CINECA
consorzio interuniversitario

PARADE Debugging

www.cineca.it

Problem description

Constant velocity

Define 6 horizons: *SS, SB, L1, L2, L3, L4*

- Define 5 Layers (regions between horizons)
- Each layer is defined by 2 horizons
- For each layer a **constant velocity** is given

Linear Velocity

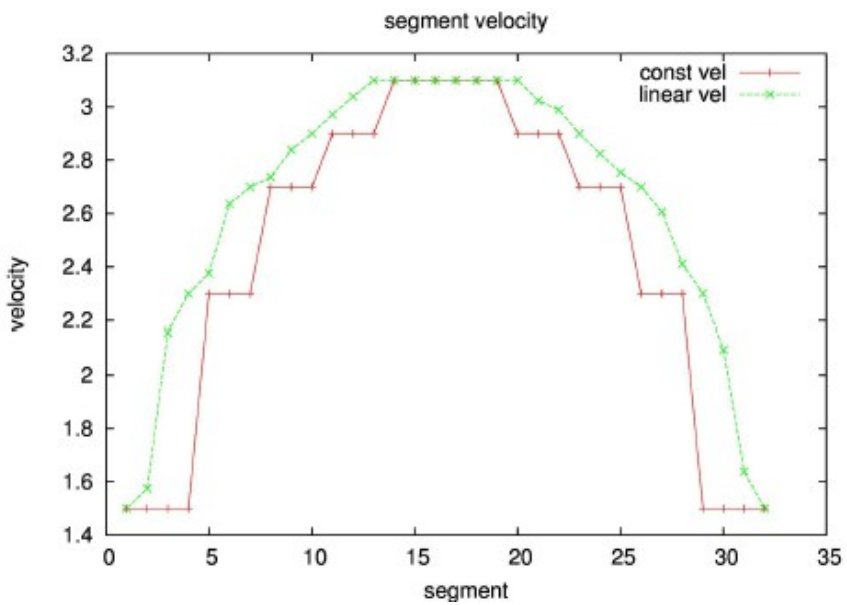
Define 6 horizons: *SS, SB, L1, L2, L3, L4*

- Define 5 Layers (regions between horizons)
- Each layer is defined by 2 horizons
- Define a **top surface velocity** for the top horizon of the layer and a **bottom surface velocity** for the bottom horizon of the layer

Problem:

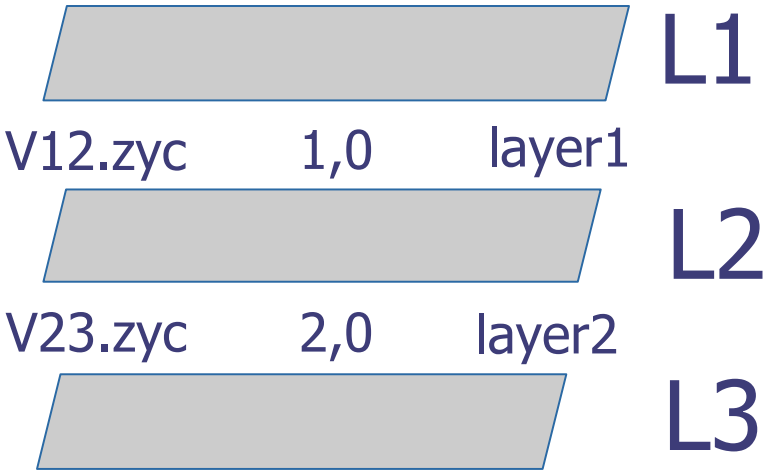
A velocity gradient is found for each tetra element in the model.

----> Recompute linear velocity function coefficients



Case Configuration

Linear velocity




```
<layers>
<layer>
<name>layer1</name>
<id>1</id>
<top-surface>L1</top-surface>
<base-surface>L2</base-surface>
<division>1</division>
<attributes>
<attribute>
<variable-attribute>
<name>P Velocity</name>
<unit>km/s</unit>
<top-surface-attribute-file>V12.zyc</top-surface-attribute-file>
<base-surface-attribute-file>V12.zyc</base-surface-attribute-file>
<rt-model-velocity-type>linear-velocity</rt-model-velocity-type>
</variable-attribute>
</attribute>
</attributes>
</layer>
<layer>
<name>layer2</name>
<id>2</id>
<top-surface>L2</top-surface>
<base-surface>L3</base-surface>
<division>1</division>
<attributes>
<attribute>
<variable-attribute>
<name>P Velocity</name>
<unit>km/s</unit>
<top-surface-attribute-file>V23.zyc</top-surface-attribute-file>
<base-surface-attribute-file>V23.zyc</base-surface-attribute-file>
<rt-model-velocity-type>linear-velocity</rt-model-velocity-type>
</variable-attribute>
</attribute>
</attributes>
</layer>
</layers>
```

computeTetraVelFunction

File rtLinearVelModel.C:

```
void rtLinearVelModel::computeTetraVelFunction(int tetra_id, int layer_flag, const
basArray1D<cgcPoint>& node_coords, const mmsNonStrucAttribute<double>* pvels) {
  //get the tetra element
  const rtTetra& tetra=d_mesh->getTetra(tetra_id);
  double vel[4];
  for(int i=0; i<4; ++i) {
 const rtSurfacePointAttribute *attr=d_surfaceset->getVertexAttribute(tetra.getNodeId(i))

if( attr ) {  //node is on the surface
       if(this->getSurfaceSet()->getLayerIDPointedByNormal(tetra.getNodeId(i))<=0) {
          vel[i]=attr->getPVelocityTop();
       } else {
            if(this-
>getSurfaceSet()>getLayerIDPointedByNormal(tetra.getNodeId(i))==tetra.getLayerId()) {
                    vel[i]=attr->getPVelocityTop();
            }
            else {
                    vel[i]=attr->getPVelocityBot();
             }
      }
  } else { //node is not on the surface
      vel[i]=(*pvels)[tetra.getNodeId(i)];
    } }
```

