Modelling of highly oscillatory phenomenon by Neural Networks

Maxime BOUCHEREAU - Université de Rennes (IRMAR)

Rencontres Doctorales Lebesgue - April 2024

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

Goal: Approximate solutions of ODE of this form:

$$\dot{y}^\varepsilon(t) = \frac{1}{\varepsilon} A y^\varepsilon(t) + g(y^\varepsilon(t)) \tag{1}$$

ODE parameters:

- Initial condition: $y_0 := y^\varepsilon(0) \in \mathbb{R}^d$
- High oscillations: $Spec(A) \subset i\mathbb{Z}$ and $\varepsilon \ll 1$.
- Other phenomenon: $g : \mathbb{R}^d \longrightarrow \mathbb{R}^d$ smooth.

Main tools used:

- Function approximations by neural networks.
- Averaging theory.

Toolbox

$$\dot{y^\varepsilon}(t) = \frac{1}{\varepsilon} A y^\varepsilon(t) + g\left(y^\varepsilon(t)\right) \tag{2}$$

Theorem (P. Chartier, N. Crouseilles, M. Lemou & F. Méhats, 2016 - Solution of (2))

There exist A^ε and g^ε such that:

- A^ε *generates a* $2\pi-$*periodic flow:* $\tau \longmapsto \phi_\tau^\varepsilon = e^{\tau A^\varepsilon}$.
- g^ε *generates a flow* $t \longmapsto \varphi_t^{g^\varepsilon}$.
- $\phi_{\frac{t}{\varepsilon}}^\varepsilon \circ \varphi_t^{g^\varepsilon} = \varphi_t^{g^\varepsilon} \circ \phi_{\frac{t}{\varepsilon}}^\varepsilon$.
- *For all* $t \in [0,T], \varepsilon \in]0,1]$: $\left| y^\varepsilon(t) - \phi_{\frac{t}{\varepsilon}}^\varepsilon\left(\varphi_t^{g^\varepsilon}(y_0)\right)\right| \leqslant C_T e^{-\frac{C_T}{\varepsilon}}$.

Problem: Very hard to compute A^ε and g^ε (formal series and derivatives[1])

[1] Philippe Chartier, Nicolas Crouseilles, and Mohammed Lemou (2016). "Averaging of highly-oscillatory transport equations". In: *arXiv preprint arXiv:1609.09819*

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:*

$$\mathcal{N}(x) = W_L \cdot \Sigma\left(\cdots W_1 \cdot \Sigma\left(W_0 \cdot x + b_0\right) + b_1 \cdots\right) + b_L \tag{3}$$

where:

- ***Number of layers:*** $L + 1$
- ***Weights:*** $W_0 \in \mathcal{M}_{d_1,d_0}(\mathbb{R}), W_1 \in \mathcal{M}_{d_2,d_1}(\mathbb{R}), \ldots, W_L \in \mathcal{M}_{d_L,d_{L-1}}(\mathbb{R})$.
- ***Neurons:*** *Lines of W_i's.*
- ***Activation function:*** $\Sigma(y_1, \ldots, y_d) = (\sigma(y_1), \ldots, \sigma(y_d))$ *is a componant-wise nonlinear mapping σ.*

W_i's and b_i's are adjustable.

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.

$$||f - \mathcal{N}||_{L^\infty(\Omega)} \leqslant \varepsilon \tag{4}$$

One can approximate every continuous function over a compact set by a neural network, large enough.[2]

[2] George Cybenko (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Application to Autonomous highly oscillatory ODE's

Main goal: Model φ^{g^ε} and ϕ^ε by neural networks:

$$\varphi_\theta(y,h,\varepsilon) \approx \varphi_h^{g^\varepsilon}(y) \text{ and } \phi_\theta(\tau,y,\varepsilon) \approx \phi_\tau^\varepsilon(y).$$

- **Construction of the dataset:** Computation with Python RK45 (approximation of exact flow of (2)) K data:

$$y_1^{(k)} = y^{\varepsilon^{(k)}}(h^{(k)}), \tag{5}$$

  where $y^{\varepsilon^{(k)}}(0) = y_0^{(k)}$, $h^{(k)}$ and $\varepsilon^{(k)}$ are randomly selected.
- **Structure of neural networks:**

$$\varphi_\theta(y,h,\varepsilon) = y + h \underbrace{R_{\theta,\varphi}(y,h,\varepsilon)}_{MLP\ w.r.t.\ (y,h,\varepsilon)} \tag{6}$$

$$\phi_\theta(\tau,y,\varepsilon) = y + \varepsilon\Big[\underbrace{R_{\theta,\phi}(\cos(\tau),\sin(\tau),y,\varepsilon)}_{MLP\ w.r.t.\ (\cos(\tau),\sin(\tau),y,\varepsilon)} - \underbrace{R_{\theta,\phi}(1,0,y,\varepsilon)}_{MLP\ w.r.t.\ (1,0,y,\varepsilon)}\Big] \tag{7}$$

Properties of φ^{g^ε} and ϕ^ε are preserved with neural networks.

