

# Parade MMT: debugging activity

A horizontal strip of a photograph showing a group of people sitting around a table in a meeting room, engaged in discussion. The image is semi-transparent and serves as a background for the contact information.

F. Cinquini  
[f.cinquini@cenea.it](mailto:f.cinquini@cenea.it)

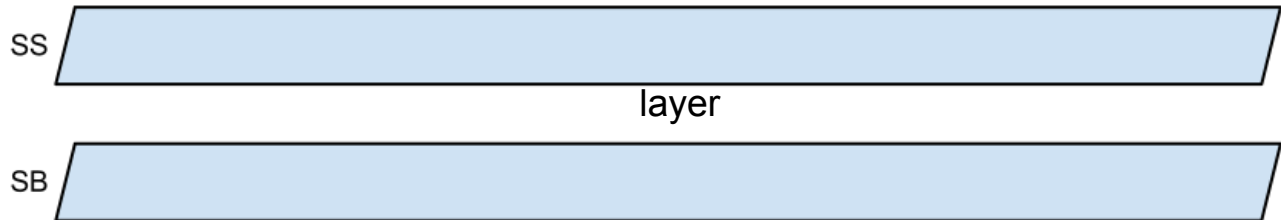


- Introduction
  - Context
- Getting started
  - Compiling the code
  - Input: Constant velocity
  - Input: Linear Velocity (map)
- Testing
  - 5 layer case: output comparison
  - Debugging: Totalview
  - Hypothesis: variable reference?
- Perspectives



*Parade:*

- **Model Builder**
- Ray Tracer
- Output Reader



SS and SB are horizons, a layer has specific velocity attributes defined by a map or a constant value.

Tetrahedrons are build in the layer between horizons: a ray is traced from one horizon to another through the tetrahedrons.

The ray will be divided into segments with their own length and velocities depending from the input velocities.

A first bug in the model builder has been fixed and the code works for a single layer case.  
(A wrong variable reference in the file `mmsCurvelinearMeshedModelBuilder.cpp`)

For  $n \text{ layer} > 1$ , the results obtained with a constant velocity and a map of constant velocity are not consistent.



# Compiling Parade

The code have been compiled on PLX machine.

Configure arguments:

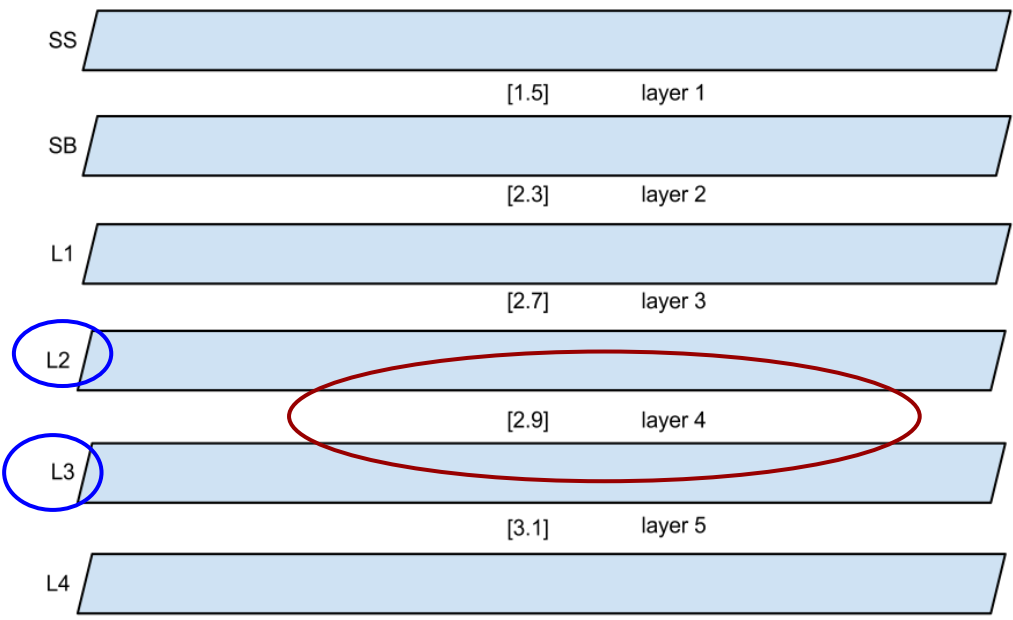
```
./configure --with-mode=dbg --with-cxx=g++ --with-w64 --with-lib64 --with-stl --enable-shared --with-thread --with-ogl --with-x --with-python --with-pio --with-rt --with-spatialindex
```

