Tutorial by Felicia Brisc, CEN/Universität Hamburg

A. Prepare your data: the mandatory dimensions, variables and attributes

- A trajectory dimension with the attribute: **`cf_role="trajectory_id"`**, with values uniquely identifying the trajectories – as strongly recommended by the CF-Conventions
- For all variables defined on the particles, the **`coordinates`** attribute

Conforming to the CF-Conventions:
- A time dimension with the attributes **`standard_name, units`** and **`calendar`**
- Longitude and latitude variables with the attributes **`standard_name`** and **`units`**
- Data sets that have a vertical dimension, the attributes: **`units`** and **`positive`**, the **`positive`** attribute with a value of **`up`** or **`down`**.

The mandatory meta tags are displayed in bold and magenta in the simple example below:

```
netcdf SimpleExample {

dimensions:
        time = UNLIMITED ; // (4 currently)
        ntraj = 3 ;
variables:
        float ntraj(ntraj) ;
                ntraj:cf_role = "trajectory_id" ;
                ntraj:long_name = "trajectories_no" ;

        float time(time) ;
                time:standard_name = "time" ;
                time:units = "days since 1994-01-01 12:00:00" ;
                time:calendar = "proleptic_gregorian" ;

        float depth(time, ntraj) ;
                depth:units = "m" ;
                depth:positive = "down" ;
                depth:coordinates = "longitude latitude" ;
//The coordinates attribute is not mandatory here, but adding them to the
//vertical dimension, will identify it as being also a variable defined on
//the particles - so that it will be possible to color the data by depth

        float latitude(time, ntraj) ;
                latitude:standard_name = "latitude" ;
                latitude:units = "degrees_north" ;
                latitude:long_name = "latitude" ;

        float longitude(time, ntraj) ;
                longitude:standard_name = "longitude" ;
                longitude:units = "degrees_east" ;
                longitude:long_name = "longitude" ;

        float salinity(time, ntraj) ;
                salinity:coordinates = "longitude latitude depth" ;
```

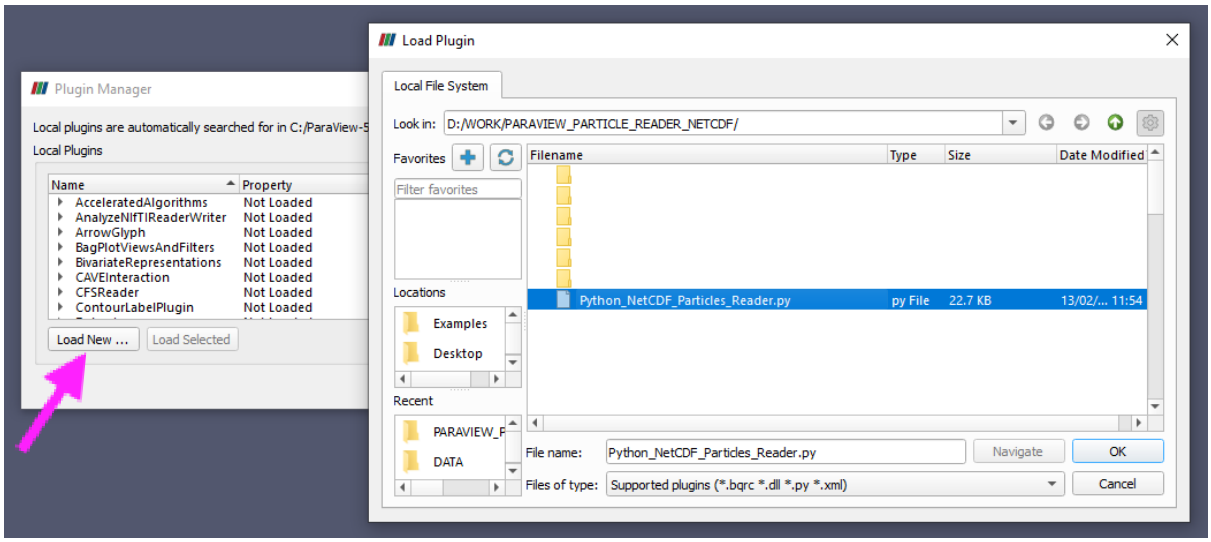
```
data:

ntraj = 1,2,3;

time =  1, 2, 3, 4;

depth = 100, 2000, 5000,
        100, 2000, 5000,
        100, 2000, 5000,
        100, 2000, 5000;

latitude = 20, 40, 60,
           25, 45, 65,
           30, 50, 70,
           35, 55, 75;

longitude = 60, 65, 70,
            61, 66, 71,
            62, 67, 72,
            63, 68, 73;

salinity = 28, 30, 35,
           28, 32, 37,
           30, 32, 31,
           32, 28, 35;
}
```

B. Adding the plugin to ParaView

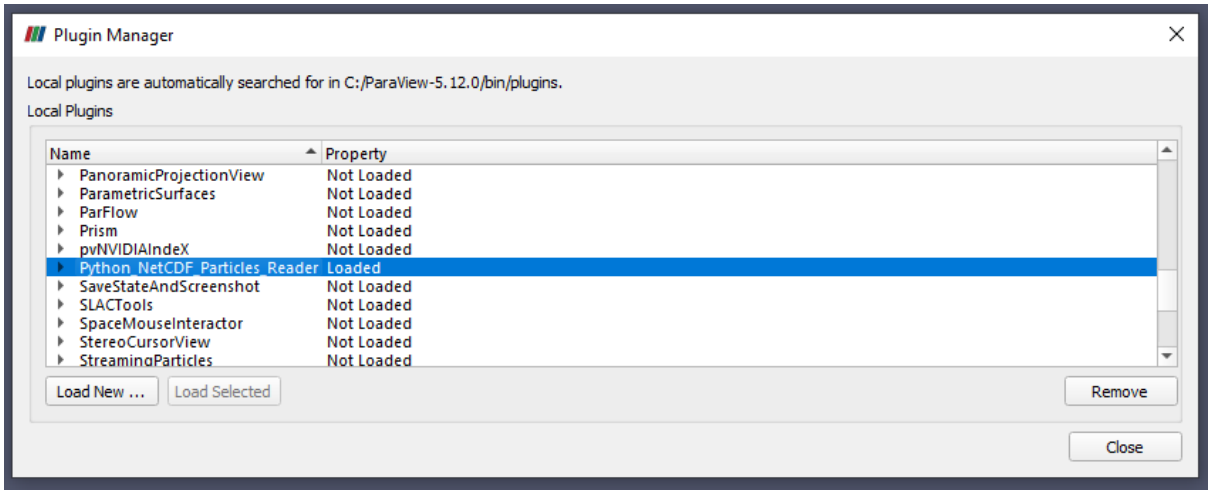
B 1. First install the netcdf4-python module, and either make sure its libraries are in the system path or add the libraries to the system path directly in the plugin Python script with sys.path.append(...)

