Neural network aided simulation of ordinary differential equations

Davide Murari

`davide.murari@ntnu.no`

In collaboration with Marta Betcke, and Lisa Maria Kreusser

Solving initial value problems with neural networks

► We aim to solve the autonomous initial value problem (IVP)

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)) \in \mathbb{R}^d, \\ \mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^d \end{cases}$$

on the time interval $[0, T]$.

Solving initial value problems with neural networks

▶ We aim to solve the autonomous initial value problem (IVP)

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)) \in \mathbb{R}^d, \\ \mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^d \end{cases}$$

on the time interval $[0, T]$.

▶ A one-step numerical method of order p is a map $\varphi^{\Delta t} : \mathbb{R}^d \to \mathbb{R}^d$ such that $\varphi^{\Delta t} = \phi^{\Delta t} + \mathcal{O}(\Delta t^{p+1})$, with $\phi^{\Delta t}$ the exact flow map of $\mathcal{F}$.

Solving initial value problems with neural networks

▶ We aim to solve the autonomous initial value problem (IVP)

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)) \in \mathbb{R}^d, \\ \mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^d \end{cases}$$

on the time interval $[0, T]$.

▶ A one-step numerical method of order p is a map $\varphi^{\Delta t} : \mathbb{R}^d \to \mathbb{R}^d$ such that $\varphi^{\Delta t} = \phi^{\Delta t} + \mathcal{O}(\Delta t^{p+1})$, with $\phi^{\Delta t}$ the exact flow map of $\mathcal{F}$.

▶ Using **neural networks** to replace or modify the map $\varphi^{\Delta t}$ can be useful when
  ▶ the dimension d is large
  ▶ one desires to have a (piecewise) continuous approximate solution
  ▶ one wants to also fit some observed data while approximately solving the IVP

Physics-informed neural networks

▶ We introduce a parametric map $\mathcal{N}_\theta(\cdot; \mathbf{x}_0) : [0, T] \to \mathbb{R}^d$ such that $\mathcal{N}_\theta(0; \mathbf{x}_0) = \mathbf{x}_0$, and choose its weights so that

$$\mathcal{L}(\theta) := \frac{1}{C} \sum_{c=1}^{C} \left\| \mathcal{N}'_\theta(t_c; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(t_c; \mathbf{x}_0)) \right\|_2^2 \to \min$$

for some collocation points $t_1, \ldots, t_C \in [0, T]$.

Physics-informed neural networks

▶ We introduce a parametric map $\mathcal{N}_\theta\left(\cdot\,;\mathbf{x}_0\right):[0,T]\to\mathbb{R}^d$ such that $\mathcal{N}_\theta\left(0;\mathbf{x}_0\right)=\mathbf{x}_0$, and choose its weights so that

$$\mathcal{L}(\theta):=\frac{1}{C}\sum_{c=1}^{C}\left\|\mathcal{N}_\theta'\left(t_c;\mathbf{x}_0\right)-\mathcal{F}\left(\mathcal{N}_\theta\left(t_c;\mathbf{x}_0\right)\right)\right\|_2^2\to\min$$

for some collocation points $t_1,\ldots,t_C\in[0,T]$.

▶ Then, $t\mapsto\mathcal{N}_\theta\left(t;\mathbf{x}_0\right)$ will solve a different IVP

$$\begin{cases}\mathbf{y}'\left(t\right)=\mathcal{F}\left(\mathbf{y}\left(t\right)\right)+\left(\mathcal{N}_\theta'\left(t;\mathbf{x}_0\right)-\mathcal{F}\left(\mathbf{y}\left(t\right)\right)\right)\in\mathbb{R}^d,\\\mathbf{y}\left(0\right)=\mathbf{x}_0\in\mathbb{R}^d,\end{cases}$$

where hopefully the residual $\mathcal{N}_\theta'\left(t;\mathbf{x}_0\right)-\mathcal{F}\left(\mathbf{y}\left(t\right)\right)$ is small in some sense.

