LaMEM (Canonical) - Model Setup and Markers initialization

~ contributed by Adina Pusok ~

There are two ways to generate initial model setups for LaMEM: 1) **internal** and 2) **external** initialization. To use a particular model setup or initialization, the command line is the following:

```
./LaMEM ... -msetup <option>
```

or you can include it in the parameters file (ParamFile) as:

```
# Model Setup
msetup <option>
```

The <options> are:

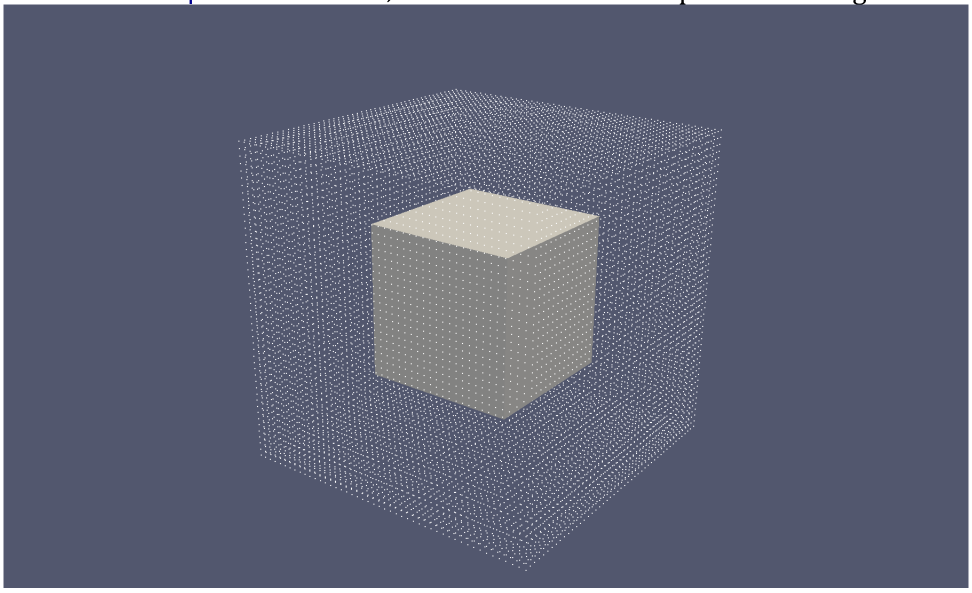
```
parallel     # external parallel initialization
redundant    # external redundant initialization
diapir       # internal diapir setup
block        # internal falling block setup
subduction   # internal pseudo-2D subduction
folding      # internal folding setup
detachment   # internal folding over detachment setup
slab         # internal slab detachment setup
spheres      # internal multiple spheres
```

Internal Initialization:

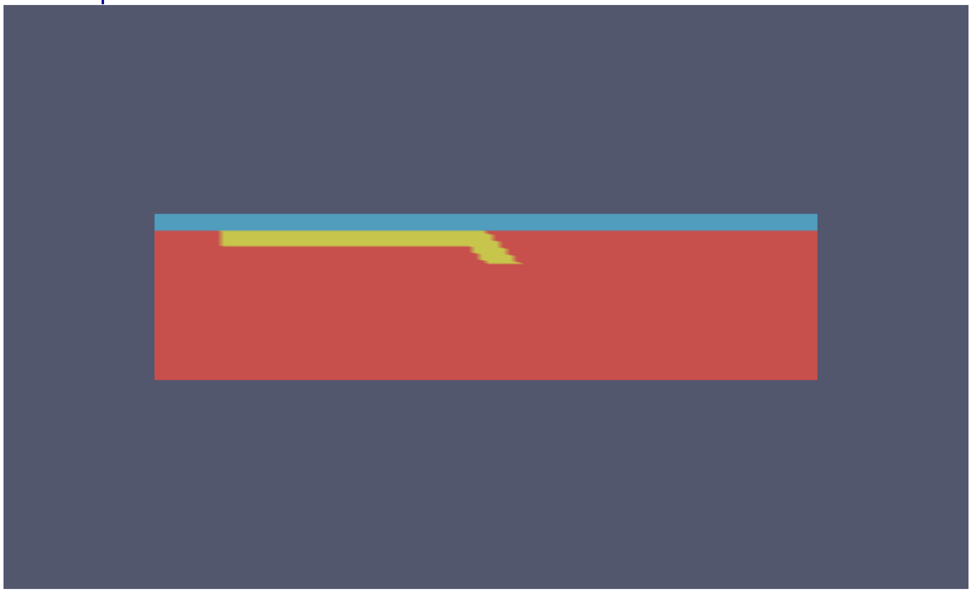
These setups have already prescribed geometries (and in some cases material properties as well) and can be easily called with the command line. Below you can find a brief description of each of the internal model setups, along with the command options for LaMEM. They are all defined in /fdstag/marker.c .

```
# 1. Falling Block Setup
-msetup block
```

