Adaptive Mesh Particle Simulator (AMPS)

User manual

al.

October 11, 2024

Contents

1 Introduction **2**

2 Installation and execution of the code **3**
2.1 Stand-alone mode . . . 3
2.1.1 CVS installation . . . 3
2.1.2 SPICE installation . . . 3
2.1.3 Installation command . . . 3
2.1.4 Arguments for `Config.pl` . . . 4
2.2 Component of SWMF . . . 5
2.2.1 Passing arguments from SWMF's Config.pl . . . 5
2.2.2 Reading SWMF's PARAM.in file . . . 5
2.2.3 Debugging . . . 5
2.2.4 Execution of AMPS in hybrid (MPI+OpenMP) mode as a component of the SWMF . . . 5
2.3 Restart Files . . . 6

3 List of input commands **8**
3.1 Main . . . 8
3.2 Species . . . 10
3.3 Include . . . 10
3.4 UserDefinitions . . . 10
3.5 BackgroundSpecies . . . 11
3.6 General . . . 12
3.7 ParticleCollisions . . . 13
3.8 Mesh . . . 13
3.9 Sampling . . . 13
3.10 IDF . . . 14
3.11 UnimolecularReactions . . . 14
3.12 Exosphere . . . 14
3.13 Additional parameters that are needed . . . 17

4 Examples **18**
4.1 Comet 67P/Churyumov-Gerasimenko . . . 18
4.2 Mercury Model . . . 19

Chapter 1

Introduction

This document describes Adaptive Mesh Particle Simulation (AMPS) code. The code is a part of Space Weather Modelling Framework (SWMF), but also can be used as a stand-alone model.

Chapter 2

Installation and execution of the code

2.1 Stand-alone mode

2.1.1 CVS installation

Full command for AMPS checkout:

```
cvs -d username@herot.engin.umich.edu:/CVS/FRAMEWORK checkout AMPS
```

2.1.2 SPICE installation

Go to the following URL to download the newest version of SPICE (cspice)

```
http://naif.jpl.nasa.gov/naif/toolkit_C.html
```

choose the proper version and download these two files

- cspice.tar.Z
- importCSpice.csh

Copy these two files into a folder where you want to have you SPICE installation, then

```
/bin/csh importCSPice.csh
```

for installation

2.1.3 Installation command

1. Add current directory '.' to your PATH, i.e. put `PATH=$PATH:.` in your .bashrc file (.bash_profile for OSX)
2. Make sure that CVS is installed on your computer.
3. Then go to a folder you wish to install AMPS in via terminal. Then type:
   `CVS checkout AMPS`
   AMPS is being set up into a subfolder called AMPS.

4. Now move into the AMPS folder:
   `cd AMPS`

5. Now choose an application you wish to use and install the code regarding your interest. In case you want to do a simulation of the Moon, type the following into the command line:
   `Config.pl -install -application=moon`
   You may choose one of the following existing applications:
   `bullet`, `CG`, `CouplerTest`, `Enceladus`, `Europa`, `Hartley2`, `Hartley2RotationBody`, `Interface`, `Mercury`, `Moon`, `Rosetta`, `Shock`
   ToDo: SHORT DESCRIPTION OF ALL THESE PRE-DEFINED CASES

6. Set the path for SPICE and for SPICE Kernels by typing the following two lines:
   `Config.pl -spice-path=/Users/Username/SPICE/cspice`
   `Config.pl -spice-kernels=/Users/Username/SPICE/Kernels`

   If you do not need any SPICE functions for you simulation it can be turned off by

   `Config.pl -spice-kernels=nospice`

Now the code is ready to compile. Type `make` into the terminal and compile the code. An executable file called "`amps`" appears in the AMPS folder. You may run the code now with the executable or submit it via a job-file on a supercomputer. It takes 6-8 hours to run on a single core. The output files are generated under the AMPS folder and data outputs are in the PT/plots folder.

