



POTSDAM INSTITUTE FOR  
CLIMATE IMPACT RESEARCH

# Development of *Aeolus 1.0*

## A Statistical-Dynamical Atmosphere Model

**Dim Coumou, Alexey V. Eliseev**

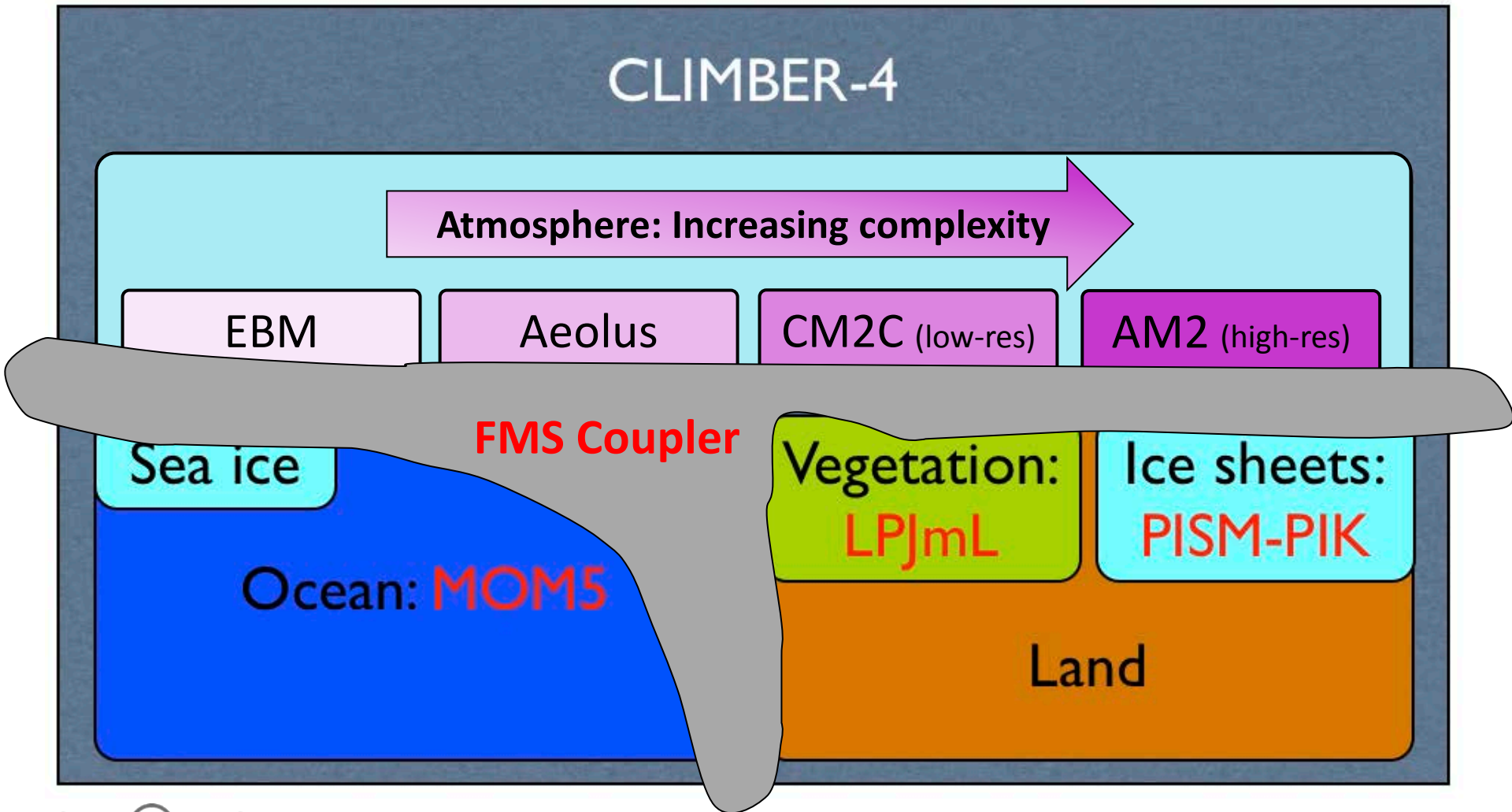
**Vladimir Petoukhov, Stefan Petri**

**ModStrat Seminar 14 Nov. 2013**

# Content

- Intro *Aeolus* and *Climber-4*
- Key Modules
- Object-oriented Code Design
  - Strict separation of *Data Storage, Gridding* and *Computations*
- Safety Mechanisms:
  - Explicit *Read-Write* Functionality, *Design-by-Contract*,  
*Automatic Range Checking*
- Documentation
- Testing / Benchmarking
  - Module-wise, Stand-alone, Fully-coupled*
- Handy Tools:
  - Totalview, valgrind, VC++* (or in general *IDEs*)

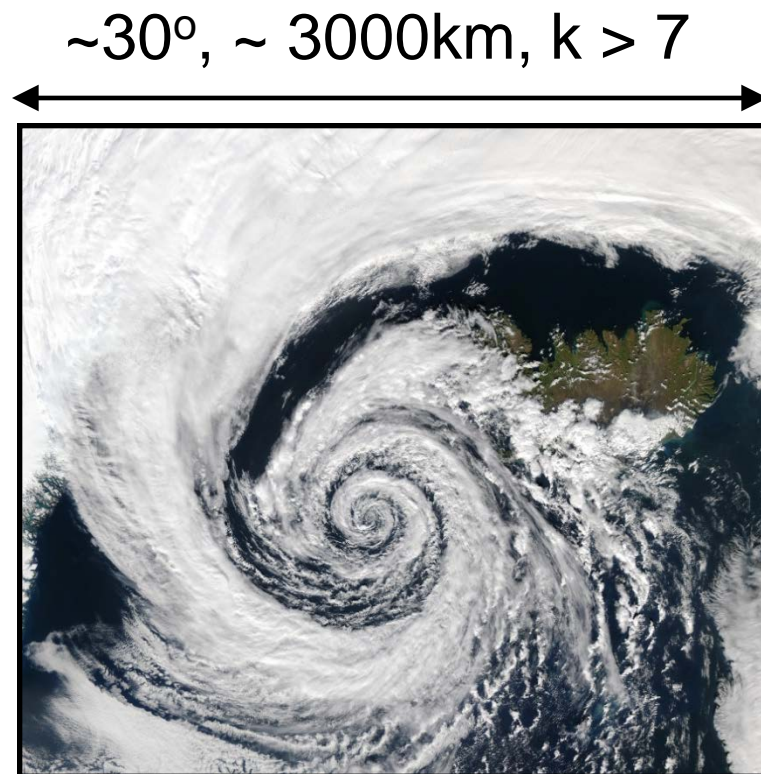
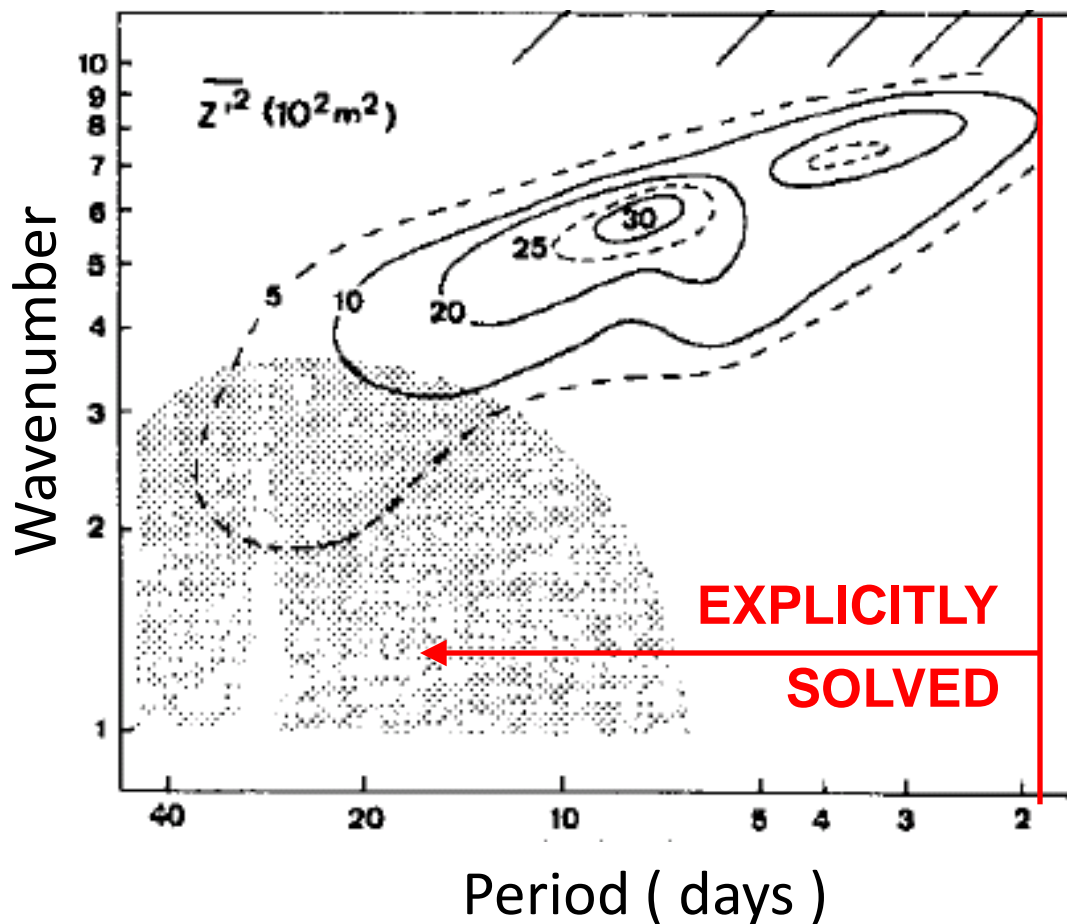
# Hierarchy of Atmosphere Models



