Solving inverse problems with PINNs

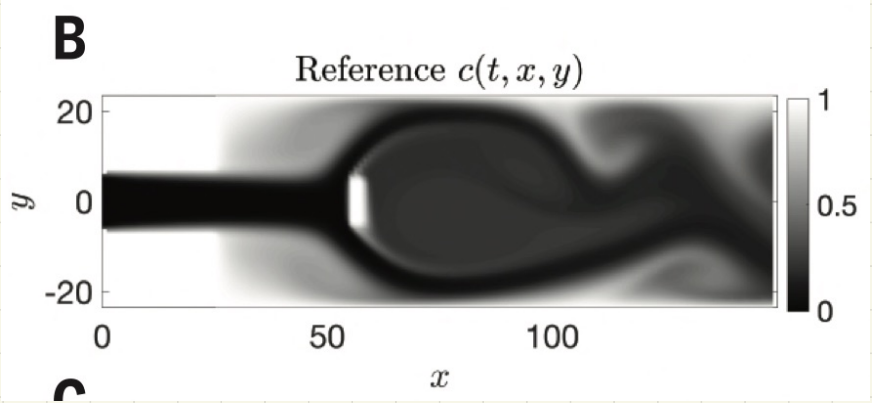
Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations

MAZIAR RAISSI , ALIREZA YAZDANI , AND , GEORGE EM KARNIADAKIS Authors Info & Affiliations

SCIENCE • 30 Jan 2020 • Vol 367, Issue 6481 • pp. 1026-1030 • DOI: 10.1126/science.aaw4741

