



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

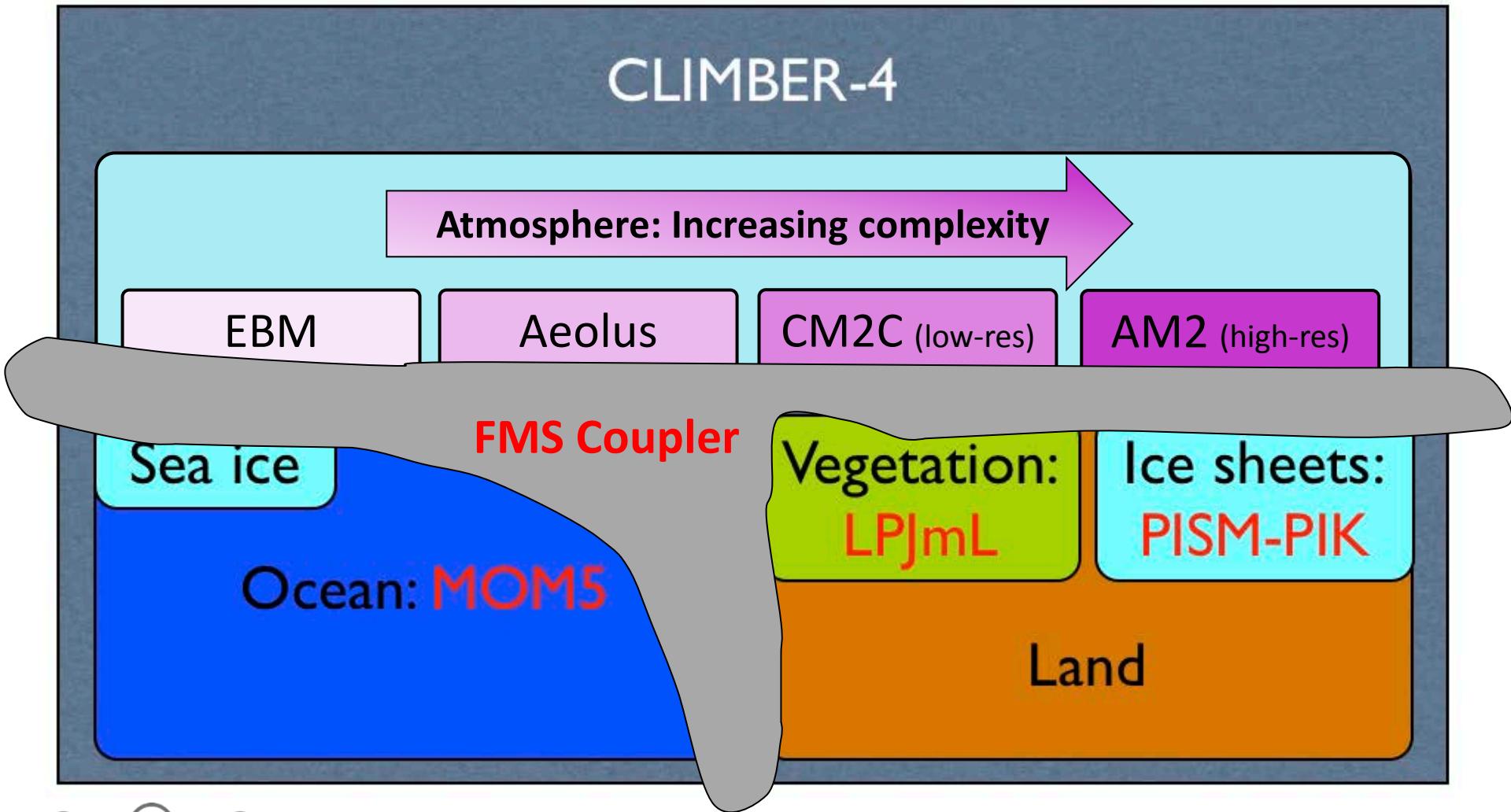
ModStrat Seminar 14 Nov. 2013

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)*



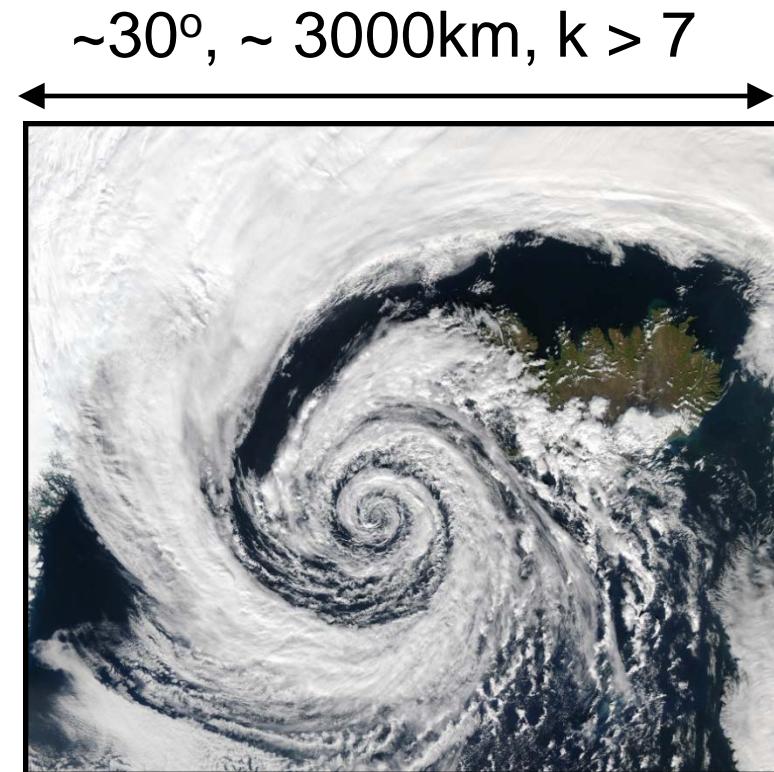
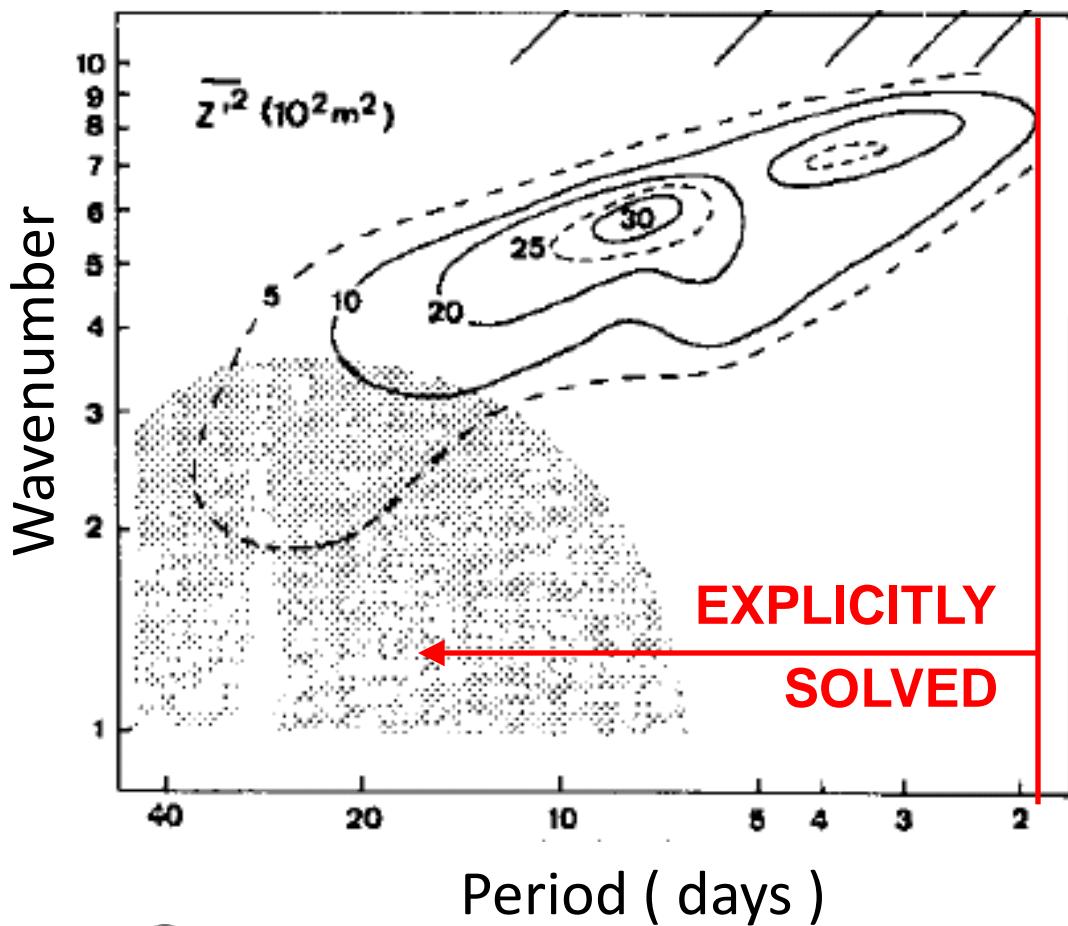
Hierarchy of Atmosphere Models