# Synoptic Scale Eddies

GCMs

## Atmospheric Power spectrum



**HIGH RES in Space & Time**

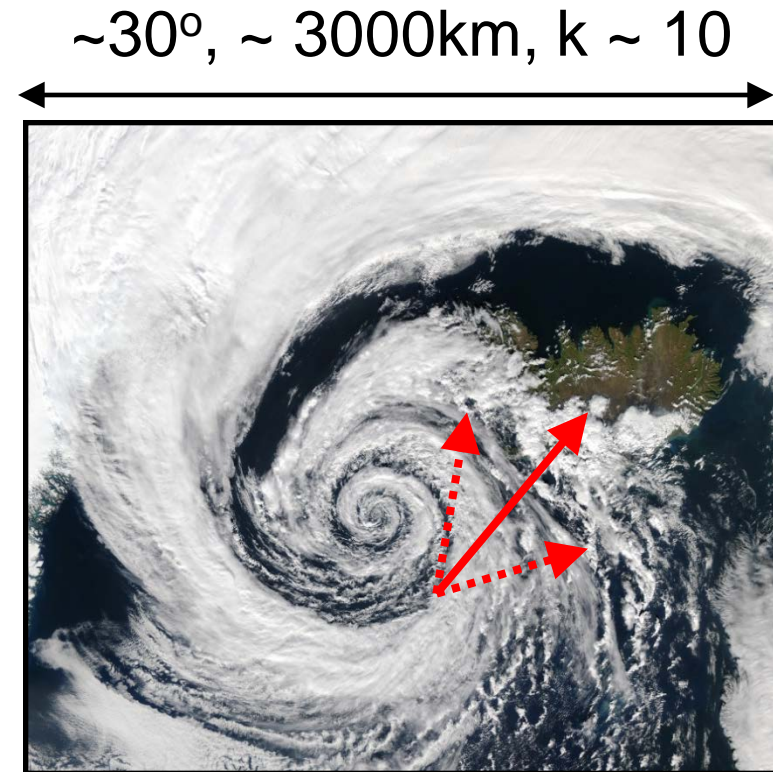
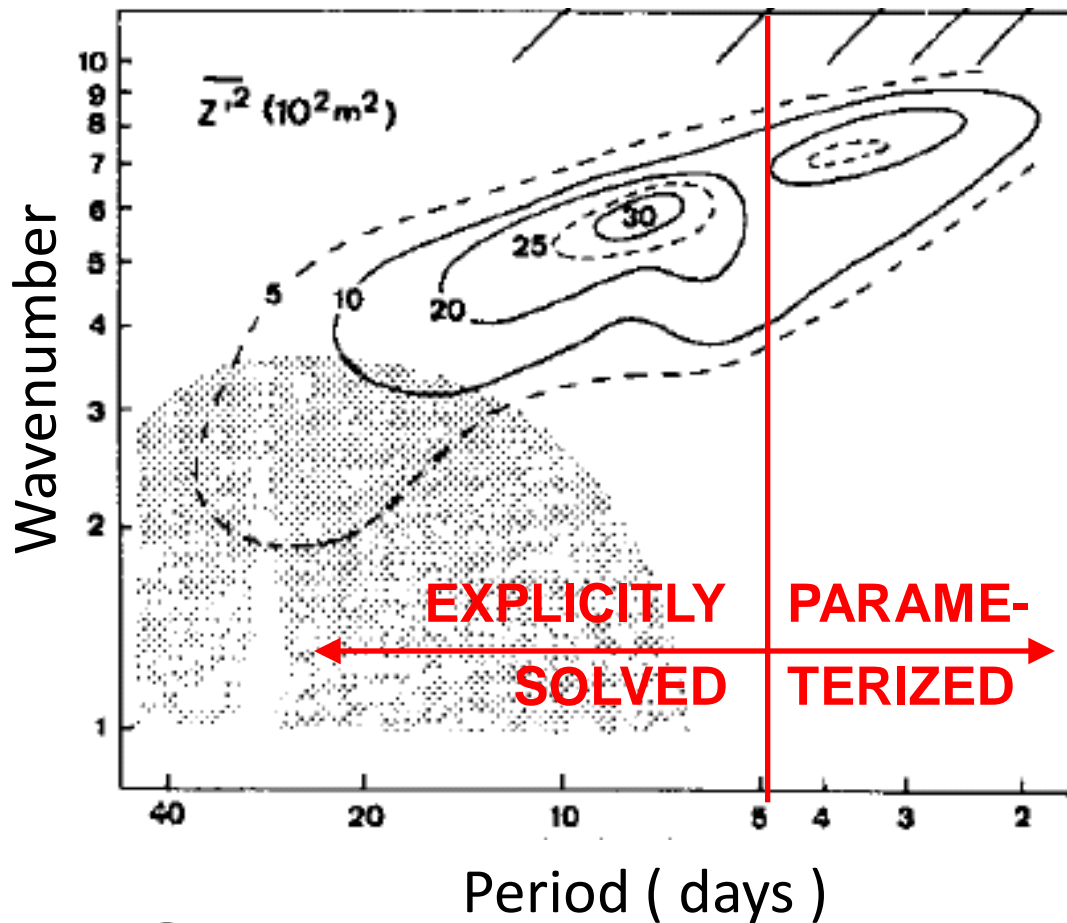