A-posteriori error estimate

Theorem: Quadrature-based a-posteriori error estimate

Let $\mathbf{x}(t)$ be the solution of the IVP

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)) \in \mathbb{R}^d, \ \mathcal{F} \in \mathcal{C}^{p+1}(\mathbb{R}^d, \mathbb{R}^d), \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases}$$

Suppose that $\mathcal{N}_\theta(\cdot; \mathbf{x}_0) : [0, \Delta t] \to \mathbb{R}^d$ is smooth and satisfies

$$\left\| \mathcal{N}_\theta'(t_c; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(t_c; \mathbf{x}_0)) \right\|_2 \leq \varepsilon, \ c = 1, \ldots, C$$

for C collocation points $0 \leq t_1 < \cdots < t_C \leq \Delta t$ defining a quadrature rule of order p. Then, there exist $\alpha, \beta > 0$ such that

$$\left\| \mathbf{x}(t) - \mathcal{N}_\theta(t; \mathbf{x}_0) \right\|_2 \leq \alpha(\Delta t)^{p+1} + \beta\varepsilon, \ t \in [0, \Delta t].$$

Training issues with neural network

► Solving a single IVP on $[0, T]$ with a neural network can require a large number of collocation points and a long training time.

Training issues with neural network

► Solving a single IVP on $[0, T]$ with a neural network can require a large number of collocation points and a long training time.

► The obtained solution can not be used to solve the same ordinary differential equation with a different initial condition.
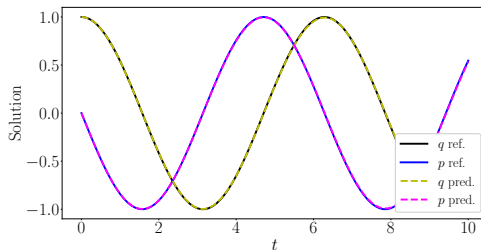
Training issues with neural network

► Solving a single IVP on $[0, T]$ with a neural network can require a large number of collocation points and a long training time.

► The obtained solution can not be used to solve the same ordinary differential equation with a different initial condition.

Figure: Solution comparison after reaching a loss value of 10^{-5}. The training time is of 87 seconds (7500 epochs with 1000 new collocation points randomly sampled at each of them).

Training issues with neural network

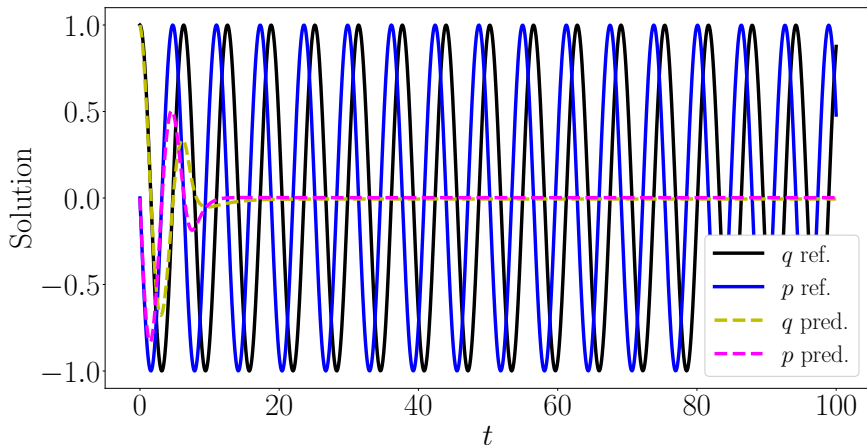It is hard to solve initial value problems over long time intervals.

Figure: Solution comparison after 10000 epochs.

Forward invariant sets: Flow map approach

▶ Suppose $\phi^t(\mathbf{x}_0) \in \Omega \subset \mathbb{R}^d$ for $t \geq 0$.

