Model inference for Ordinary Differential Equations by parametric polynomial kernel regression

UNCECOMP 2019 - Crete, Greece

David K. E. Green
dgreen@turing.ac.uk
(Joint work with F. Rindler)

The Alan Turing Institute

London, United Kingdom

WARWICK
THE UNIVERSITY OF WARWICK

Overview

ODE transition model Maximum Likelihood problem

$$\frac{d}{dt}u(t) = f(t, u(t))$$

Find f given observations of u

Kernels for reducing Neural Network dimensionality

Finding ways to reduce the complexity of Neural Networks for continuous model inference.

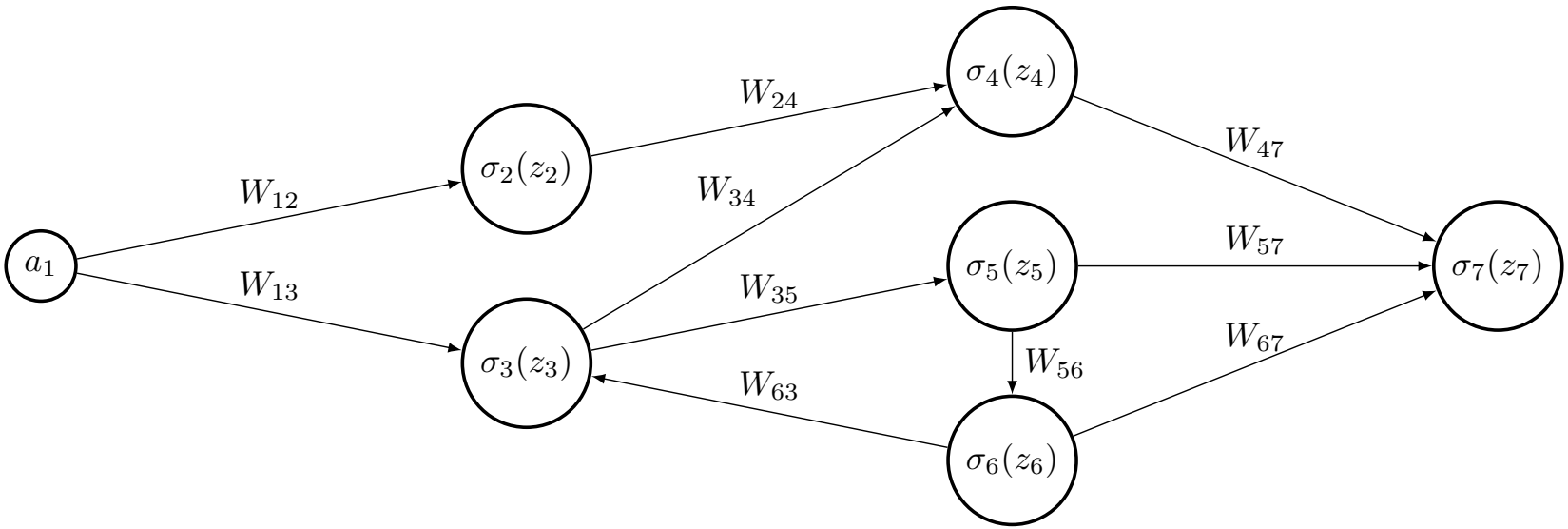
- Compute graph optimisation.
- Ordinary Differential Equation model inference.
- Parametric polynomial kernel regression.
- Numerical analysis of the Lorenz-Emanuel system.

Background on Compute Graphs

- Efficient optimisation for function representations.
- Complex nonlinearity by composition of simpler functions.