**NB: Interested in Climate rather than Weather !**

# Synoptic Scale Eddies

SDAMs

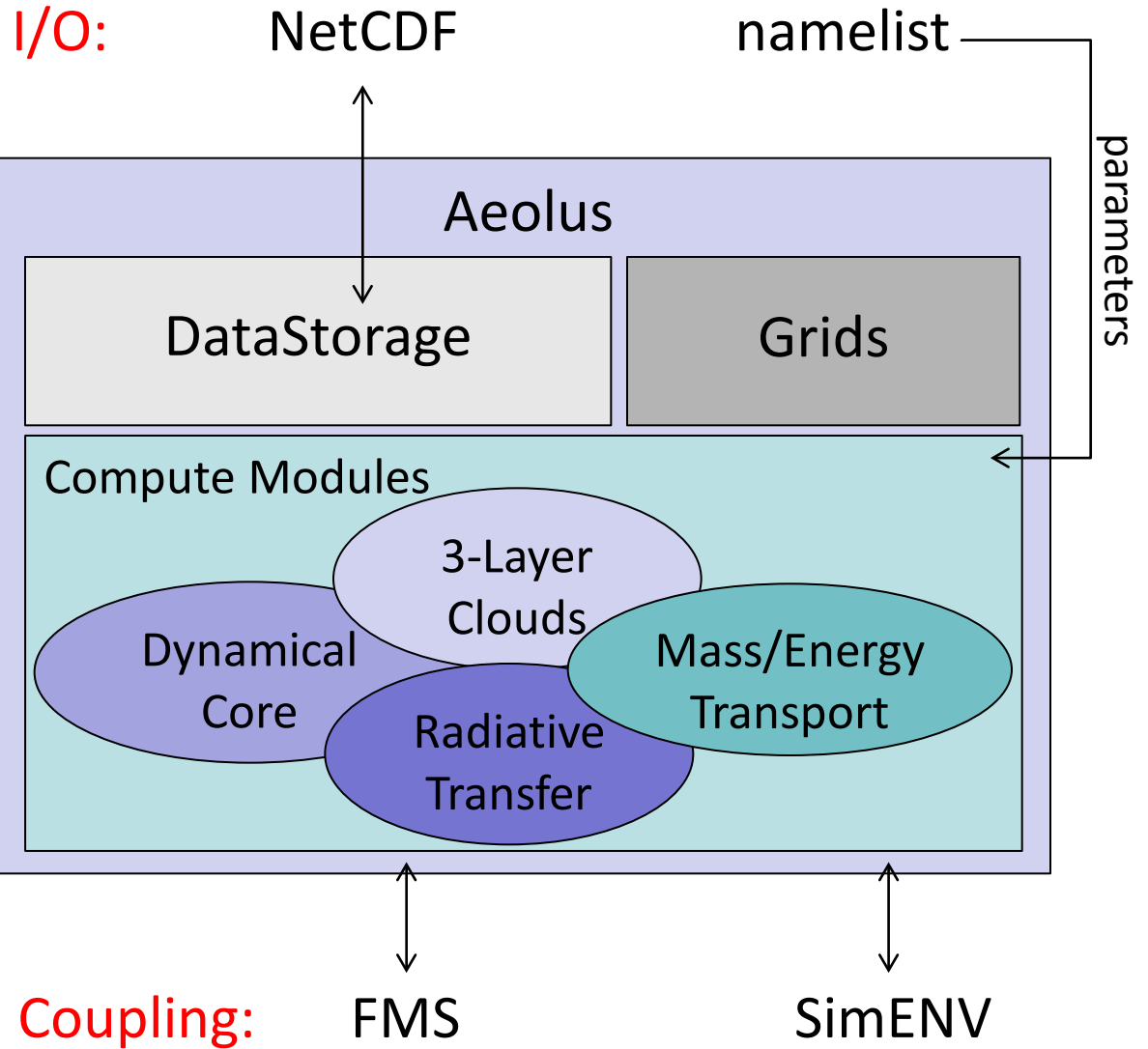
## Atmospheric Power spectrum



$$\vec{V} = \langle \vec{V} \rangle + \vec{V}'$$

# Aeolus

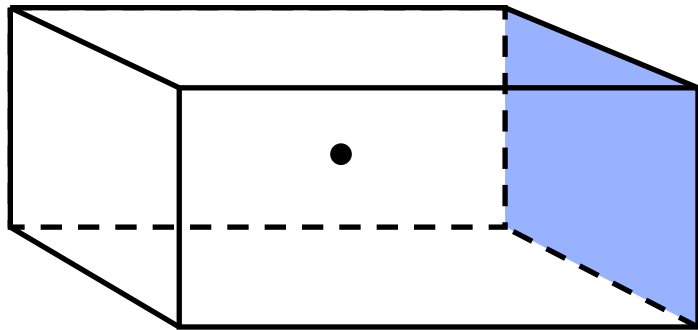
- Object-oriented C++ implementation
- ~45k code lines
- *svn* version control
- *MPI* Parallel Computing



# Grid: Core Building Blocks

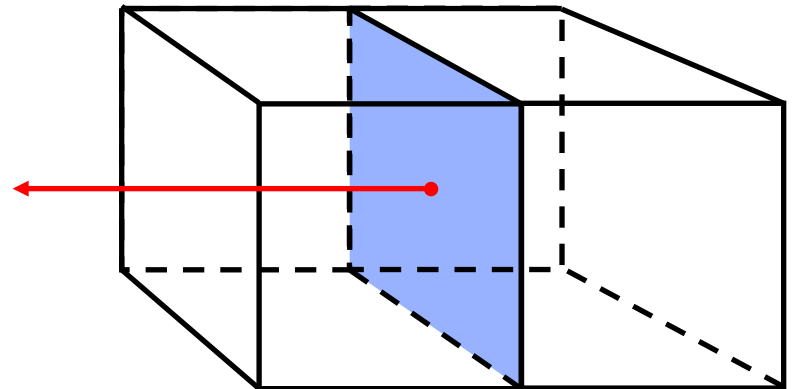
## Cell

- Coordinates
- Volume
- ID
- Boundary Flag  
(SPOLE / NPOLE / etc)
- Neighbors  
(Cells & Interfaces)



## Interface

- Coordinates
- Area
- ID
- Normal vector
- Boundary Flag  
(SPOLE / NPOLE / etc)
- Neighbors  
(Cells)





# Grids

## SphericalGrid

## Standard Spherical Grid

— vect  
— vect  
— ...  
— Func  
fast  
spec  
and

### Numerical Stability

**Finite Difference:**  $\Delta t < \frac{2\Delta x^2}{K}$

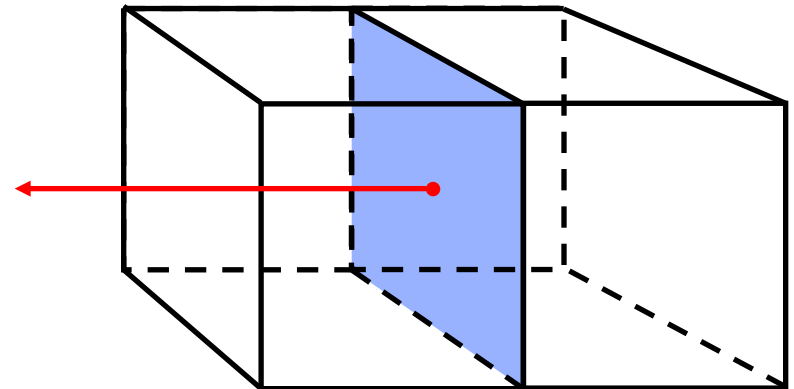
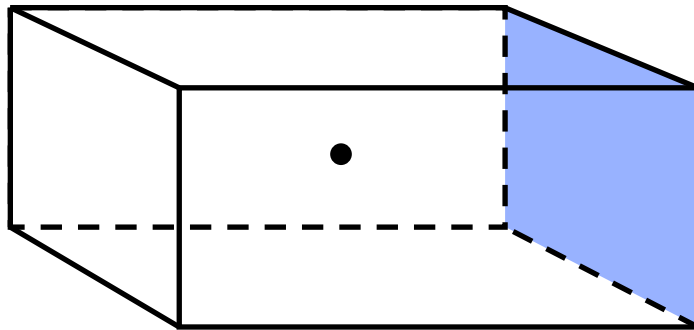
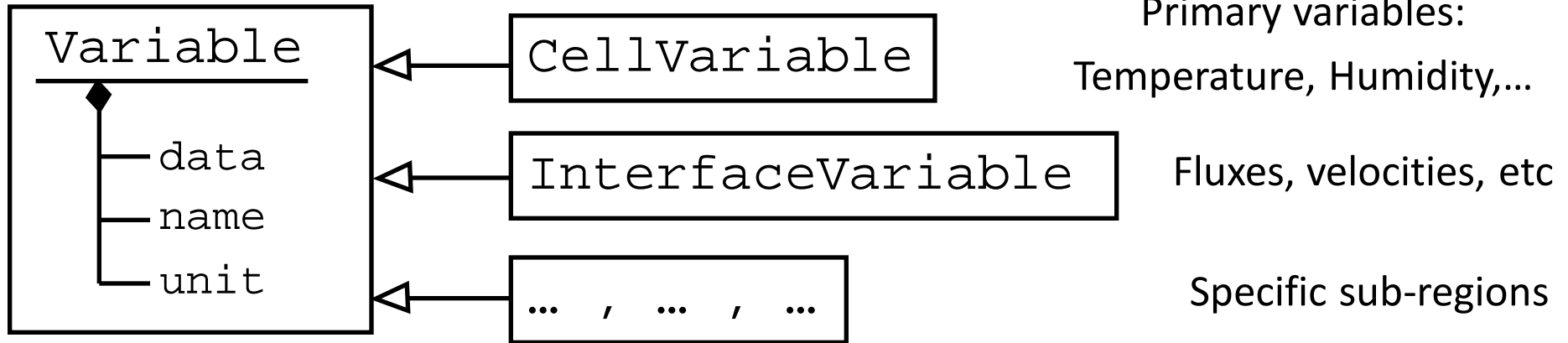
**Finite Volume:**  $\Delta t < \frac{\Delta x}{|\vec{V}|}$

‘Internally consistent, self-described grid’

