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

Leibniz Gemeinschaft

Content

- Intro *Aeolus* and *Climber-4*
- Key Modules
- Object-oriented Code Design

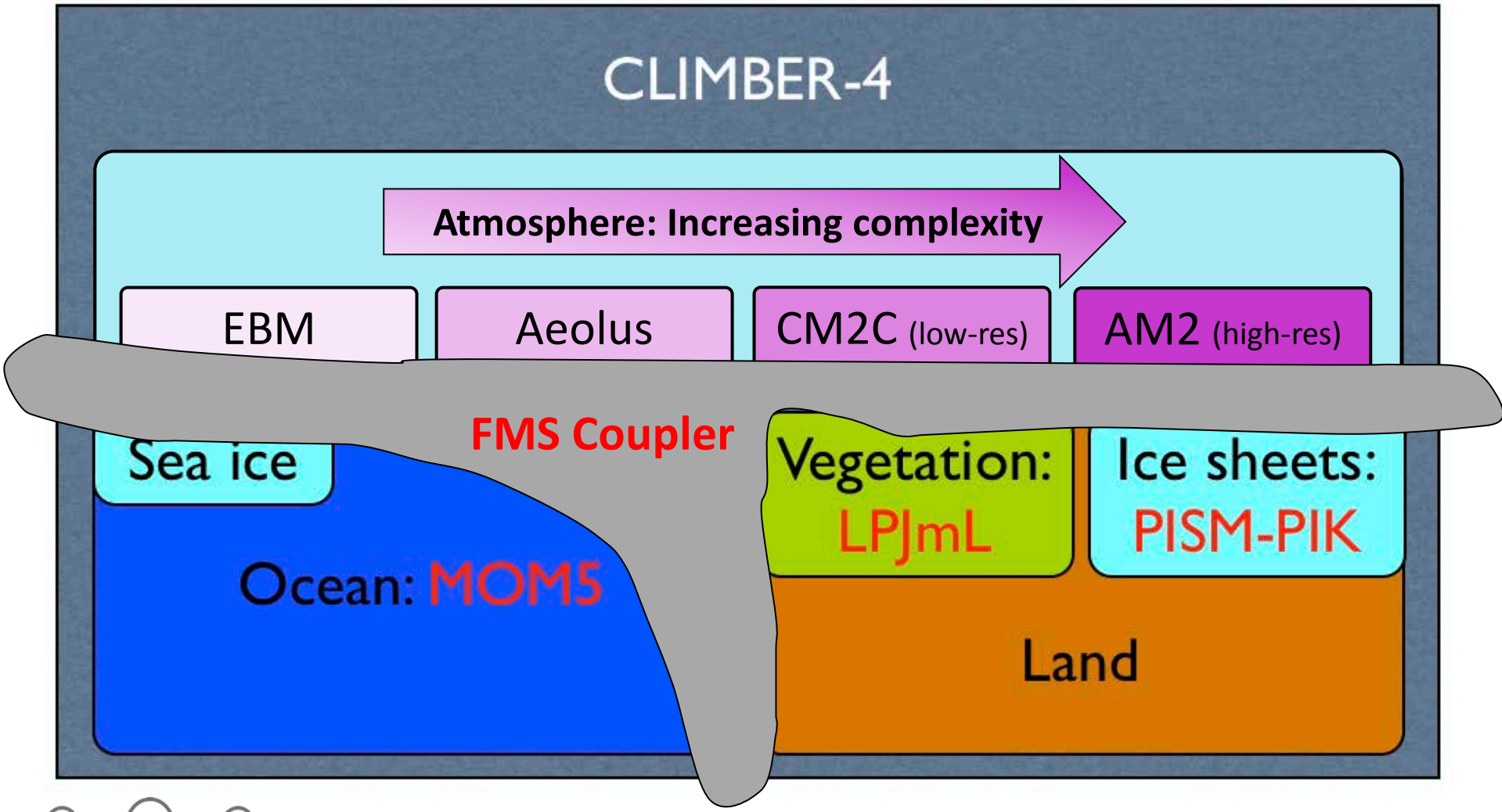
  Strict separation of *Data Storage*, *Gridding* and *Computations*
- Safety Mechanisms:

  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*)

