Analysis and lightweight verification of Fortran code

Dominic Orchard

24th April - EGU23 DataWave side event

UNIVERSITY OF CAMBRIDGE

Institute of Computing for Climate Science

University of Kent

Programming Languages and Systems for Science laboratory

work also with Matthew Danish, Andrew Rice, Mistral Contrastin, Ben Orchard

thanks also to Bloomberg

UKRI Engineering and Physical Sciences Research Council

SCHMIDT FUTURES

Validation

Did we implement the right equations?

better climatologies

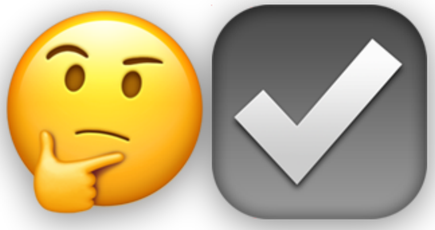
vs

Verification

Did we implement the equations right?

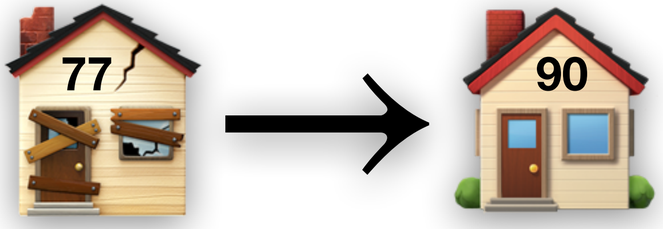
CamFort

Verification



Analysis

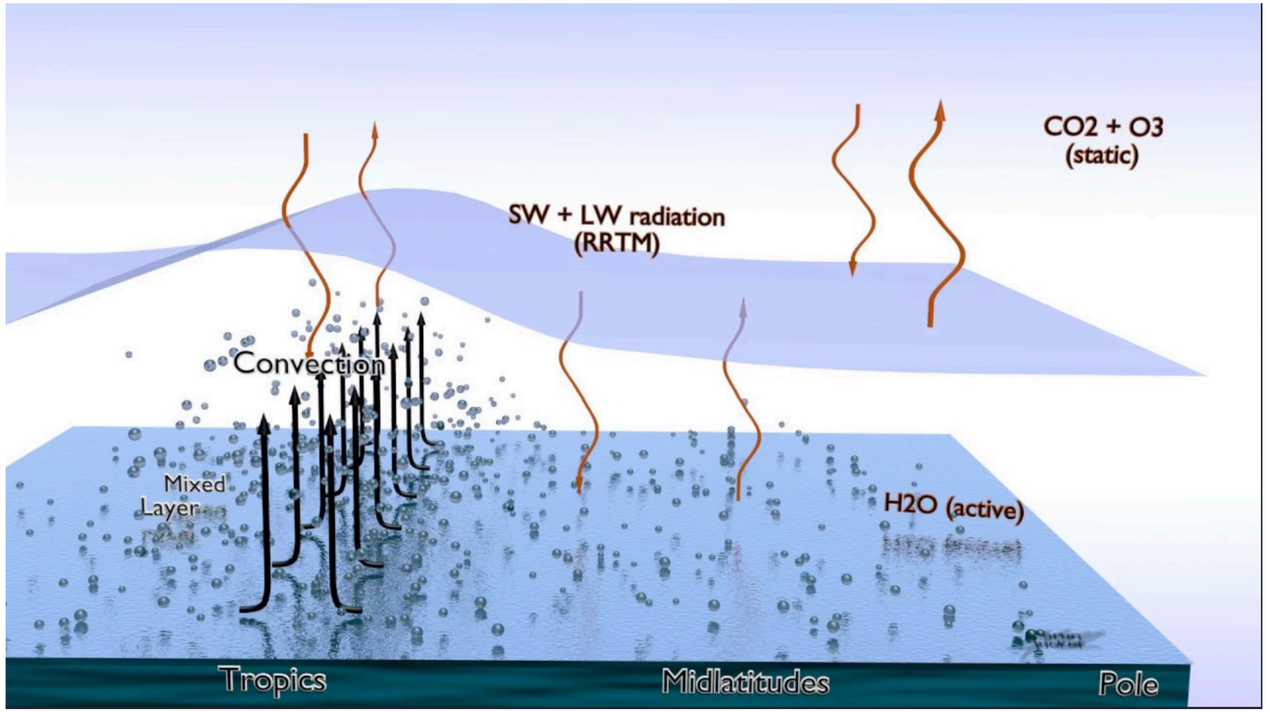
Refactoring



https://github.com/camfort/camfort/

Demo using MiMA as target

https://github.com/mjucker/MiMA/blob/master/src/atmos_param/moist_conv/moist_conv.f90



```
camfort alloc-check
```

Memory performance & safety:
All allocated arrays freed, no double free, or use after free

```
camfort fp-check
```

Numerical stability:
No equality (or inequality) on FP

```
camfort use-check
```

