# Moving Boundary Solver

12/3/14

The Moving Boundary C++ Solver implements the Moving Boundary Parabolic Problem as defined in the Novak, Slepchenko paper. It currently supports a single species with no reaction term. A single compartment with an impermeable (zero flux) membrane, and no species outside the membrane is implemented. Values are solved using a fairly direct translation of the advection diffusion equation and it is therefore currently an explicit solver.

MovingBoundary outputs data in an HDF5 file. Additional output may be generated by debug settings.

The program is driven by an XML input file. The scope of this document is documenting the format. The format may change as new requirements become apparent. Ideally at some point we will have a w3c schema (e.g. XSD file) to formally document the structure.

# Input file format

The root is MovingBoundarySetup

Two required and three optional subelements are supported.

Required:

* problem
* report

Optional:

* save
* matlabDebug
* trace

*Notes:* The order of child elements is not important. Boolean options may use true/false or 0/1.

## problem

problem: **required**

The following are required in the problem section:

<xLimits>

<low>-1.5</low>

<high>2.1</high>

</xLimits>

<yLimits>

<low>-1.1</low>

<high>2.1</high>

</yLimits>

<numNodesX>40</numNodesX>

<numNodesY>40</numNodesY>

<maxTime>1</maxTime>

<timeStep>2.84444e-05</timeStep>

<diffusionConstant>1</diffusionConstant>

<advectVelocityFunctionX>0</advectVelocityFunctionX>

<advectVelocityFunctionY>0</advectVelocityFunctionY>

<frontVelocityFunctionX>1</frontVelocityFunctionX>

<frontVelocityFunctionY>0</frontVelocityFunctionY>

*physiology, see below*

xLimits and yLimits define the universe of the problem. During the simulation, there must be a minimum of one layer of outside nodes to the boundary.

Functions may contain variables x,y, and t.

The timeStep element is a maximum; if the Solver determines it is too large, it will reduce it. An optional <hardTime>true</hardTime> XML element overrides this behavior and instead stops the simulation if the step is too large.

**physiology section**

<physiology>

<species name="a"> <source>x/(x\*x+y\*y)^0.5\*j1(1.841183781340659\*(x\*x+y\*y)^0.5)+j1(1.841183781340659)</source>

</species>

<species>

<initial>1</initial>

<source>exp(-x)</source>

</species>

</physiology>

The physiology section must contain 1 to n *species* elements.

The name attribute is optional ; a default is generated (*u0,u1,* etc)

The source term is required. It’s an expression with symbols, x,y, t and the species names.

The initial term is optional; if omitted, the source term is used for initial values. It’s an expression with x,y, and t as symbols.

### boundary (front)

The front is defined either using the FronTier library from SUNY at Stony Brook or simple fixed geometries.

If frontier is used, the following elements are required in *problem*.

<levelFunction>x^2 + y^2 - 1</levelFunction>

<frontToNodeRatio>5</frontToNodeRatio>

The initial boundary is where levelFunction evaluates to zero.

The frontToNode ratio defines the granularity frontier will operate at relative the problem node spacing.

The fixed boundaries may be either:

<specialFront>

<rectangle>

<height>2</height>

<width>2</width>

<originx>-0.5</originx>

<originy>-0.5</originy>

<velocityx>1</velocityx>

</rectangle>

</specialFront>

With origin being the lower left corner, or:

<specialFront>

<circle>

<originx>0</originx>

<originy>0</originy>

<radius>1</radius>

<velocityx>1</velocityx>

<thetaIncrement>.01</thetaIncrement>

</circle>

</specialFront>

with origin being the center of the circle. The circle is modeled as a polygon with thetaIncrement (radians) defining the interior angle subtending each chord. (Thus, π / thetaIncrement is the number of segments in the polygon.

## report

The report section consists of multiple optional elements.

deleteExisting is a Boolean indicating the outputfile may be overwritten if present.

outputFilename is a string. If not present in the input file, it must passed on the command line.

datasetName is used to label data inside the output file. If omitted, a program generated default is used.

annotation is used to add arbitrary information to the output file. It should include child elements and values.

### timeReport

Report may contain 0 to n timeReport children. These control when a data slice is written into the output file. The values at the initial and final times are always written. Each timeReport element contains a startTime which determines went it becomes active, replacing the settings of earlier time reports. timeReports may be one of:

<timeReport>

<startTime>0</startTime>

<step>1500</step>

</timeReport>

“step,” a positive integer, is how often to record data in generation steps.

<timeReport>

<startTime>0.2</startTime>

<interval>0.01</interval>

</timeReport>

“interval,” a floating point, is how often to record data in problem domain time units.

<timeReport>

<startTime>0.5</startTime>

<quiet/>

</timeReport>

Quiet records no data. It is present to stop recording as specified by another timeReport setting.

## save

This optional element directs the Solver to save state information in a binary file to allow restarting. The elements are:

<save>

<filenamePrefix>sim</filenamePrefix>

<startTime>0.8</startTime>

<increment>0.01</increment>

</save>

filenamePrefix is the root name for the series of binary files.

startTime is when to start saving, and increment is how often.

## progress

The progress element directs moving boundary to output status at specific estimated percentage of completion. The elements are:

<progress>

<percent>1</percent>

<estimateProgress>true</estimateProgress>

</progress>

percent is a required integer between 1 and 99. The optional estimateProgress boolean directs MovingBoundary to estimate time remaining until the simulation is complete. By default, time estimates are off.

## trace

The optional trace section contains debugging output using levels of granularity; currently: verbose, trace, debug, info, warn, fatal. In addition to the overall setting some parts of the code may have their logging levels set explicitly. Note any level below “warn” may produce gigabytes of text on a finer mesh.

## matlabDebug

The optional matlabDebug section causes the Solver to output debug Matlab script (\*.m) files to evaluate specific modules. Note some settings may produce hundreds of \*.m files.

The currently available settings are:

edgefind:

collectmass:

frontmove:

tiling:

tilingMovie:

<matlabDebug>

<token>tiling</token>

<token>frontmove</token>

</matlabDebug>