Peter Ukkonen (DMI, NBI)

Accelerating radiation computations in dynamical models by using neural networks (and optimized code)

Background

- Climate/weather prediction models need physics parameterizations
- Trend toward greater complexity (e.g. higher resolution)
- Larger simulations already consume lots of energy
- Accuracy and computational efficiency are linked

Radiation: important but expensive

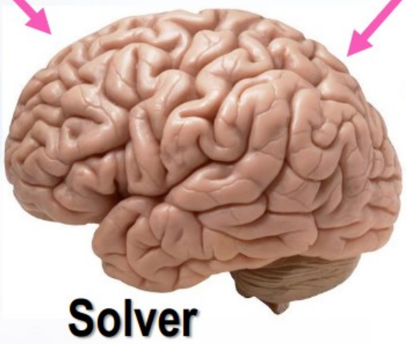
- Radiative transfer is well-understood, but expensive to solve accurately in large-scale models. Even fast radiation parameterizations can take ~50% of computational time in climate models

Conclusion: Radiation is a key bottleneck in predictive modeling.

Goal: Improve the accuracy/speed ratio of radiation codes by using neural networks and code optimization

The four components of a radiation scheme

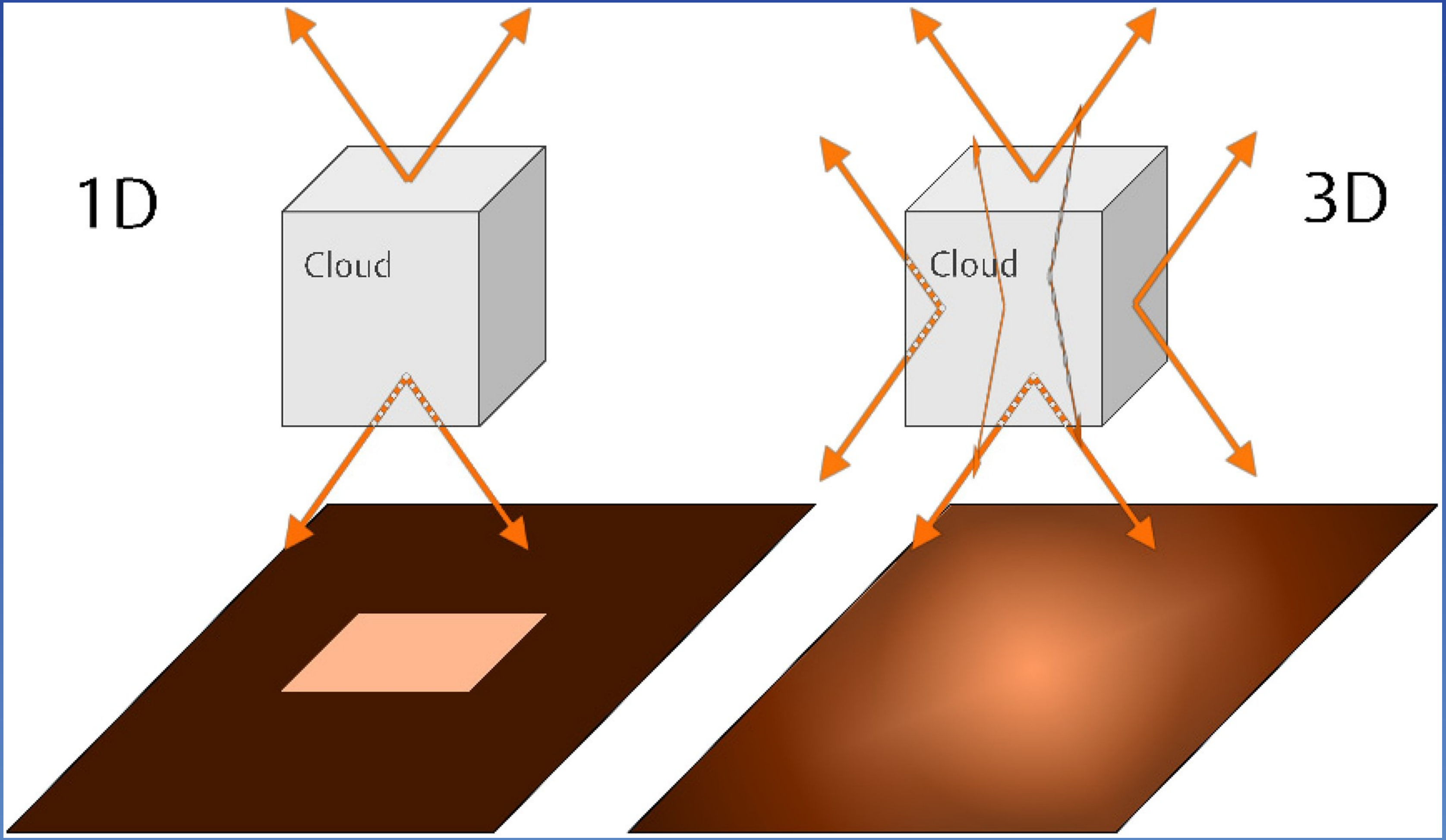
- *Determines spectral resolution*
- RRTM-G uses 252 spectral intervals