Computation graphs

- Computations can be represented as graphs
- Computations without recursion: directed acyclic graphs



For all nodes with m parents, $\{v_i\}_i^m$, in the compute graph:

$$\text{Input to a node: } z_i = \sum_{v_i}^{m} W_{ki} a_k$$

$$\text{Output from a node: } a_i = \sigma_i(z_i)$$

Training for directed compute graphs

Setup: Given training data samples: $\{(x_i, f(x_i))\}_i^N$ (input and output) and some compute graph.
Hypothesis space parameterised by edge weights.
Approximate:

$$\tilde{f}_W(x) \stackrel{?}{=} f(x) \text{ for } W \in \Theta$$

Assign loss (or error) functional:

$$J(W) = \|f(x) - \tilde{f}_W(x)\| \text{ so } \tilde{f}^*_W(x) = \underset{W \in \Theta}{\text{argmin}}\ J(W)$$

Train: Minimise $J(W)$ over training samples, pairs $\{(x_i, f(x_i))_i^N\}$, by gradient descent:

$$W \leftarrow W - \alpha \nabla_W J(W) \text{ for all } (x_i, f(x_i))$$

Process: Hope that $\tilde{f}^*_W(x)$ is ok for predicting $f(x)$ given new x.

Passing gradients through Directed Acyclic Graphs

- How to calculate $\nabla_W J(W)$?
- Can pass error gradients backward through a graph using the chain rule. This is backpropagation.
- This calculates derivative of inputs (weights) with respect to outputs (loss functions).

$$\delta_i = \frac{\partial J}{\partial z_i}$$

$$\delta_i = \sum_{j=1}^{N} \delta_j \frac{\partial z_j}{\partial a_i} \frac{\partial a_i}{\partial z_i} = \sum_{j=1}^{N} \delta_j W_{ij} \sigma_i'(z_i)$$

$$\nabla_W J(W) \ni \frac{\partial J}{\partial W_{ij}} = \delta_j \frac{\partial z_j}{\partial W_{ij}} = \delta_j a_i$$

Model inference for ODEs

- Discretising ODEs for parametric inverse problems.

Continuous model inference

ODE transition model Maximum Likelihood problem

$$\frac{d}{dt}u(t) = f(t, u(t))$$

Find f given observations of u.

Summary of technique:

- Discretise ODE integral/derivative representation.
- Represent f parametrically by f_W.
- Use observations of u to find optimal parameters W.

Infer ODEs from data - Derivation 1/2

Convert to integral form and insert model f_W (for $W \in \Theta$):

$$\frac{d}{dt}u(t) = f_W(t, u(t))$$

$$\tilde{u}(t) = u(0) + \int_0^t f_W(\tau, u(\tau))d\tau$$

We want to solve for W:

$$\begin{aligned}\text{Minimise } J(W) &= \int_0^T \left|u(t) - \tilde{u}(t)\right|^2 dt \\ &= \int_0^T \left|u(t) - \left(u(0) + \int_0^t f_W(\tau, u(\tau))d\tau\right)\right|^2 dt\end{aligned}$$

given observations of $u(t)$.

Infer ODEs from data - Derivation 2/2

Average over the set of training data, observations $(t_i, u(t_i))$, to approximate the loss functional:

$$\tilde{J}(W) = \frac{1}{N}\sum_{i=1}^{N}\left|u(t_i) - \left(u(0) + \int_0^{t_i} f_W(\tau, u(\tau))d\tau\right)\right|^2$$

Discretising the integral term gives an equation that can be solved. For example, using Forward Euler:

$$\tilde{J}_F(W) := \frac{1}{N-1}\sum_{i=1}^{N-1}\left|u(t_{i+1}) - (u(t_i) + |t_{i+1} - t_i| f_W(t_i, u(t_i))\right|^2$$

We show that for a differentiable representation of f_W, J can be minimised by Backpropagation and Stochastic Gradient Descent (i.e. Artificial Neural Network training methods).

Parametric polynomial regression

- What compute graph architecture to use?
- Standard Neural Network activation functions perform poorly for ODE inference tasks.
- Parametric polynomial kernels work well, but have poor performance for time series inverse problems.
- Use a parametric polynomial kernel instead.

Nonparametric polynomial regression

Polynomial kernel:

$$K(x, y) = (\langle x, y \rangle + 1)^d$$

implicitly represents all polynomial terms up to order d.

Given observations $\{(x_i, f(x_i))\}_i^N$, $f(x)$ can be approximated by:

$$f(x) \approx f_k(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)$$

$$\alpha_i \in \alpha = (K + \lambda I)^{-1} f(x)$$

with K the matrix with entries $K_{ij} = K(x_i, x_j)$.