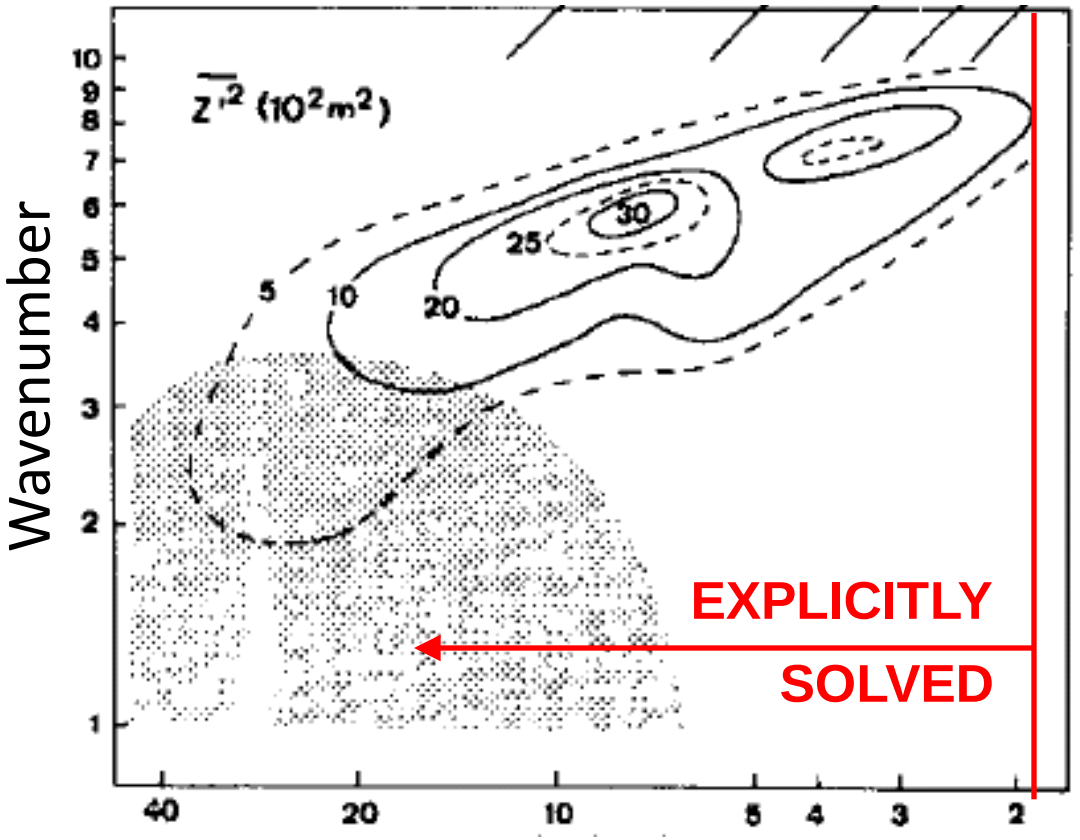
Hierarchy of Atmosphere Models



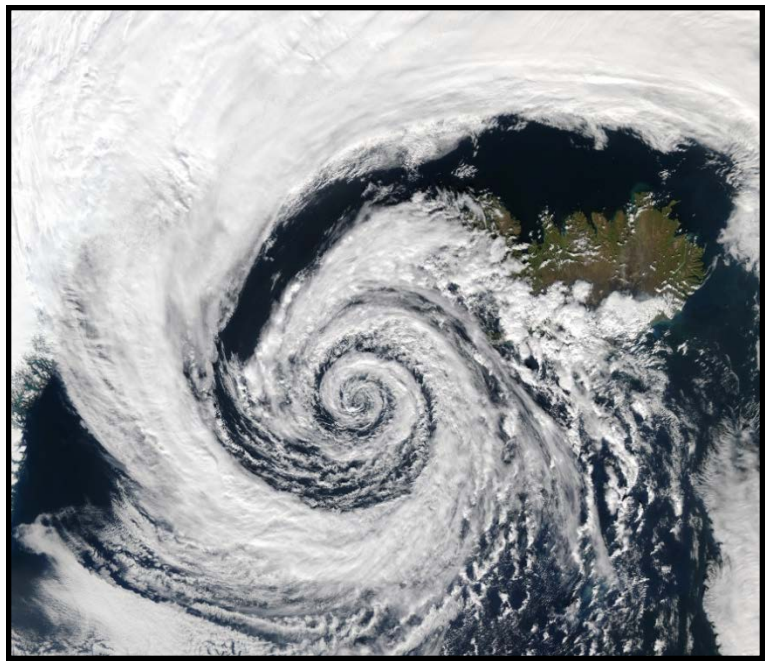
Synoptic Scale Eddies

GCMs

Atmospheric Power spectrum



~30°, ~ 3000km, k > 7



HIGH RES in Space & Time

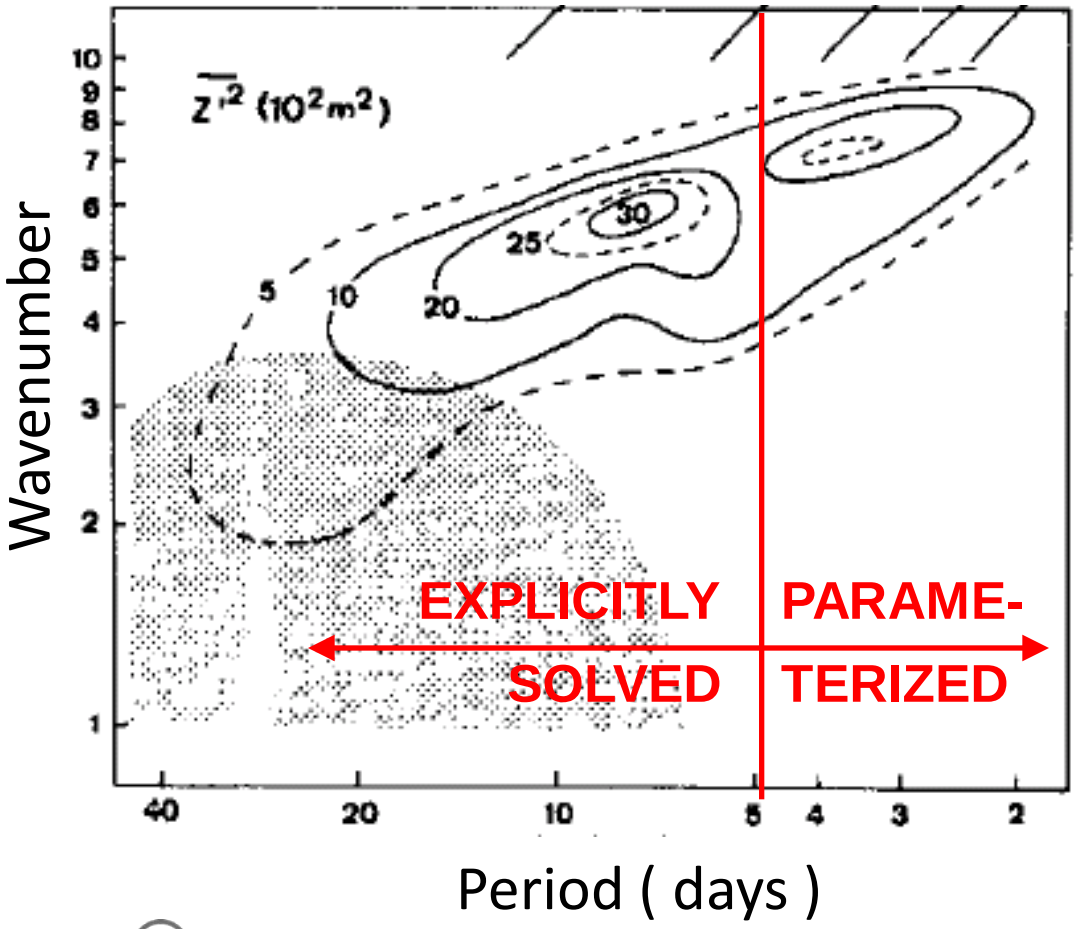
PIK

NB: Interested in Climate rather than Weather !