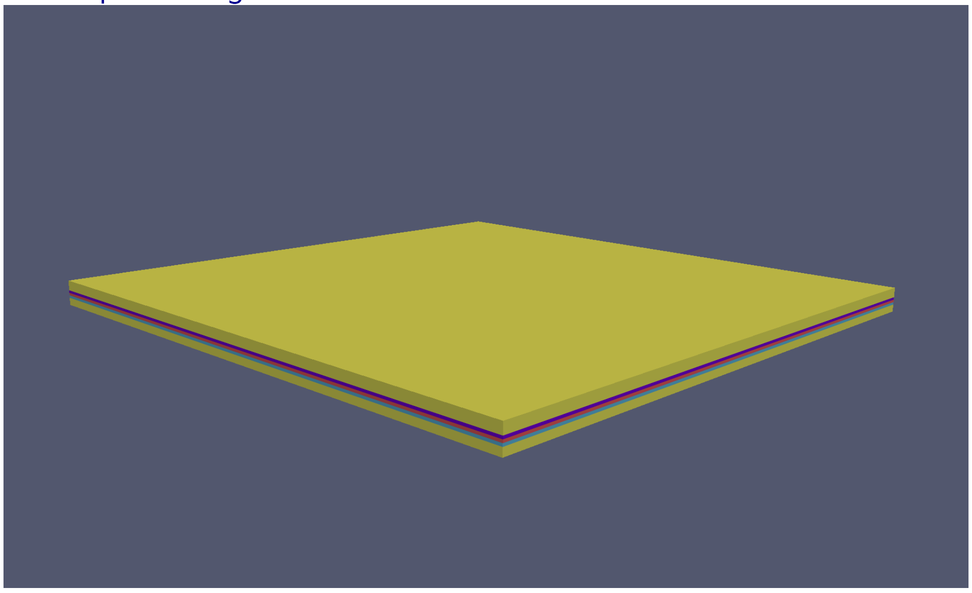
Note: if `msetup` is not defined, the default model setup is the 'Falling Block'



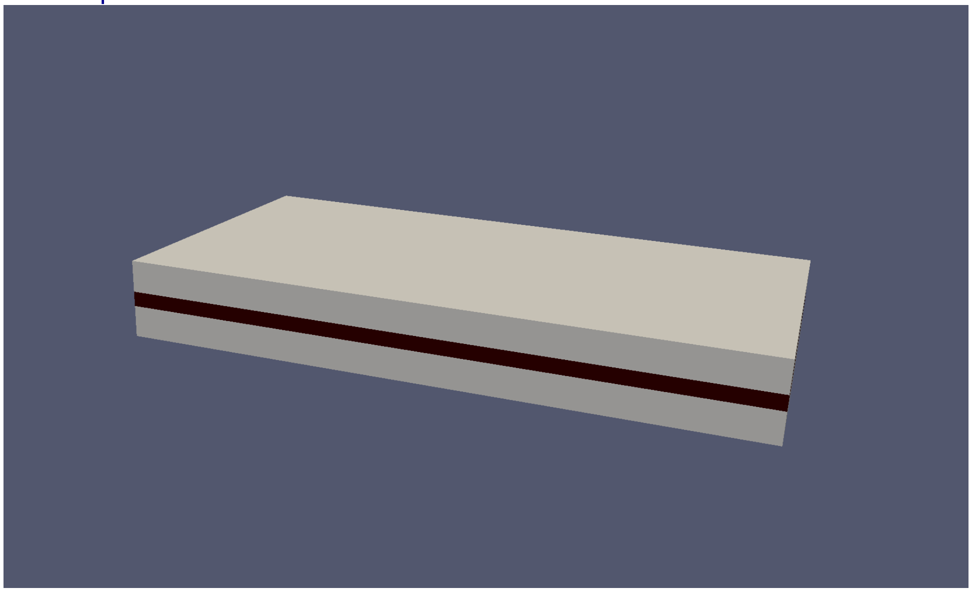
```
# 2. Subduction with Sticky Air
-msetup subduction
```



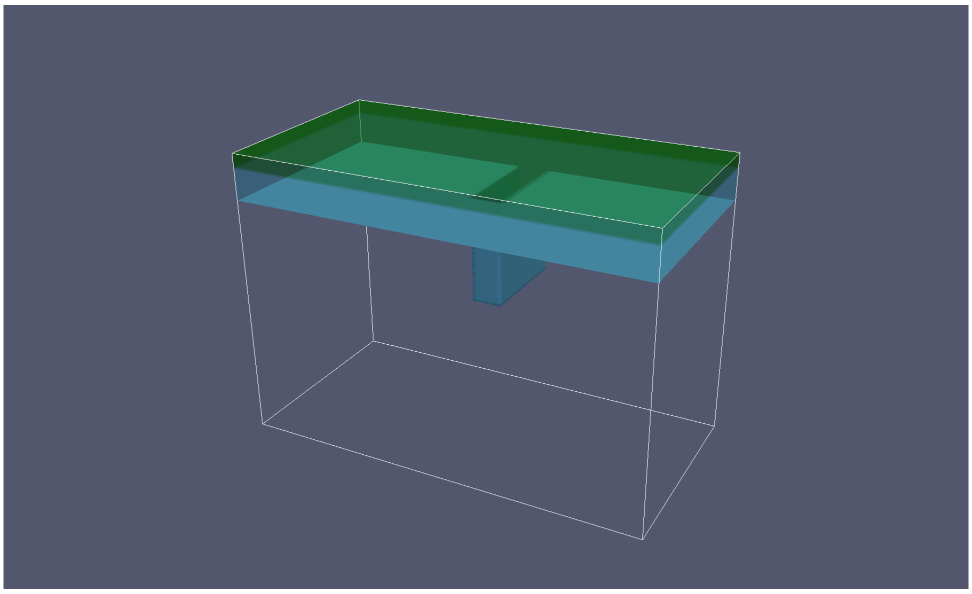
```
# 3. Multilayer Folding Setup
-msetup folding
```