Inverse problem learning goal

Given: Training data: $\{t^n, x^n, y^n, z^n, c^n\}_{n=1}^N$

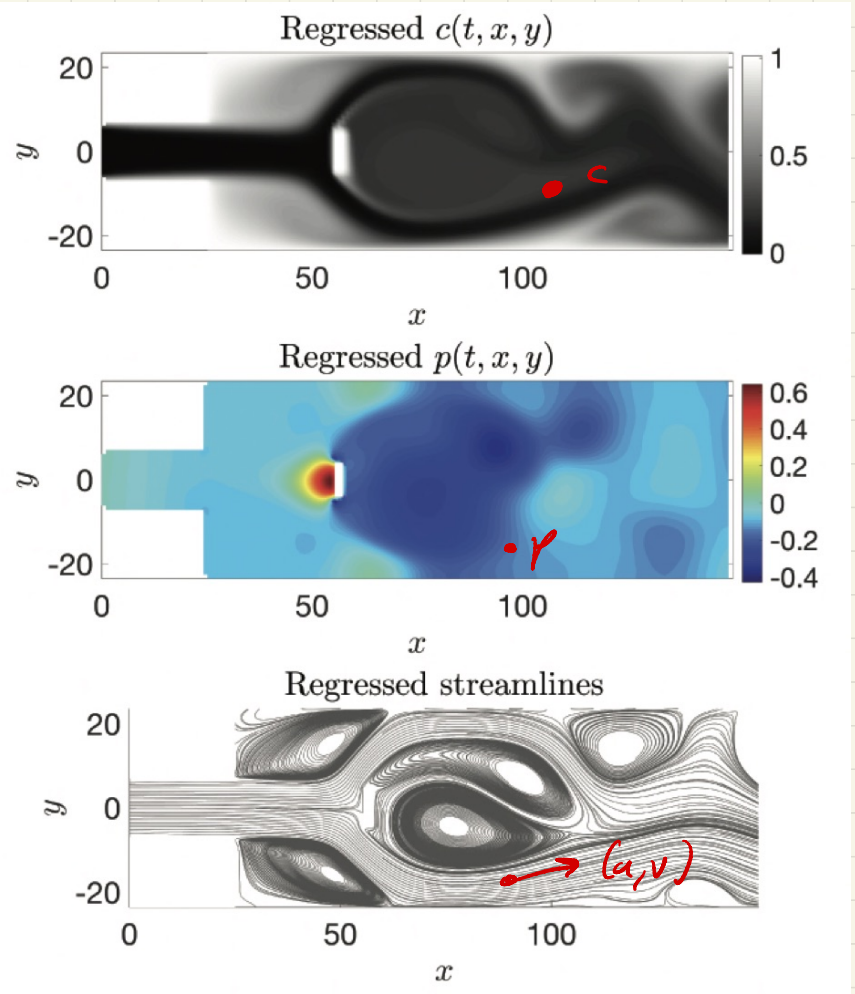


Find: Learned model: $(t, x, y, z) \longmapsto (c, u, v, w, p)$

Should match data and "obey physics"

Learned model

$$(t, x, y, z) \longmapsto (c, u, v, w, p)$$



Physics

Continuity equation:

$$c_t + uc_x + vc_y + wc_z = \mathrm{Pe}^{-1}(c_{xx} + c_{yy} + c_{zz}),$$

Navier-Stokes equations:

$$\begin{aligned}
u_t + uu_x + vu_y + wu_z &= -p_x + \mathrm{Re}^{-1}(u_{xx} + u_{yy} + u_{zz}),\\
v_t + uv_x + vv_y + wv_z &= -p_y + \mathrm{Re}^{-1}(v_{xx} + v_{yy} + v_{zz}),\\
w_t + uw_x + vw_y + ww_z &= -p_z + \mathrm{Re}^{-1}(w_{xx} + w_{yy} + w_{zz}),\\
u_x + v_y + w_z &= 0.
\end{aligned}$$

Training setup

Residual points : $\{t^m, x^m, y^m, z^m\}_{m=1}^{M}$

Residuals :

$$
\begin{aligned}
e_1 &:= c_t + uc_x + vc_y + wc_z - \mathrm{Pe}^{-1}(c_{xx} + c_{yy} + c_{zz}),\\
e_2 &:= u_t + uu_x + vu_y + wu_z + p_x - \mathrm{Re}^{-1}(u_{xx} + u_{yy} + u_{zz}),\\
e_3 &:= v_t + uv_x + vv_y + wv_z + p_y - \mathrm{Re}^{-1}(v_{xx} + v_{yy} + v_{zz}),\\
e_4 &:= w_t + uw_x + vw_y + ww_z + p_z - \mathrm{Re}^{-1}(w_{xx} + w_{yy} + w_{zz}),\\
e_5 &:= u_x + v_y + w_z.
\end{aligned}
$$

Loss :

$$
MSE = \frac{1}{N}\sum_{n=1}^{N}\left|c(t^n, x^n, y^n, z^n) - c^n\right|^2 \quad \longleftarrow \text{match data}
$$

$$
+\sum_{i=1}^{5}\frac{1}{M}\sum_{m=1}^{M}\left|e_i(t^m, x^m, y^m, z^m)\right|^2 \quad \longleftarrow \text{physics}
$$

