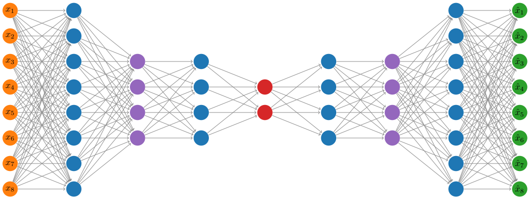
# Structure-Preserving Transformers for Learning Parametrized Hamiltonian Systems


Project Members: Benedikt Brantner[1,2], Guillaume de Romemont[3,4], Zeyuan Li[1,2]

Coordinator: Michael Kraus[1,2]


August 24, 2023


(1) Max–Planck–Institut für Plasmaphysik
(2) Technische Universität München
(3) École nationale supérieure d'arts et métiers
(4) ONERA.

# Outline


Data-Driven Reduced Order Modelling

Hamiltonian Systems

SympNets

Transformers

Structure-Preserving Transformers

Experiments

# Data-Driven Reduced Order Modelling

# Motivation: Parametric PDEs and Solution Manifolds

- multi-query contexts (optimisation, inverse problems, control, ...) require the repeated solution of parametric PDEs.
- parametrised PDE problem for $u \in V$ and $\mu \in \mathbb{P}$

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

- numerical algorithms seek approximate solutions $u_h \approx u$ in finite-dimensional spaces $V_h \approx V$.
- without **model reduction** the space $V_h = \mathbb{R}^{2N}$ must be very large!
- Reduced Order Modelling (ROM) aims at alleviating this cost by using machine learning (ML) techniques.

# Snapshot Matrix

- Construct a **snapshot matrix** $M = [\hat{u}(t_0), \hat{u}(t_1), \ldots, \hat{u}(t_f)]$.
- each column $\hat{u}(t_i)$ of $M$ is a vector $\in V_h$.
- $M$ are the **data** from which the reduced basis is constructed.

$$M = \left[\begin{array}{c:c:c:c} \hat{u}_1(t_0) & \hat{u}_1(t_1) & \ldots & \hat{u}_1(t_f) \\ \hat{u}_2(t_0) & \hat{u}_2(t_1) & \ldots & \hat{u}_2(t_f) \\ \hat{u}_3(t_0) & \hat{u}_3(t_1) & \ldots & \hat{u}_3(t_f) \\ \ldots & \ldots & \ldots & \ldots \\ \hat{u}_{2N}(t_0) & \hat{u}_{2N}(t_1) & \ldots & \hat{u}_{2N}(t_f) \end{array}\right]$$

- Goal: Reduce dimension from $2N$ to $2n$ ($n \ll N$) with mappings $\mathcal{P}$ and $\mathcal{R}$.

# Offline and Online Phase

- Offline phase: construct mappings $\mathcal{P}$ and $\mathcal{R}$ in a data-driven way (ML).
- In the **online phase** the smaller system is solved. This also requires machine learning techniques!

$$
\begin{tikzcd}
\mathrm{FOM}^0 \arrow[d, "\mathcal{P}"] & \mathrm{FOM}^1 \\
\mathrm{ROM}^0 \arrow[r, "\mathcal{N}\mathcal{N}"] & \mathrm{ROM}^1 \arrow[u, "\mathcal{R}"']
\end{tikzcd}
$$

- All maps should be symplectic! The **goal of the project** was to do it for $\mathcal{N}\mathcal{N}$.

# Hamiltonian Systems

# Hamiltonian Systems and Symplectic Maps

- Canonical Hamiltonian systems (Hamiltonian ODE):

$$\dot{q} = \frac{\partial H}{\partial p}, \quad \dot{p} = -\frac{\partial H}{\partial q}, \quad \dot{z} = \mathbb{J}_{2n} \nabla H(z), \quad z = (q, p) \in \mathcal{M} = \mathbb{R}^{2n}, \quad H \in C^{\infty}(\mathbb{R}^{2n})$$

with $\mathbb{J}_{2n}$ being the **canonical symplectic matrix** (anti-symmetric, non-degenerate):

$$\mathbb{J}_{2n} = \begin{pmatrix} \mathbb{0}_n & \mathbb{1}_n \\ -\mathbb{1}_n & \mathbb{0}_n \end{pmatrix}, \qquad \mathbb{J}_{2n}^{-1} = \mathbb{J}_{2n}^T.$$