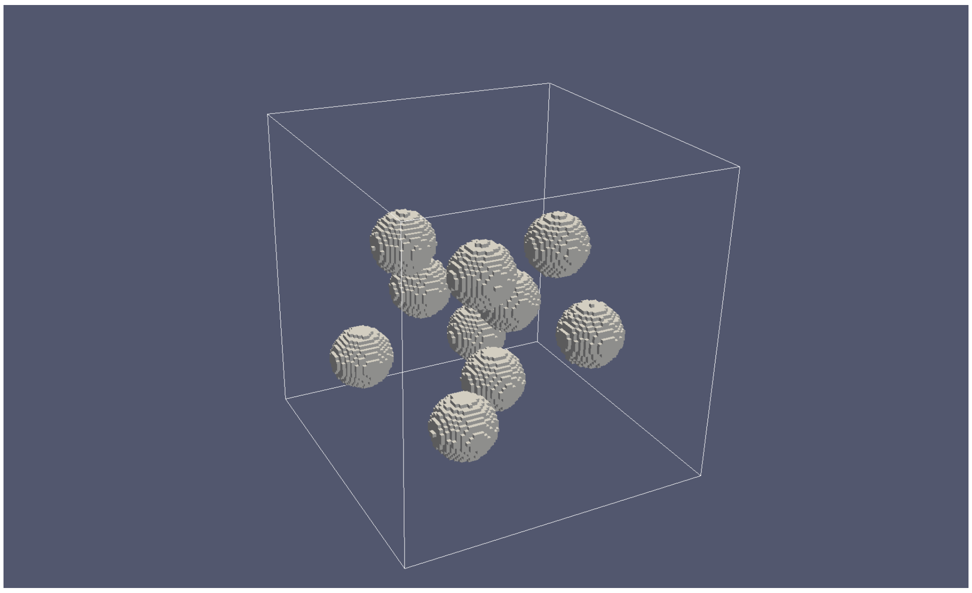
```
# 4. 1 layer over detachment Setup
-msetup detachment
```



```
# 5. Slab Detachment Setup
-msetup slab
```



```
# 6. Multiple Falling Spheres Setup
-msetup spheres
```



External Initialization (with input markers files):

The preferred method to initialize LaMEM model setups is to create initial marker distribution in Matlab. This allows the user to create variable geometry model setups.

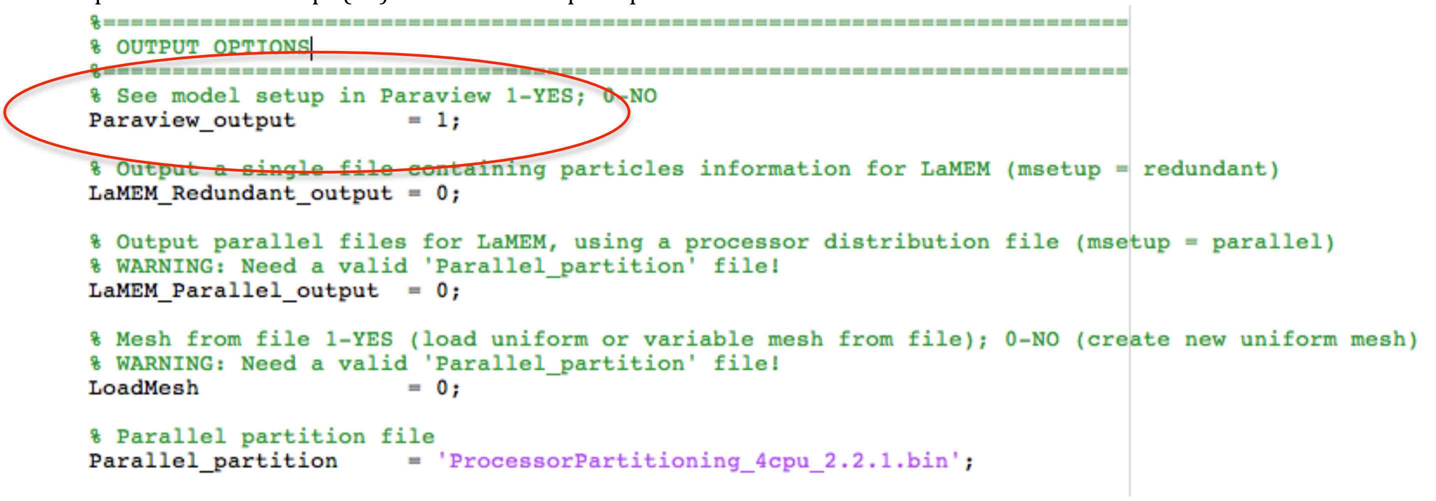
HOW TO:

1) Create and run a 3D subduction setup with a uniform mesh - Redundant

Note: This method is easy to work with, but it is not computationally efficient!! Use this method ONLY for small setups; for large setups, use the parallel routines, otherwise you will NOT be able to run them with LaMEM!