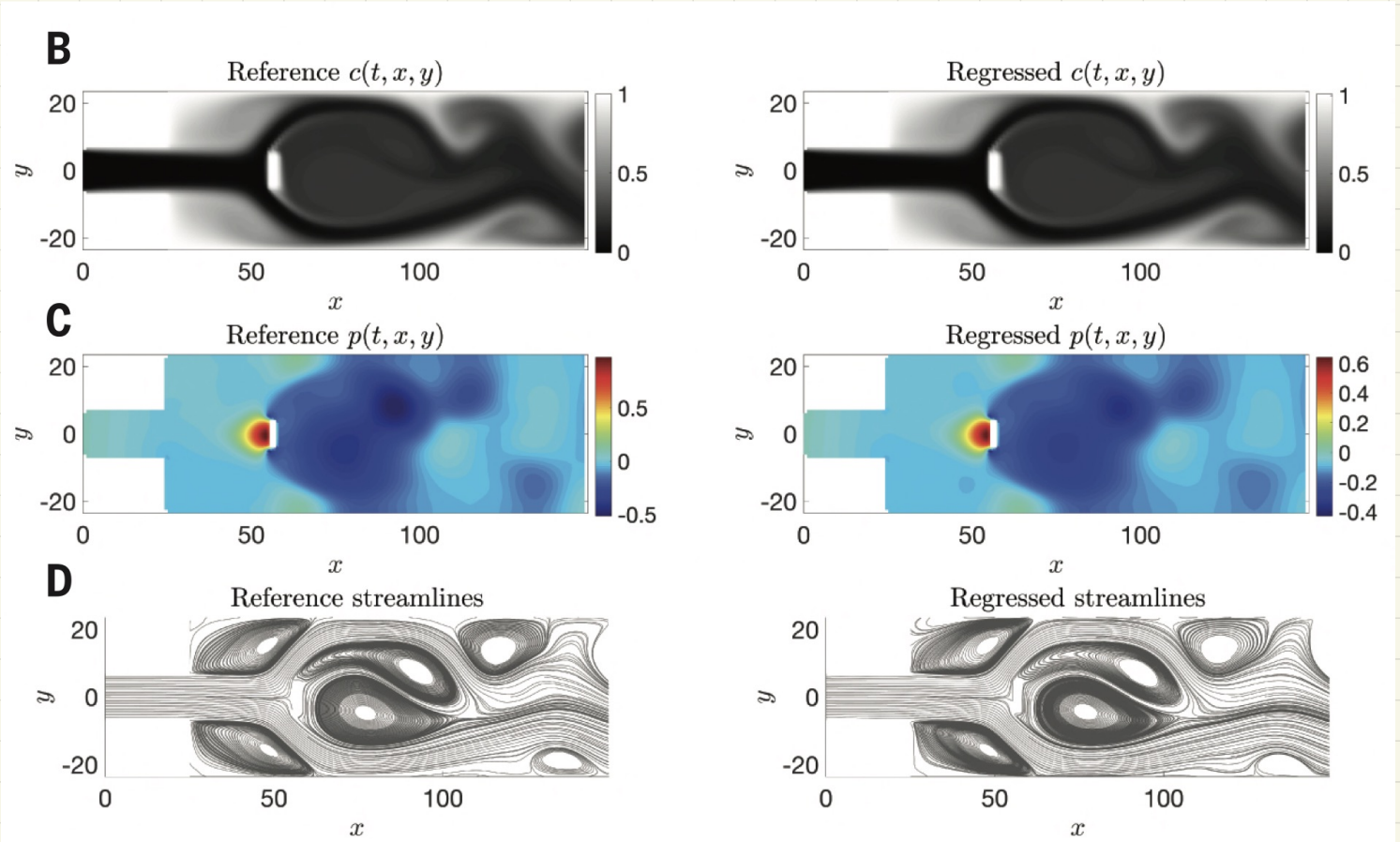
no BC !

