Climate Model Development and the Role of Machine Learning

Daniel Williamson, Wenzhe Xu, Bertrand Nortier

University of Exeter, UK.

Machine Learning is not Magic

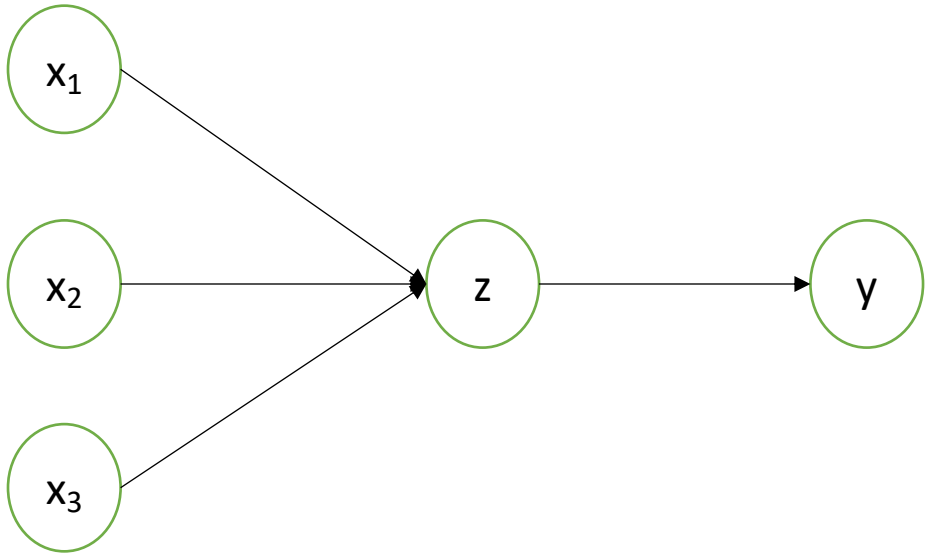
- What is now called “machine learning” is really just “statistics” with modern computation.
- Statistical models can:
  - Unlock patterns in data ✓
  - Predict a short time into the future based on the past ✓
  - Enrich our physical understanding of the world ✓
  - Replace all of physics ✗

Statistical (machine learning) models

$$y \sim \mathrm{Ber}(z(x))$$

$$z(x) = \phi\left(w_0 + \sum_{i=1}^{N_1} w_i x_i\right)$$

$$w_i \sim \mathrm{N}(0, \sigma^2)$$



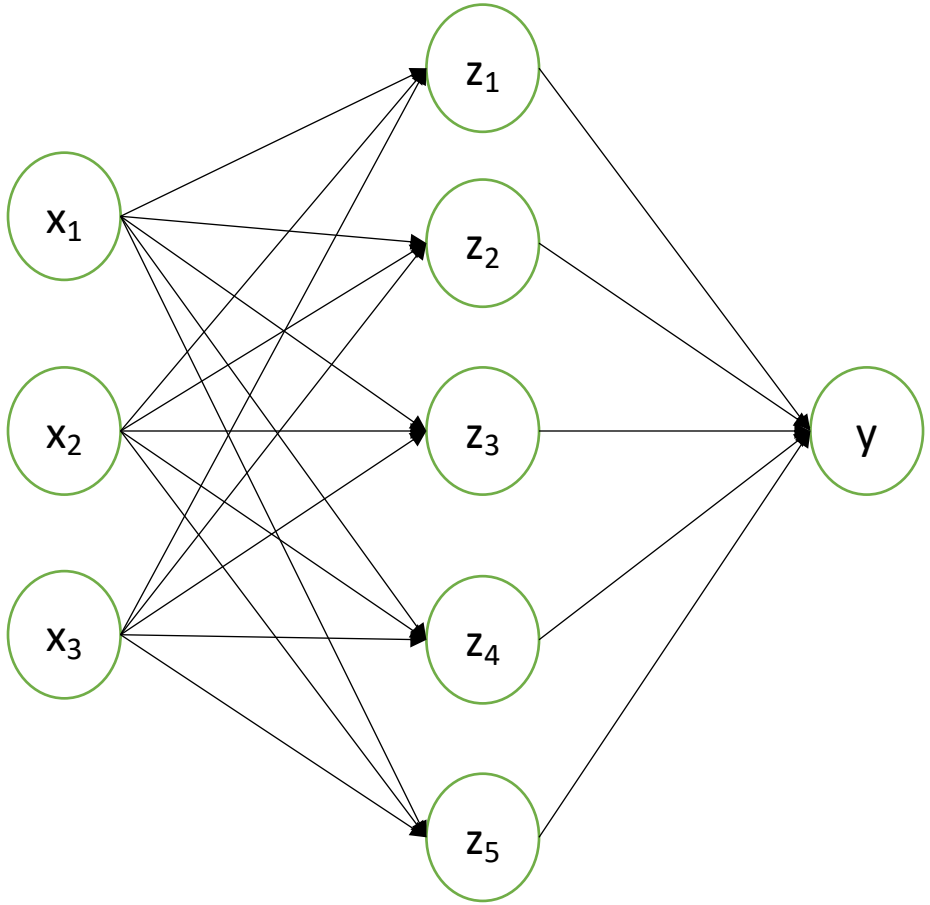
(Bayesian) Logistic Regression

Statistical (machine learning) models

$$y = w_{00} + \sum_{i=1}^{N_1} w_{0i} z_i(x)$$

$$z_i(x) = \phi\left(w_{i0} + \sum_{i=1}^{N_x} w_{ij} x_j\right)$$

$$w_{ij} \sim \mathrm{N}(0, \sigma^2)$$



Single layer (Bayesian) Neural Network

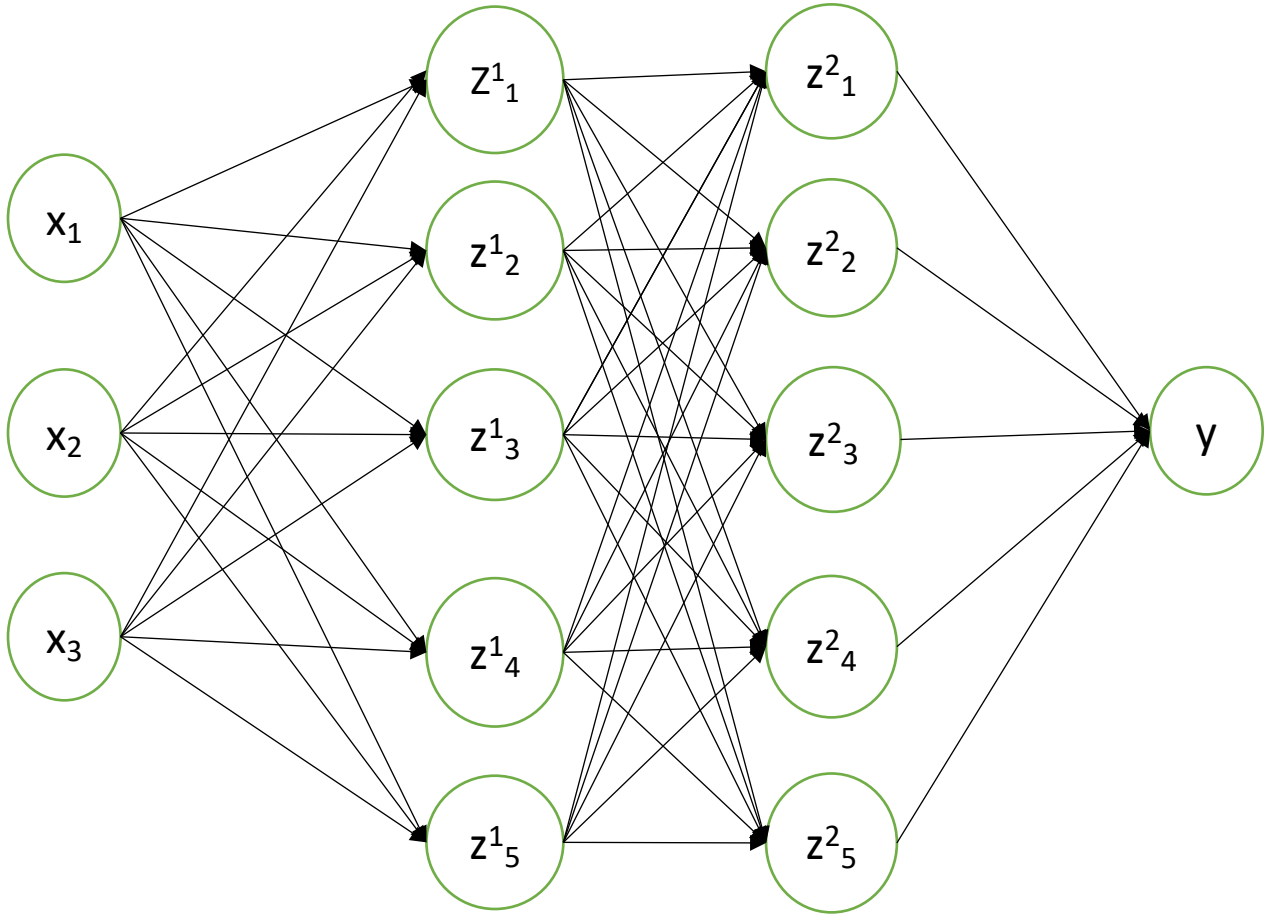
Statistical (machine learning) models

$$y = w_{00} + \sum_{i=1}^{N_2} w_{0i} z_i^2(x)$$

$$z_i^2(x) = \phi\left(w_{i0}^2 + \sum_{i=1}^{N_1} w_{ij}^2 z_j^1 \right)$$

$$z_i^1(x) = \phi\left(w_{i0}^1 + \sum_{i=1}^{N_x} w_{ij}^1 x_j \right)$$

$$w \sim \mathrm{N}(0, \sigma^2)$$



Deep (two layer) (Bayesian) Neural Network

Statistical (machine learning) models

For any number of layers:

$$N_i \to \infty \implies$$

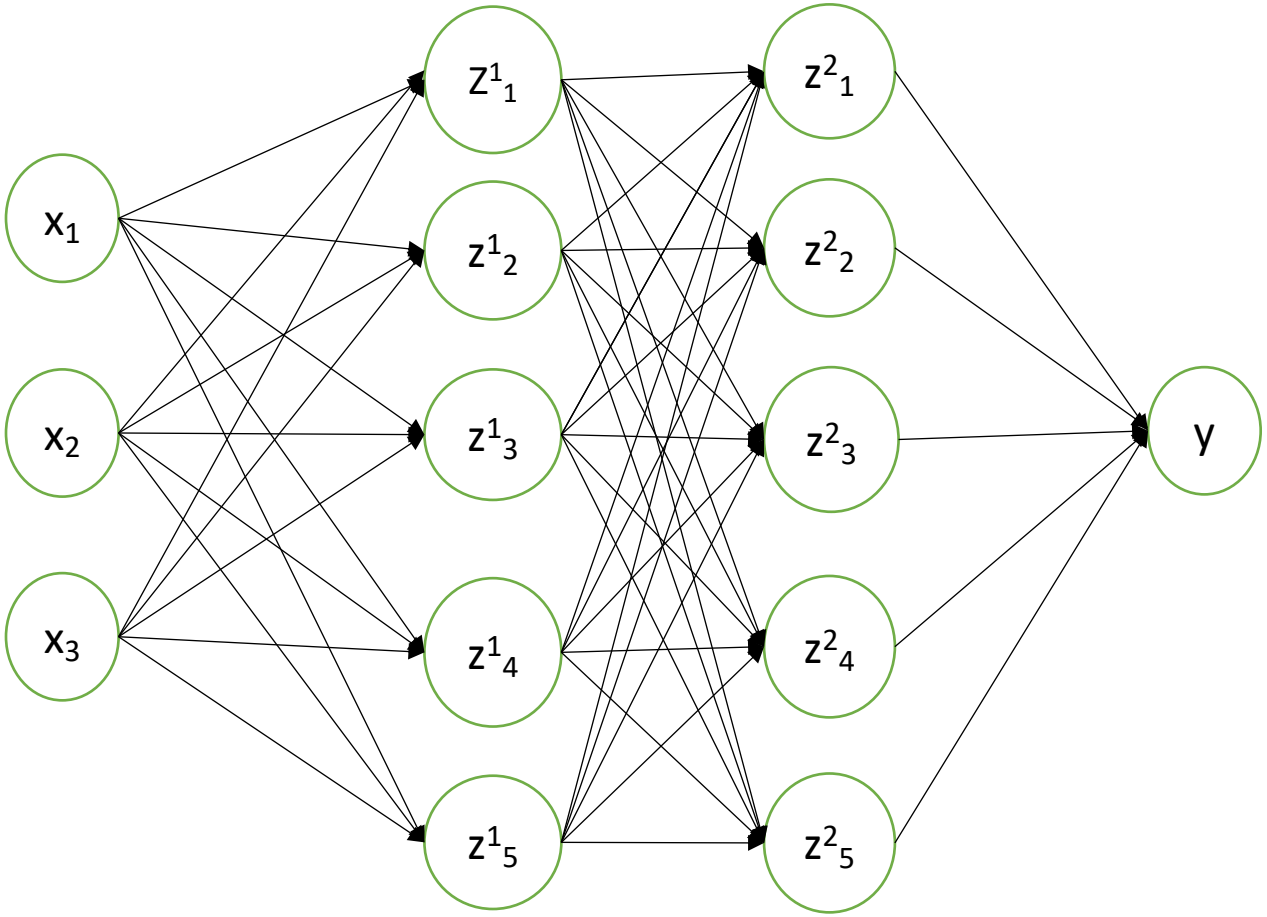
$$y(x) \sim \mathrm{GP}(0, K^l(x, x'))$$

Where, for any finite collection $x_1, \ldots, x_n$

$$y(x_1), \ldots, y(x_n) \sim \mathrm{MVN}(0, \Sigma)$$

$$\Sigma_{ij} = K^l(x_1, x_n)$$

Gaussian Process

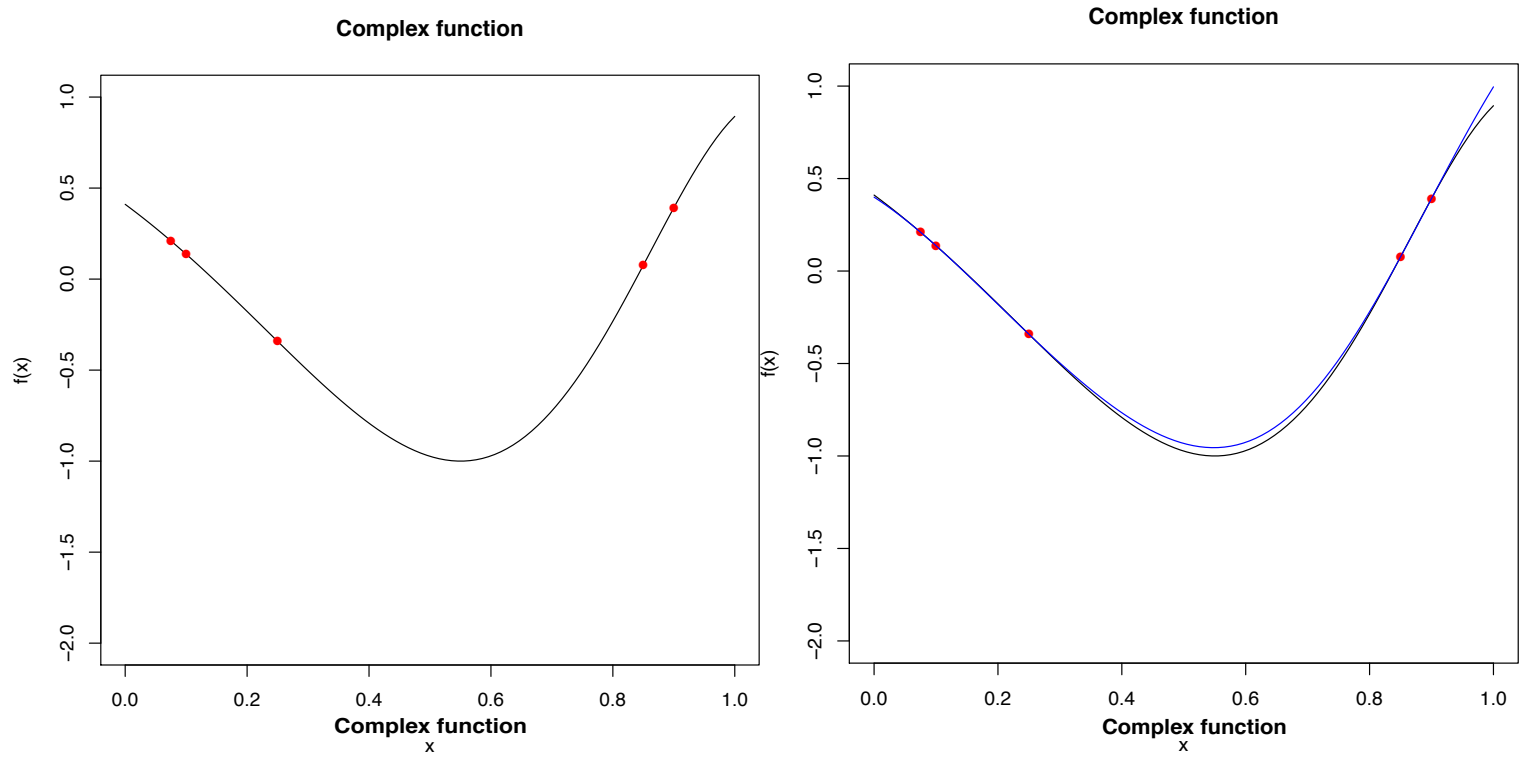


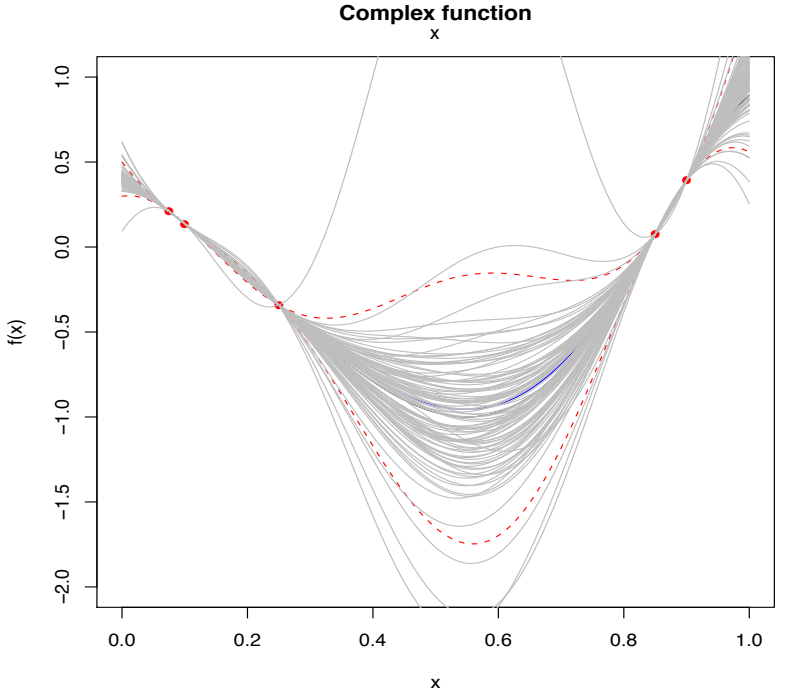
Harnessing Machine Learning

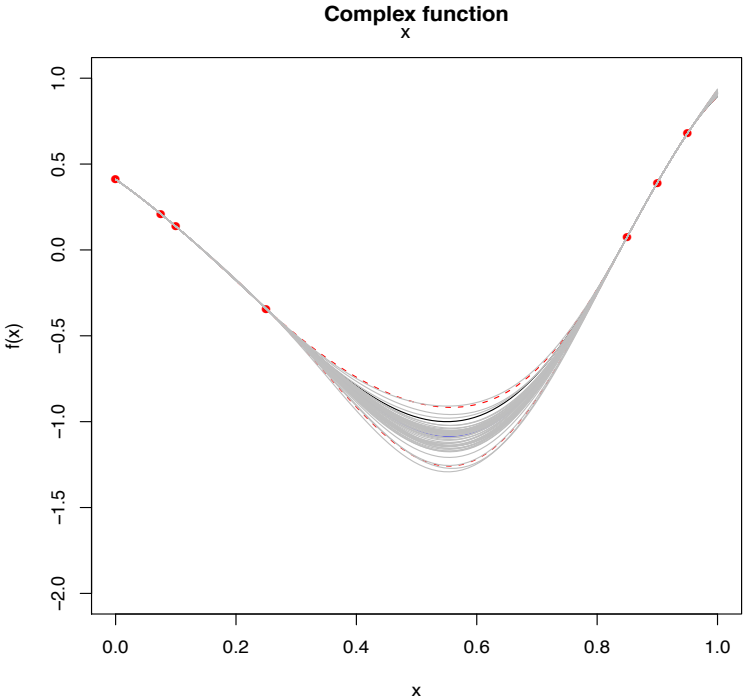
- Climate model parameterisations are complex functions of parameters, y(x).
- When there are many parameters, x, and solving y(x) is not very fast, we can use machine learning to learn y(x).
- I call this “building an emulator” (it’s also called surrogate modelling).
- I prefer Gaussian processes for emulation, but many statistical models have been tried.