Problem: nonparametric approach grows with size of data set. Not workable for time series data. Explicit polynomial expansion has factorial explosion in number of degrees of freedom.

Parametric polynomial regression

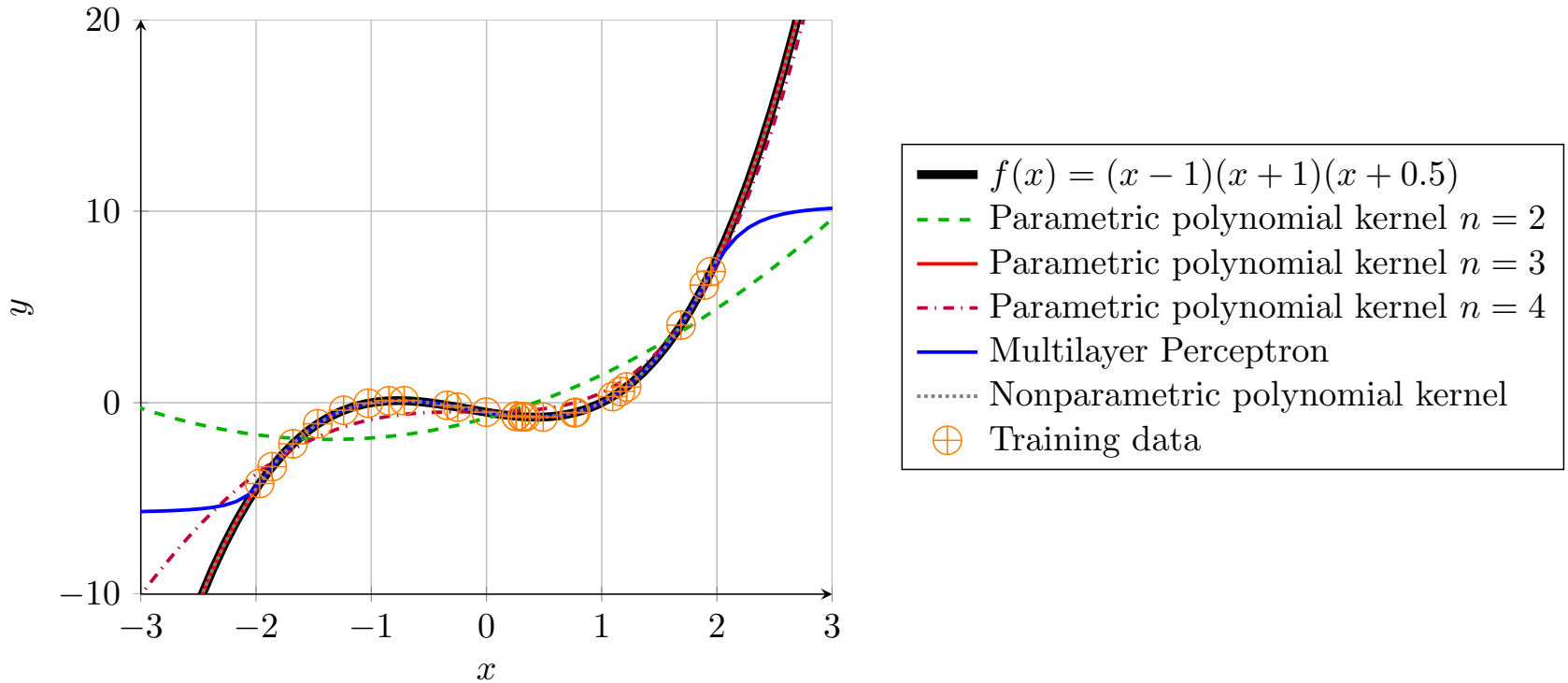
Parametric polynomial kernel representation

$$f_W(x) = W_2 \left[(W_1 x + B_1) \circ^d (W_1 x + B_1) \right] + B_2$$

for matrices W_1, W_2 and vectors B_1, B_2 where $\circ^d$ is an elementwise product (Hadamard product), repeated d times.