Never executed!

computeTetraVelFunction

computeTetraVelFunction gets vertex attributes for all vertex on geological surface

```
const rtSurfacePointAttribute *attr=d_surfaceset->
getVertexAttribute(tetra.getNodeId(i));

if( attr ) {  //node is on the surface
...
```

??

attr is always equal to NULL pointer, why??

computeTetraVelFunction is called before setting vertex attributes

Let's take a look to source code…

computeTetraVelFunction

File rtLayerModelBuilder.C:

```
void rtLayerModelBuilder::setModelAttributes(mmsLayerModelBuilder& builder) {
…
…
  d_model_p->setAttribute(rt_attrs);
#ifdef DEBUG
  cout<<"Set attributes for all surface points of curvelinear tetra
model"<<endl;
#endif
  if(attr_type!=mmsAttribute::CELL_TYPE) {
    rtSurfaceSet *surf_set=(rtSurfaceSet*)(d_model_p->getSurfaceSet());
    rtSurfacePointAttribute pa;
    //now set attributes for those vertices located on the surface
   …
   …
if(flag&&surf_set->getVertexNormal(i)) {
const basArray1DNumeric<double>& pvels=vel->getAttributeSample(i);
pa.setPVelocityTop(pvels[0]);
pa.setPVelocityBot(pvels[pvels.size()-1]);
…
…
surf_set->addVertexAttribute(i, pa);
```

1

2

computeTetraVelFunction

File rtLinearVelModel.h:

```
virtual void setAttribute(rtAttribute<double>* attribute) {
    d_attribute=attribute;
    if(d_attribute->getVelocityType()!=rtVelocityType::LINEAR_VELOCITY)
throw rtException("inconsistent velocity type between model and
attribute");
    if(d_attribute->getPVelocityAttribute()->getAttrType()!
=mmsAttribute::NODE_TYPE) {
      throw rtException("Mesh attribute type must be node type",
rtException::INVALID_ATTRIBUTE_TYPE);
    }
    //computer linear function velocity function for each tetra
elements of the mesh
    this->computeTetraVelFunctions();
  }
```

1

Vertex attributes have not yet been initialized!!

Let us modify the source code

computeTetraVelFunction

File rtLinearVelModel.h:

```
virtual void setAttribute(rtAttribute<double>* attribute) {
    d_attribute=attribute;
    if(d_attribute->getVelocityType()!=rtVelocityType::LINEAR_VELOCITY) throw
rtException("inconsistent velocity type between model and attribute");
    if(d_attribute->getPVelocityAttribute()->getAttrType()!
=mmsAttribute::NODE_TYPE) {
      throw rtException("Mesh attribute type must be node type",
rtException::INVALID_ATTRIBUTE_TYPE);
    }
 this->computeTetraVelFunctions();
}
```

Remove computeTetraVelFunction calling from setAttribute function

computeTetraVelFunction

File rtLayerModelBuilder.C:

```
void rtLayerModelBuilder::setModelAttributes(mmsLayerModelBuilder& builder)
{
…
…
  d_model_p->setAttribute(rt_attrs);
#ifdef DEBUG
  cout<<"Set attributes for all surface points of curvelinear tetra
model"<<endl;
#endif
  if(attr_type!=mmsAttribute::CELL_TYPE) {
    rtSurfaceSet *surf_set=(rtSurfaceSet*)(d_model_p->getSurfaceSet());
    rtSurfacePointAttribute pa;
    //now set attributes for those vertices located on the surface
   …
   …
if(flag&&surf_set->getVertexNormal(i)) {
const basArray1DNumeric<double>& pvels=vel->getAttributeSample(i);
pa.setPVelocityTop(pvels[0]);
pa.setPVelocityBot(pvels[pvels.size()-1]);
…
surf_set->addVertexAttribute(i, pa);
d_model_p->computeTetraVelFunctions();
```

Insert computeTetraVelFunction calling after addVertexAttribute function

computeTetraVelFunction

File rtLinearVelModel.C:

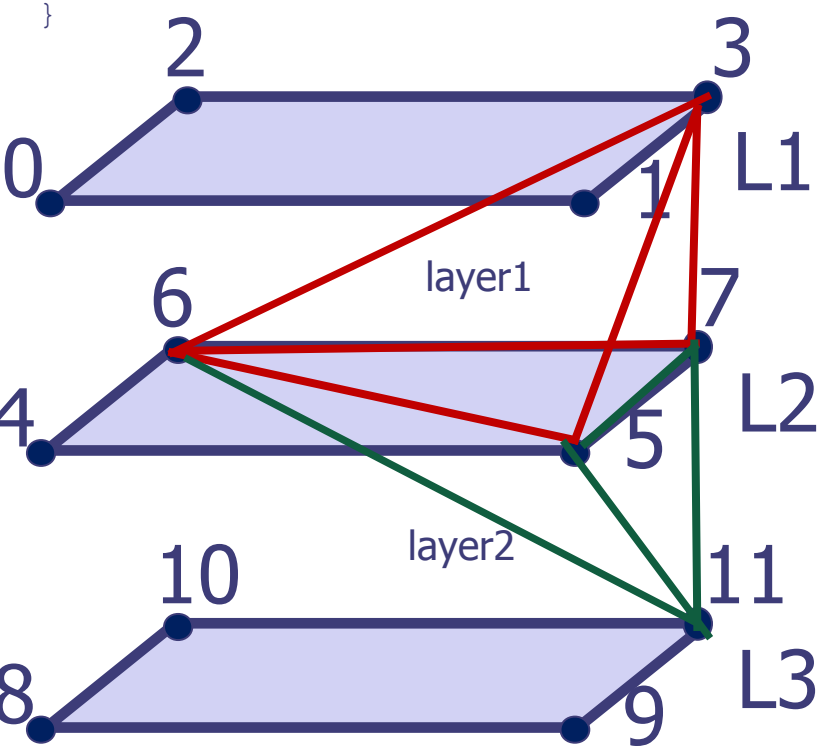
```
void rtLinearVelModel::computeTetraVelFunction(int tetra_id, int layer_flag, const
basArray1D<cgcPoint>& node_coords, const mmsNonStrucAttribute<double>* pvels) {
  //get the tetra element
  const rtTetra& tetra=d_mesh->getTetra(tetra_id);
  double vel[4];
  for(int i=0; i<4; ++i) {
 const rtSurfacePointAttribute *attr=d_surfaceset-
>getVertexAttribute(tetra.getNodeId(i));
if( attr ) {  //node is on the surface
       if(this->getSurfaceSet()->getLayerIDPointedByNormal(tetra.getNodeId(i))<=0) {
                          vel[i]=attr->getPVelocityTop();
         }else {
         if(this->getSurfaceSet()-
>getLayerIDPointedByNormal(tetra.getNodeId(i))==tetra.getLayerId()) {
                  vel[i]=attr->getPVelocityTop();
         }
else  vel[i]=attr->getPVelocityBot();
      }
}
else { //node is not on the surface
      vel[i]=(*pvels)[tetra.getNodeId(i)];
    }
  }
```

