Modelling of Highly oscillatory phenomenon by neural networks

Maxime BOUCHEREAU - Université de Rennes (IRMAR)

May 24, 2023
GTT - Paris

Ph. D supervisors:
Francois CASTELLA - Philippe CHARTIER
Mohammed LEMOU - Florian MEHATS

Goal: solve highly oscillatory ODE's, of the form:

$$\begin{cases} \dot{y}^\varepsilon(t) & = & f\left(\frac{t}{\varepsilon}, y^\varepsilon(t)\right) \\ y^\varepsilon(0) & = & y_0 \end{cases} \tag{1}$$

where $\tau \mapsto f(\tau, \cdot)$ is 2π-periodic, by using numerical methods performed by machine learning. ε is a small parameter.

Main tools used:

- Function approximations by neural networks and structure preservation
- Modified field theory for autonomous ODE's
- Averaging theory & Numerical methods for highly oscillatory ODE's

Introduction

Definition (Neural network - MLP)

*A **Multi-Layer Perceptron (MLP)**, is a mapping* $\mathcal{N} : \mathbb{R}^{d_0} \longrightarrow \mathbb{R}^{d_L}$ *given, for all* $x \in \mathbb{R}^{d_0}$*, by:*

$$\mathcal{N}(x) = W_L \cdot \Sigma\left(\cdots W_1 \cdot \Sigma\left(W_0 \cdot x + b_0\right) + b_1 \cdots\right) + b_L \tag{2}$$

where:

- $L+1$ *is the number of **layers**. **Shallow network:*** $L = 1$*, **Deep network:*** $L \geqslant 2$*. Layers* 1 *to* $L-1$ *are named **hidden layers**.*
- $b_0 \in \mathbb{R}^{d_0}, b_1 \in \mathbb{R}^{d_1}, \ldots, b_L \in \mathbb{R}^{d_L}$ *are the **bias**.*
- $W_0 \in \mathcal{M}_{d_1,d_0}(\mathbb{R}), W_1 \in \mathcal{M}_{d_2,d_1}(\mathbb{R}), \ldots, W_L \in \mathcal{M}_{d_L,d_{L-1}}(\mathbb{R})$ *are the **weights**. Lines of* W_i*'s are **neurons**.*
- $\Sigma(y_1, \ldots, y_d) = (\sigma(y_1), \ldots, \sigma(y_d))$ *is a componant-wise nonlinear mapping* σ*, e.g.* tanh*, named **activation function**.*

Theorem (Universal approximation)

Let $f \in \mathcal{C}^0(\Omega, \mathbb{R}^k)$ where $\Omega \subset \mathbb{R}^d$ is compact. Then, for all $\varepsilon > 0$, there exists $\mathcal{N} : \mathbb{R}^d \longrightarrow \mathbb{R}^k$ a MLP s.t.

$$||f - \mathcal{N}||_{L^\infty(\Omega)} \leqslant \varepsilon \tag{3}$$

Rate of convergence w.r.t. number of weights:

- **Polynomial decay ($L = 1$):** Anastassiou, G. *Quantitative approximations*. Chapman and Hall/CRC, 2000.
- **Polynomial-Exponential decay ($L = 3$):** De Ryck, T., Lanthaler, S., & Mishra, S. (2021). On the approximation of functions by tanh neural networks. Neural Networks, 143, 732-750.

Structure preservation. Example: hamiltonian structure of the neural network. For all $x \in \mathbb{R}^{2d}$

$$\mathcal{N}(x) = J\nabla\mathcal{H}(x) \tag{4}$$

where $\mathcal{H} : \mathbb{R}^{2d} \longrightarrow \mathbb{R}$ is a MLP.

- **Hamiltonian structure (HNN):** David, M., Méhats, F. *Symplectic learning for Hamiltonian neural networks.* arXiv preprint arXiv:2106.11753, 2021.
- **Free-divergence structure (VP-Nets):** Zhu, A., Zhu, B., Zhang, J., Tang, Y., Liu, J. *VPNets: Volume-preserving neural networks for learning source-free dynamics.* arXiv preprint arXiv:2204.13843, 2022.

Autonomous case: f is independant from τ.

