The Climate Modelling Toolkit

Joy M. Monteiro
Rodrigo Caballero

Princeton, Nov 2-4

Stockholm University | SeRC Swedish e-Science Research Centre

What?

- A Python based toolkit
- A ***flexible*** problem solving environment
- Efficiency of compiled code
- Intuitive Python data structures

Configuring Computations [Beg et. al]

- Compile time (Some GCM parameters)
- Configuration file (Most GCMs)
- GUI (MATLAB's Simulink, PUMA/PLASIM)
- Domain specific language (DSL), i.e, run code within code (dedalus, climlab, CliMT)

Why DSLs?

- Higher level abstractions → Lesser code to specify model
- Full development environment available
- Full pipeline - Configure, Execute, Analyse, Plot
- Easier to use and reproduce

Why CliMT?

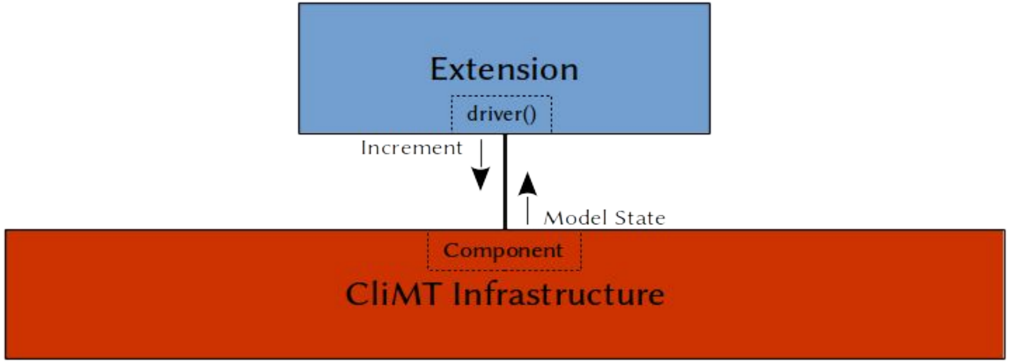
- Research standard components, classroom standard usability
- Easy to start with a simple model and build a more sophisticated one
- Integrates into the Python scientific computing ecosystem

For Whom?

- Not the High-* community!
- Idealised/Simplified Modelling
- Algorithm development
- Sensitivity studies
- Classroom

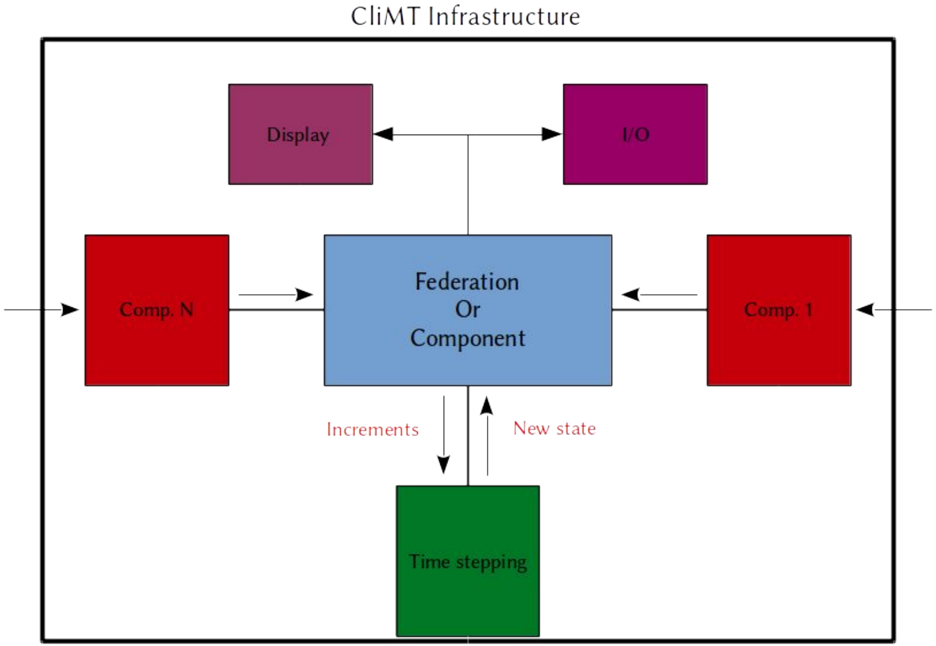
How?

