Developing Weather and Climate Models in Python

Jeremy McGibbon
AI2 Climate Modeling Team
Lucas Harris, Rusty Benson (GFDL)

AI2 Climate Modeling: Who We Are

ML group

Chris Bretherton

Oli Fuhrer

DSL group

Noah Brenowitz

Anna Kwa

Oli Watt-Meyer

Jeremy McGibbon

Florian Deconinck

Eddie Davis

Rhea George

Spencer Clark

Andre Perkins

Brian Henn

Elynn Wu

Johann Dahm

Oliver Elbert

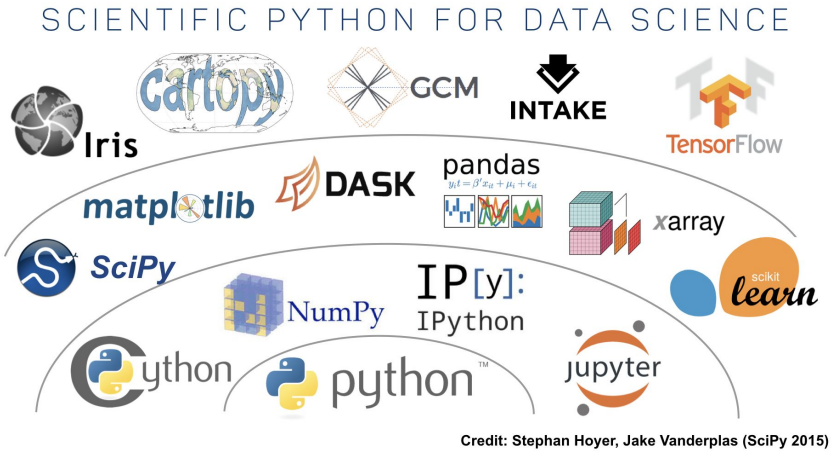
Tobias Wicky

Atmospheric Models in Python

AI2

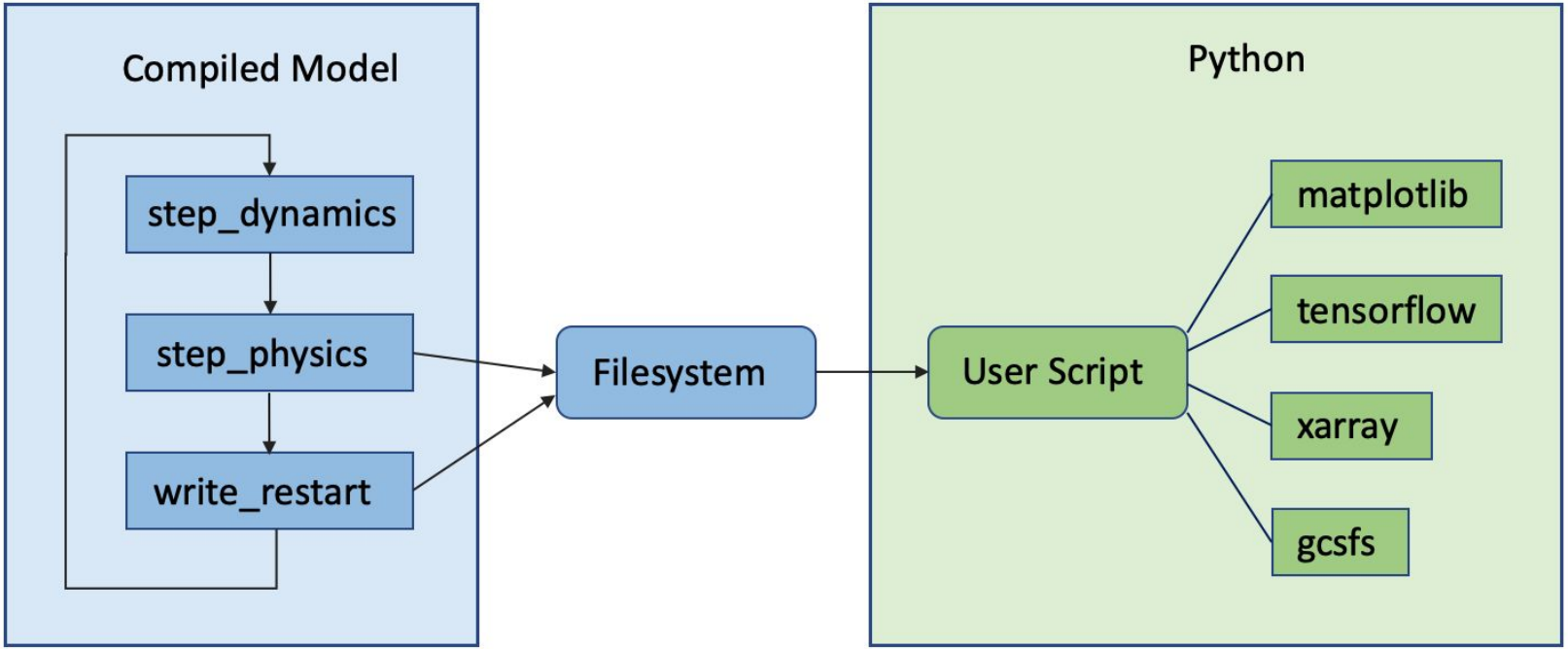
Why do we use Python?

