EILMER CHEATSHEET *(v5.0.0)*

Common commands and files

Meta commands

`lmr help`	print general help
`lmr help -a`	list all commands
`lmr help <cmd-name>`	print help about *cmd-name*
`lmr version`	print version information related to compilation
`lmr revision-id`	print repository revision ID

Preparation

`lmr prep-gas -i <input> -o <output>`	prepare a gas model file
`lmr prep-reactions -g <gasfile> -i <input> -o <output>`	prepare reaction scheme
`lmr prep-grid [-j <file>]`	prepare a grid (default: `job.lua`)
`lmr prep-sim [-j <file>]`	prepare a simulation (default: `job.lua`)

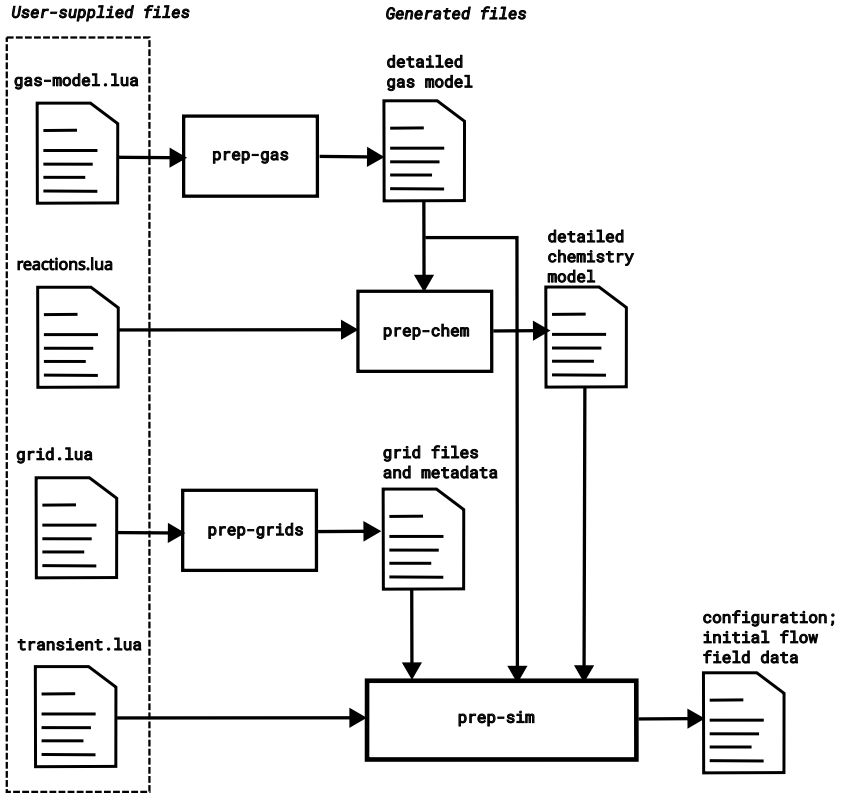
Running simulations

`lmr run`	run using shared memory
`lmr run -s 2`	restart simulation from snapshot 2
`lmr run --max-wall-clock=<s>`	limit run time to *s* seconds
`mpirun -np 8 lmr-mpi-run`	run MPI code using real-valued numbers
`mpirun -np 8 lmrZ-mpi-run`	run MPI code using complex-valued numbers
`lmr plot-diagnostics`	real-time or offline residual monitoring