- Linear Symplectic Mappings ($A \in \mathbb{R}^{2n \times 2n}$):

$$A^T \mathbb{J}_{2n} A = \mathbb{J}_{2n}.$$

- Nonlinear Symplectic Mappings:

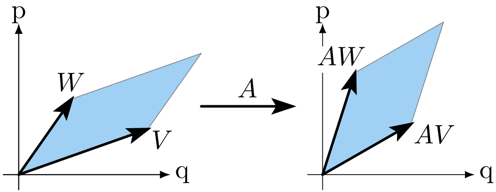
$$\psi : (\mathbb{R}^{2n}, \mathbb{J}_{2n}) \to (\mathbb{R}^{2n}, \mathbb{J}_{2n}) \quad \text{such that} \quad (\nabla_z \psi)^T \mathbb{J}_{2n} (\nabla_z \psi) = \mathbb{J}_{2n}.$$

# Symplectic Maps

- the flow $\phi_H$ of a Hamiltonian system is a symplectic map in phase space

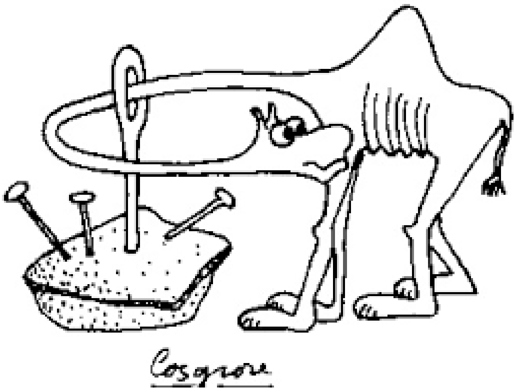
$$(p^1, q^1) = \phi_H(t_1, t_0)(p^0, q^0)$$

- a linear map $A : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is called symplectic if $A^T \mathbb{J}_{2n} A = \mathbb{J}_{2n}$
- a nonlinear map $\phi : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is called symplectic if $(\mathrm{D}\phi)^T \mathbb{J}_{2n} (\mathrm{D}\phi) = \mathbb{J}_{2n}$
- The **flow of a Hamiltonian ODE is symplectic** $\implies$ consequences: preservation of phasespace area as well as higher Poincaré integral invariants
- symplecticity dramatically restricts the number of possible mappings!

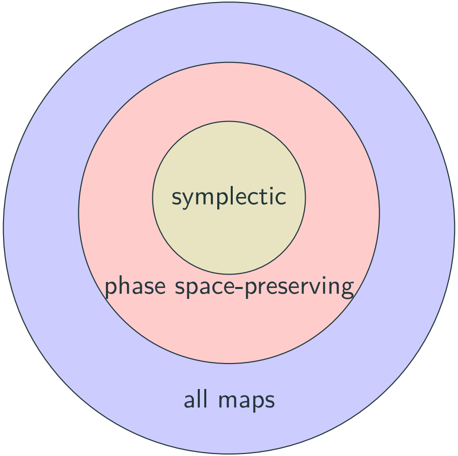


# Gromov's Non-Squeezing Theorem

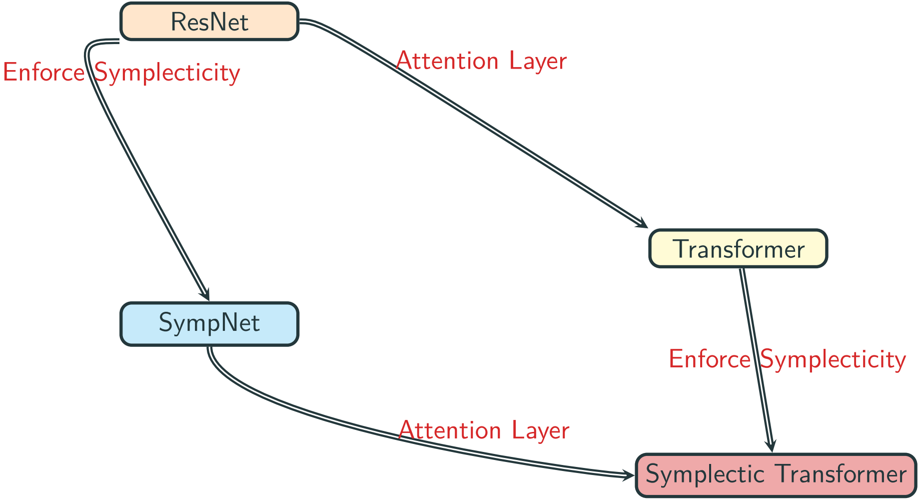
**Theorem**
*A symplectic transformation cannot map a ball* $B(R) = \{z \in \mathbb{R}^{2n} : ||z|| < R\}$ *into a cylinder* $Z(r) = \{z \in \mathbb{R}^{2n} : x_1^2 + y_1^2 < r^2\}$ *if* $r < R$.



