

Modelling of Highly oscillatory phenomenon by neural networks

Maxime BOUCHEREAU - Université de Rennes (IRMAR)

CJCMA - 2023

Ph. D supervisors:

Francois CASTELLA - Philippe CHARTIER

Mohammed LEMOU - Florian MEHATS



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

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

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

1 Introduction

- Function approximations by neural networks and structure preservation
- Modified field theory for autonomous ODE's
- Highly oscillatory ODE's: theory & UA methods

2 Autonomous ODE's

- General framework
- Machine Learning method
- Convergence result
- Numerical tests - Rigid Body system - Forward Euler

3 Highly oscillatory ODE's

- General Framework
- Machine Learning method
- Convergence result
- Numerical test - Logistic equation - Forward Euler

4 Outlook

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:

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

where:

- $L + 1$ is the number of **layers**. **Shallow network**: $L = 1$, **Deep network**: $L \geq 2$. Layers 1 to $L - 1$ are named **hidden layers**.
- $b_0 \in \mathbb{R}^{d_0}, b_1 \in \mathbb{R}^{d_1}, \dots, b_L \in \mathbb{R}^{d_L}$ are the **bias**.
- $W_0 \in \mathcal{M}_{d_1, d_0}(\mathbb{R}), W_1 \in \mathcal{M}_{d_2, d_1}(\mathbb{R}), \dots, W_L \in \mathcal{M}_{d_L, d_{L-1}}(\mathbb{R})$ are the **weights**. Lines of W_i 's are **neurons**.
- $\Sigma(y_1, \dots, y_d) = (\sigma(y_1), \dots, \sigma(y_d))$ is a component-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 $\mathcal{N} : \mathbb{R}^d \rightarrow \mathbb{R}^k$ a MLP s.t.

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

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 $x \in \mathbb{R}^{2d}$

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

where $\mathcal{H} : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ 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:*

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

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

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

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

Proposition (Property of the modified field)

- **Perturbation w.r.t. h :** If Φ is of order p , then
$$\tilde{f}_h(y) = f(y) + h^p R(y, h)$$

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 $t \in \mathbb{R}$:

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

where:

- F^ε is called **averaged field**. Structure:

$F^\varepsilon(y) = \langle f \rangle(y) + \varepsilon F_1(y) + \varepsilon^2 F_2(y) + \dots$, where $\langle f \rangle$ is the average field w.r.t. time variable:

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

- $\phi_\tau^\varepsilon(y) = y + \varepsilon \cdot G^\varepsilon(\tau, y)$ (Near to identity mapping) and is 2π -periodic w.r.t. τ .

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.*

$$\max_{0 \leq n \leq N} |(\Phi_h)^n(y_0) - y^\varepsilon(nh)| \leq Ch^r \quad (9)$$

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.

Example: Forward Euler - Micro-Macro method

We solve with Forward Euler the system:

$$\begin{cases} \dot{v}(t) &= F^{[1]}(v(t)) \\ \dot{w}(t) &= f\left(\frac{t}{\varepsilon}, \Phi_{\frac{t}{\varepsilon}}^{[1]}(v(t)) + w(t)\right) - \left(\frac{1}{\varepsilon} \partial_{\tau} \Phi_{\frac{t}{\varepsilon}}^{[1]} + \partial_y \Phi_{\frac{t}{\varepsilon}}^{[1]} F^{[1]}\right)(v(t)) \end{cases}$$

where $F^{[1]} = F^{\varepsilon} + \mathcal{O}(\varepsilon^2)$ and $\Phi^{[1]} = \Phi^{\varepsilon} + \mathcal{O}(\varepsilon^2)$ are computable with explicit formulas. This is a UA-method of order 1. We get:

$$y^{\varepsilon}(t) = \Phi^{[1]}(v(t)) + w(t) \quad (10)$$

Autonomous ODE's

● Autonomous ODE:

$$\begin{cases} \dot{y}(t) &= f(y(t)) \\ y(0) &= y_0 \end{cases} \quad (11)$$

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

- **Structure of f_θ :**

$$f_\theta(y, h) = f(y) + h^p R_\theta(y, h)$$

- **Data creation:** Computation of exact solutions $y_1^{(k)} = \varphi_{h^{(k)}}^f(y_0^{(k)})$ with accurate and expensive integrator, where, for all $0 \leq k \leq K-1$, $y_0^{(k)} \in \Omega \subset \mathbb{R}^d$, $h^{(k)} \in [h_-, h_+]$ are randomly selected.
- **Training of the neural network:** Optimization of **MSE Loss**:

$$Loss_{Train} = \frac{1}{K_0} \sum_{k=0}^{K_0-1} \frac{1}{h^{(k)2p+2}} \left| \underbrace{\Phi_{h^{(k)}}^{f_\theta(\cdot, h^{(k)})}(y_0^{(k)})}_{=\hat{y}_1^{(k)}} - y_1^{(k)} \right|^2$$

