

Exemplo de Equações Diferenciais Universais - Lotka Volterra

Laura Costa Pereira Miranda – lauram@lncc.br

20 de maio de 2021

O exemplo

No exemplo é dado o seguinte sistema de Equações Diferenciais Ordinárias:

$$\dot{x} = \alpha x + \beta xy$$

$$\dot{y} = \gamma xy - \delta y$$

O sistema de EDOU

Supondo que α e δ sejam conhecidos à priori, queremos resolver o seguinte sistema de EDOUs:

$$\begin{aligned}\dot{x} &= \alpha x + U_1(x, y) \\ \dot{y} &= U_2(x, y) - \delta y\end{aligned}$$

no qual $U : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, tal que $U = [U_1, U_2]$, é chamada aproximador universal e representa a rede neural a ser treinada.

O sistema de EDOU

Outra abordagem é usando a rede para o sistema inteiro, de forma que:

$$\dot{x} = U_1(x, y)$$

$$\dot{y} = U_2(x, y)$$

O código

Bibliotecas utilizadas:

- OrdinaryDiffEq;
- Plots;
- Flux;
- DiffEqFlux;
- Optim.

O sistema de EDOs

Costituído de funções que correspondem ao sistema de EDOs, no qual:

- a condição inicial se dá por: $u_0 = [0.5, 1.5]$;
- o domínio de t é dado por $t \in [0, 10]$;
- os parâmetros do sistema são: $\alpha = 1.25$, $\beta = 0.7$, $\delta = 1.1$ e $\gamma = 0.9$;
- a solução é aproximada por um método Runge-Kutta de 7ª ordem com passo de tempo adaptativo, o Vern7(), com erros absoluto de relativo da ordem de $1 \cdot 10^{-11}$ e $1 \cdot 10^{-10}$, respectivamente.

Rede Neural

Os dados de treinamento são gerados à partir do sistema de EDOs.

- quantidade de dados para treinamento da rede: 100;
- intervalo de treinamento: $t \in [0, 4.5]$.

Rede Neural

Em ambas as abordagens, as redes possuem:

- 2 camadas internas de 9 neurônios;
- entrada e saída por 2 neurônios;
- funções de ativação: gaussiana na primeira camada; e sigmóide na segunda camada.

Rede Neural

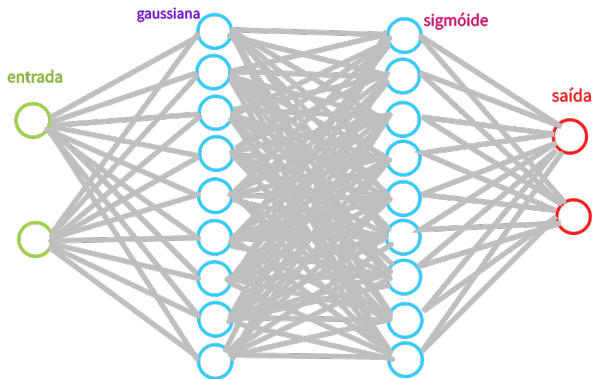


Figura: Rede neural utilizada

A função custo

Dentro da função custo, a solução encontrada pela rede é comparada com os dados de treinamento, através da seguinte função:

$$C = \frac{1}{n} \left(\sum_{i=0}^n (\text{teste}_i - \text{dado}_i)^2 \right)^2 .$$

Erro entre as abordagens

A termos de comparação, foi calculado o erro absoluto entre as soluções obtidas em ambas as redes. O erro calculado foi o erro absoluto, no qual:

$$erro = |output_1 - output_2|,$$

onde $output_1$ é a saída obtida pela rede apenas na parte não-linear do sistema, e $output_2$ é a saída obtida pela rede que simula todo o sistema.