2.1.4 Arguments for `Config.pl`

When using the Config.pl for setting up the code we give values to its arguments by using the `=` sign, for example:
`Config.pl -install -application=moon`
passes the value `moon` for the `application` argument.
The possible arguments are:

- `Config.pl -application` when setting up the code before compiling it we can choose the object of our interest: `bullet`, `CG` , `CouplerTest` , `Enceladus` , `Europa` , `Hartley2` , `Hartley2RotationBody` , `Interface` , `Mercury` , `Moon` , `Rosetta` , `Shock`

- `Config.pl -help` self-explanatory.

- `Config.pl -ices-path` in case we use the code in stand-alone mode, we have to use a pair of input files generaded by BATSRUS: icesCellCenterCoordinates.MHD.dat and ices.data.dat. We have to set the path for these files with this argument.

- `Config.pl -show` shows the current settings of Config.pl and AMPS, showing the paths and application we currently use.

- `Config.pl -spice-kernels` sets the path for SPICE Kernels.

- `Config.pl -spice-path` sets the path for SPICE.

2.2 Component of SWMF

2.2.1 Passing arguments from SWMF's Config.pl

2.2.2 Reading SWMF's PARAM.in file

2.2.3 Debugging

2.2.4 Execution of AMPS in hybrid (MPI+OpenMP) mode as a component of the SWMF

Running of AMPS on the same nodes with BATSRUS

The SWMF LAYOUT.in
This is the layout for running OH and PT on ivy nodes.

```
Name First Last Stride
=====================
#COMPONENTMAP
PT 0 9999 20 ! PT runs on all PE-s
OH 0 9999 1 ! OH runs on all PE-s
#END
```

The job script on Pleiades:

```
#PBS -S /bin/csh
#PBS -N SWMF
#PBS -l select=4:ncpus=20:model=ivy
#PBS -l walltime=01:00:00
#PBS -q devel
#PBS -j oe
#PBS -W group_list=s0799
#PBS -m e
module load comp-intel/2015.3.187
module load mpi-sgi/mpt
cd $PBS_O_WORKDIR

setenv OMP_NUM_THREADS 20

setenv MPI_MSGS_PER_HOST 100000
setenv MPI_MSGS_PER_PROC 100000
setenv MPI_MSGS_MAX 100000
mpiexec ./SWMF.exe ¿ runlog_d́ate +
exit

if(! -f SWMF.SUCCESS)exit
```

Running of AMPS on nodes separate from those used by BATSRUS

The layout for running OH and PT:

Name First Last Stride
==========
#COMPONENTMAP
PT 0 1 1 ! PT runs on all PE-s
OH 2 9999 1 ! OH runs on all PE-s
#END

The job script for Pleiades:

#PBS -S /bin/csh
#PBS -N SWMF
#PBS -l select=2:mpiprocs=1:ompthreads=40:model=ivy+2:mpiprocs=20:model=ivy
#PBS -l walltime=01:00:00
#PBS -q devel
#PBS -j oe
#PBS -W group_list=s0799
#PBS -m e
module load comp-intel/2015.3.187
module load mpi-sgi/mpt
cd
$PBS_O_WORKDIR

setenv MPI_DSM_DISTRIBUTE 0

setenv MPI_MSGS_PER_HOST 100000
setenv MPI_MSGS_PER_PROC 100000
setenv MPI_MSGS_MAX 100000
mpiexec ./SWMF.exe ¿ runlog_d́ate +
exit

if(! -f SWMF.SUCCESS)exit

2.3 Restart Files

AMPS allows users to save particle data at a customed frequency and also to restart from saved particle data. One can also choose overwrite existing restart files ("overwrite") or create new files ("newfile") when saving the restart files.

```
Examples in input files:
(save restart files)
 SaveParticleRestartFile=on \\  !on,off
 file=ParticleData.restart \\
 SaveMode=overwrite  \\ !overwrite, newfile
 IterationInterval=20
```

```
(read restart files)
RecoverParticleData=on    \\ !on, off
file=ParticleData.restart
```