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

$$Loss_{Test} = \frac{1}{K - K_0} \sum_{k=K_0}^{K-1} \frac{1}{h^{(k)2p+2}} \left| \hat{y}_1^{(k)} - y_1^{(k)} \right|^2$$

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

$$e_n^* = \left(\Phi_h^{f_\theta(\cdot, h)} \right)^n (y_0) - \varphi_{nh}^f(y_0) \quad (12)$$

Denoting the **learning error** by

$$\delta := \operatorname{Max}_{(y, h) \in \Omega \times [h_-, h_+]} \frac{\left| \tilde{f}_h(y, h) - f_\theta(y, h) \right|}{h^p} \quad (13)$$

Theorem (M.B., P.Chartier, M.Lemou, F.Méhats - 2023¹)

Assuming that

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

$$\forall 0 \leq h \leq h_+, \quad \left\| \Phi_h^{f_1} - \Phi_h^{f_2} \right\|_{L^\infty(\Omega)} \leq Ch \|f_1 - f_2\|_{L^\infty(\Omega)} \quad (14)$$

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 $\forall 0 \leq h \leq h_+, \forall (y_1, y_2) \in \Omega^2$:

$$\left| \Phi_h^f(y_1) - \Phi_h^f(y_2) \right| \leq (1 + Lh) |y_1 - y_2|. \quad (15)$$

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

$$\max_{0 \leq n \leq N} |e_n^*| \leq \frac{C\delta h^p}{\tilde{L}} \left(e^{\tilde{L}T} - 1 \right) \quad (16)$$

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

$$\begin{cases} \dot{y}_1 &= -\frac{1}{6}y_2y_3 \\ \dot{y}_2 &= \frac{2}{3}y_1y_3 \\ \dot{y}_3 &= -\frac{1}{2}y_1y_2 \end{cases},$$

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:	$\Omega = \{x \in [-2, 2]^2 : 0.98 \leq x \leq 1.02\}$
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

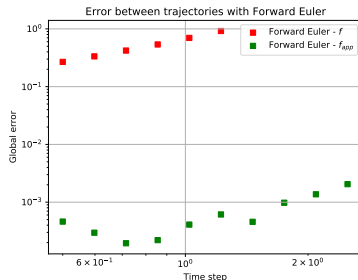
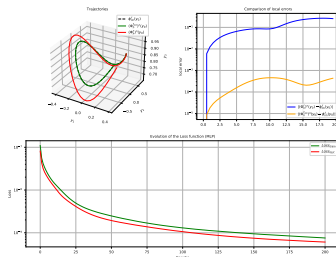


Figure: Left: Comparison between $Loss$ decays (green: $Loss_{Train}$, red: $Loss_{Test}$), trajectories (dashed dark: exact flow, red: numerical flow with f , green: numerical flow with $f_\theta(\cdot, h)$) and local error (blue: exact flow and numerical flow with f , yellow: exact and numerical flow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

Highly oscillatory ODE's

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

$$\begin{cases} \dot{y}^\varepsilon(t) &= f\left(\frac{t}{\varepsilon}, y^\varepsilon(t)\right) \\ y^\varepsilon(0) &= y_0 \end{cases} \quad (17)$$

- **General strategy:** Use the Micro-Macro UA-method

$$\begin{cases} \dot{v}(t) &= F^{[1]}(v(t)) \\ \dot{w}(t) &= f\left(\frac{t}{\varepsilon}, \Phi_{\frac{t}{\varepsilon}}^{[1]}(v(t)) + w(t)\right) - \left(\frac{1}{\varepsilon} \partial_\tau \Phi_{\frac{t}{\varepsilon}}^{[1]} + \partial_y \Phi_{\frac{t}{\varepsilon}}^{[1]} F^{[1]}\right)(v(t)) \end{cases}$$

$$y^\varepsilon(t) = \Phi^{[1]}(v(t)) + w(t)$$

and transform this system into an autonomous system in order to use the strategy used for autonomous ODE's.

- **Transformation to autonomous system:**

Let denote $W(t) = (v(t), w(t))$. We denote the Micro-Macro system:

$$\dot{W}(t) = G^\varepsilon \left(\frac{t}{\varepsilon}, W(t) \right) \quad (18)$$

If we denote $X(t) = \left(\frac{t}{\varepsilon}, W(t) \right)$, we have this system:

$$\dot{X}(t) = \mathcal{G}^\varepsilon(X(t)) \quad (19)$$

where $\mathcal{G}^\varepsilon(\tau, W) = \left(\frac{1}{\varepsilon}, G^\varepsilon(\tau, W) \right)$. This is an autonomous system.