Resultados

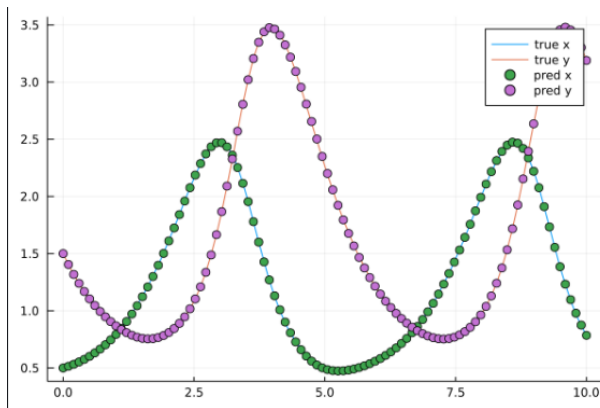


Figura: Primeiro teste

Resultados

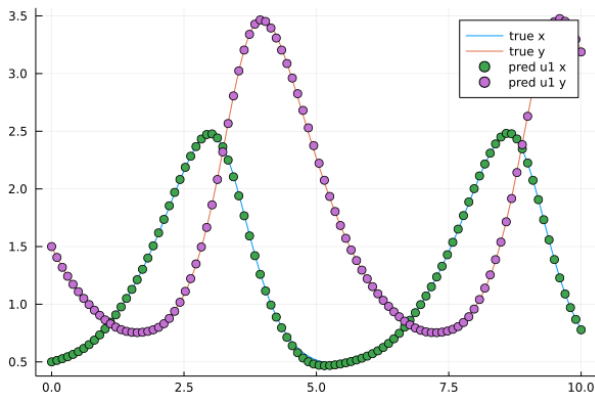


Figura: Teste com as duas redes; primeira rede

Resultados

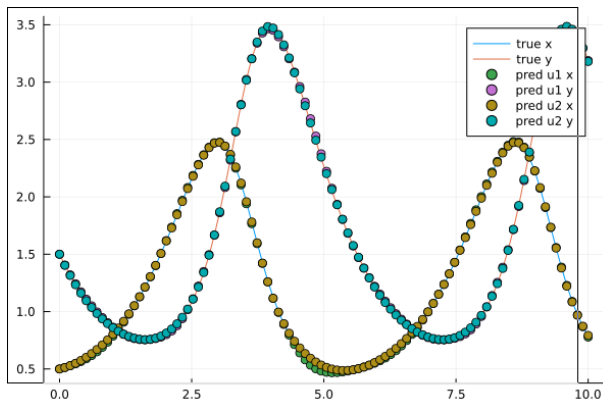


Figura: Teste com as duas redes

Resultados

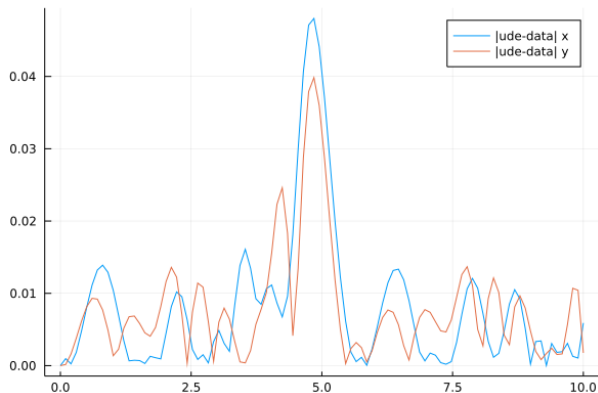


Figura: Erro entre a primeira rede e os dados

Resultados

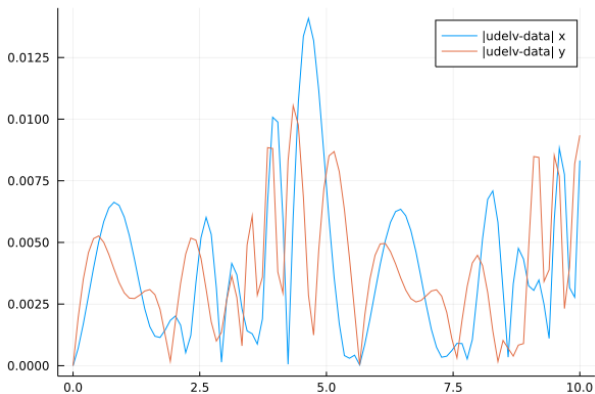


Figura: Erro entre a segunda rede e os dados

Resultados

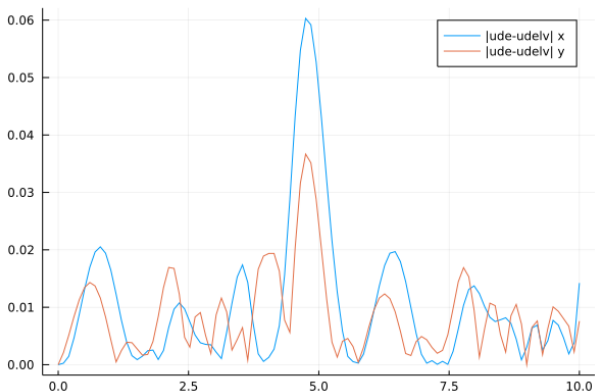


Figura: Erro entre ambas as redes

Resultados

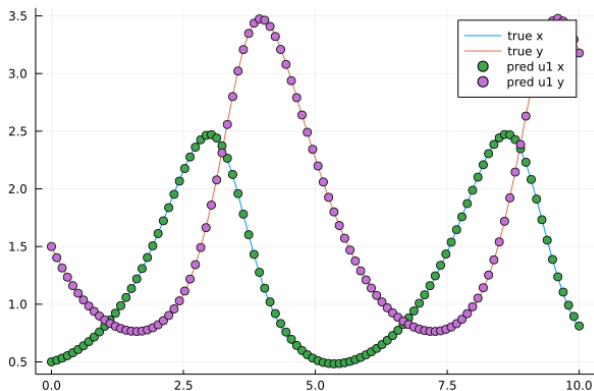


Figura: Teste com Runge-Kutta de 4ª ordem adaptativo (primeira rede)