B 2. Copy the **Python_NetCDF_Particles_Reader.py** in a local folder

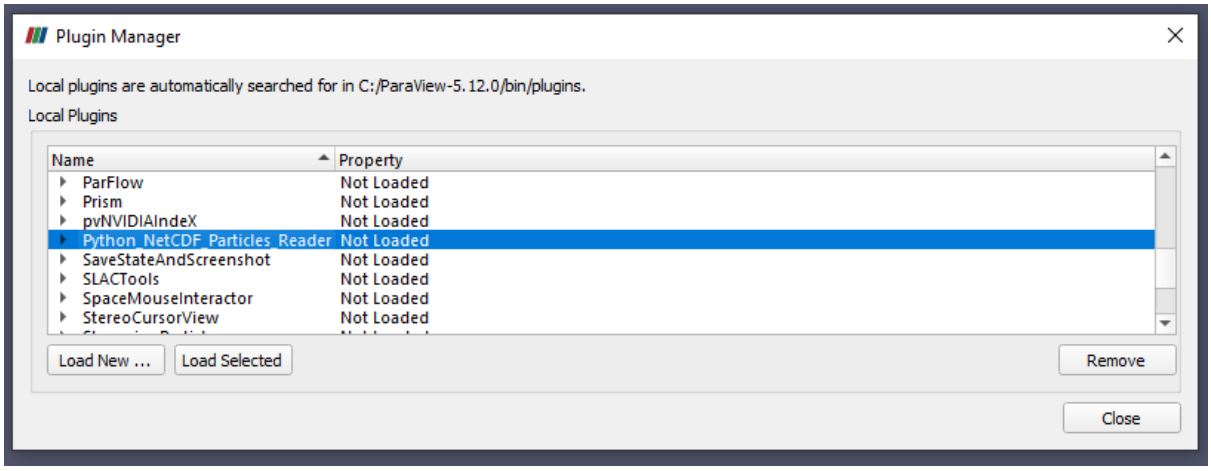
B 3. Start ParaView and load the plugin via the Plugin Manager: **Tools → Manage Plugins→ Load New → select the Python_NetCDF_Particles_Reader.py** file in the opening **Load Plugin** window



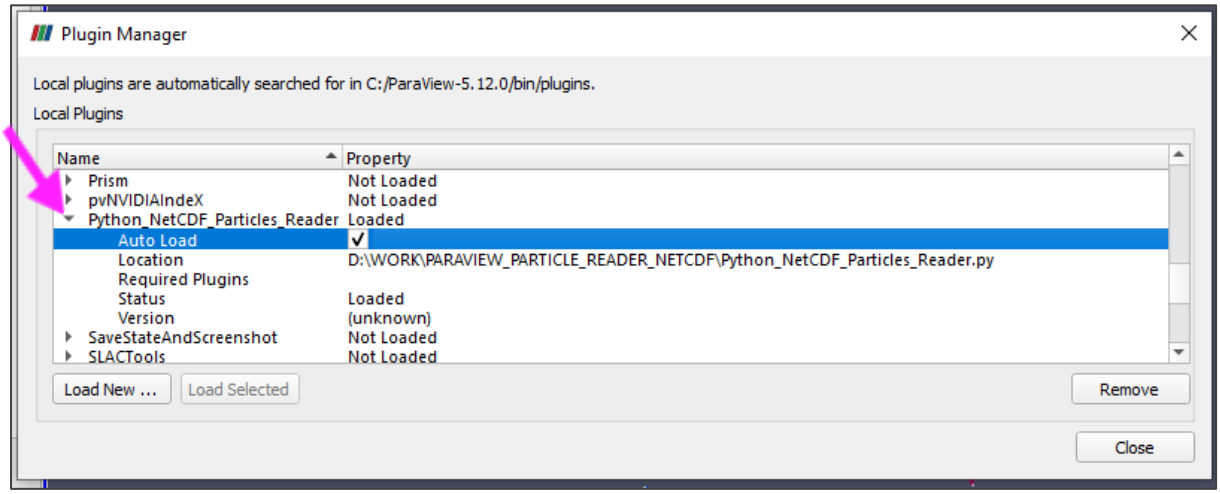
B 4. The plugin will now appear as **Loaded** in the **Plugin Manager**.



NOTE: Every time you will restart ParaView, the plugin will be still visible in the **Plugin Manager**, but as **Not Loaded**. Therefore, first you will need to select the plugin, then press the **Load Selected** button.

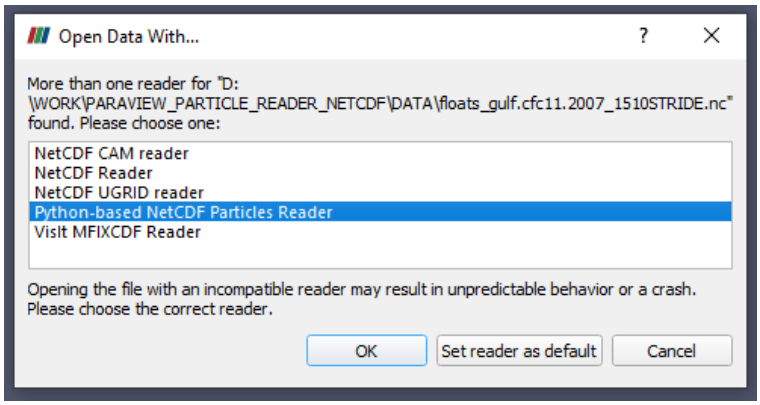


Alternatively, clicking on the left hand side arrow will reveal the option to **Auto Load** the plugin. Check the **Auto Load** box to have the plugin **load automatically when starting ParaView**.