- **"Modified field" (Forward Euler):** We get:

$$W(t_0 + h) = W(t_0) + h \cdot \tilde{G}_h^\varepsilon \left(\frac{t_0}{\varepsilon}, W(t_0) \right) \quad (20)$$

- **Data creation:** Computation of exact solutions

$W_1^{(k)} = \varphi_{t_0^{(k)}, h^{(k)}}^{G^{\varepsilon^{(k)}}}(W_0^{(k)})$ where $t_0^{(k)} \in [0, 2\pi]$ with accurate and expensive integrator, where $W_0^{(k)}$ is randomly selected in the compact set $\Omega \subset \mathbb{R}^{2d}$, $h^{(k)}$ and $\varepsilon^{(k)}$ are randomly selected in $[h_-, h_+]$ and $[\varepsilon_-, \varepsilon_+]$, for all $0 \leq k \leq K-1$.

- **Neural network structure:**

$$G_\theta(\tau, W, \varepsilon, h) = G^\varepsilon(\tau, W) + hR_\theta(\cos(\tau), \sin(\tau), W, \varepsilon, h) \quad (21)$$

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

$$Loss_{Train} = \frac{1}{K_0} \sum_{k=0}^{K_0-1} \frac{1}{h^{(k)^4}} \left| \hat{W}_1^{(k)} - W_1^{(k)} \right|^2$$

where $\hat{W}_1^{(k)} = W_0^{(k)} + hG_\theta\left(\frac{t_0^{(k)}}{\varepsilon^{(k)}}, W_0^{(k)}, \varepsilon^{(k)}, h^{(k)}\right)$

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

$$Loss_{Test} = \frac{1}{K - K_0} \sum_{k=K_0}^{K-1} \frac{1}{h^{(k)^4}} \left| \hat{W}_1^{(k)} - W_1^{(k)} \right|^2$$

We get an approximated solution of the ODE:

Denoting the **learning error** by

$$\delta := \max_{(\tau, W, \varepsilon, h) \in [0, 2\pi] \times \Omega \times [\varepsilon_-, \varepsilon_+] \times [h_-, h_+]} \frac{|G_\theta(\tau, W, \varepsilon, h) - \tilde{G}_h^\varepsilon(\tau, W)|}{h}$$

If we denote $(W_n^*)_{0 \leq n \leq N}$ the sequence given by $W_0^* = W(0)$ and, for all $n \in \mathbb{N}$:

$$W_{n+1}^* = W_n^* + h \cdot G_\theta\left(\frac{nh}{\varepsilon}, W_n^*, \varepsilon, h\right) \quad (22)$$

we get the following proposition:

Proposition

For $T > 0$, there exist constants $C, L > 0$ s.t. for all $n \in \llbracket 0, N \rrbracket$:

$$\max_{0 \leq n \leq N} |W_n^* - W(nh)| \leq \frac{e^{LT} - 1}{L} C \delta h \quad (23)$$

$$\dot{y}^\varepsilon(t) = y^\varepsilon(t) (1 - y^\varepsilon(t)) + \sin\left(\frac{t}{\varepsilon}\right) \quad (24)$$

Parameters	
# Math Parameters:	
Interval where time steps h are selected:	$[h_-, h_+] = [0.001, 0.1]$
Interval where small parameters ε are selected:	$[\varepsilon_-, \varepsilon_+] = [0.01, 1]$
Time for ODE simulation:	$T = 1$
Time step for ODE simulation:	$h = 0.1$
Small parameter for ODE simulation:	$\varepsilon = 0.1$
# AI Parameters:	
Domain where data are selected:	$\Omega = [0, 2] \times [-2, 2]$
Number of data:	$K = 10\,000$
Proportion of data for training:	80% - $K_0 = 8\,000$
Hidden layers per MLP:	2
Neurons on each hidden layer:	200
Epochs:	2000

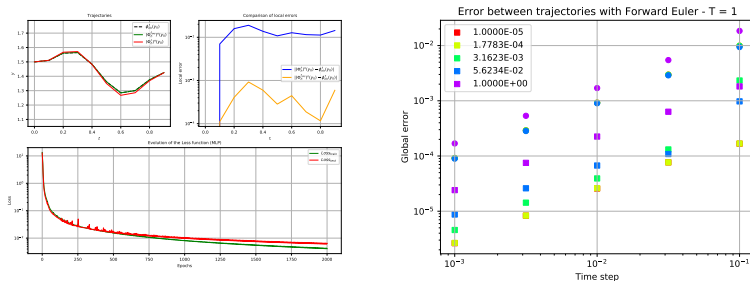


Figure: Left: Comparison between $Loss$ decays (green: $Loss_{Train}$, red: $Loss_{Test}$), trajectories (dashed dark: exact flow, red: numerical flow with f , green: numerical flow with $G_\theta(\cdot, h)$) and local error (blue: exact flow and numerical flow with f , yellow: exact and numerical flow with $G_\theta(\cdot, h)$). Right: Integration errors (circles: integration with G , squares: integration with $G_\theta(\cdot, h)$). Integration with Machine Learning is very powerful for small values of ϵ .

- Improve results for highly oscillatory ODE's with Micro-Macro scheme.
- Use other Neural Network structure.

Thanks for your attention !