Definition (Modified field w.r.t. a numerical method)

Let consider a one step numerical method $\Phi_h(\cdot)$. The ***modified vector field w.r.t.*** Φ_h*, denoted $\tilde{f}_h$, is defined by the relation:*

$$\varphi_{nh}^{f}(y_0) \quad = \quad \left(\Phi_h^{\tilde{f}_h}\right)^n (y_0) \tag{5}$$

Example: Forward Euler scheme, linear ODE: $\dot{y}(t) = ay(t)$, $f(y) = ay$.

$$y(nh) \quad = \quad e^{anh} y_0 \quad = \quad \left(1 + h \cdot \frac{e^{ha} - 1}{h}\right)^n y_0 \tag{6}$$

thus $\tilde{f}_h(y) = \left(\frac{e^{ha}-1}{h}\right) y$

Proposition (Properties of the modified field)

- ***Formal serie w.r.t.*** *h: If Φ is of order p, then* $\tilde{f}_h(y) = f(y) + h^p f^{[1]}(y) + h^{p+1} f^{[2]}(y) + \cdots$
- ***Hamiltonian structure:*** *If f is hamiltonian, then f_i's and $\tilde{f}_h$ are hamiltonian.*

Backward error analysis: Hairer, E., Lubich, C., Wanner, G. *Geometric Numerical integration: structure-preserving algorithms for ordinary differential equations.* Springer, 2006.

Theorem (Solution of highly oscillatory ODE)

For all $t \in \mathbb{R}$:

$$y^\varepsilon(t) \quad = \quad \phi^\varepsilon_{\frac{t}{\varepsilon}}\left(\varphi_t^{F^\varepsilon}(y_0)\right) \tag{7}$$

where:

- F^ε *is called* ***averaged field****. Structure:* $F^\varepsilon(y) = \langle f\rangle(y) + \varepsilon F^{[1]}(y) + \varepsilon^2 F^{[2]}(y) + \cdots$, *where* $\langle f\rangle$ *is the average field w.r.t. time variable:*

$$\langle f\rangle(y) \quad := \quad \frac{1}{2\pi}\int_0^{2\pi} f(\tau, y)\mathrm{d}\tau \tag{8}$$

- $\phi^\varepsilon_\tau(y) = y + \varepsilon \cdot G^\varepsilon(\tau, y)$ *(Near to identity mapping) and is* 2π*-periodic w.r.t.* τ.
- ***Hamiltonian structure:*** *If* f *is hamiltonian w.r.t.* y, *then* F^ε, $\langle f\rangle$ *and* F_i*'s are hamiltonian,* ϕ *is symplectic w.r.t.* y.

Theorem (P. Chartier, M. Lemou, F. Méhats, G. Vilmart - 2020)

*There exists a numerical method of order r, named **uniformly accurate method**, $\Phi_h(\cdot)$, s.t.*

$$\underset{0\leqslant n\leqslant N}{Max} \left|\left(\Phi_h\right)^n(y_0) - y^\varepsilon(nh)\right| \leqslant Ch^r \tag{9}$$

where $h = \frac{T}{N}$ and the constant C is independant from ε.

Uniformy accurate methods: Chartier, P., Lemou, M., Méhats, F., & Vilmart, G. (2020). *A new class of uniformly accurate numerical schemes for highly oscillatory evolution equations*. Foundations of Computational Mathematics, 20, 1-33.

Autonomous ODE's

- **Autonomous ODE:**

$$\begin{cases} \dot{y}(t) & = & f\left(y(t)\right) \\ y(0) & = & y_0 \end{cases} \tag{10}$$

- **Numerical method:** $\Phi_h(\cdot)$, assumed to be of order p.
- **Goal:** Approximate the modified field $\tilde{f}_h$ by a neural network $f_{app}(\cdot, h)$ in order to get approximated solution $y_n^* = \left(\Phi_h^{f_{app}(\cdot,h)}\right)^n (y_0)$ very close to the exact solution $y(nh)$.

