Facilitating science analyses with uncertainty-aware emulation

Jouni Susiluoto[1], Amy Braverman[1], Ziad Haddad[1], Otto Lamminpaa[1], Houman Owhadi[2], Sai Prasanth[1], and others

[1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA
[2]California Institute of Technology, Pasadena, CA 91125, USA

ESTF, June 22, 2023

Jet Propulsion Laboratory
California Institute of Technology

Here are some examples of the types of problems that we try to help address

Assume you have a model that you use to answer tricky questions, such as:

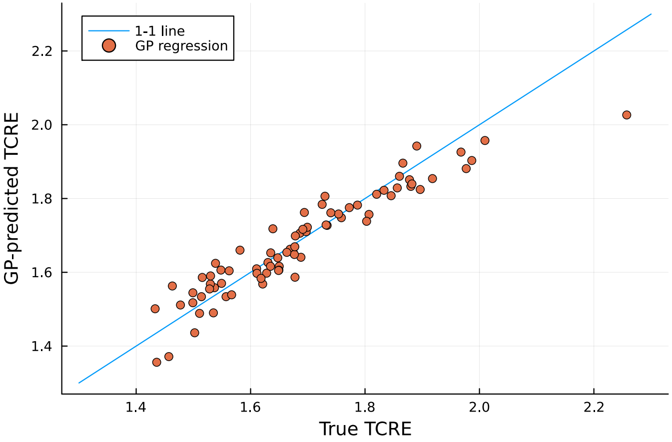
- How sensitive is climate to carbon emissions?
- How much carbon do we have in the atmosphere?
- Where do we see water stress effects on Earth right now?
- What data should we collect in order to maximize impact of a remote sensing mission?

- **Emulators** are fast approximations of computationally more expensive models.
  - However, we specifically also consider arbitrary input-output relations in what we do (data-driven model construction).
- Accurate and fast model emulation can help answer the questions above.
- Emulators are particularly useful in the context of inverse problems (remote sensing retrievals, Bayesian analysis, etc.).

Example 1: Climate sensitivity

Emulation of **Transient Climate Response to Cumulative Carbon Emissions (TCRE)** with **data from the University of Victoria Earth System Climate Model (UVICESCM)**. Based on work by Antti-Ilari Partanen / Carla di Natale at Finnish Meteorological Institute.

- Emulating 20 inputs to one output ($\mathbb{R}^{20} \to \mathbb{R}$)
- Training data size: 200 simulations
- Testing data size: 77 simulations
- Sub-optimal training data design
- Data generation took a full month on Europe's fastest supercomputer.
- Emulation results (including writing the script) were produced over a coffee break.



How should such an emulator look like?

Must-haves:

- It must be sufficiently accurate

How should such an emulator look like?

Must-haves:

- It must be sufficiently accurate
- It must be able to model complex non-linear phenomena

How should such an emulator look like?

Must-haves:

- It must be sufficiently accurate
- It must be able to model complex non-linear phenomena
- It must be fast

How should such an emulator look like?

Must-haves:

- It must be sufficiently accurate
- It must be able to model complex non-linear phenomena
- It must be fast

Nice-to-haves:

- Training should be fast, robust, and easy

How should such an emulator look like?

Must-haves:

- It must be sufficiently accurate
- It must be able to model complex non-linear phenomena
- It must be fast

Nice-to-haves:

- Training should be fast, robust, and easy
- The emulators should be portable and easy to use

How should such an emulator look like?

Must-haves:

- It must be sufficiently accurate
- It must be able to model complex non-linear phenomena
- It must be fast

Nice-to-haves:

- Training should be fast, robust, and easy
- The emulators should be portable and easy to use
- Training and running the emulator should not require specialized hardware

How should such an emulator look like?

Must-haves:

- It must be sufficiently accurate
- It must be able to model complex non-linear phenomena
- It must be fast

Nice-to-haves:

- Training should be fast, robust, and easy
- The emulators should be portable and easy to use
- Training and running the emulator should not require specialized hardware
- The software should be open source.

How **does** our emulator look like?