- Extensions → Compiled Code
- Components → CliMT Interface

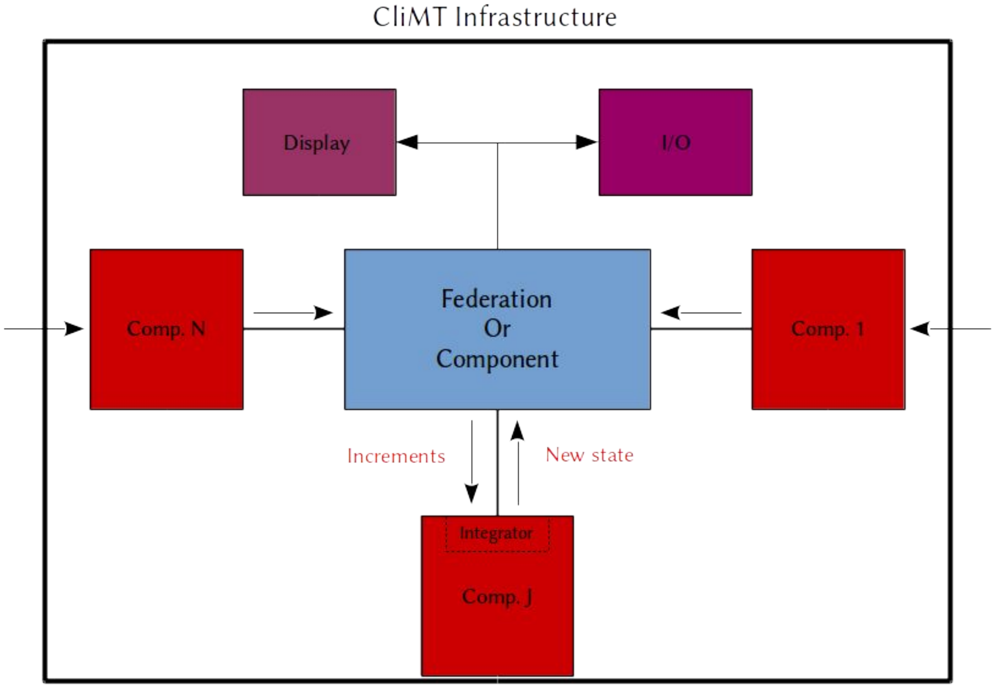


How?

- *Federation* → Σi Component(i)



How?

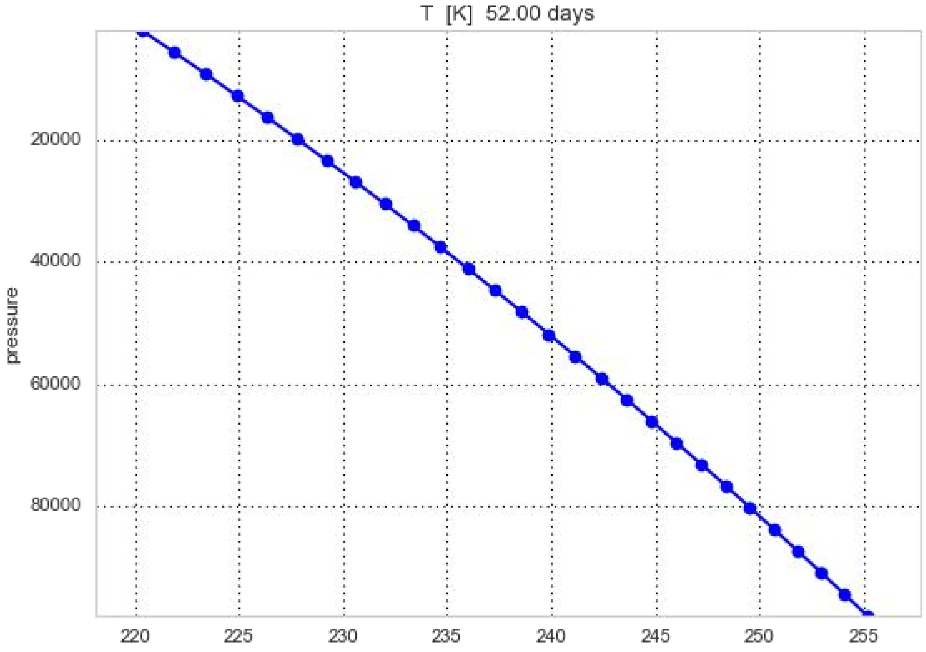

CliMT Infrastructure
Display
I/O
Comp. N
Federation
Or
Component
Comp. 1
Increments
New state
Integrator
Comp. J

How?