**Symplecticity is a strong property that dramatically resricts the number of possible maps!**

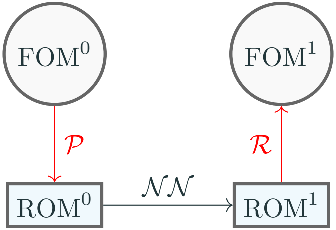


# Parameter Dependence and Symplecticity



# PSD and Symplectic Autoencoders

- Hamiltonian PDE $\implies$ high-dimensional Hamiltonian ODE $\implies$ low-dimensional Hamiltonian ODE $\implies$ high-dimensional Hamiltonian ODE.




- There is a symplectic version of POD: PSD[1].
- Symplectic Autoencoders[2]

[1] Liqian Peng and Kamran Mohseni. "Symplectic model reduction of Hamiltonian systems". In: *SIAM Journal on Scientific Computing* 38.1 (2016), A1–A27.
[2] `GeometricMachineLearning.jl`

# SympNets

# SympNets I[3]

- SympNets can approximate arbitrary canonical symplectic maps
- Each layer of a SympNet only transforms $q$ or $p$ with a map that depends exclusively on the other variable. This preserves the symplectic structure.

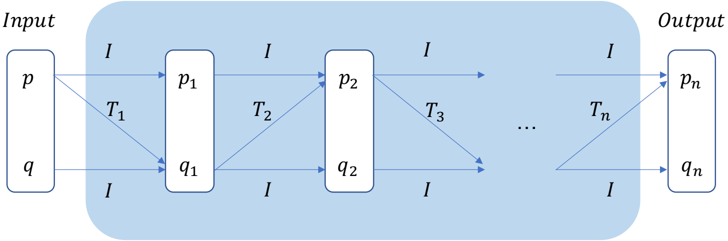



**Figure 1:** Schematic diagram of a SympNet, image taken from Jin et al. [3].

[3]Pengzhan Jin et al. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Networks* 132 (2020), pp. 166–179.

# Universal Approximators

- **Feedforward neural networks are universal approximators!**[4]

- **Sympnets are universal approximators in the set of canonical symplectic maps!**[5]

$$\psi \in \mathrm{Symp}(\mathbb{R}^{2N}) \iff \psi : (\mathbb{R}^{2N}, \mathbb{J}_{2N}) \to (\mathbb{R}^{2N}, \mathbb{J}_{2N}) \ \&\ (\nabla_z \psi)^T \mathbb{J}_{2N} (\nabla_z \psi) = \mathbb{J}_{2N}.$$

[4] Kurt Hornik, Maxwell Stinchombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[5] Pengzhan Jin et al. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Networks* 132 (2020), pp. 166–179.

# SympNets II

- Gradient-type Layers

$$\begin{pmatrix} q \\ p \end{pmatrix} \mapsto \begin{pmatrix} q + K^T \mathrm{diag}(a)\sigma(Kp + b) \\ p \end{pmatrix}, \quad \begin{pmatrix} q \\ p \end{pmatrix} \mapsto \begin{pmatrix} q \\ p + K^T \mathrm{diag}(a)\sigma(Kq + b) \end{pmatrix}$$

  where $K$ is an $M \times n$ matrix, $b$ and $a$ are vectors of size $M$. $M$ can be chosen arbitrarily.
- The collection of all possible combinations of these layers is referred to as *G-SympNets* $\Psi_G$.

*$\Psi_G$ is $r$-uniformly dense on compacta in $\mathcal{SP}^r(U)$, where $U \subset \mathbb{R}^{2N}$.*

This means that for every $f \in \mathcal{SP}^r(U)$, $K \subset U$ compact and $\epsilon > 0$, $\exists \psi \in \Psi_G$ such that $||f - \psi||_{r,K} < \epsilon$ with $||f||_{r,K} = \sum_{|\alpha| \leq r} \max_{1 \leq i \leq N} \sup_{x \in K} |D^\alpha f_i(x)|$.

