(Structure-Preserving) Numerical Integration (of Dynamical Systems) using Transformer Neural Networks

Benedikt Brantner, Zeyuan Li, Guillaume de Romemont and Michael Kraus
(benedikt.brantner@ipp.mpg.de)

Max–Planck–Institut für Plasmaphysik
Technische Universität München, Zentrum Mathematik

Reduced Complexity Modelling

Motivation: Parametric PDEs and Solution Manifolds

- multi-query contexts (optimisation, inverse problems, control, ...) require the repeated solution of parametric partial differential equations

- denote the parameter space by $\mathbb{P} \subset \mathbb{R}^p$ and the solution space by V

- parametrised ODE problem (e.g. semi-discretised PDE) for $z \in V$ and $\mu \in \mathbb{P}$

$$F(z(\mu); \mu) = 0$$

- numerical algorithms seek approximate solutions $z_h \approx z$ in finite-dimensional spaces $V_h \approx V$; typically z_h is represented by a degree-of-freedom vector $\hat{z} \in \mathbb{R}^{N_h} \simeq V_h$ where $N_h = \dim V_h$

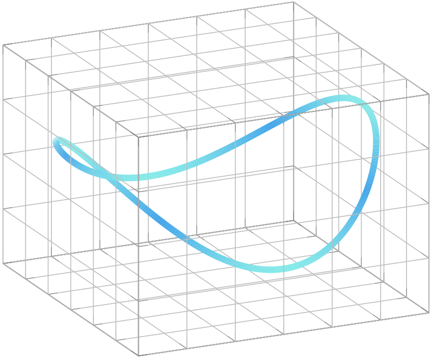
- with traditional numerical methods, the space V_h is typically not adapted to the problem and therefore needs to be rather large, resulting in high computational costs

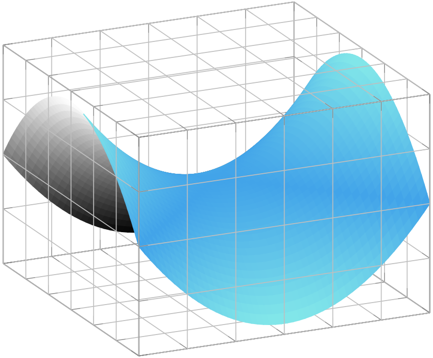
- the actual solution manifold $\mathcal{M}$ is a much smaller space

$$\mathcal{M} = \{ z(\mu) \in V : F(z(\mu); \mu) = 0, \ \mu \in \mathbb{P} \} \subset V_h$$

Motivation: Parametric PDEs and Solution Manifolds

- the solution manifold $\mathcal{M} = \{z(\mu) \in V : F(z(\mu);\mu) = 0,\ \mu \in \mathbb{P}\} \subset V$ is a very small space





- goal: construct an approximation of the solution manifold $\mathcal{M}_h$ and the embedding map $\mathcal{M}_h \hookrightarrow V_h$

Data-driven Model Order Reduction

- Strategy: Learn a low-dimensional representation of a system that captures relevant physical properties
- from a dataset M of solutions $\hat{z}(\mu)$ for different values of the parameter μ construct
  - a mapping $\mathcal{P}$ from V_h to the low-dimensional space V_r (reduction)
  - a mapping $\mathcal{R}$ from the low-dimensional space V_r to V_h (reconstruction)
  - a reduced system of equations $\tilde{F}(\tilde{z}(\mu); \mu) = 0$, i.e. a low dimensional ODE V_μ
- the mappings $\mathcal{P}$ and $\mathcal{R}$ are chosen such that they minimize the reconstruction error:

$$\min_{\mathcal{P},\mathcal{R}} \frac{1}{2} ||\mathsf{M} - \mathcal{R} \circ \mathcal{P}(\mathsf{M})||^2$$

- important properties of the high order model, such as **symplecticity** or conservation of invariants, need to be accounted for in the construction of $\mathcal{P}$, $\mathcal{R}$ and $\tilde{F}$

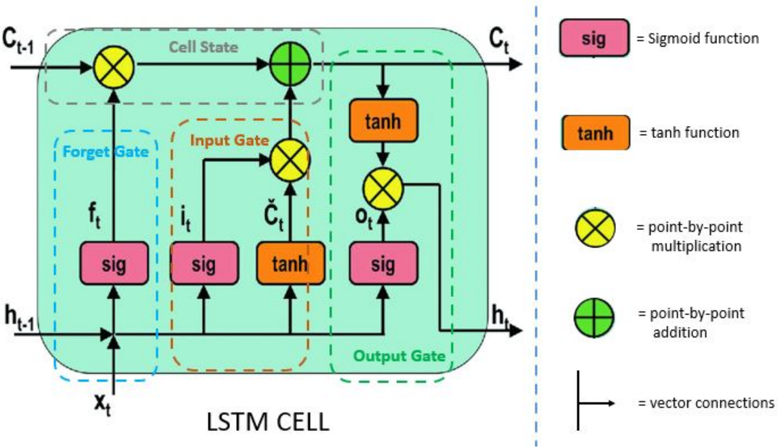
Caveat

$T\mathcal{P} \circ V \circ \mathcal{R}$ is very expensive to evaluate!!!!!

The vector field still has to be evaluated and this is (one of) the most expensive part(s) of the high-order model.