Correctly executed! attr is not NULL pointer now

Surface Vertex Attribute

Linear velocity function coefficients are now computed in the right way but surface vertex attributes are not set correctly…

```
void rtLayerModelBuilder::setModelAttributes(mmsLayerModelBuilder& builder) {
…
surf_set->addVertexAttribute(i, pa);
cout<<"Vertex Attribute for vertex "<<i<<" is "<<*(surf_set->getVertexAttribute(i))<<endl;
}
```



Vertex Attribute for vertex **3** is
p-velocity on the top of surface:1
p-velocity on the bottom of surface: 1

Vertex Attribute for vertex **5** is
p-velocity on the top of surface:2
p-velocity on the bottom of surface: 2

Vertex Attribute for vertex **6** is
p-velocity on the top of surface:2
p-velocity on the bottom of surface: 2

Vertex Attribute for vertex **7** is
p-velocity on the top of surface:2
p-velocity on the bottom of surface: 2

Vertex Attribute for vertex **11** is
p-velocity on the top of surface:2
p-velocity on the bottom of surface: 2

Error!!

Surface Vertex Attribute

Where are vertex attributes first defined?

`rtLayerModelBuilder`→
`mmsLayerModelBuilder`→
`mmsCurveLinearMeshedLayerModelBuilder`→
`build()`→
`createAttributes()`

Surface Vertex Attribute

File mmsCurvelinearMeshedLayerModelBuilder.C:

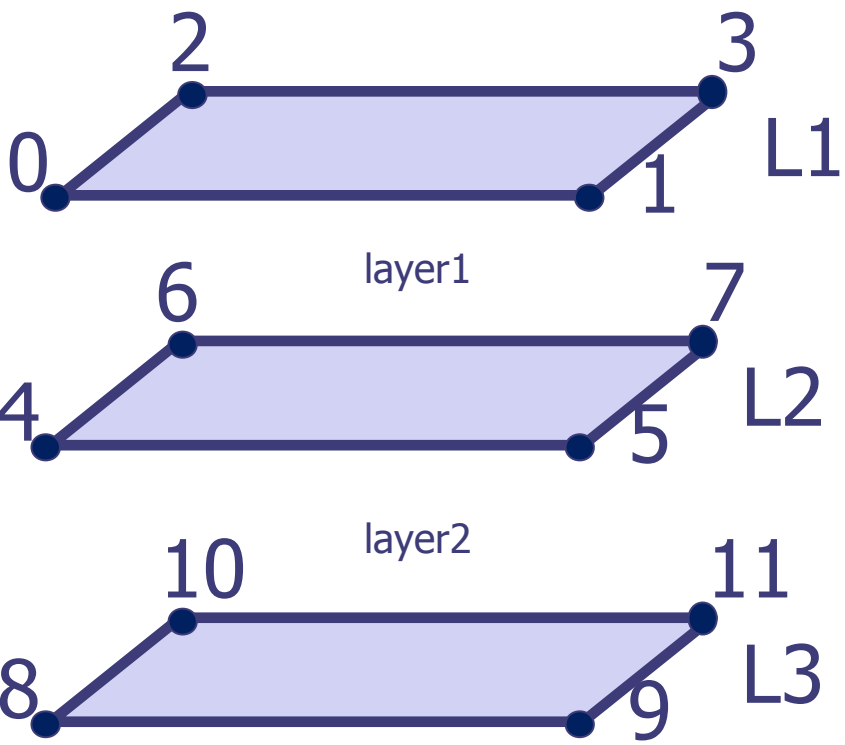
```
void mmsCurvelinearMeshedLayerModelBuilder::createAttributes()
throw(std::runtime_error) {
  …
 for(i=0; i<this->layers_spec.size(); ++i) {
   …
for(ix=0; ix<nx; ++ix) {
  for(iy=0; iy<ny; ++iy) {
    mesh.getCoordinates(ix,iy,nz1,p1);
    mesh.getCoordinates(ix,iy,nz2,p2);
    dz=(p2[2]-p1[2]);
    if(fabs(dz)<1.0e-7) coef=1;
    else {
      /** z coordinate of the surface has the same value of attribute for mmsGSurfField */
      (attr_top->getMesh()).getCoordinates(ix, iy, s1);
      (attr_base->getMesh()).getCoordinates(ix, iy, s2);
      coef=(s2[2]-s1[2])/dz;
}
    for(iz=nz1; iz<=nz2; ++iz) {
      mesh.getCoordinates(ix,iy,iz,p);
      v[0]=coef*(p[2]-p1[2])+s1[2];
      v[1]=v[0];
      …
      attr->setAttributeSample(v, ix, iy, iz);
      cout<<"Set Attribute for ("<<ix<<","<<iy<<","<<iz<<") "<<v<<endl;
```