Must-haves:

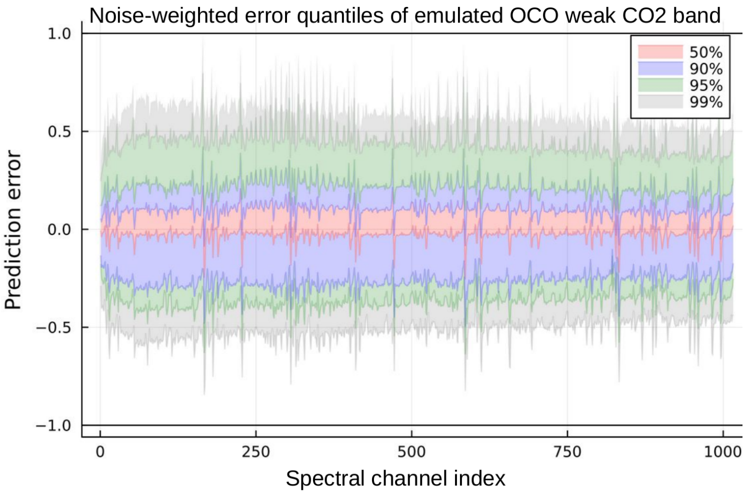
- It **is** accurate
- It **is** able to model complex non-linear phenomena
- It **is** fast

Nice-to-haves:

- Training **is** fast, robust, and easy
- The emulators **are close to being** portable and easy to use
- Training and running the emulator **does not** require specialized hardware
- The software **is** open source.

Example 2: Orbiting Carbon Obsevatory

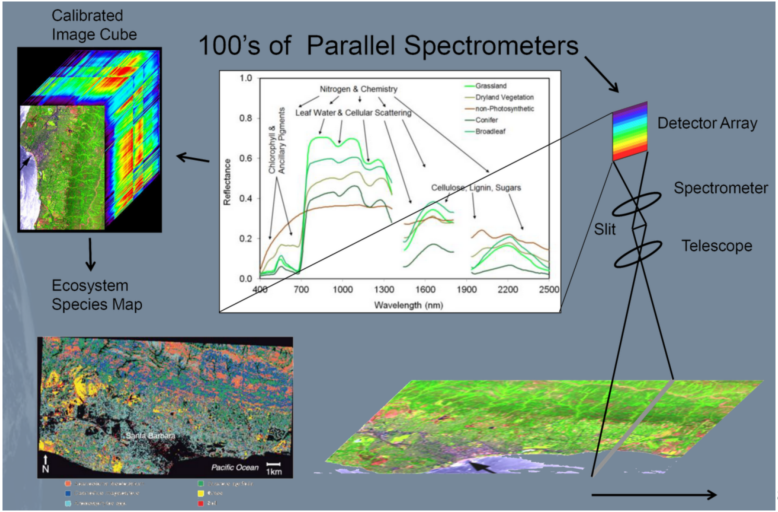
OCO-2/3 Level 2 retrievals require running a radiative transfer model. Here we emulate one of three quantities, the weak CO2 band. The target is to be under instrument noise level (±1 on the y-axis). This is true also for the two other quantities (O2 and strong CO2 bands). A two-level emulator was used to obtain these results.



Example 3: Imaging specroscopy (1/2)

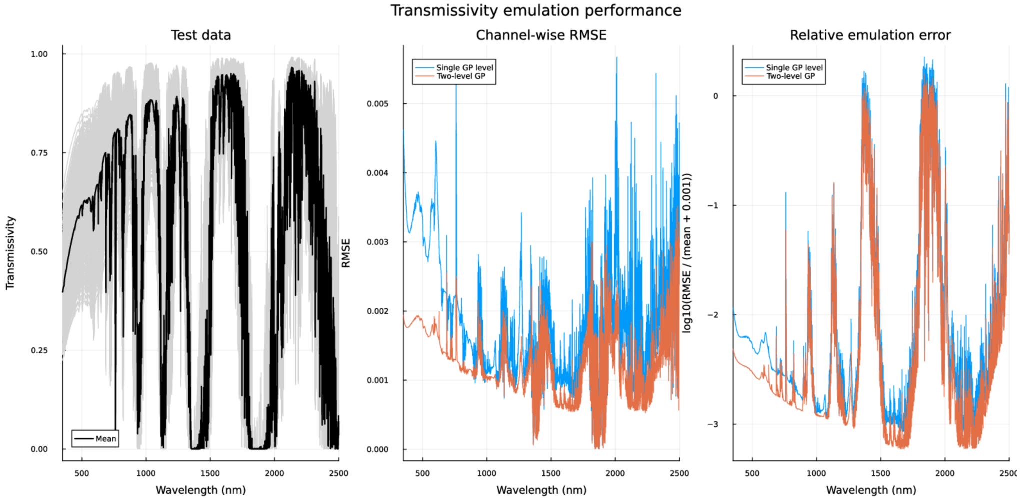
- Instruments: SBG, EMIT, AVIRIS-NG, PRISMA, EnMAP, etc.
- Instruments retrieve spectral surface reflectance
- Hundreds of spectral bands
- Technology allows observing a wider range of phenomena ranging from carbon emissions to minerology, water stress, algae, snow properties, vegetation properties / state, etc.