Users can also choose to save and read additional data in the restart files. In order to do this, users need to define the functions to save and read the customized additional information in the restart data. The argument of the functions are pointers to the restart file. Then the function PIC::Restart::SetUserAdditionalRestartData() is called to pass the user defined functions to core modules.
Below is one example of applying the functions.

```
...

void saveRestartData(FILE* fname){
// Only the root processor can write.
  if (PIC::Mesh::mesh->ThisThread==0) {
    fwrite(&iterNumber, sizeof(long int),1, fname);
    std::cout<<"save iter number="<<iterNumber<<std::endl;
  }
}

void readRestartData(FILE* fname){
  fread(&iterNumber, sizeof(long int),1, fname);
  std::cout<<"read iter number="<<iterNumber<<std::endl;
}

...

int main(){
...
 PIC::Restart::SetUserAdditionalRestartData(&readRestartData,&saveRestartData);
// should be put before the iteration loop.
...
  for (long int niter=0;niter<nTotalIterations;niter++) {

...
  }









}
```

Chapter 3

List of input commands

Input commands are set in logical blocks, which are marked by lines `#NAME` - beginning of a block **NAME** - and `#endNAME` - end of that block. Input commands are listed below sorted by blocks.

The file containing the input parameters ends with the keyword `#end`.

3.1 Main

The section should start with `#main` and finish with `#endmain`.

- **SpeciesList**=*species-list*

  Definition of the species used in the simulation, names of the species should be the same as they are used in the source code.
  `SpeciesList=Na,H,O2PLUS,O_THERMAL,NA_Plus`

- **makefile**

  Obsolete but still can be used. Introduces changes in the Makefile.
  `makefile MPIRUN=mpirun -np 16`
  `makefile SPICE=/Users/MyUser/MySpiceFolder`
  `makefile SOURCES=src` self-explanatory
  `makefile RUNDIR=run` self-explanatory
  `makefile ExternalModules=` [models/exosphere]

- **ForceRepeatableSimulationPath**=[off,on]
  Force AMPS to run a repeatable calculations by disabling all code optimizations/procedures that are executed based on the actual execution time
  `ForceRepeatableSimulationPath=on`
  `ForceRepeatableSimulationPath=off`

- **DebuggerMode**=[off,on]

  Switches ON/OFF debugger mode of execution

- **CouplerMode**=[off,swmf,ices]

  Defines the mode of coupler execution, three are available:
  `CouplerMode=off` - no coupling,
  `CouplerMode=swmf` - coupling within SWMF,
  `CouplerMode=ices` - coupling using ICES tool.

- **SourceDirectory**=*source-directory*

  The directory where the main source code is located, usually it is `src` folder.
  `SourceDirectory=src`

- **ProjectSourceDirectory**=*project-source-directory*

  The directory where the source code for particular project is located.
  `ProjectSourceDirectory=srcEuropa`

- **WorkingSourceDirectory**=*working-source-directory*

  The directory where the code will be assembled and compiled (choose the one that doesn't exist in the repository in order to avoid deletion of source code files!)
  `WorkingSourceDirectory=srcTemp`

- **InputDirectory**=*input-model-data-directory*
  The location of the data files that will be used in the model run.

- **TrajectoryIntersectionWithBlockFaces**=[on,off]

- **StdOutErrorLog**=[off,on]
  Switch (on/off) that controls output of the error messages on screen

- **TimeStepMode**=[SingleGlobalTimeStep,SpeciesGlobalTimeStep, SingleLocalTimeStep,SpeciesLocalTimeStep]

  Defines the way the time step is performed, four are available:
  `TimeStepMode=SingleGlobalTimeStep` - the same time step for the entire domain and all species,
  `TimeStepMode=SpeciesGlobalTimeStep` - time step is different for different species, but the same for the entire domain,
  `TimeStepMode=SingleLocalTimeStep` - time step is the same for all species, but differs within the domain,
  `TimeStepMode=SpeciesLocalTimeStep` - time step is different for different species and differs within the domain.