Surface Vertex Attribute

```
cout<<"Set Attribute for ("<<ix<<","<<iy<<","<<iz<<") "<<v<<endl;
    }
basArray1DNumeric<double>& samples1=attr->getAttributeSample(ix,iy,nz1);
    //cout<<"Point "<<ix<<iy<<nz1<<endl;
    if(nz1==0) samples1[0]=s1[2];
    samples1[1]=s1[2];
#ifdef DEBUG
    if(samples1[0]<min_val) min_val=samples1[0];
    if(samples1[0]>max_val) max_val=samples1[0];
    //cout<<"Attribute value at z index "<<nz1<<": "<<samples1<<" s1: "<<s1<<endl;
#endif

basArray1DNumeric<double>& samples2=attr->getAttributeSample(ix,iy,nz2);
    if(nz2==nz-1||nz1==nz2-1) {
          samples2[1]=s2[2];
        }
    samples2[0]=s2[2];
#ifdef DEBUG
    if(samples2[0]<min_val) min_val=samples2[0];
    if(samples2[0]>max_val) max_val=samples2[0];
#endif*/
  }
}
```

Surface Vertex Attribute



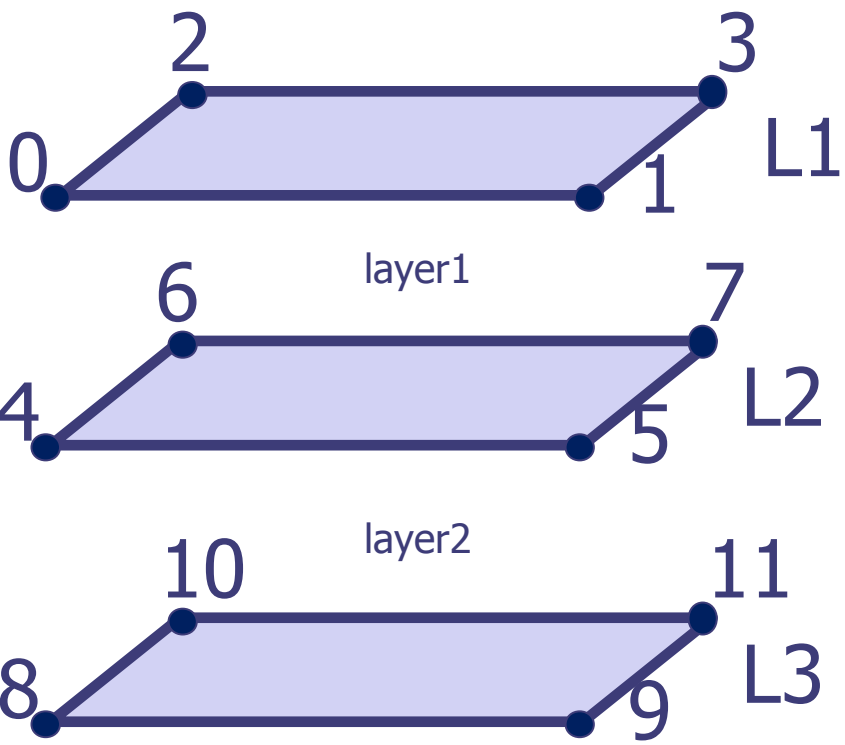
Vertex Id	Coord	Pvelocity 1 cycle	Pvelocity 2 cycle	Final Value
0	0 0 0	1 1		1 1
1	1 0 0	1 1		1 1
2	0 1 0	1 1		1 1
3	1 1 0	1 1		1 1
4	0 0 1	1 1	2 2	2 2
5	1 0 1	1 1	2 2	2 2
6	0 1 1	1 1	2 2	2 2
7	1 1 1	1 1	2 2	2 2
8	0 0 2		2 2	2 2
9	1 0 2		2 2	2 2
10	0 1 2		2 2	2 2
11	1 1 2		2 2	2 2

Vertexes between two layers are visited twice. Top and bottom velocities are updated at each computing cycle.

Surface Vertex Attribute

```
void mmsCurvelinearMeshedLayerModelBuilder::createAttributes()
                                throw(std::runtime_error) {
…
cgcPoint s1_pl, s2_pl;
attr->setAttributeSample(v, ix, iy, iz);
if (count==0){
    s1_pl=s1;
    s2_pl=s2;
    }
    basArray1DNumeric<double>& samples1=attr-
>getAttributeSample(ix,iy,nz1);
if(nz1==0) samples1[1]=s1_pl[2];
    samples1[0]=s1_pl[2];
#ifdef DEBUG
    if(samples1[0]<min_val) min_val=samples1[0];
    if(samples1[0]>max_val) max_val=samples1[0];
#endif
}
}
s1_pl=s1;
s2_pl=s2;
```