- **Training of the Neural Networks:** We minimize the MSE $Loss_{Train}$ over K_0 ($\leqslant K$ data) with gradient descent w.r.t. weights of neural networks:

$$\begin{aligned} Loss_{Train} &= \frac{1}{K_0}\sum_{k=0}^{K_0-1}\left|y_1^{(k)} - \phi_\theta\left(\frac{h^{(k)}}{\varepsilon^{(k)}}, \varphi_\theta(y_0^{(k)}, h^{(k)}, \varepsilon^{(k)}), \varepsilon^{(k)}\right)\right|^2 \\ &+ \frac{1}{K_0}\sum_{k=0}^{K_0-1}\left|\varphi_\theta\left(\phi_\theta\left(\frac{h^{(k)}}{\varepsilon^{(k)}}, y_0^{(k)}, \varepsilon^{(k)}\right), h^{(k)}, \varepsilon^{(k)}\right)\right. \\ &\left. -\phi_\theta\left(\frac{h^{(k)}}{\varepsilon^{(k)}}, \varphi_\theta(y_0^{(k)}, h^{(k)}, \varepsilon^{(k)}), \varepsilon^{(k)}\right)\right|^2 \end{aligned} \quad (8)$$

Learned properties: Structure of the flow & Commutativity of both flows.

- **Validation of the training of the Neural Networks:** At each step, we test our trained Neural Networks over the $K - K_0$ remaining data by observing the MSE $Loss_{Test}$:

$$\begin{aligned} Loss_{Test} &= \frac{1}{K-K_0}\sum_{k=K_0}^{K-1}\left|y_1^{(k)} - \phi_\theta\left(\frac{h^{(k)}}{\varepsilon^{(k)}}, \varphi_\theta(y_0^{(k)}, h^{(k)}, \varepsilon^{(k)}), \varepsilon^{(k)}\right)\right|^2 \\ &+ \frac{1}{K-K_0}\sum_{k=K_0}^{K-1}\left|\varphi_\theta\left(\phi_\theta\left(\frac{h^{(k)}}{\varepsilon^{(k)}}, y_0^{(k)}, \varepsilon^{(k)}\right), h^{(k)}, \varepsilon^{(k)}\right)\right. \\ &\left. -\phi_\theta\left(\frac{h^{(k)}}{\varepsilon^{(k)}}, \varphi_\theta(y_0^{(k)}, h^{(k)}, \varepsilon^{(k)}), \varepsilon^{(k)}\right)\right|^2 \end{aligned} \tag{9}$$

Good training: $Loss_{Train}$ and $Loss_{Test}$ have a similar decay as optimization steps go by. Same principle than linear regression.

- **Numerical integration:** We plot the points:

$$y^{\varepsilon}_{\theta,n} = \phi_\theta\left(\frac{nh}{\varepsilon}, \varphi_\theta(\cdot,h,\varepsilon)^n(y^{\varepsilon}(0))\right), \tag{10}$$

for all $n \in [\![0, N]\!]$ and $h = \frac{T}{N}$

Theorem (M. B., P. Chartier, M. Lemou & F. Méhats, 2024 - Approximated solution of (2))

Let denote the following learning errors:

$$\delta_\phi := ||\phi^{\varepsilon} - \phi_\theta||_{L^\infty} \ \ \textit{and}\ \delta_\varphi := \left|\left|\frac{\varphi^{g^{\varepsilon}} - \varphi_\theta}{h}\right|\right|_{L^\infty}. \tag{11}$$

Then there exists positive constant $\Lambda_{\theta,T}$ such that:

$$\underset{0\leqslant n\leqslant N}{Max}\left|y^{\varepsilon}_{\theta,n} - y^{\varepsilon}(t_n)\right| \leqslant \Lambda_{\theta,T}(\delta_\phi + \delta_\varphi) + C_T e^{-\frac{C_T}{\varepsilon}}. \tag{12}$$

- **System of equations:**

$$\begin{cases} \dot{q^\varepsilon} & = & \frac{1}{\varepsilon} p^\varepsilon \\ \dot{p^\varepsilon} & = & -\frac{1}{\varepsilon} q^\varepsilon + \left(\frac{1}{4} - {q^\varepsilon}^2 \right) p \end{cases} \tag{13}$$

