(i,j) = vx(VE(i,j))

! v\_ele\_y(i,j) = vy(VE(i,j))

! enddo

! enddo

open(2222,FILE='v\_elements.bln',STATUS='REPLACE') !ewn grid 3/5/14

4444 format(I2,',',I2) !ewn grid

do j=1,v\_elements

write(2222,4444) 5,0 !ewn grid

do i=1,4

v\_ele\_x(i,j) = vx(VE(i,j))

v\_ele\_y(i,j) = vy(VE(i,j))

write(2222,\*) v\_ele\_x(i,j), v\_ele\_y(i,j) !ewn grid

if(i.EQ.4) write(2222,\*) v\_ele\_x(1,j), v\_ele\_y(1,j) !ewn grid

enddo

enddo

close(2222) !ewn grid

! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIND ADJACENT ELEMENTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

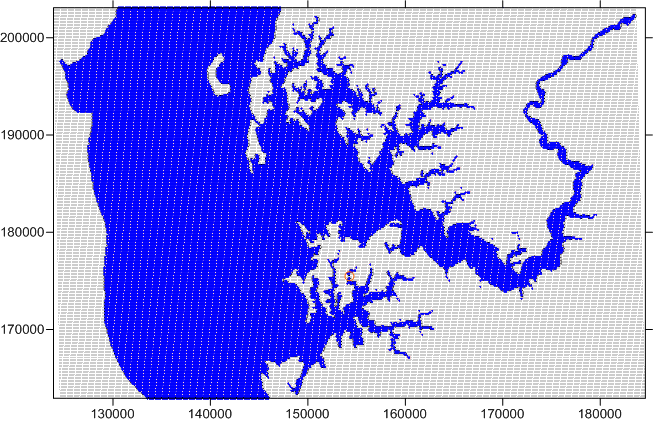


Fig. 1. Missing u-element location (orange circle) in ChopROMS domain.

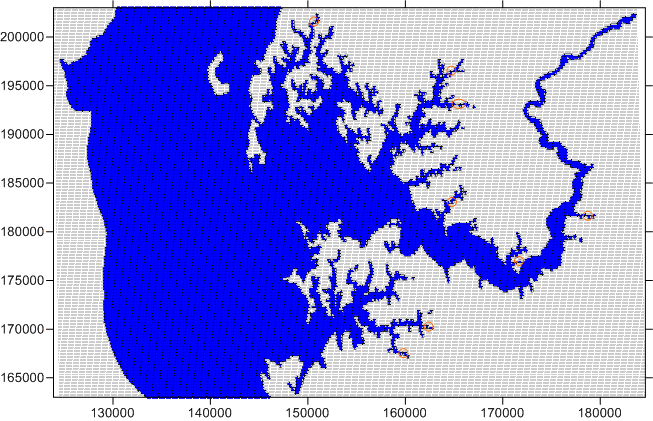


Fig. 2. Missing v-element locations (orange circles) in ChopROMS domain.