Surface Vertex Attribute



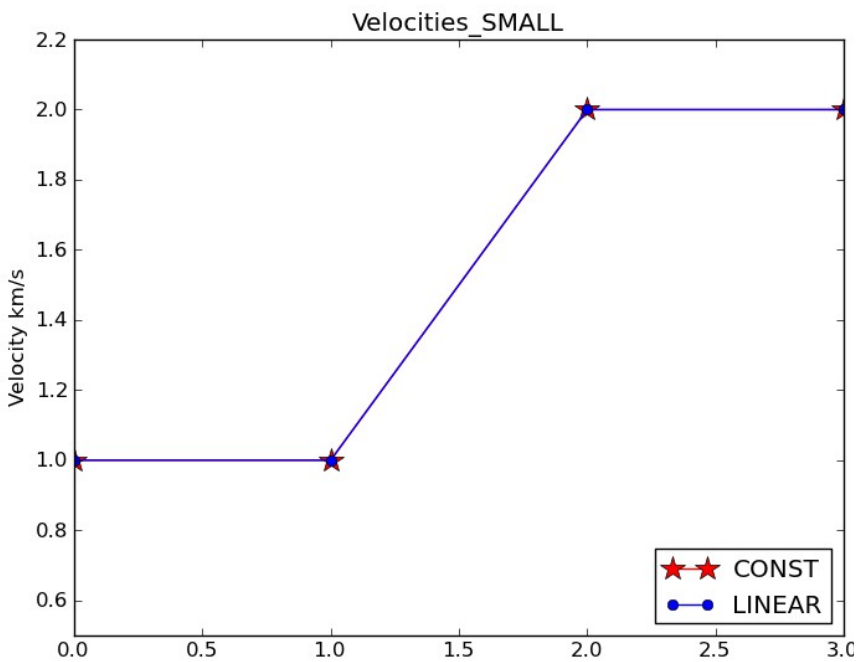
Vertex Id	Coord	Pvelocity 1 cycle	Pvelocity 2 cycle	Final Value
0	0 0 0	1 1		1 1
1	1 0 0	1 1		1 1
2	0 1 0	1 1		1 1
3	1 1 0	1 1		1 1
4	0 0 1	1 1	1 2	1 2
5	1 0 1	1 1	1 2	1 2
6	0 1 1	1 1	1 2	1 2
7	1 1 1	1 1	1 2	1 2
8	0 0 2		2 2	2 2
9	1 0 2		2 2	2 2
10	0 1 2		2 2	2 2
11	1 1 2		2 2	2 2

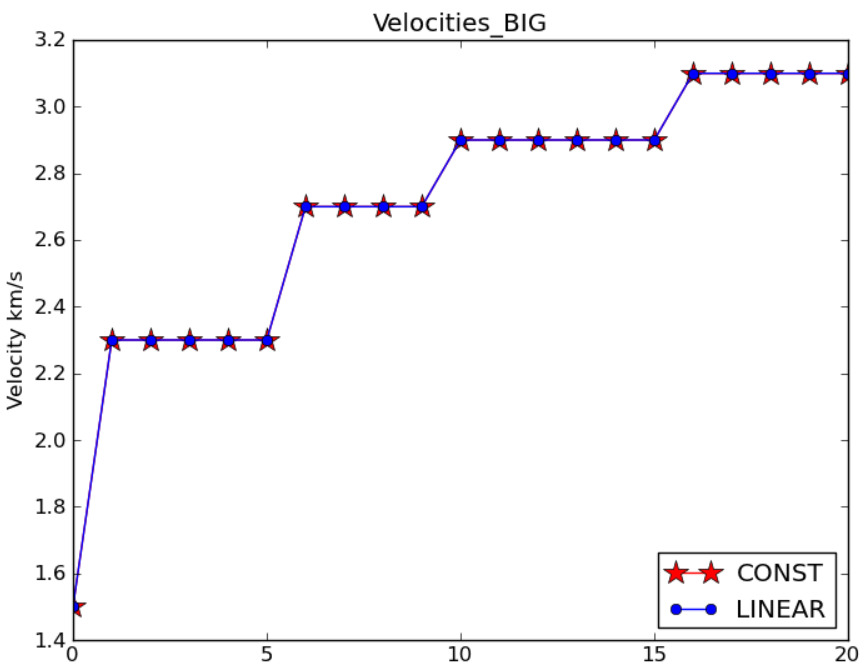
Vertexes between two layers are visited twice. Top and bottom velocities are updated at each computing cycle.

Test case Result

CASE NAME	SURFACE SIZE	#Layer	Division factor
SMALL	2X2	2	1
BIG	56 X 95	5	1

OK





Test case Result

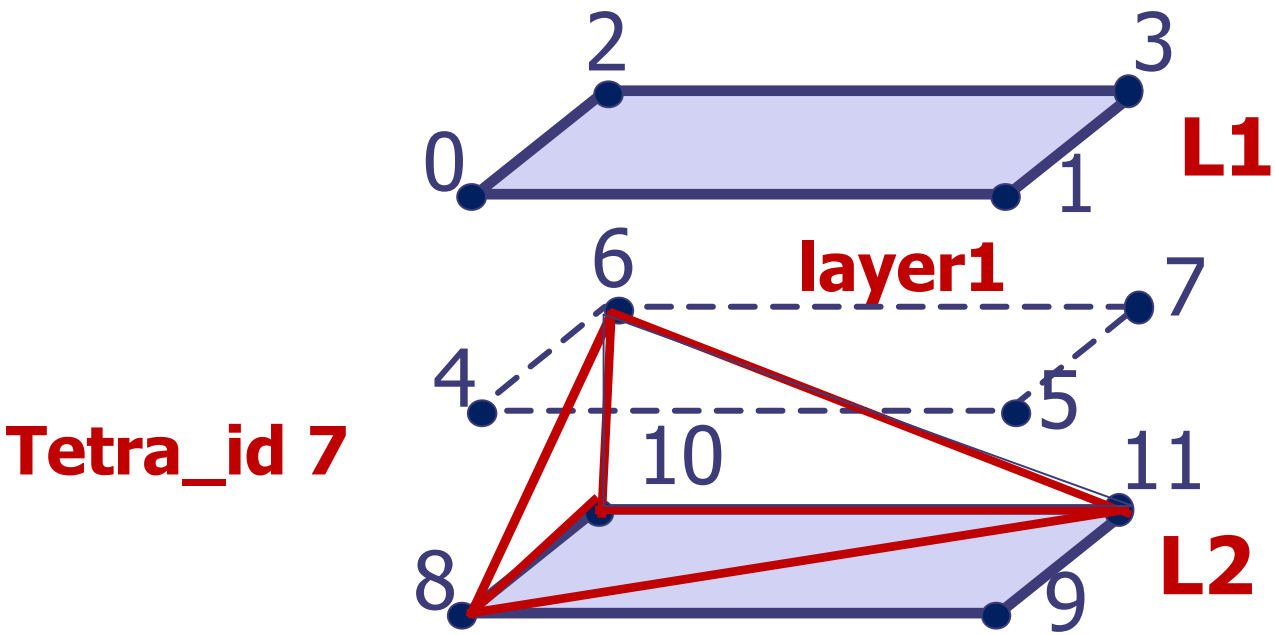
CASE NAME	SURFACE SIZE	#Layer	Division factor	Division factor
SMALL	2X2	2	1	2
BIG	56 X 95	5	1	2

Error!