- **Structure of f_{app}: Sum of N_t terms**

$$\begin{aligned} f_{app}(y,h) &= f(y) + h^p f_1(y) + h^{p+1} f_2(y) + \cdots \\ &+ h^{N_t+p-2} f_{N_t-2}(y) + h^{N_t+p-1} R_a(y,h) \end{aligned}$$

- **Data creation:** Computation of exact solutions $y_1^{(k)} = \varphi^f_{h^{(k)}}(y_0^{(k)})$ with accurate and expensive integrator, where $y_0^{(k)}$ is randomly selected in the compact set $\Omega \subset \mathbb{R}^d$, $h^{(k)}$ is randomly selected in $[h_-, h_+]$, for all $0 \leqslant k \leqslant K-1$

- **Training of the neural network:** Optimization of:

$$Loss_{Train} = \frac{1}{K_0}\sum_{k=0}^{K_0-1}\frac{1}{{h^{(k)}}^{2p+2}}\left|\underbrace{\Phi_{h^{(k)}}^{f_{app}(\cdot,h^{(k)})}\left(y_0^{(k)}\right)}_{=\hat{y_1}^{(k)}}-y_1^{(k)}\right|^2$$

- **Good training:** $Loss_{Train}$ has the same decay pattern than:

$$Loss_{Test} = \frac{1}{K-K_0}\sum_{k=K_0}^{K-1}\frac{1}{{h^{(k)}}^{2p+2}}\left|\Phi_{h^{(k)}}^{f_{app}(\cdot,h^{(k)})}\left(y_0^{(k)}\right)-y_1^{(k)}\right|^2$$

- **Numerical integration:** $f_{app}(\cdot,h)$ is an accurate approximation of $\tilde{f}_h$, thus we get a small numerical error:

$$e_n^* \quad = \quad \left(\Phi_h^{f_{app}(\cdot,h)}\right)^n (y_0) - \varphi_{nh}^f(y_0) \tag{11}$$

Denoting the **learning error** by

$$\delta \quad := \quad \underset{(y,h)\in\Omega\times[h_-,h_+]}{\mathrm{Max}} \frac{\left|\tilde{f}_h(y,h) - f_{app}(y,h)\right|}{h^p} \tag{12}$$

Theorem (M.Bouchereau, P.Chartier, M.Lemou, F.Méhats - 2023[1])

Assuming that

- *For any pair smooth vector fields f_1 and f_2, we have*

$$\forall 0 \leq h \leq h_+, \quad \left|\left|\Phi_h^{f_1} - \Phi_h^{f_2}\right|\right|_{L^\infty(\Omega)} \leqslant Ch\,||f_1 - f_2||_{L^\infty(\Omega)} \tag{13}$$

  for some positive constant C, independent of f_1 and f_2;

- *For any smooth vector field f, there exists a constant $L > 0$ such that $\forall 0 \leq h \leq h_+, \forall (y_1, y_2) \in \Omega^2$:*

$$\left|\Phi_h^f(y_1) - \Phi_h^f(y_2)\right| \leqslant (1 + Lh)\,|y_1 - y_2|\,. \tag{14}$$

Then there exist two constants $\tilde{C}, \tilde{L} > 0$ such that:

$$\underset{0 \leqslant n \leqslant N}{Max} |e_n^*| \leqslant \frac{C\delta h^p}{\tilde{L}} \left(e^{\tilde{L}T} - 1\right) \tag{15}$$

[1] Bouchereau, M., Chartier, P., Lemou, M., & Méhats, F. (2023). *Machine Learning Methods for Autonomous Ordinary Differential Equations*. arXiv preprint arXiv:2304.09036.

$$\begin{cases} \dot{y_1} &= \left(\frac{1}{I_3} - \frac{1}{I_2}\right) y_2 y_3 \\ \dot{y_2} &= \left(\frac{1}{I_1} - \frac{1}{I_3}\right) y_1 y_3 \\ \dot{y_3} &= \left(\frac{1}{I_2} - \frac{1}{I_1}\right) y_1 y_2 \end{cases},$$

with $(I_1, I_2, I_3) = (1, 2, 3)$.