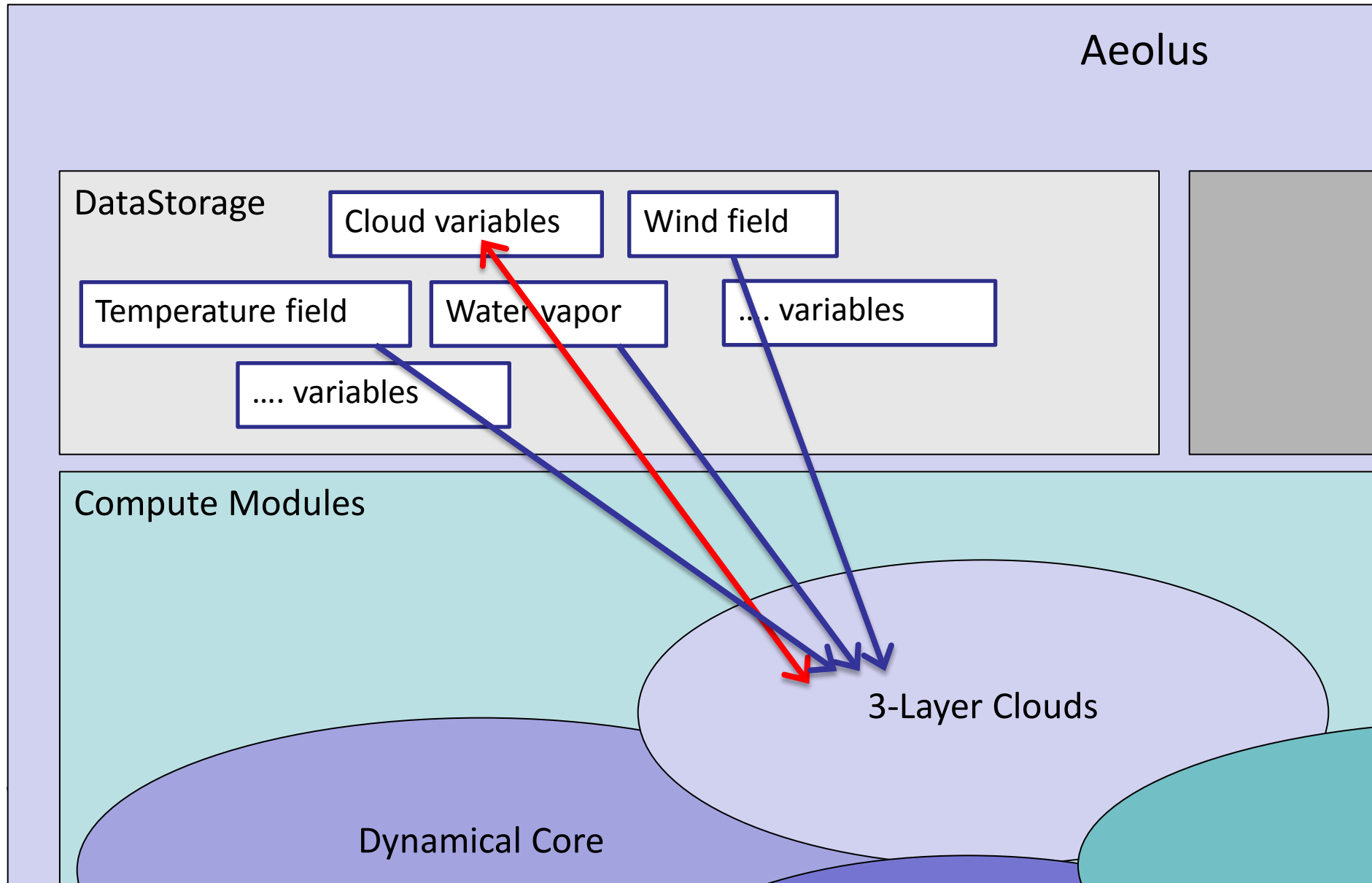
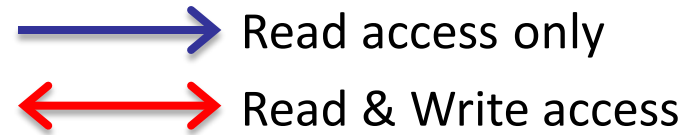




# Data Storage



# Explicit Read / Write access



# Explicit Read / Write access

Only **one** *Compute* class has write access to **one** *DataStorage* class  
This is done using specific *Access-functions* and *friend* definition

example: 1 layer cloud scheme

```
class Clouds
{
public:
    Clouds(const SphericalGrid&);
    ~Clouds(void);

    // Provide read functionality
    inline const Variable& cumulus_() const { return cumulus; };
    inline const Variable& stratus_() const { return stratus; };
    inline const Variable& total_() const { return total; };

    // Give specific write access to one compute class
    friend class Cloudiness;

private:
    Variable cumulus, stratus, total;
};
```

# Data Storage – Write-access:

## Compute classes

## DataStorage classes

<code>class Cloudiness_3Layers</code>	↔	<code>class Clouds_3Layers</code>
<code>class LargeScaleDynamics</code>	↔	<code>class LargeScaleWind</code>
<code>class SynopticScaleDynamics</code>	↔	<code>class SynopticScaleField</code>
<code>class PlanWave</code>	↔	<code>class PlanetaryWave</code>
<code>class HeatTransfer</code>	↔	<code>class Temperature</code>
<code>class WaterVaporTransfer</code>	↔	<code>class Humidity</code>
<code>class RadiativeTransfer</code>	↔	<code>class RadiativeFluxes</code>
<code>class PlanetaryBoundaryLayerPhysics</code>	↔	<code>class PlanetaryBoundaryLayer</code>
<code>class SurfaceFluxes</code>	↔	<code>class SurfaceLayer</code>



# Design-by-Contract (on the function-level)

```
void FunctionBlaBla (const Variable& v1, const Variable& v2, Variable& v3)
{
    // Valid input?
    v1.IsValid();
    v2.IsValid();

    // Calculate v3:

    ...Black box...

    // Valid output?
    v3.IsValid();
};
```



→ Checks done in *Debug-mode* only, turned-off in *Release-mode*

# Content

- Intro *Aeolus* and *Climber-4*
- Key Modules
- Object-oriented Code Design
  - Strict separation of *Data Storage*, *Gridding* and *Computations*
- Safety Mechanisms:
  - Explicit *Read-Write* Functionality, *Design-by-Contract*, *Automatic Range Checking*
- **Documentation**
- Testing / Benchmarking
  - Module-wise*, *Stand-alone*, *Fully-coupled*
- Handy Tools:
  - Totalview*, *valgrind*, *VC++* (or in general *IDEs*)



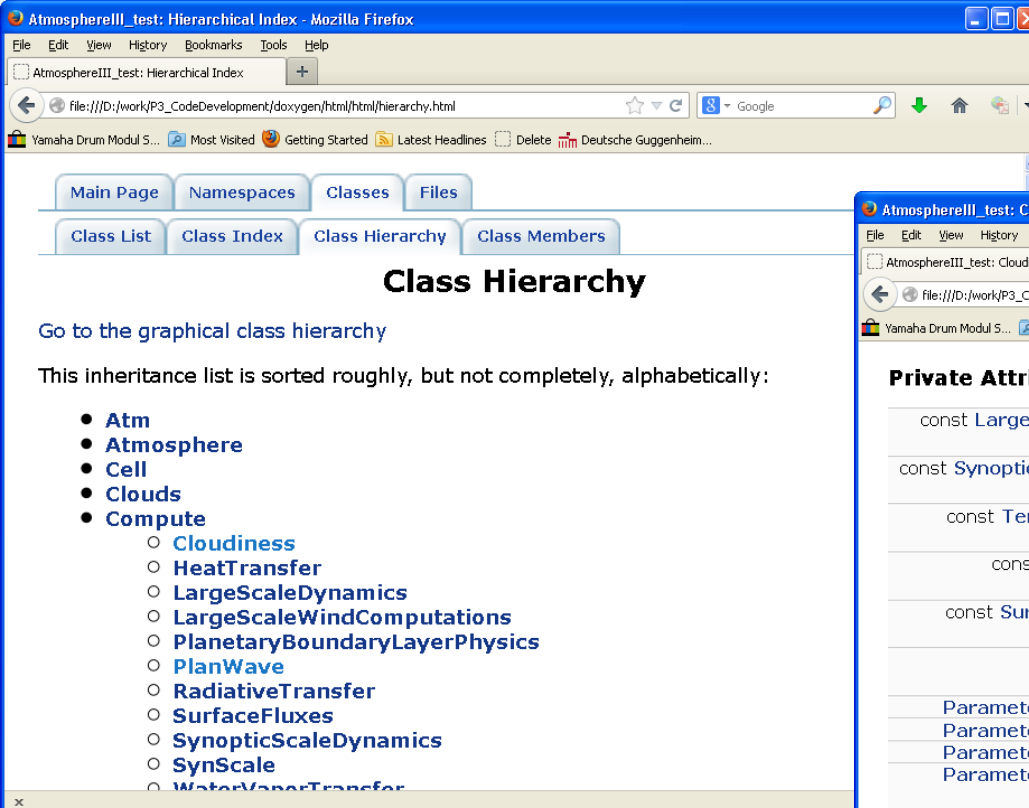
# Documentation

- **Peer reviewed**
  - Synoptic Dynamics (Coumou et al, 2011)
  - 3-Layer Clouds (Eliseev et al, 2013)
  - Dynamical Core (in progress...)
- **Technical reports (Mathematical Model Description)**

