Artificial intelligence's methods for autonomous ODE's

Presentation - April 15th 2022

CENTRE
HENRI LEBESGUE
CENTRE DE MATHÉMATIQUES

UNIVERSITÉ DE RENNES 1

- Theory
  - Recall: Euler's method
  - Problem: modified equation
  - Neural Network: Multi-Layer Perceptron
- Fixed step time
  - Problem
  - Numerical simulations
- Step time into data
  - New formulation of the problem
  - Numerical simulations
- Outlook

- **ODE (autonomous):** $\dot{y} = f(y)$, $f \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R}^d)$, $y(0) \in \mathbb{R}^d$
- **Euler's method:** Approximation of $y(nh) = \varphi^f_{nh}(y_0)$ on $[0, T]$ by $(y_n)_{0 \leqslant n \leqslant N}$ defined by $y_0 = y(0)$:

$$y_{n+1} = y_n + hf(y_n)$$

$h = \frac{T}{N}$: Time step

- **Result of convergence:**

$$\underset{0 \leqslant n \leqslant N}{Max} |y_n - y(nh)| \leqslant Ch$$

- **General comment:** Cheap for computations, but not accurate method.

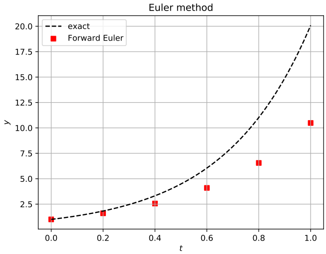


Figure: Example with $\dot{y} = 3y$

- **Modified equation:** $\dot{y} = \widetilde{f_h}(y)$, $\widetilde{f_h} \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R}^d)$: modified vector field s.t. if $(z_n)_{0 \leqslant n \leqslant N}$ is defined by $z_0 = y(0)$ and:

$$z_{n+1} = z_n + h\widetilde{f_h}(z_n)$$

  we have $z_n = y(nh)$
- **Advantages:** Easy to program and gives the exact solution.

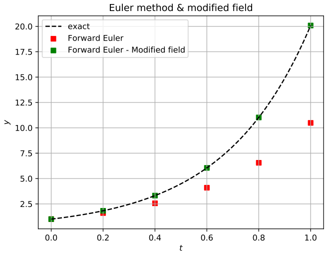


Figure: Example with $\dot{y} = 3y$

- **Theory:** Structure of $\widetilde{f_h}$: $\widetilde{f_h}(y) = f(y) + hR(y, h)$
- **Goal:** Approximate $\widetilde{f_h}$ by a neural network $f_{app,h}$

- **Artificial neuron:** Mapping $x \mapsto \sigma\left(w_1 x_1 + \cdots + w_k x_k + b\right)$ $w, x \in \mathbb{R}^k, b \in \mathbb{R}$, σ : Transfer function (tanh for example)
- **MLP:** mapping:

$$F : \quad \mathbb{R}^k \quad \longmapsto \quad \mathbb{R}^d$$

$$F(x) = \underbrace{W_L}_{\in \mathcal{M}_{\zeta,k}(\mathbb{R})} \Sigma \left(\underbrace{W_{L-1}}_{\in \mathcal{M}_{\zeta}(\mathbb{R})} \Sigma \left(\cdots \Sigma \left(\underbrace{W_0}_{\in \mathcal{M}_{\zeta,k}(\mathbb{R})} x + \underbrace{b_0}_{\in \mathbb{R}^k} \right) \cdots \right) + \underbrace{b_{L-1}}_{\in \mathbb{R}^{\zeta}} \right) + \underbrace{b_L}_{\in \mathbb{R}^d}$$

ζ : Number of neurons on each layer, $W_0, \cdots, W_L$: Weights of the MLP, $b_0, \cdots, b_L$: bias, $\Sigma(x) = (\sigma(x_1), \cdots, \sigma(x_k))$

- **Universal approximation:** Let $g \in \mathcal{C}^0(\mathbb{R}^k, \mathbb{R}^d)$, $\Omega \subseteq \mathbb{R}^d$ compact, $\varepsilon > 0$ If weights and bias are correctly choosen and ζ is large enough:

$$||F - g||_{L^\infty(\Omega)} \leqslant \varepsilon$$

- **Structure of $f_{app,h}$:** $f_{app,h}(y) = f(y) + h \cdot R_{app}(y, h)$ (learning of the perturbation)