Synoptic Scale Eddies

GCMs

Atmospheric Power spectrum



HIGH RES in Space & Time

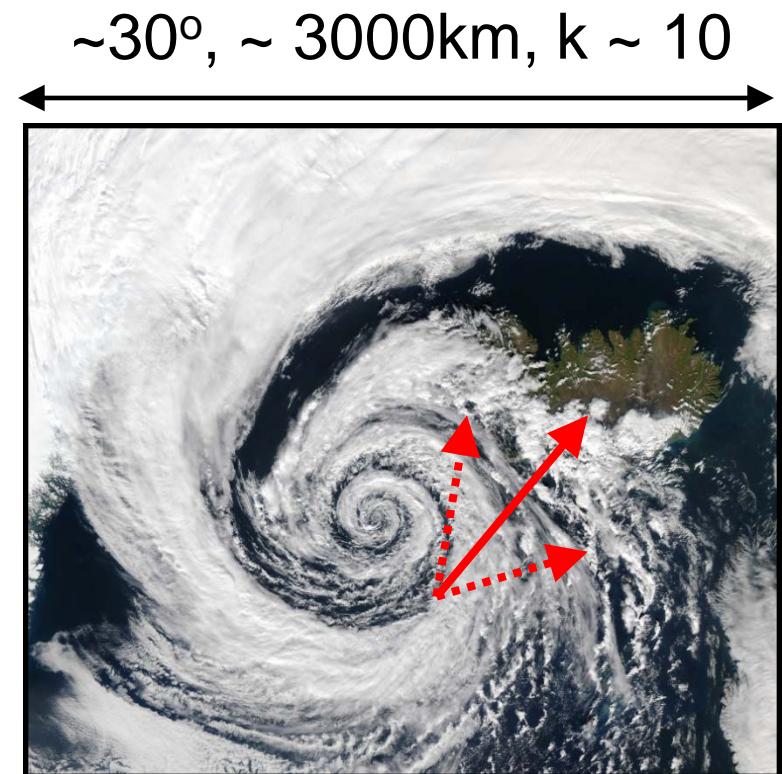
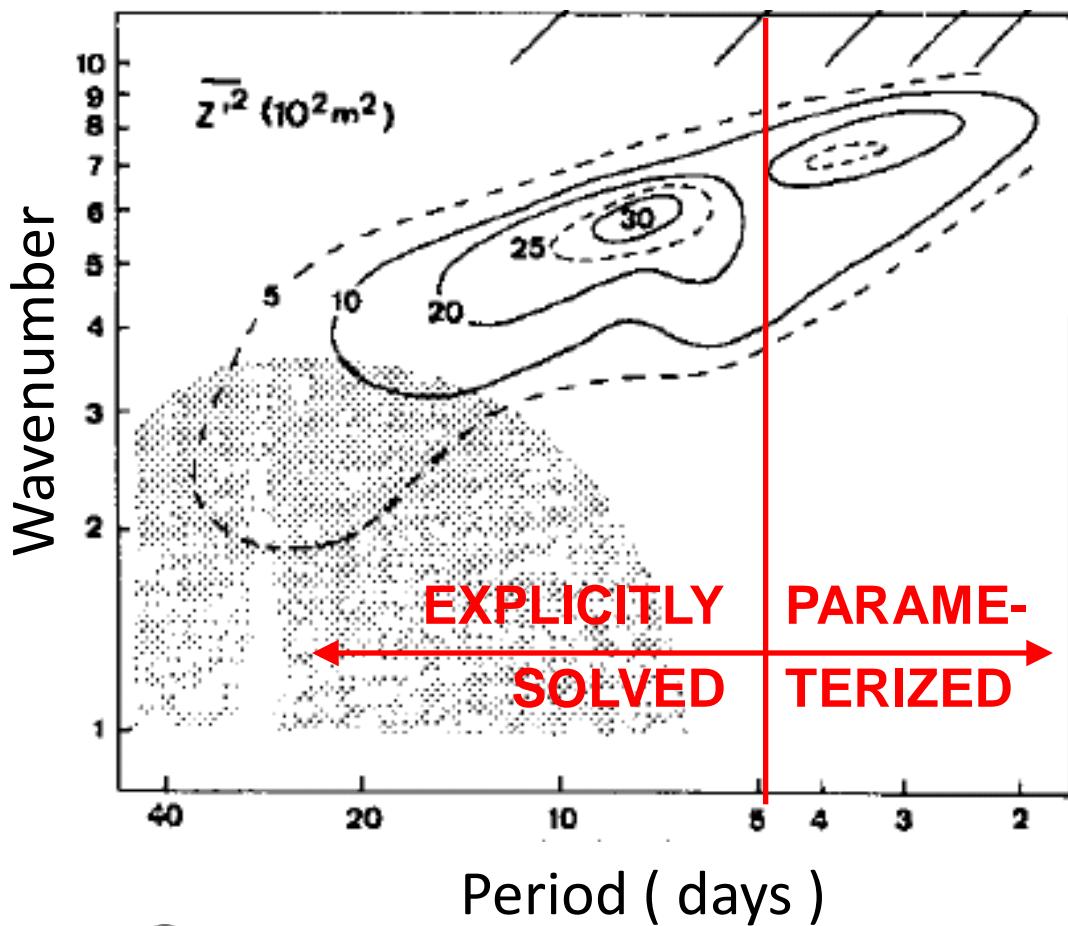


NB: Interested in Climate rather than Weather !