Aim: One for each *Compute* module
- **Doxygen**

Generates html/LaTeX documentation directly from source-code



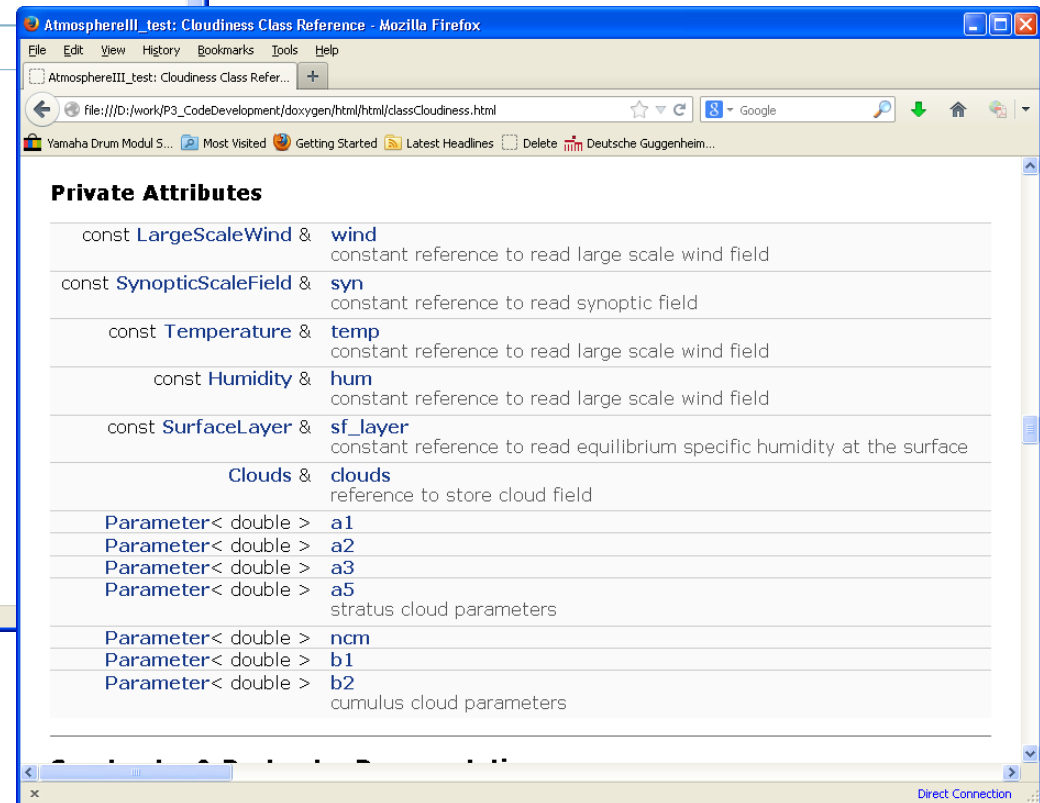


## Class Hierarchy

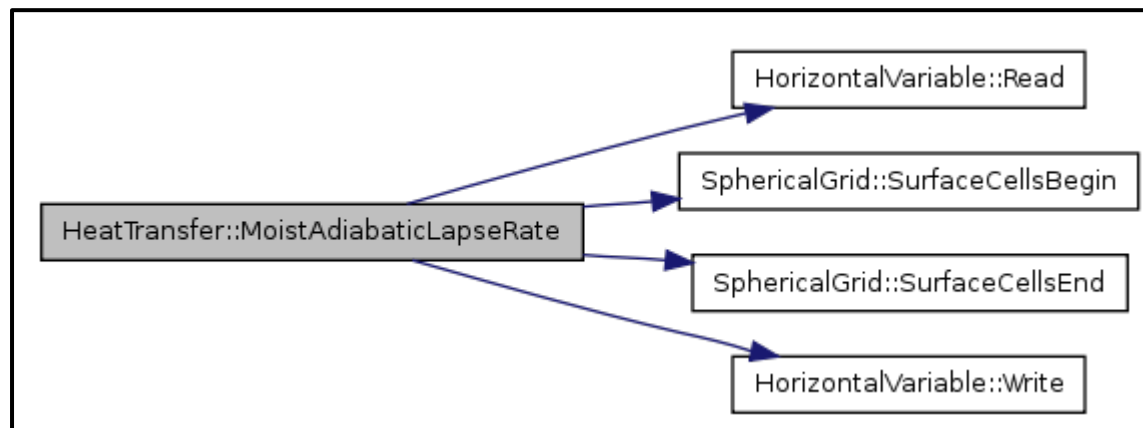
Go to the graphical class hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

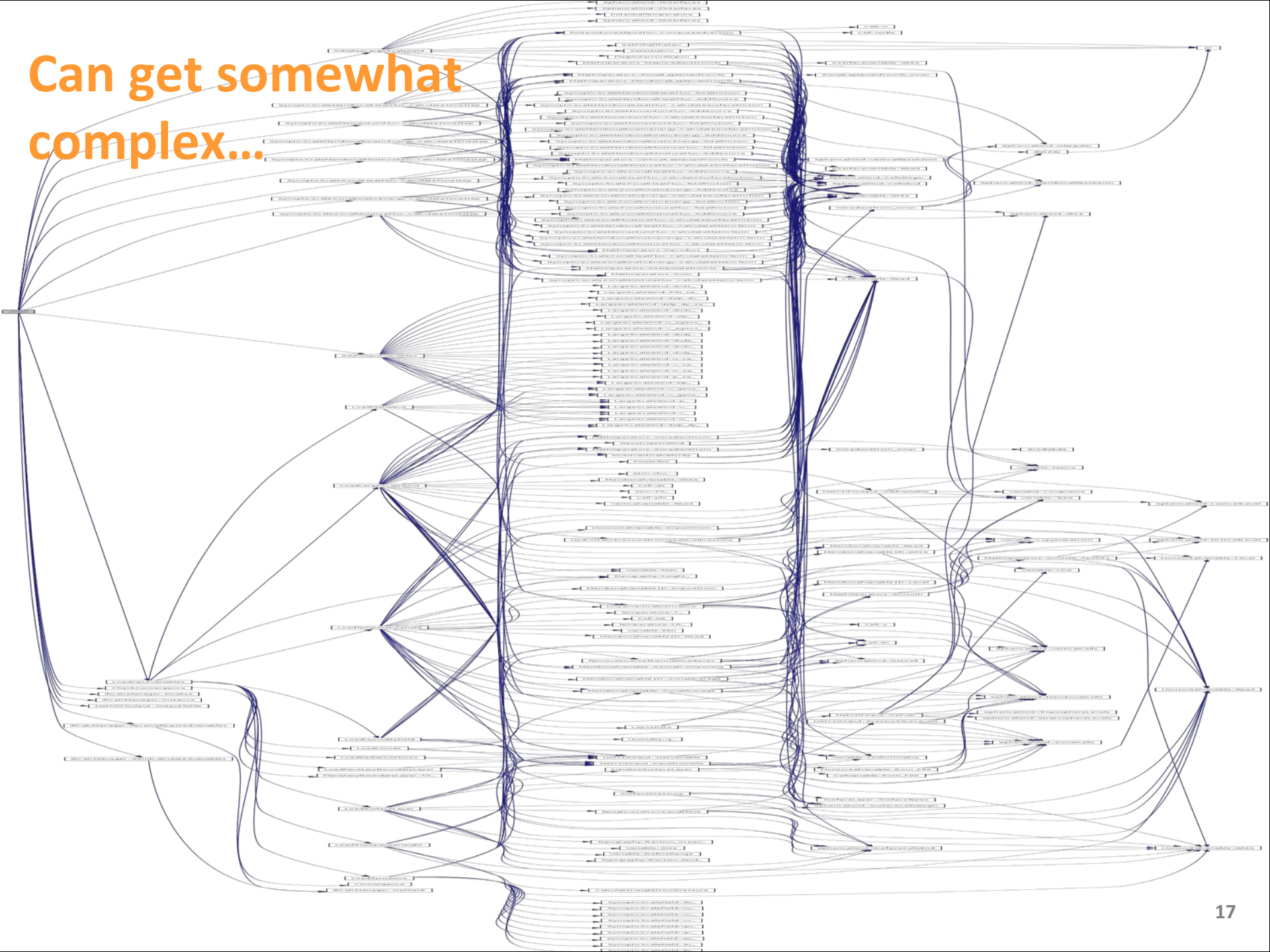
- **Atm**
- **Atmosphere**
- **Cell**
- **Clouds**
- **Compute**
  - **Cloudiness**
  - **HeatTransfer**
  - **LargeScaleDynamics**
  - **LargeScaleWindComputations**
  - **PlanetaryBoundaryLayerPhysics**
  - **PlanWave**
  - **RadiativeTransfer**
  - **SurfaceFluxes**
  - **SynopticScaleDynamics**
  - **SynScale**
  - **WaterVaporTransfer**



## Call-graphs / Dependency-graphs



Can get somewhat complex...