- *Determines how sophisticated interactions with clouds will be*

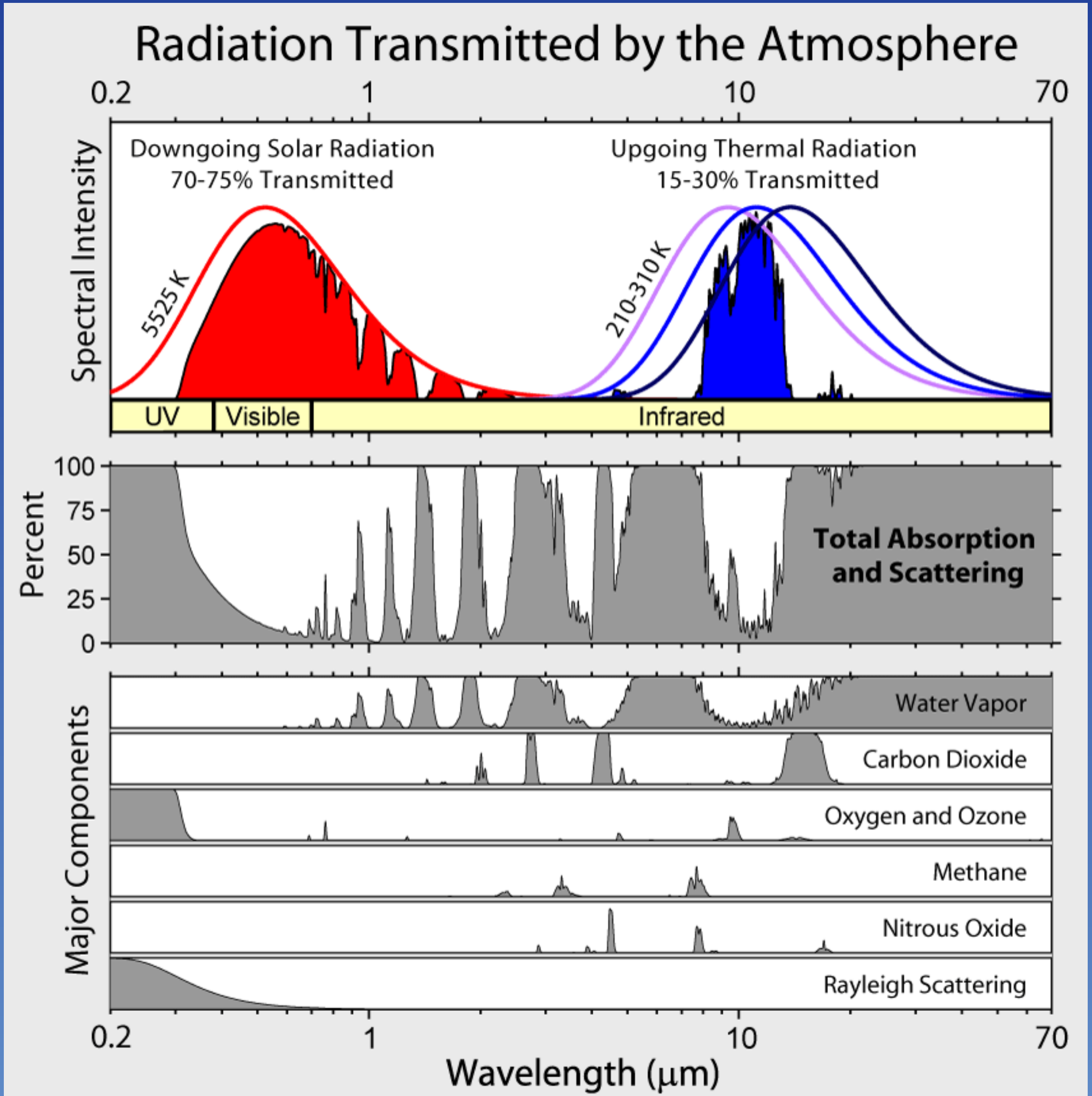
- Codes should be *modular*, allowing components to be changed independently

Solver: physical equations, but big assumptions



Gas optics

- Spectral complexity
- Changes in greenhouse gases particularly important for climate