Resultados

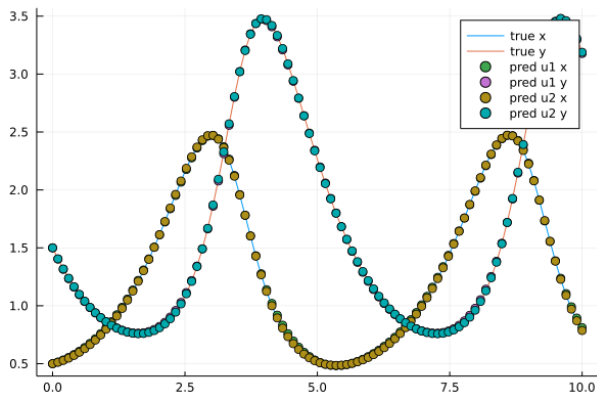


Figura: Teste com Runge-Kutta de 4ª ordem adaptativo (segunda rede)

Resultados

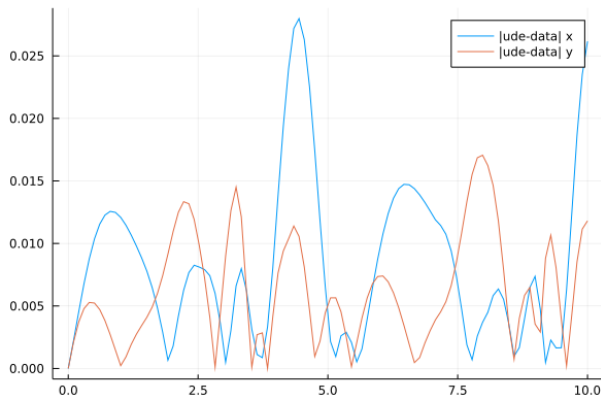


Figura: Teste com Runge-Kutta de 4ª ordem adaptativo (erro entre a primeira rede e os dados)

Resultados

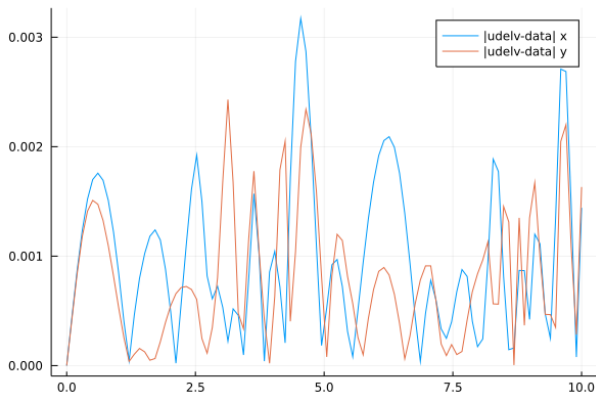


Figura: Teste com Runge-Kutta de 4ª ordem adaptativo (erro entre a segunda rede e os dados)

Resultados

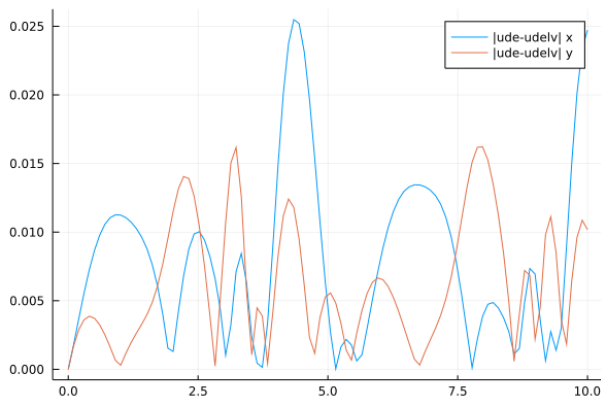







Figura: Teste com Runge-Kutta de 4ª ordem adaptativo (erro entre a primeira e a segunda rede)

Referências

-  DiffEqFlux, Disponível em:
<<https://diffeqflux.sciml.ai/stable/Scimltrain/>>
-  Flux, Disponível em:
<<https://fluxml.ai/Flux.jl/v0.4/training/optimisers.html>>
-  Optim, Disponível em:
<<https://juliansolvers.github.io/Optim.jl/stable/>>
-  OrdinaryDiffEq, Disponível em:
<https://diffeq.sciml.ai/stable/tutorials/ode_example/ode_example>
-  Plots, Disponível em:
<<http://docs.juliaplots.org/latest/>>