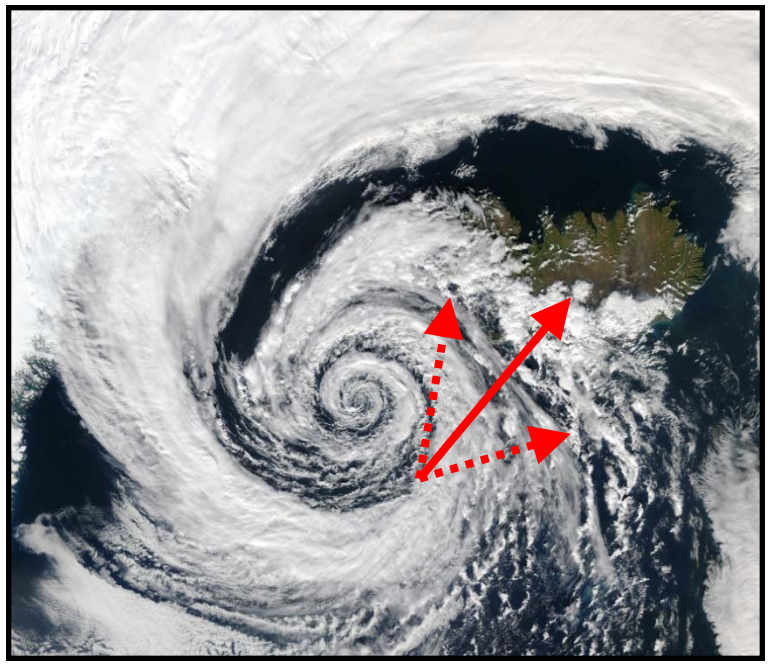
Synoptic Scale Eddies

SDAMs

Atmospheric Power spectrum



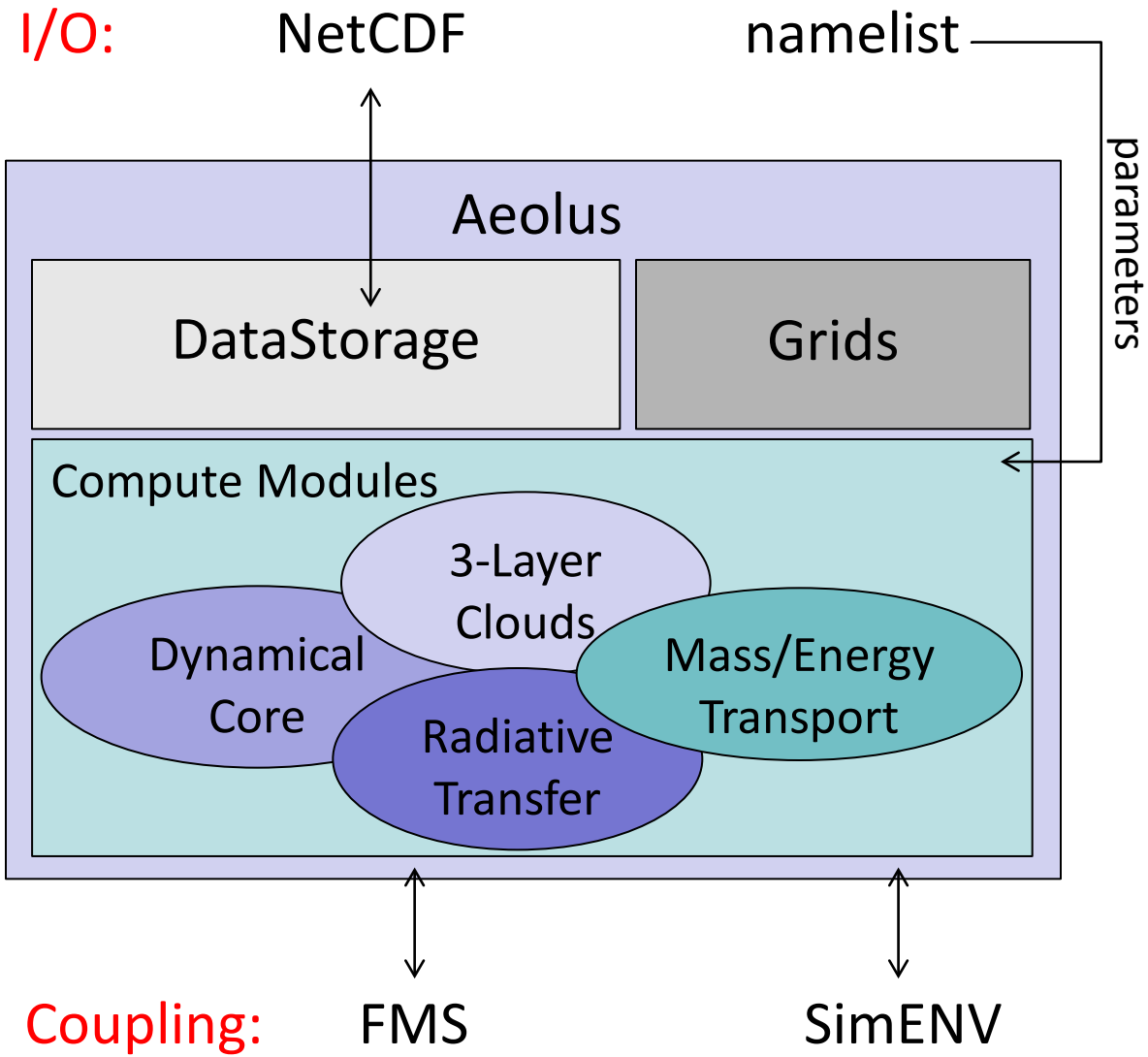
~30º, ~ 3000km, k ~ 10



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

Aeolus

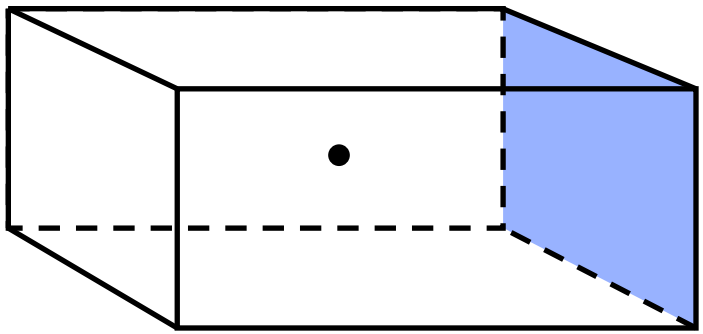
- Object-oriented *C++* implementation
- ~45k code lines
- *svn* version control
- *MPI* Parallel Computing



Grid: Core Building Blocks

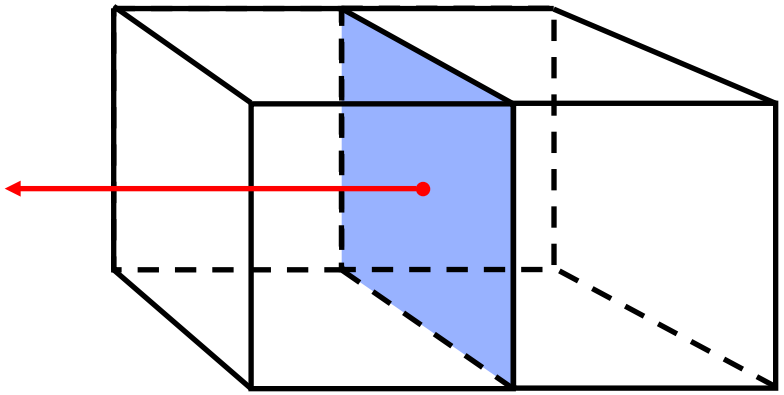
Cell

- 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

Standard Spherical Grid

SphericalGrid

- vect
- vect
- …
- Func

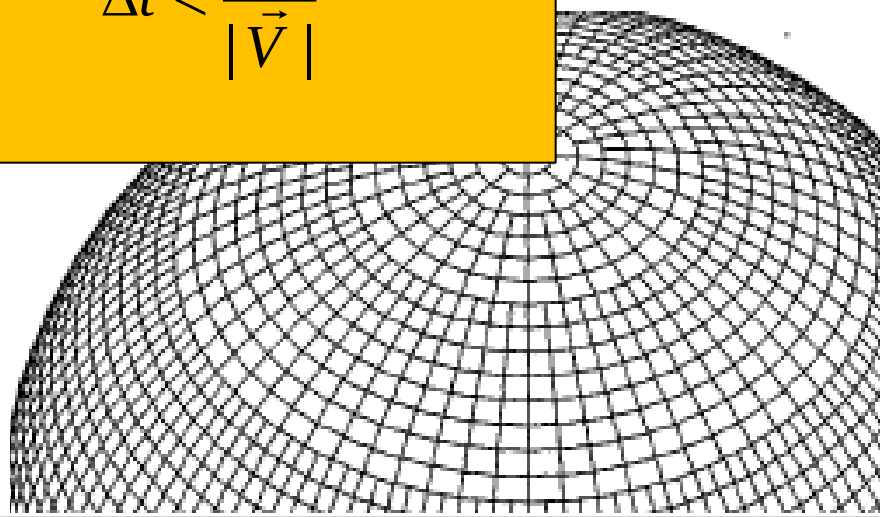
fast
spec
and

Numerical Stability

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

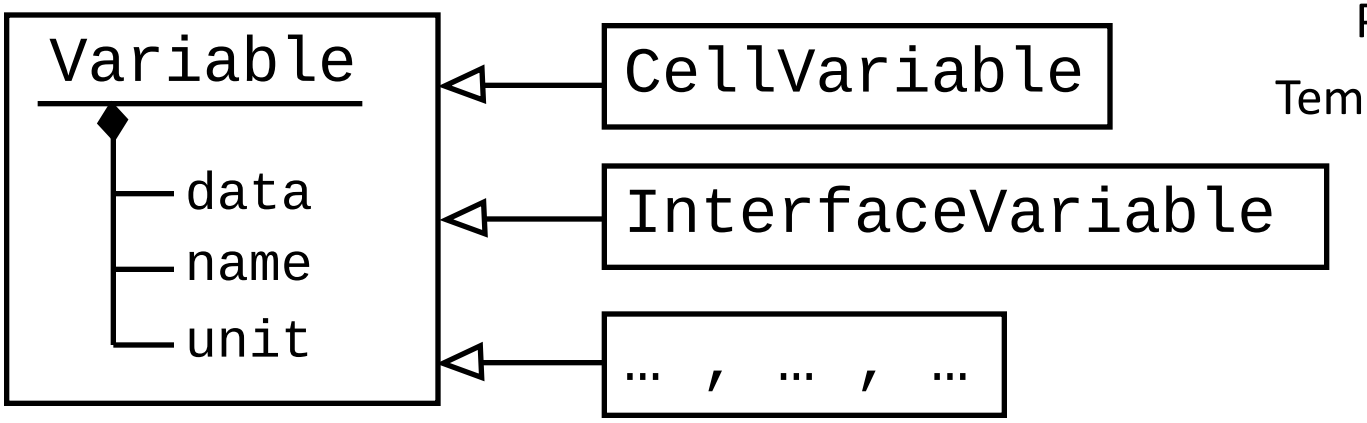
Finite Volume: $\Delta t < \frac{\Delta x}{|\vec{V}|}$