C 1. How to use the Plugin – example of a data set without vertical dimension

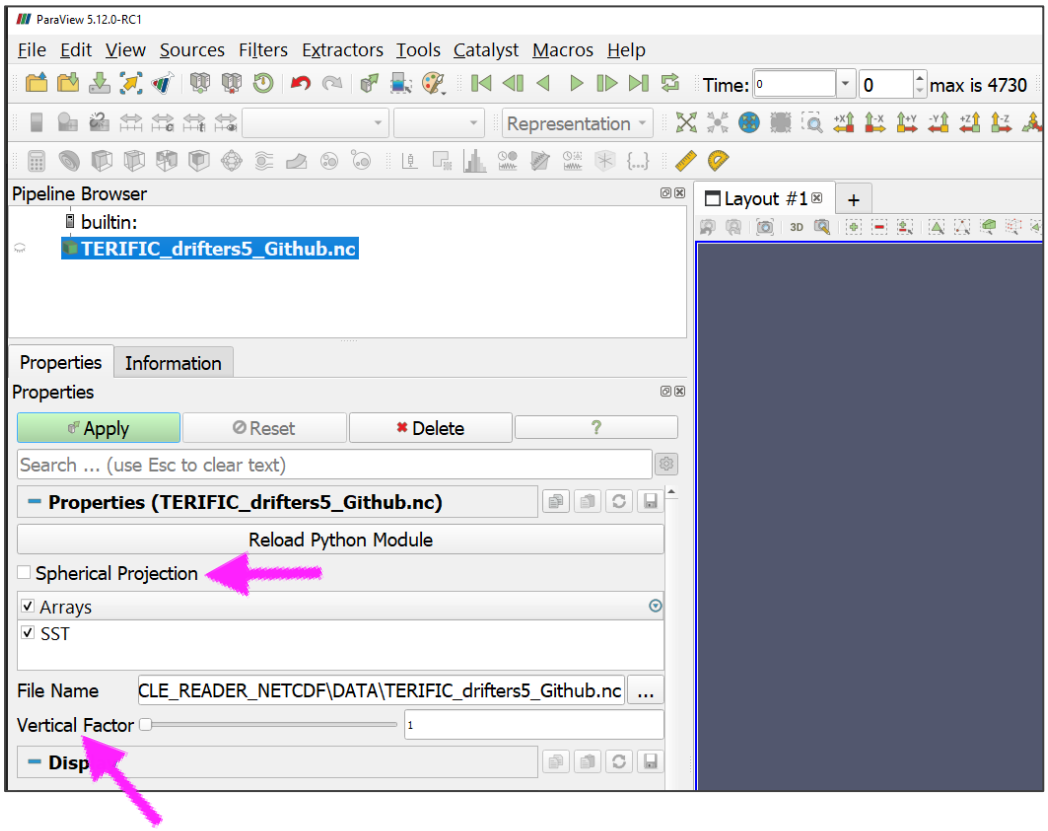
C 1.1. Open your NetCDF particles trajectories file (example TERIFIC_drifters5_Github.nc): **File -> Open -> Select your .nc file**. A window will open listing the available ParaView readers for NetCDF files. Select the **Python-based NetCDF Particles Reader**



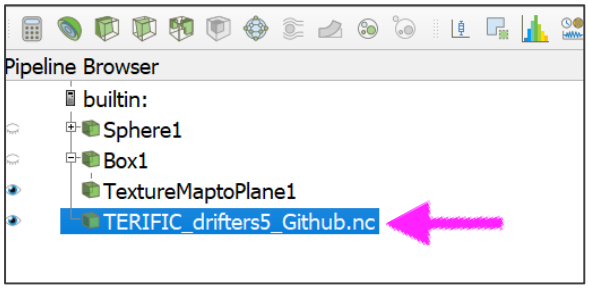
C 1.2. The NetCDF particles trajectories data set will be shown in the **Pipeline Browser**. The list of variables will appear in the **Properties** tab. Two options are also present in the **Properties** tab: **Spherical Projection** and **Vertical Factor**. The **Vertical Factor** is of significance only for data sets that have a vertical dimension. If you don't wish to do modifications to any of these two options, click **Apply**.

Otherwise, check the **Spherical Projection** box and/or modify the **Vertical Factor** by using the slider or typing in the box field the desired value. Click **Apply** after each modification, for it to register with ParaView. Both options can be modified at a later point, with some limitations (please view §D, p. 11)

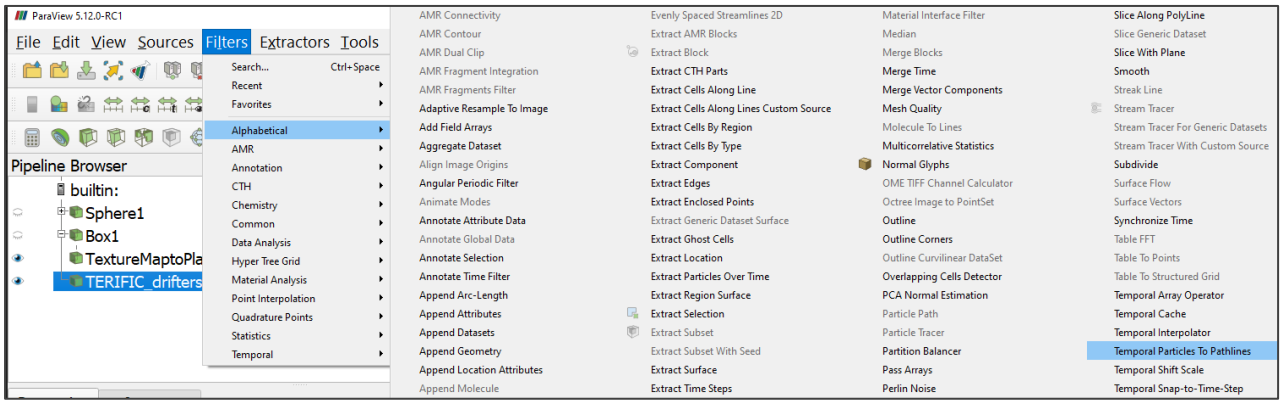
In our current example, the data set has no vertical dimension and we won't change to **Spherical Projection** at this point, so we will simply click **Apply**.



C 1.3. We will create now the points and paths corresponding to particles and their trajectories, respectively. Make sure the **data set is selected** in the **Pipeline Browser**: simply click on it with the left mouse button. Selected data will appear on a dark blue background.



From the **Filters** menu, choose **Alphabetical**, then the **Temporal Particles to Pathlines** filter. Or **Filters** → **Search** → type in **Temporal Particles …** and choose the respective filter.



In the **Properties Tab** three options are most relevant.

- The **Mask Points** option is useful when we have a big number of particles, and would like to track only some of them. For example, setting this field to 10, will result in only each 10-th particle being tracked.