The trick behind modern radiation codes: Correlated-k method

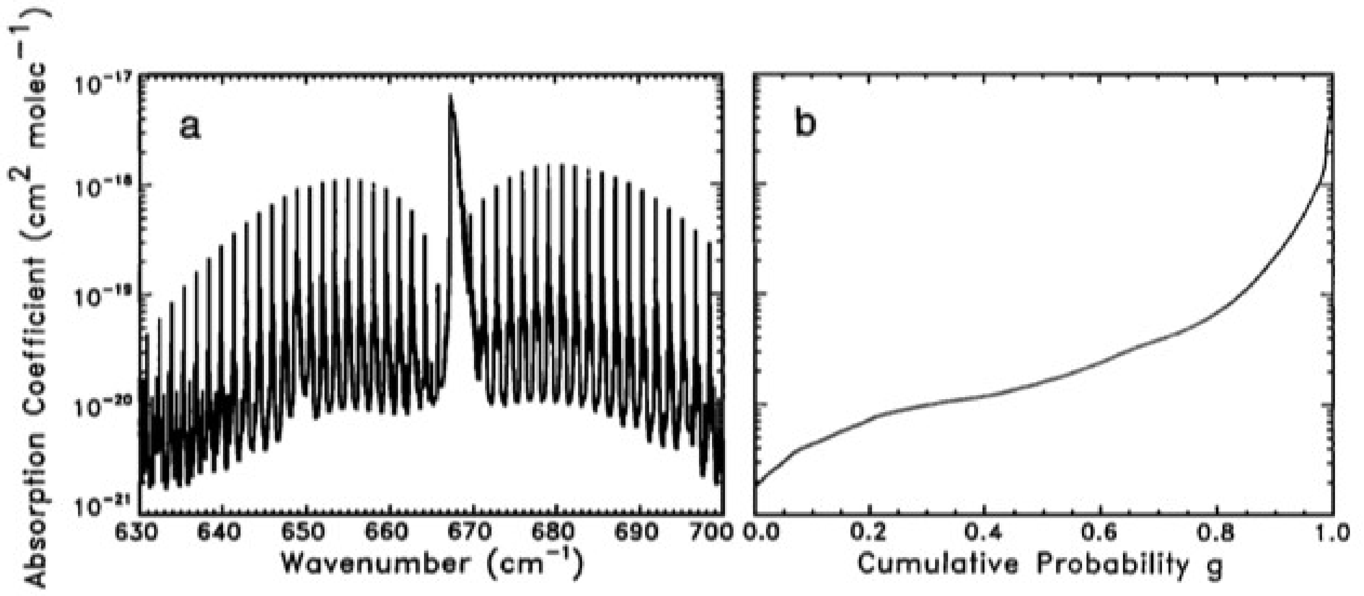


Figure 1. Absorption coefficients due to carbon dioxide for a layer (P = 507 mbar) in the midlatitude summer atmosphere for the spectral range 630–700 cm^{-1} (a) as a function of wavenumber and (b) after being rearranged in ascending order.

1) Rearrange absorption coefficients (k's) in ascending order ("k-distribution")
- $k(\nu) \rightarrow k(g)$

2) Break into sub-intervals ('g-points'), compute average k for each g for various mixing fractions, temperature, pressure
3) Store these k(g,T, p, m) in a look up table
4) Interpolate from this look up table to obtain k(g) at given T, p, gas mixing ratios.

Methods

- Use new radiation code RRTMGP to predict optical properties for a large number of atmospheric profiles
- RRTMGP takes high number of gases as input and has high spectral resolution (16 bands, 256 g-points)
- Train neural networks to predict the optical properties for a given atmospheric layer
- Plug neural network model back in the RRTMGP Fortran code
- Hope it’s faster and no less accurate

Preparing data

- Obtain profiles of atmospheric conditions and gas concentrations from:
  - Reanalyses
  - climate projections
  - Idealized profiles

- Sample present-day, pre-industrial, future, LGM..