- Component is configured using a dictionary
- Optionally, Components are combined in a Federation
- Model runs within a simple while/for loop
- Model state is accessed by treating Fed/Comp as dictionary

Radiative Equilibrium

```
import numpy as np
import climt
import seaborn as sns
sns.set_style('whitegrid',rc={'grid.linestyle':'dotted', 'grid.color':'0.0'})

# All configuration is done via a python dictionary
kwargs = {}
kwargs['MonitorFields'] = ['T']
kwargs['solin'] = 400.
rad = climt.radiation(scheme='newgreygas', **kwargs)

# Fields are accessed by treating Component as dictionary
T = np.zeros(rad.nlev) + (kwargs['solin']/2./5.67e-8)**0.25
rad['T'] = T

for i in range(5000):
       rad.step()
```



Radiative Equilibrium

Radiative-Convective Equilibrium

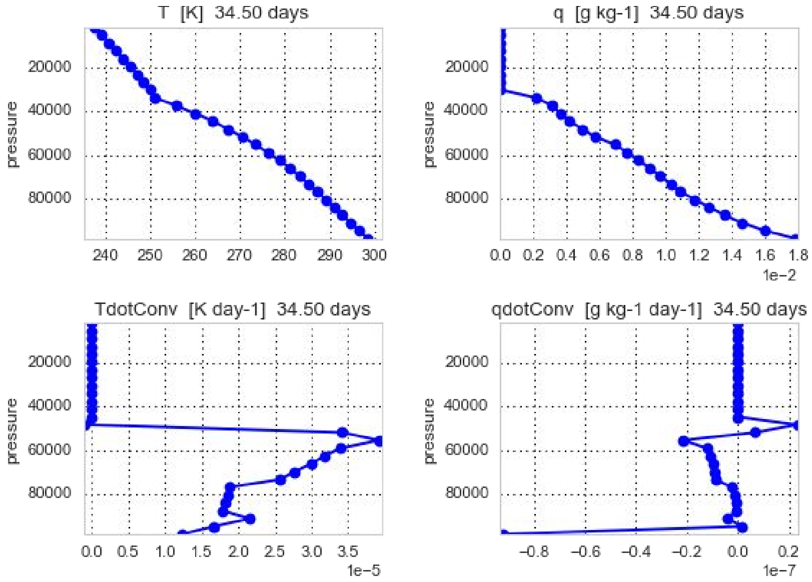
```
import climt
from climt.simple_physics_custom import simple_physics_custom

global_time_step = 300.

#Initialise radiation
rad = climt.radiation(scheme='newgreygas')

#Initialise simple physics
kwargs = {}
kwargs['lsc'] = False
kwargs['use_ext_ts'] = True
kwargs['dt'] = global_time_step

solar_in = 450.
Ts = (solar_in/5.67e-8)**0.25*np.ones((1,1))
kwargs['Ts'] = Ts

phys = simple_physics_custom(**kwargs)
```

Radiative-Convective Equilibrium

```
#Initialise convection
kwargs = {}
kwargs['dt'] = global_time_step

conv = climt.convection(scheme='emanuelnew', **kwargs)

#Initialise federation
kwargs = {}
kwargs['MonitorFields'] = ['T','q','TdotConv','qdotConv']
T = np.zeros((1,1,rad.nlev)) + (solar_in/2./5.67e-8)**0.25
U = 20.*np.ones((1,1,rad.nlev))

kwargs['dt'] = global_time_step
kwargs['T'] = T
kwargs['U'] = U
kwargs['Ts'] = Ts

fed = climt.federation(rad, phys, conv, **kwargs)
```

Radiative-Convective Equilibrium

```
for i in range(10000):
        #Maintain boundary layer winds
        climt_U = fed['U']
        dU = -(1./86400)*global_time_step*(climt_U - U)
        fed.step(Inc={'U':dU})
        #Set negative values of q to zero
        q = fed['q']
        q[q<0] = 0
        fed['q'] = q
```