Post processing

`lmr snapshot2vtk`	convert field data to VTK format
`lmr snapshot2vtk --add-vars="mach"`	add field variables when generating VTK
`lmr slice-flow -n "var1,var2" -l "blk-range,i-range,j-range,k-range" -o slice.dat`	extract variables from a slice through the domain, place in `slice.dat` file
`lmr extract-line -l "x0,y0,z0,x1,y1,z1,n" -o line.dat`	extract line of data using n samples from (x_0, y_0, z_0) to (x_1, y_1, z_1), place in `line.dat` file
`lmr probe-flow -l "x0,y0,z0" [-o probe.dat]`	sample flow field at location extract (x, y, z) [, place in `probe.dat` file]

lmr files overview



EILMER CHEATSHEET *(v5.0.0)*

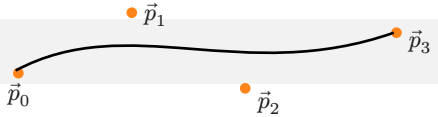
Common Lua input

Common config settings

All options are accessed as `config.`option, *e.g.* `config.solver_mode`

`solver_mode`	`transient` | `steady`
`viscous`	set `true` to include viscous terms (Navier-Stokes)
`dimensions`	2 | 3
`axisymmetric`	set `true` for 2D geometries with axial symmetry about $y = 0$
`gasdynamic_update_scheme`	common explicit schemes: `'predictor-corrector'`, `'classic-rk3'` moving grid choices: `'moving-grid-1-stage'`, `'moving-grid-2-stage'` implicit schemes: `'backward-euler'`, `'implicit_rk1'`
`cfl_value`	*float* value for target Courant-Friedrichs-Lewy number
`cfl_schedule`	table of `{time, CFL}` pairs such as `{{0.0, 0.25}, {1.0e-4, 0.75}}`
`dt_init`	*float* for initial timestep
`max_time`	*float* for maximum simulated time
`max_step`	*int* for maximum number of update steps
`dt_plot`	*float* period between writing out flow field
`dt_history`	*float* period between writing out history data
`interpolation_order`	2: high order reconstruction 1: no reconstruction
`flux_calculator`	common choices include: `'ausmdv'`, `'adaptive_hlle_asumdv'`, `'adaptive_hanel_ausmdv'`

Geometry and patches and clustering (Oh my!)

`Vector3:new{x=x0, y=y0, z=z0}`	Creates Cartesian vector object at (x_0, y_0, z_0)
`Line:new{p0=a, p1=b}`	
`Arc:new{p0=a, p1=b, centre=c}`	Arc from $\vec{a}$ to $\vec{b}$ about centre $\vec{c}$
`Arc3:new{p0=a, pmid=b, p1=c}`	Arc through 3 points: $\vec{a}$, $\vec{b}$ and $\vec{c}$
`Bezier:new{points={p0, p1, p2, p3}}`	
`Spline:new{points={p0, p1, ...}}`	Spline through points $(p_0, p_1, ...)$
`Spline2:new{filename='file.txt'}`	Spline through points in `file.txt`
`Polyline:new{segments={pathA, pathB, ...}}`	Single path built from joined segments
`CoonsPatch:new{north=..., south=..., east=..., west=...}`	Coons patch with edges *N*, *S*, *E* and *W*
`AOPatch:new{north=..., south=..., east=..., west=..., [nx=..., ny=...]}`	Area-orthogonal patch with background grid dimensions (n_x, n_y)
`RobertsFunction:new{end0, end1, beta}`	Roberts clustering function. Set `end0|1` true to cluster towards `end0|1`. `beta` controls strength of clustering.
`GaussianFunction:new{m, s, ratio}`	Gaussian clustering centred at m with width s and tightness controlled by *ratio*

Grids and block construction

`StructuredGrid:new{psurface, niv, njv, cfList}`	Initialise `StructuredGrid` object
`UnstructuredGrid:new{sgrid}`	Initialise `UnstructuredGrid` from a structured grid
`UnstructuredGrid:new{filename="grid.su2", fmt="su2text"}`	Initialise `UnstructuredGrid` from `grid.su2` in SU^2 format
`registerFluidGrid{grid=..., fsTag=..., bcTags=...}`	Register grids at `prep-grid` stage
`identifyGridConnections()`	Automate search for grid connections and apply
`makeFluidBlocks(bcDict, flowDict)`	From registered grids, make fluid blocks