rical Grid



´Internally consistent, self-described grid´

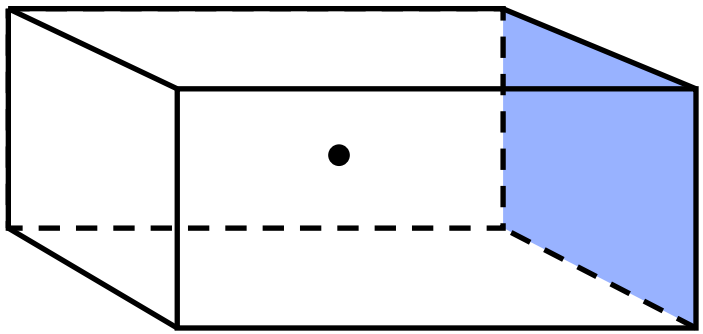
Data Storage

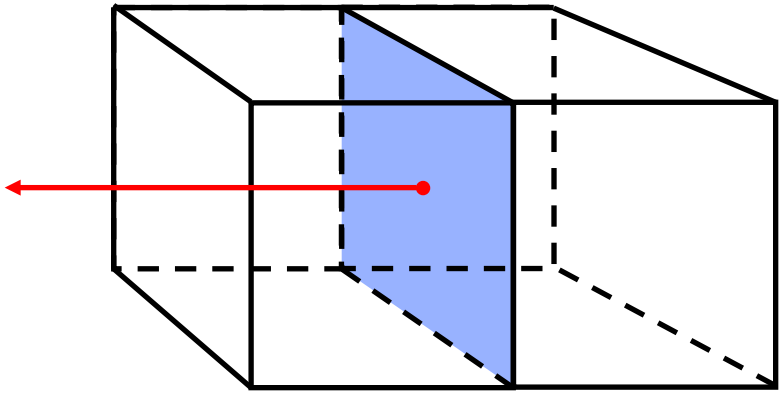


Primary variables:
Temperature, Humidity,…

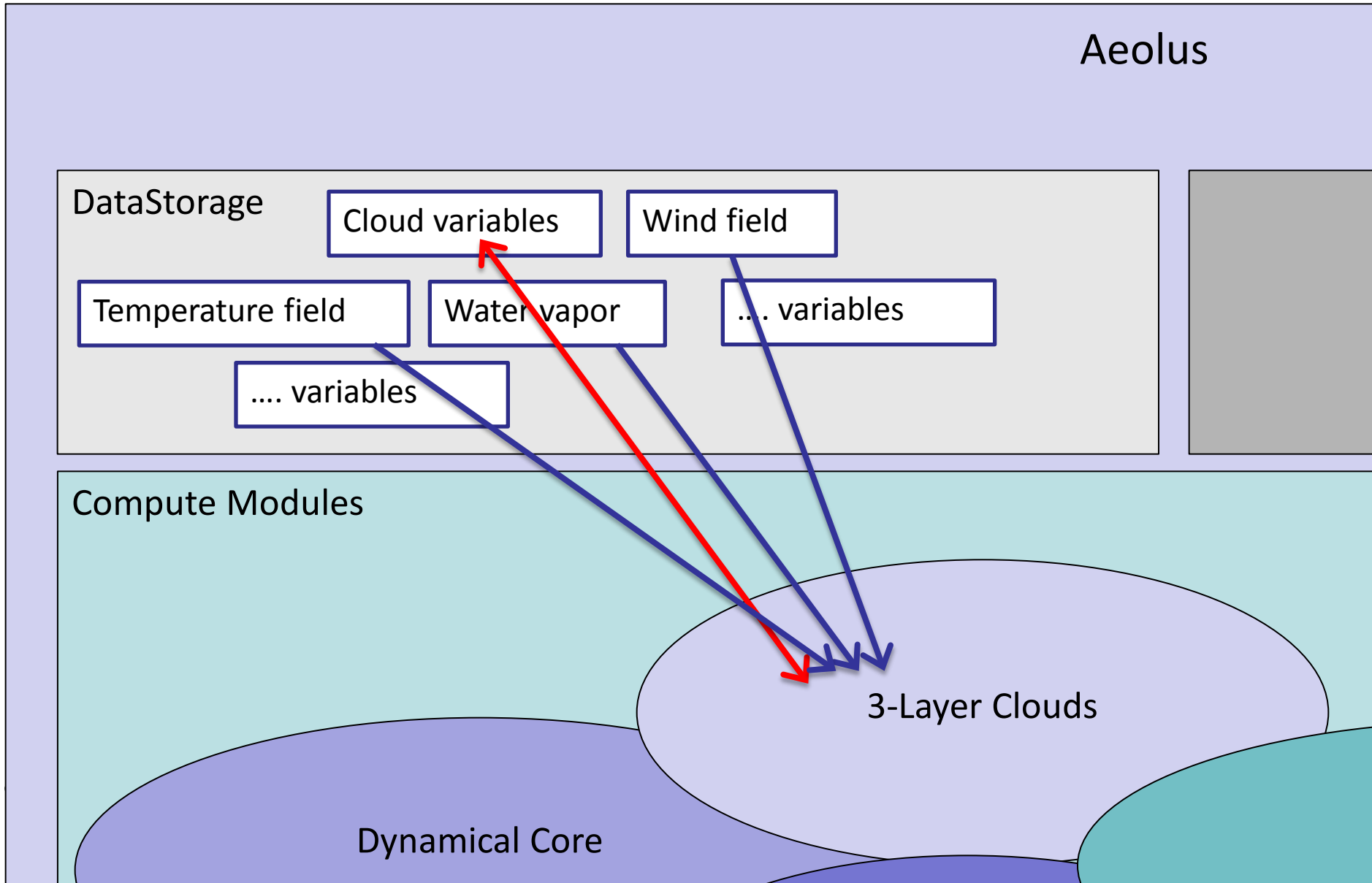
Fluxes, velocities, etc

Specific sub-regions





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

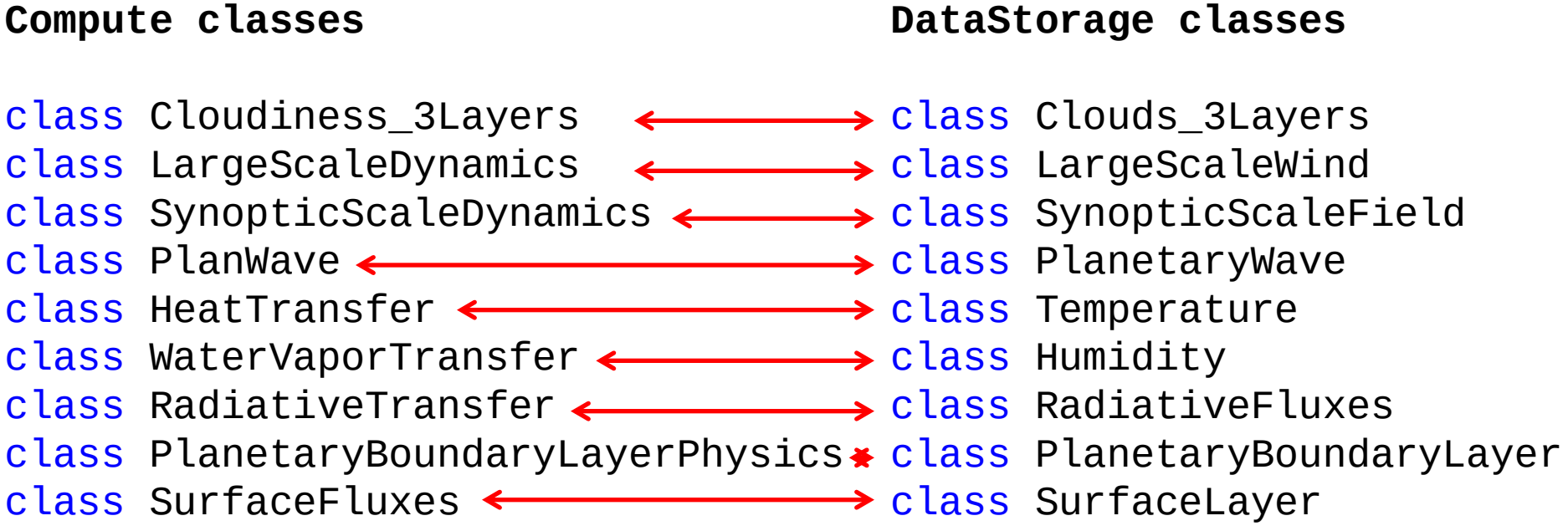
```
class Clouds
{
public:
  Clouds(const SphericalGrid&);
  ~Clouds(void);

  // 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:



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:

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

NaN check, range check

…Black box…

→ 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*)

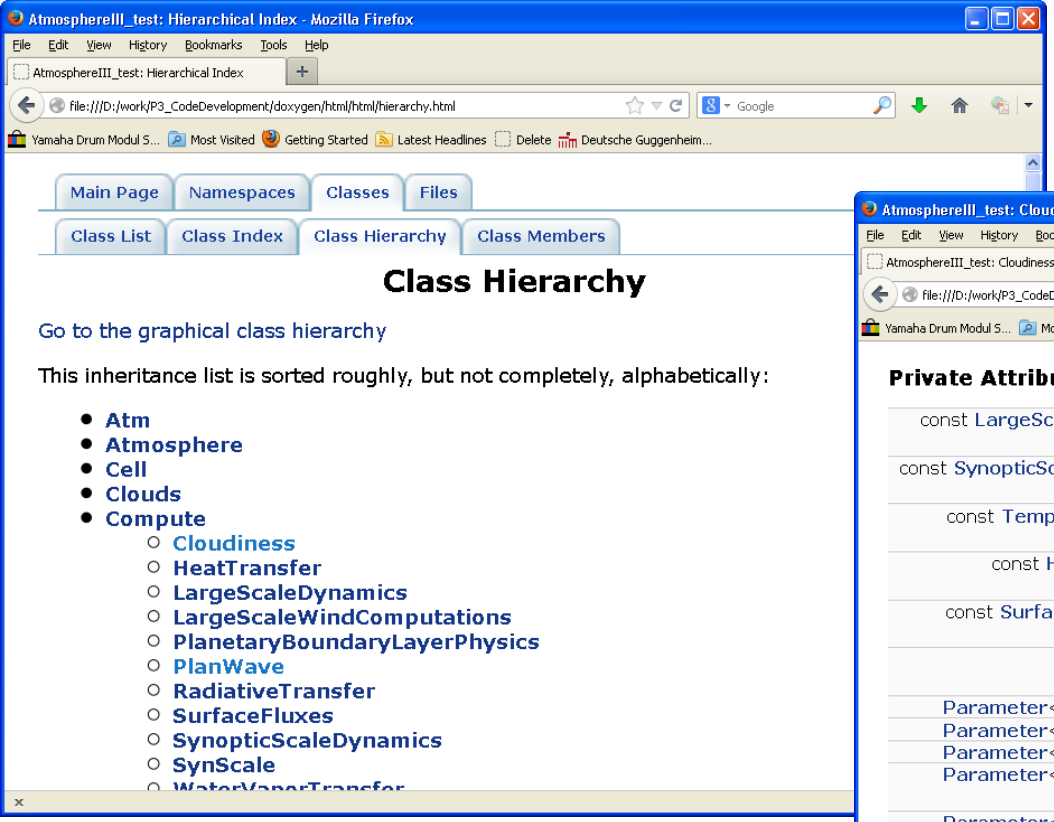
Documentation

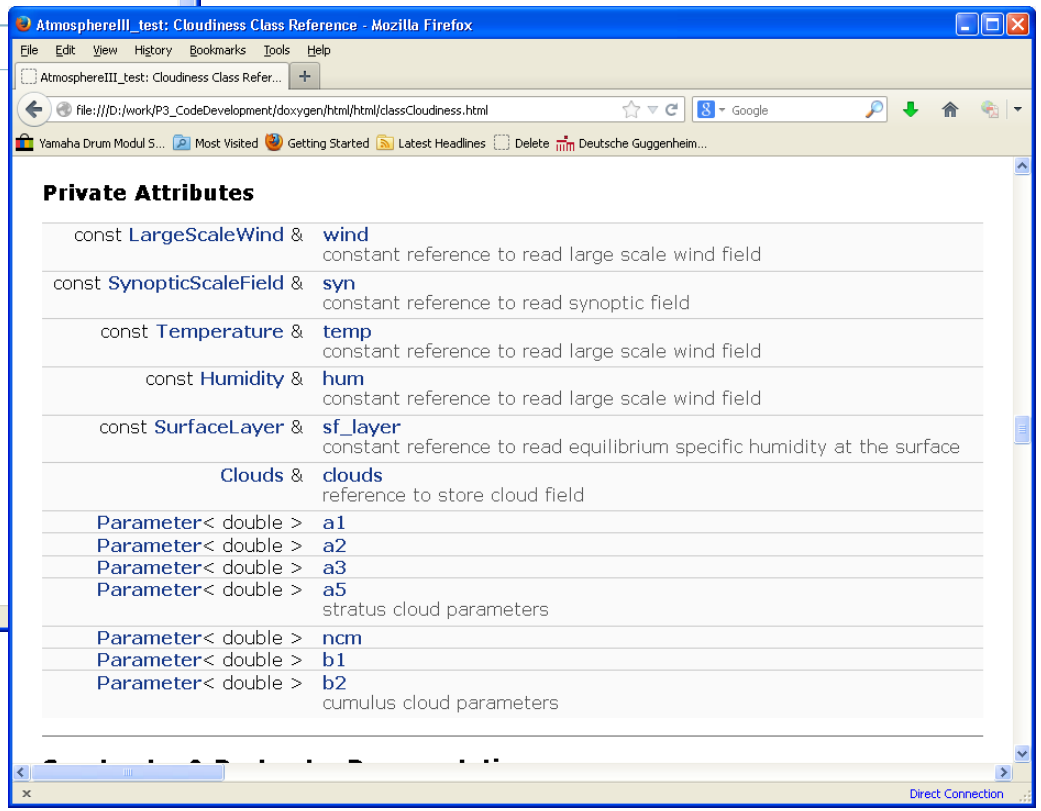
- **Peer reviewed**
  - Synoptic Dynamics (Coumou et al, 2011)
  - 3-Layer Clouds (Eliseev et al, 2013)
  - Dynamical Core (in progress…)
- **Technical reports (Mathematical Model Description)**