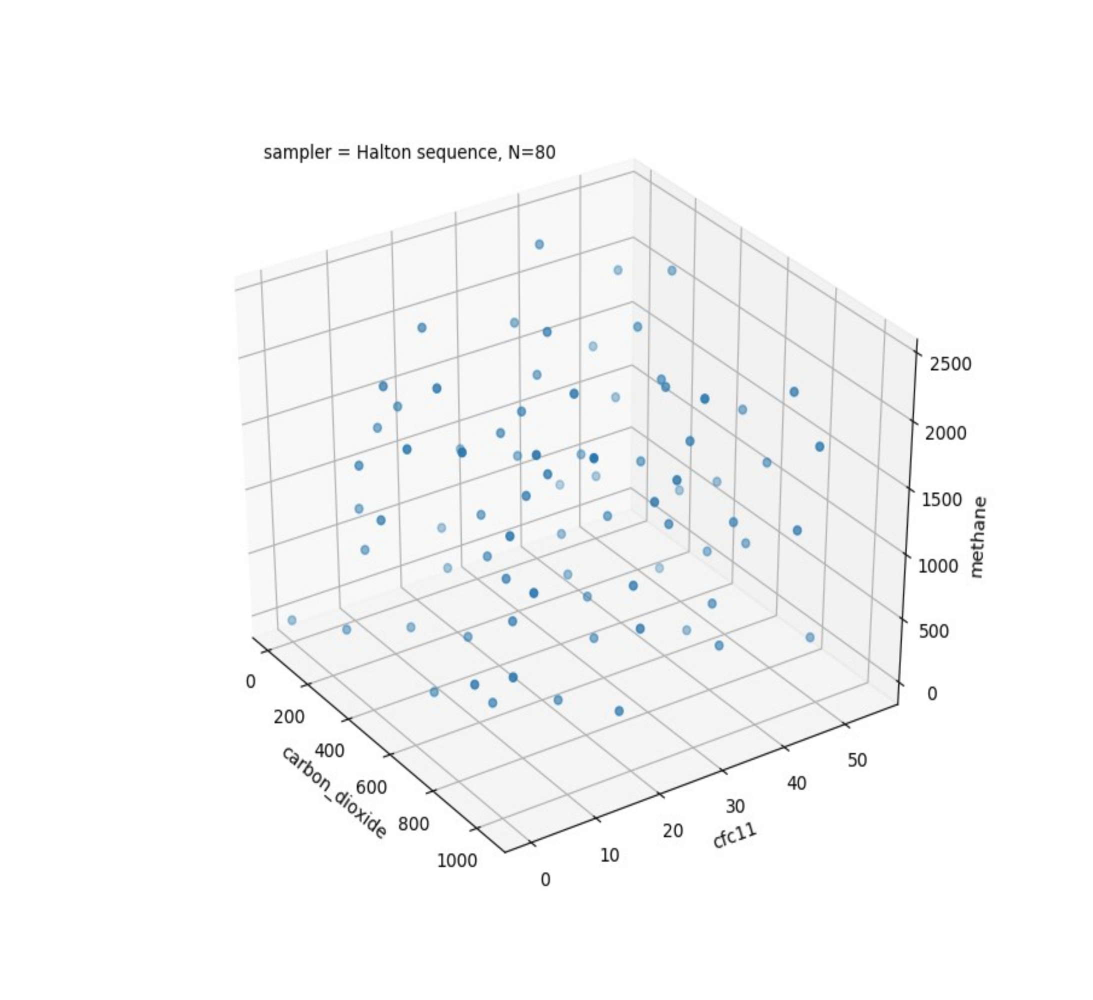
Preparing data

- Obtain profiles of atmospheric conditions and gas concentrations from:
  - Reanalyses
  - climate projections
  - Idealized profiles

- Sample present-day, pre-industrial, future, LGM..

- More data: hypercube sampling of gases, +- temperature with constant RH, etc.

  → 7 million training samples



Why it's efficient

Original code:

- For each layer $j = 1...J$ in each column $k = 1...K$
  - For band $b = 1...B$
    - Compute the g-point vector $\overline{\tau, maj}_{b,j,k}$ by 3D linear interpolation in T, p and η.
    - For each minor gas
      - Compute τ_{min} by 2D linear interpolation in temperature and η
    - $\tau_{i,j,g} = \tau_{maj} + \tau_{min}$

Neural network:

- Predicts all NGPT spectral points (256) and NGAS gas contributions simultaneously $\mathbf{Y_{NGPT} = f (X_{Ngas})}$, where Y and X are vectors, and f() is modelled by the neural net.
- Possible to further collapse layers J and columns K into M=J*K and obtain batch predictions $\mathbf{Y_{NGPT,\ M}}$ where Y is now a matrix.
- The core computations are then **matrix-matrix multiplications** which we can delegate to a optimized library (GEMM calls to a BLAS library such as MKL)

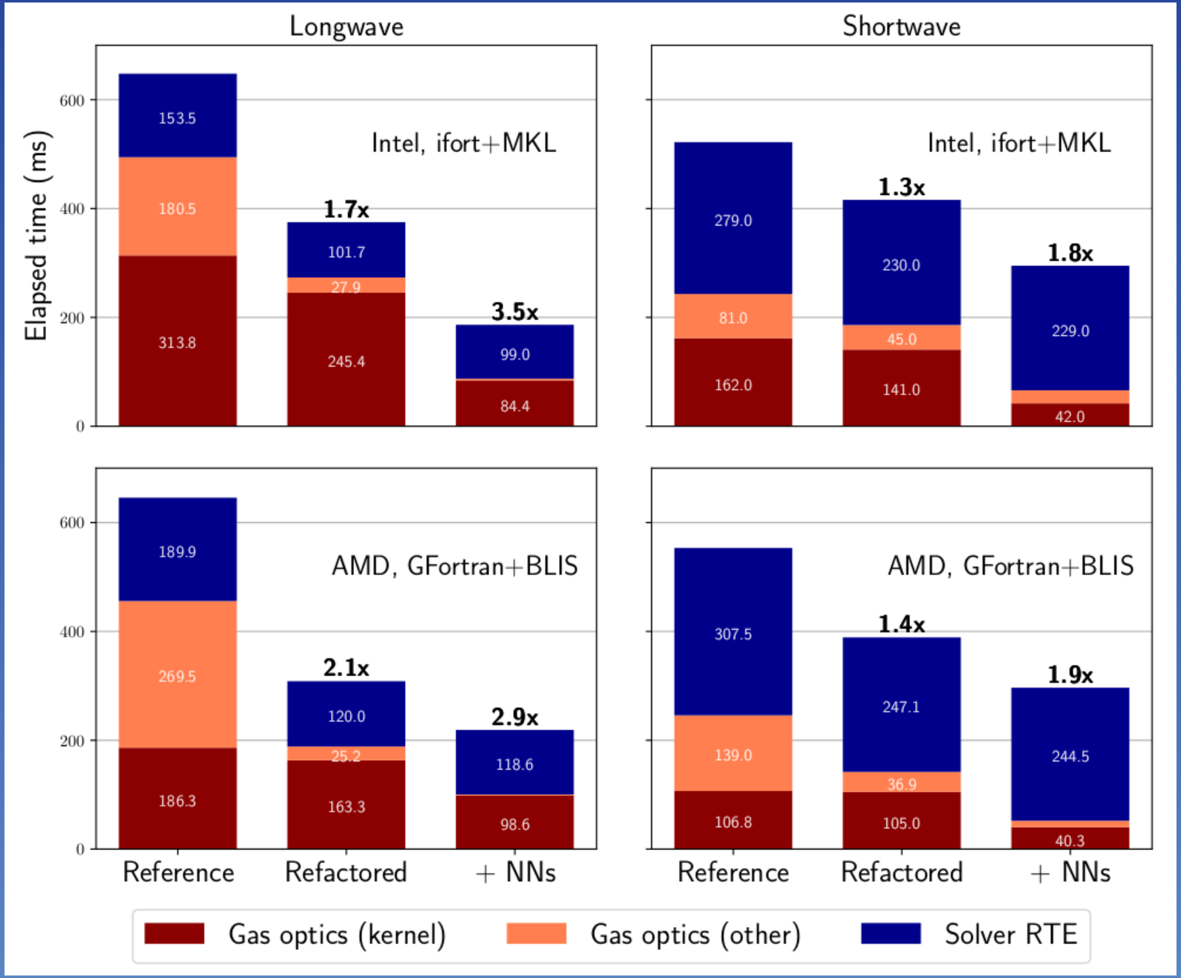
Code refactoring
(boring to some, but climate models need fast code)