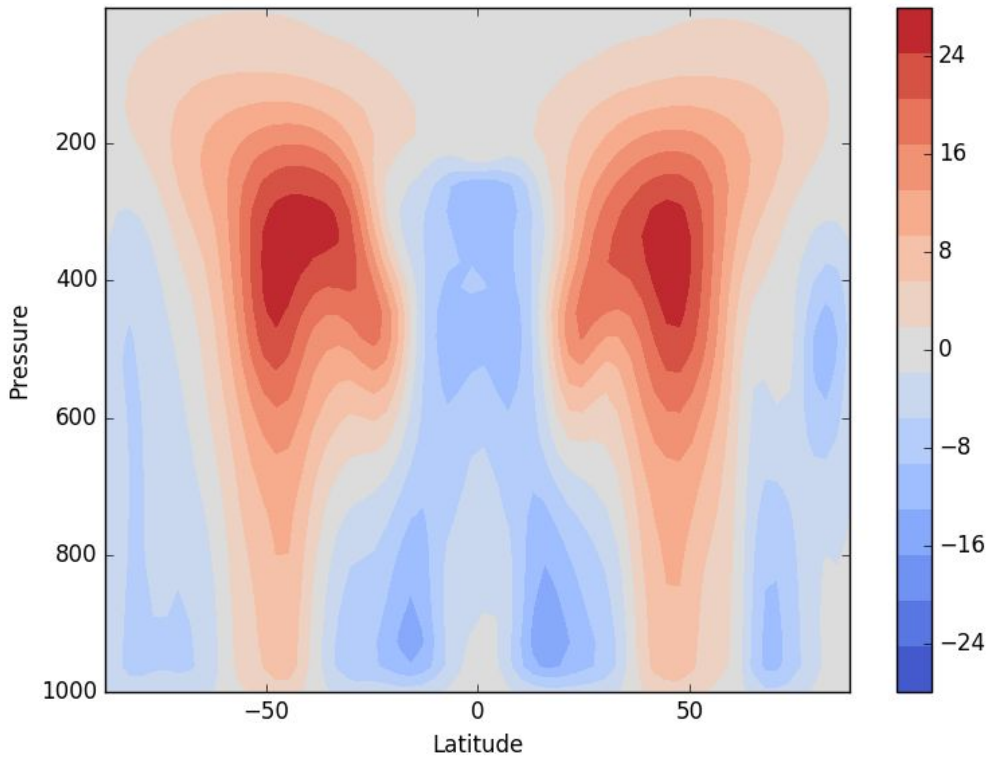
In contrast to...

- Model Configuration → Build configuration file
- Physics Configuration → Namelist
- Execution → Shell script