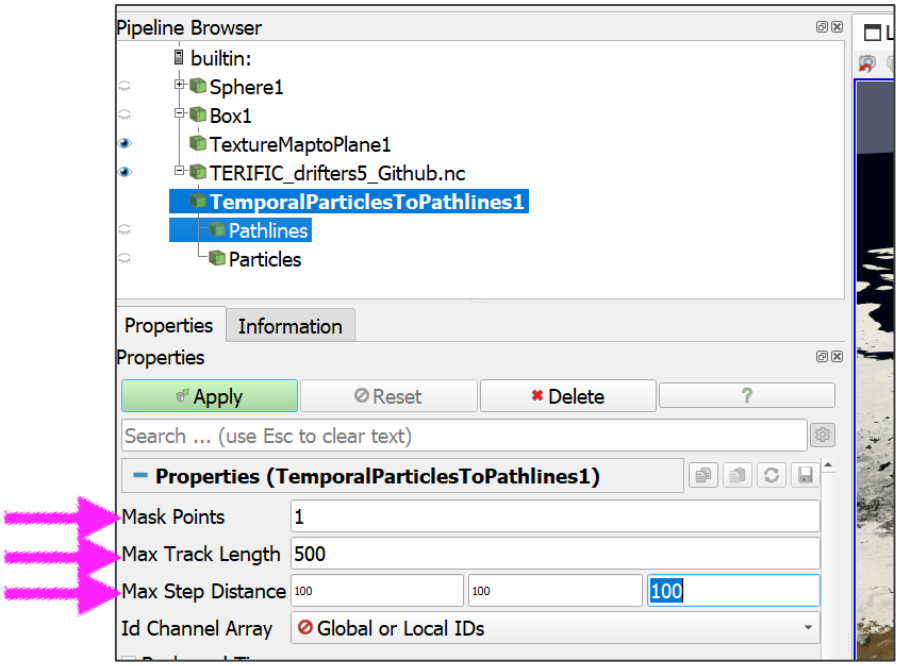
In the current example, we only have few particles, so that we type **1** in this field, meaning that **each particle will be tracked.**

- The **Max Track Length** field practically represents the particle lifetime and refers to the duration in time steps a particle will be displayed. This is especially useful when we have many time steps and displaying the particles across the entire time span would result in long, overlapping and messy path lines. Choosing a smaller value here will result in shorter path lines.

Our example has 4730 time steps, so we'll choose a smaller value, e.g. **500** - the paths will have a length corresponding to **500 time steps**.

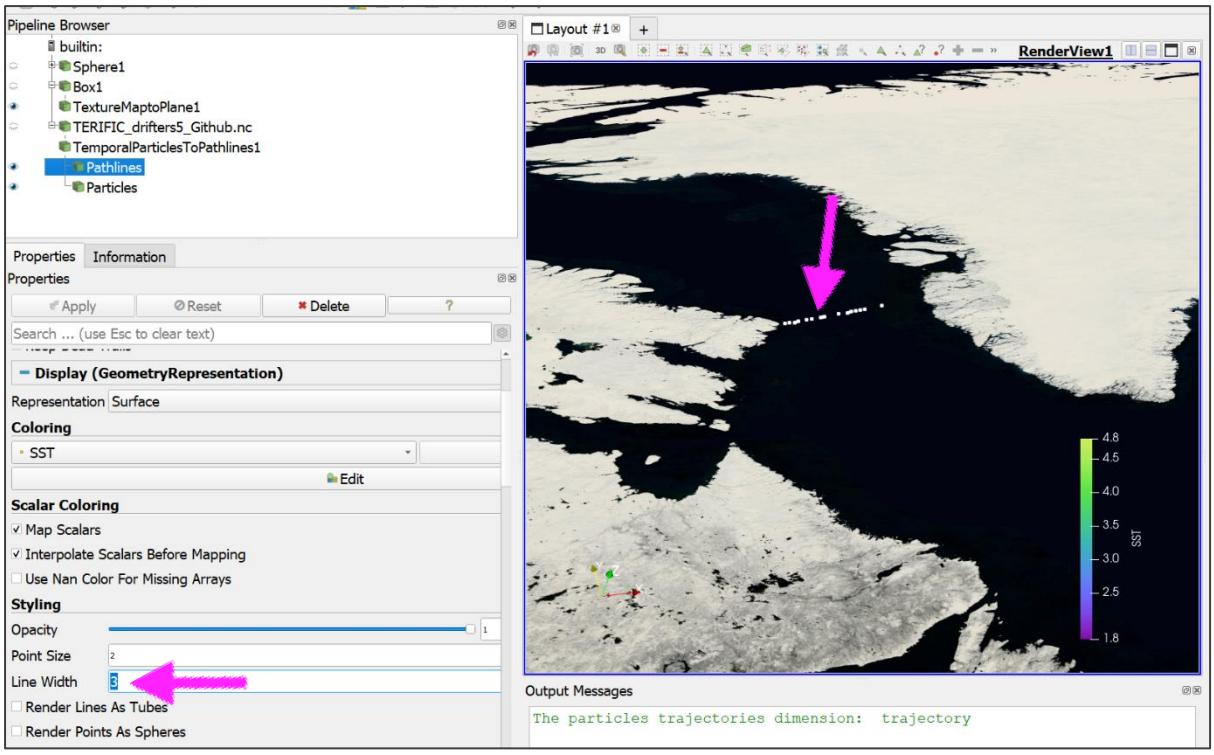
- The **Max Step Distance** field stands for the maximum distance a particle can travel during **one time step** along the x, y, z axes. If the particle exceeds this distance, it will be dropped from tracking. If you are not sure how big is this distance, simply put here a generous value to cover all possibilities.

In our example, we put **100** along each of the three axes. Click **Apply** after you did the necessary modifications.