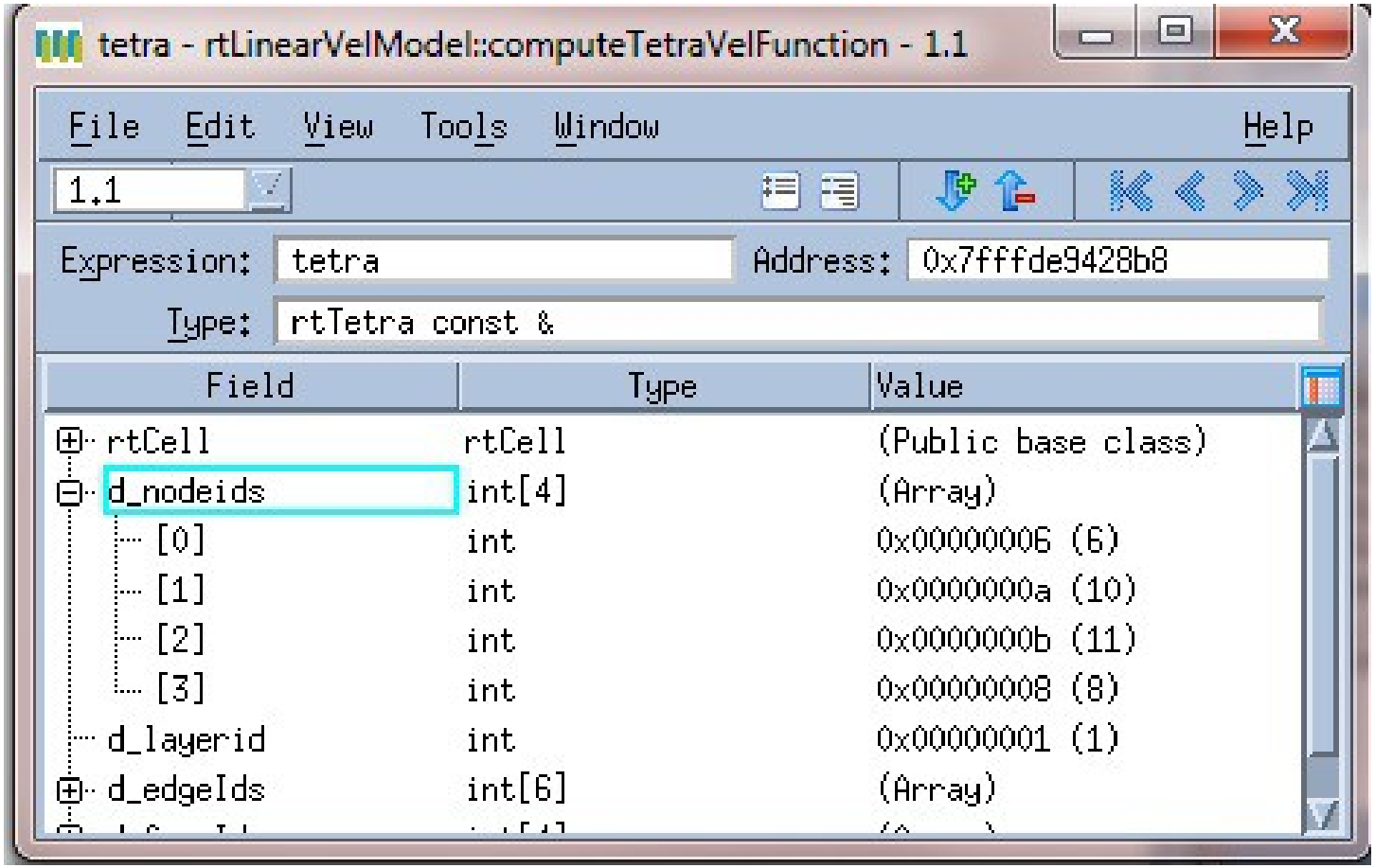
Problem with normals of surface vertex …

Test Case

CASE NAME	SURFACE SIZE	#Layer	Division factor
SMALL	2X2	2	2



Test Case Debugging with Totalview



Test Case Debugging with Totalview

Top and bottom vertex velocities: **OK**

Normals of surface vertex: **KO**

/gpfs/scratch/userinternal/ainverni/MMT/swamp/MMT-BUG/swamp_4.1.2_Work...

File Edit View Window Help

1.1

Expression	Value
tetra.getNodeId(i)	0x00000006 (6)
attr->getPVelocityTop()	1
attr->getPVelocityBot()	1
this->getSurfaceSet()->getLayerIDPointedByNormal(tetra.getNodeId(i))	0x00000001 (1)
tetra.getLayerId()	0x00000001 (1)

/gpfs/scratch/userinternal/ainverni/MMT/swamp/MMT-BUG/swamp_4.1.2_Work...

File Edit View Window Help

1.1

Expression	Value
tetra.getNodeId(i)	0x0000000a (10)
attr->getPVelocityTop()	1
attr->getPVelocityBot()	2
this->getSurfaceSet()->getLayerIDPointedByNormal(tetra.getNodeId(i))	0x00000002 (2)
tetra.getLayerId()	0x00000001 (1)

Test Case Debugging with Totalview

Top and bottom vertex velocities: **OK**

Normals of surface vertex: **KO**

/gpfs/scratch/userinternal/ainverni/MMT/swamp/MMT-BUG/swamp_4.1.2_Work...