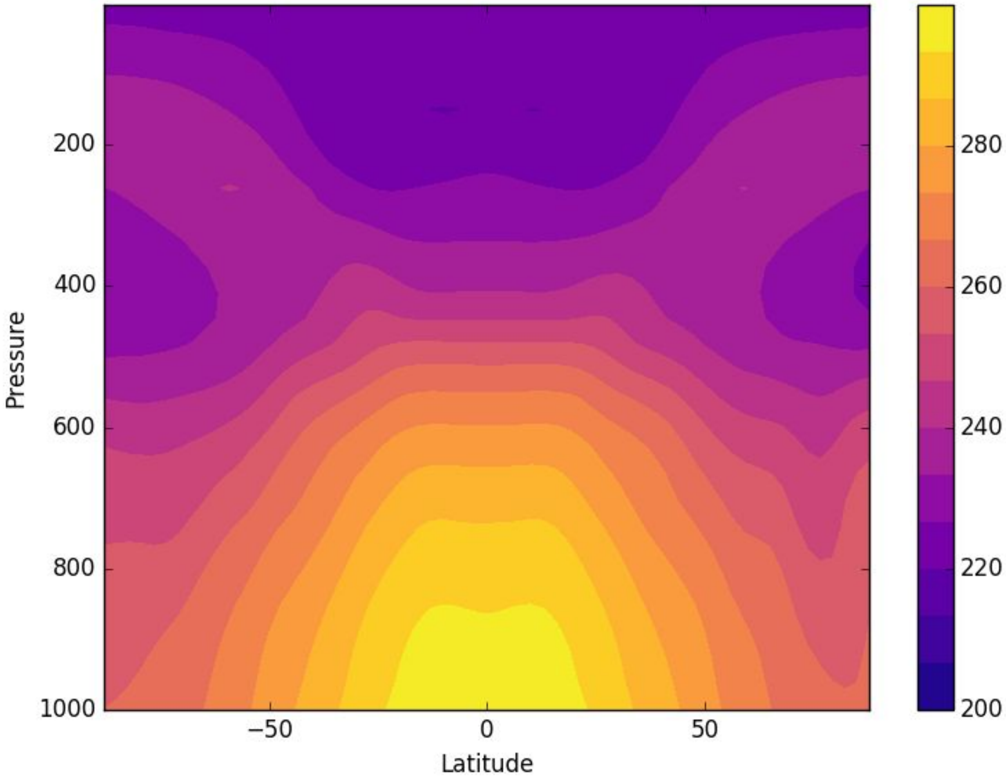
Idealised GCM

- Just add dynamical core to federation!
- Small catch: need to propagate dynamical core grid

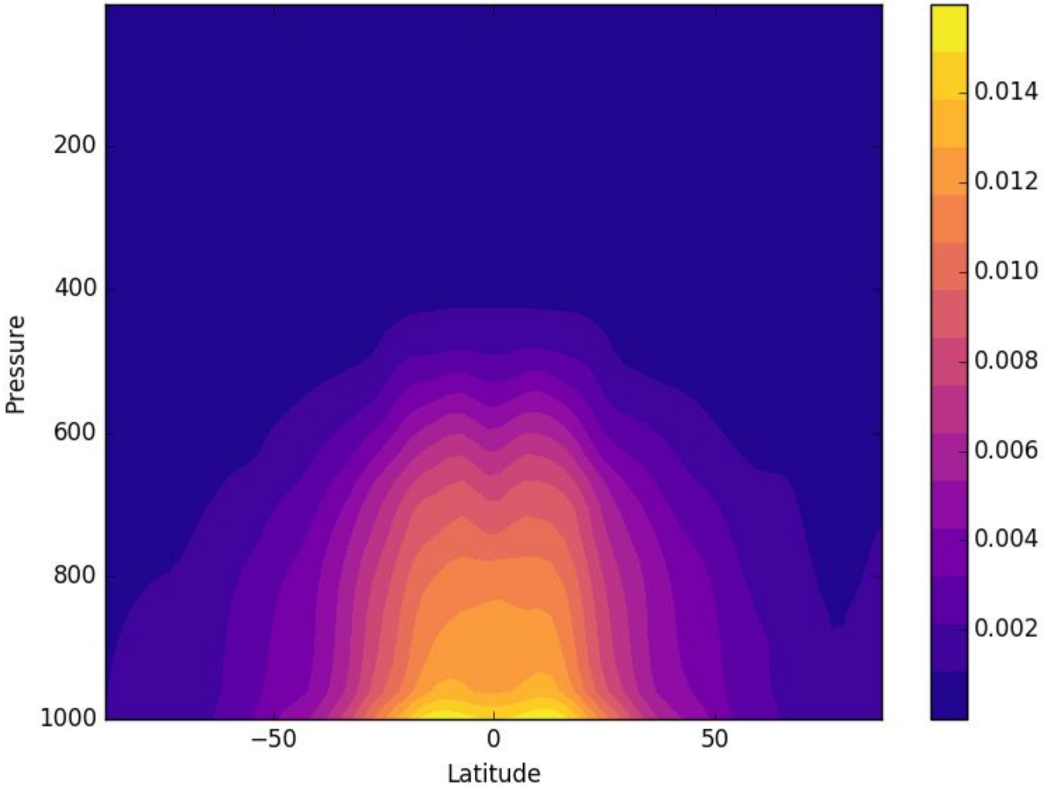
```
fed = federation(dycore, rad, phys, conv, **kwargs)
```