Harnessing Machine Learning

- The GP acts as a prior over the parameterisation.
- As we run it (red dots), we learn more about it.
- We can sample from the function easily and calculate it's mean and variance directly.







Harnessing Machine Learning

- We have a true process, **y**, imperfect observations **z**, and parameterisation **f(x)**.
- To avoid inappropriate[1] optimisation of **f(x)**, I prefer the technique of **History Matching**.
- Use a small training set to fit an emulator, cut all regions of parameter space that are "too far" from the observations: $||\mathbf{z} - \mathbf{f}(\mathbf{x}_0)||_h > T$
- Run a new training set in the smaller parameter space and repeat.
- The norm $||.||_h$ is the *implausibility:* E.g.

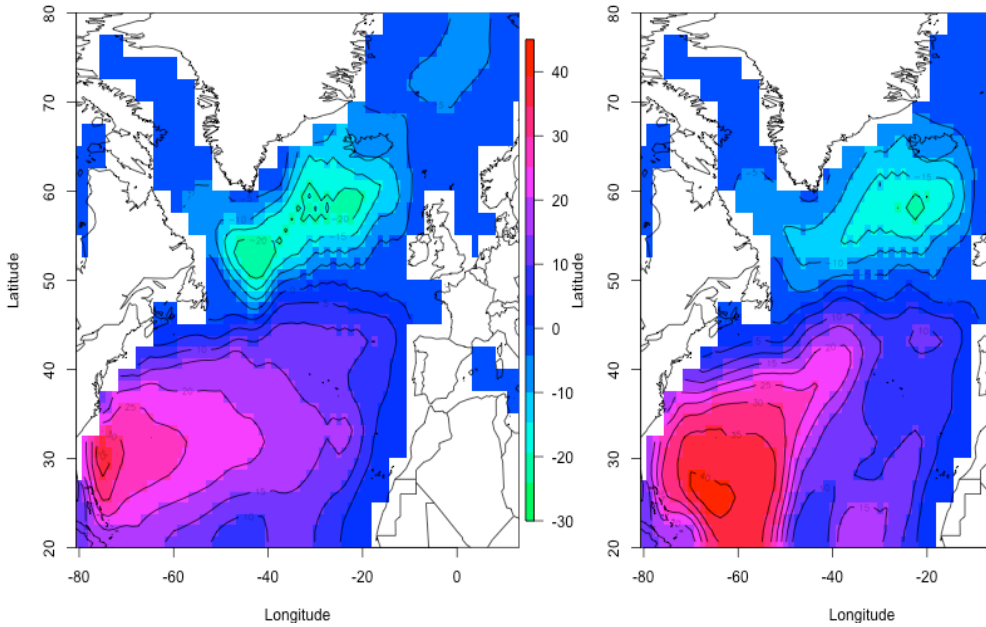
$$||\mathbf{z} - \mathbf{f}(\mathbf{x})||_h = (\mathbf{z} - \mathrm{E}\left[\mathbf{f}(\mathbf{x})\right])^T \mathrm{Var}\left[\mathbf{z} - \mathrm{E}\left[\mathbf{f}(\mathbf{x})\right]\right]^{-1} (\mathbf{z} - \mathrm{E}\left[\mathbf{f}(\mathbf{x})\right])$$

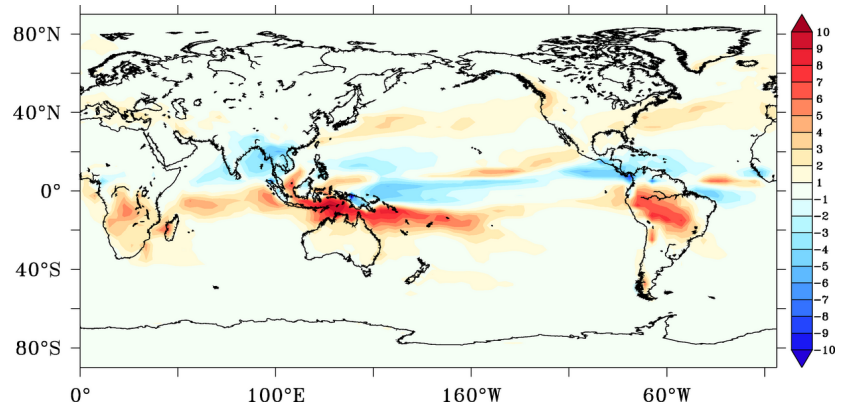
[1]Salter, J. M. et al. (2019) Uncertainty Quantification for Computer models with spatial output using calibration-optimal bases, *Journal of the American Statistical Association, 114, 1800-1814.*

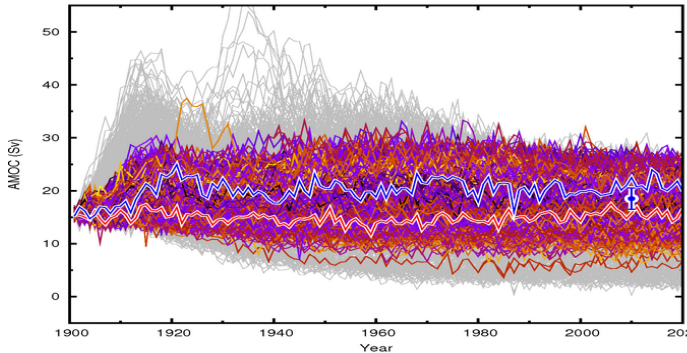
Kernel History Matching

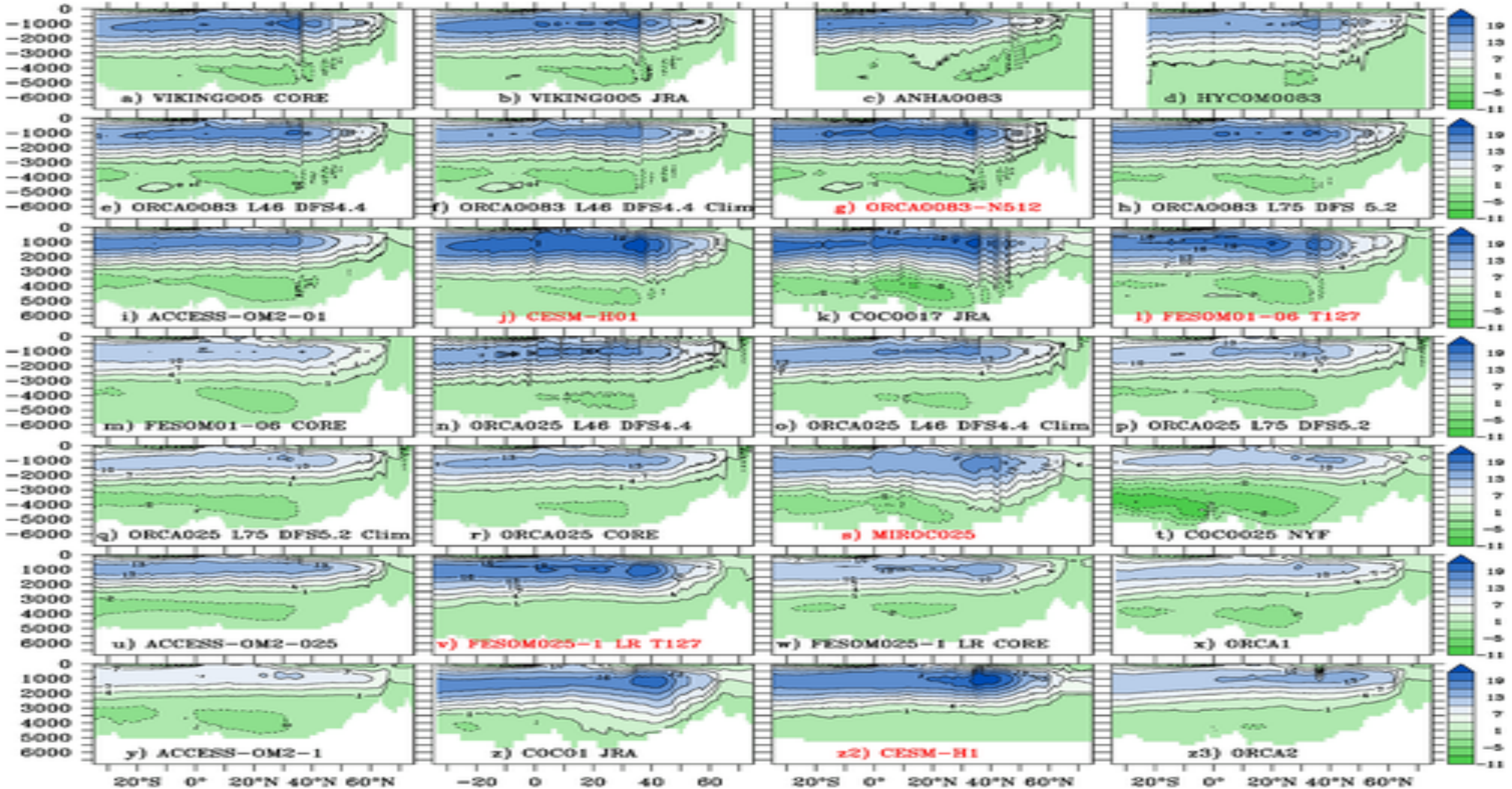
- The output of parameterisations is usually spatio-temporal.
- Whilst HM is most often done with simple metrics, often it is this spatio-temporal structure that is most important to capture.
- It is possible to use GPs for every grid-cell/time point, but not a good idea.
- In previous work[1] we use basis emulators to reduce the problem to a few metrics:

1. Use ensemble $f(x_1)$, …, $f(x_n)$, to obtain EOFs , B_1, …, B_n across the ensemble.
2. Keep q<<n EOFs to retain most ensemble signal.
3. Project ensemble onto basis to obtain q coefficients for each ensemble member: $c(x_i) = (B^T B)^{-1} B^T f(x_i)$
4. Emulate the coefficients and use history matching as normal.