▶ We can then work with $\mathcal{N}_\theta : [0, \Delta t] \times \Omega \to \mathbb{R}^d$, where[1]

$$\mathcal{L}(\theta) := \frac{1}{C} \sum_{c=1}^{C} \left\| \mathcal{N}_\theta' \left(t_c, \mathbf{x}_0^c \right) - \mathcal{F} \left(\mathcal{N}_\theta \left(t_c, \mathbf{x}_0^c \right) \right) \right\|_2^2 \to \min .$$
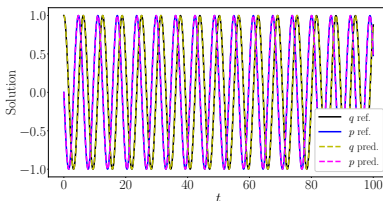

Figure: Network trained with $\Delta t = 1$ and applied up to $T = 100$.

[1] Sifan Wang and Paris Perdikaris. "Long-time integration of parametric evolution equations with physics-informed DeepONets". In: *Journal of Computational Physics* 475 (2023), p. 111855.

Another option for long-time simulations

▶ In case it is not known of a forward invariant set Ω, one could split the time domain $[0, T]$ into smaller sub-intervals of size $[0, \Delta t]$, and solve a sequence of IVPs.

Another option for long-time simulations

- In case it is not known of a forward invariant set Ω, one could split the time domain $[0, T]$ into smaller sub-intervals of size $[0, \Delta t]$, and solve a sequence of IVPs.

- Let $t_n = n\Delta t$, $n = 0, ..., N$, $\Delta t = T/N$. We can solve in sequence the initial value problems

$$\begin{cases} \mathbf{x}'_n(t) = \mathcal{F}(\mathbf{x}_n(t)) \in \mathbb{R}^d, \ t \in [t_n, t_n + \Delta t], \\ \mathbf{x}_n(t_n) = \hat{\mathbf{x}}_n, \end{cases}$$

with $t \mapsto \mathcal{N}_{\theta_n}(t - t_n; \hat{\mathbf{x}}_n)$. We set $\hat{\mathbf{x}}_0 = \mathbf{x}_0$ and $\hat{\mathbf{x}}_{n+1} = \mathcal{N}_{\theta_n}(\Delta t; \hat{\mathbf{x}}_n)$.

Another option for long-time simulations

▶ In case it is not known of a forward invariant set Ω, one could split the time domain $[0, T]$ into smaller sub-intervals of size $[0, \Delta t]$, and solve a sequence of IVPs.

▶ Let $t_n = n\Delta t$, $n = 0, ..., N$, $\Delta t = T/N$. We can solve in sequence the initial value problems

$$\begin{cases} \mathbf{x}'_n(t) = \mathcal{F}(\mathbf{x}_n(t)) \in \mathbb{R}^d, \ t \in [t_n, t_n + \Delta t], \\ \mathbf{x}_n(t_n) = \hat{\mathbf{x}}_n, \end{cases}$$

with $t \mapsto \mathcal{N}_{\theta_n}(t - t_n; \hat{\mathbf{x}}_n)$. We set $\hat{\mathbf{x}}_0 = \mathbf{x}_0$ and $\hat{\mathbf{x}}_{n+1} = \mathcal{N}_{\theta_n}(\Delta t; \hat{\mathbf{x}}_n)$.

▶ **Problem**: Training these N networks can be extremely expensive.

Extreme learning machines (ELMs)

▶ ELMs are networks where **only part of the weights are trained**.

▶ We consider two-layer ELMs $\mathcal{N}_\theta\left(\cdot;\mathbf{x}_0\right): [0, \Delta t] \to \mathbb{R}^d$ of the form

$$\mathcal{N}_\theta\left(t;\mathbf{x}_0\right) = \mathbf{x}_0 + \theta\left(\sigma\left(\mathbf{a}t + \mathbf{b}\right) - \sigma(\mathbf{b})\right), \ \mathbf{a}, \mathbf{b} \in \mathbb{R}^H, \tag{1}$$

with $\theta \in \mathbb{R}^{d \times H}$, $a_h, b_h \in \mathcal{U}\left([-1, 1]\right)$, $h = 1, \ldots, H$.