File Edit View Window Help

1.1

Expression	Value
tetra.getNodeId(i)	0x0000000b (11)
attr->getPVelocityTop()	1
attr->getPVelocityBot()	2
this->getSurfaceSet()->getLayerIDPointedByNormal(tetra.getNodeId(i))	0x00000002 (2)
tetra.getLayerId()	0x00000001 (1)

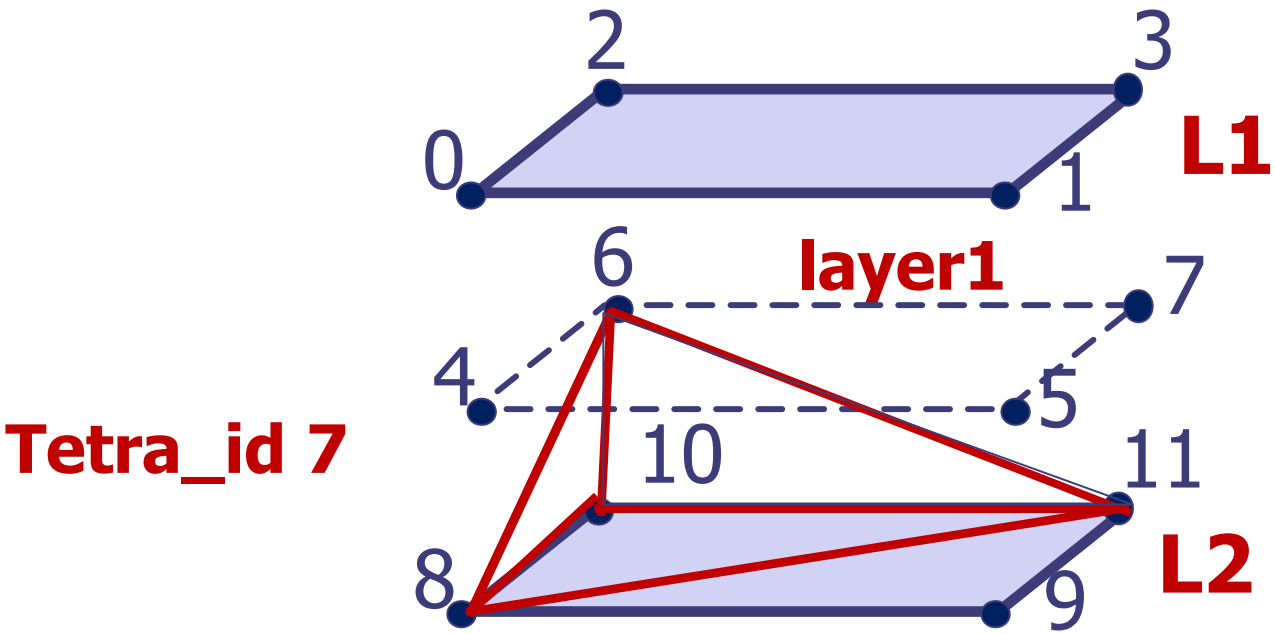
/gpfs/scratch/userinternal/ainverni/MMT/swamp/MMT-BUG/swamp_4.1.2_Work...

File Edit View Window Help

1.1

Expression	Value
tetra.getNodeId(i)	0x00000008 (8)
attr->getPVelocityTop()	1
attr->getPVelocityBot()	2
this->getSurfaceSet()->getLayerIDPointedByNormal(tetra.getNodeId(i))	0x00000001 (1)
tetra.getLayerId()	0x00000001 (1)

Test Case



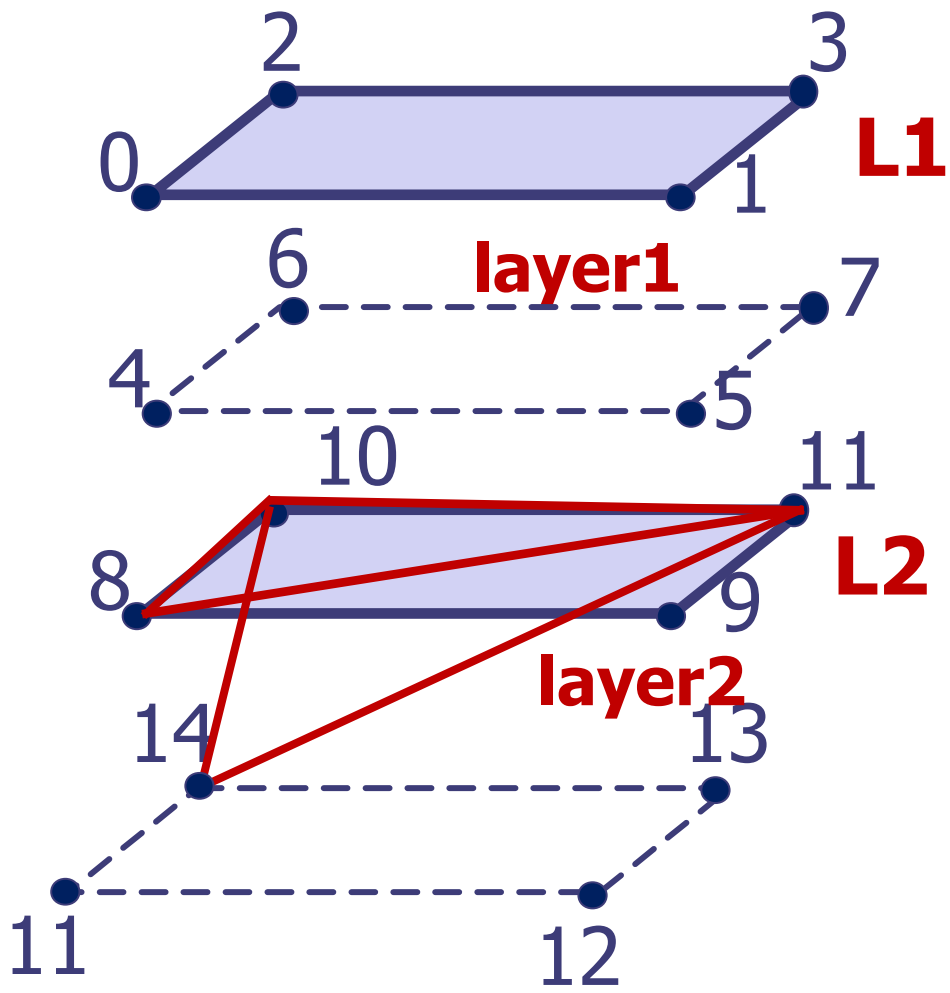
Vertex velocities for Tetra_id 7 = 1.0,2.0,2.0,1.0 →
A velocity gradient is found for this tetra element!!