# Transformers

# Sequential data

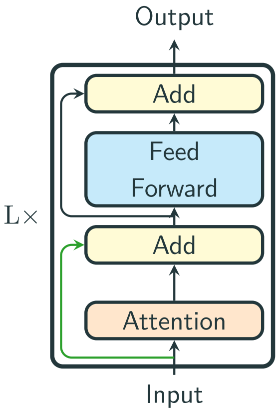
- Sequential input data:

$$\mathcal{P}(M)_{[\mathtt{u:u+T-1}]} = \left[\begin{array}{c:c:c:c} q_1^{(1)} & q_1^{(2)} & \dots & q_1^{(T)} \\ q_2^{(1)} & q_2^{(2)} & \dots & q_2^{(T)} \\ \dots & \dots & \dots & \dots \\ q_n^{(1)} & q_n^{(2)} & \dots & q_n^{(T)} \\ p_1^{(1)} & p_1^{(2)} & \dots & p_1^{(T)} \\ p_2^{(1)} & p_2^{(2)} & \dots & p_2^{(T)} \\ \dots & \dots & \dots & \dots \\ p_n^{(1)} & p_n^{(2)} & \dots & p_n^{(T)} \end{array}\right] =: [z^{(1)}, z^{(2)}, \dots, z^{(T)}] \equiv Z.$$

- Sympnets (and the majority of other neural networks) cannot process sequential data!

# Transformers

- Composition of ResNets[a] and attention[b] layers.
- (Multihead) Attention enables processing time-series data.
- Reweights input vectors based on a learned correlation matrix: $Z \mapsto Z^T A Z$.
- $\mathrm{Attention} : Z \mapsto Z\Lambda(Z)$, where $\Lambda(Z) = \mathrm{softmax}(Z^T A Z)$.
- The softmax is applied column-wise and returns **probability vectors** as output: $[\mathrm{softmax}(C)]_{ij} = e^{c_{ij}} / \left( \sum_{i'=1}^{T} e^{c_{i'j}} \right)$.
- $\mathrm{softmax}(C) =: Y = [y^{(1)}, y^{(2)}, \ldots, y^{(T)}]$ and $[z^{(1)}, \ldots, z^{(T)}]\, Y = [\sum_{i=1}^{T} y_1^{(1)} z^{(i)}, \ldots, \sum_{i=1}^{T} y_1^{(T)} z^{(i)}]$.




[a] $x \mapsto x + \sigma(Ax + b)$
[b] The "Add" connection is optional.

# Attention Layer

**Transformers perform a reweighting of the input based on a learned correlation matrix $\Lambda(Z)$ so that sequential data can be processed!**

- **Correlation matrix** (with learned $A$): $C = Z^T A Z$.
- **Probability vectors:** $\mathrm{softmax}(C) =: Y = [y^{(1)}, y^{(2)}, \ldots, y^{(T)}]$.
- **Reweighting of input:** $[z^{(1)}, \ldots, z^{(T)}]\, Y = [\sum_{i=1}^T y_1^{(1)} z^{(i)}, \ldots, \sum_{i=1}^T y_1^{(T)} z^{(i)}]$.

The last step constitutes a convex combination of the input vectors.

# Structure-Preserving Transformers

# Structure-Preserving Transformer

- ResNet $\rightarrow$ SympNet.
- Symplecticity for $Z \equiv [z^{(i)}, \ldots, z^{(T)}] \in \mathbb{R}^{2n} \times \cdots \times \mathbb{R}^{2n}$. $\implies$ Symplecticity for multistep method.
  - Feng Kang suggests to study the symplecticity for multistep method:

    $$\sum_{j=0}^{k} a_j y_{n+j} = h \sum_{j=0}^{k} b_j f(y_{n+j})$$

    via its underlying one-step method, i.e. step-transition operator.[6]
- The scaled dot-product is not a standard map for input sequence.
- Nonlinear activation function,i.e., softmax, can not preserve the sympleciticity.

"Symplectifying" the attention layer $\implies$ difficult !!!

[6] Kang Feng. "The step-transition operators for multi-step methods of ODE's". In: *Journal of Computational Mathematics* (1998).