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

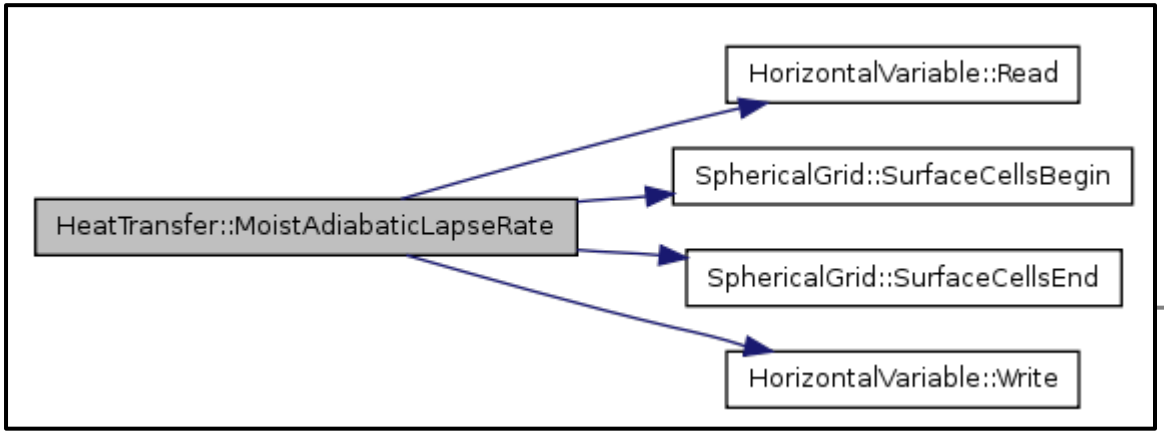
  Generates html/LaTeX documentation directly from source-code

Doxygen

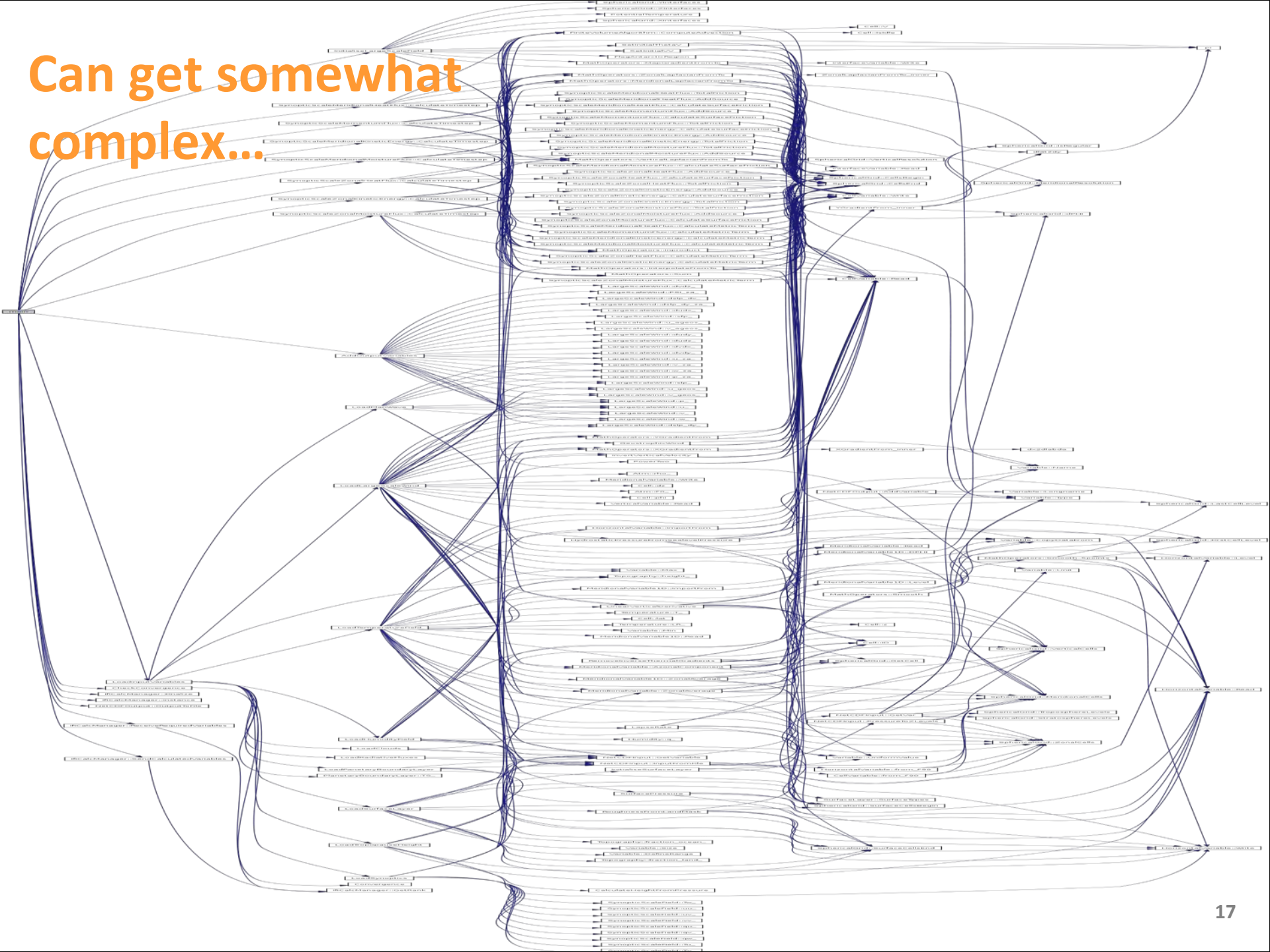




Call-graphs / Dependency-graphs



Can get somewhat complex…



Content

- Intro *Aeolus* and *Climber-4*
- Key Modules
- Object-oriented Code Design

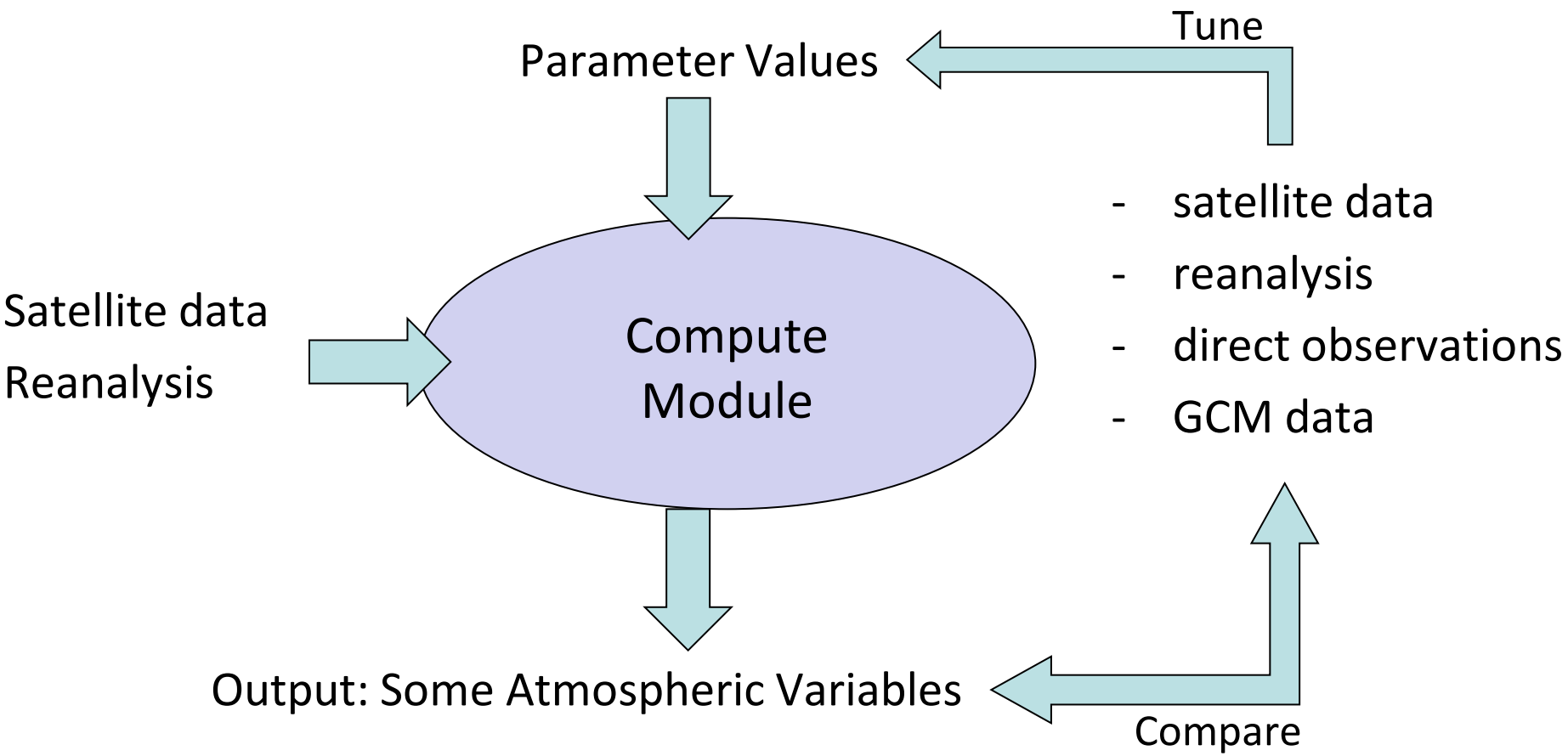
  Strict separation of *Data Storage*, *Gridding* and *Computations*
- Safety Mechanisms:

  Explicit *Read-Write* Functionality, *Design-by-Contract*, Automatic *Range Checking*