Theorem: Approximation properties of ELMs

For any smooth activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and any set $\{(t_c, \mathbf{y}_c)\}_{c=1}^C$, if $H = C$ then with probabilty one there is matrix $\theta \in \mathbb{R}^{d \times H}$ in (1) such that $\mathcal{N}_\theta'\left(t_c; \mathbf{x}_0\right) = \mathbf{y}_c$ for $c = 1, ..., C$[a].

---

[a]Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: Theory and applications". In: *Neurocomputing* 70.1-3 (2006), pp. 489–501.

9

Exploiting the efficiency of ELMs to solve IVPs

► One option would be to split the time domain into subintervals and solve each of the obtained IVPs with an ELM[2]. Each of these ELMs could be trained as a normal physics-informed neural network.

---

[2]Gianluca Fabiani et al. "Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 33.4 (2023), p. 043128.

Exploiting the efficiency of ELMs to solve IVPs

▶ One option would be to split the time domain into subintervals and solve each of the obtained IVPs with an ELM[2]. Each of these ELMs could be trained as a normal physics-informed neural network.

▶ Possible problems with this approach are:
  ▶ Solutions completely based on networks are not always reliable.
  ▶ Increased accuracy has to come from more collocation points.

[2]Fabiani et al., "Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs".

Exploiting the efficiency of ELMs to solve IVPs

▶ One option would be to split the time domain into subintervals and solve each of the obtained IVPs with an ELM[2]. Each of these ELMs could be trained as a normal physics-informed neural network.

▶ Possible problems with this approach are:
  ▶ Solutions completely based on networks are not always reliable.
  ▶ Increased accuracy has to come from more collocation points.

▶ We consider a hybrid approach: replacing the coarse propagator in the Parareal method with an ELM.

---

[2]Fabiani et al., "Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs".

Parareal method

▶ Divide the time interval $[0, T]$ into N sub-intervals of size $\Delta t = T/N$.

▶ Choose a (cheap) coarse integrator $\varphi_C^{\Delta t}$, and a more accurate fine integrator $\varphi_F^{\Delta t}$.

▶ Update iteratively the approximate solution $\mathbf{x}_n^i$ at the time $t = t_n = n\Delta t$ as follows:

$$\mathbf{x}_0^0 = \mathbf{x}_0, \ \mathbf{x}_n^0 = \varphi_C^{\Delta t}\left(\mathbf{x}_{n-1}^0\right), \ n \geq 1$$

$$\mathbf{x}_{n+1}^{i+1} = \varphi_F^{\Delta t}\left(\mathbf{x}_n^i\right) + \varphi_C^{\Delta t}\left(\mathbf{x}_n^{i+1}\right) - \varphi_C^{\Delta t}\left(\mathbf{x}_n^i\right), \ n \geq 1, \ i \geq 0.$$

Convergence of Parareal

Theorem: Convergence of the Parareal method

Let $\varphi_F^{\Delta t} = \phi^{\Delta t}$. Assume that there exist $p \in \mathbb{N}$, a set of continuously differentiable functions $c_{p+1}, c_{p+2}, \cdots$, and $\alpha > 0$ such that

$$\varphi_F^{\Delta t}(\mathbf{x}) - \varphi_C^{\Delta t}(\mathbf{x}) = c_{p+1}(\mathbf{x})(\Delta t)^{p+1} + c_{p+2}(\mathbf{x})(\Delta t)^{p+2} + \cdots, \quad \text{and}$$
$$\left\| \varphi_F^{\Delta t}(\mathbf{x}) - \varphi_C^{\Delta t}(\mathbf{x}) \right\|_2 \leq \alpha(\Delta t)^{p+1}$$

for every $\mathbf{x} \in \mathbb{R}^d$, and also that there exists $\beta > 0$ such that

$$\left\| \varphi_C^{\Delta t}(\mathbf{x}) - \varphi_C^{\Delta t}(\mathbf{y}) \right\|_2 \leq (1 + \beta\Delta t) \left\| \mathbf{x} - \mathbf{y} \right\|_2, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