Remarks:

- OK with gnu C++ compiler v. 4.1.2 or older.
- Error with gnu C++ compiler v. 4.4.6 or newer: an include has to be modified (.
- To compile on eni0x and on PLX **"tk"** support should be disabled (tk libraries are too recent)

simple case - 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



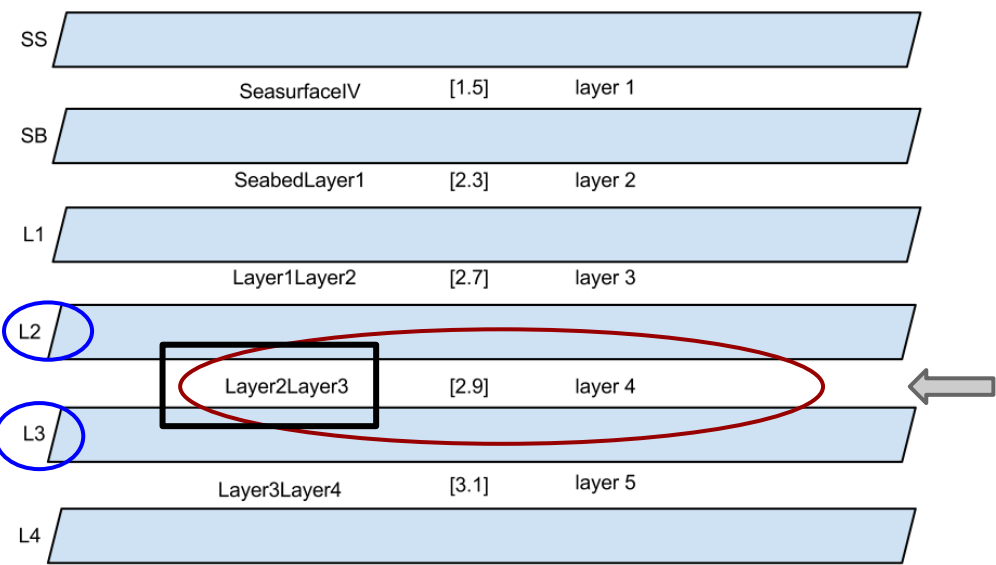
Layer definition example:

```
<layer>
<name>layer4</name>
<id>4</id>
<top-surface>L2</top-surface>
<base-surface>L3</base-surface>
<attributes>
<attribute>
<constant-attribute>
<name>P Velocity</name>
<unit>km/s</unit>
<value>2.9</value>
<rt-model-velocity-type>velocity</rt-model-velocity-type>
</constant-attribute>
</attribute>
</attributes>
</layer>
```