Fixed step time

Problem & strategy

Data selection: $y_0^{(k)} \in [-R, R]^d$ selected and h fixed

Computation: $y_1^{(k)} = \varphi_h^f\left(y_0^{(k)}\right) = y_0^{(k)} + h\widetilde{f_h}\left(y_0^{(k)}\right)$

Prediction: $\hat{y_1}^{(k)} = y_0^{(k)} + hf_{app,h}\left(y_0^{(k)}\right)$

Training: Optimization of the MSE Loss

$$Loss_{Train} = \frac{1}{K_0 h^4} \sum_{k=0}^{K_0-1} \left| \hat{y_1}^{(k)} - y_1^{(k)} \right|^2$$

Test: Good decay of Loss-Test

$$Loss_{test} = \frac{1}{(K-K_0)h^4} \sum_{k=K_0}^{K-1} \left| \hat{y_1}^{(k)} - y_1^{(k)} \right|^2$$

Integration: Numerical integration with the $f_{app,h} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ which minimizes $Loss_{Train}$

- **Advantages:** Efficient training, even with 10000 data
- **Disadvantages:** New training has to be done if we want to change h

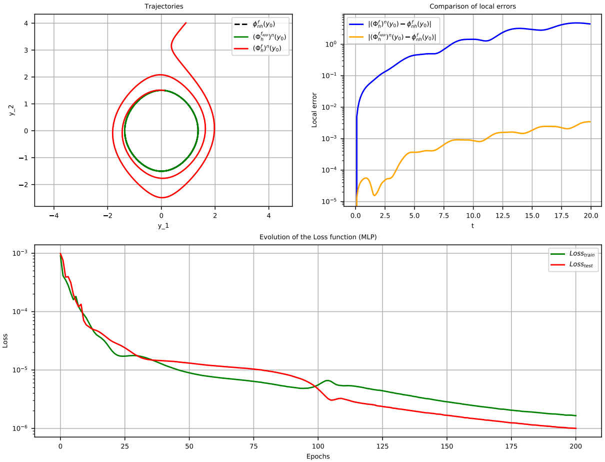
- **Dynamical system:** Pendulum

$$\begin{cases} \dot{p} & = & -\sin(q) \\ \dot{q} & = & p \end{cases}$$

- **Parameters:**
  - Time step: $h = 0.1$
  - Duration of integration: $T = 20$
  - Data: $K = 25000$
  - Data for training: $K_0 = 20000$ (80%)
  - Amplitude for data selection: $R = 2$
  - Hidden layers: $L = 2$
  - Neurons per hidden layer: $\zeta = 200$
  - Epochs for training: 200

Fixed step time

Numerical simulations



Data selection: $y_0^{(k)} \in [-R, R]^d$ & $h^{(k)} \in [h_-, h_+]$ selected

Computation: $y_1^{(k)} = \varphi^f_{h^{(k)}}\left(y_0^{(k)}\right) = y_0^{(k)} + h^{(k)} \tilde{f}_{h^{(k)}}\left(y_0^{(k)}\right)$

Prediction:
$\hat{y_1}^{(k)} = y_0^{(k)} + h^{(k)} f_{app,h^{(k)}}\left(y_0^{(k)}\right)$

Training: Optimization of the MSE Loss

$$Loss_{Train} = \frac{1}{K_0} \sum_{k=0}^{K_0-1} \frac{1}{{h^{(k)}}^4} \left| \hat{y_1}^{(k)} - y_1^{(k)} \right|^2$$

Test: Good decay of Loss-Test

$$Loss_{test} = \frac{1}{(K-K_0)} \sum_{k=K_0}^{K-1} \frac{1}{{h^{(k)}}^4} \left| \hat{y_1}^{(k)} - y_1^{(k)} \right|^2$$

Integration: Numerical integration with the $f_{app,h} : \mathbb{R}^{d+1} \to \mathbb{R}^d$ which minimizes $Loss_{Train}$

- **Advantages:** Only one training is necessary for many step times, allows to study error of the method.
- **Disadvantages:** Many data are necessary to ensure a good training