# Structure-Preserving Transformer

- Modifying the input sequence format
- Denote the input as $Z = \begin{pmatrix} Q_1 \\ Q_2 \\ P_1 \\ P_2 \end{pmatrix}$

$$
\left[\begin{array}{c:c:c:c}
q_1^{(1)} & q_1^{(2)} & \dots & q_1^{(T)} \\
q_2^{(1)} & q_2^{(2)} & \dots & q_2^{(T)} \\
\dots & \dots & \dots & \dots \\
q_n^{(1)} & q_n^{(2)} & \dots & q_n^{(T)} \\
p_1^{(1)} & p_1^{(2)} & \dots & p_1^{(T)} \\
p_2^{(1)} & p_2^{(2)} & \dots & p_2^{(T)} \\
\dots & \dots & \dots & \dots \\
p_n^{(1)} & p_n^{(2)} & \dots & p_n^{(T)}
\end{array}\right]
\mapsto
\begin{bmatrix}
q_1^{(1)} \\ q_1^{(2)} \\ \dots \\ q_1^{(T)} \\ q_2^{(1)} \\ \dots \\ q_n^{(T)} \\ p_1^{(1)} \\ \dots \\ p_1^{(T)} \\ p_2^{(1)} \\ \dots \\ p_n^{(T)}
\end{bmatrix}
$$

# New Activation Function for the Attention Layer

- We replace the softmax activation function:

$$\sigma(C) = \mathrm{Cayley}(\texttt{upper\_triangular\_asymmetrize}(C))$$

- $$[\texttt{upper\_triangular\_asymmetrize}(C)]_{ij} = \begin{cases} c_{ij} & \text{if } i < j \\ -c_{ji} & \text{if } i > j \\ 0 & \text{else.} \end{cases}$$
- $\mathrm{Cayley}(Y) = (\mathbb{I}_T - Y)(\mathbb{I}_T + Y)^{-1}$.
  They Cayley transform maps skew-symmetric matrices to orthonormal matrices!
- $\Lambda(Z) = \sigma(Z^T A Z)$

# The Attention Matrix

- Apply $\Lambda(Z)$ to the big vector Z!
- $\Lambda(Z)$ **orthonormal** $\implies$ $\tilde{\Lambda}(Z)$ symplectic, i.e.

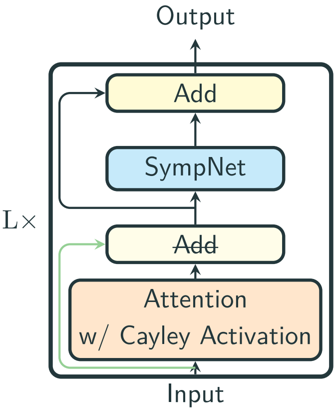
$$\tilde{\Lambda}(Z)^T \mathbb{J}_{2nT} \tilde{\Lambda}(Z) = \mathbb{J}_{2nT},$$

where we define a **big symplectic matrix**
$\mathbb{J}_{2nT} = \begin{pmatrix} \mathbb{O} & \mathbb{I}_{nT} \\ -\mathbb{I}_{nT} & \mathbb{O} \end{pmatrix}$.

$$\tilde{\Lambda}(Z)Z := \begin{pmatrix} \Lambda(Z) & \mathbb{O} & \cdots & \mathbb{O} \\ \mathbb{O} & \Lambda(Z) & \cdots & \mathbb{O} \\ \cdots & \cdots & \ddots & \cdots \\ \mathbb{O} & \mathbb{O} & \cdots & \Lambda(Z) \end{pmatrix} \begin{bmatrix} q_1^{(1)} \\ q_1^{(2)} \\ \cdots \\ q_1^{(T)} \\ q_2^{(1)} \\ \cdots \\ q_n^{(T)} \\ p_1^{(1)} \\ \cdots \\ p_1^{(T)} \\ p_2^{(1)} \\ \cdots \\ p_n^{(T)} \end{bmatrix}$$