Hirschi et al. 2020. The AMOC in High Resolution Models. JGR Oceans.

Kernel History Matching

- We imagine a mapping of model output into a higher dimensional feature space via $\phi(f(x))$.
- We specify a kernel to represent the dot product in our feature space:

$$k(f(x), f(x')) =< \phi(f(x)), \phi(f(x')) >= \phi(f(x))^T \phi(f(x')).$$

- We don't (need to) know $\phi(\cdot)$ (or the dimension of the space).
- This kernel property allows us to perform PCA in the feature space and compute coefficients $C(x_i)$ using the kernel.
- We obtain standard EOFs with $k(f(x), f(x')) = f(x)^T f(xi)$.
- Non-linear kernels such as $k(f(x), f(x') = \exp\left\{-\frac{||f(x)-f(x')||^2}{2\sigma^2}\right\}$ allow non-linear features to be extracted.

Kernel History Matching

- History matching in feature space requires a slight reformulation.
- We first incorporate Ψ = observation + structural error within the kernel via

$$k(f(x), f(x')) = \omega f(x)^T \Psi^{-1} f(x') + (1-\omega) \exp\left\{ -\frac{1}{\sigma}(f(x) - f(x'))^T \Psi^{-1} (f(x) - f(x')) \right\}$$

- We then have access to implausibility

$$\mathcal{I}(x) = ||\phi(z) - \mathrm{E}\left[\phi(f(x))\right]||^2$$

through the kernel, and can cut space with

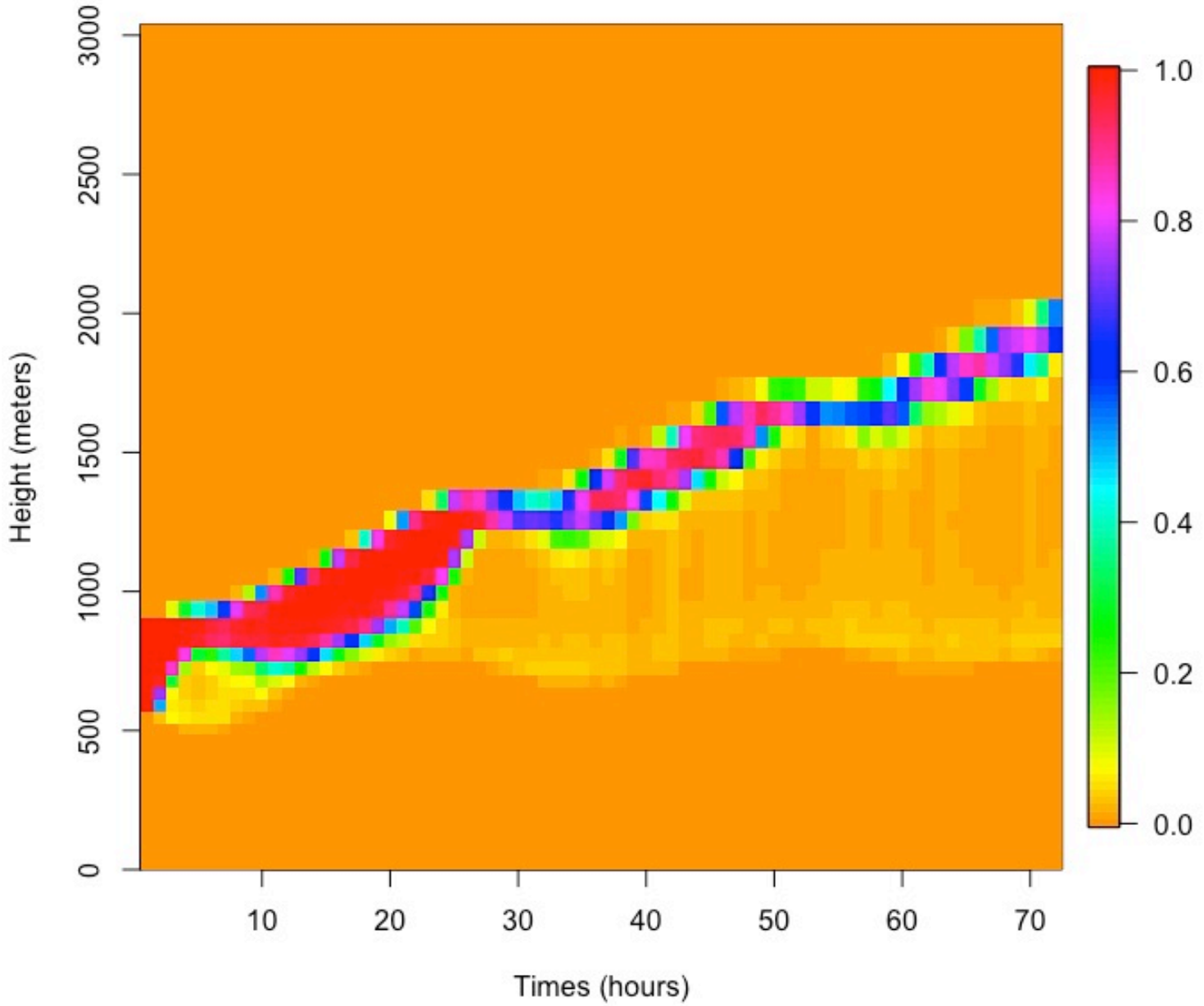
$$\mathcal{I}(x) > T(x)$$

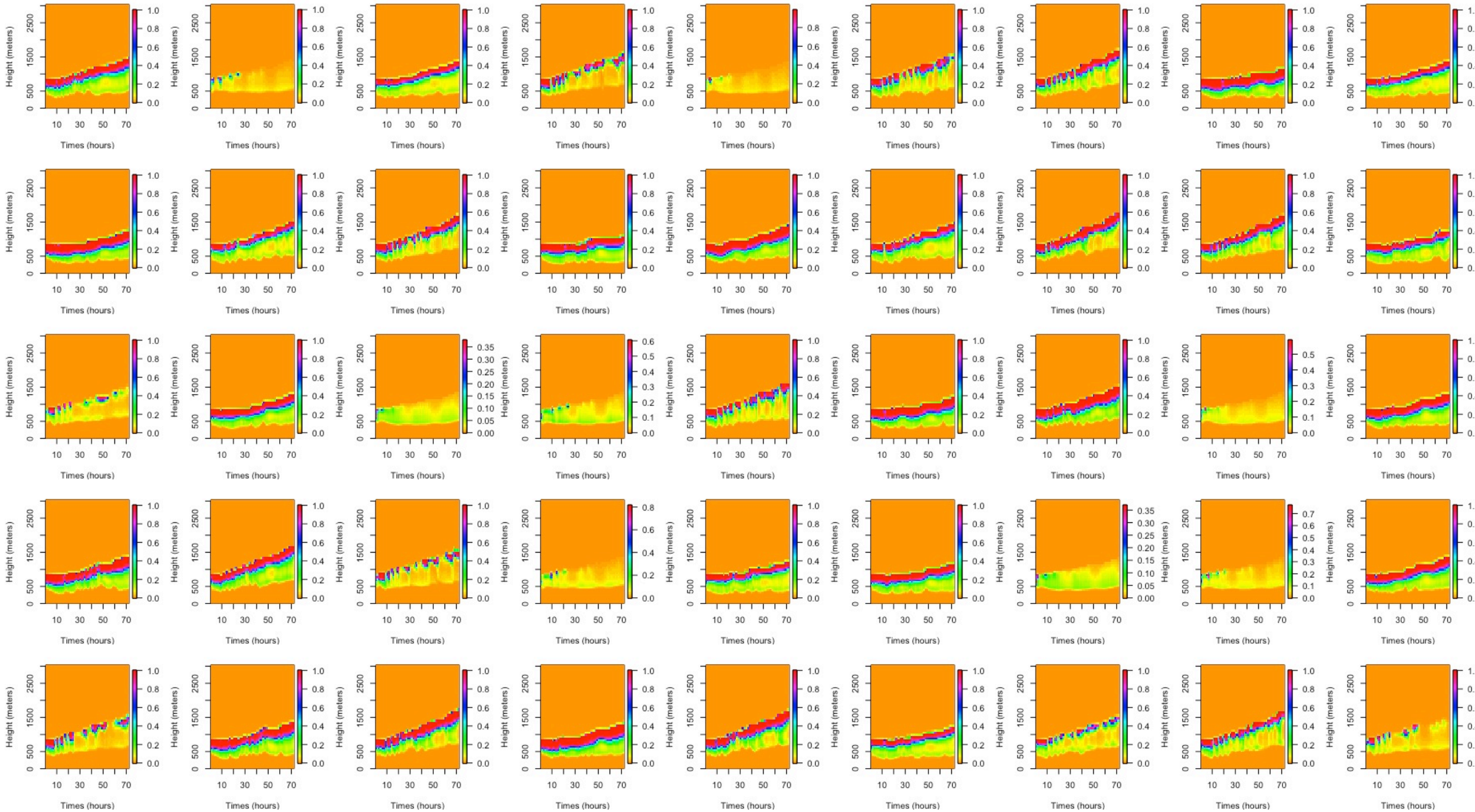
where

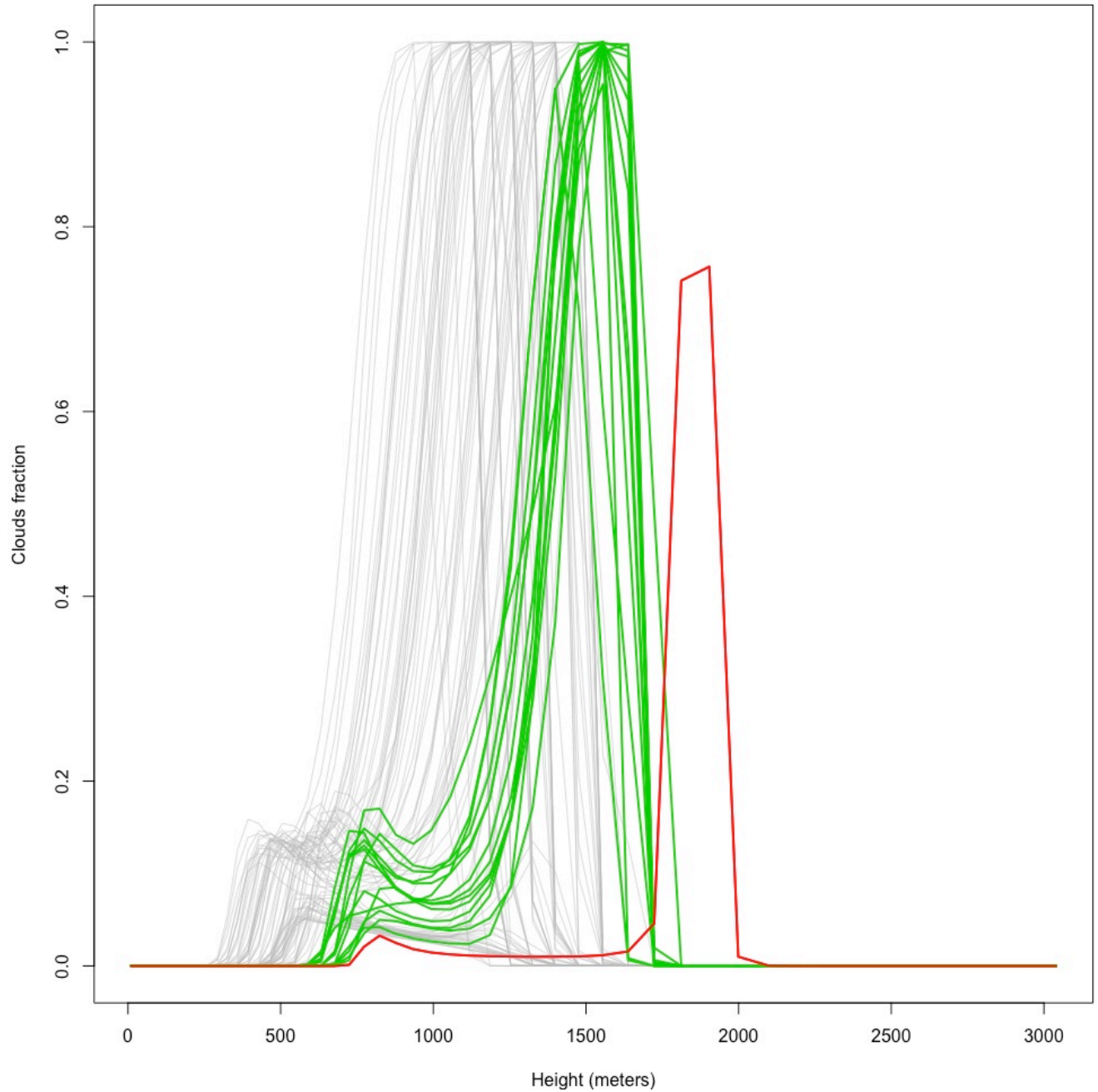
$$T(x) = \sum_{k=1}^{q} \mathrm{Var}(C_k(x)) + 3\sqrt{2\left(\sum_{k=1}^{q} \mathrm{Var}(C_k(x))\right)^2 + const}$$

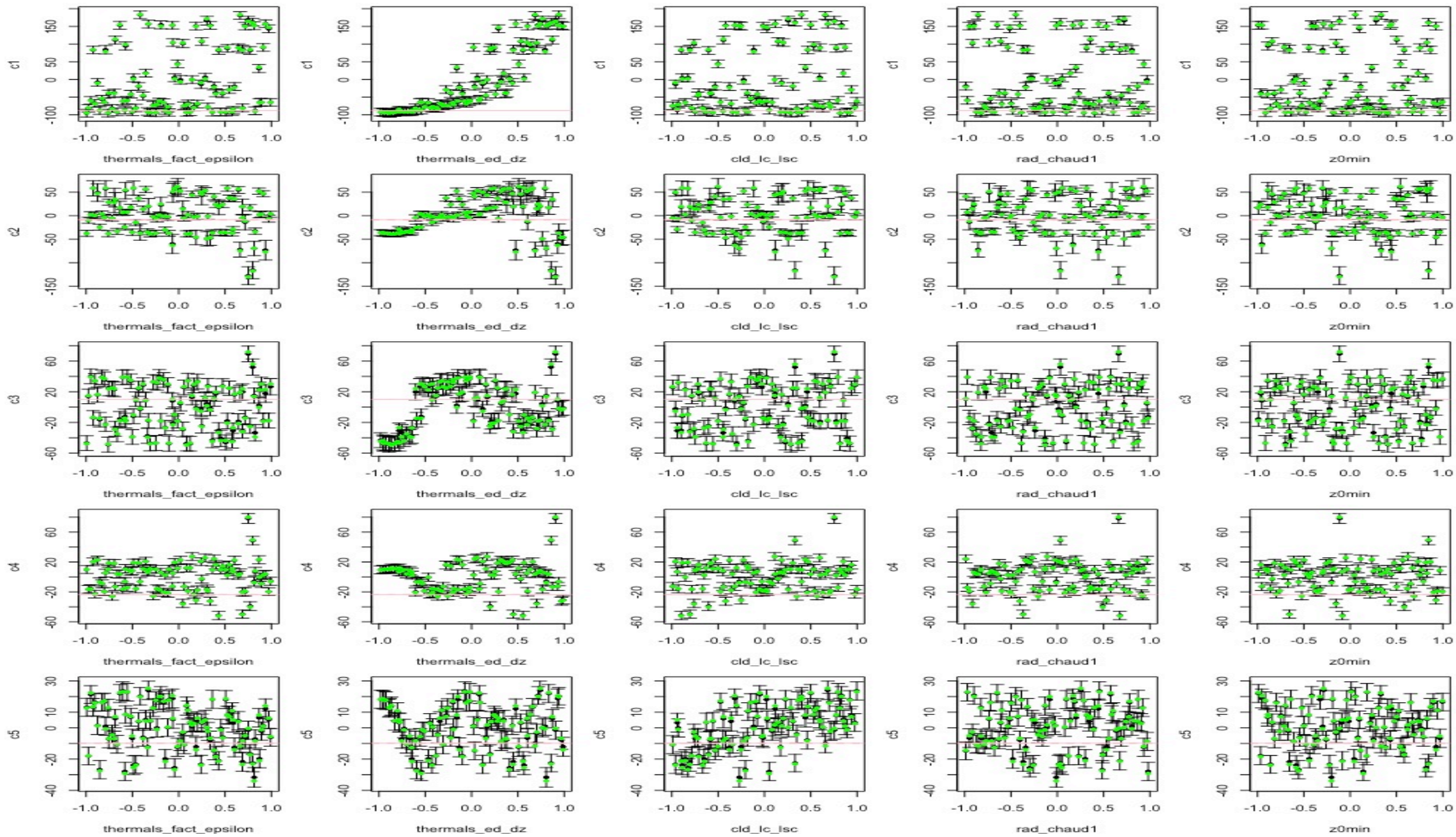
Xu, W. and W. (2021) Kernel History Matching, in prep.

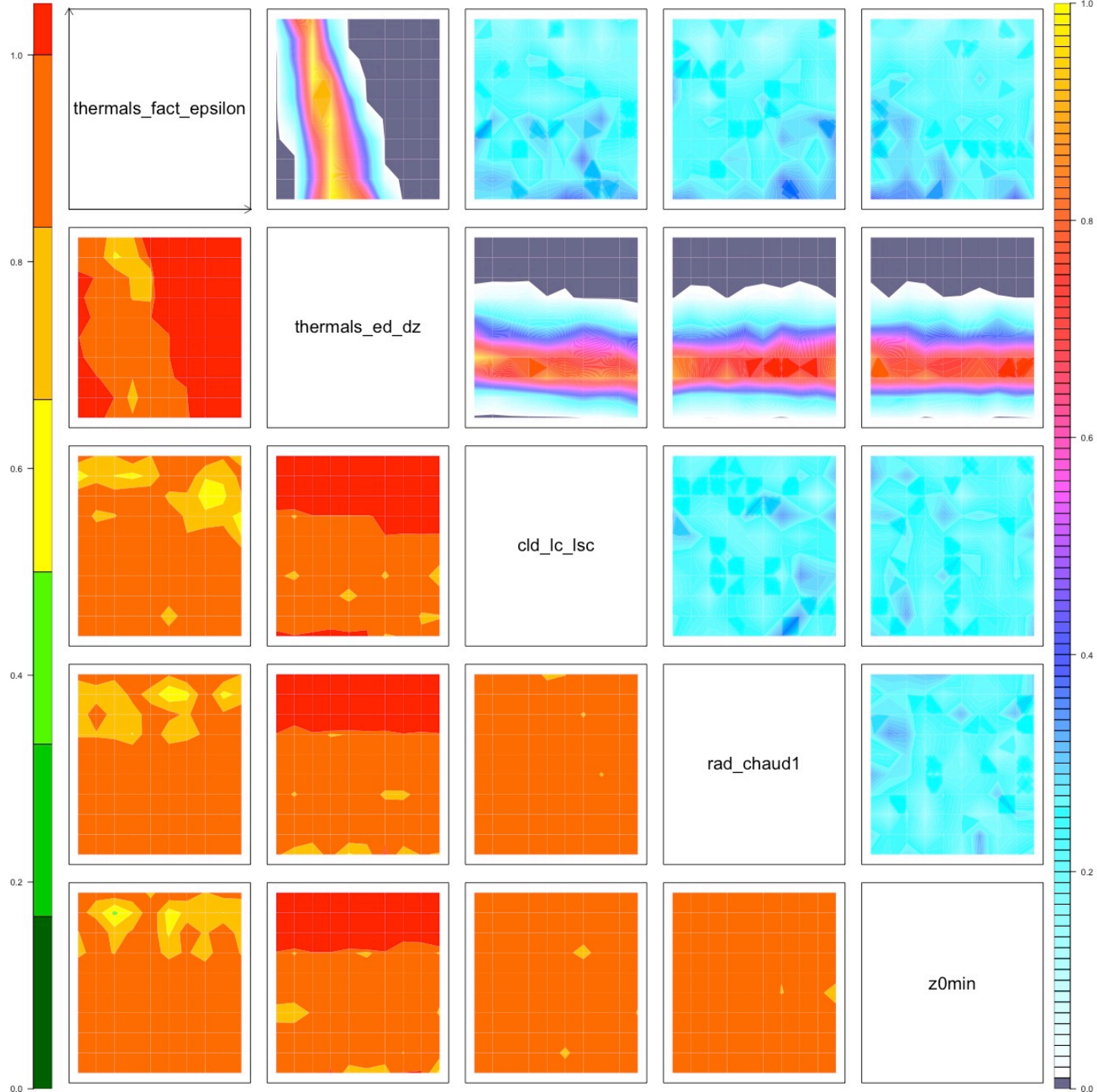
- Cloud fraction evolution of LES for SANDU/REF case in LMDZ.
- Which structures in the LES are important for the SCM to capture?
- Which structures are not important?
- How can we know it is right for the right reasons?