Then there exists a positive constant γ such that

$$\left\| \mathbf{x}(t_n) - \mathbf{x}_n^i \right\|_2 \leq \frac{\alpha}{\gamma} \frac{\left(\gamma(\Delta t)^{p+1} \right)^{i+1}}{(i+1)!} (1 + \beta\Delta t)^{n-i-1} \prod_{j=0}^{i} (n-j).$$

Parareal method with ELMs

Algorithm Hybrid Parareal algorithm based on ELMs

1: **while** $i <$ `max_it` **and** `error` $>$ `tol` **do**
2:      `error` $\leftarrow 0$
3:      $\mathbf{x}_{n+1}^F \leftarrow \varphi_F^{\Delta t}(\mathbf{x}_n^i), \; n = 0$ **to** $N-1$ $\triangleright$ Fine integrator, **Parallel**
4:      $\mathbf{x}_0^i \leftarrow \mathbf{x}_0$
5:      **for** $n = 0$ **to** $N-1$ **do**
6:          Find $\theta_n^{i+1} = \arg\min_{\theta \in \mathbb{R}^{H \times d}} \sum_{c=1}^{C} \left\| \mathcal{N}_\theta' \left(t_c; \mathbf{x}_n^{i+1} \right) - \mathcal{F} \left(\mathcal{N}_\theta \left(t_c; \mathbf{x}_n^{i+1} \right) \right) \right\|_2^2$
7:          $\mathbf{x}_{n+1}^S \leftarrow \mathcal{N}_{\theta_n^{i+1}} \left(\Delta t; \mathbf{x}_n^{i+1} \right)$ $\triangleright$ Next coarse approximation
8:          $\mathbf{x}_{n+1}^{i+1} \leftarrow \mathbf{x}_{n+1}^F + \mathbf{x}_{n+1}^S - \mathbf{x}_{n+1}^{S,-1}$ $\triangleright$ Parareal correction
9:          $\mathbf{x}_{n+1}^{S,-1} \leftarrow \mathbf{x}_{n+1}^S$
10:        `error` $\leftarrow \max \left\{ \text{error}, \left\| \mathbf{x}_{n+1}^{i+1} - \mathbf{x}_{n+1}^i \right\|_2 \right\}$
11:      **end for**
12:      $i \leftarrow i + 1$
13: **end while**

The resulting piecewise smooth approximation

▶ After the training, we end up with the hybrid Parareal
approximations $\mathbf{x}_n$ at the time instants $t_n = n\Delta t$

The resulting piecewise smooth approximation

▶ After the training, we end up with the hybrid Parareal approximations $\mathbf{x}_n$ at the time instants $t_n = n\Delta t$

▶ But we also have a piecewise smooth approximation of the solution given by

$$\tilde{\mathbf{x}}(t) = \mathcal{N}_{\theta_n}\left(t - t_n; \mathbf{x}_n\right), \ t \in [t_n, t_{n+1}).$$

▶ This function is the one we plot in the following experiments.

Example with the SIR problem

Timing breakdown	ELM	Flow
Offline training phase	0s	~20 minutes
Average cost coarse step in the zeroth iterate	0.0009773s	0.0002729s
Total	0.3940s	0.8047s

ELM
Average computational time: 0.39s

Flow map approach
Average computational time: 0.8s
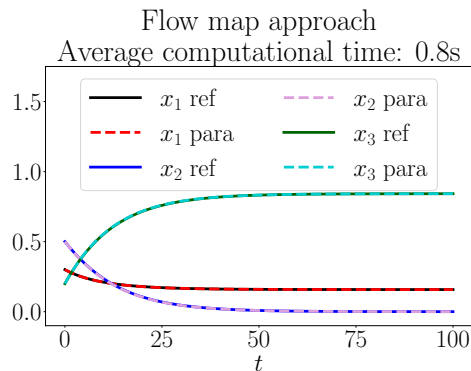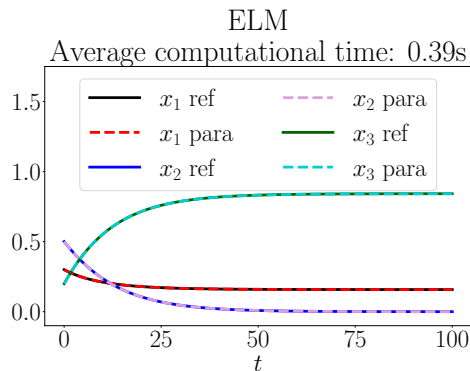
Example with the Arenstorf orbit