Parameters	
# Math Parameters:	
Interval where time steps are selected:	$[h_-, h_+] = [0.5, 2.5]$
Time for ODE simulation:	$T = 20$
Time step for ODE simulation:	$h = 0.5$
# AI Parameters:	
Domain where data are selected:	$\Omega = \left\{x \in [-2, 2]^2 : 0.98 \leqslant \lvert x \rvert \leqslant 1.02\right\}$
Number of data:	$K = 100\,000\,000$
Proportion of data for training:	80% - $K_0 = 80\,000\,000$
Number of terms in the perturbation (MLP's):	$N_t = 1$
Hidden layers per MLP:	2
Neurons on each hidden layer:	250
Epochs:	200

Computational time for training: 1 Day 21 h 59 min 51 s

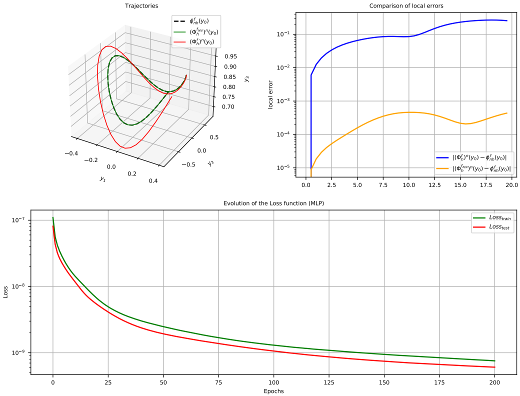

Figure: Comparison between $Loss$ decays (green: $Loss_{Train}$, red: $Loss_{Test}$), trajectories (dashed dark: exact flow, red: numerical flow with f, green: numerical flow with $f_{app}(\cdot, h)$) and local error (blue: exact flow and numerical flow with f, yellow: exact and numerical flow with $f_{app}(\cdot, h)$)

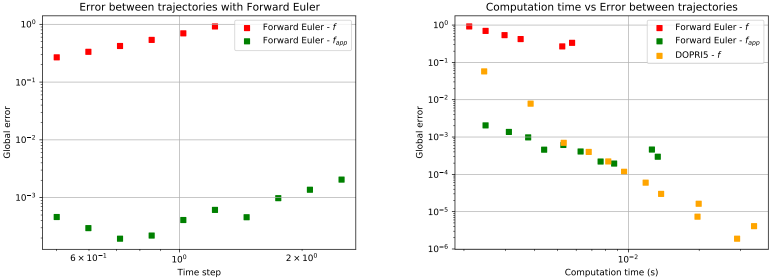

Figure: Left: Integration errors (green: integration with f, red: integration with $f_{app}(\cdot, h)$). Right: Comparison between computational time and integration error (red: numerical method with f, green: integration with $f_{app}(\cdot, h)$, yellow: integration with DOPRI5).

$$\begin{cases} \dot{y}_1 & = & -\sin(y_2) \\ \dot{y}_2 & = & y_1 \end{cases},$$

Hamiltonian function:

$$H : y \quad \mapsto \quad \frac{1}{2}y_1^2 + (1 - \cos(y_2)) \tag{16}$$

Parameters	
# Math Parameters:	
Interval where time steps are selected:	$[h_-, h_+] = [0.05, 0.5]$
Time for ODE simulation:	$T = 20$
Time step for ODE simulation:	$h = 0.25$
# AI Parameters:	
Domain where data are selected:	$\Omega = [-2, 2]^2$
Number of data:	$K = 20\,000\,000$
Proportion of data for training:	80% - $K_0 = 16\,000\,000$
Number of terms in the perturbation (MLP's):	$N_t = 1$
Hidden layers per MLP:	2
Neurons on each hidden layer:	200
Epochs:	200

Computational time for training: 9 h 47 min 51 s

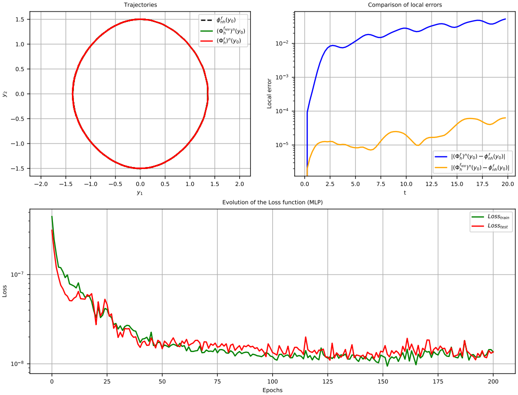

Figure: Comparison between *Loss* decays (green: $Loss_{Train}$, red: $Loss_{Test}$), trajectories (dashed dark: exact flow, red: numerical flow with f, green: numerical flow with $f_{app}(\cdot, h)$) and local error (blue: exact flow and numerical flow with f, yellow: exact and numerical flow with $f_{app}(\cdot, h)$)