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \text{ and } g(q,p) = \begin{bmatrix} 0 \\ \left(\frac{1}{4} - q^2 \right) p \end{bmatrix} \tag{14}$$

- **Parameters:**

Parameters	
# Math Parameters:	
Interval where time steps are selected:	$[h_-, h_+] = [0.001, 0.1]$
Interval where small parameters are selected:	$[\varepsilon_-, \varepsilon_+] = [0.001, 0.2]$
Time for ODE simulation:	$T = 1$
Initial datum:	$y^\varepsilon(0) = (0.5, 0.5)$
# Machine Learning Parameters:	
Domain where initial data are selected:	$\Omega = [-2, 2]^2$
Number of data:	$K = 100\,000$
Proportion of data for training:	80% - $K_0 = 80\,000$
Hidden layers per MLP:	2
Neurons on each hidden layer:	200
Learning rate:	$2 \cdot 10^{-3}$
Epochs:	200

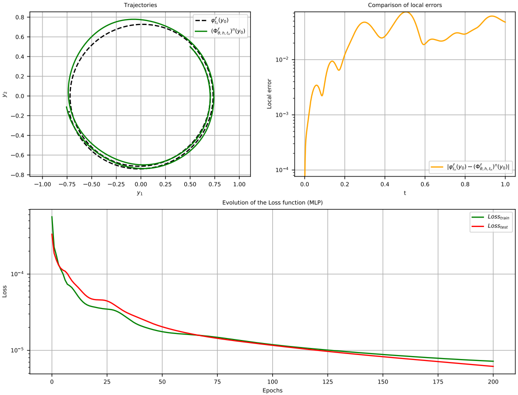

Figure: Comparison between *Loss* decays (green: $Loss_{Train}$, red: $Loss_{Test}$), trajectories (dashed dark: exact flow, green: numerical flow with learned vector fields and local error (yellow) for the Van der Pol oscillator in the case $\varepsilon = 0.1$.

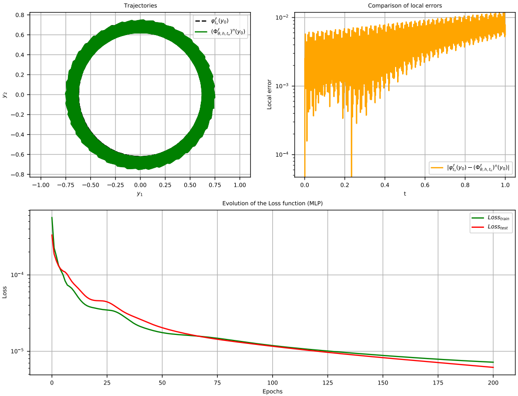

Figure: Comparison between *Loss* decays (green: $Loss_{Train}$, red: $Loss_{Test}$), trajectories (dashed dark: exact flow, green: numerical flow with learned vector fields and local error (yellow) for the Van der Pol oscillator in the case $\varepsilon = 0.001$.

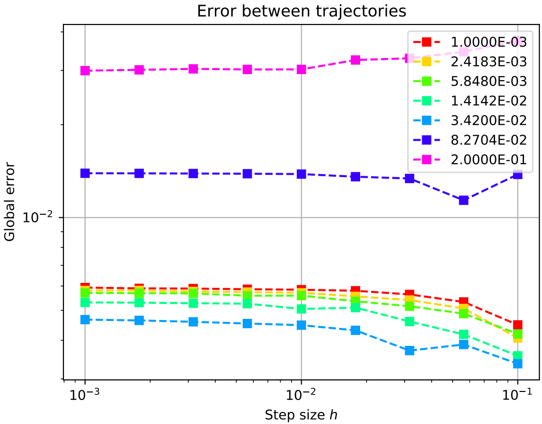

Figure: Integration errors (each color corresponds to a high oscillation parameter ε) of Van der Pol oscillator.

Conclusion

- **Main result:** Good approximation of $t \longmapsto y^{\varepsilon}(t)$ with reduced computationnal time (faster than Python RK45) for $\varepsilon \ll 1$.
- **Outlook:** Geometric properties (e.g. Hamiltonian, divergence-free).

Chartier, Philippe and Crouseilles, Nicolas and Lemou, Mohammed, *Averaging of highly-oscillatory transport equations*, arXiv preprint arXiv:1609.09819, 2016

CYBENKO, George, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems, 1989

Thanks for your attention !