|  |  |  |  |
| --- | --- | --- | --- |
| **Component/function** | **Contents** | **Input arguments** | **Resulting object** |
| Model input | Governing equations / “modelled physics”  Grid type and dimensions  Physical parameters  Numerical parameters | Inputfile / commandline /  external program | Model\_data  Grid\_data  Parameters  Error handling? |
| Grid setup | Setup of the computational grid, including connectivity, etc. | Grid\_data | Grid  Error handling? |
| Model setup | Setup of the model:  Initialization of the solution vector, primary and secondary variables, stencils,  Discretization operators in space and time | Model\_data,  Grid,  parameters | Variable\_dictionary?  Discretization\_operators / functions?  Error handling?  -> bundled in a  model object? |
| Timeloop,  Containing:  single\_timestep | Time evolution of solution | Model, grid, parameters | Solution\_vector for each (output) time step  Meta data,  error handling  -> bundled in an  output object? |
| Finalization | Clean up of program | Model, grid? | Return code |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Model/problem (equations)
* Solution\_algorithm
* Splitting data and operations!
* Split program in smaller independent components
* Clear definition of interfaces (input/output)
* List of variables used.
  + Group them in primary, secondary, …?
  + Group them in structs or other AbstractTypes? -> thinking about in which model component they are needed as input or output

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Example** | **Input** | **Output** |
| Model input |  | Ndim, grid type and grid, initial conditions, boundary conditions, parameters, model components, discretization type, solution algorithm, time integration, hardware, | Model\_input  Grid\_input  BC\_input  Parameters |
| Initialize variables/arrays |  | Model\_input  Grid\_input | Model\_state |
| Set-up grid/domain |  | Grid\_input | Grid |
| Prepare BCs |  | BC\_input | BCs |
| Set-up model/problem/matrix | A = …  B = … | Model\_state, BCs | Model (different for 1D, 2D and 3D?) |
| Update | un+1 = (A-I)\*un+1 + bn | Model: A, u (model state vector), b | New state |
| Solve ( for each variable?) | un+1 = A\b | Model: A (stencil-operator form), b | New variable |
| Discretization | for cell i:  add edge-contribution of gradient | Model | A, b |
| Vector gradient | \naba(u) or ∆u |  | Operator |
| Scalar gradient | du/dx |  | Component of operator |
| Finite difference | uR-uL/dx | u, dx | Expression in component |
|  |  |  |  |

Solve function:

Time integration/update function: un+1 = A\*un+1 + b\*un

Discretization function: e.g. for cell i: add edge-contribution of gradient

Vector gradient function: e.g. \nabla(u) or ∆u

Scalar gradient function: e.g. du/dx

Finite difference function: e.g. uR-uL/dx