Neural Networks for Learning Low-Dimensional Dynamics

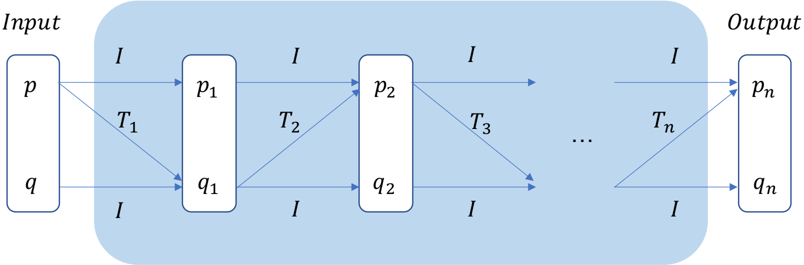
LSTMs



C_{t-1}
Cell State
C_t
Forget Gate
Input Gate
tanh
f_t
i_t
$\check{C}_t$
o_t
sig
sig
tanh
sig
h_{t-1}
h_t
Output Gate
x_t
LSTM CELL
sig
= Sigmoid function
tanh
= tanh function
= point-by-point multiplication
= point-by-point addition
= vector connections

SympNets

- SympNets can approximate arbitrary canonical symplectic maps
- universal model representing the solution of Hamiltonian systems
- no traditional symplectic integration methods required (in contrast to HNNs)
- the SympNet itself provides a time-stepping method for a Hamiltonian system

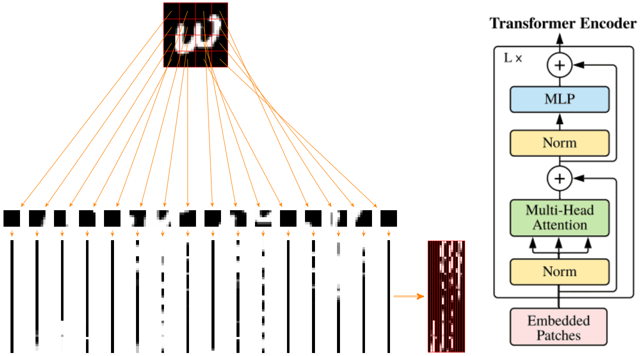


- application: non-intrusive, structure-preserving time integration of reduced systems

P. Jin, Z. Zhang, A. Zhu, Y. Tang, G. E. Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. Neural Networks 132, pp. 166-179, 2020.

Transformers

- Transformers were originally devised for Natural Language Processing (NLP).
- They have gradually replaces LSTMs and other recurrent neural network architectures for these kinds of data and are competing with CNNs in image processing.

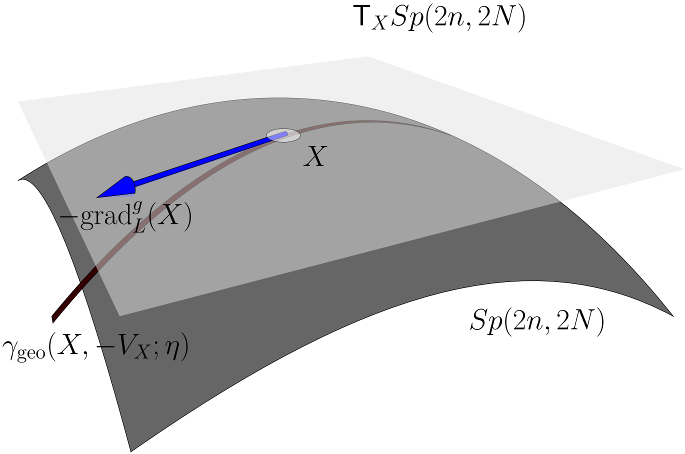


Multihead Attention

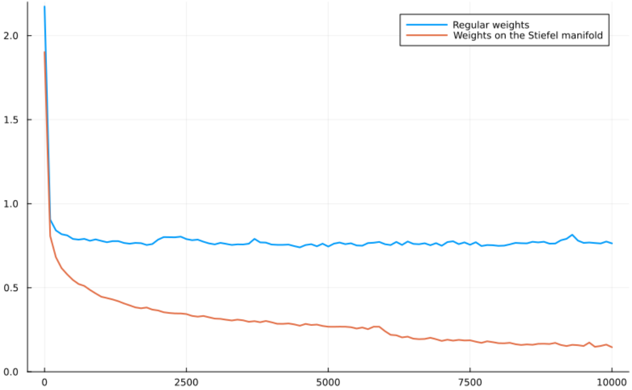
- Can find correlations in the input data.
- $\mathrm{MHA} : X \mapsto [P_1^V X \sigma((P_1^K X)^T P_1^Q X), \ldots, P_{N_h}^V X \sigma((P_{N_h}^K X)^T P_{N_h}^Q X)]$, where the P^V, P^K and P^Q are *projection matrices*.
- More concisely a MHA layer performs N_h single head attention operations: $\mathrm{SHA} : (V, K, Q) \mapsto V\sigma(K^T Q)$, where $\sigma = \mathrm{softmax}$. Therefore:

$$\mathrm{MHA}(X) = [\mathrm{SHA}(X_1^V, X_1^K, X_1^Q), \ldots, \mathrm{SHA}(X_{N_h}^V, X_{N_h}^K, X_{N_h}^Q)] \text{ and } X_i^V = P_i^V X \text{ etc.}$$

Optimization on Manifolds



1. $\mathtt{model} = \mathtt{Lux.Chain(Transformer(patch\_length2, n\_heads, L, Stiefel = false)}$, $\mathtt{Classification(patch\_length2, 10))}$,
2. $\mathtt{model} = \mathtt{Lux.Chain(Transformer(patch\_length2, n\_heads, L, Stiefel = true)}$, $\mathtt{Classification(patch\_length2, 10))}$



Plan for Project

1. Get familiar with the transformer and find what is important for training it.
2. Find an ODE for which transformers are outperforming ResNets (and LSTMs).
3. Make it structure-preserving and see how it compares with SympNets.