Synoptic Scale Eddies

SDAMs

Atmospheric Power spectrum

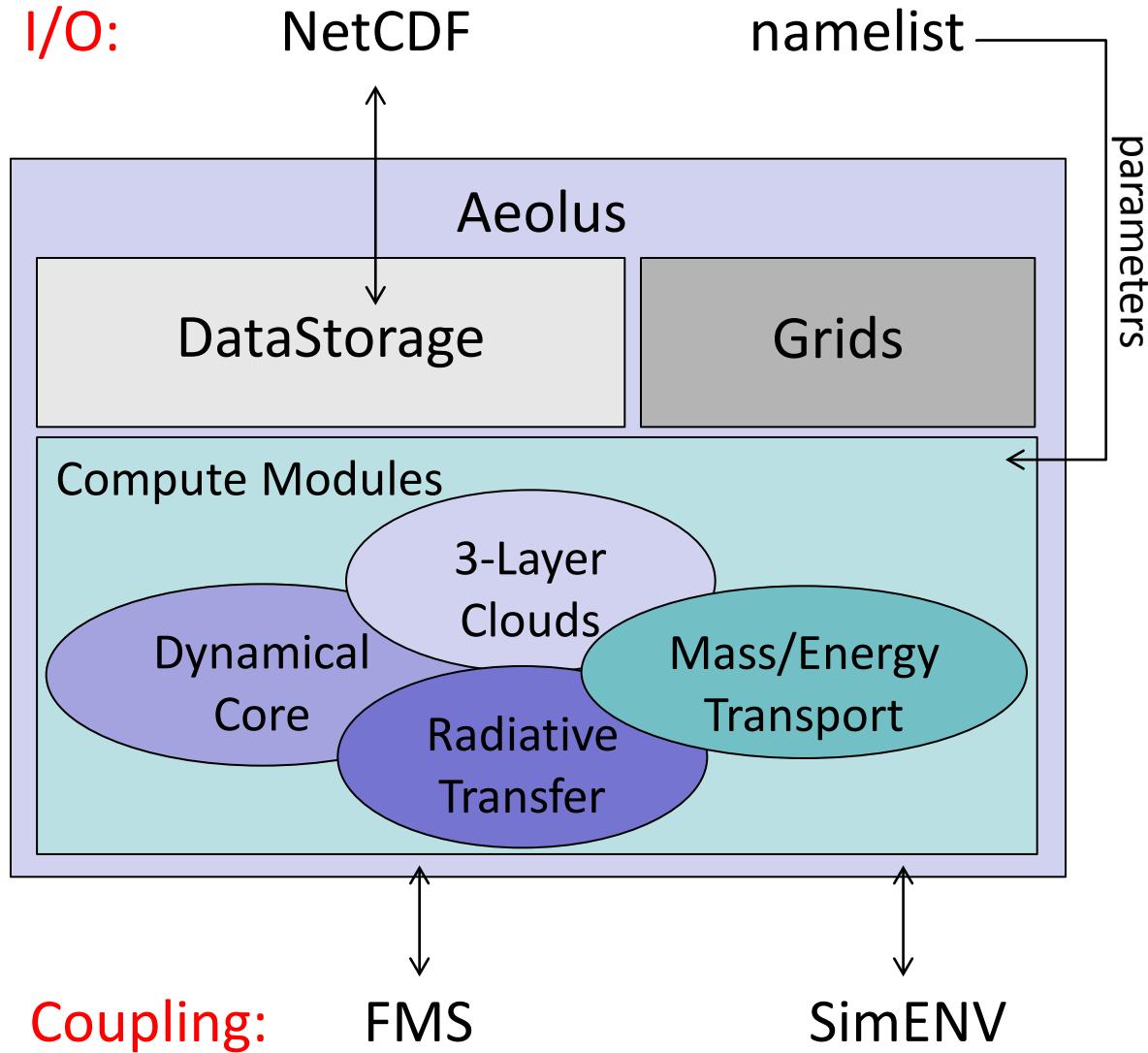


$$\vec{V} = \langle \vec{V} \rangle + \vec{V}'$$



Aeolus

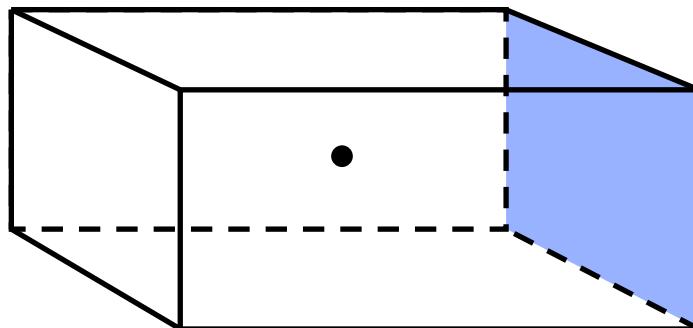
- Object-oriented *C++* implementation
- ~45k code lines
- *svn* version control
- *MPI* Parallel Computing



Grid: Core Building Blocks

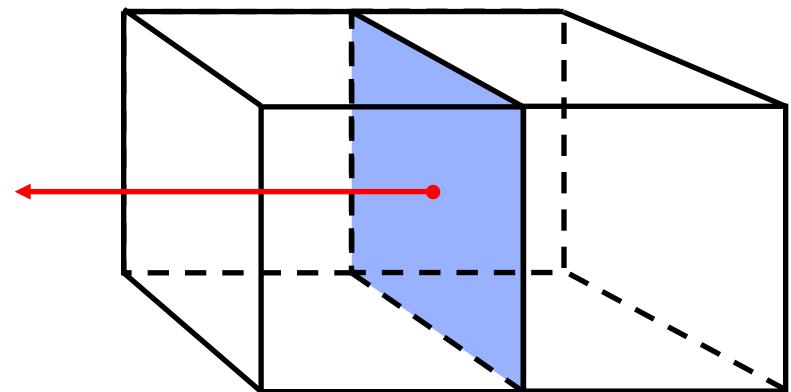
Cell

- 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

Spherical Grid

vector
vector
...
functions
fast
spectral
and

Numerical Stability

Finite Difference:

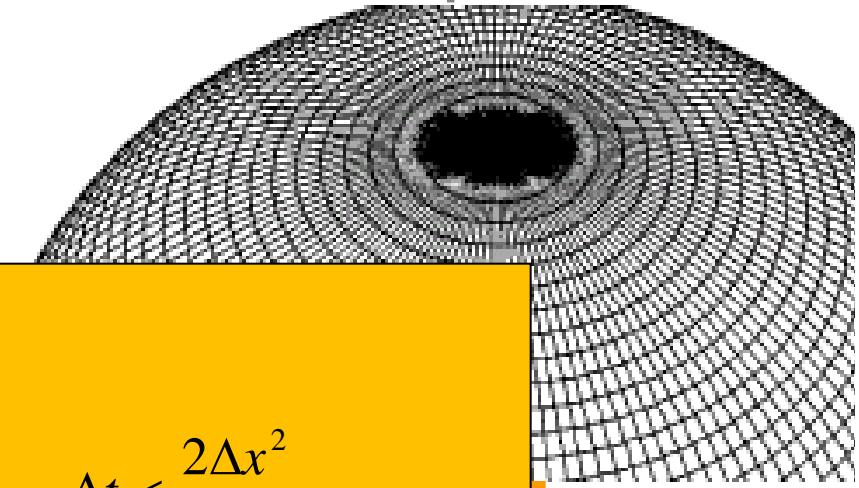
$$\Delta t < \frac{2\Delta x^2}{K}$$

Finite Volume:

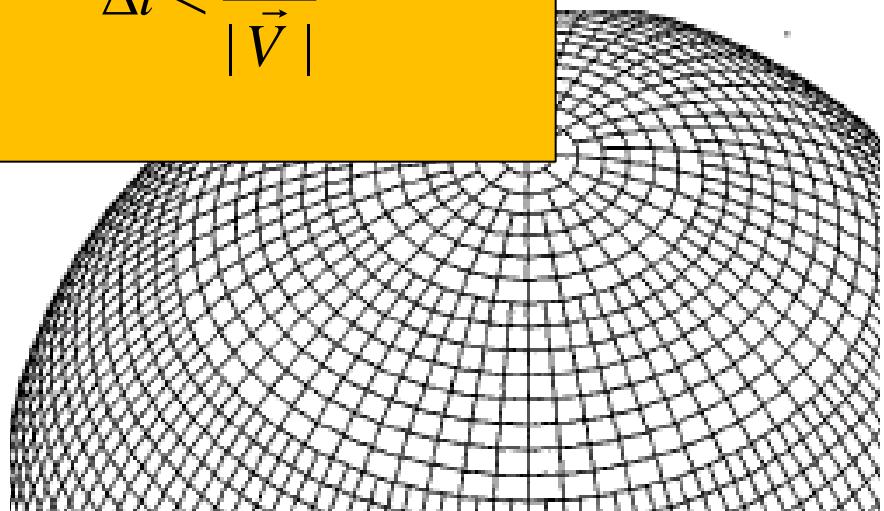
$$\Delta t < \frac{\Delta x}{|\vec{V}|}$$

‘Internally consistent, self-described grid’

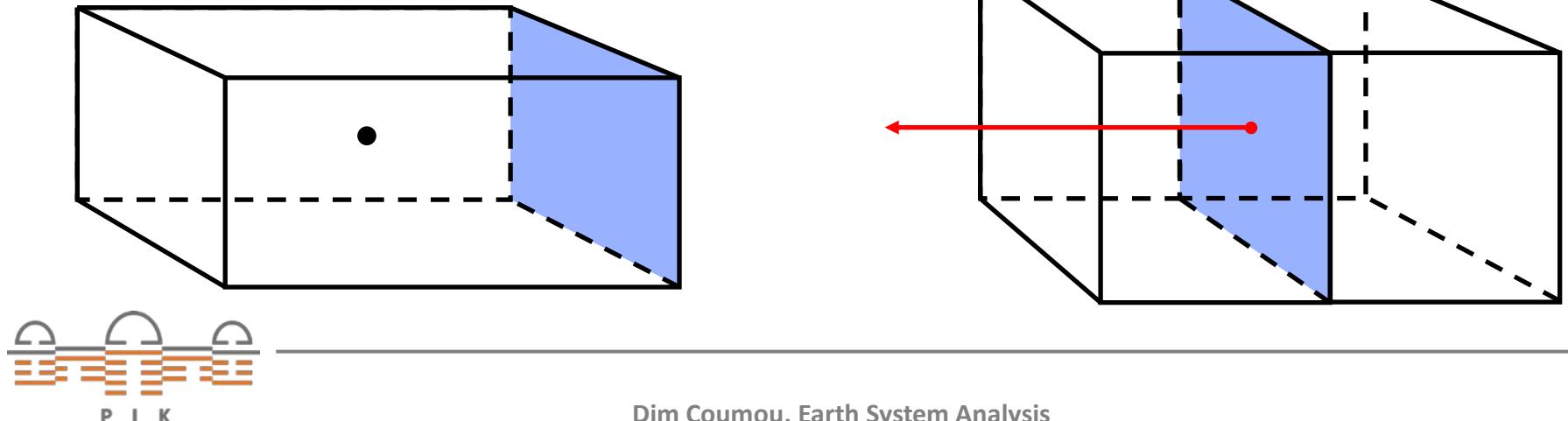
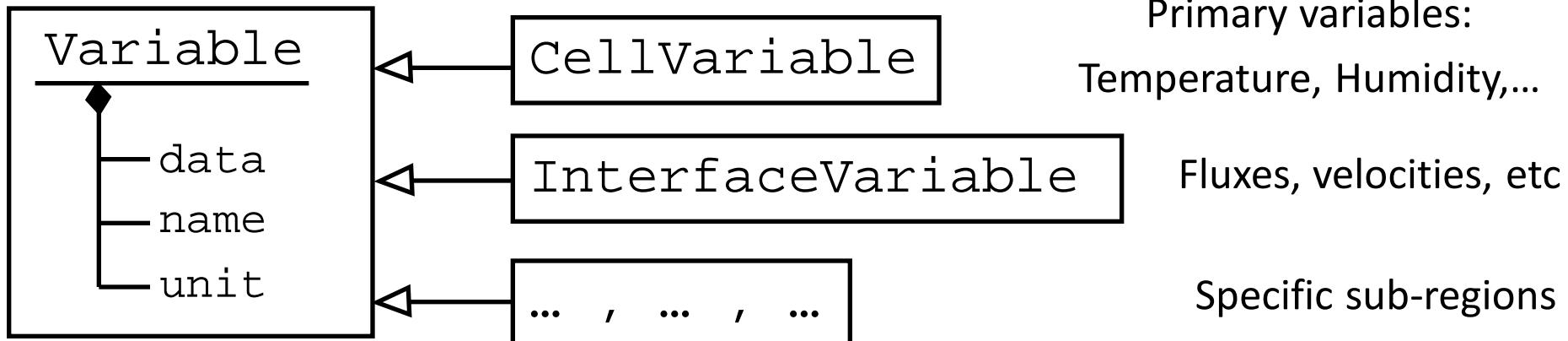
Standard Spherical Grid