- Documentation
- **Testing / Benchmarking**

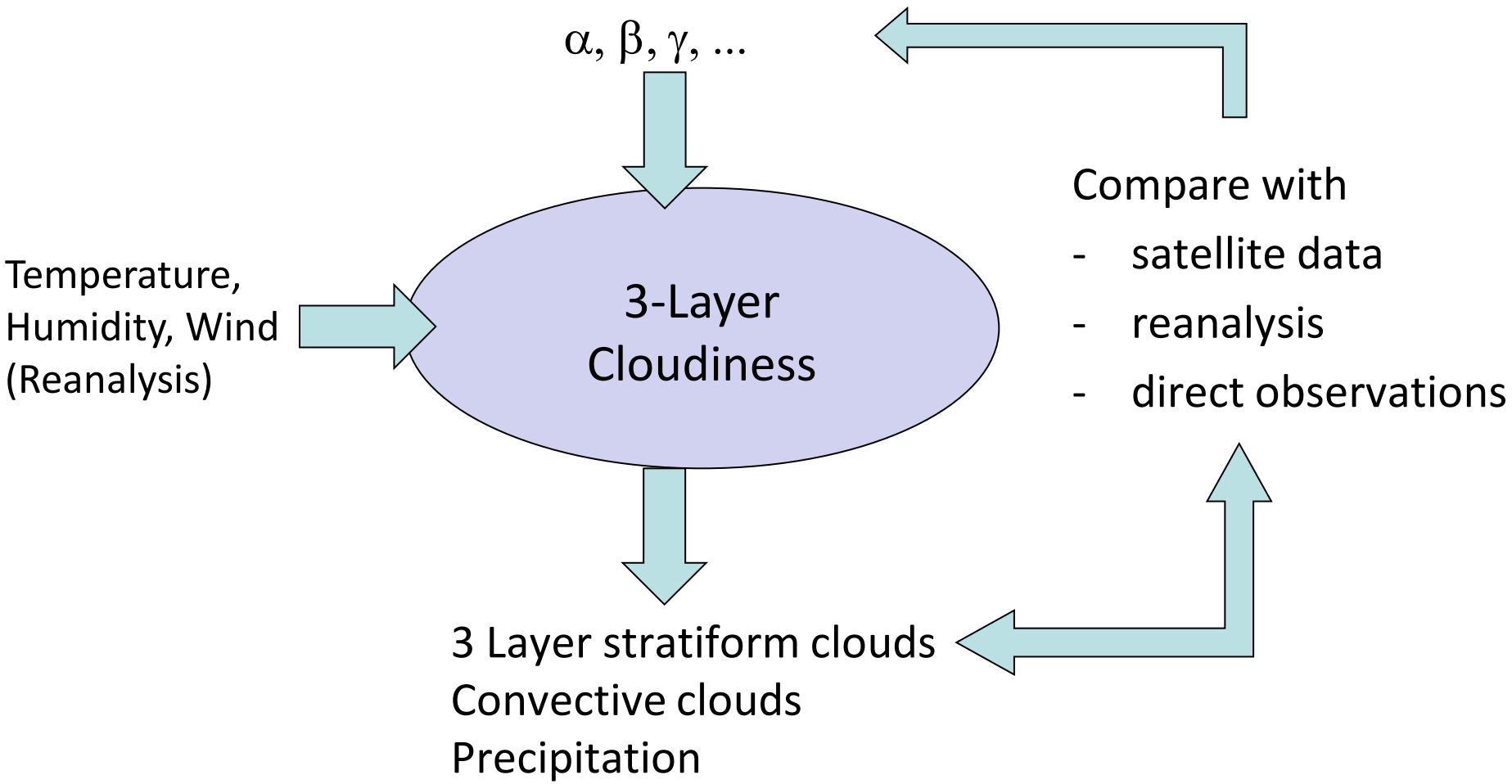
  Module-wise, *Stand-alone*, *Fully-coupled*
- Handy Tools:

  Totalview, *valgrind*, *VC++* (or in general *IDEs*)

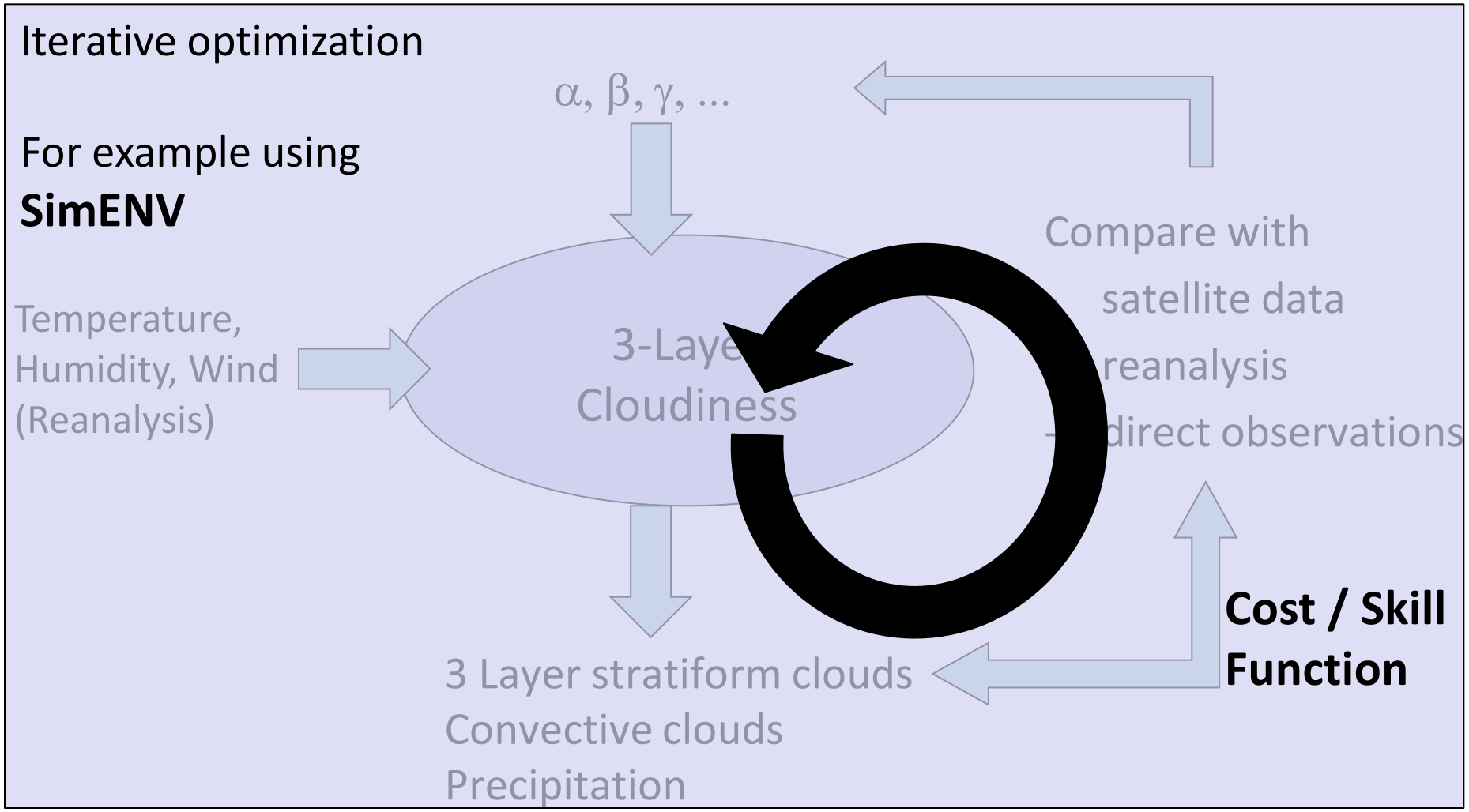
Tuning: Module-wise



Tuning: Example Cloudiness

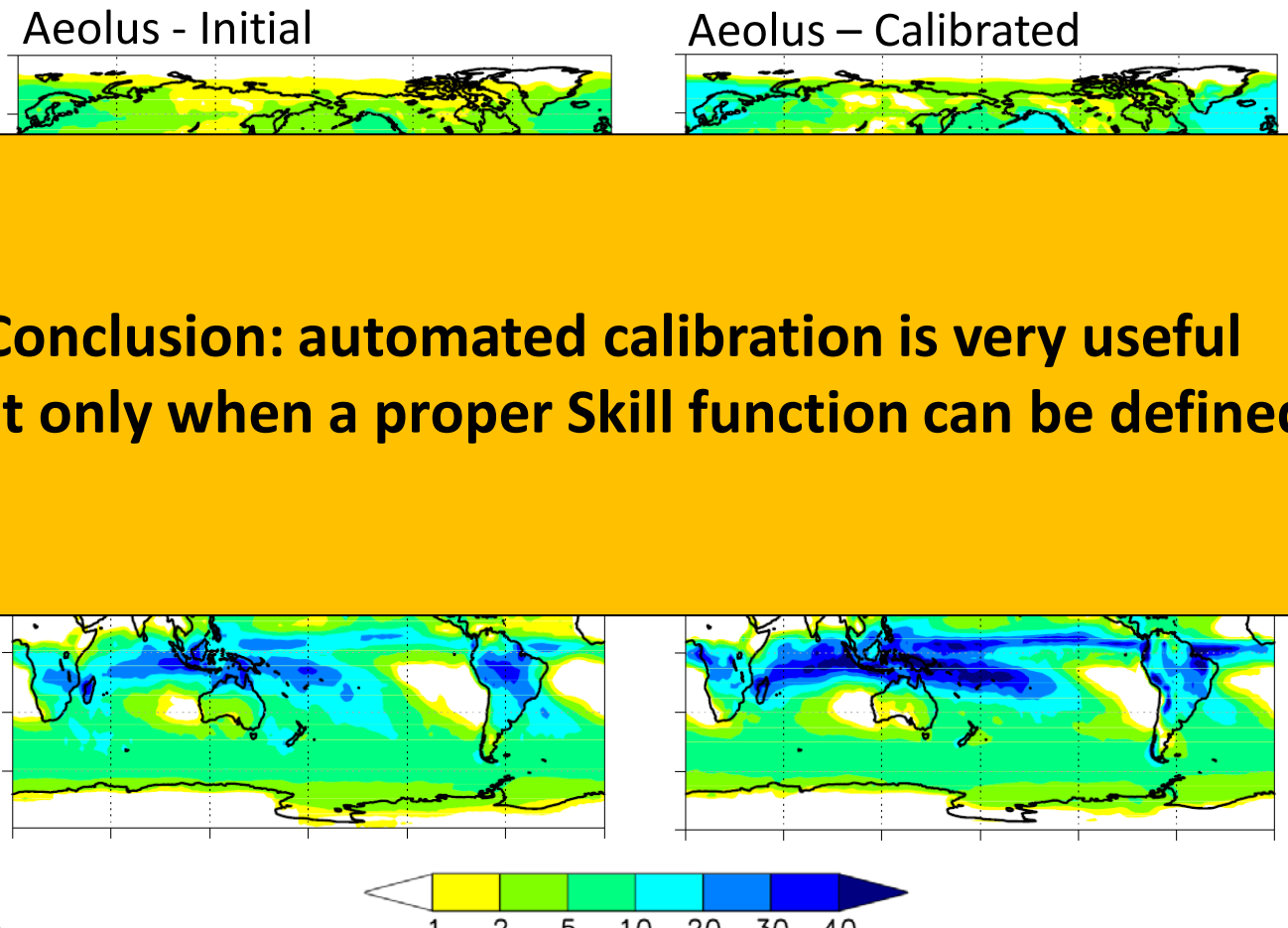


Tuning: Example Cloudiness



Tuning: Module Calibration

Improves low precipitation in major desert regions

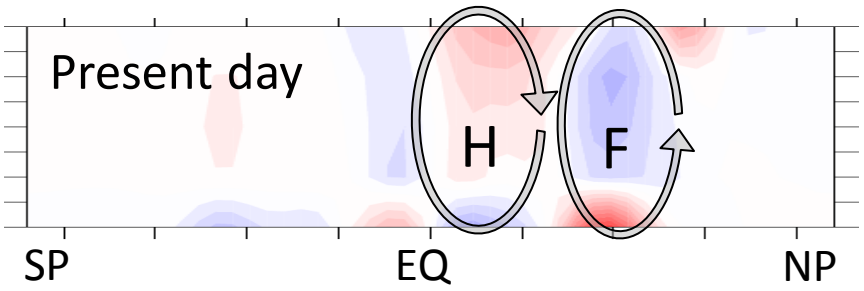