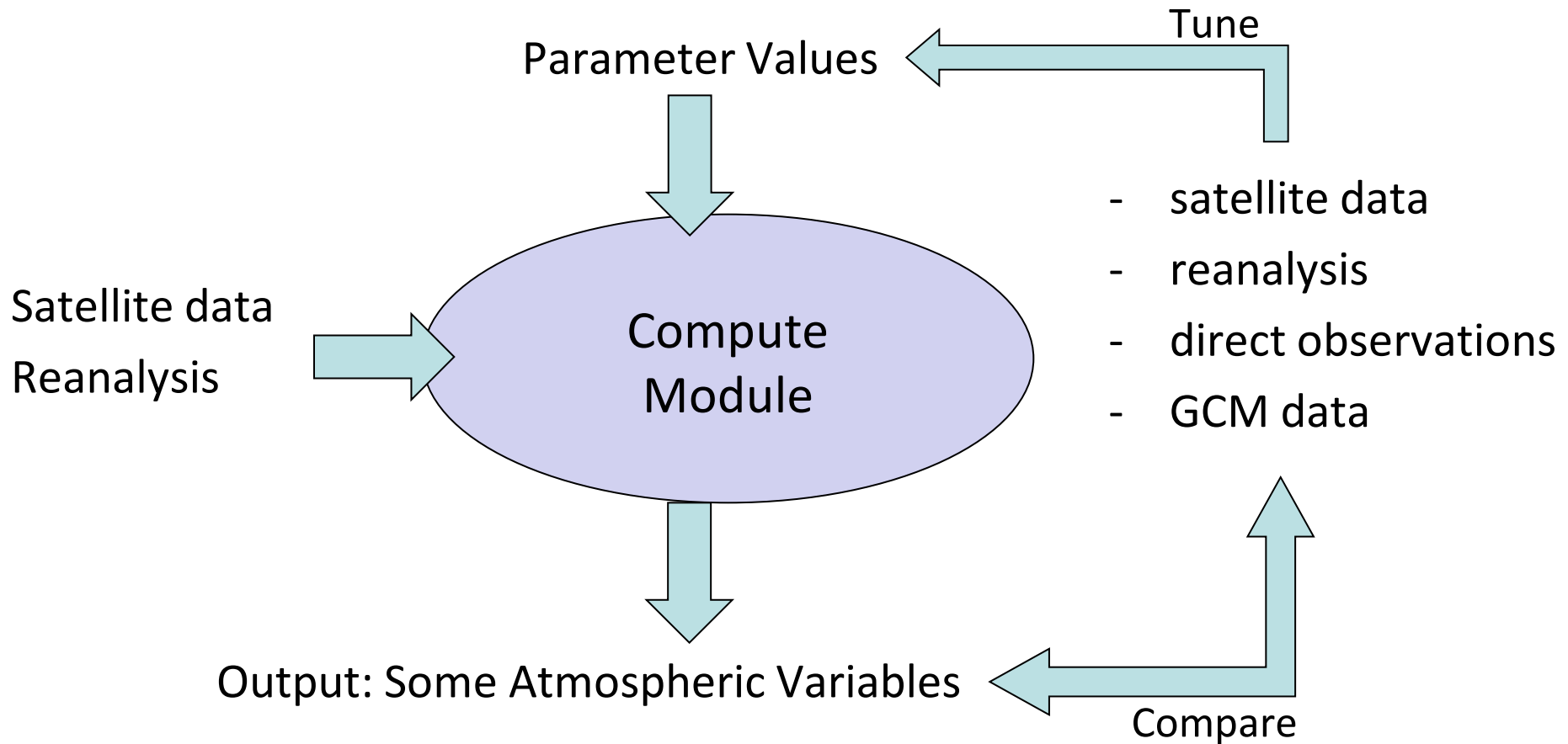


# Content

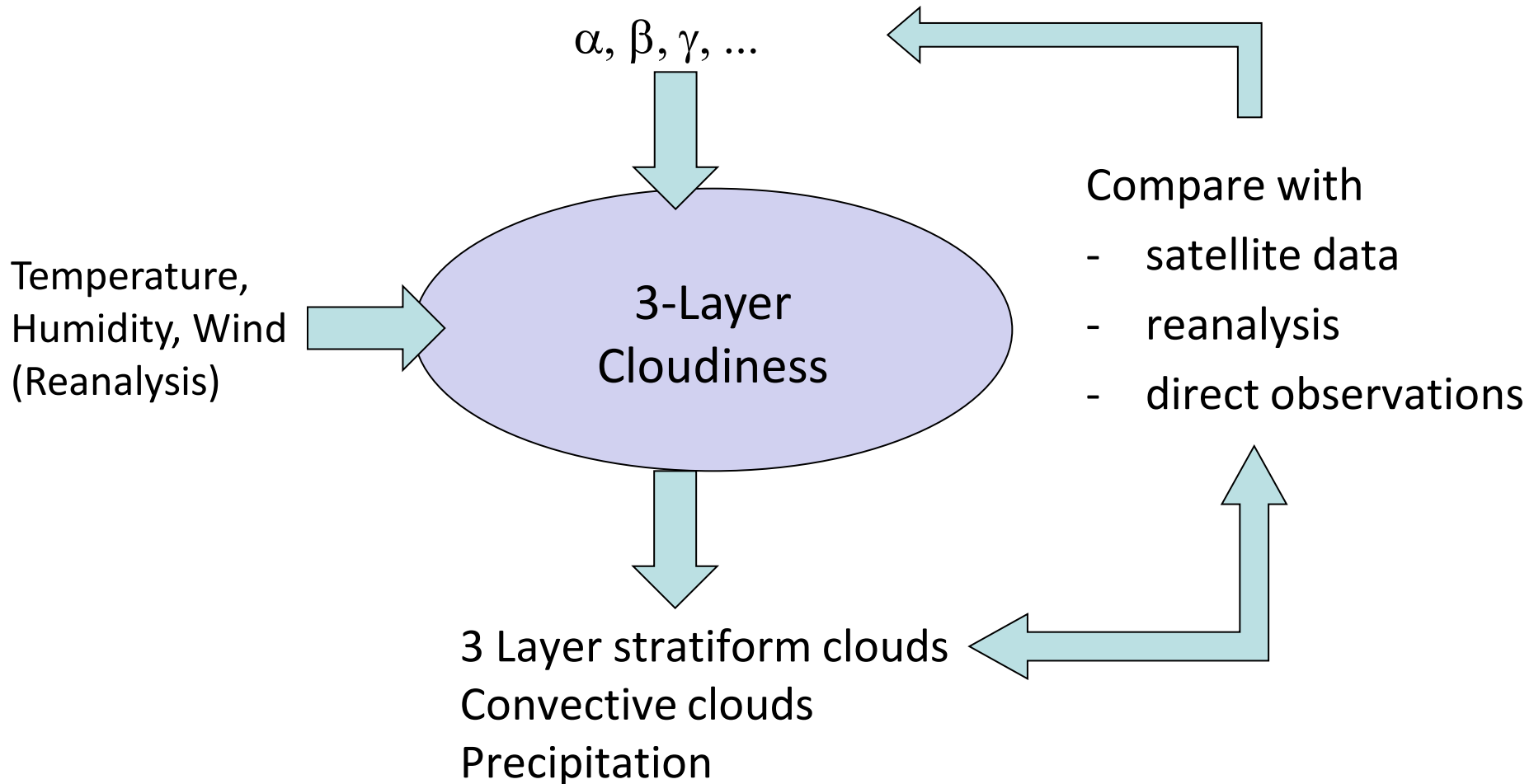
- Intro *Aeolus* and *Climber-4*
- Key Modules
- Object-oriented Code Design
  - Strict separation of *Data Storage, Gridding* and *Computations*
- Safety Mechanisms:
  - Explicit *Read-Write* Functionality, *Design-by-Contract*, *Automatic Range Checking*
- Documentation
- **Testing / Benchmarking**
  - Module-wise, Stand-alone, Fully-coupled*
- Handy Tools:
  - Totalview, valgrind, VC++* (or in general *IDEs*)