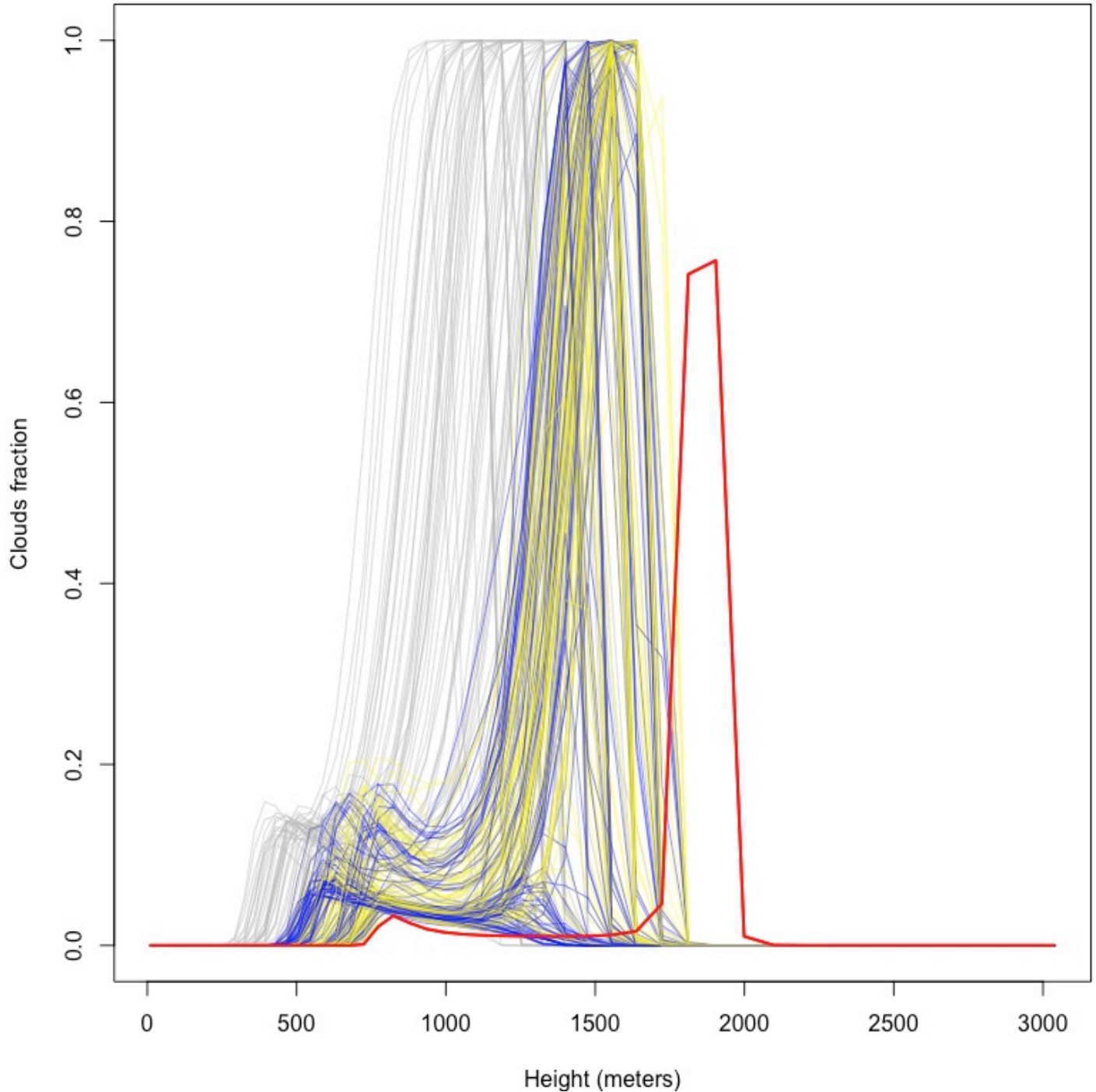



Height (meters)
0
500
1500
2500
Times (hours)
10
30
50
70


Clouds fraction
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
Height (meters)


c1
c2
c3
c4
c5
thermals_fact_epsilon
thermals_ed_dz
cld_lc_lsc
rad_chaud1
z0min


thermals_fact_epsilon
thermals_ed_dz
cld_lc_lsc
rad_chaud1
z0min
0.0
0.2
0.4
0.6
0.8
1.0

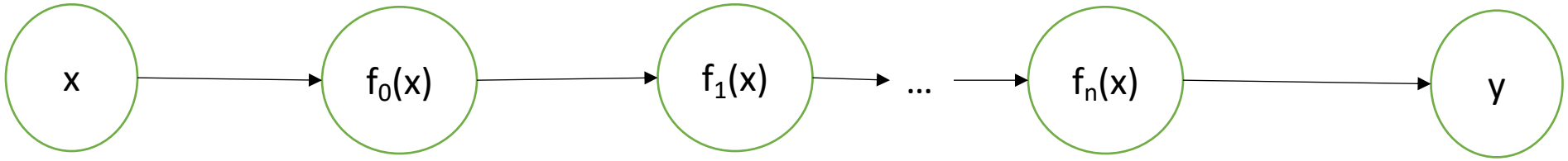

Clouds fraction
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
Height (meters)

Coupled models as Deep Gaussian Processes

- Like Neural Networks, there are ‘Deep’ Gaussian Processes too:

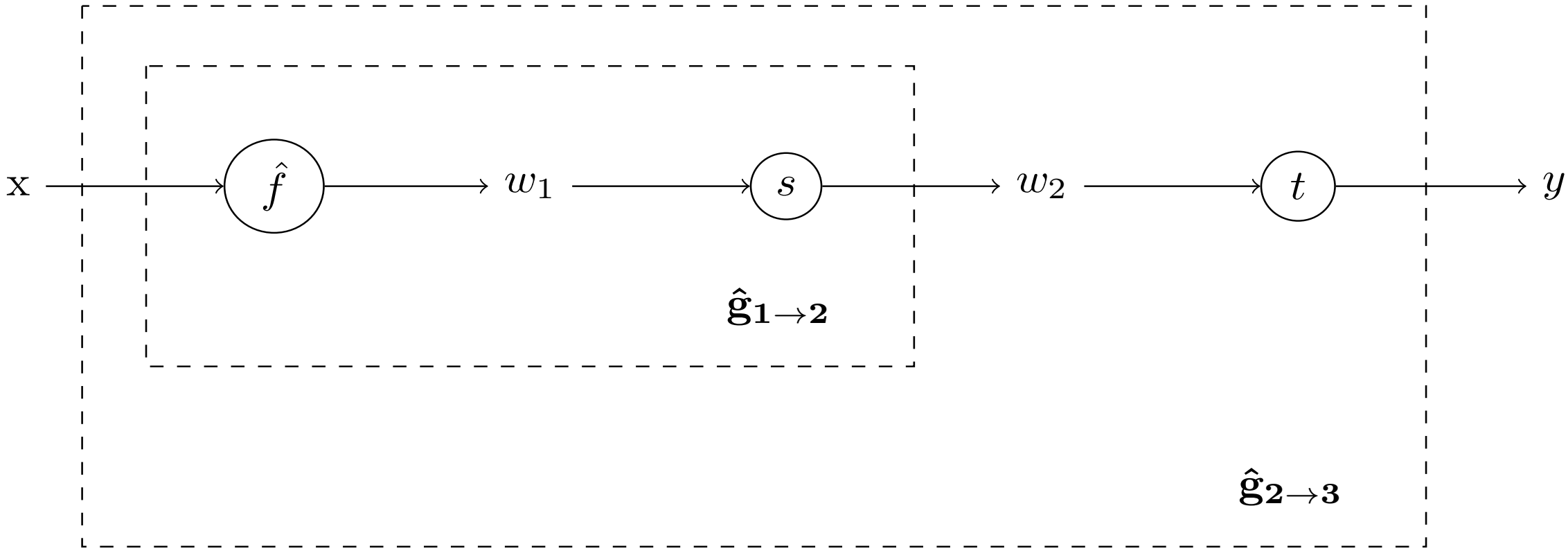
$$
\begin{aligned}
y \mid f_n(x) &\sim \mathrm{N}(f_n(x), \sigma^2) \\
f_n(x) \mid f_{n-1}(x) &\sim \mathrm{GP}(0, k(f_{n-1}, f'_{n-1})) \\
&\dots \\
f_0(x) &\sim \mathrm{GP}(0, k(x, x'))
\end{aligned}
$$

DGPs are particularly good at capturing non-stationarity.