- Order of dimensions changed form (col, lay, g-point) to (g-point, lay, col)
  → fluxes can be computed inside a column loop, reducing memory use

- Further changes to improve vectorization and reduce memory by inlining computations and combining loops

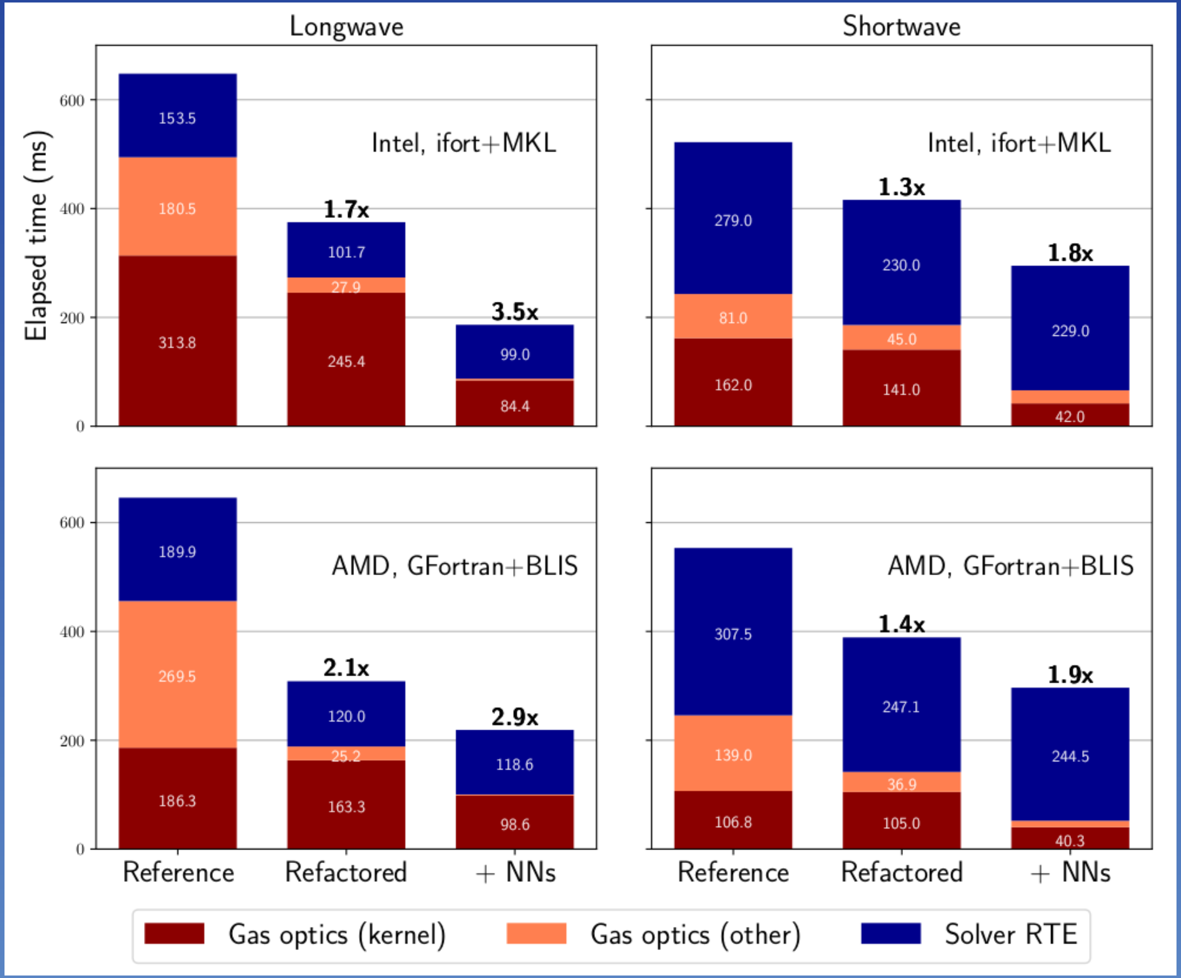
- Inefficiency often stems from the processor having to wait for slow memory accesses, not being exposed to parallelism, or both