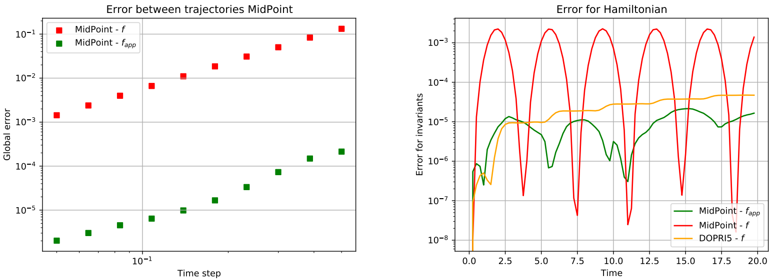

Figure: Left: Integration errors (green: integration with f, red: integration with $f_{app}(\cdot, h)$). Right: Evolution of the error between Hamiltonian $H : y \mapsto (1 - \cos(y_2)) + \frac{1}{2}y_1^2$ over the numerical flow and Hamiltonian at $t = 0$, $H(y_0)$.

Highly oscillatory ODE's

- **Goal:** Find an approximation of the solution of

$$\left\{\begin{array}{lcl} \dot{y}^{\varepsilon}(t) & = & f\left(\frac{t}{\varepsilon}, y^{\varepsilon}(t)\right) \\ y^{\varepsilon}(0) & = & y_0 \end{array}\right. \tag{17}$$

- **General strategy:** Use the decomposition

$$y^{\varepsilon}(t) \quad = \quad \phi^{\varepsilon}_{\frac{t}{\varepsilon}}\left(\varphi^{F^{\varepsilon}}_{t}(y_0)\right) \tag{18}$$

in order to approximate F^{ε} and $\phi^{\varepsilon}_{\cdot}(\cdot)$ with neural networks, denoted F^{ε}_{app} and $G^{\varepsilon}_{app}(\cdot,\cdot)$.

- **Structure of F_{app}^{ε}: Sum of $N_t + 2$ terms.**

$$
\begin{aligned}
F_{app}^{\varepsilon}(y) &= F_0(y) + \varepsilon F_1(y) + \varepsilon^2 F_2(y) + \cdots \\
&+ \varepsilon^{N_t} F_{N_t}(y) + \varepsilon^{N_t+1} R(y, \varepsilon)
\end{aligned}
$$

- **Structure of G_{app}^{ε}: Identity & truncated Fourier serie.**

$$
\begin{aligned}
G_{app}^{\varepsilon}(y, \tau) &= y + \varepsilon \Big[H_{app}^{1,s}(y, \varepsilon) \sin(\tau) + \cdots + H_{app}^{N,s}(y, \varepsilon) \sin(N\tau) \\
&+ H_{app}^{1,c}(y, \varepsilon) (\cos(\tau) - 1) + \cdots + H_{app}^{N,c}(y, \varepsilon) (\cos(N\tau) - 1) \Big],
\end{aligned}
$$

- **Data creation:** Computation of exact solutions $y_1^{(k)} = \varphi_{h^{(k)}}^{f}(y_0^{(k)})$ with accurate and expensive integrator, where $y_0^{(k)}$ is randomly selected in the compact set $\Omega \subset \mathbb{R}^d$, $h^{(k)}$ is randomly selected in $[h_-, h_+]$, for all $0 \leqslant k \leqslant K - 1$