Spherical Grid



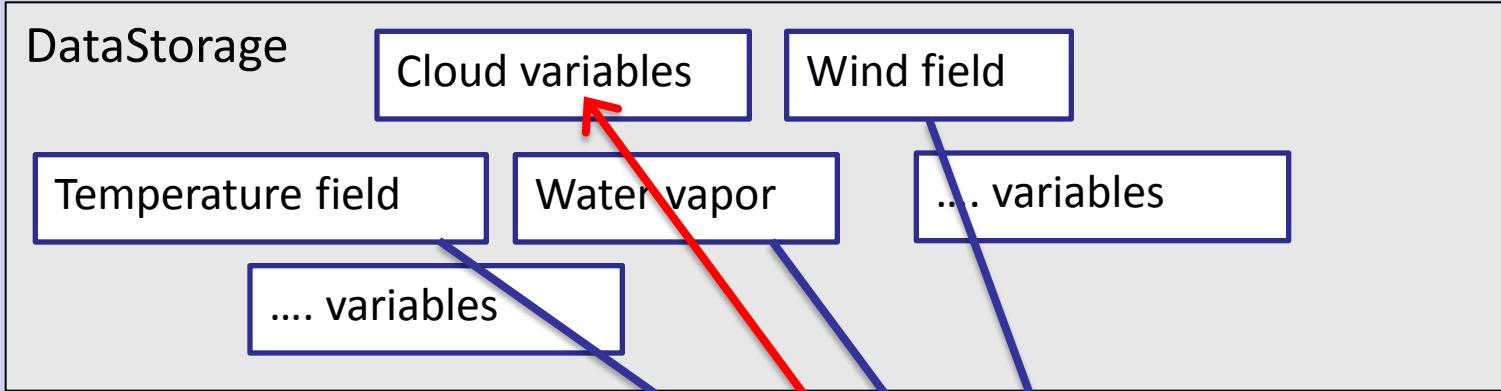
Data Storage



Explicit Read / Write access

→ Read access only
↔ Read & Write access

Aeolus



Compute Modules

3-Layer Clouds

Dynamical Core

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

```
class Cloudiness_3Layers    ←→ class Clouds_3Layers
class LargeScaleDynamics   ←→ class LargeScaleWind
class SynopticScaleDynamics ←→ class SynopticScaleField
class PlanWave              ←→ class PlanetaryWave
class HeatTransfer           ←→ class Temperature
class WaterVaporTransfer    ←→ class Humidity
class RadiativeTransfer     ←→ class RadiativeFluxes
class PlanetaryBoundaryLayerPhysics * class PlanetaryBoundaryLayer
class SurfaceFluxes          ←→ class SurfaceLayer
```