# Architecture

1. Softmax → Cayley activation.
2. Remove "add connection" after attention layer.
3. Feedforward → SympNet.



```
model = Chain(Dense(sys_dim, transformer_dim, tanh),
              MultiHeadAttention(transformer_dim, num_heads),
              ResNet(transformer_dim, tanh),
              MultiHeadAttention(transformer_dim, num_heads),
              ResNet(transformer_dim, tanh),
              Dense(transformer_dim, sys_dim, identity)
              )

loss(ps) = loss(model, ps)
ps = initialparameters(CUDABAckend(), Float32, model)

o = Optimizer(AdamOptimizer(), ps)

n_training_steps_per_epoch = Int(ceil(n_time_steps/batch_size))
n_training_steps = n_epochs*n_training_steps_per_epoch

progress_object = Progress(n_training_steps; enabled=true)

for t in 1:n_training_steps
    draw_batch!(batch, output, seq_length, prediction_window)
    loss_val, pullback = Zygote.pullback(loss, ps)
    dx = pullback(1)[1]
    optimization_step!(o, model, ps, dx)
    ProgressMeter.next!(progress_object; showvalues = [(:TrainingLoss,loss_val)])
end
```

# Experiments

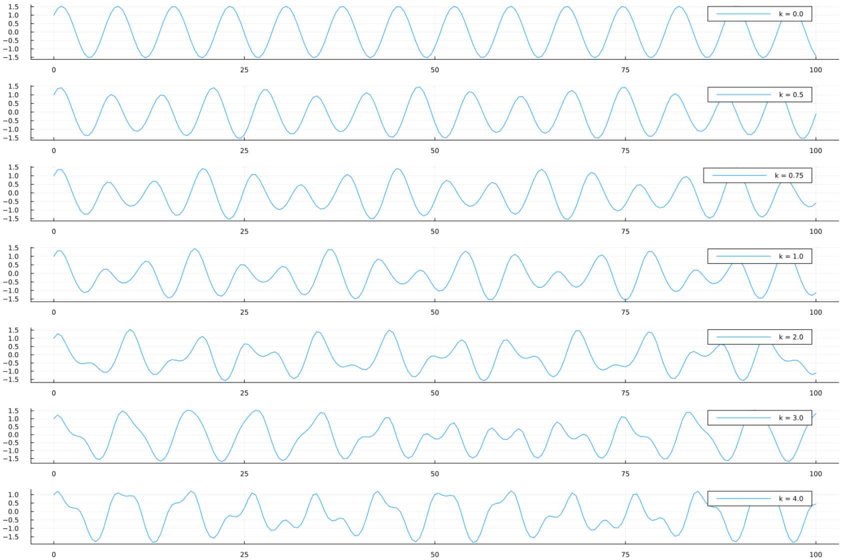
# System of Coupled Harmonic Oscillators

$$H(q_1, q_2, p_1, p_2) = \frac{p_1^2}{2m_1} + \frac{p_2^2}{2m_2} + k_1 \frac{q_1^2}{2} + k_2 \frac{q_2^2}{2} + k\sigma(q_1)\frac{(q_1 - q_2)^2}{2},$$

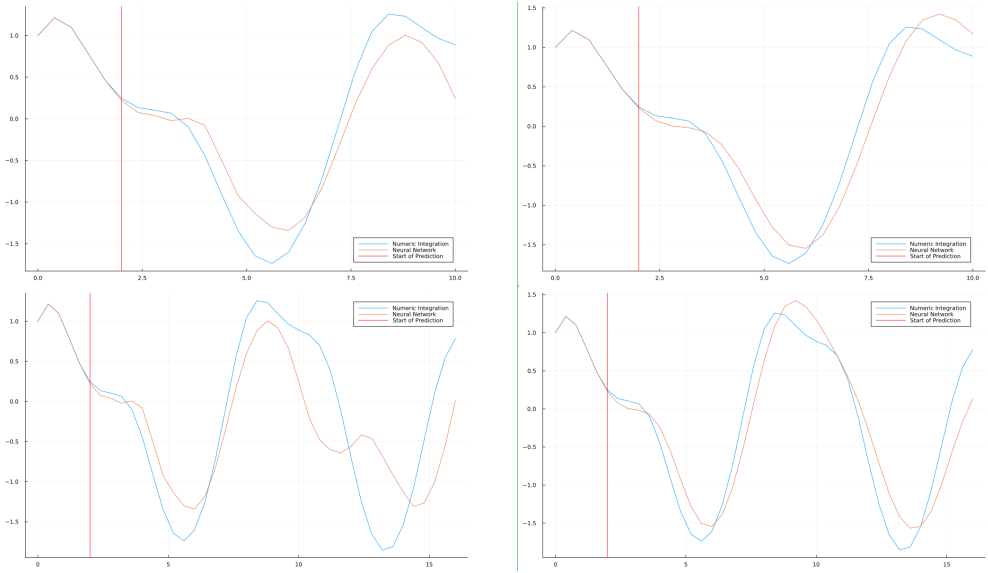
with $\sigma(x) = 1/(1 + e^{-x})$.