- Machine learning libraries
- Cloud file storage libraries
- Comfortable for all of our scientists
- Interactive execution
- Built-in introspection libraries

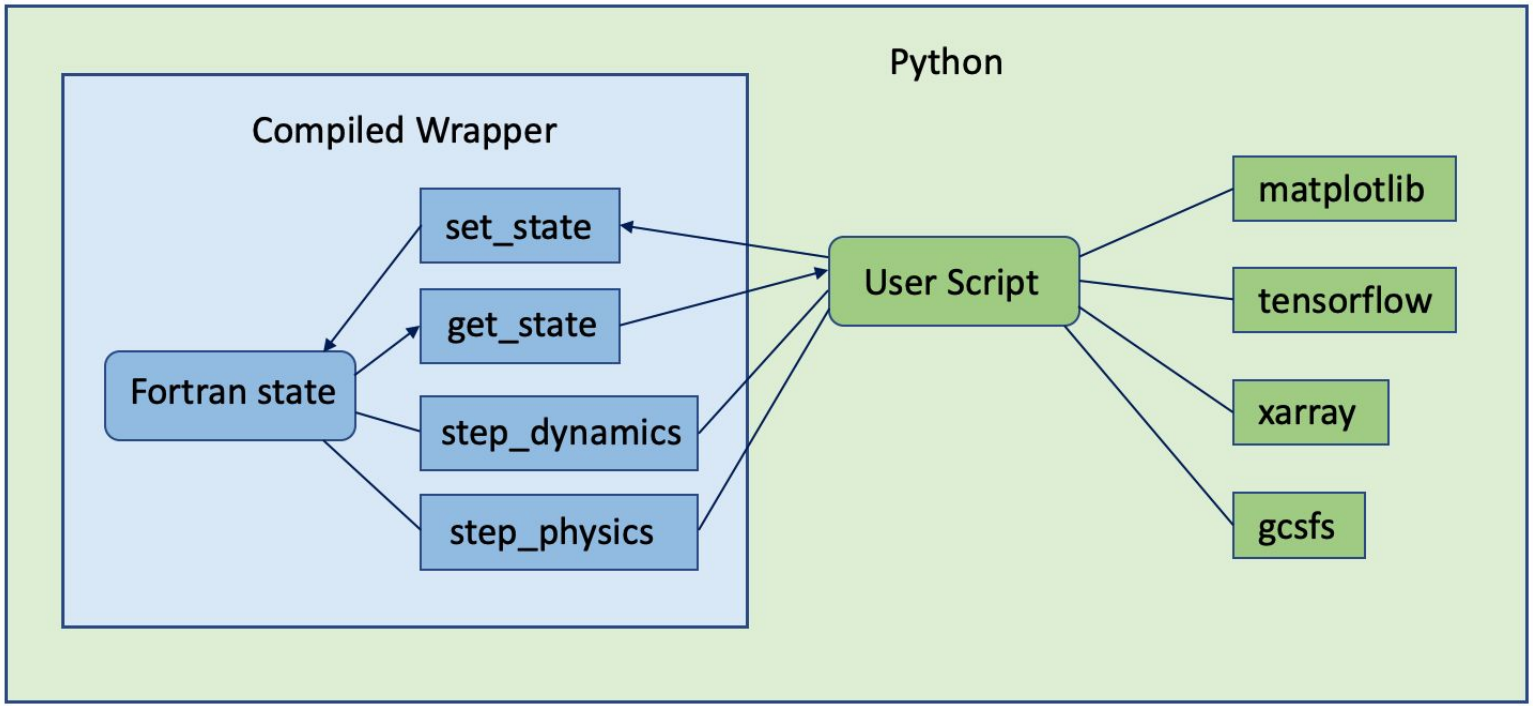


Credit: Stephan Hoyer, Jake Vanderplas (SciPy 2015)

Many model users use Python



Online Python Modeling



fv3gfs-wrapper

Simple Python main loops provide a clear way to modify the model

```
import fv3gfs.wrapper

if __name__ == "__main__":
    fv3gfs.wrapper.initialize()
    for i in range(fv3gfs.wrapper.get_step_count()):
        fv3gfs.wrapper.step_dynamics()
        fv3gfs.wrapper.step_physics()
        fv3gfs.wrapper.save_intermediate_restart_if_enabled()
    fv3gfs.wrapper.cleanup()
```