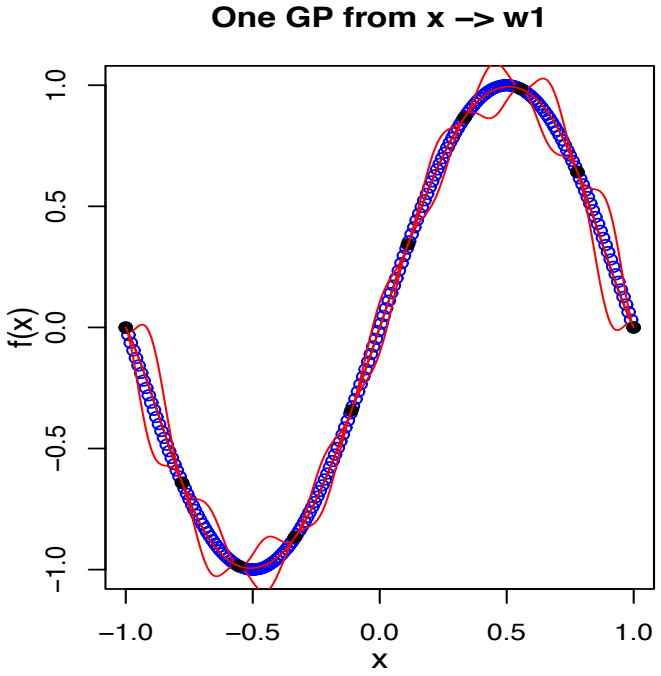


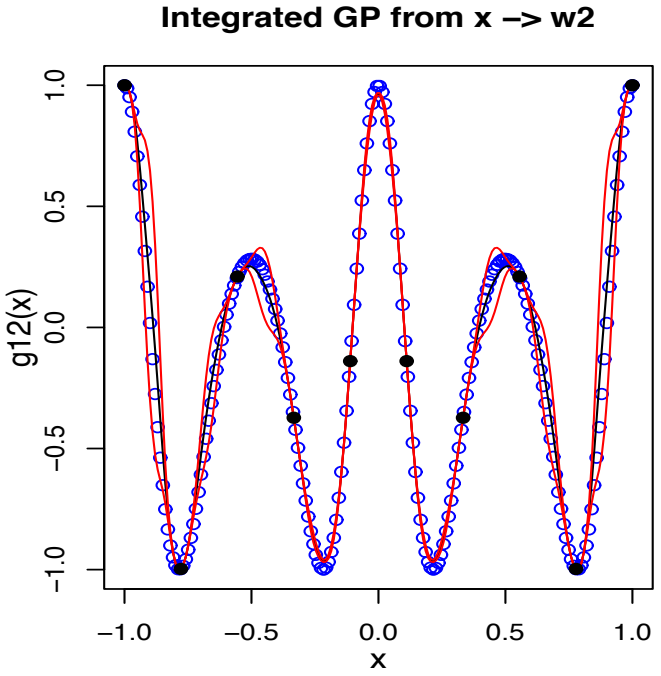
Coupled models as Deep Gaussian Processes

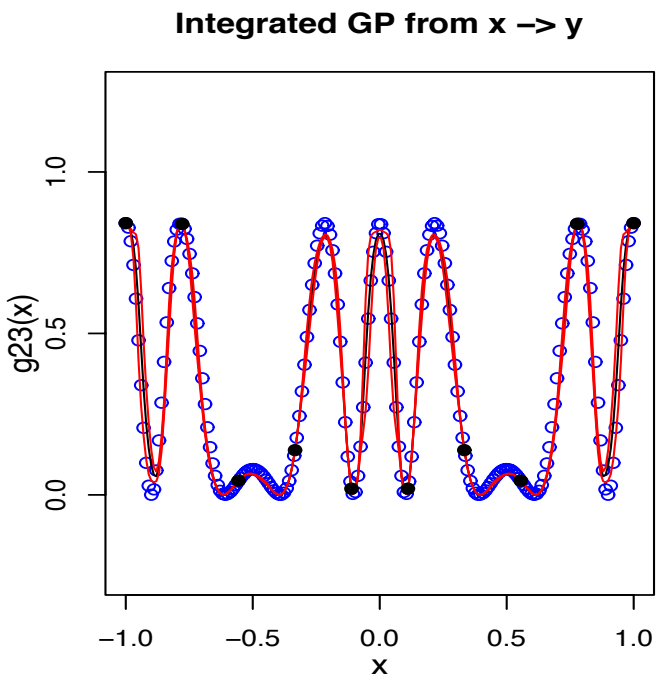
Idea: If we treat coupled models as DGPs, can we benefit from exposure of the hidden layers?

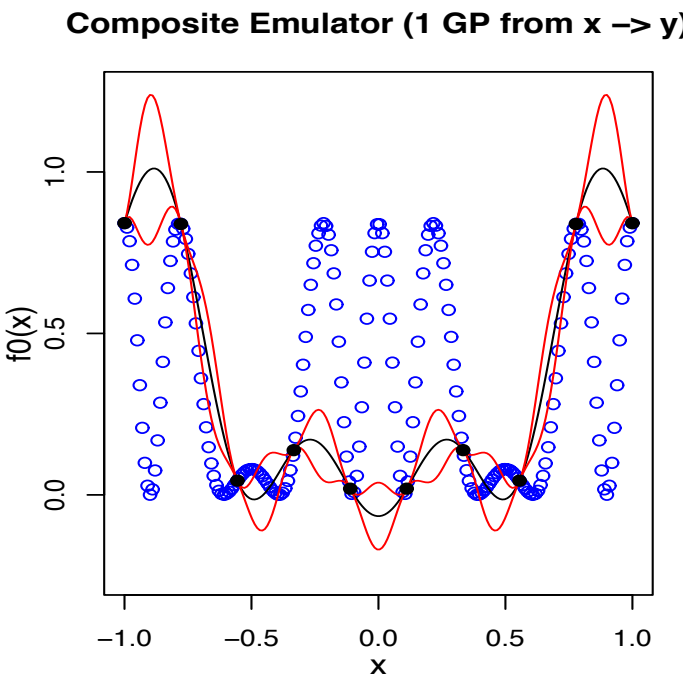


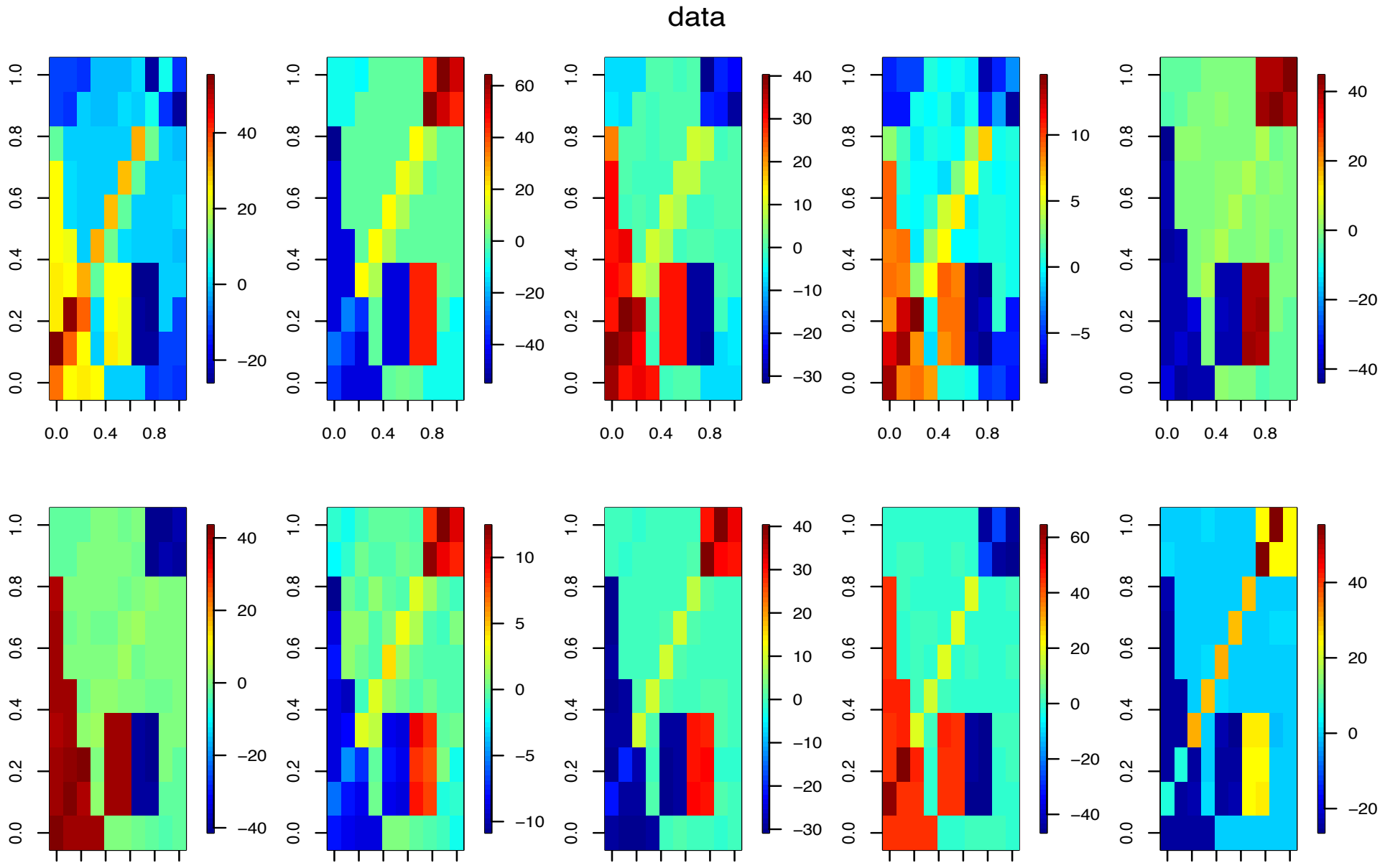
Ming and Guillas (2021), Integrating Gaussian Process Emulators, https://arxiv.org/abs/1912.09468