- **ParticleWeightMode**=[SingleGlobalParticleWeight,SpeciesGlobalParticleWeight, SingleLocalParticleWeight,SpeciesLocalParticleWeight]

  Defines the way the statistical weights of particles are defined, for are available:
  `ParticleWeightMode=SingleGlobalParticleWeight`,
  `ParticleWeightMode=SpeciesGlobalParticleWeight`,
  `ParticleWeightMode=SingleLocalParticleWeight`,
  `ParticleWeightMode=SpeciesLocalParticleWeight`.

- **ParticleWeightCorrectionMode**=[off,on]
  Switches ON/OFF.

- **ErrorLog**=*file-name*

  Name of the file to write error information to.
  `ErrorLog=error.log`

- **Prefix**=*prefix*
  The prefix used for on screen output of AMPS
  `Prefix=AMPS`

- **DiagnosticStream**=[screen,*file-name*]
  The location of the diagnostic information output. The run-time diagnostic can be printed either on screen or in a file:
  `DiagnosticStream=amps.log`
  `DiagnosticStream=screen`

- **OutputDirectory**=*output-directory*
  Name of the folder to write results to.
  `OutputDirectory=out`

3.2 Species

The file contains database of the physical parameters of the model species. During compilation the configuration script extract the parameters only of the species used in the current model runs.

The file must begin with `#species` and ends with `#endspecies`. The following are other commands

- **#component**=*specie-symbol*
  Defines the symbolic name for the specie
  `#component=O`

- **mass**=*molecule/atom mass-mass*
  The molecule/atom mass [kg]
  `mass=16*_AMU_`

- **charge**=*electric-charge*
  Electric charge [e]
  `charge=0`

3.3 Include

Parts of the input parameters can be stored in separate files and included when the code is compiling using command `#include`.

Example `#include species.input`.

3.4 UserDefinitions

Obsolete but still can be used. Execution of AMPS can be controlled through files "UserDefinition.Exosphere.h", "UserDefinition.meshAMR.h", "UserDefinition.PIC.h", and "UserDefinition.PIC.PhysicalMode that contains some settings of the model.

The switch determines which of the files will be included into the sources of AMPS during compiling.

- **Mesh** = [On/Off]

- **PIC** = [On/Off]

3.5 BackgroundSpecies

Describes interaction of the model particles with the background species

- **BackgroundSpecies**=[on,off]
  The switch defines whether the model will be used in the run.

- **SpeciesList**=*species-list*
  The list of the background species used in the simulation. Example:

  ```
  SpeciesList=CO2,O
  ```

- **CollisionMode**=[on,off]
  The switch determines whether collisions with the background species will be simulated

- **LoadUserDefinitions**=[on,off],UserDefinitionsFileName=*file-name*
  Makes the compiler to include the file *file-name* in compiling. Example:

  ```
  LoadUserDefinitions=on \\
  UserDefinitionsFileName=UserDefinition.PIC.BackgroundAtmosphere.h
  ```

- **CollisionCrossSection**=[function,const]
  Defines the collision cross section between the model particles and the background species. Parameters `const` is used to define the hard sphere collision cross section, an `function` is for a user-defined collision cross section. Example (function):

  `CollisionCrossSection= function \\`
  `FunctionName=`*function-name*

  Example (const):

  ```
  CollisionCrossSection= const \\
  const(Na,CO)=3e-15 \\
  const(Na,O)=3e-14
  ```

- **CollisionScatteringAngle**=[function,isotropic]
  Defines the model for distributing of the scattering angle after collision. `isotropic` is for the isotropic angle distribution, and `function` is for a user-defined function. Example (isotropic):

  ```
  CollisionScatteringAngle=isotropic
  ```

  Example (function):

  `CollisionScatteringAngle=function \\`
  `FunctionName=`*function-name*

- **InjectConditionBackgroundParticle**=[on,off]
  Inject background particle into the system is a injection condition is met.

- **RemoveConditionModelParticle**=[on,off]
  Remove a model particle after collision with the background species if a condition is met.

3.6 General

- **InitialSampleLength**=*sample-length*
  The number of iterations that will be used in the simulation. Could be changed during the run.