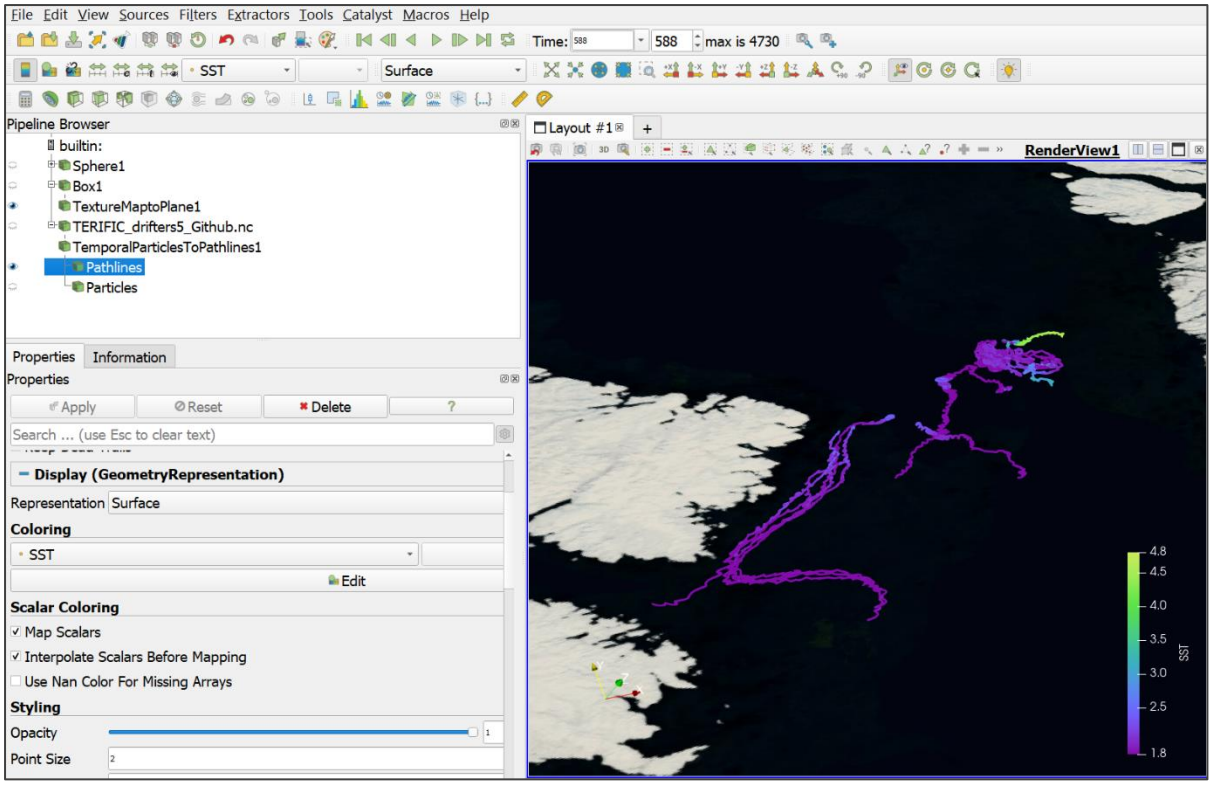


After clicking **Apply** the particles will appear in the 3D viewport, as points of **Solid Color**, i.e. a uniform color. It is not possible to color the particles by any variable.

However, the **Particle Paths** can be colored by variables. In our example we choose **SST** (sea surface temperature). We also increase the **Line Width** of the paths to 3, to have thicker paths with better visibility.

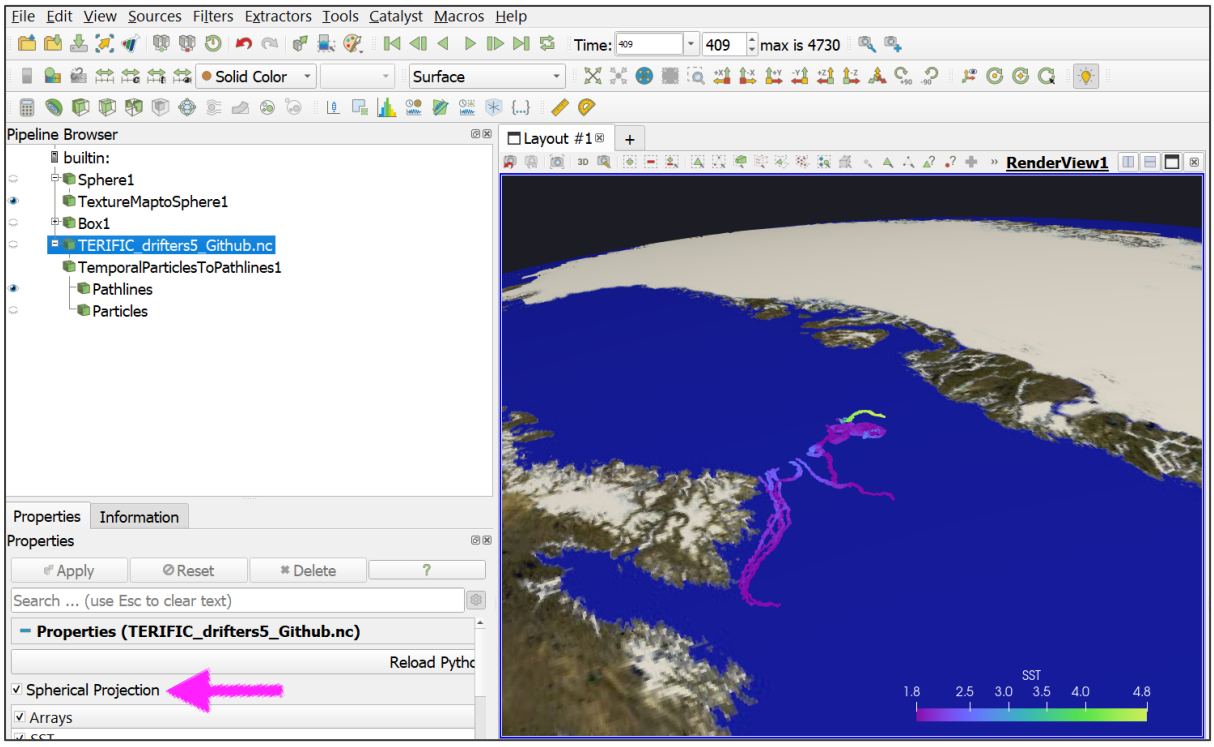


If we play forward the animation controls, the particles path lines will be displayed.



Let us try the **Spherical Projection**. Select the original data set and check **the Spherical Projection** box. Then play again the time steps ahead.

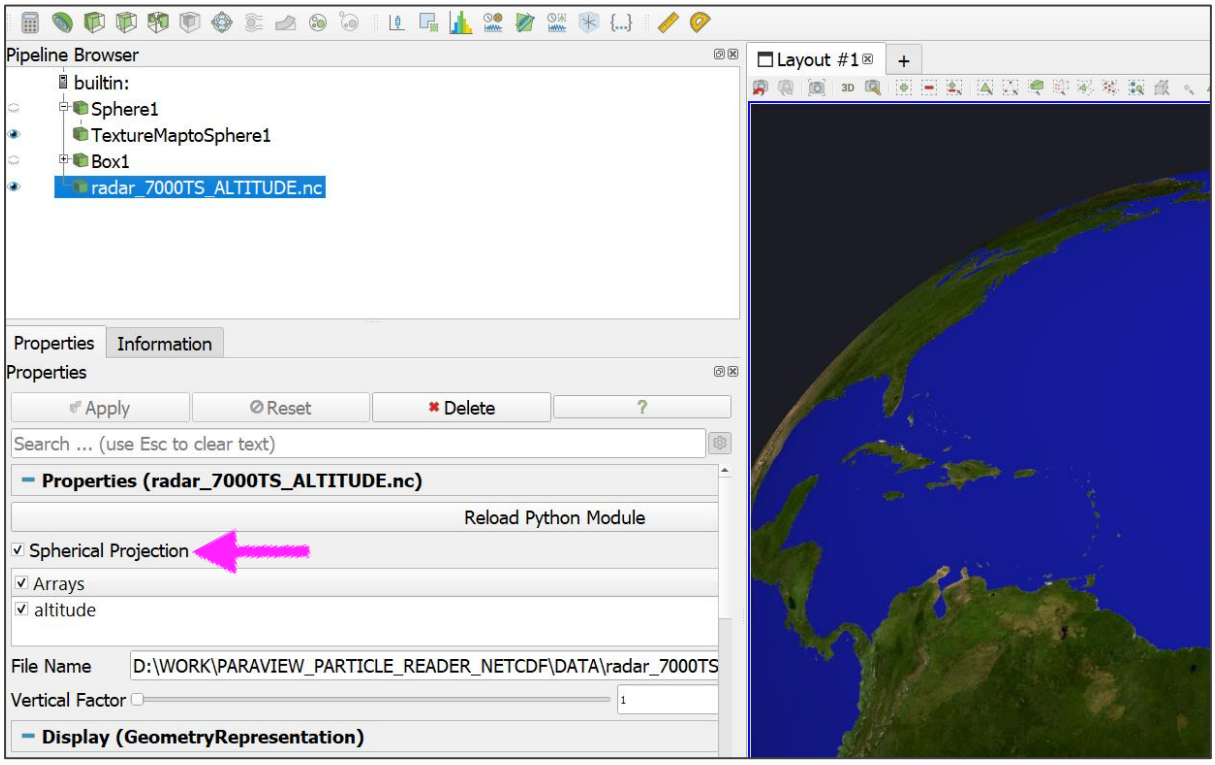
NOTE: While it should be possible to change the projection to Spherical and back at any time step of the animation, ParaView often crashed during this change. For your project to be safe**, it is strongly recommended to go back to the first time step and then change the projection**. Please view more details in §D, p. 11.