- Expand to intermediate dimension M with W_1, then reduce back down to dimension of $f_W(x)$ with W_2.
- Differentiable, can use Gradient Descent and Backpropagation (standard differentiable compute graph training).
- Expanding the above gives sets of polynomial equations similar to the $\alpha K(x, x_i)$ terms in the nonparametric polynomial estimator.
- Support vector machines meet Artificial Neural Networks. The above technique is a cousin of radial basis function networks.

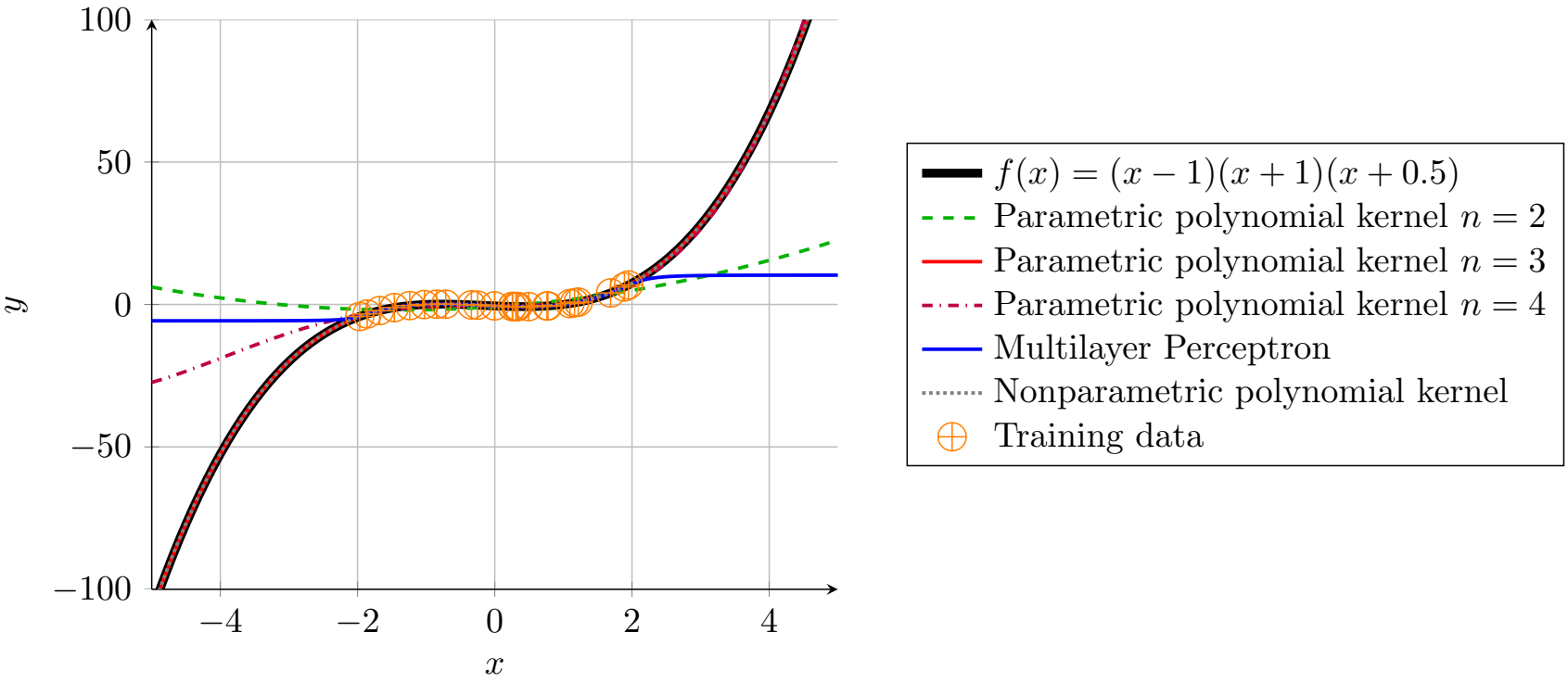
Simple numerical example



Fit performance.