Tidy code:
No equality (or inequality) on FP

```
camfort array-check
```

Computational performance:
Column-major order traversal

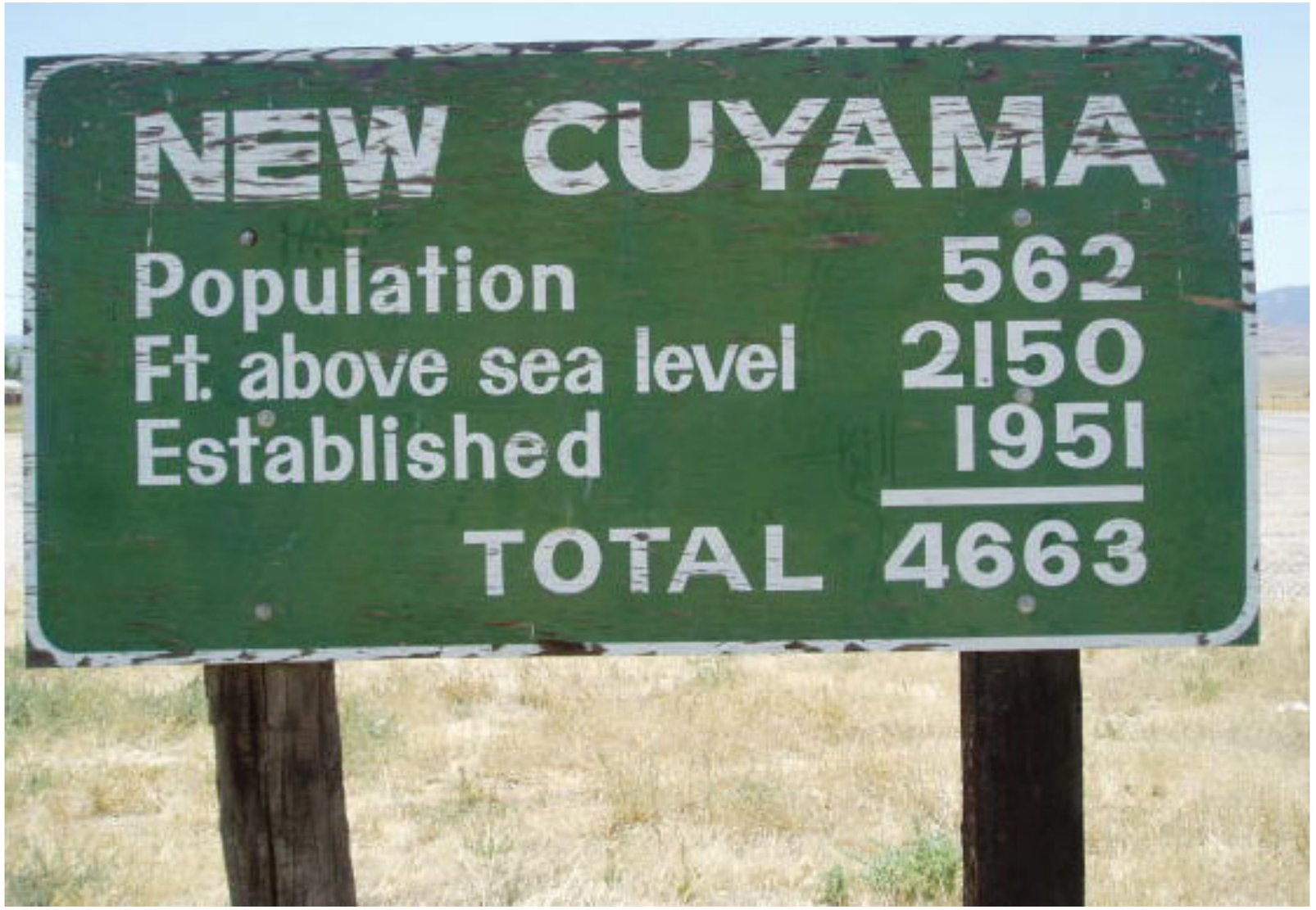


photo from Andrew Kennedy's website
http://research.microsoft.com/en-us/um/people/akenn/units/

Units-of-measure verification

```
1   program energy
2     real :: mass = 3.00, gravity = 9.91, height = 4.20
3     real :: potential_energy
4
5     potential_energy = mass * gravity * height
6   end program energy
```

Suggest

```
$ camfort units-suggest energy1.f90
Suggesting variables to annotate with unit specifications in 'energy1.f90'
…
energy1.f90: 3 variable declarations suggested to be given a
specification:
    energy1.f90 (2:43)    height
    energy1.f90 (2:14)    mass
    energy1.f90 (3:14)    potential_energy
```

Units-of-measure verification

```
1   program energy
2     != unit kg :: mass
3     != unit m  :: height
4     real :: mass = 3.00, gravity = 9.91, height = 4.20
5     != unit kg m**2/s**2 :: potential_energy
6     real :: potential_energy
7
8     potential_energy = mass * gravity * height
9   end program energy
```

Check

```
$ camfort units-check energy1.f90

energy1.f90: Consistent. 4 variables checked.
```

Units-of-measure verification

```
1   program energy
2     != unit kg :: mass
3     != unit m  :: height
4     real :: mass = 3.00, gravity = 9.91, height = 4.20
5     != unit kg m**2/s**2 :: potential_energy
6     real :: potential_energy
7
8     potential_energy = mass * gravity * height
9   end program energy
```

Synthesise

```
$ camfort units-synth energy1.f90 energy1.f90

Synthesising units for energy1.f90
```

Units-of-measure verification

```
1   program energy
2     != unit kg :: mass
3     != unit m  :: height
4     != unit m/s**2  :: gravity
5     real :: mass = 3.00, gravity = 9.91, height = 4.20
6     != unit kg m**2/s**2 :: potential_energy
7     real :: potential_energy
8
9     potential_energy = mass * gravity * height
10  end program energy
```

Synthesise

```
$ camfort units-synth energy1.f90 energy1.f90

Synthesising units for energy1.f90
```

Check

Does *it do what I think it does?*

Infer

What does it do?

Synthesise

Capture what it does for documentation & future-proofing

Suggest

Where should I add a specification to get the most information?