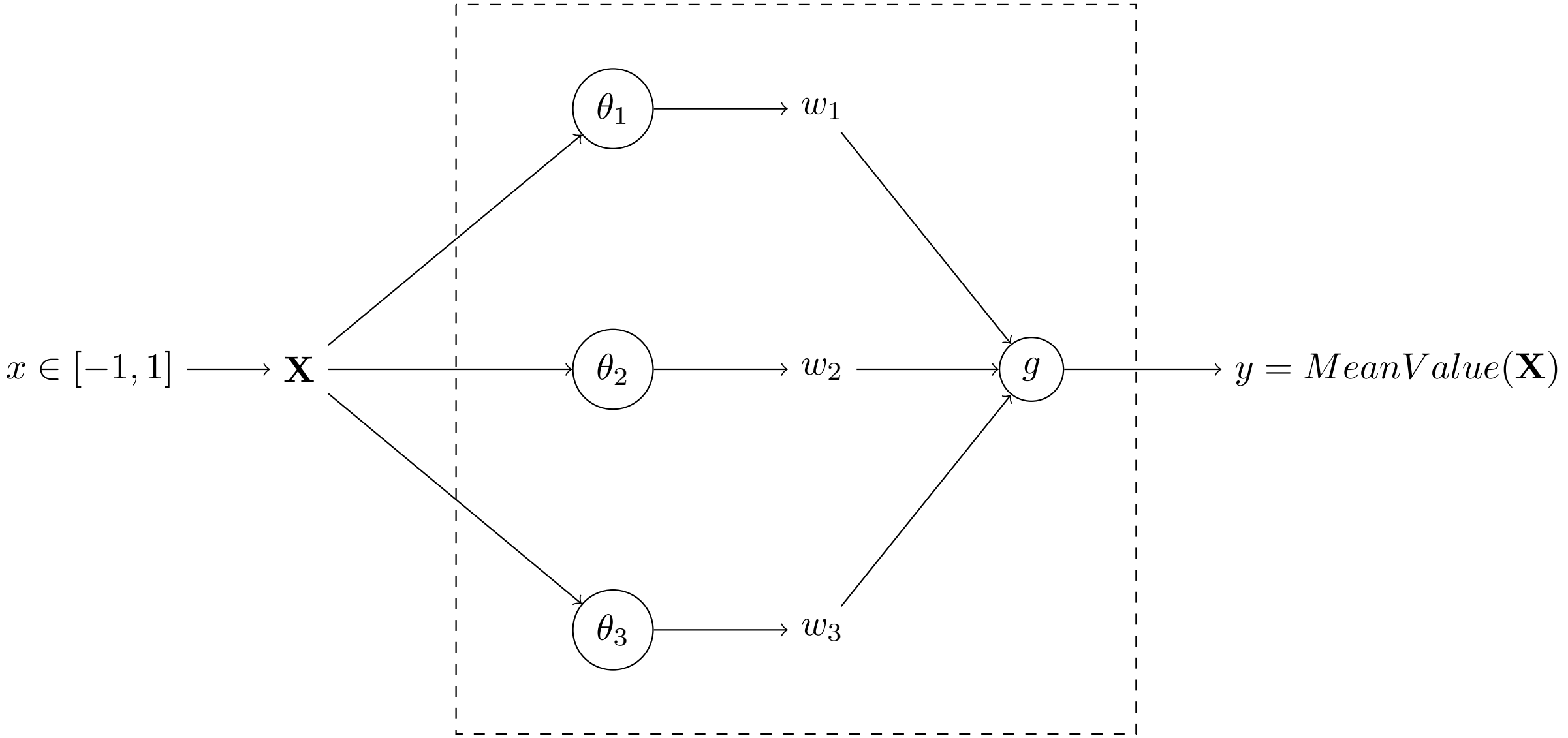

One GP from x –> w1
f(x)
x
−1.0
−0.5
0.0
0.5
1.0

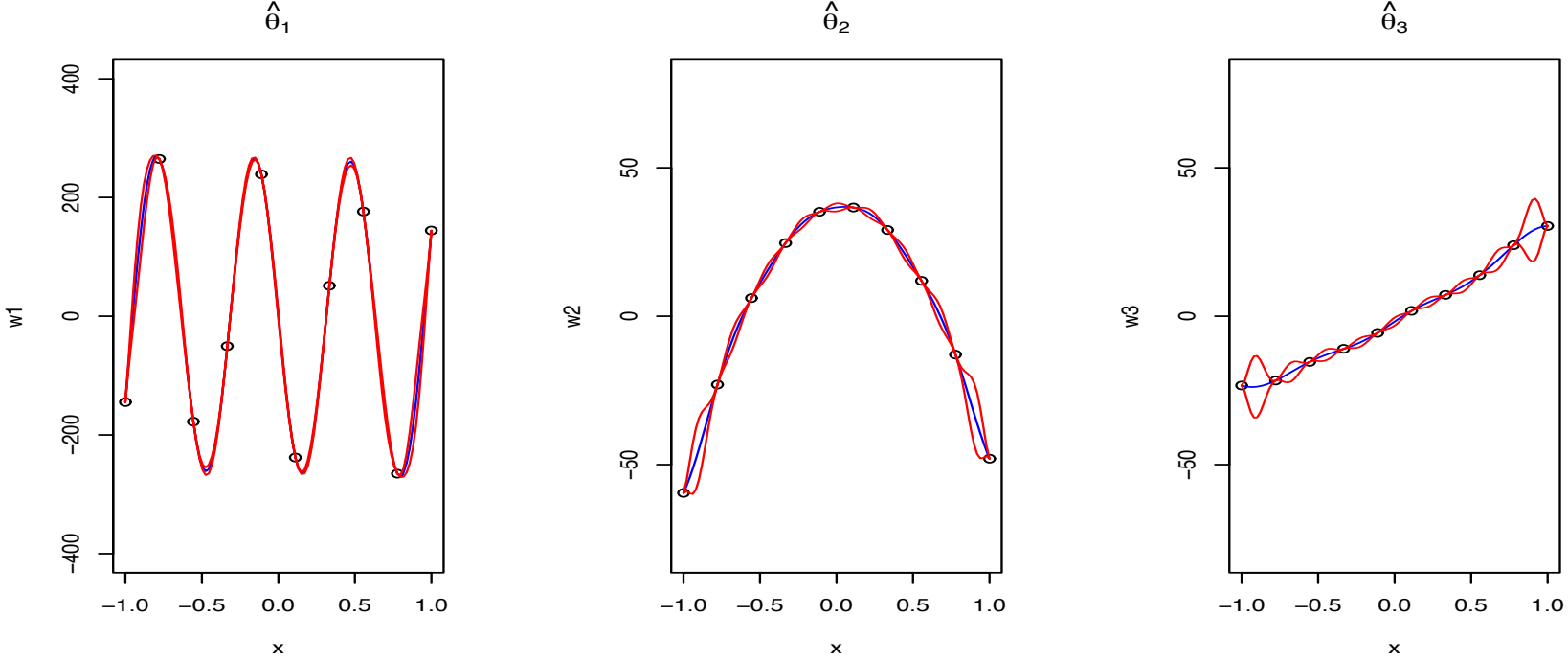

Integrated GP from x –> w2
g12(x)
x
−1.0
−0.5
0.0
0.5
1.0

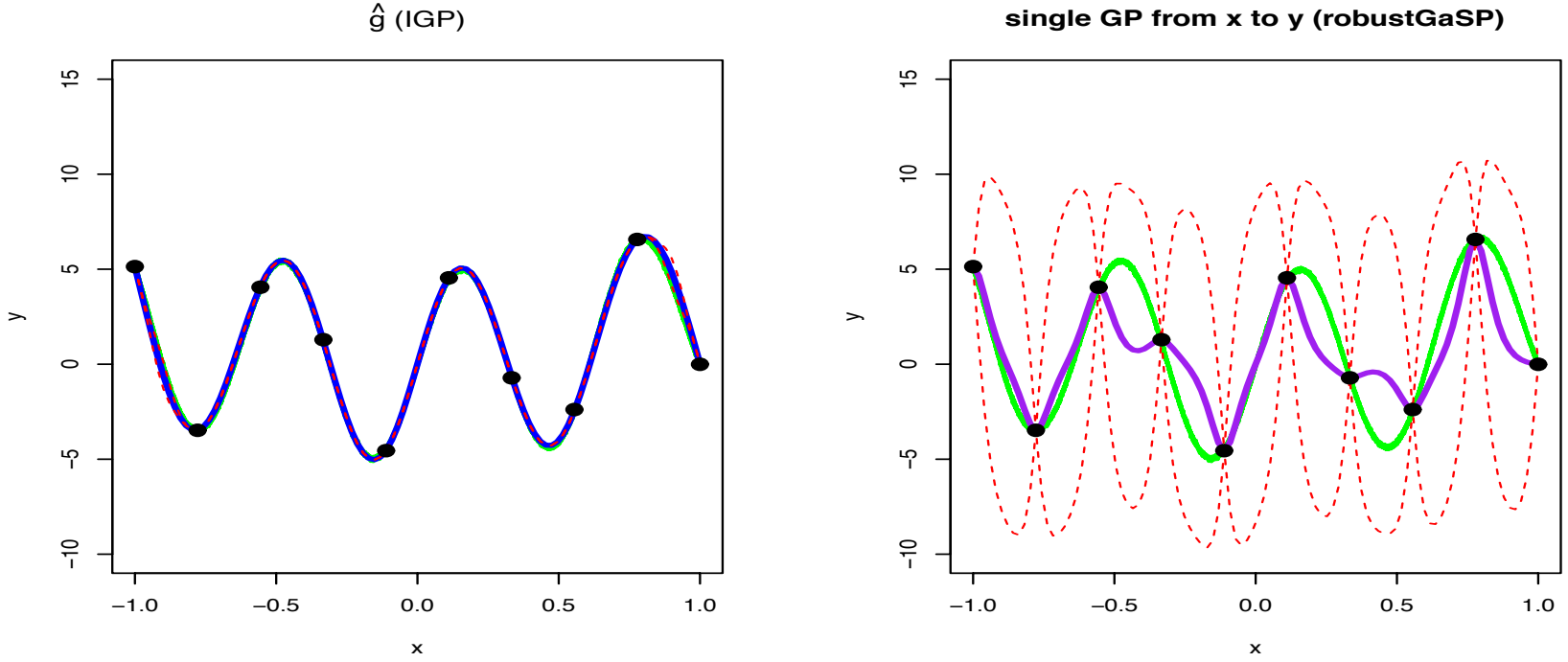

Integrated GP from x –> y
g23(x)
x
0.0
0.5
1.0


Composite Emulator (1 GP from x –> y)
f0(x)
x
0.0
0.5
1.0


data
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
40
20
0
−20
60
40
20
0
−20
−40
40
30
20
10
0
−10
−20
−30
10
5
0
−5
40
20
0
−20
−40
40
20
0
−20
−40
10
5
0
−5
−10
40
30
20
10
0
−10
−20
−30
60
40
20
0
−20
−40
40
20
0
−20


$x \in [-1, 1]$
$\mathbf{X}$
θ_1
θ_2
θ_3
w_1
w_2
w_3
g
$y = MeanValue(\mathbf{X})$


$\hat{\theta}_1$
$\hat{\theta}_2$
$\hat{\theta}_3$
w1
w2
w3
x
400
200
0
−200
−400
50
−50
−1.0
−0.5
0.0
0.5
1.0


$\hat{g}$ (IGP)
single GP from x to y (robustGaSP)
y
x
15
10
5
0
−5
−10
−1.0
−0.5
0.0
0.5
1.0