Constant velocity tracer  
is employed by default

complex case - 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



Layer definition example:

```

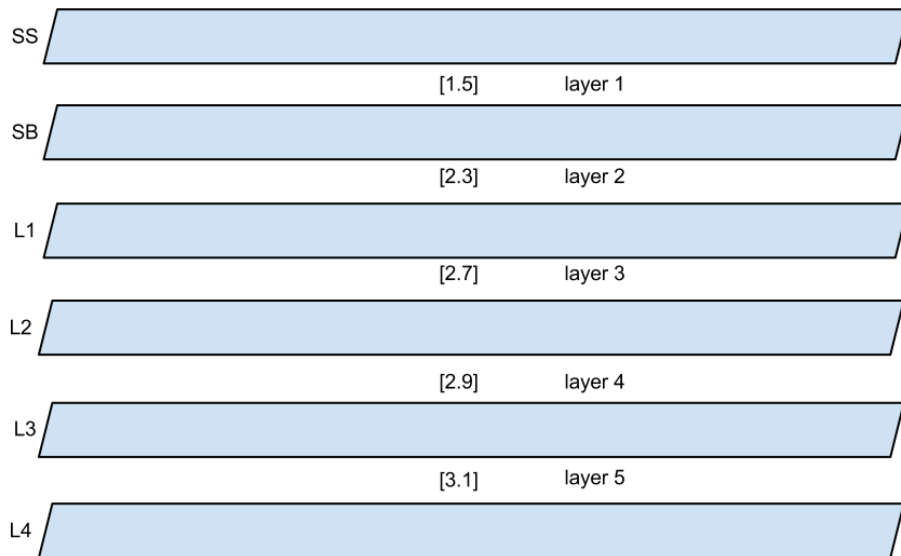
<layer>
<name>layer4</name>
<id>4</id>
<top-surface>L2</top-surface>
<base-surface>L3</base-surface>
<division>1</division>
<attributes>
<attribute>
<variable-attribute>
<name>P Velocity</name>
<unit>km/s</unit>
<top-surf-attr-file>layer2layer3IV.zyc</top-surf-attr-file>
<base-surf-attr-file>layer2layer3IV.zyc</base-surf-attr-file>
<rt-model-velocity-type>linear-velocity</rt-model-velocity-type>
</variable-attribute>
</attribute>
</attributes>
</layer>
  
```

Function: ComputeTetraVelFunction  
(Recompute linear velocity function coefficients)

IF NO GRADIENT IS FOUND  
FOR A TETRA ELEMENT  
--> back to constant velocity tracer

# Test case: different output

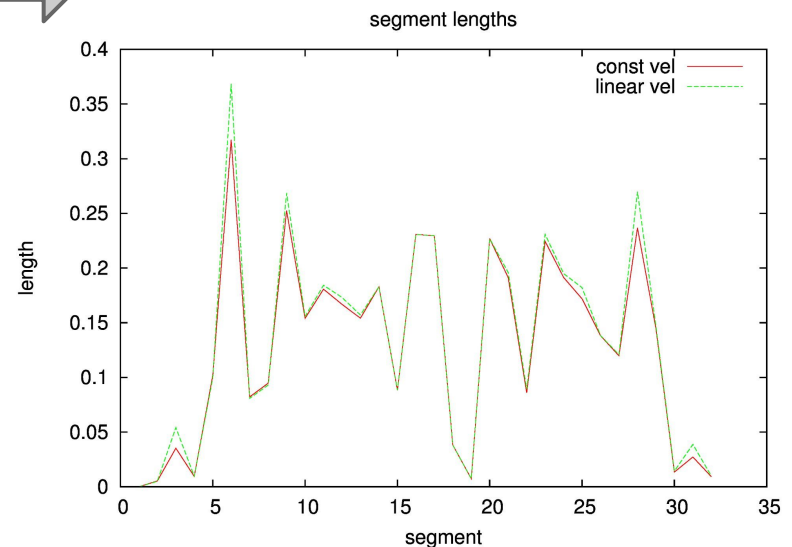
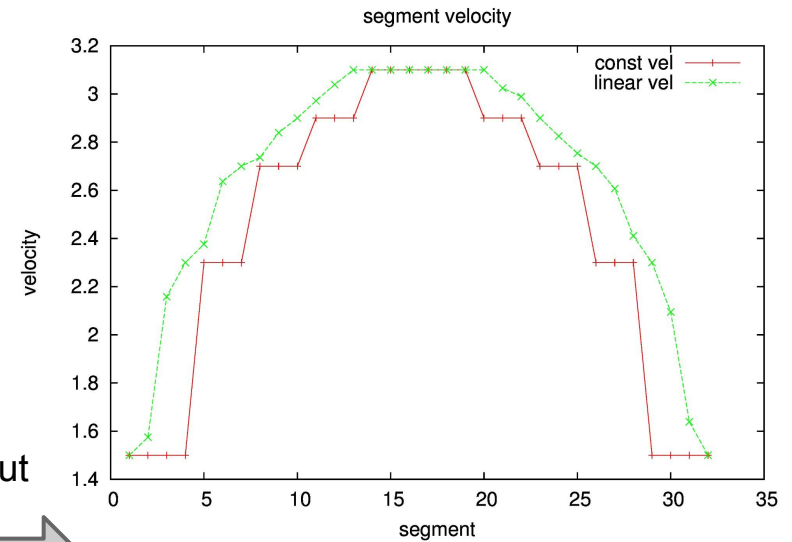
Input