Timing breakdown	ELM
Average cost coarse step in the zeroth iterate	0.001912s
Average cost to produce the solution	9.7957s

Average computational time: 9.8s
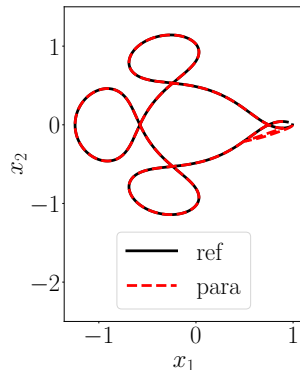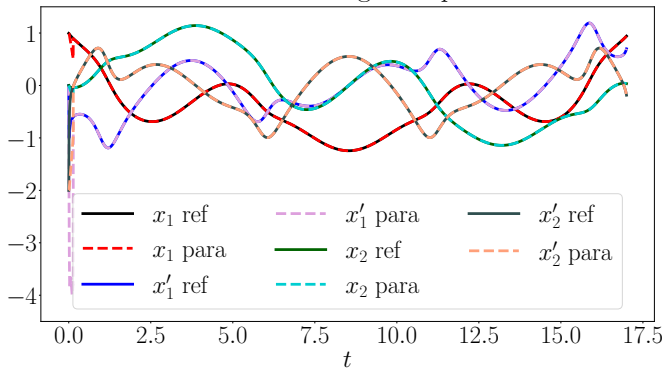
Example with the Lorenz equation

Timing breakdown	Uniform	Lobatto
Average cost coarse step in the zeroth iterate	0.0009430s	0.0009371s
Average cost to produce the solution	1.8312s	1.8184s

Uniform nodes
Average computational time: 1.83s

Lobatto nodes
Average computational time: 1.82s
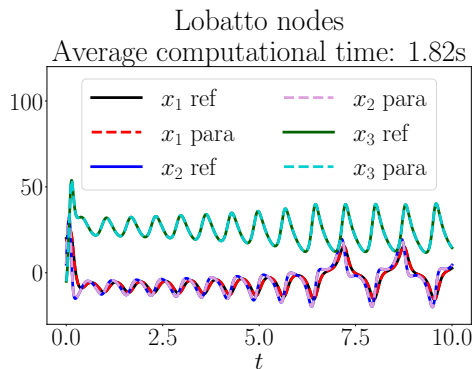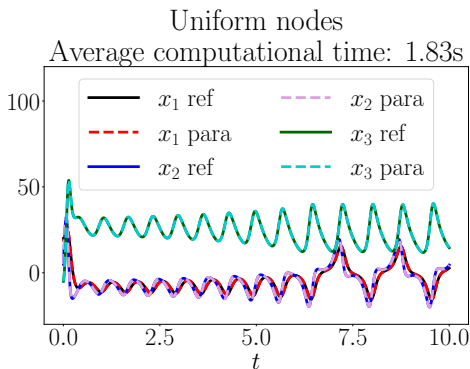
THANK YOU FOR THE ATTENTION

davide.murari@ntnu.no

Link to the preprint: davidemurari.com/ELM.pdf

📄 Fabiani, Gianluca et al. "Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 33.4 (2023), p. 043128.

📄 Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: Theory and applications". In: *Neurocomputing* 70.1-3 (2006), pp. 489–501.

📄 Wang, Sifan and Paris Perdikaris. "Long-time integration of parametric evolution equations with physics-informed DeepONets". In: *Journal of Computational Physics* 475 (2023), p. 111855.