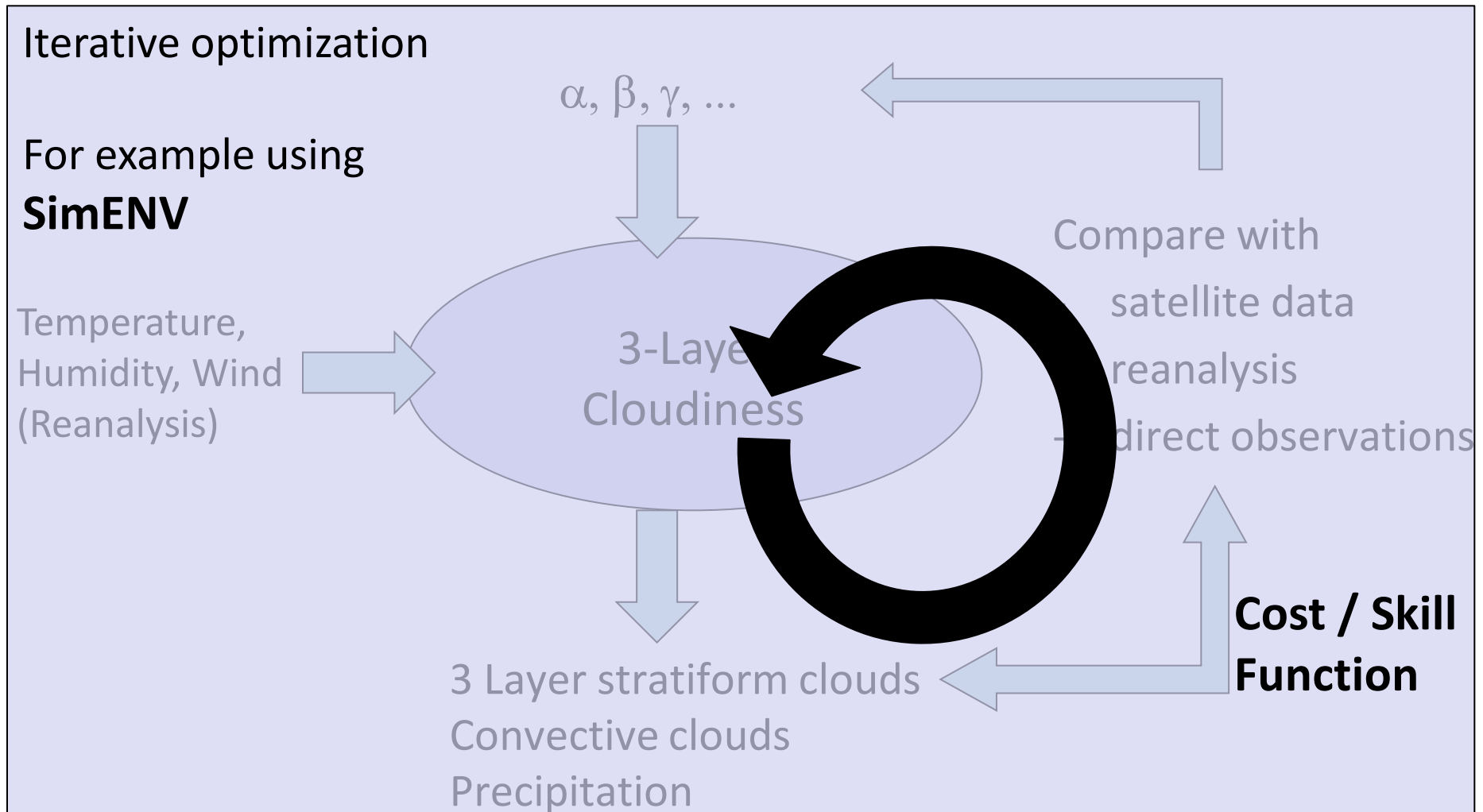
# Tuning: Module-wise



# Tuning: Example Cloudiness



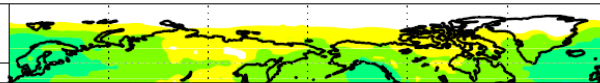
# Tuning: Example Cloudiness



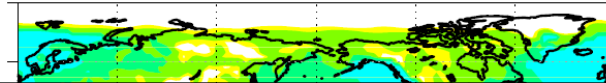
# Tuning: Module Calibration

Improves low precipitation in major desert regions

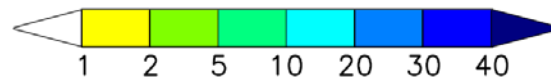
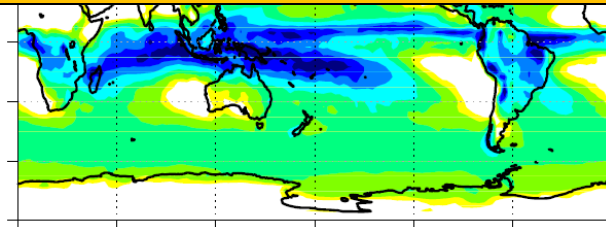
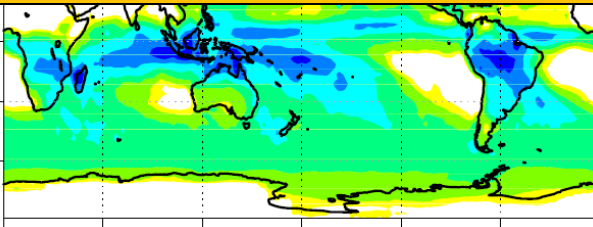
Aeolus - Initial



Aeolus – Calibrated



**Conclusion: automated calibration is very useful  
...but only when a proper Skill function can be defined!**

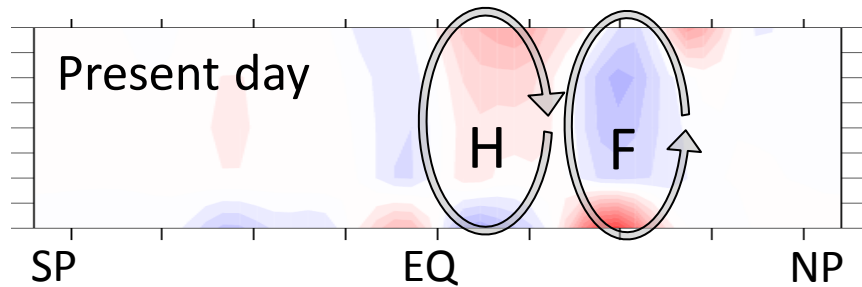


January Precipitation cm/mo

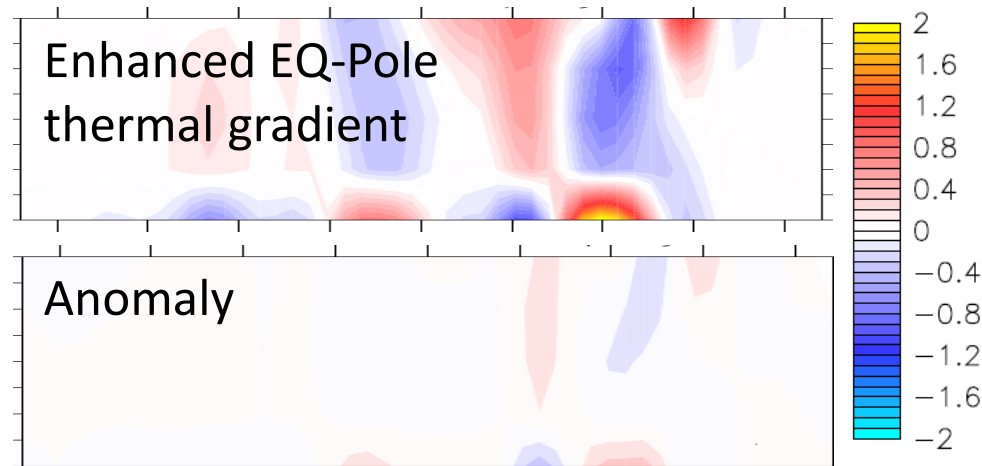


# Tuning: Stand-alone *Aeolus*

**Next Step: Sensitivity-analysis using SimENV**



Hadley & Ferrel Cell  
strengthen and widen for  
stronger T-gradients

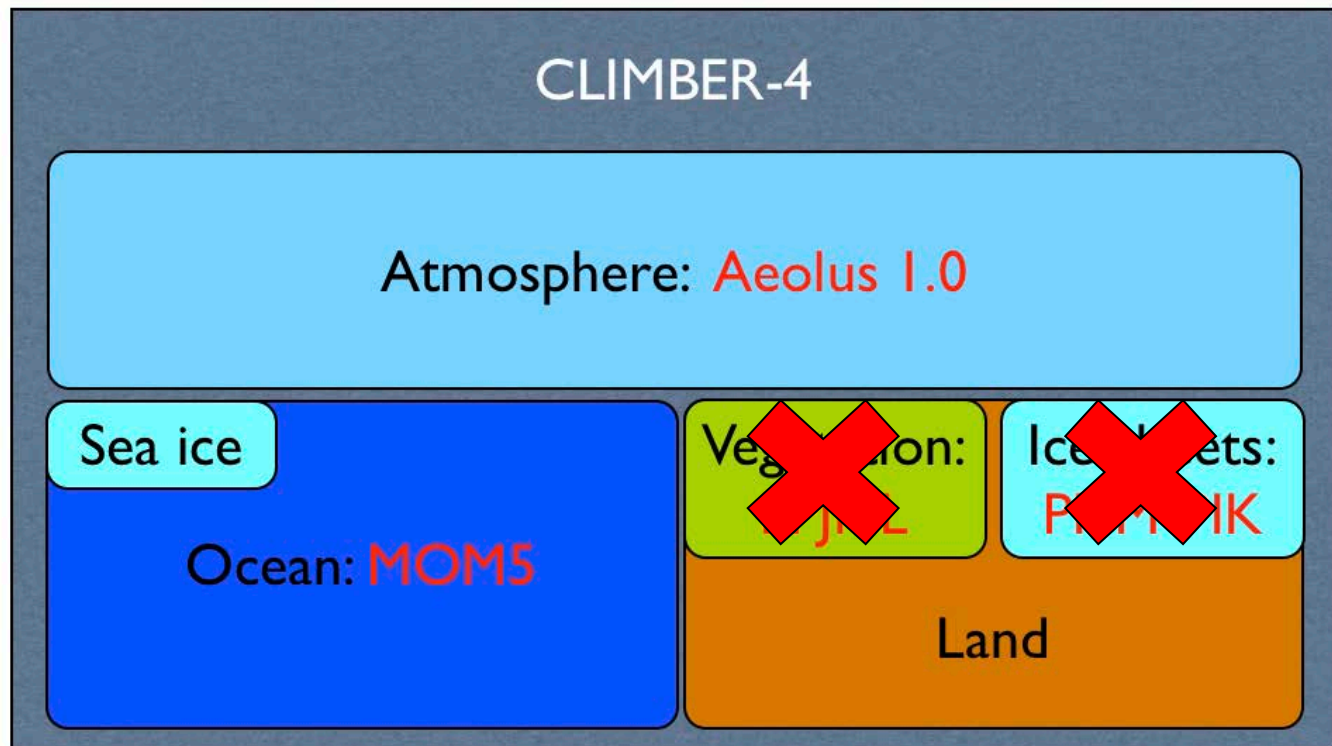


(Erik Peukert)