Speed-up

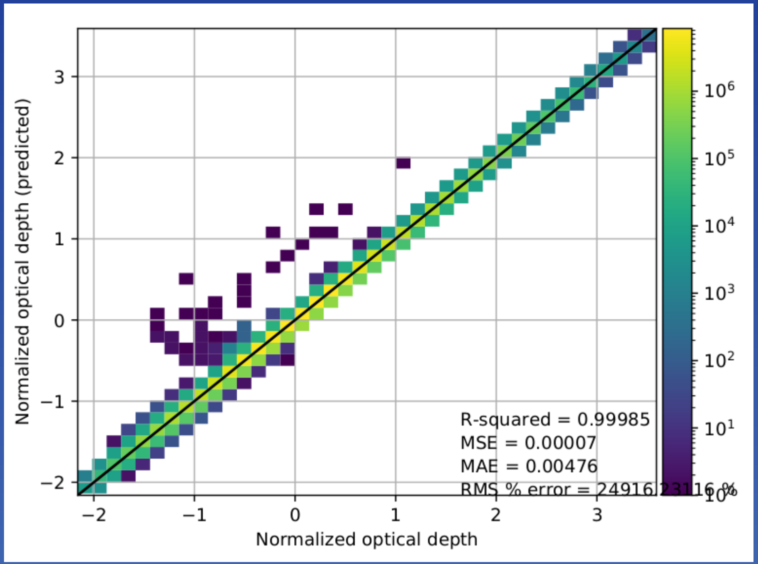


Longwave
Shortwave
Elapsed time (ms)
0
200
400
600
Intel, ifort+MKL
AMD, GFortran+BLIS
153.5
180.5
313.8
1.7x
101.7
27.9
245.4
3.5x
99.0
84.4
279.0
81.0
162.0
1.3x
230.0
45.0
141.0
1.8x
229.0
42.0
189.9
269.5
186.3
2.1x
120.0
25.2
163.3
2.9x
118.6
98.6
307.5
139.0
106.8
1.4x
247.1
36.9
105.0
1.9x
244.5
40.3
Reference
Refactored
+ NNs
Gas optics (kernel)
Gas optics (other)
Solver RTE