McGibbon et al. 2021, “fv3gfs-wrapper: a Python wrapper of the FV3GFS atmospheric model”

AI2

Examples

Random forest

Can add online functionality by getting and setting the Fortran model state

```python
import fv3gfs.wrapper
import fv3gfs.wrapper.examples
import f90nml
from datetime import timedelta

if __name__ == "__main__":
    # load timestep from the namelist
    namelist = f90nml.read("input.nml")
    timestep = timedelta(seconds=namelist["coupler_nml"]["dt_atmos"])
    # initialize the machine learning model
    rf_model = fv3gfs.wrapper.examples.get_random_forest()
    fv3gfs.wrapper.initialize()
    for i in range(fv3gfs.wrapper.get_step_count()):
        fv3gfs.wrapper.step_dynamics()
        fv3gfs.wrapper.step_physics()

        # apply an update from the machine learning model
        state = fv3gfs.wrapper.get_state(rf_model.inputs)
        rf_model.update(state, timestep=timestep)
        fv3gfs.wrapper.set_state(state)

        fv3gfs.wrapper.save_intermediate_restart_if_enabled()
    fv3gfs.wrapper.cleanup()
```

Watt-Meyer et al. 2021, “Correcting Weather and Climate Models by Machine Learning Nudged Historical Simulations”

MPI-Enabled

MPI for Python (mpi4py) for distributed-memory parallelism

```python
import fv3gfs.wrapper
import numpy as np
from mpi4py import MPI

ROOT = 0

if __name__ == "__main__":
    fv3gfs.wrapper.initialize()
    # MPI4py requires a receive "buffer" array to store incoming data
    min_surface_temperature = np.array(0.0)
    for i in range(fv3gfs.wrapper.get_step_count()):
        fv3gfs.wrapper.step_dynamics()
        fv3gfs.wrapper.step_physics()

        # Retrieve model minimum surface temperature
        state = fv3gfs.wrapper.get_state(["surface_temperature"])
        MPI.COMM_WORLD.Reduce(
            state["surface_temperature"].view[:].min(),
            min_surface_temperature,
            root=ROOT,
            op=MPI.MIN,
        )
        if MPI.COMM_WORLD.Get_rank() == ROOT:
            units = state["surface_temperature"].units
            print(f"Minimum surface temperature: {min_surface_temperature}
            {units}")

        fv3gfs.wrapper.save_intermediate_restart_if_enabled()
    fv3gfs.wrapper.cleanup()
```

Overhead is minimal

Example	Runtime (s)
Fortran baseline	110
Wrapper baseline	110
Random forest	116
Minimum surface pressure	110

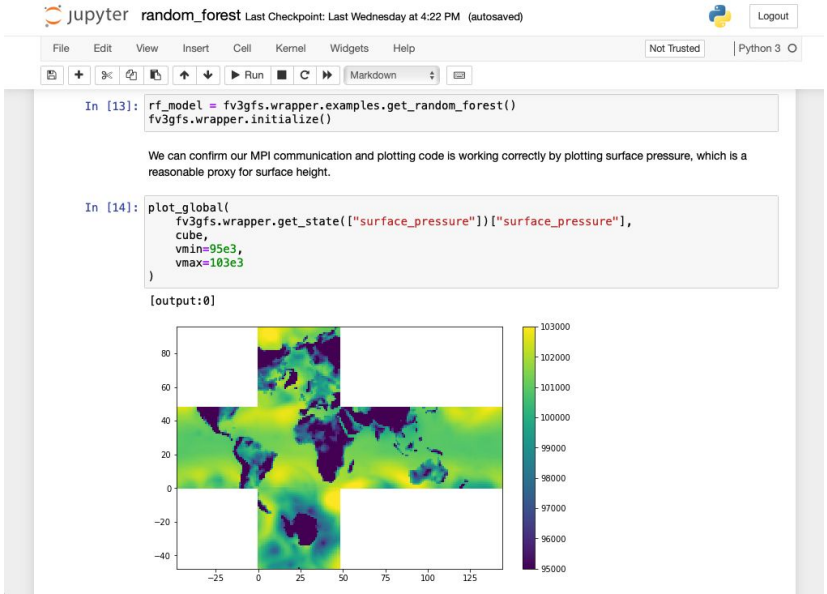
6h simulation time at C48 on 2019 Macbook Pro

McGibbon et al. 2021, “fv3gfs-wrapper: a Python wrapper of the FV3GFS atmospheric model”

Developing in Python

MPI-Parallel Jupyter Notebooks

- Everything we've seen can be run in a Jupyter notebook
- Useful for communicating/learning
- Scientifically evaluate model components