Comparison of parametric polynomial kernel method with standard Artificial Neural Networks (multilayer perceptron) and polynomial kernel ridge regression.

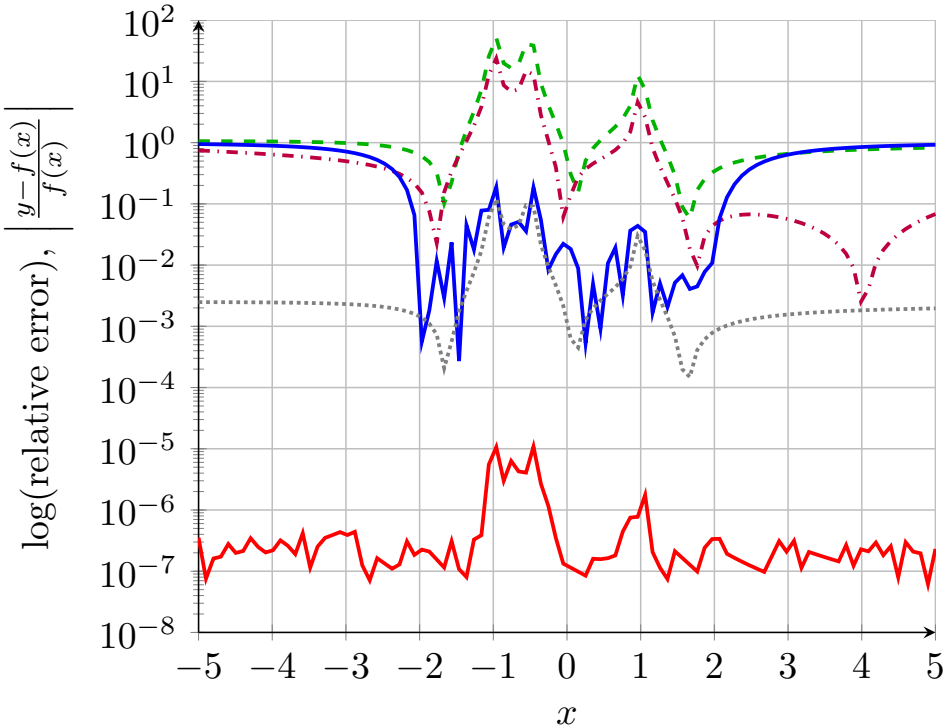
Simple numerical example

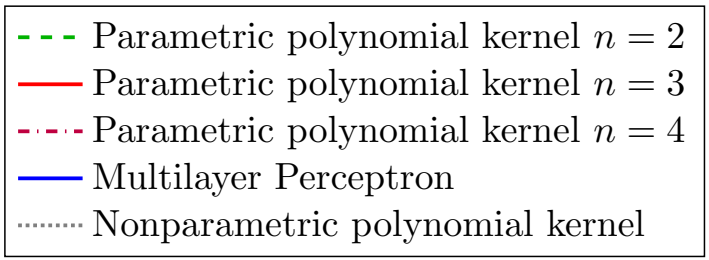


Zoomed out view.

Comparison of parametric polynomial kernel method with standard Artificial Neural Networks (multilayer perceptron) and polynomial kernel ridge regression.

Simple numerical example





Fit errors.

Comparison of parametric polynomial kernel method with standard Artificial Neural Networks (multilayer perceptron) and polynomial kernel ridge regression.

Numerical example

- Chaotic dynamical system.

Test model: Lorenz-Emmanuel system

- Consider N variables, u_i for $1 \leq i \leq N$, arranged periodically s.t. $u_{N+1} = u_1$ and $u_0 = u_N$ and $u_1 = u_{N-1}$.
- The Lorenz-Emmanuel system equation is given by:

$$\frac{du_i}{dt} = (u_{i+1} - u_{i-2})\, u_{i-1} - u_i + F$$

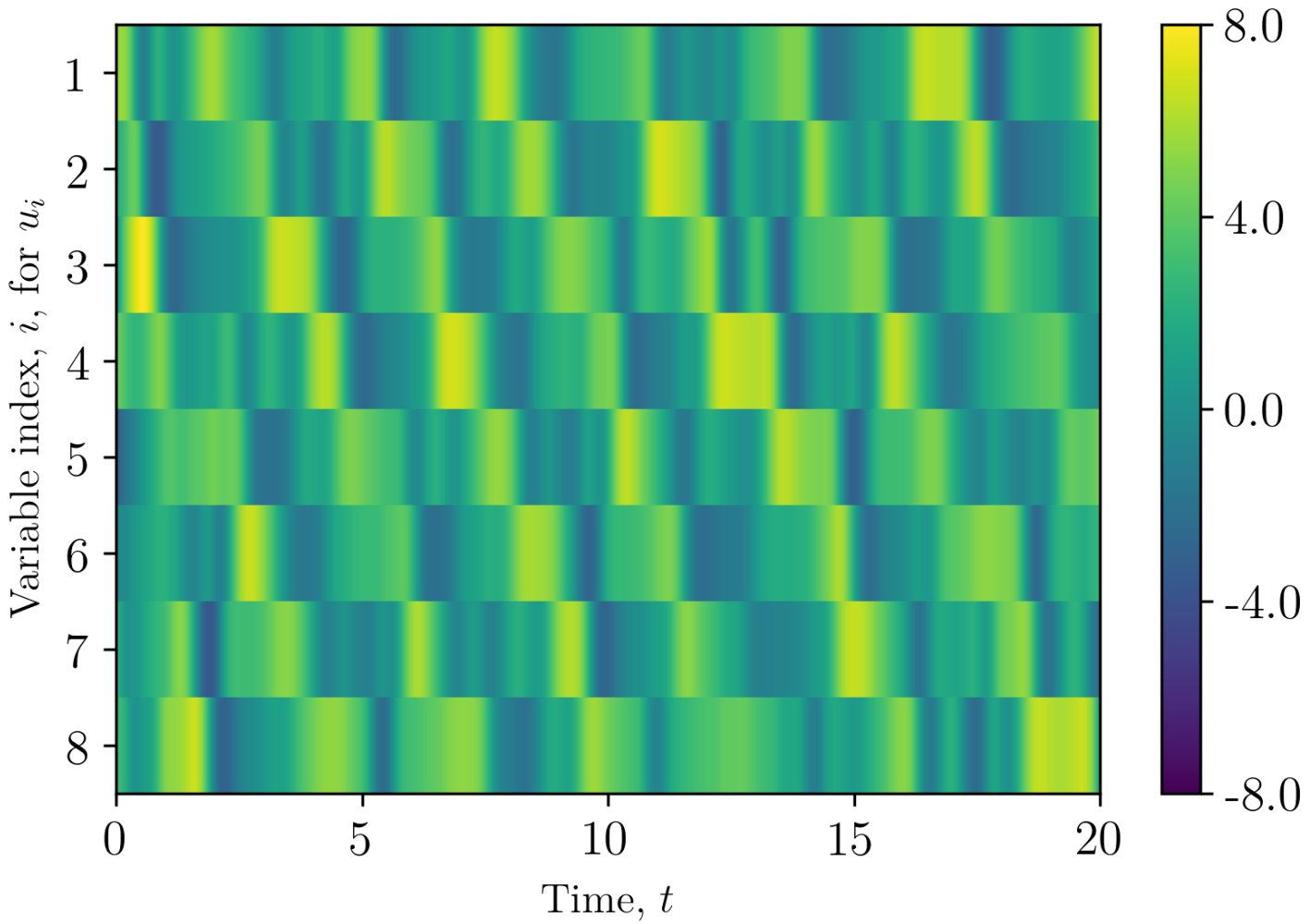
for $u := \{u_i\}_{i=1}^{N}$ and $F \in \mathbb{R}$.