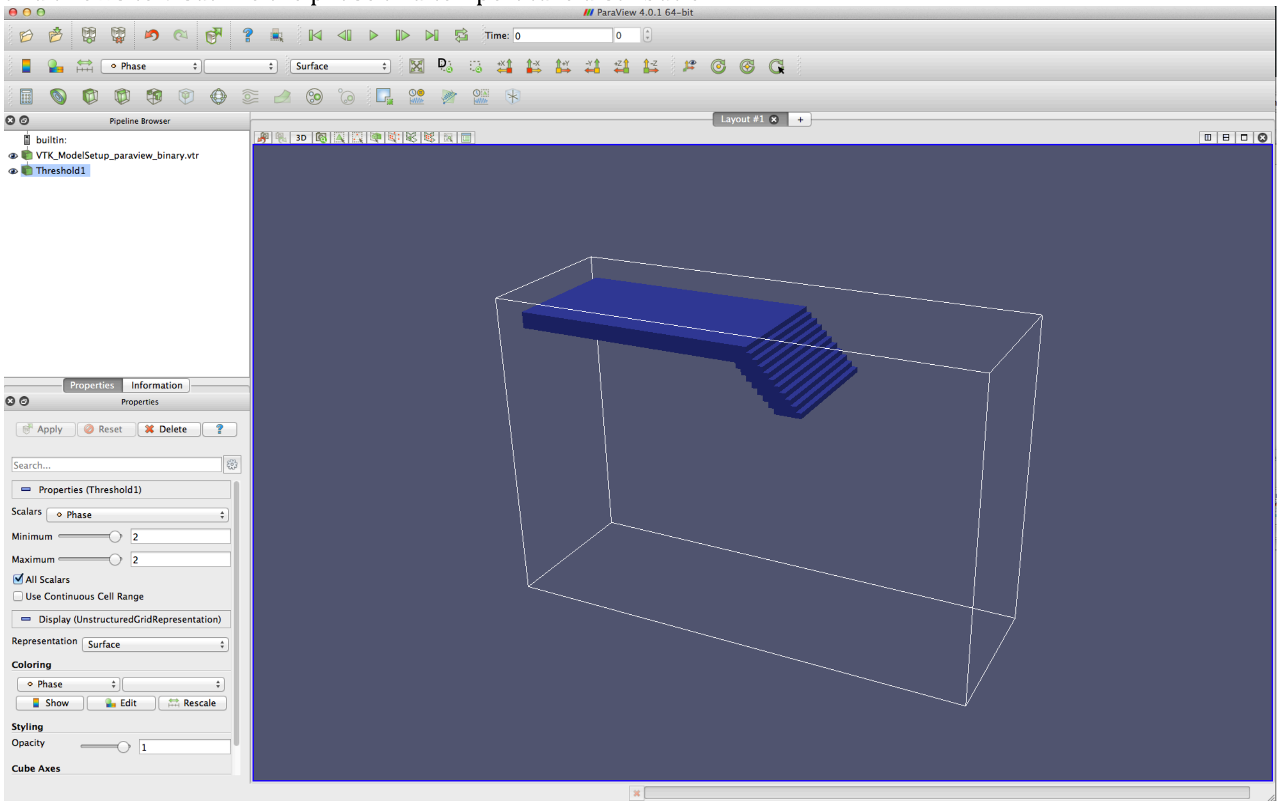
1. Go to /LaMEM/input_models/PROJECTS_FDSTAG_CANONICAL/.
   There are two files of interest for now: CreateMarkers_Subduction_parallel.m and Subduction_MATLAB_Particles_canonical_redundant.dat

2. Open the Matlab script (.m) and set the Output Options like this:



```
%=========================================================================
% OUTPUT OPTIONS
%=========================================================================
% See model setup in Paraview 1-YES; 0-NO
Paraview_output          = 1;

% Output a single file containing particles information for LaMEM (msetup = redundant)
LaMEM_Redundant_output = 0;

% Output parallel files for LaMEM, using a processor distribution file (msetup = parallel)
% WARNING: Need a valid 'Parallel_partition' file!
LaMEM_Parallel_output  = 0;

% Mesh from file 1-YES (load uniform or variable mesh from file); 0-NO (create new uniform mesh)
% WARNING: Need a valid 'Parallel_partition' file!
LoadMesh                 = 0;

% Parallel partition file
Parallel_partition       = 'ProcessorPartitioning_4cpu_2.2.1.bin';
```

3. Run the script. A new file was created VTK_ModelSetup_paraview_binary.vtr which can be opened in Paraview and allows to visualize the phase and temperature distribution.



Pipeline Browser
builtin:
VTK_ModelSetup_paraview_binary.vtr
Threshold1

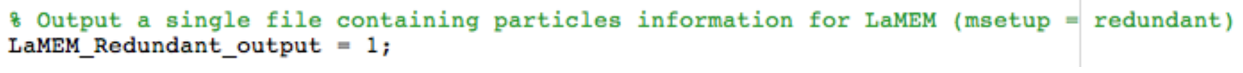
Properties (Threshold1)
Scalars: Phase
Minimum: 2
Maximum: 2
[✓] All Scalars
[ ] Use Continuous Cell Range
Display (UnstructuredGridRepresentation)
Representation: Surface
Coloring: Phase
Styling
Opacity: 1
Cube Axes

4. Go back to the Matlab script and try to understand how the geometry was created. Look at sections 'Domain Parameters', 'Mesh Grid' – uniform mesh, 'Phases', 'Setup Geometry'. You will notice that you need to specify:
   a. domain parameters (W,L,H) in dimensional units
   b. number of markers in x, y, z direction
   c. number of markers/cell
   d. other model specific parameters (dimensional)

   The mesh grid is created in such way that each cell has a constant size and will have a constant number of markers, distributed uniformly.

   You can play with the parameters to construct your desired model setup. Ideas: 1) create upper plate with a new phase 2) widen your domain in the y-direction such as L = 300 km and modify the equivalent no. of markers. Use Paraview to visualize your changes.