- **define** *macro macro-value*
  Add definition of *macro* into pic/picGlobal.dfn

- **TrajectoryTracing**=[on, off]
  Turns on and off the procedure for tracing of the individual particle trajectories

- **MaxSampledTrajectoryNumber**=*toatl-sampled-trajectories-number*
  The total number of the particle trajectories that will be traced for each species

- **MaxMeshRefinmentLevel**=*max-mesh-refinment-level*
  The maximum number of the refinement levels

- **EnforceRequestedMeshResolution**=[on,off]
  Die if the requested mesh resolution is finer than that permitted by the settings of the model

- **TestRunTotalIteration**=*iteration-number*
  The number of the iterations before the output of the code test simulation results for the nightly test

- **ReferenceInjectionParticleNumber**=*reference-number*
  The number of the model particles that should be injected by the code each iterations. Used for evaluations of the particle weight.

- **NastranSurfaceUserData**=[on,off]
  Use the NASTRAN surface

- **ControlParticleInsideNastranSurface**=[on,off]
  Check whether a particle is inside the surface defined by a NASTRAN mesh after finishing the particle moving step. Can significantly decrease the efficiency of the particle trajectory integration procedure. But can be useful for debugging.

- **BlockCells**=[$nCells_x$,$nCells_y$,$nCells_z$]
  The number of the "real" cells that populate a block in each direction

- **GhostCells**=[$nGhostCells_x$,$nGhostCells_y$,$nGhostCells_z$]
  The number of the "ghost" cells that populate a block in each direction

- **MaxCutCellVolumeRelativeError**=*error*
  The maximum relative error of the volume calculation. Implemented cut-cells will make the error smaller than the requested value. Can significantly degrade the run-time of the volume calculation procedure without any benefit because the cut-faces will give enough accuracy.

- **CutCellVolumeCalculationMaxRefinmentLevel**=*max-refinment-level*
  used in calculation of the cut-cell volume to limit the maximum refinement of the volume calculation (CutCell::GetRemainedBlockVolume).

3.7 ParticleCollisions

The section described the particle collision model used in the simulation. The block has to begin with `#ParticleCollisions` and ends with `endParticleCollisions`. Additional commands:

- **model** = [off,HS]
  the name of the model that will be used

- **SampleCollisionFrequency** = [on,off]
  the switch turns sampling of the collision frequentcy

- **CollisionCrossSection**=[const,function]
  defines a collision cross section
  `CollisionCrossSection=function` *function-name*

  `CollisionCrossSection=const`
  when constant collision cross section is used its value has to be defined for each pair of species with a constant or a function
  `const (H2O,H2O)= 2.0E-19`
  if a collision cross section is not defined for a pair that its value is assumed being zero (no collisions between the species)

3.8 Mesh

The block contains settings of the AMPS mesh. The block begins with `#mesh` and ends with `#endmesh`.

- **define** Macro *Macro-Value*
  The value of the macro is saved in /src/meshAMR/meshAMRdef.h

3.9 Sampling

The section controls the non-standard AMPS sampling routines. The section begins with `#Sampling` and ends with `#endSampling`.

- **SampleParallelTangentialKineticTemperature**=[on,off], direction=[]
  Turn on sampling of the parallel and tangential temperatures. The direction can be defined by a constant vector or a function
  `direction=const(lx,ly,lx)`
  `direction=function(`*function-name*`)`

- **VelocityDistributionSampling**=[on, off], x=(), nSampleIntervals,vmin,vmax
  Define locations for sampling the velocity distribution function. x is the list of the locations, *nSampleIntervals* is the number of the intervals in the velocity space, and *xmin* and *xmax* are the minimum and maximum velocity limits of the distribution function. Example:

  ```
  VelocityDistributionSampling=on !on,off \\
  x=(7.6E5,6.7E5,0.0), (2.8E5,5.6E5,0.0), (-2.3E5,3.0E5,0.0) \\
  nSampleIntervals=500 \\
  vmin=-40e3, vmax=40e3
  ```

3.10 IDF

The section defines the model of the internal degrees of freedom. The section has to begin with `#IDF` and ends with `#endIDF`.