Test model: Lorenz-Emmanuel system

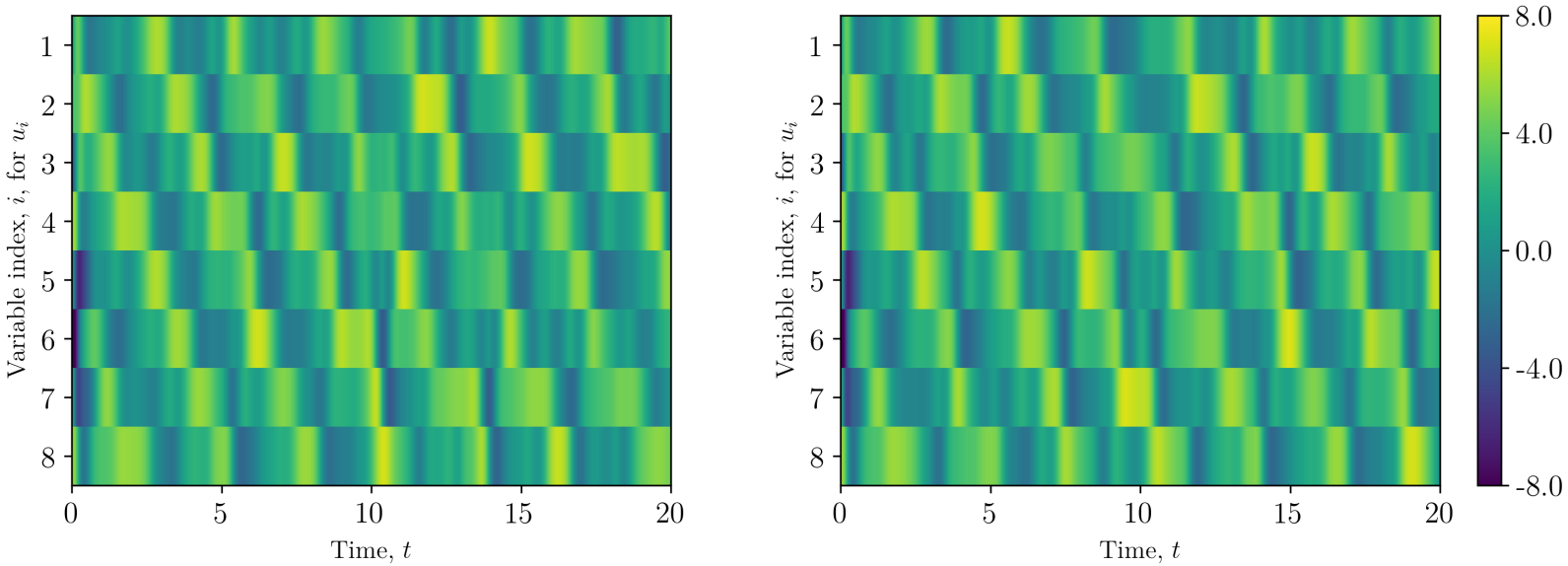
Analysed case:

- $F = 5$, $N = 8$.
- Training data collected from a random initial condition, $u(t = 0) \sim \mathcal{N}(\mu = 0, \sigma = 3)$, run for 20 time units, 1000 samples per time unit.
- ODE discretisation for loss functional: Adams-Moulton Linear Multistep (equations in the paper!).