5. Once you are satisfied with a certain geometry, phase and temperature distribution, go to the 'Output Options' and rerun the script with the option:

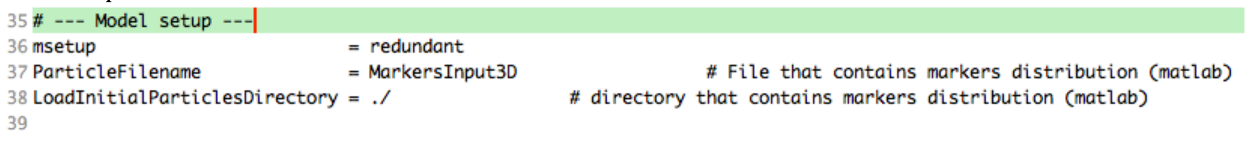


```
% Output a single file containing particles information for LaMEM (msetup = redundant)
LaMEM_Redundant_output = 1;
```

   This created a file called MarkersInput3D.dat

6. Before running LaMEM it is important to check the input parameters file to be consistent with your model setup that you created in Matlab!!! Open Subduction_MATLAB_Particles_canonical_redundant.dat and check the parameters described in step 4.

7. Set these parameters as:



```
# --- Model setup ---
msetup                      = redundant
ParticleFilename            = MarkersInput3D            # File that contains markers distribution (matlab)
LoadInitialParticlesDirectory = ./               # directory that contains markers distribution (matlab)
```

8. Run LaMEM on N processors:

```
mpiexec -n N LaMEM –ParamFile Subduction_MATLAB_Particles_canonical_redundant.dat
```

2) Create and run a 3D subduction setup with a uniform mesh – Parallel

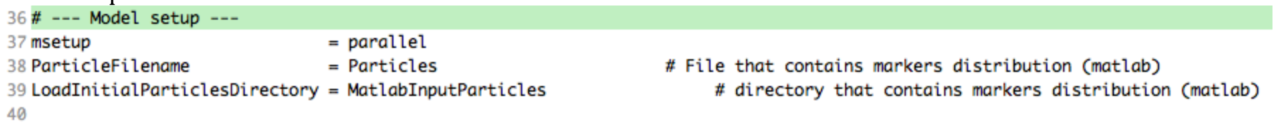
Note: If this is the first time using LaMEM, first do the steps in **HOW TO: 1) Create and run a 3D subduction setup with a uniform mesh - Redundant**

This method partitions the markers information (i.e. contained in MarkersInput3D.dat) in N files, where N is the number of processors on which LaMEM is going to run. For example, I want to run LaMEM on 8 processors, so the markers information will be split into 8 files, and each processor will load only the corresponding part of the domain/file. This method allows users to run high resolution setups on a large number of cores. So try to use this method by default!

1. Create your geometry in Matlab (follow steps 1-4 in **HOW TO: 1) Create and run a 3D subduction setup with a uniform mesh - Redundant**). Only after you have a final geometry proceed to the next step!

2. Open Subduction_MATLAB_Particles_canonical_parallel.dat and check that the input parameters are consistent with your model setup that you just created in Matlab. Double check!

3. Set these parameters as:



```
# --- Model setup ---
msetup                      = parallel
ParticleFilename            = Particles                 # File that contains markers distribution (matlab)
LoadInitialParticlesDirectory = MatlabInputParticles          # directory that contains markers distribution (matlab)
```

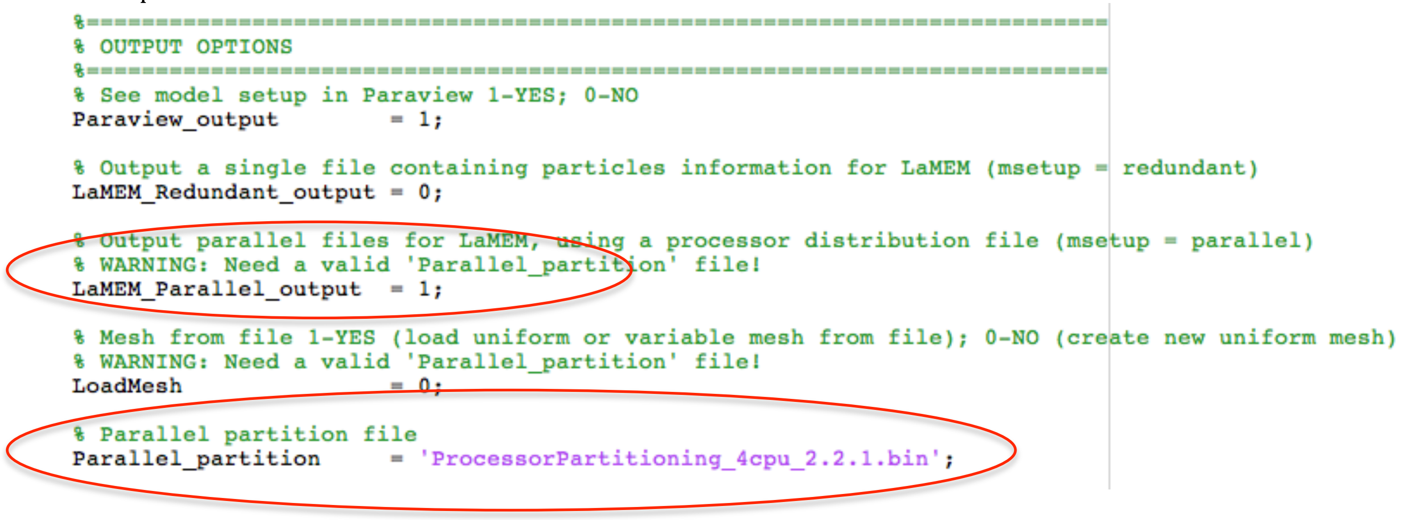
4. Run LaMEM with the flag `-SavePartitioning` to create the processor partitioning file. Try on 4 cpus:

```
   mpiexec -n 4 LaMEM -ParamFile Subduction_MATLAB_Particles_canonical_parallel.dat -
SavePartitioning
```

   It will give an error, but don't worry about it! The important thing is that it created the processor partitioning file **ProcessorPartitioning_4cpu_2.2.1.bin .**

Note: This step cannot be avoided because the parallel partitioning is decided by PETSc for a given domain configuration, so it's not unique! Be careful to always generate the right partitioning file for the setup you want!