Linear velocities case:  
Function: ComputeTetraVelFunction  
(Recompute linear velocity function coefficients)

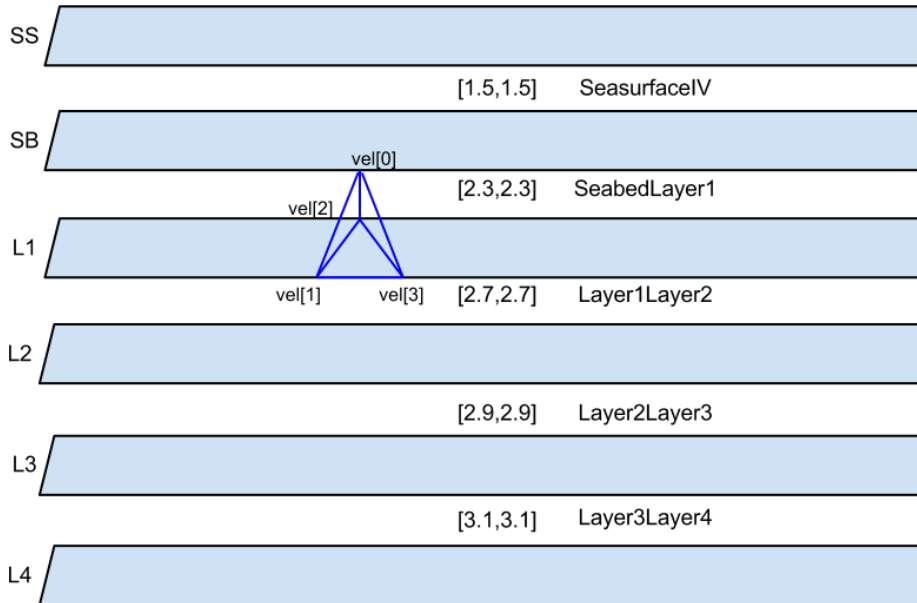
A GRADIENT IS FOUND FOR TETRA ELEMENTS

Output





# Debugging: Totalview



File Edit View Tools Window	File Edit View Tools Window	File Edit View Tools Window	File Edit View Tools Window
1.1	1.1	1.1	1.1
Expression: vel[0]	Expression: vel[1]	Expression: vel[2]	Expression: vel[3]
Type: double	Type: double	Type: double	Type: double
Value	Value	Value	Value
2.3	2.7	2.7	2.7

Debugger window showing the execution of a program. The title bar indicates the path: /gpfs/scratch/userinternal/fcinquin/swamp/MMT-DEBUG/swamp/rt/bin/rtLayerModelBuilder.

The main window displays the source code of the function `rtLinearVelModel::computeTetraVelFunction`. A red arrow points to the line `for(i=0; i<4; ++i) {`. Another red arrow points to the line `//cout<<"v1: "<<vel[0]<<" v2="<<vel[1]<<" v3="<<vel[2]<<" v4="<<vel[3]<<endl;`.