Zonal Winds



Temperature



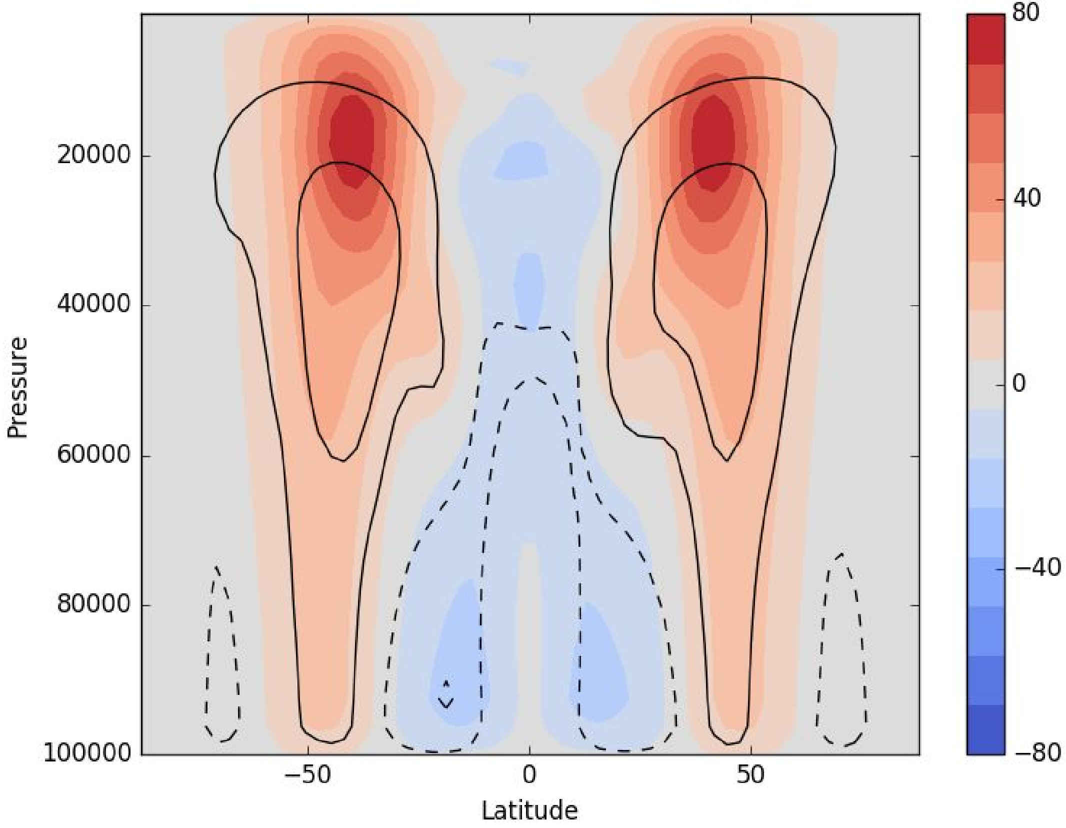
Specific Humidity

And if you change...

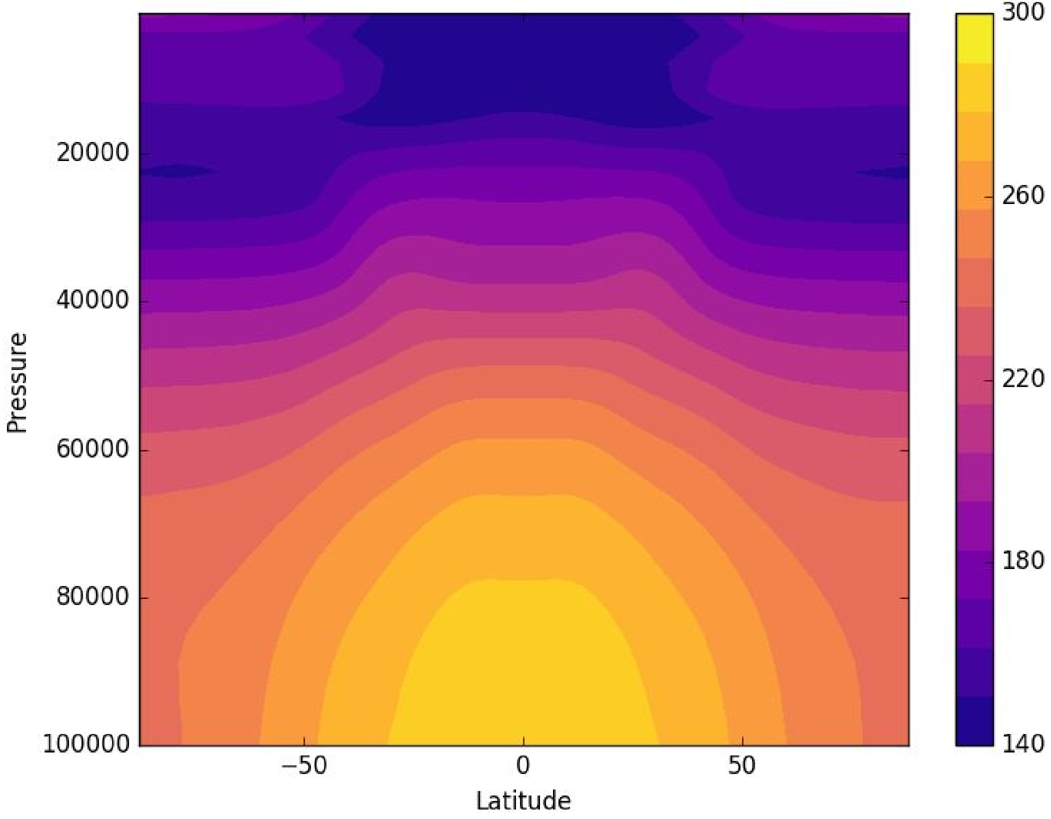
```
rad = climt.radiation(scheme='newgreygas', **kwargs)
```