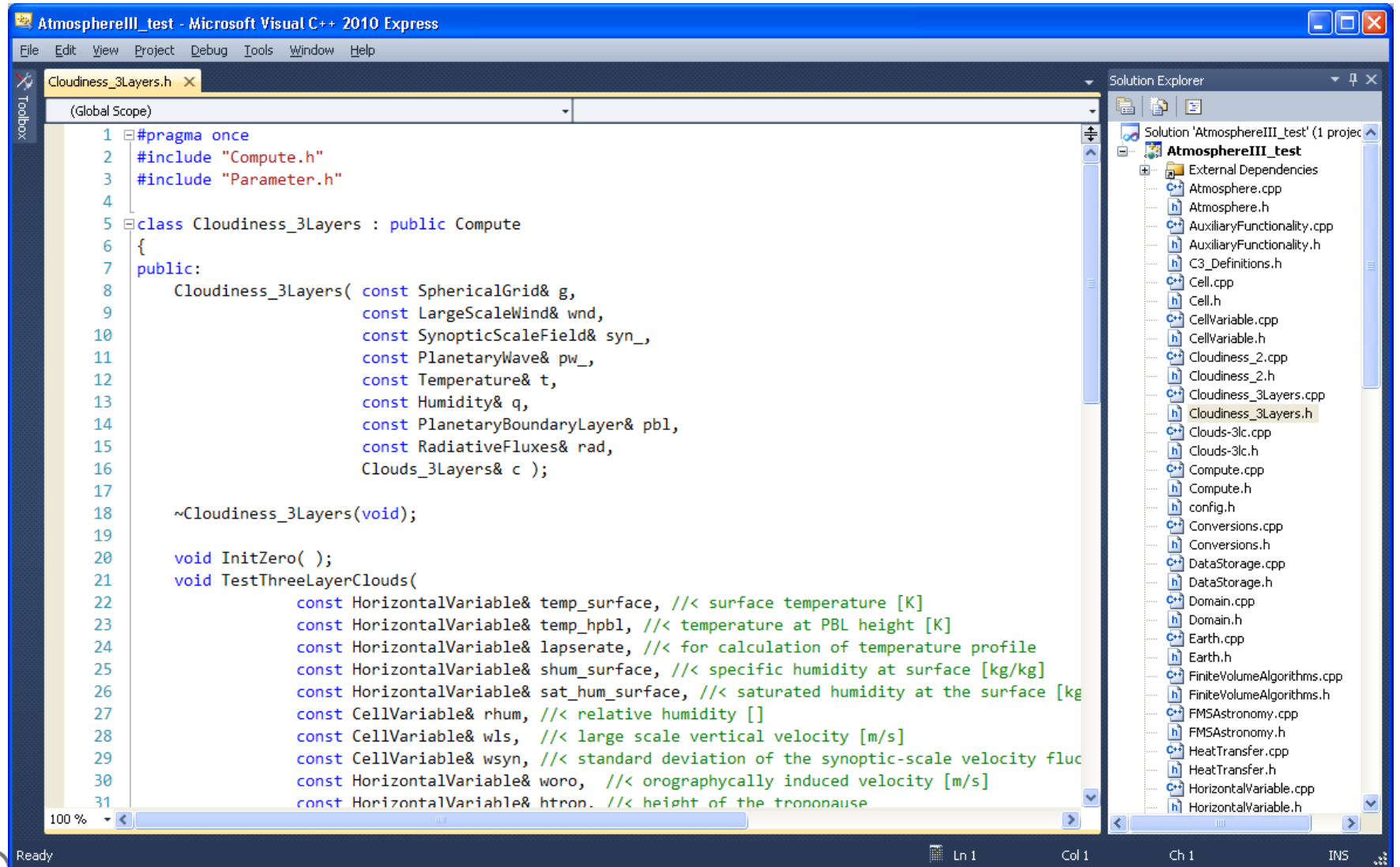
# Tuning: Fully-Coupled (*MOM5 - LAD - Aeolus*)

with Georg & Stefan:

- Technical coupling completed
- Latest version runs for ~50 years
- Model drift: small but non-zero cooling



# Tools: Visual C++ (IDE)



# Tools: Totalview (Debugger)

main\_Aeolus <@login0> Ubuntu

File Edit View Group Process Thread Action Point Debug Tools Window

Group (Control) Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live

Process 1 (20761): main\_Aeolus (At Breakpoint 3)

Thread 1 (47860583807264) (At Breakpoint 3)

Stack Trace

C++ main, FP=7fff4660d1c0  
\_libc\_start\_main, FP=7fff4660d270  
\_start, FP=7fff4660d280

Stack Frame

Function "main":  
argc: 0x00000001 (1)  
argv: 0x7fff4660d298 -> 0x7fff4660f10d  
Local variables:  
tag: (std::string)  
do\_aquaplanet: false (0)  
main\_Aeolus\_nml: 0x009a70c0 -> (NML)  
use\_3layer\_clouds: (class Parameter<bool>)  
do\_dynamics: (class Parameter<bool>)  
do\_physics: (class Parameter<bool>)

Function main in main\_Aeolus.cpp

```
187
188 #ifdef USE_MPI_DOMAIN_DECOMPOSE
189     GB(MPI::COMM_WORLD);
190     printf("P%d has %u meridionalcells\n", MPI::COMM_WORLD.Get_rank(), grid.MeridionalCe
191     cout.flush();
192     GB(MPI::COMM_WORLD);
193 #endif
194
195 // start computations for each month
196 // month: 0 = Jan, 1 = Feb, 2 = Apr, 3 = Jun, 4 = Jul, 5 = Aug, 6 = Oct, 7 = Dec (for ER
197 // for (int month = 0; month < 12; month++) // DC changed N_months -> 12 !
198 // for (int loop = 0; loop < 4; loop++) // march only
199 //
200     int loop = 0; // start with DJF data
201     int season = loop;
202     cout<<"*** Starting calculations for season ("<<season<<) *** "<< endl;
203     int count_out = 0;
204
205 // load data from file and init the model
206 if(do_aquaplanet)
207 {
208     AquaplanetTopography( topo );
209     InitZonalMeanAquaPlanet( grid, PBL, PLANWAVE, SYNOPTIC, WIND, ENERGY_CYCLE, VAPO
210 }
211 else
212 {
213     // first load ERA-40 data for later reference
214     //LoadInputVariables( Model.without_topo(), month, grid, atm, topo, wind, stand
215     PBL.LoadFromFile( season );
216     PBL.AdaptPoleToEquatorGradient( dTdy_factor(), Ta_factor() );
217
218     VAPOR_CYCLE.LoadFromFile( season );
219     SURFACE_FLUX.LoadFromFile( season, Model.without_topo() );
220
221 // now initialize the Aeolus
222 InitPresentDayClimate ( grid, pbl, TS (), humidity.qS (), PBL, PLANWAVE, SYNOPTIC
223 cout<<"*** Era-40 input data loaded and model initialized.... *** "<< endl;
224
225 cout<<"*** Start timestepping *** "<< endl;
226 double start_day; // approximate mid-month day (360 day year fo
227 if (season == 0) start_day = 15.; // DJF
228 else if(season == 1) start_day = 105.; // MAM
229 else if(season == 2) start_day = 195.; // JJA
```

Action Points Processes Threads

STOP 2 main\_Aeolus.cpp#179 main+0x16db  
STOP 3 main\_Aeolus.cpp#225 main+0x1ec4  
STOP 1 main\_Aeolus.cpp#252 main+0x220d

Array Statistics <@login02>

File Edit Window

((class Variable &)pbl.TS).data\_M\_data  
(at 0x02357480) Type: double[3588]  
Slice: [:]  
Filter:

Count: 3588  
Zero Count: 0  
Sum: 999670.490768433  
Minimum: 239.526408672333  
Maximum: 299.261733436585  
Median: 283.261411857605  
Mean: 278.614963982283  
Standard Deviation: 17.9981291769422  
First Quartile: 262.953419137001  
Third Quartile: 296.372978019714  
Lower Adjacent: 239.526408672333  
Upper Adjacent: 299.261733436585

NaN Count: 0  
Infinity Count: 0  
Denormalized Count: 0

Checksum: 46746

Update

File Window