The stack trace shows the function `rtLinearVelModel::computeTetraVelFunction` is the active function. The stack frame shows the following variables:

- `this:` `0x12091870 -> (class rtLinearVelModel)`
- `tetra_id:` `0x000064fc (25852)`
- `layer_flag:` `0x00000000 (0)`
- `node_coords:` `(class basArrayID<cgcpPoint> const)`
- `pvels:` `0x125b2040 -> (class mmsNonStruc)`

The local variables section shows:

- `tetra:` `(rtTetra const &)`
- `vel:` `(double[4])`
- `i:` `0x00000003 (3)`

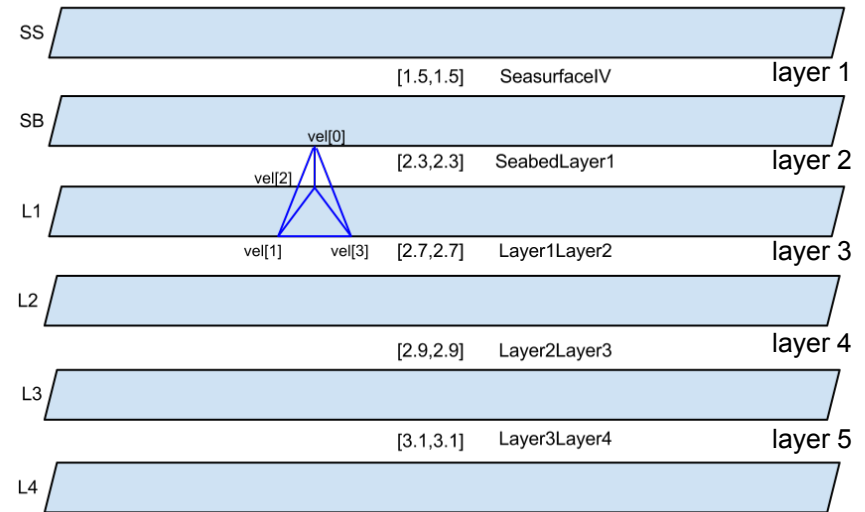
The Action Points section shows the function `computeTetraVelFunction` is the active function.





# Variables storage

```
<layer>
<name>layer2</name>
<id>2</id>
<top-surface>SB</top-surface>
<base-surface>L1</base-surface>
<division>1</division>
<attributes>
<attribute>
<variable-attribute>
<name>P Velocity</name>
<unit>km/s</unit>
<top-surf-attr-file>Seabedlayer11V.zyc</top-surf-attr-file>
<base-surf-attr-file>Seabedlayer11V.zyc</base-surf-attr-file>
<rt-model-velocity-type>linear-velocity</rt-model-velocity-type>
</variable-attribute>
</attribute>
</attributes>
</layer>
<layer>
<name>layer3</name>
<id>3</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-surf-attr-file>layerlayer21V.zyc</top-surf-attr-file>
<base-surf-attr-file>layerlayer21V.zyc</base-surf-attr-file>
<rt-model-velocity-type>linear-velocity</rt-model-velocity-type>
</variable-attribute>
</attribute>
</attributes>
</layer>
```



Horizon L1 keeps the last velocity defined  
(as top surface of **layer 3**).

The tetrahedron vertices assume different  
values along z --> a gradient is found



Output velocities are distributed as a gradient



- Hands on the code:
  - **check the** first bug-fix
  - **verify analogies** with other parts of the source code  
(same variable references to be modified?)

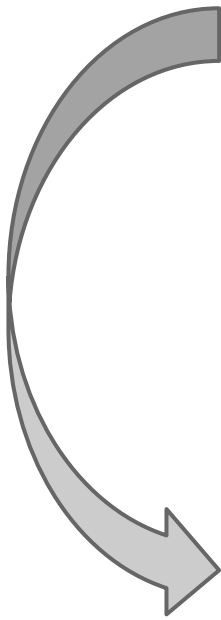


# A step back: bug report

**mmsCurvelinearMeshedLayerModelBuilder.C** have been modified.

```
basArray1DNumeric& samples=attr->getAttributeSample(ix,iy,nz1);  
    samples[1]=s1[2];
```

```
    samples=attr->getAttributeSample(ix,iy,nz2);  
    mesh.getCoordinates(ix,iy,nz2,p);  
    samples[0]=s2[2];
```



```
basArray1DNumeric& samples1=attr->getAttributeSample(ix,iy,nz1);  
    samples1[1]=s1[2];
```

```
basArray1DNumeric& samples2=attr->getAttributeSample(ix,iy,nz2);  
    samples2[0]=s2[2];
```

**Is it possible that the same variable is defined somewhere else in the code?**