- **First learning:** Approximation of F^ε by F^ε_{app}
  - **Data creation:** Computation of exact solutions at time $t = 2\pi\varepsilon^{(k)}$ (stroboscopic time)

$$y_1^{(k)} = \varphi^f_{2\pi\varepsilon^{(k)}}(y_0^{(k)}) = \phi^\varepsilon_{2\pi}\left(\varphi^{F^{\varepsilon^{(k)}}}_{2\pi\varepsilon^{(k)}}\left(y_0^{(k)}\right)\right) = \varphi^{F^{\varepsilon^{(k)}}}_{2\pi\varepsilon^{(k)}}\left(y_0^{(k)}\right)$$

  with accurate integrator where $y_0^{(k)}$ is randomly selected in the compact set $\Omega \subset \mathbb{R}^d$, $\varepsilon^{(k)}$ is randomly selected in $[\varepsilon_-, \varepsilon_+]$, for all $0 \leqslant k \leqslant K-1$
  - **Loss optimization:** If we consider a numerical method of order p denoted $\Phi_\cdot(\cdot)$, then we optimize the $Loss_{Train}$ function

$$Loss_{Train} = \frac{1}{K_0}\sum_{k=0}^{K_0-1}\left(\frac{1}{2\pi\varepsilon^{(k)}}\right)^2\left|\Phi^{F^{\varepsilon^{(k)}}_{app}}_{2\pi\varepsilon^{(k)}}\left(y_0^{(k)}\right) - y_1^{(k)}\right|^2$$

  - We get F^ε_{app} as an approximation of the modified field $F^{\tilde{\varepsilon}}_{2\pi\varepsilon} = F^\varepsilon + \mathcal{O}(\varepsilon^p)$ with a learning error δ

- **Second learning:** Approximation of F^{ε} by F^{ε}_{app}
  - **Data creation:** Computation of exact solutions at time $t = \tau^{(k)}\varepsilon^{(k)}$

$$y_2^{(k)} = \varphi^{f}_{\tau^{(k)}\varepsilon^{(k)}}(y_0^{(k)}) = \phi^{\varepsilon^{(k)}}_{\tau^{(k)}}\left(\varphi^{F^{\varepsilon^{(k)}}}_{\tau^{(k)}\varepsilon^{(k)}}\left(y_0^{(k)}\right)\right)$$

and exact flow associated to $F^{\varepsilon^{(k)}}_{app}$ at time $t = \tau^{(k)}\varepsilon^{(k)}$

$$z_0^{(k)} = \varphi^{F^{\varepsilon^{(k)}}_{app}}_{\tau^{(k)}\varepsilon^{(k)}}\left(y_0^{(k)}\right) \tag{19}$$

with accurate integrator where $y_0^{(k)}$ is randomly selected in the compact set $\Omega \subset \mathbb{R}^d$, $\varepsilon^{(k)}$ is randomly selected in $[\varepsilon_-, \varepsilon_+]$, $\tau^{(k)}$ is randomly selected in $[0, 2\pi]$, for all $0 \leqslant k \leqslant K' - 1$
  - **Loss optimization:** If we consider a numerical method of order p denoted $\Phi.(\cdot)$, then we optimize the $Loss_{Train}$ function

$$Loss_{Train} = \frac{1}{K_0'}\sum_{k=0}^{K_0'-1}\left|G^{\varepsilon^{(k)}}_{app}\left(\tau^{(k)}, z_0^{(k)}\right) - y_2^{(k)}\right|^2$$

  - We get $G^{\varepsilon}_{app}(\cdot,\cdot)$ as an approximation of the map ϕ with a learning error δ'.

We get an approximated solution of the ODE:

$$y^*(t) \quad = \quad G^{\varepsilon}_{app}\left(\frac{t}{\varepsilon}, \varphi_t^{F^{\varepsilon}_{app}}(y_0)\right) \tag{20}$$

Proposition

For $T > 0$, there exists a constant $C > 0$ s.t.

$$\underset{0 \leqslant t \leqslant T}{Max} |y^*(t) - y^{\varepsilon}(t)| \quad \leqslant \quad C\left(\delta + \delta'\right) \tag{21}$$

$$\begin{cases} \dot{q} & = & p \\ \dot{p} & = & -q + \varepsilon \left(\frac{1}{4} - q^2 \right) p \end{cases} \tag{22}$$

by the first variable change $t \mapsto \frac{t}{\varepsilon}$ and then the second variable change, i.e. multiplication by the matrix:

$$\begin{bmatrix} \cos\left(\frac{t}{\varepsilon}\right) & -\sin\left(\frac{t}{\varepsilon}\right) \\ \sin\left(\frac{t}{\varepsilon}\right) & \cos\left(\frac{t}{\varepsilon}\right) \end{bmatrix}$$

we get the system:

$$\begin{cases} \dot{y_1}(t) & = & -\sin\left(\frac{t}{\varepsilon}\right)\left[\frac{1}{4} - \left(y_1(t)\cos\left(\frac{t}{\varepsilon}\right) + y_2(t)\sin\left(\frac{t}{\varepsilon}\right)\right)^2\right]\left[-y_1(t)\sin\left(\frac{t}{\varepsilon}\right) + y_2(t)\cos\left(\frac{t}{\varepsilon}\right)\right] \\ \dot{y_2}(t) & = & \cos\left(\frac{t}{\varepsilon}\right)\left[\frac{1}{4} - \left(y_1(t)\cos\left(\frac{t}{\varepsilon}\right) + y_2(t)\sin\left(\frac{t}{\varepsilon}\right)\right)^2\right]\left[-y_1(t)\sin\left(\frac{t}{\varepsilon}\right) + y_2(t)\cos\left(\frac{t}{\varepsilon}\right)\right] \end{cases}$$

Parameters	
# Math Parameters:	
Interval small parameters ε are selected:	$[\varepsilon_-, \varepsilon_+] = [0.01, 0.1]$
Time for ODE simulation:	$T = 10$
Small parameter for ODE simulation:	$\varepsilon = 0.1$
# AI Parameters:	
Domain where data are selected (both training):	$\Omega = [-2, 2]^2$
Number of data (both training):	$K = 1\,000\,000$
Proportion of data for training (both training):	80% - $K_0 = 800\,000$
Number of terms in F^{ε}_{app} (MLP's):	$N_t = 5$
Number of Fourier coefficients in $G^{\varepsilon}_{app}(\cdot)$:	$N = 4$
Hidden layers per MLP (both training):	2
Neurons on each hidden layer (first training):	200
Neurons on each hidden layer (second training):	25
Epochs (both training):	200

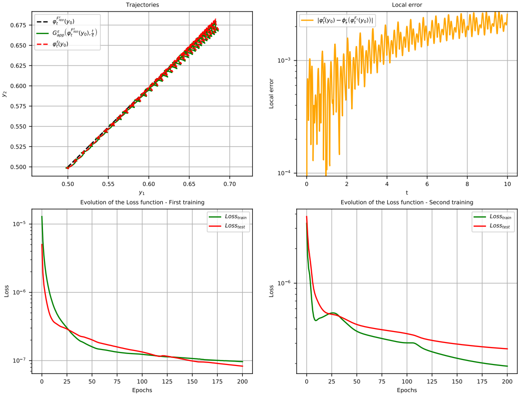

Figure: Integration (green: approximated solution, red: exact solution, orange: numerical error), Loss decays (green: $Loss_{Train}$, red: $Loss_{Test}$, Left: first training, Right: second training)

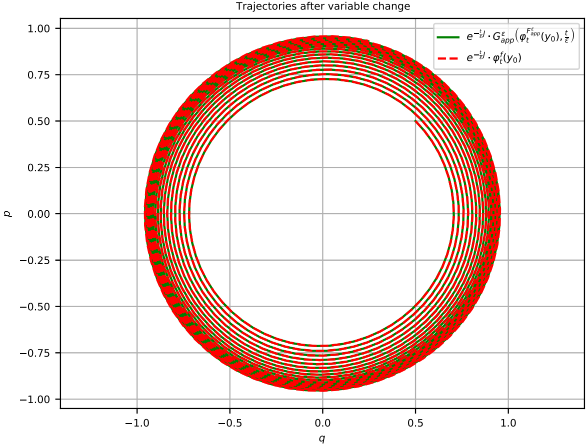

Figure: Integration after inverse variable change (green: approximated solution, red: exact solution)

- Get a consistent numerical method for highly oscillatory ODE's like in autonomous case. Idea: performing existing UA methods & adaptation of modified field theory to nonautonomous ODE's.
- Study geometric properties and energy conservation (e.g. hamiltonian highly oscillatory ODE's).

Thanks for your attention !