DataStorage classes



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

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

← ***Nan check, range check***

...Black box...

```
    // Calculate v3:

    // 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
- **Doxxygen**

Generates html/LaTeX documentation directly from source-code



AtmosphereIII_test: Hierarchical Index - Mozilla Firefox

File Edit View History Bookmarks Tools Help

AtmosphereIII_test: Hierarchical Index

file:///D:/work/P3_CodeDevelopment/doxygen/html/html/hierarchy.html

Yamaha Drum Modul S... Most Visited Getting Started Latest Headlines Delete Deutsche Guggenheim...

Main Page Namespaces Classes Files

Class List Class Index Class Hierarchy Class Members

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](#)

AtmosphereIII_test: Cloudiness Class Reference - Mozilla Firefox

File Edit View History Bookmarks Tools Help

AtmosphereIII_test: Cloudiness Class Reference

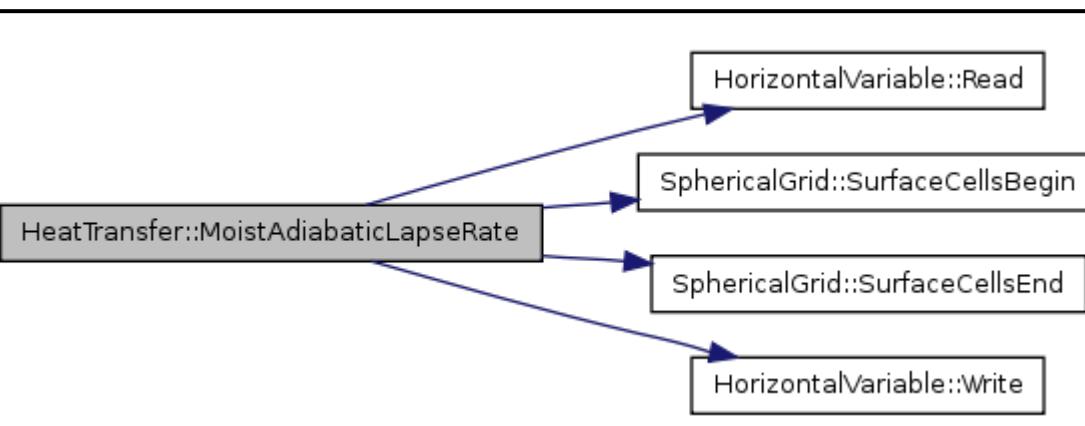
file:///D:/work/P3_CodeDevelopment/doxygen/html/html/classCloudiness.html

Yamaha Drum Modul S... Most Visited Getting Started Latest Headlines Delete Deutsche Guggenheim...

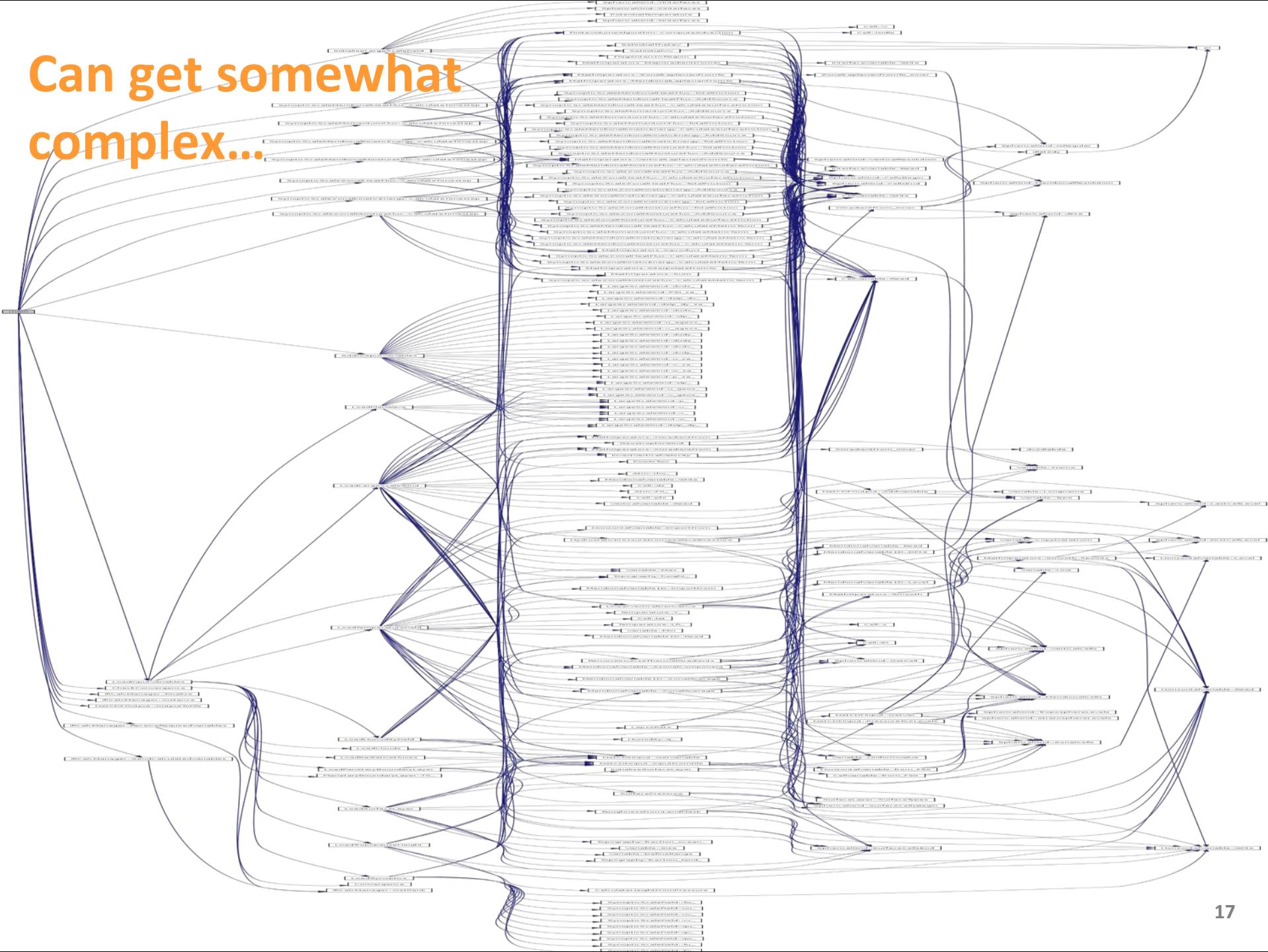
Private Attributes

| | | |
|--|--------------------------|---|
| const LargeScaleWind & | wind | constant reference to read large scale wind field |
| const SynopticScaleField & | syn | constant reference to read synoptic field |
| const Temperature & | temp | constant reference to read large scale wind field |
| const Humidity & | hum | constant reference to read large scale wind field |
| const SurfaceLayer & | sf_layer | constant reference to read equilibrium specific humidity at the surface |
| Clouds & | clouds | reference to store cloud field |
| Parameter< double > | a1 | |
| Parameter< double > | a2 | |
| Parameter< double > | a3 | |
| Parameter< double > | a5 | stratus cloud parameters |
| Parameter< double > | ncm | |
| Parameter< double > | b1 | |
| Parameter< double > | b2 | cumulus cloud parameters |

Call-graphs / Dependency-graphs



Can get somewhat complex...

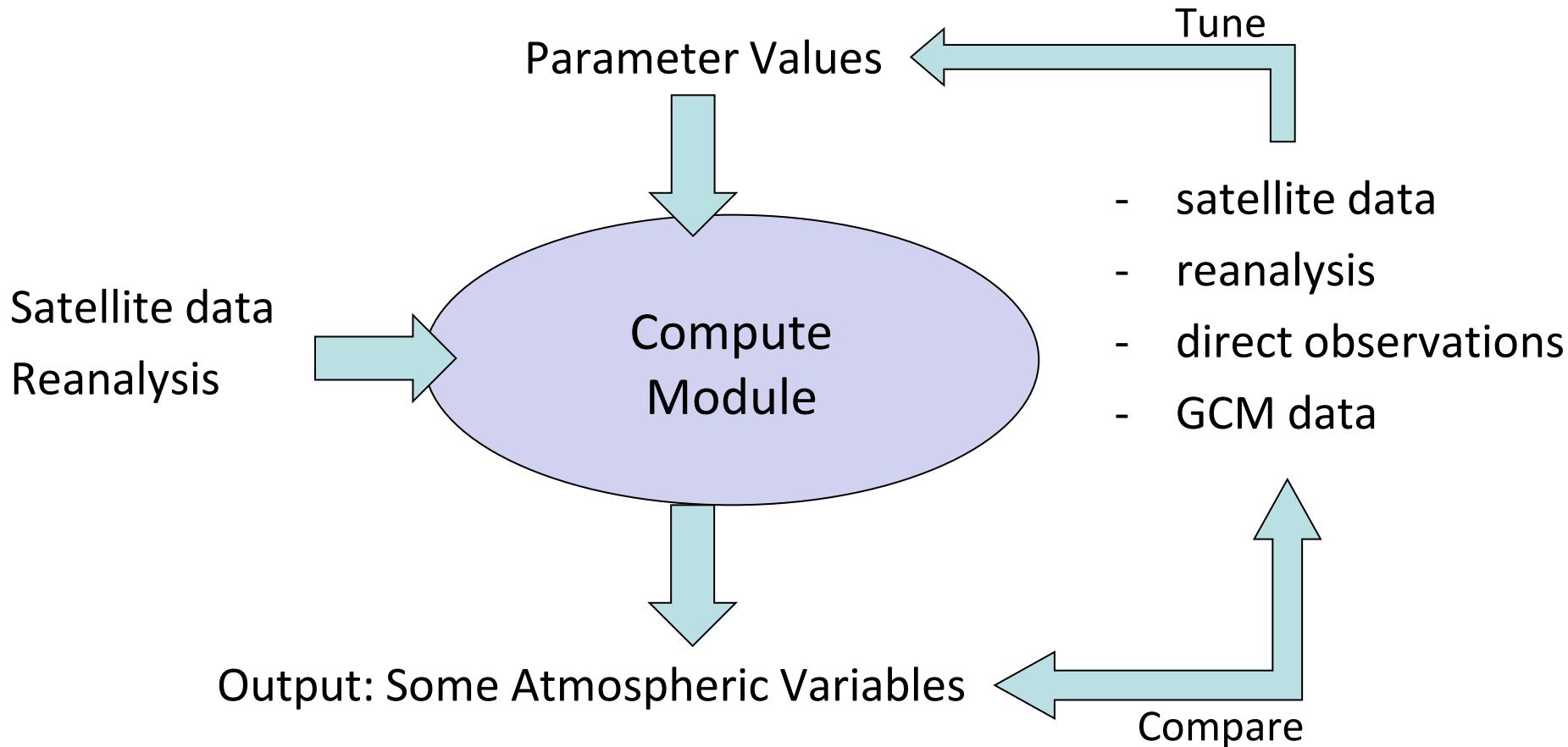


Content