- **Property:** We have:

$$\underset{0 \leqslant n \leqslant N}{Max} |y_n^* - y(nh)| \leqslant \delta h$$

where $(y_n^*)_{0 \leqslant n \leqslant N}$ is the numerical solution computed with Forward Euler for $f_{app,h}$ and δ depends on the error between $\widetilde{f_h}$ and $f_{app,h}$

- **Dynamical system:** Pendulum.

$$\begin{cases} \dot{p} & = & -\sin(q) \\ \dot{q} & = & p \end{cases}$$

- **Parameters:**
  - Time step (interval): $[0.01, 0.5]$
  - Duration of integration: $T = 20$
  - Data: $K = 1000000$
  - Data for training: $K_0 = 800000$ (80%)
  - Amplitude for data selection: $R = 2$
  - Hidden layers: $L = 1$
  - Neurons per hidden layer: $\zeta = 200$
  - Epochs for training: 200

Step time into data

Numerical simulations

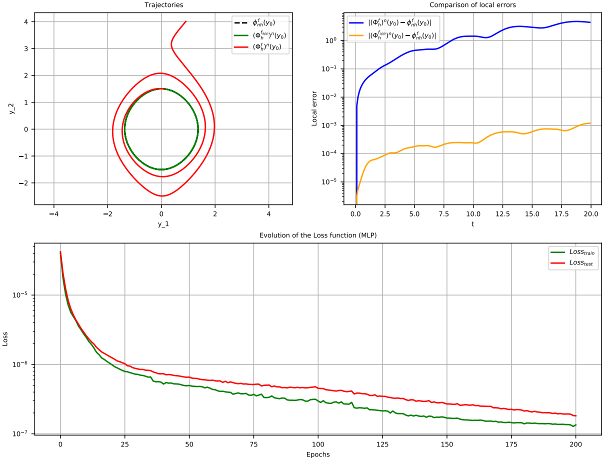


Figure: Simulation for $h = 0.1$

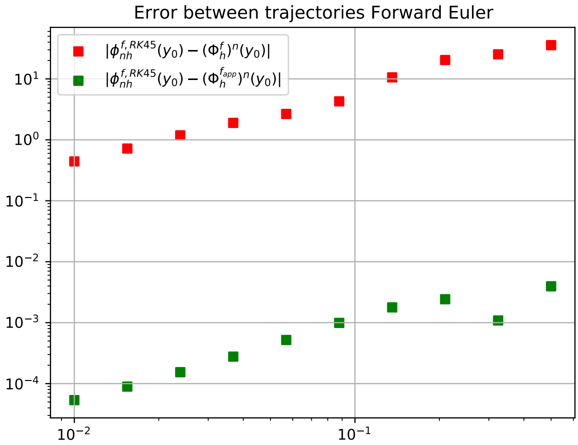


Figure: Global errors for various time steps - Forward Euler without training (red) & Forward Euler with training (green)

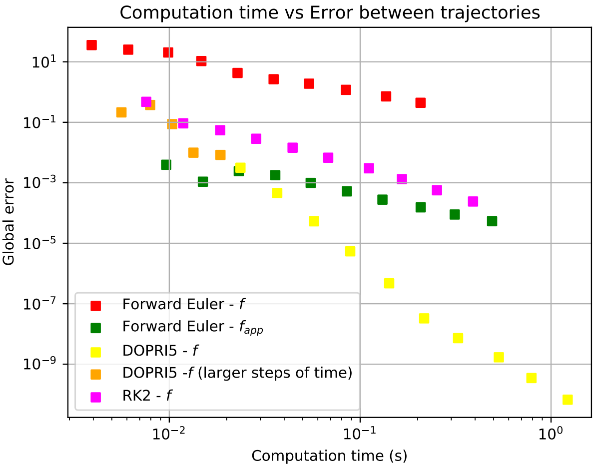


Figure: Global errors vs Computation time for various time steps and numerical methods

Outlook

- **Change the numerical scheme:** Example: Runge-Kutta 2
- **Non autonomous systems:** Example of highly oscillatory equations:

$$\dot{y} = f\left(\frac{t}{\varepsilon}, y\right)$$

where $\varepsilon \to 0$ and $\tau \mapsto f(\tau, \cdot)$ is periodic

Thanks for your attention !