5. Go back to the Matlab script and using the new partitioning file, change the following Output Options and rerun the script:



```
%==========================================================================
% OUTPUT OPTIONS
%==========================================================================
% See model setup in Paraview 1-YES; 0-NO
Paraview_output         = 1;

% Output a single file containing particles information for LaMEM (msetup = redundant)
LaMEM_Redundant_output = 0;

% Output parallel files for LaMEM, using a processor distribution file (msetup = parallel)
% WARNING: Need a valid 'Parallel_partition' file!
LaMEM_Parallel_output   = 1;

% Mesh from file 1-YES (load uniform or variable mesh from file); 0-NO (create new uniform mesh)
% WARNING: Need a valid 'Parallel_partition' file!
LoadMesh                = 0;

% Parallel partition file
Parallel_partition      = 'ProcessorPartitioning_4cpu_2.2.1.bin';
```

6. The markers information is now located in the newly formed **./MatlabInputParticles** directory. There should be 4 files, one for each processor. You can now rerun LaMEM with the parallel files:

```
mpiexec -n 4 LaMEM -ParamFile Subduction_MATLAB_Particles_canonical_parallel.dat
```

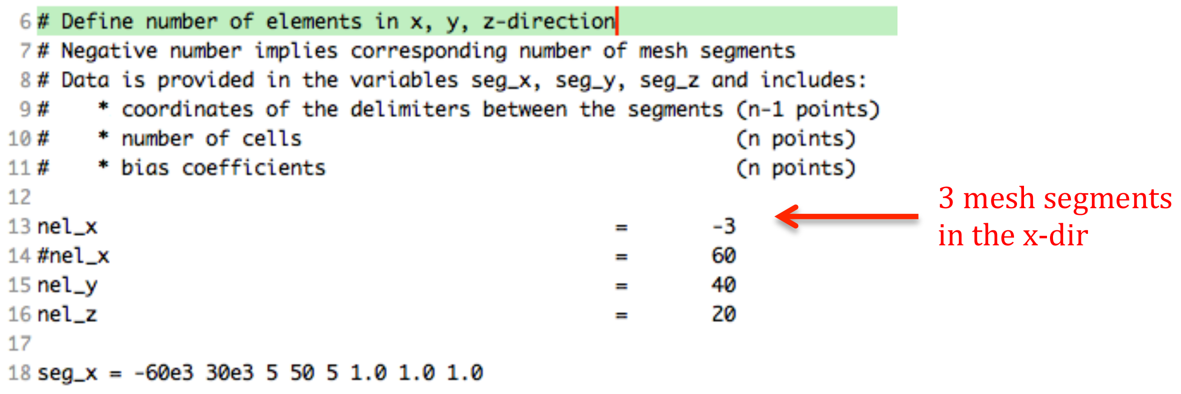
Note: If you change partitioning files (i.e. want to partition from 1 cpu to 4 cpu), delete the **./MatlabInputParticles** directory with the old partitioning to avoid subsequent errors. Always make sure that LaMEM parameters are consistent with the ones in Matlab!

3) Create and run a 3D subduction setup with variable mesh (Redundant or Parallel)

Note: This method requires the parallel partitioning file described in the previous section.

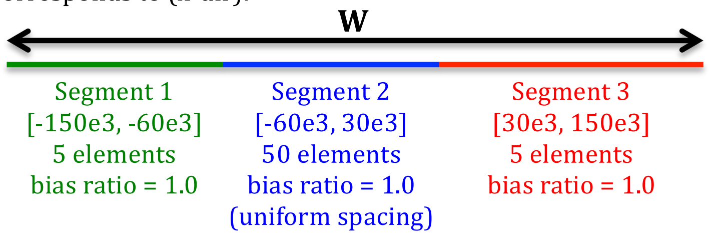
1. Create your model geometry in Matlab on a uniform mesh (the geometry will be adapted to the variable grid after the partitioning file is generated). See **HOW TO: 2) Create and run a 3D subduction setup with a uniform mesh – Parallel.**

2. Prepare the ParamFile Subduction_MATLAB_Particles_canonical_parallel.dat. Check the input parameters and tell LaMEM to create a variable mesh in the following way (for W=300e3, x_left = -150e3):



```
# Define number of elements in x, y, z-direction
# Negative number implies corresponding number of mesh segments
# Data is provided in the variables seg_x, seg_y, seg_z and includes:
#    * coordinates of the delimiters between the segments (n-1 points)
#    * number of cells                                    (n points)
#    * bias coefficients                                  (n points)

nel_x                                   =       -3
#nel_x                                  =       60
nel_y                                   =       40
nel_z                                   =       20

seg_x = -60e3 30e3 5 50 5 1.0 1.0 1.0
```