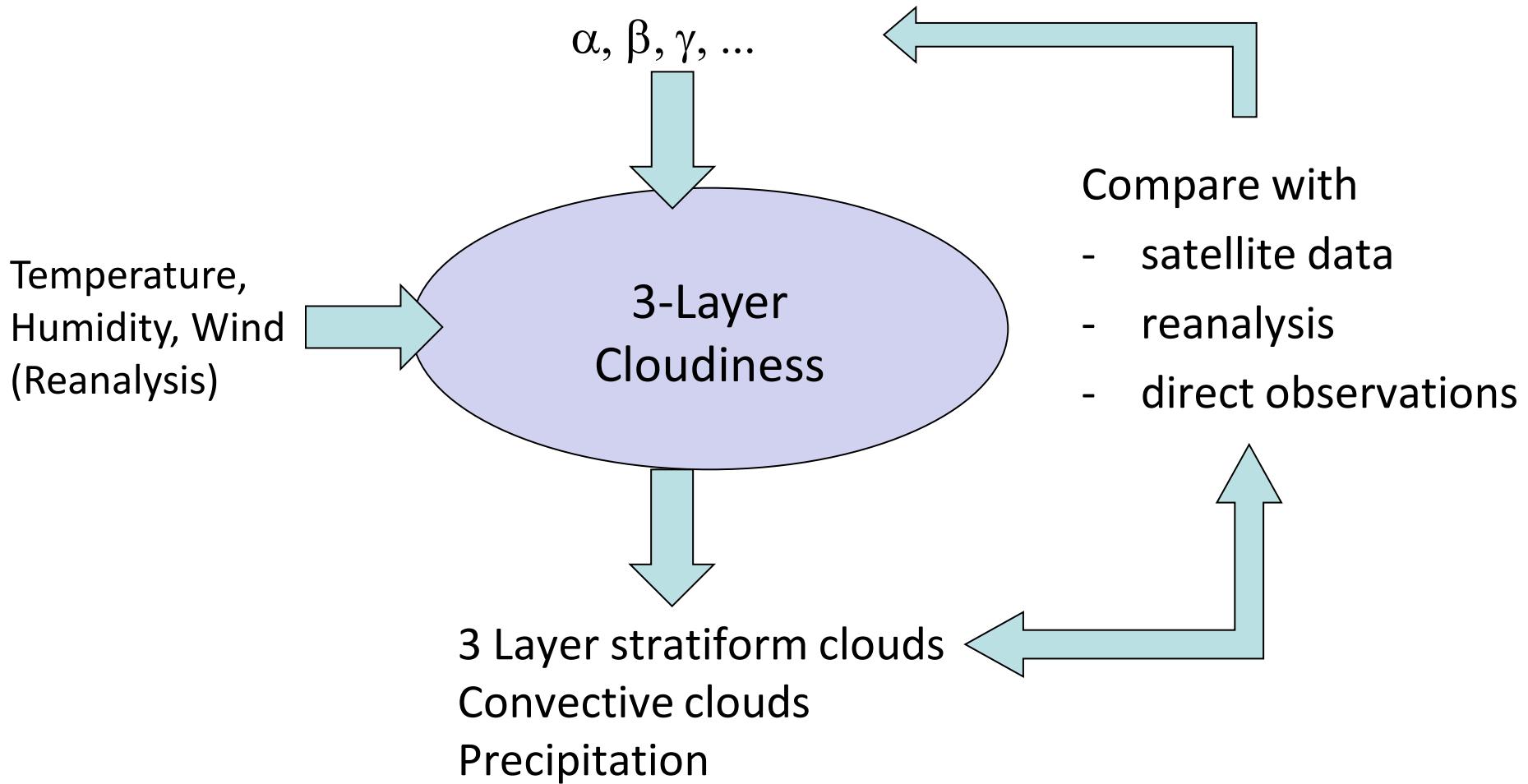
- Intro *Aeolus* and *Climber-4*
- Key Modules
- Object-oriented Code Design
 - Strict separation of *Data Storage, Gridding* and *Computations*
- Safety Mechanisms:
 - Explicit *Read-Write* Functionality, *Design-by-Contract*,
Automatic Range Checking
- Documentation
- **Testing / Benchmarking**
 -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



Tuning: Module-wise



Tuning: Example Cloudiness



Tuning: Example Cloudiness

Iterative optimization

For example using
SimENV

Temperature,
Humidity, Wind
(Reanalysis)

$\alpha, \beta, \gamma, \dots$

3-Layer
Cloudiness

3 Layer stratiform clouds
Convective clouds
Precipitation



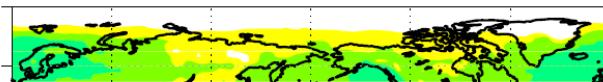
**Cost / Skill
Function**

Compare with
satellite data
reanalysis
- direct observations

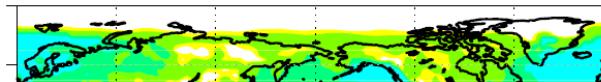
Tuning: Module Calibration

Improves low precipitation in major desert regions

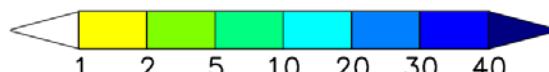
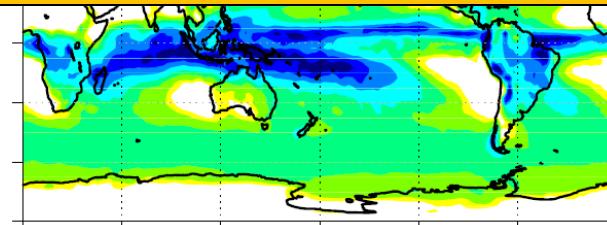
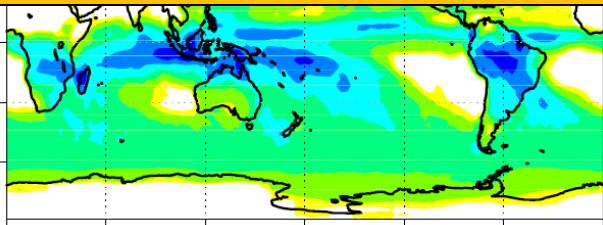
Aeolus - Initial



Aeolus – Calibrated



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



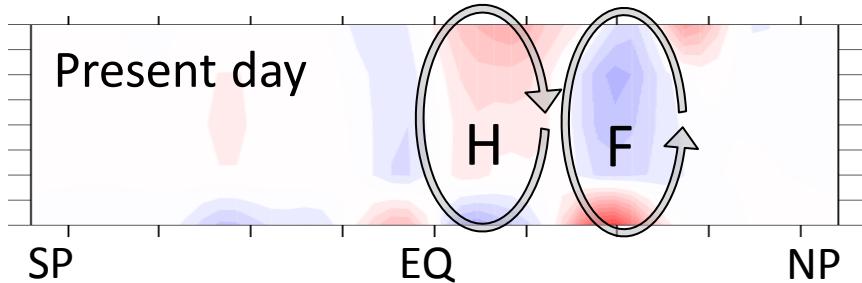
January Precipitation cm/mo



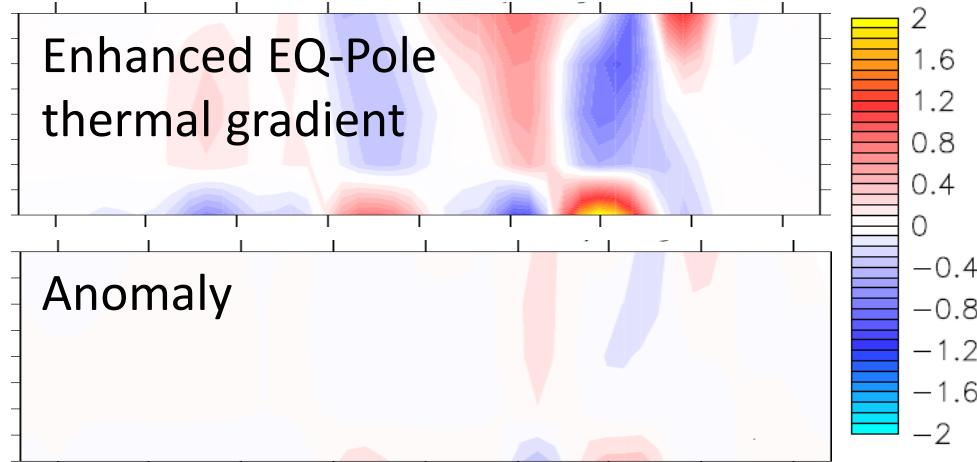
Eliseev et al, 2013

Tuning: Stand-alone Aeolus

Next Step: Sensitivity-analysis using SimENV



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

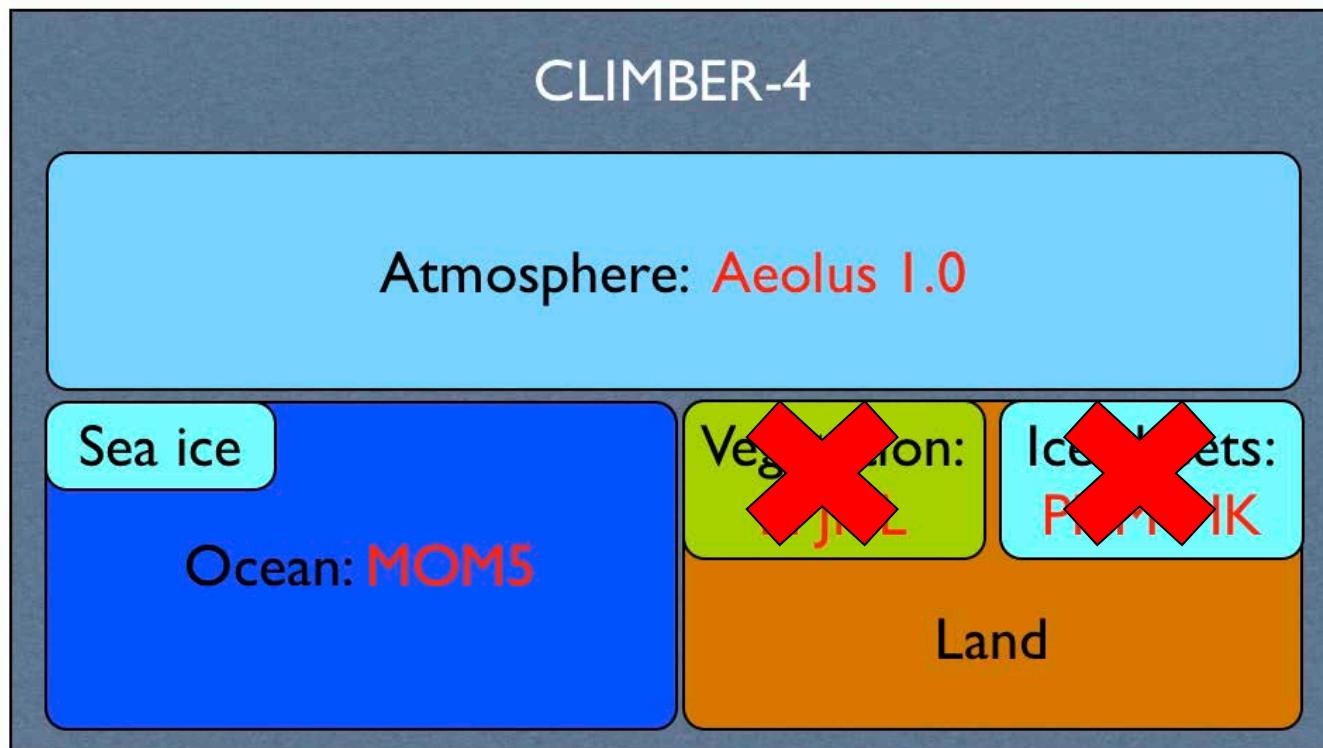


(Erik Peukert)

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)

The screenshot shows the Microsoft Visual Studio 2010 Express IDE interface. The main window displays the code for `Cloudiness_3Layers.h` in the Global Scope. The code defines a class `Cloudiness_3Layers` that inherits from `Compute`. It has a constructor taking various atmospheric parameters and a destructor. It also contains methods `InitZero()` and `TestThreeLayerClouds()`, which likely handle initializations and specific calculations for three-layer clouds. The Solution Explorer on the right shows the project structure for 'AtmosphereIII_test' with files like `Atmosphere.cpp`, `AuxiliaryFunctionality.cpp`, and `Cloudiness_3Layers.cpp`.