Pytest / unit tests

```
pytest test.py
```

```
pytest test.py --pdb
```

```python
def test_nonnegative_model_outputs():
    hyperparameters = DenseHyperparameters(
        ["input"], ["output"], nonnegative_outputs=True
    )
    model = DenseModel("sample", ["input"], ["output"], hyperparameters,)
    batch = xr.Dataset(
        {
            "input": (["x"], np.arange(100)),
            # even with negative targets, trained model should be nonnegative
            "output": (["x"], np.full((100,), -1e4)),
        }
    )
    model.fit([batch])
    prediction = model.predict(batch)
    assert prediction.min() >= 0.0
```

AI2

Pytest / unit tests

```
pytest test.py
```

```
platform darwin -- Python 3.7.10, pytest-6.0.1, py-1.9.0, pluggy-0.13.1
Matplotlib: 3.2.2
Freetype: 2.6.1
rootdir: /Users/jeremym/python/fv3net, configfile: pytest.ini
plugins: typeguard-2.10.0, hypothesis-6.17.4, regtest-1.4.4, mpl-0.12
collected 14 items

tests/test_loss.py ..............                                        [100%]

========================== 14 passed, 5 warnings in 2.92s ==========================
```

```
    def test_weighted_mae(weights, std, y_true, y_pred, reference):
        loss = _weighted_mae(weights, std)
        result = loss(y_true, y_pred)
>       np.testing.assert_almost_equal(result, reference)
E       AssertionError:
E       Arrays are not almost equal to 7 decimals
E        ACTUAL: <tf.Tensor: shape=(), dtype=float32, numpy=8.0>
E        DESIRED: 7.0

tests/test_loss.py:177: AssertionError
============================== short test summary info ==============================
FAILED tests/test_loss.py::test_weighted_mae[double_single_feature_loss] - Asser...
==================== 1 failed, 13 passed, 5 warnings in 3.24s ====================
```

Pytest / unit tests

```
pytest test.py --pdb
```

```
>       np.testing.assert_almost_equal(result, reference)
E       AssertionError:
E       Arrays are not almost equal to 7 decimals
E        ACTUAL: <tf.Tensor: shape=(), dtype=float32, numpy=8.0>
E        DESIRED: 7.0

tests/test_loss.py:177: AssertionError
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> entering PDB >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

>>>>>>>>>>>>>>>>>>> PDB post_mortem (IO-capturing turned off) >>>>>>>>>>>>>>>>>>>
> /Users/jeremym/python/fv3net/external/fv3fit/tests/test_loss.py(177)test_weighted_mae()
-> np.testing.assert_almost_equal(result, reference)
(Pdb) h

Documented commands (type help <topic>):
========================================
EOF    cl         down      j         next      return  tbreak     w
a      clear      enable    jump      p         retval  u          whatis
alias  commands   exit      l         pp        run     unalias    where
args   condition  h         list      q         rv      undisplay
b      d          help      ll        quit      s       unt
break  disable    ignore    longlist  r         source  until
bt     display    interact  n         restart   step    up

Miscellaneous help topics:
==========================
exec  pdb

Undocumented commands:
======================
c  cont  continue  debug
```

Pytest / unit tests

- Unit tests make debugging very fast
- Removes fear of breaking things, even with many developers
- Tests are examples of how to use the code
- Can add `--pdb` to get a debugging terminal whenever a test fails

```python
def test_nonnegative_model_outputs():
    hyperparameters = DenseHyperparameters(
        ["input"], ["output"], nonnegative_outputs=True
    )
    model = DenseModel("sample", ["input"], ["output"], hyperparameters,)
    batch = xr.Dataset(
        {
            "input": (["x"], np.arange(100)),
            # even with negative targets, trained model should be nonnegative
            "output": (["x"], np.full((100,), -1e4)),
        }
    )
    model.fit([batch])
    prediction = model.predict(batch)
    assert prediction.min() >= 0.0
```

With Python, Google is your friend

Introspection tools

- Built-in libraries for code analysis (inspect, ast)
- Well-tested templating libraries developed for HTML generation can be used for code generation (e.g. Jinja)

```
import ast
import inspect

def func():
    a = 1
    return a

source = inspect.getsource(func)
# code -> abstract syntax tree
tree = ast.parse(source)
```

For more introspection magic:
http://hackflow.com/blog/2015/03/29/metaprogramming-beyond-decency/

AI2

Stay tuned: Pace

- Re-writing FV3/GFS/SHiELD in Python-based DSL gt4py
- Early results presented by Oliver Fuhrer earlier in this session
- Unified code base for CPU and GPU hardware
- Will be presented at AMS 2021 Annual Meeting

Thank You!