- **Model**=[off,LB,qLB]
- **vtRelaxation**=[on,off]
- **rtRelaxation**=[on,off]
- **vvRelaxation**=[on,off]
- **nVibModes**
  the number of the vibrational modes
  Example: `nVibModes(CO2=1,H2O=2)`
- **nRotModes**
  the number of the rotational modes
  Example: `nRotModes(CO2=2,H2O=3)`
- **RotZnum**
  the rotational Z_ν number. $1/Z_\nu$ is the probability of the rotationa-translational energy exchange during a collision event.
  `RotZnum(H2O=3,CO2=2)`
- **VibTemp**
  the characteristic vibrational temperature
  `VibTemp(H2O=2000,CO2=1000)`
- **TemperatureIndex***($spec_1$, $spec_1$)=index*
  Example:

  `TemperatureIndex(H2O,H2O)=0.5`

3.11 UnimolecularReactions

- **UNIMOLECULARREACTIONS**=[on,off]
- **REACTIONLIFETIMEMULTIPLIER**=*multiplier*
- **PROCESSOR**=*function-name*
- **LIFETIMEUSERFUNCTION**=*function-name*
- **REACTION** (PRODUCT, NONE), ID=*reaction-symbolic-id*, LIFETIME

3.12 Exosphere

Most of the description of the comet or planet environment will be done in this section. The following is the set of commands used in the section.

- **ReferenceGrounBasedObservationTime**

- **SpiceKernels**
- **IcesLocationPath**=*the path to ICES*
  Location of ICES
- **IcesModelCase**=*model-case*
  the set of the BATSRUS restart files that will be used with ICES
- **Define** *Macro New-Macro-Value*
  define *Macro*
- **SurfaceDataStructure**=[default,*file-name*]
  the command defines the data structure that is used by the exosphere model to keep its data on the surface of the internal boundary used in the simulation. The *default* data structure is a part of the exosphere model. The user can specify its own data structure by indicating the name of the header file (*file-name*) where the definition is located
- **SPICE**=[on,off]
  the switch allows the model to use the SPICE routines
- **ExosphereSamplingMode**=[on,off]
  ????
- **SimulationStartTime**=*the start-time of the simulation*
  Defines the physical time of the simulation. Needed when multiple astronomical bodies (Sun, planets) are involved in the simulation.
  Example: `SimulationStartTime=2009-01-24T00:00:00`
- **AddPhysicalModelHeader**=*header-list*
  The variable defines the list of headers where the project variables and functions are defined.
  Example: `addPhysicalModelHeader=Comet.h,Exosphere_Helium.h`
- **PhotolyticReactions**=[on,off], ReactionProcessor=[],LifeTime=[]
  Describes the functions that are used for modeling of the photolytic reactions (photo-ionization, photo-dissociation)

  ReactionProcessor is the function that is called by the core to model the reaction (inject the product of the reaction into the system)
  `ReactionProcessor=Comet::ExospherePhotoionizationReactionProcessor`

  LifeTime is the function that returns the numerical value of the species lifetime depending of the locations of a particle (could be in a shadow of the planet)
  `LifeTime=Comet::ExospherePhotoionizationLifeTime`
- **InitSurfaceSourceDistribution**=*function-name*
  Defines a user-defined function that is used for the initialization of the surface volatile distribution
- **TypicalSolarWindConditions** = v(v_x,v_y,v_z), B(B_x,B_y,B_z), T, n
  Definition of the typical solar wind conditions. Here, v, B, T, and n are bulk velocity, magnetic field, temperature, and number density of solar wind

- **SurfaceVolatileDensity**=[const, function]
  Defines the number density of volatiles at the surface of the internal body. The number density could be given either by a constant or a function

  Example of a constant
  `SurfaceVolatileDensity=const \\`
  `const(Na)=2.3E16`

  Example of a function
  `SurfaceVolatileDensity=function \\`
  `function=`*function-name*

- **AccommodationCoefficient**=[constant,function]
  Defines the accommodation coefficient for modeling of the particle/surface interaction. The coefficient can be defined either by a set of the species-dependent constants or a function.