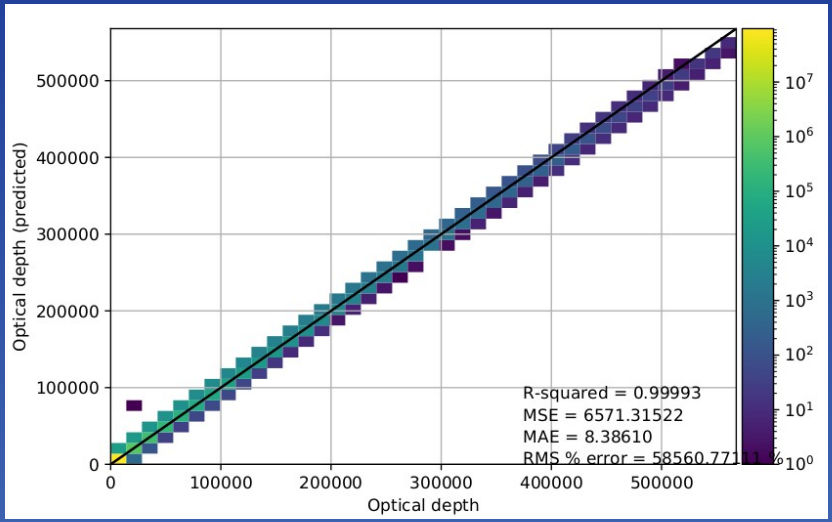
Speed-up

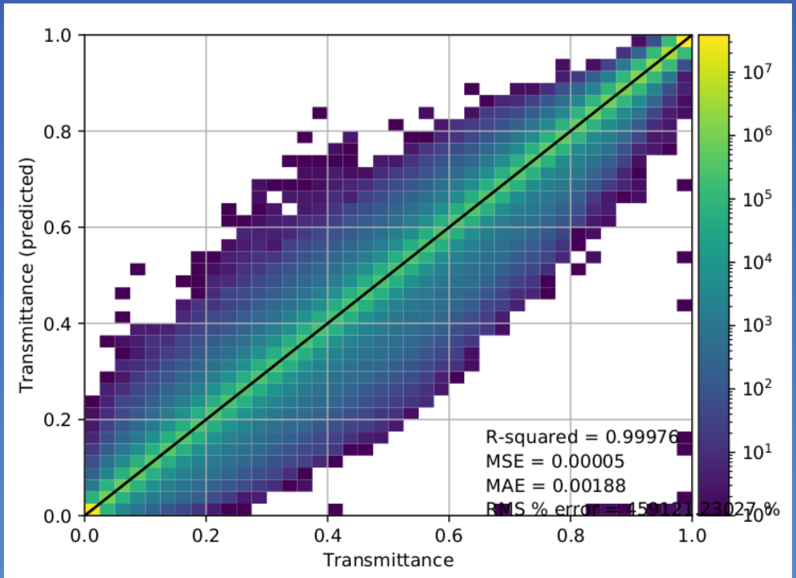


- The neural network actually does 4x more floating points operations...
- But with much better efficiency (7-8 times more operations per second)



$y \longrightarrow \tau$





Transmittance
T = exp(-τ)

Solver (up and downward radiative transfer through 1D atmosphere)

--> Broadband fluxes u,v (W/m2)

--> Heating rates (K/day)

Accuracy

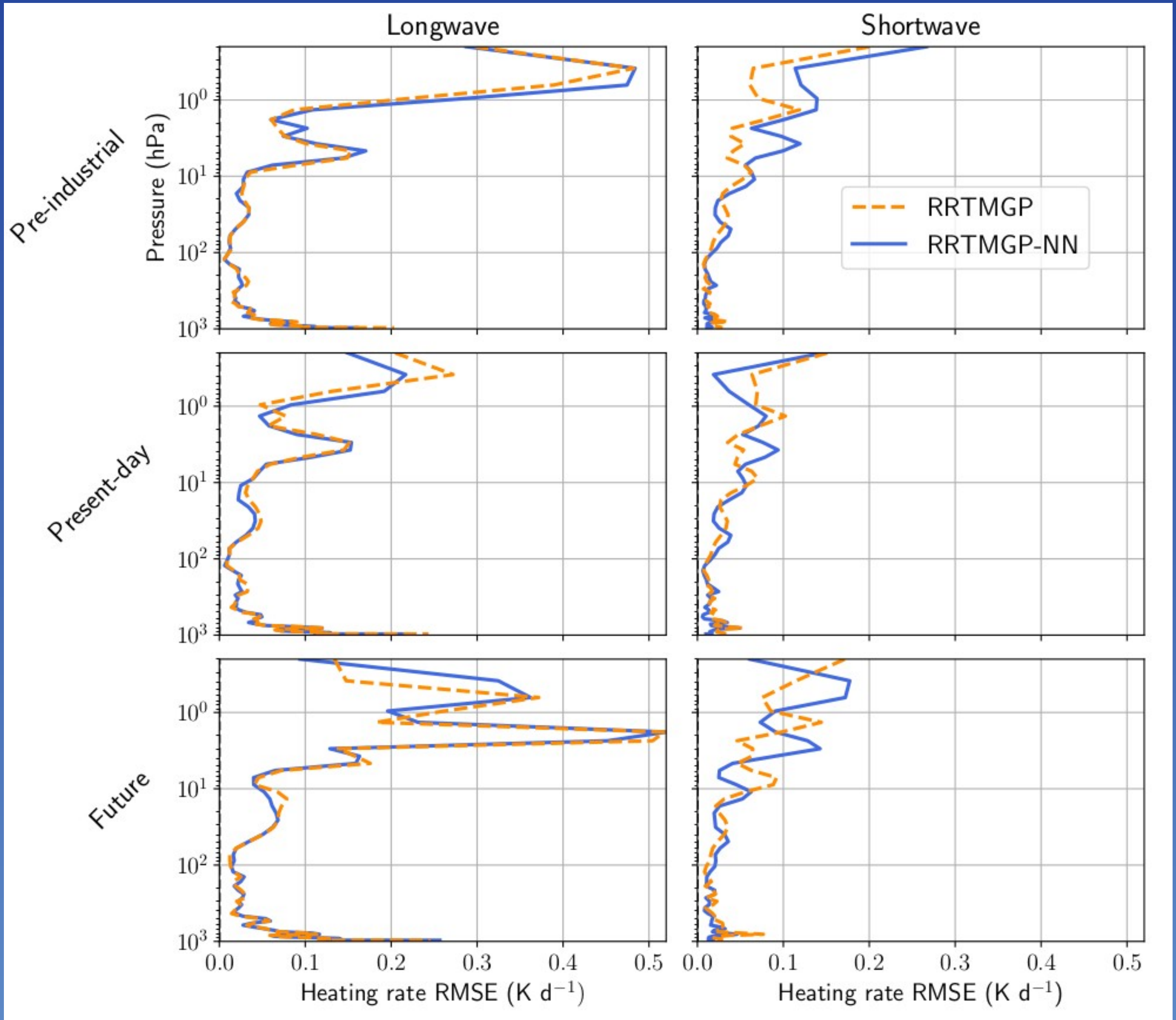
- Accuracy was evaluated using benchmark line-by-line computations on independent data