```
#pragma once
#include "Compute.h"
#include "Parameter.h"

class Cloudiness_3Layers : public Compute
{
public:
    Cloudiness_3Layers( const SphericalGrid& g,
                        const LargeScaleWind& wnd,
                        const SynopticScaleField& syn_,
                        const PlanetaryWave& pw_,
                        const Temperature& t,
                        const Humidity& q,
                        const PlanetaryBoundaryLayer& pbl,
                        const RadiativeFluxes& rad,
                        Clouds_3Layers& c );

    ~Cloudiness_3Layers(void);

    void InitZero( );
    void TestThreeLayerClouds(
        const HorizontalVariable& temp_surface, //< surface temperature [K]
        const HorizontalVariable& temp_hpbl, //< temperature at PBL height [K]
        const HorizontalVariable& lapseRate, //< for calculation of temperature profile
        const HorizontalVariable& shum_surface, //< specific humidity at surface [kg/kg]
        const HorizontalVariable& sat_hum_surface, //< saturated humidity at the surface [kg
        const CellVariable& rhum, //< relative humidity []
        const CellVariable& wls, //< large scale vertical velocity [m/s]
        const CellVariable& wsyn, //< standard deviation of the synoptic-scale velocity fluc
        const HorizontalVariable& woro, //< orographically induced velocity [m/s]
        const HorizontalVariable& htrn. //< height of the tronopause
    );
}
```

Tools: Totalview (Debugger)

main_Aeolus <@login02> Ubuntu VM Help

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 Stack Frame

C++ main, FP=7fff4660d1c0 _libc_start_main, FP=7fff4660d270 _start, FP=7fff4660d280

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

Action Points Processes Threads

P- P+ T- T+

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> Help

((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

:((class Variable &)pbl.TS).data._M_data[0:3587] <@login02>

File Window

Y Axis X Axis

299.
279.
259.
239.
0.00 1.00e+03 2.00e+03 3.00e+03 4.00e+03