Changing $k$ changes the shapes of the trajectories $\implies$ test bed for Transformer!

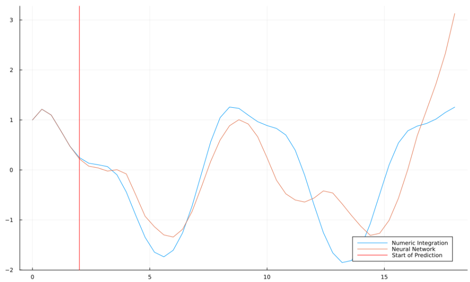
# Trajectories for different values of $k$ and same initial conditions

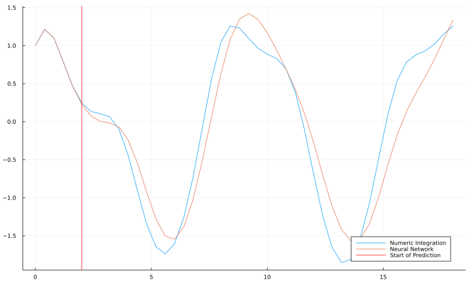


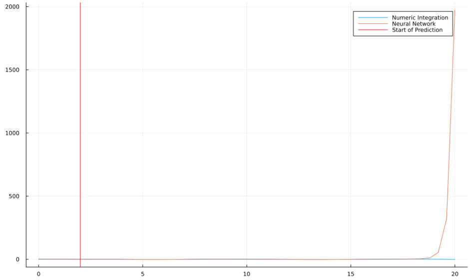
# ResNet v SympNet I

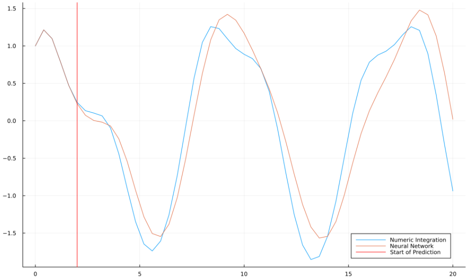


# ResNet v SympNet II



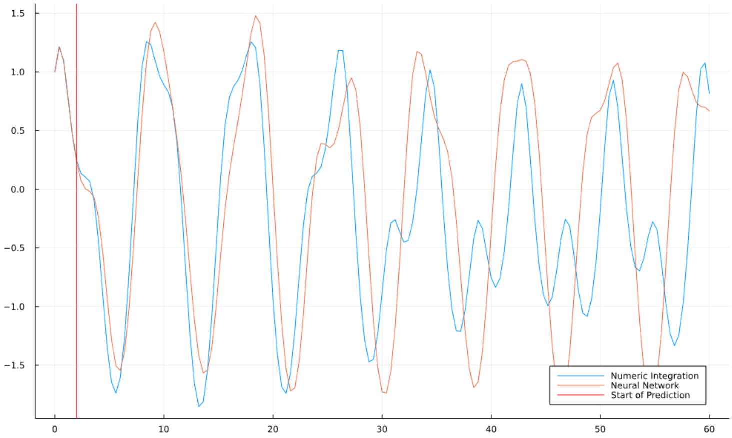



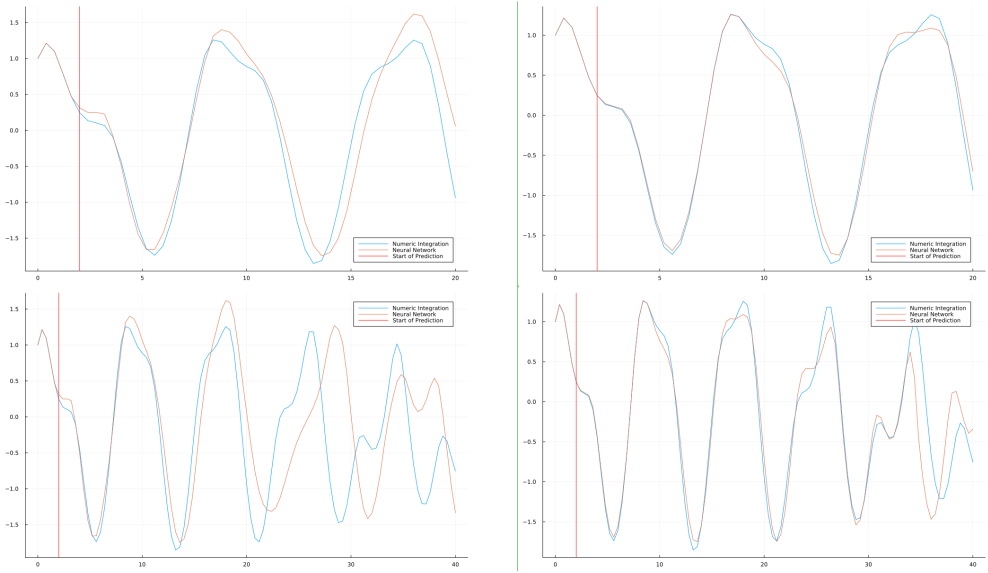



# SympNets don't work here

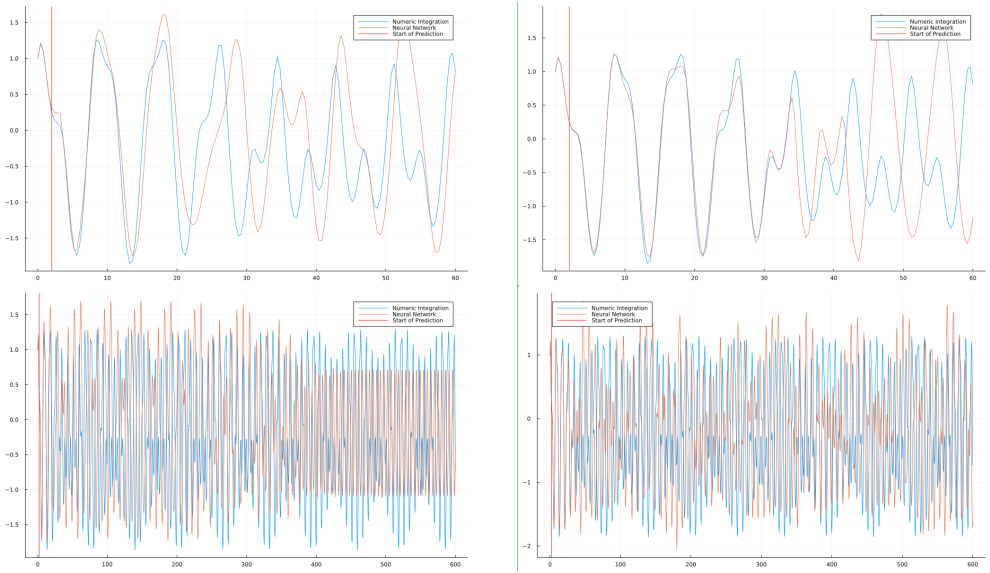
SympNets give guaranteed long-time stability but cannot process data coming from different parameters!



# Transformer v Structure-Preserving Transformer I



# Transformer v Structure-Preserving Transfromer II



# Drawback(s) and Next Steps

- Parallelization of the activation function. The new architecture is slower than the transformer by a factor of about 10! [7]
- Using the structure-preserving transformer in the online stage of a ROM model.
- Investigating paths to making it truly symplectic or checking if the current architecture is sufficient (perhabs it is "conjugate symplectic"[8]).

[7] Approx. 30min for training as opposed to 3min on GeForce RTX 4090.
[8] Robert I McLachlan and Christian Offen. "Backward error analysis for conjugate symplectic methods". In: *arXiv preprint arXiv:2201.03911* (2022).

# References


[1] Kang Feng. "The step-transition operators for multi-step methods of ODE's". In: *Journal of Computational Mathematics* (1998).

[2] Kurt Hornik, Maxwell Stinchombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[3] Pengzhan Jin et al. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Networks* 132 (2020), pp. 166–179.

[4] Robert I McLachlan and Christian Offen. "Backward error analysis for conjugate symplectic methods". In: *arXiv preprint arXiv:2201.03911* (2022).

[5] Liqian Peng and Kamran Mohseni. "Symplectic model reduction of Hamiltonian systems". In: *SIAM Journal on Scientific Computing* 38.1 (2016), A1–A27.