- Errors in fluxes, heating rates and radiative forcings were all very similar to RRTMGP

Accuracy

• Accuracy was evaluated using benchmark line-by-line computations on independent data

• Errors in fluxes, heating rates and radiative forcings were all very similar to RRTMGP

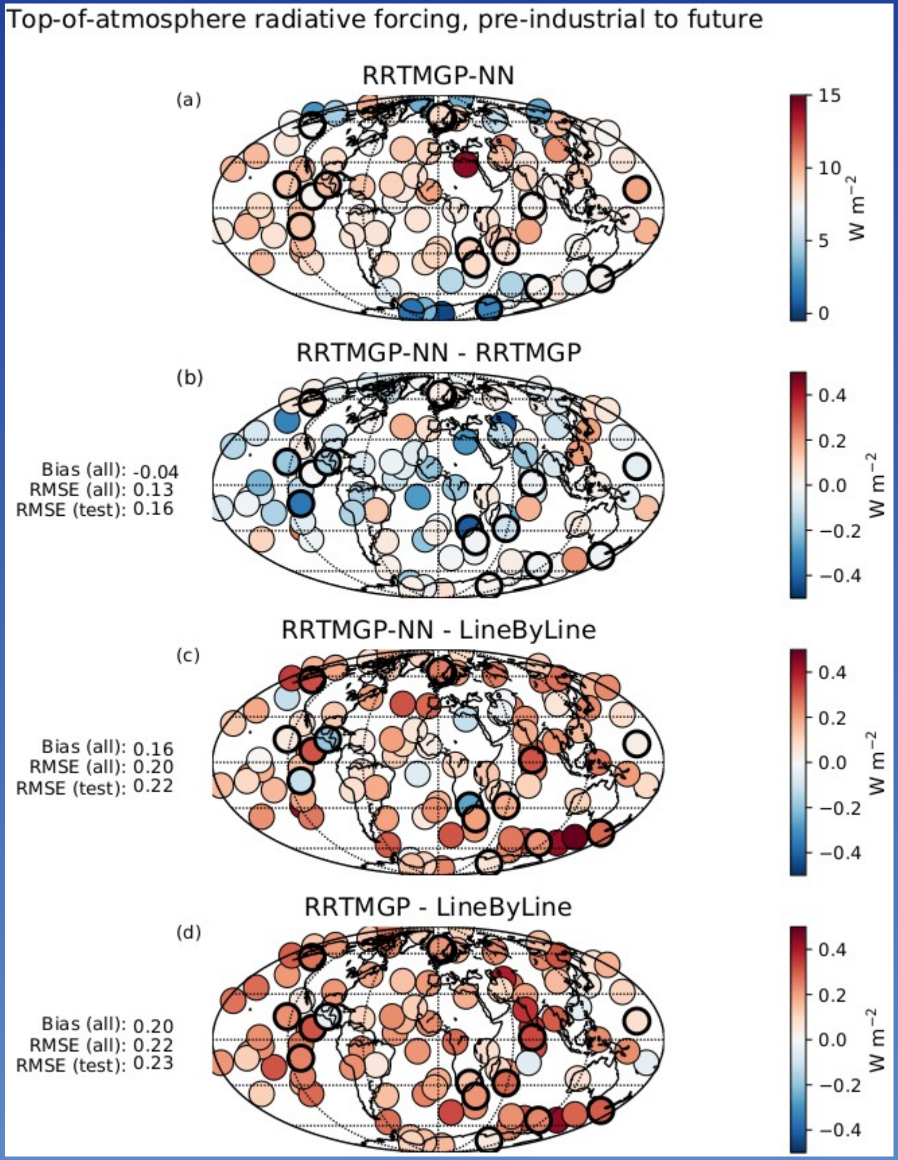


• Accuracy was evaluated using benchmark line-by-line computations on independent data

• Errors in fluxes, heating rates and radiative forcings were all very similar to RRTMGP

NN errors

REF errors



! Here neural networks were used to accelerate an existing scheme, but they can also be used to develop new, more realistic/accurate physics schemes by training with high-resolution models, or even observations !

Line-by-line modeling

LBLRTM

Cut out the middle man?

RRTMGP

Neural network

Parameterizations

Application

Fluxes and Heating Rates for climate and weather models

Conclusion

- A new radiation scheme was accelerated by 2-3X (note: clear-sky flux computations) by using targeted machine learning and code optimization

- The accuracy is virtually identical to the original scheme

Conclusion

- A new radiation scheme was accelerated by 2-3X (note: clear-sky flux computations) by using targeted machine learning and code optimization

- The accuracy is virtually identical to the original scheme

- Why use ML and particularly neural networks
    - Scientific advantages (better physics?):

      Flexible non-linear data fitting tools, can be used to model arbitrary relationships IF appropriate training data is available and skip the need for overly simplistic assumptions and ad-hoc equations
    - Computational advantages (faster physics):

      High performance, portable code, future-proof (fast on GPUs, etc)