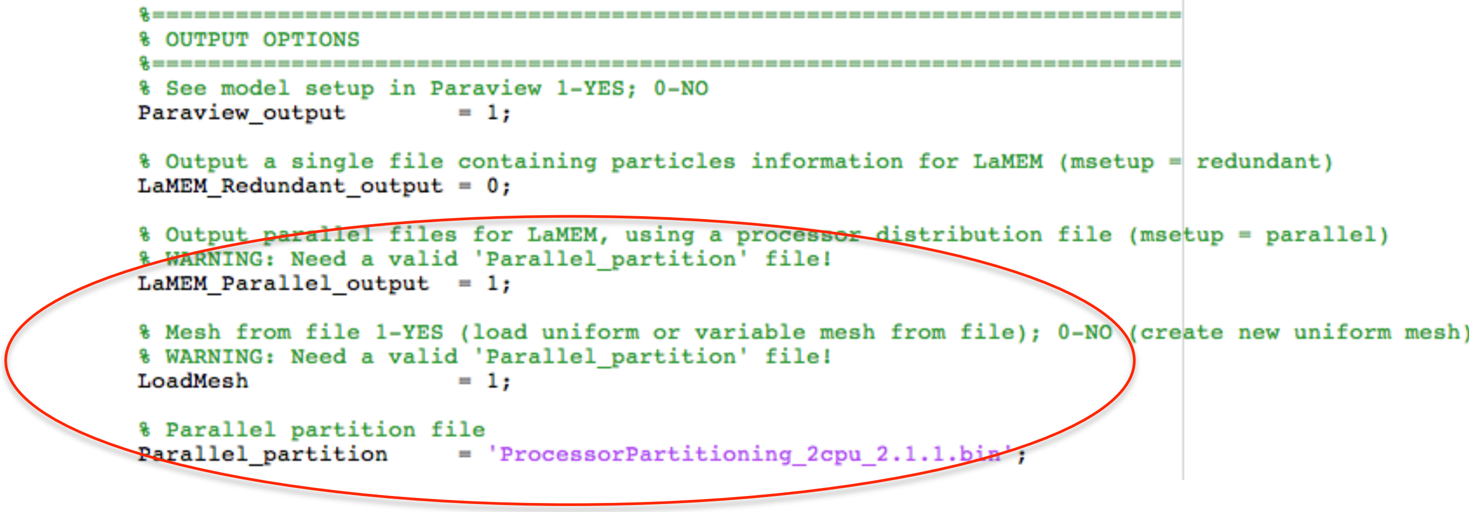
Which corresponds to (x-dir):



3. Run LaMEM to create the processor partitioning file (**ProcessorPartitioning_2cpu_2.1.1.bin)**:

```
   mpiexec -n 2 LaMEM -ParamFile Subduction_MATLAB_Particles_canonical_parallel.dat -
SavePartitioning
```

4. In the Matlab script change the Output Options and rerun the script. Check with Paraview the variable grid.

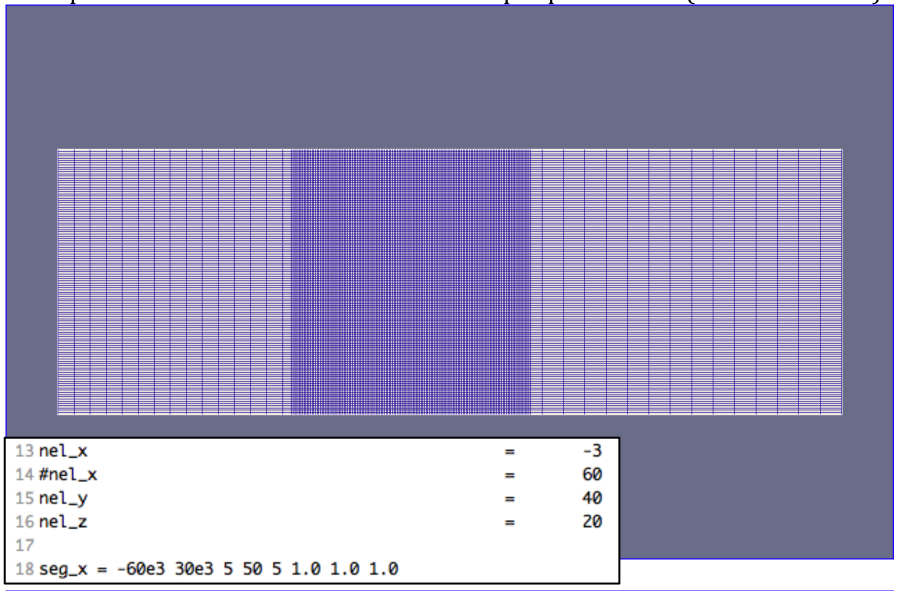


```
%==========================================================================
% OUTPUT OPTIONS
%==========================================================================
% See model setup in Paraview 1-YES; 0-NO
Paraview_output        = 1;

% Output a single file containing particles information for LaMEM (msetup = redundant)
LaMEM_Redundant_output = 0;

% Output parallel files for LaMEM, using a processor distribution file (msetup = parallel)
% WARNING: Need a valid 'Parallel_partition' file!
LaMEM_Parallel_output  = 1;

% Mesh from file 1-YES (load uniform or variable mesh from file); 0-NO (create new uniform mesh)
% WARNING: Need a valid 'Parallel_partition' file!
LoadMesh               = 1;

% Parallel partition file
Parallel_partition     = 'ProcessorPartitioning_2cpu_2.1.1.bin';
```