Training data

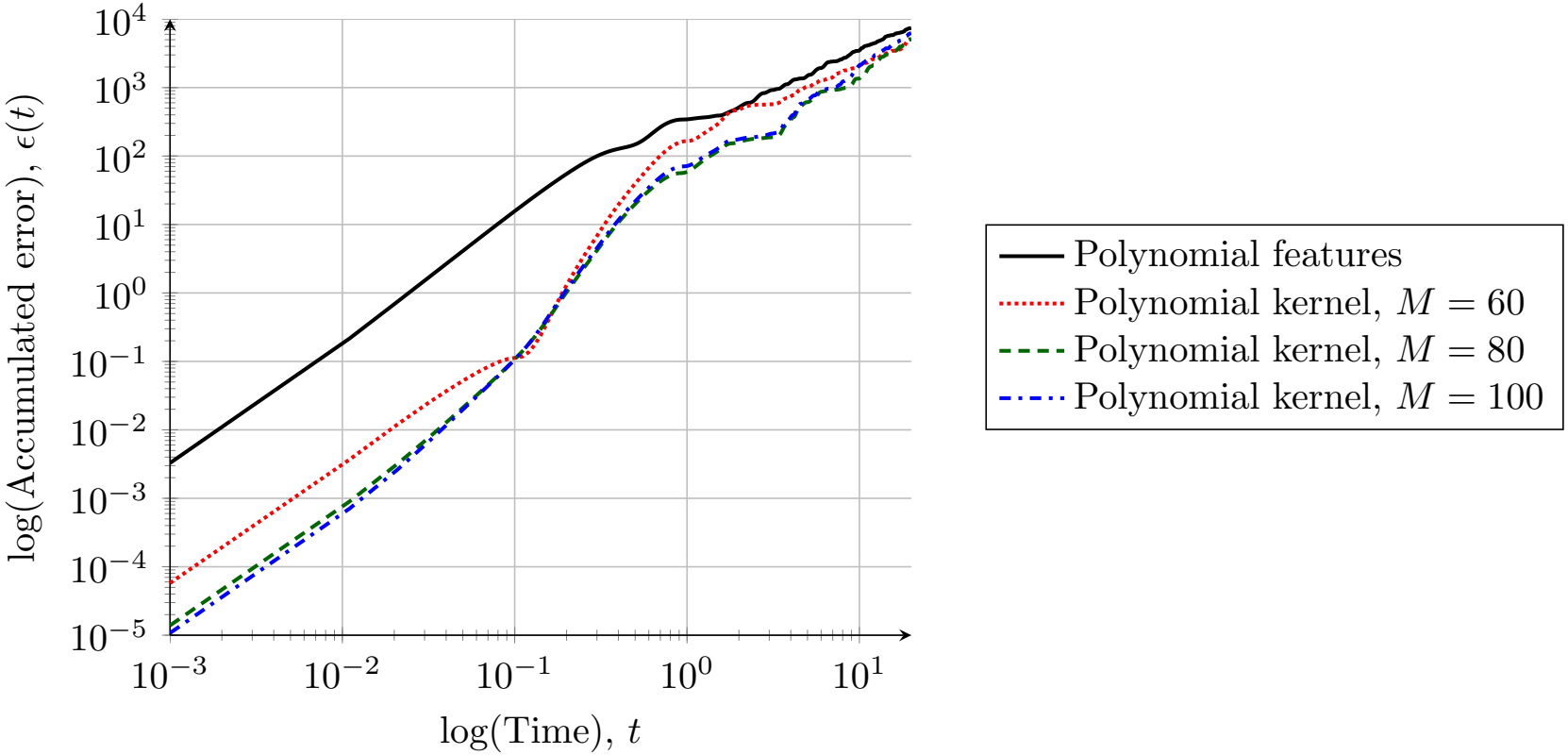


Lorenz-Emmanuel System prediction



True Lorenz-Emmanuel system model (left) vs model predicted from very few earlier observations (right). Accurate over times longer much than data sampling rate.

Lorenz-Emmanuel System prediction



Error comparison for polynomial kernel estimator. Note that parametric polynomial kernel outperforms explicit second order parametric polynomial feature representation. Previously unseen initial condition.

Conclusions

- Short introduction to directed compute graphs
- Model inference for ODEs - discretisation to generate inverse problem objective functions.
- Reducing problem dimensionality (and increasing accuracy) using implicit polynomial representations (kernels).
- Standard extension to Bayesian regression analysis via Gibbs measure:

$$P(W) \propto \exp\left(-\|J(W)\|^2\right)$$

Thank you!

Model recovery - derive optimisation objective

ODE transition model

$$\frac{d}{dt}u(t) = f(t, u(t))$$

Option 1 - Fit $u_W(t)$ and differentiate to recover $f(t, u(t))$

$$J(W) = \|u_W(t) - u(t)\|$$
$$f(t, u(t)) = \frac{d}{dt}u_W(t)$$

Option 2 - Fit $f_W(t, u(t))$ by integrating to compare to $u(t)$

$$J(W) = \left\| \int_t^{t+h} f_W(t, u(t))dt - u(t) \right\|$$