to

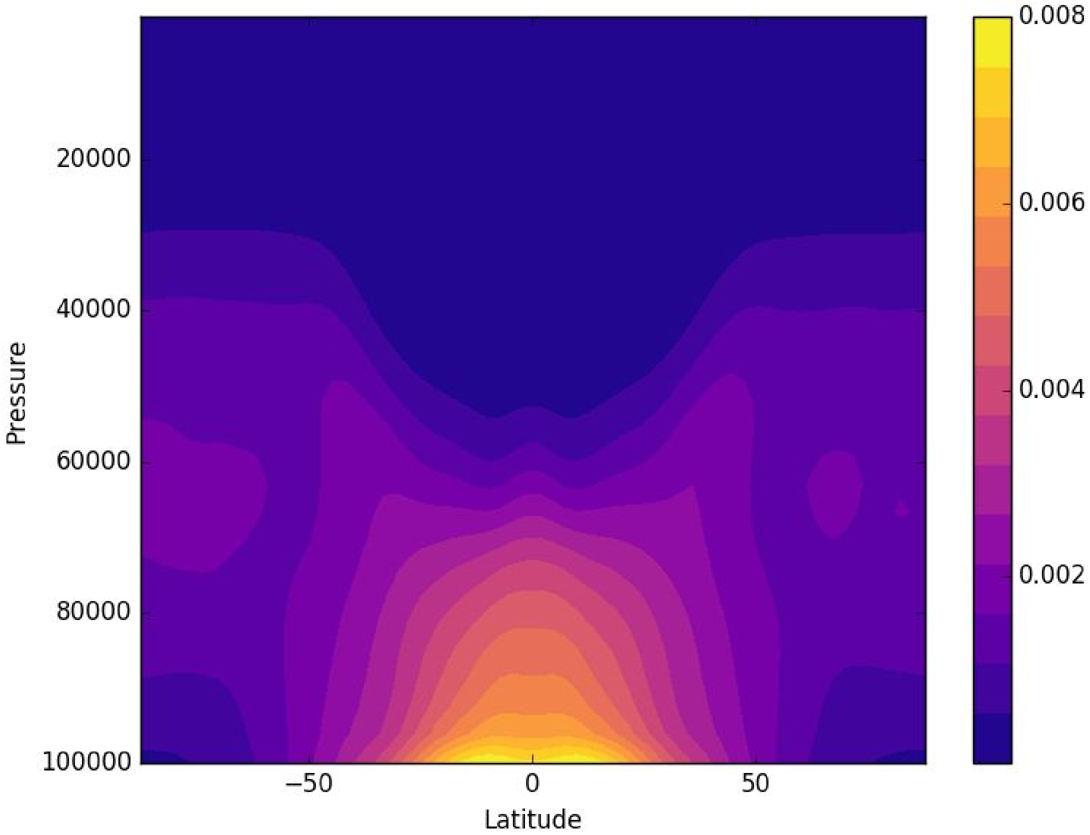
```
rad = climt.radiation(scheme='ccm3', **kwargs)
```


20000
40000
60000
80000
100000
Pressure
−50
0
50
Latitude
80
40
0
−40
−80

Zonal Winds



Temperature



Specific Humidity

What can you try out?

- Radiative-Convective Equilibrium
- Held-Suarez
- DCMIP tests (Dry BC wave, Maintain basic state, Trop. Cyclone)
- Idealised moist GCM with grey or realistic radiation

Towards CLiMT 1.0

- Feb-end 2017
- Will actually be usable!
  - Documentation
  - Regression testing
  - Packaging (pip, hopefully conda)

https://github.com/CLiMT/climt-python