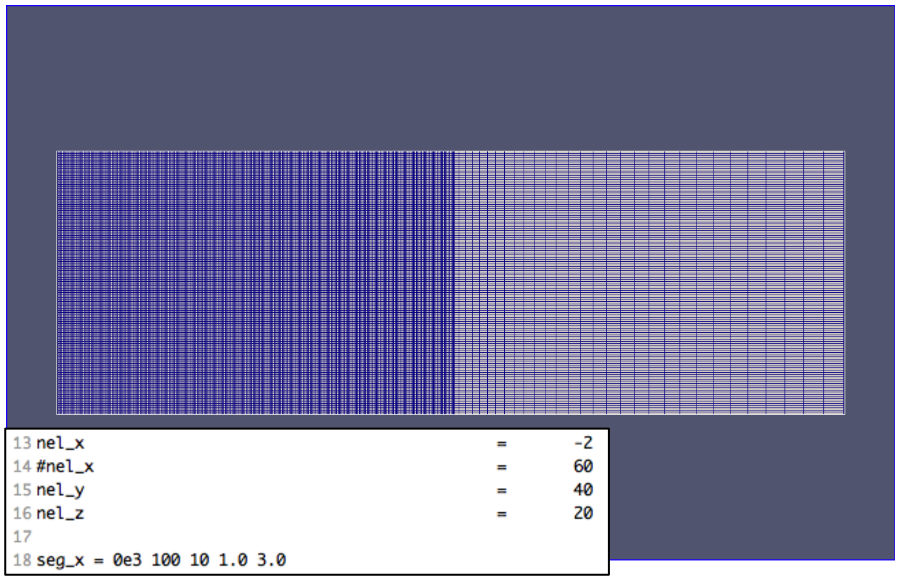
5. Rerun LaMEM with the new files in the **./MatlabInputParticles**

```
mpiexec -n 2 LaMEM -ParamFile Subduction_MATLAB_Particles_canonical_parallel.dat
```

6. Examples of variable meshes and LaMEM input parameters (marker domain):



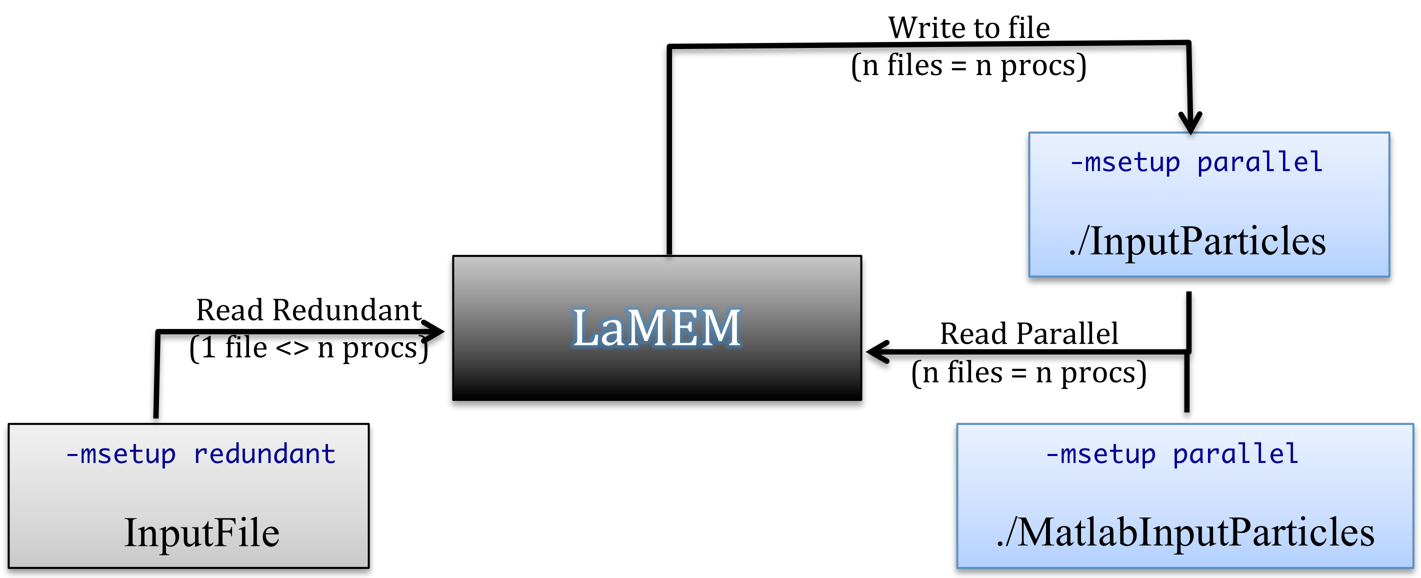
```
nel_x                                          =      -3
#nel_x                                         =      60
nel_y                                          =      40
nel_z                                          =      20

seg_x = -60e3 30e3 5 50 5 1.0 1.0 1.0
```



```
nel_x                                          =      -2
#nel_x                                         =      60
nel_y                                          =      40
nel_z                                          =      20

seg_x = 0e3 100 10 1.0 3.0
```

4) Write markers to file in LaMEM and re-use them as initial markers (Parallel)

You can write to file markers after a certain timestep in LaMEM and re-use them as initial markers (more info later). See diagram below to see how it works


Write to file
(n files = n procs)
-msetup parallel
./InputParticles
Read Redundant
(1 file <> n procs)
LaMEM
Read Parallel
(n files = n procs)
-msetup redundant
InputFile
-msetup parallel
./MatlabInputParticles