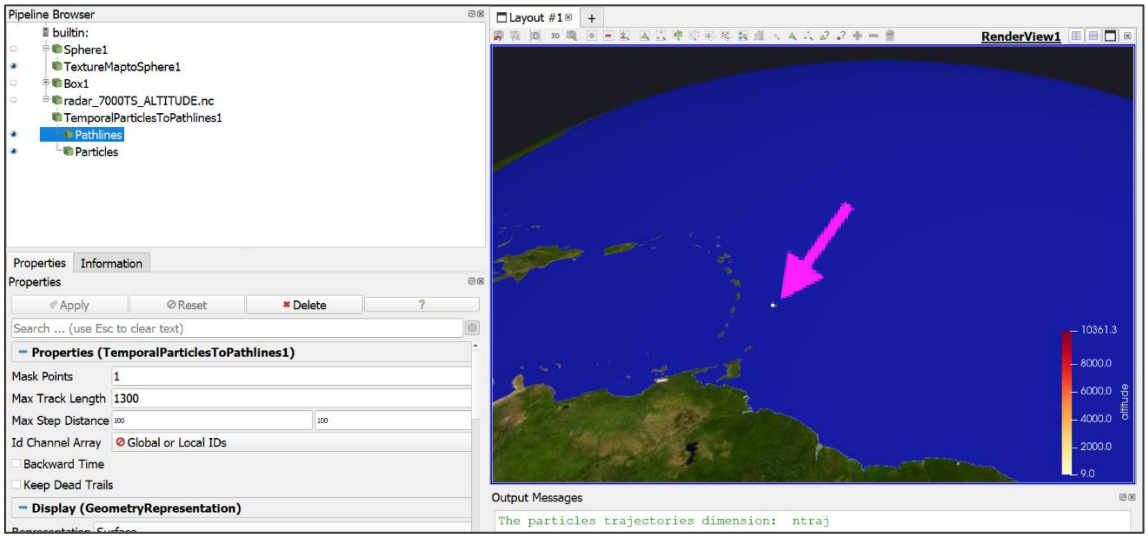
C2. How to use the Plugin – examples of data sets that have a vertical dimension

C 2.1. We open the NetCDF particles trajectories file, as described at C 1.1 (example file radar_7000TS_ALTITUDE.nc): **File -> Open -> Select your .nc file**. A window will open listing the available ParaView readers for NetCDF files, select the **Python-based NetCDF Particles Reader**

C 2.2. The NetCDF particles trajectories data set will be shown in the **Pipeline Browser.** This example shows the flight path of an ascending plane and has a vertical dimension. We would like to display the data on the Earth sphere, so we check the **Spherical Projection** box. As for the **Vertical Factor**, we will modify it at a later stage - first we'll let a few time steps run to see how the flight path is displayed.

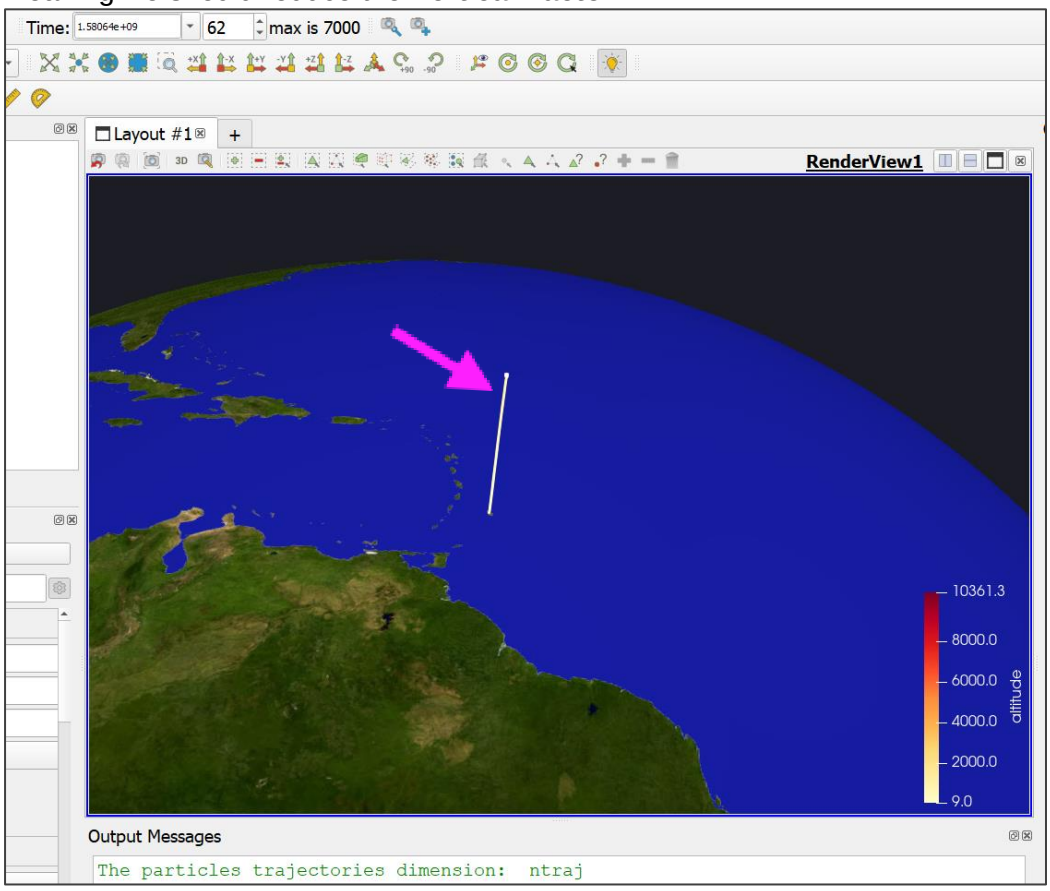


C 2.3. We first select the data set in the Pipeline Browser, then apply the **Temporal Particles to Pathlines** filter, as described in C 1.3.