  Example of a constant
  `AccommodationCoefficient=constant \\`
  `const(NA) = 0.2, const(NAPLUS)=0.2`

  Example of a function
  `AccommodationCoefficient=function \\`
  `function=`*function-name*

- **Source:ThermalDesorption**=[on,off],uThermal,VibrationalFrequency
  Defines parameters of the thermal desorption (where it was described?) . Example:
  `Source:ThermalDesorption=on \\`
  `uThermal(NA)=1.85*eV2J, VibrationalFrequency(NA)=1.0E13`

- **Source:PhotonStimulatedDesorption**= [on,off],PhotonFlux=[],CrossSection[],InjectionVelocityRange=[
  Parameters of the photon stimulated desorption injection model. Example:
  `Source:PhotonStimulatedDesorption=on \\`
  `PhotonFlux_1AU=2.0E14*1.0E4 \\`
  `CrossSection(NA)=3.0E-21*1.0E-4, InjectionVelocityRange(NA)=(10,10.0E3)`

- **Source:ImpactVaporization**=[on,off],HeliocentricDistance=[],SourceRatePowerIndex=[], SourceRate=[],SourceTemperature=[]
  Impact vaporization model. Example:
  `Source:ImpactVaporization=on \\`
  `HeliocentricDistance=1.0*_AU_ \\`
  `SourceRatePowerIndex=0.0 \\`
  `SourceRate(H2O)=2.0E26, SourceTemperature(H2O)=200.0`

- **Source:Sputtering**=[on,off], Yield=[], InjectionVelocityRange=[]
  Sputtering model. Example:
  `Source:Sputtering=on \\`
  `Yield(NA)=0.1, InjectionVelocityRange(NA)=(10,10.0E3)`

- **Source:VerticalInjection**=[on,off], SourceRate[], BulkVelocity[]
  the source model generates particles moving vertically to the local normal with constant bulk velocity. the source model is useful for testing purposes. Example:
  `Source:VerticalInjection=on \\`
  `SourceRate(O2)=1.69E22, BulkVelocity(O2)=6000.0`

- **Source:ExternalDomainBoundaryInjection**=[on,off]
  Initialize the injection processes ID for particle injection through the external boundary of the domain

- **Source:DefineSourceID**=*user-defined-source-id*
  can be use to register in AMPS' core a user-defined ID for a source process. The id is saved as a macro _EXOSPEHRE_SOURCE__ID__USER_DEFINED__*user-defined-source-id*_ in the file Exosphere.dfn. The ID can be use to distinguish particles that are produced via different source processes for the sampling purposes.

- **Source:UserDefined**=[on,off], SourceProcessCode=*code*, SourceRate=*function-name*, GenerateParticleProperties=*function-name*, ModifySurfaceSpeciesAbundance=[true,false], InitSurfaceSourceDistribution=*function-name*

  The structure describe the user defined particle injection boundary conditions. Example:
  `Source:UserDefined=on` \\
  `SourceProcessCode=Jet` \\
  `SourceRate=Comet::GetTotalProductionRateJet` \\
  `GenerateParticleProperties=Comet::GenerateParticlePropertiesBjorn` \\
  `ModifySurfaceSpeciesAbundance=false` \\
  `InitSurfaceSourceDistribution=HeliumDesorption::SurfaceInjectionDistribution.Init`

3.13 Additional parameters that are needed

1. The accuracy of the volume calculation of the cut-cells (meshAMRgeneric.h%8286)

Chapter 4

Examples

4.1 Comet 67P/Churyumov-Gerasimenko

This application of AMPS is used to model the rarefied atmosphere, the so-called coma, of the Rosetta target comet 67P/Churyumov-Gerasimenko in 3D. Both the gas and dust coma can be modeled using a realistic nucleus shape derived from inversion of a light curve obtained with Hubble Space Telescope measurements [**?**]. The gas flux distribution and the surface temperature of the nucleus are adapted from the thermophysical model from **???**, previously used in [**??**], from a spherical nucleus to a more realistic shape using the angle between the local normal and the direction of the Sun.