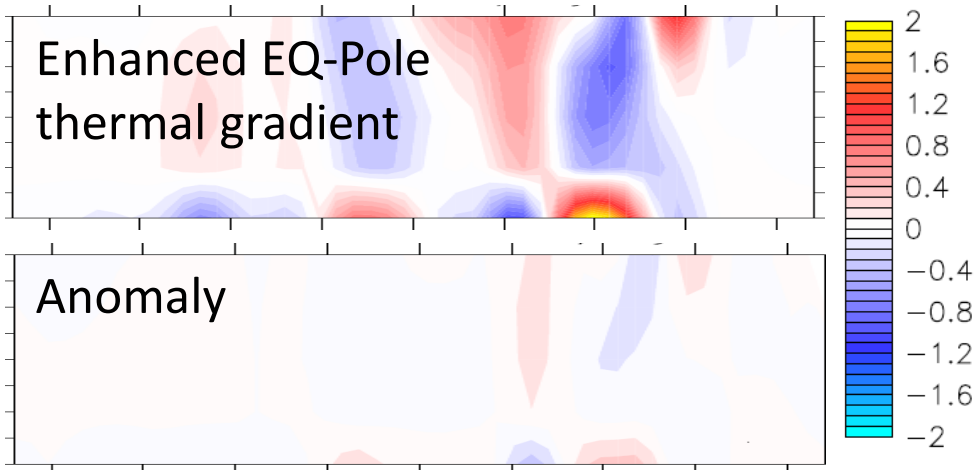
Conclusion: automated calibration is very useful
…but only when a proper Skill function can be defined!

Eliseev et al, 2013

Tuning: Stand-alone *Aeolus*

<u>Next Step:</u> Sensitivity-analysis using SimENV





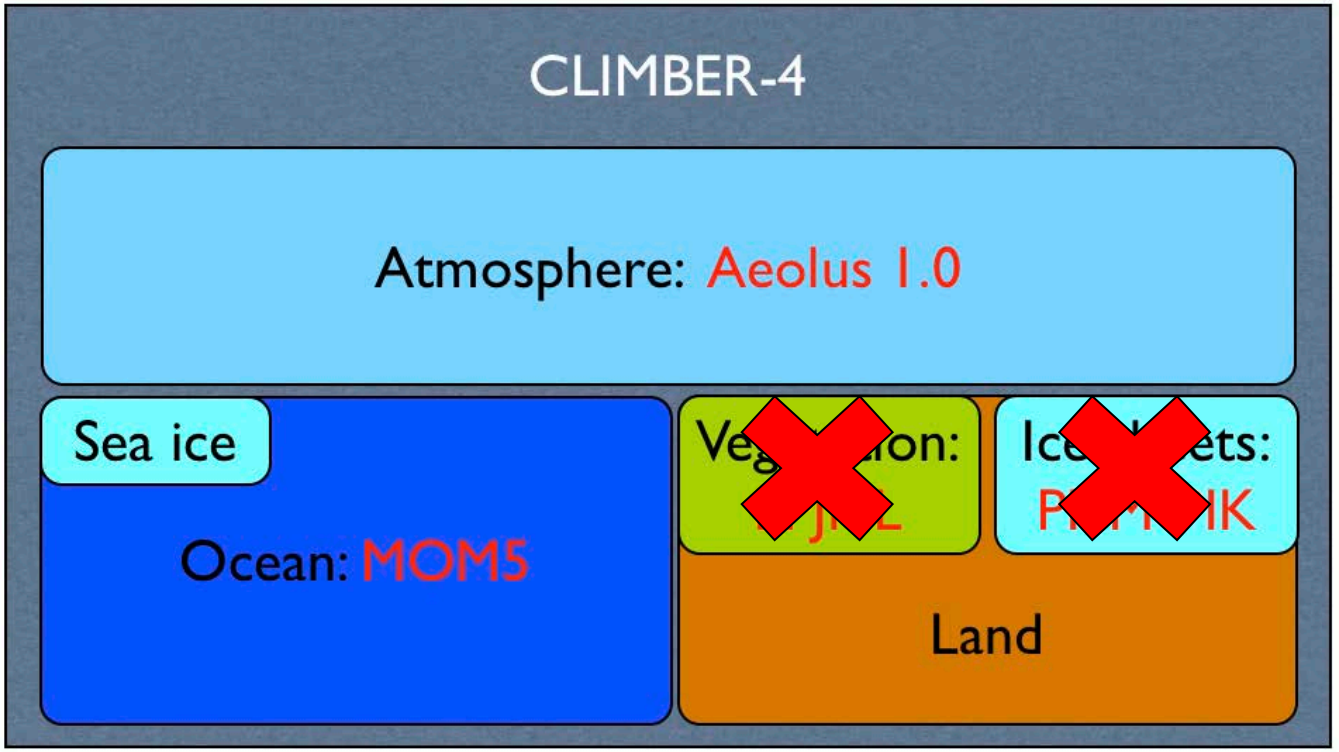
Hadley & Ferrel Cell strengthen and widen for stronger T-gradients

(Erik Peukert)

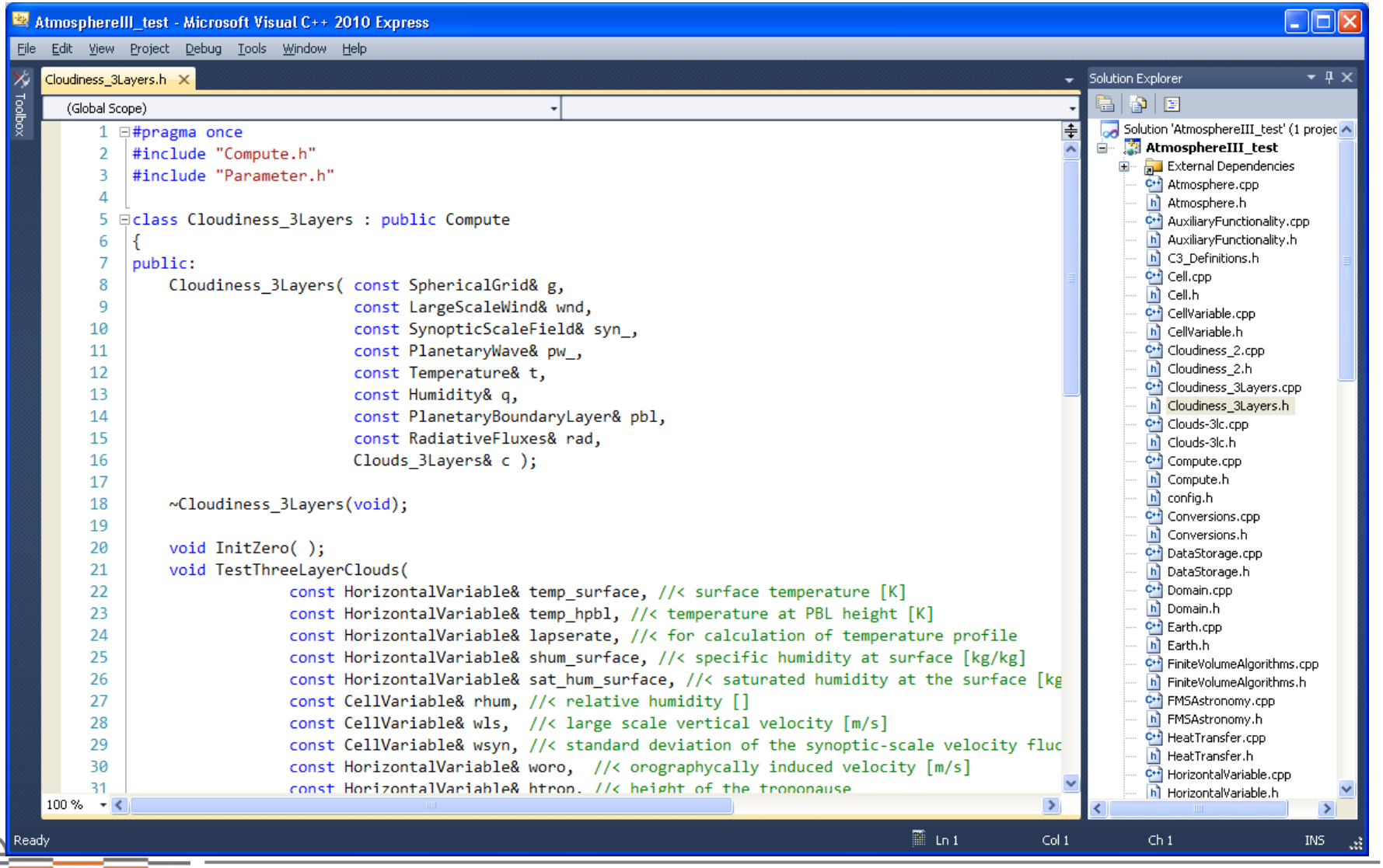
Tuning: Fully-Coupled (*MOM5 - LAD - Aeolus*)

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)