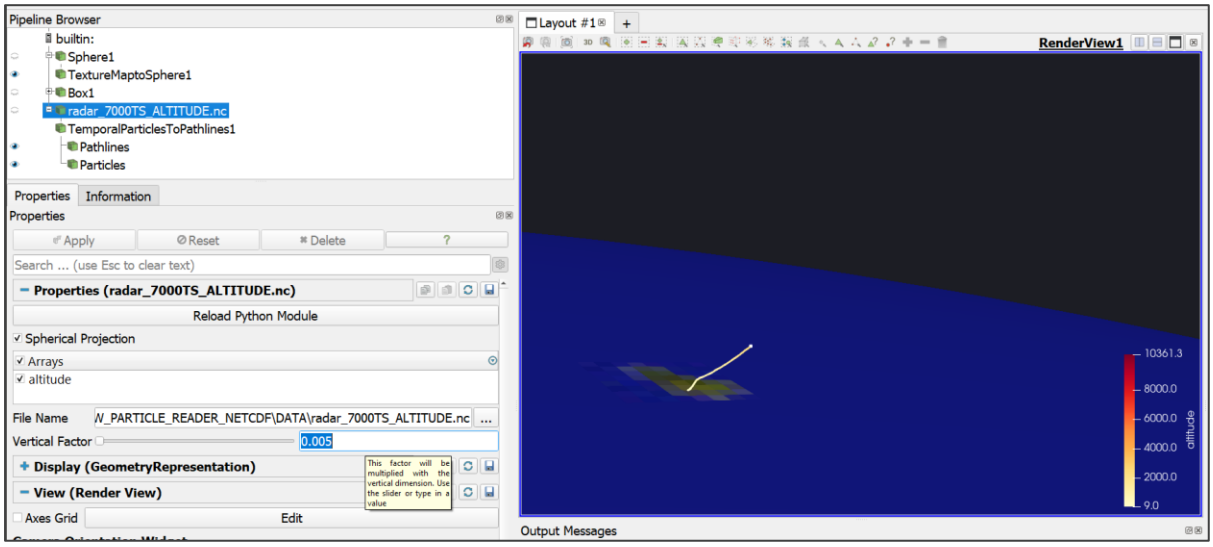


- We will choose again 1 for **Mask Points** - we have a single point, which of course we will track, so we won't mask (i.e. exclude) any points.
- **Max Track Length** will be for example1300 - this is a dataset with 7000 time steps, we'd like to avoid displaying a too long path.
- The **Max Step Distance** is 100 on all three axes.

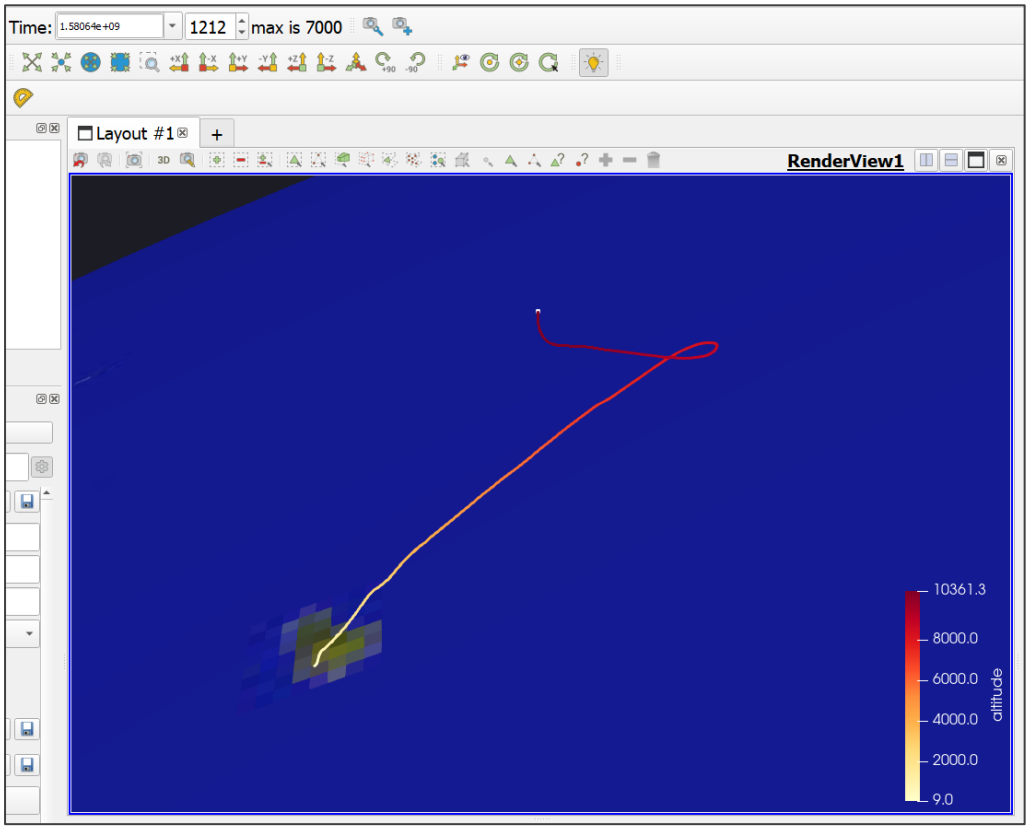
We play forward the time steps and have a look at the path – which in this case is too steep, meaning we should reduce the **Vertical Factor**.