Input commands related to this application need to be marked by `#block ../cg.pl` at the beginning and `#endblock` at the end.

- **Gravity3D**=[off,on]

  Defines the gravity mode that will be used, two models are available:
  `Gravity3D=off` - the computation of the gravity acceleration will be done using a spherical approximation of the nucleus,
  `Gravity3D=on` - the computation of the gravity acceleration will be done summing over the contribution of about 1250 tetrahedrons meshing the nucleus shape from **?**.

- **HeliocentricDistance**=*the comets heliocentric distance*

  Defines the heliocentric distance at which the comet is located. The user can used the _AU_ notation give the distance in astronomical unit.
  Examples: `HeliocentricDistance=3.3*_AU_` for a comet located at 3.3 AU from the Sun.

- **SubsolarPointAzimuth**=*the angle between the Sun position and the x-axis*

  Defines the angle between the x-axis and the Suns position in the Z-plane, enabling to reproduce different geometries. The value of the angle must be given in radians. A value of 0 will align the Sun with the positive values of the x-axis.
  Examples: `SubsolarPointAzimuth=Pi/2` places the Sun 90 degrees with respect to the x-axis.

- **RadiativeCoolingMode**=[off,Crovisier]

  Selects the water radiative cooling mode that will be used in the simulation, two modes are available:

`RadiativeCoolingMode=off` - no radiative cooling,
`RadiativeCoolingMode=Crovisier` - uses the radiative cooling computed by **?**.

- **ndist**=[0,1,2,3,4]

  Determines the column that will be used from the table of inner boundary conditions derived from the thermophysical model from **???**. Five different distributions are available:
  `ndist=0` - corresponds to the 1.3 AU distribution,
  `ndist=1` - corresponds to the 2.0 AU distribution,
  `ndist=2` - corresponds to the 2.7 AU distribution,
  `ndist=3` - corresponds to the 3.0 AU distribution,
  `ndist=4` - corresponds to the 3.3 AU distribution,

- **BjornProductionRate**

  Gives the gas production rate that will be distributed using the thermophysical model from **???**.
  Examples: `BjornProductionRate(H2O)=1.0e24`

- **DustTotalMassProductionRate**

  Gives the total dust mass flux that will be injected in the simulation.
  Examples: `DustTotalMassProductionRate=1.0`

- **PowerLawIndex**

  Defines the index used in the power law of the dust grain size distribution that will be injected: $f = (a/a_0)^{-N}$ with N the index given in argument. Typical values of N vary between 2.7 and 4.5. It is necessary to point out that this size distribution corresponds to the injected particles, and that since the expansion velocity of the grains depends on their size, the size distribution in the coma will be different than the one injected. The
  Examples: `PowerLawIndex=4`

- **dustRmin**

  Defines the minimal radius size of the dust grains.
  Examples: `dustRmin=1.0e-7`

- **dustRmax**

  Defines the maximal radius size of the dust grains.
  Examples: `dustRmax=1.0e-2`

- **nDustRadiusGroups**

  Defines the number of dust sampling bins that will be used in the simulation.
  Examples: `nDustRadiusGroups=10`

4.2 Mercury Model

This document is a one-stop location, easier to pick through than the 2021 research article, containing information on our AMPS runs.

For the runs used in the publication, For each of two fields sets, We launched 500,000 particles, sampling all trajectories 1. Distributed uniformly across 618 h local time 2. Distributed uniformly across all latitudes except those within 20 of the north pole (to the exclusion of latitudes that would

have placed particles within the northern cusp at the start) 3. Distributed uniformly from 1.01 to 1.6 RM altitude 4. Given 1 eV of energy on launch, with a random velocity vector direction. 5. The resolution of the AMPS domain was set to .5 Rm up to 1 Rm, then to 80 km between 1 Rm and 2 Rm, and then to 7 Rm beyond 2 Rm. 6. nTotalIterations=5605