Normals of surface vertex

File rtLayerModelBuilder.C:

```
rtMesh* rtLayerModelBuilder::createRtMesh(mmsLayerModelBuilder& builder) {
…
CYCLE on EACH TRIANGLE (t) OF THE MESH
//set the layer to which the normal is pointing to
t.setLayerIDPointedByNormal(layerid);
if(data.is_on_surface) {
      if(surf_set->getLayerIDPointedByNormal(t.getNodeId(0))<=0){
surf_set->setLayerIDPointedByNormal(t.getNodeId(0), layerid);
}
      if(surf_set->getLayerIDPointedByNormal(t.getNodeId(1))<=0){
surf_set->setLayerIDPointedByNormal(t.getNodeId(1), layerid);
}
      if(surf_set->getLayerIDPointedByNormal(t.getNodeId(2))<=0){
surf_set->setLayerIDPointedByNormal(t.getNodeId(2), layerid);
}
…
}
```

Normals of surface vertex



First compute triangle [14,10,11]
Then compute triangle [10,8,11]

```
Triangle Cell identifier: 38 node ids:
[14, 10, 11]
Normal: 0 0 0
identifier of layer to which the
normal is pointing: 2
```

```
Triangle Cell identifier: 20 node ids:
[10, 8, 11]
Normal: 0 0 0
identifier of layer to which the
normal is pointing: 1
```

Normals of surface vertex

File rtLinearVelModel.C:

```
void rtLinearVelModel::computeTetraVelFunctions() {
#ifdef DEBUG
  cout<<"compute linear velocity interpolation function:
rtLinearVelModel::computeTetraVelFunctions()"<<endl;
#endif

  d_elementVelFunctions=new mmsNonStrucAttribute<rtTetraVelFunction>("tetra
velfunction", "", mmsAttribute::CELL_TYPE);
  d_elementVelFunctions->allocAttributeBuffer(0, d_mesh->getTetraCount()-1);

  //get node coordinate array
  const basArray1D<cgcPoint>& node_coords=d_mesh-
>getNodeSet().getNodeCoordinates();

  //get p-velocity attributes
  const mmsNonStrucAttribute<double>* pvels=d_attribute-
>getPVelocityAttribute();
  int i;
  double values[4];
  for(i=1; i<d_mesh->getTetraCount(); ++i) {
    const rtTetra& element=d_mesh->getTetra(i);
this->getAttributes(element, pvels, values);
```

1

Normals of surface vertex

File rtModel.C:

1

```
double rtModel::getAttributeAtNode(int i, const rtTetra& tetra,
const mmsNonStrucAttribute<double>* attr, int attr_id) const {
…
//check if this node is on the surface
    if(this->getSurfaceSet()->getVertexAttribute(i)) {
      int layerid=this->getSurfaceSet()-
>getLayerIDPointedByNormal(i);
      const rtSurfacePointAttribute* vt=this->getSurfaceSet()-
>getVertexAttribute(i);
      if(layerid<=0 || layerid==tetra.getLayerId()) {
//return attribute on the top of surface
switch(attr_id) {
case 0:
  return vt->getPVelocityTop();
…}
} else {
switch(attr_id) {
case 0:
  return vt->getPVelocityBot();
…
}
```

Normals of surface vertex

File rtLinearVelModel.C:

```
void rtLinearVelModel::computeTetraVelFunctions(){

…
for(; surface_it!=surfaces.end(); ++surface_it) {  //loop through each
surface
    const rtSurface& surface=(*surface_it).second;
    const rtSurface::TriangleContainer& triangles=surface.getTriangles();
    rtSurface::TriangleContainer::const_iterator
triangle_it=triangles.begin();
    for(; triangle_it!=triangles.end(); ++triangle_it) {
      const rtTriangle& triangle=(*triangle_it).second
      int tetra_id=triangle.getElementId(0);
      if(tetra_id!=0 && triangle.getElementId(1)!=0) {
this->computeTetraVelFunction(tetra_id, 0, node_coords, pvels);
tetra_id=triangle.getElementId(1);
this->computeTetraVelFunction(tetra_id, 1, node_coords, pvels);
```

2

Normals of surface vertex

File rtLinearVelModel.C:

```
void rtLinearVelModel::computeTetraVelFunction(int tetra_id, int layer_flag, const
basArray1D<cgcPoint>& node_coords,
       const mmsNonStrucAttribute<double>* pvels) {
  //get the tetra element
  const rtTetra& tetra=d_mesh->getTetra(tetra_id);

  double vel[4];
  int i;
for(i=0; i<4; ++i) {
const rtSurfacePointAttribute *attr=d_surfaceset->getVertexAttribute(tetra.getNodeId(i));
    if( attr ) {  //node is on the surface
      if(this->getSurfaceSet()->getLayerIDPointedByNormal(tetra.getNodeId(i))<=0) {
vel[i]=attr->getPVelocityTop();
      } else {
      if(this->getSurfaceSet()-
>getLayerIDPointedByNormal(tetra.getNodeId(i))==tetra.getLayerId()) {
vel[i]=attr->getPVelocityTop();
      }
      else vel[i]=attr->getPVelocityBot();
      }
} else { //node is not on the surface
      vel[i]=(*pvels)[tetra.getNodeId(i)];
    }
  }
```

2