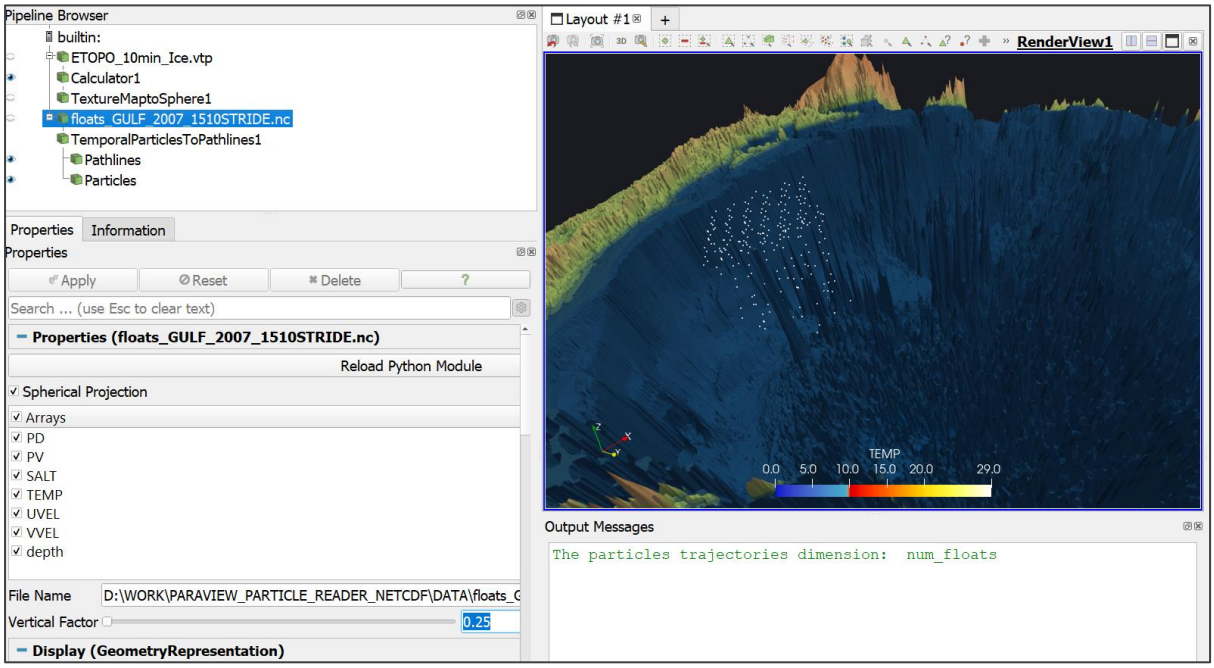
The **Vertical Factor** should be modified according to the specifics of each visualization. In our current example, here is the result after reducing the **Vertical Factor** to 0.005:

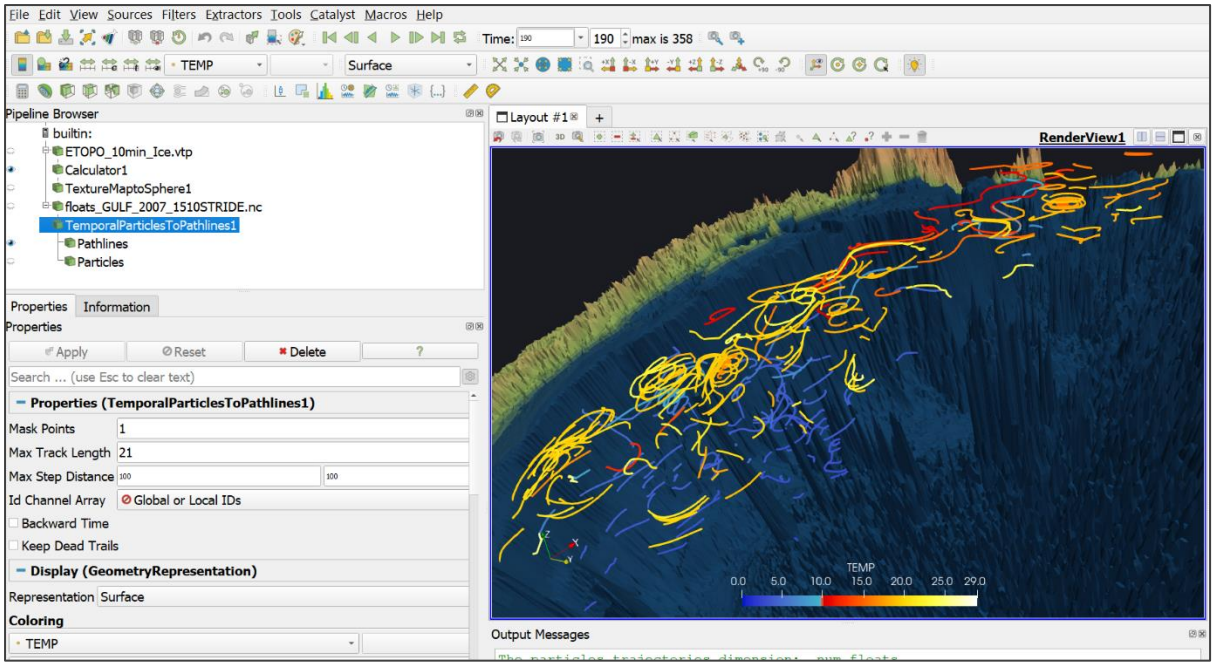


After playing more time steps, we are happy with the result so we left the **Vertical Factor** at value 0.005. It should be possible to modify the **Vertical Factor** and **Spherical Projection** at any time step, however in our tests ParaView sometimes crashes - please view §D, p. 11.



C 2.4 We provided a second data set example with a vertical dimension for self-study: floats_GULF_2007_1510STRIDE.nc, with daily ocean floats showing the Gulf Stream flow. The images below show an example configuration with **Spherical Projection** and **Vertical Factor** of 0.25. The Temporal Particles to Pathlines parameters are: **Mask Points**= 1, **Max Track Length**= 21, **Max Step Distance** is 100 on all three axes.





D. Important notes: limitations of the Plugin

Please note that the plugin was developed in a **Python 3.10** environment and tested **only with ParaView 5.12 for Windows –** ParaView 5.12 has the same Python version.

Tests have shown a couple of issues:

1. Modifying the projection to **Spherical Projection** and back, as well as the value of the **Vertical Factor** seems to function best when done either at the first time step or after playing a relatively small number of time steps.

The bigger the data set, the less time steps should be used to estimate how to apply these modifications.

When applying these modifications at the end of the time line, or a bigger number of time steps, ParaView crashes (possibly a memory issue).

2. ParaView crashes also appear to happen when:

- **jumping a big number of steps** forward or backward (when typing a new value in the **Time** field or jumping at the last/first frame from the animation controls)

- **changing the color map** for the **Pathlines**, specifically the **ParaView preset color maps**.

This is why it is strongly recommended to save your project as soon as you applied the **Temporal Particles to Pathlines** filter.

Then, in case you experience any of the issues above, simply reopen your project and try to make the **Spherical Projection**, **Vertical Factor** and **color map** modifications at the first time step, or after a small number of time steps.

Also, where it applies, avoid time line jumps, and play the animation only with **consecutive time steps** from the animation controls.