Imaging spectroscopy example from the AVIRIS-NG website, https://aviris-ng.jpl.nasa.gov/aviris-ng.html



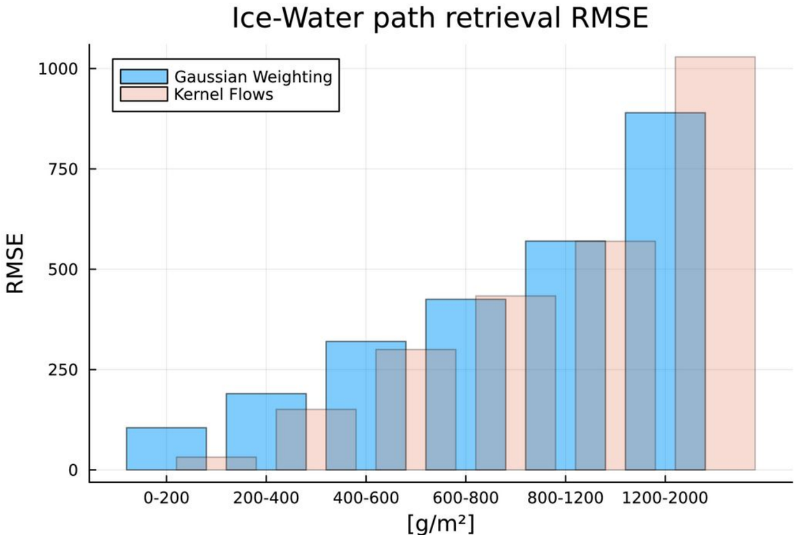
Example 3: Imaging specroscopy (2/2)

- Inferring surface reflectance requires simulating radiative transfer quantities multiple times
- Space-based instruments measure tens of thousands of spectra per second



Examples of other emulator use cases that we have considered:

- On the right: Ice-water path retrievals and convective storm now-casting.
  - Improvement over operational GPROF algorithm
  - Data is very noisy
  - Low-end improvement is physically important
- Other applications:
  - Mission design via simulation studies
  - Other radiative transfer applications (snow etc.)
  - Spatio-temporal inference
  - Reliability analysis and rare events



What is under the hood?

Construction

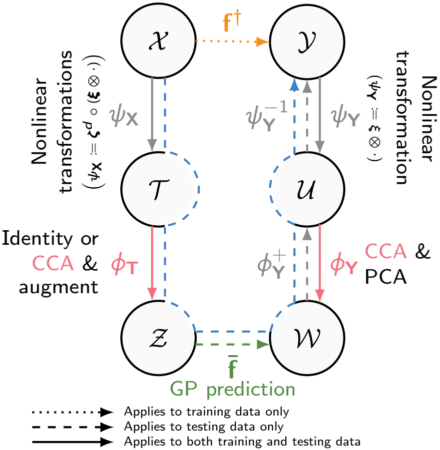
- The underlying technology is a non-standard multivariate Gaussian process model
- Heavily modified parametric Kernel Flows algorithm (Owhadi et al. 2019)
- Model training is based on cross validation
- Non-linear transformations of inputs
- Input and output dimension reduction

Performance

- Training takes 1–3 minutes on a laptop
- Single-threaded prediction is around 1 ms
- Accuracy is 1-2 orders of magnitude better than with vanilla off-the-shelf GP models
- Implementation languages: Julia for training; prediction straightforward to implement in any languate

Schema

$$\mathbf{f}^{\dagger} \approx \mathbf{f} := \psi_{\mathbf{X}} \circ \phi_{\mathbf{T}} \circ \bar{\mathbf{f}} \circ \phi_{\mathbf{Y}}^{+} \circ \psi_{\mathbf{Y}}^{-1}$$



Questions / comments

- No public reference yet; manuscript in prep. Description of the standard Kernel Flows algorithm is available in the canonical reference:

  H. Owhadi and G. R. Yoo. *Kernel flows: From learning kernels from data into the abyss.* Journal of Computational Physics, 389:22-47, 2019.
- Open source code will be available later this summer.
- For further info, please just get in touch at jouni.i.susiluoto@jpl.nasa.gov