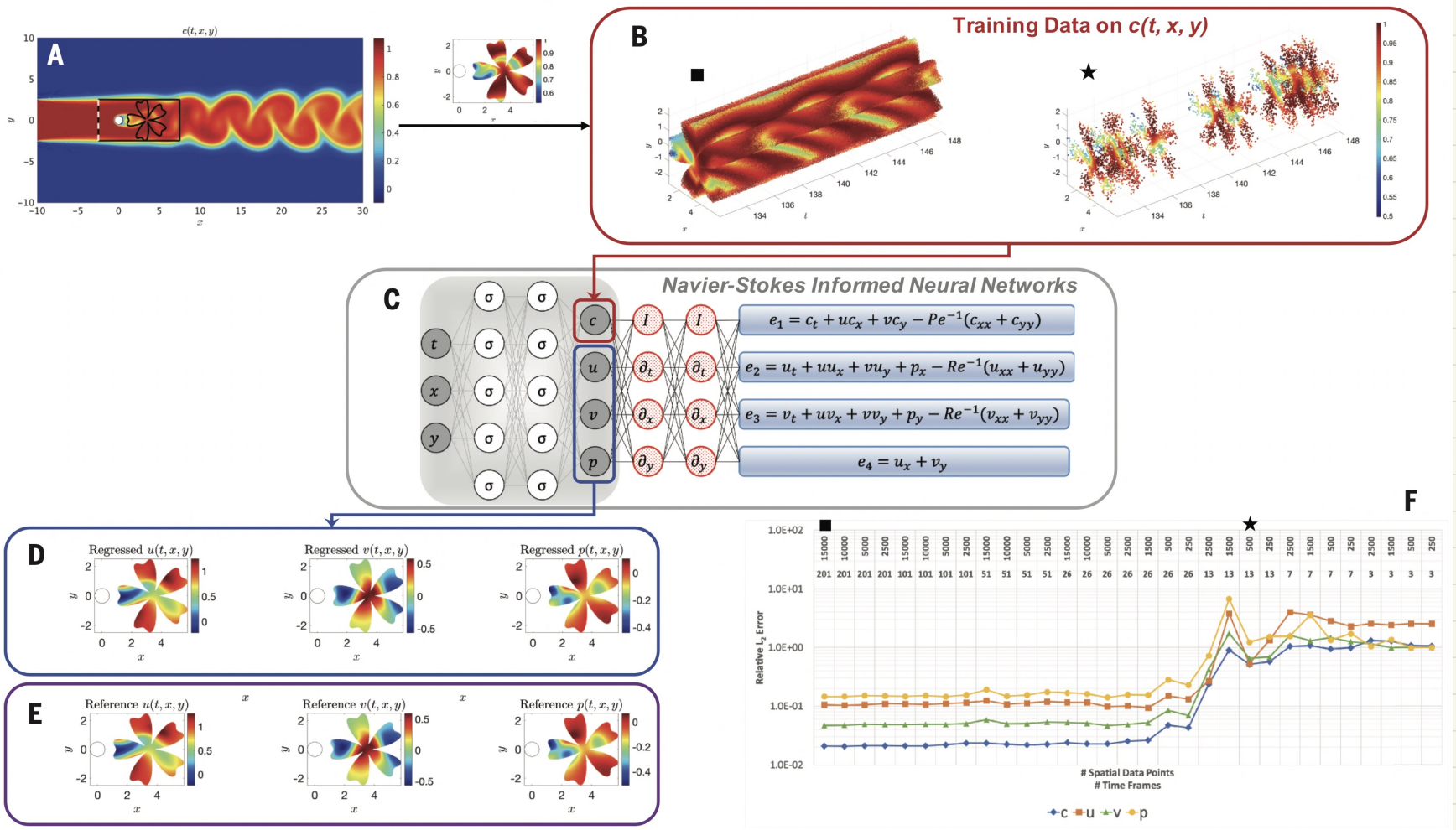
Notes

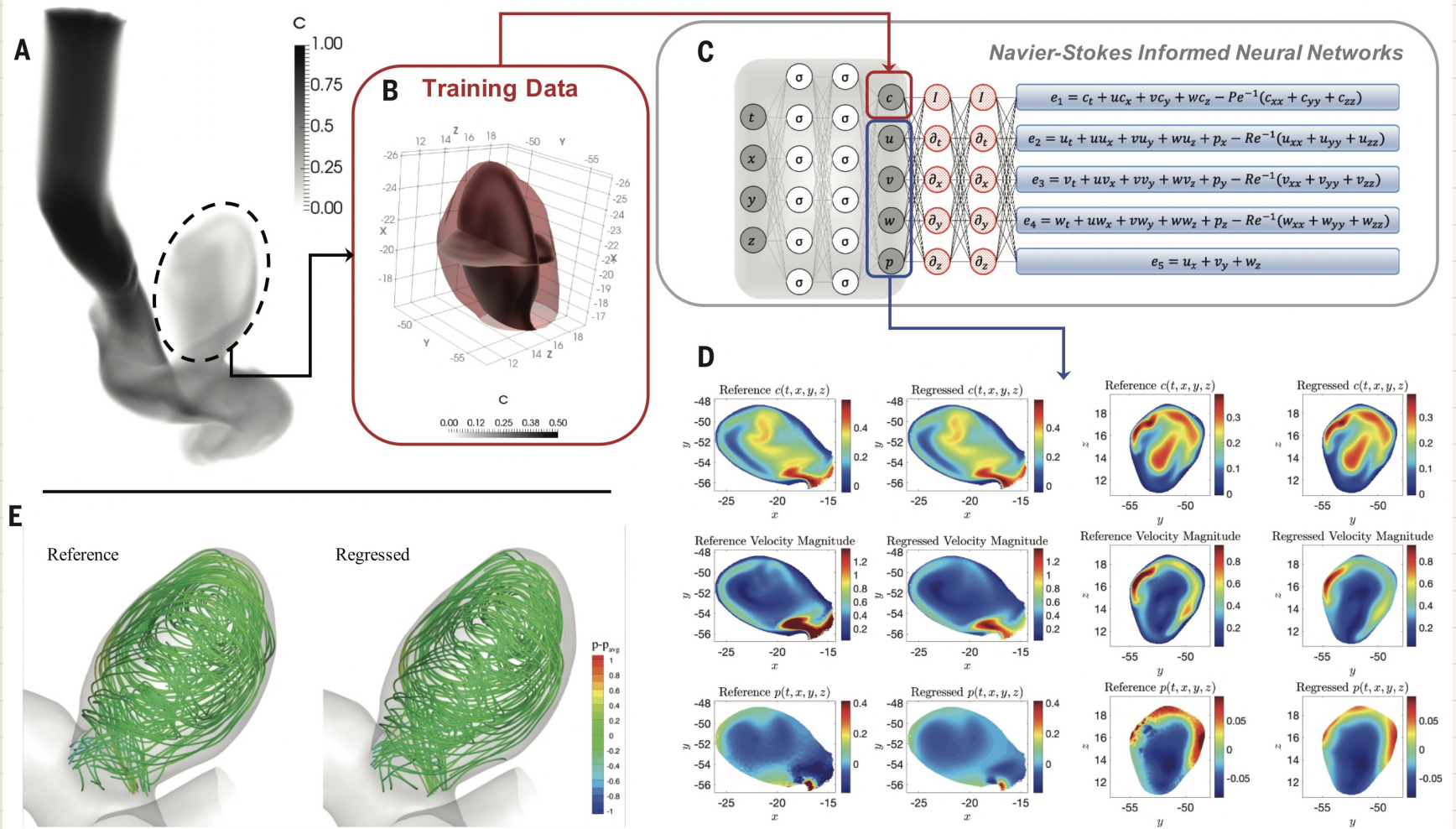
- Network: 10 layers
  250 neurons per layer
- swish activation function:

$$\text{swish}(x) := x\,\text{sigmoid}(x) = x/(1+\exp(-x))$$

Verification against test data




A
$c(t,x,y)$
B
Training Data on $c(t, x, y)$
C
Navier-Stokes Informed Neural Networks
$e_1 = c_t + uc_x + vc_y - Pe^{-1}(c_{xx} + c_{yy})$
$e_2 = u_t + uu_x + vu_y + p_x - Re^{-1}(u_{xx} + u_{yy})$
$e_3 = v_t + uv_x + vv_y + p_y - Re^{-1}(v_{xx} + v_{yy})$
$e_4 = u_x + v_y$
D
Regressed $u(t,x,y)$
Regressed $v(t,x,y)$
Regressed $p(t,x,y)$
E
Reference $u(t,x,y)$
Reference $v(t,x,y)$
Reference $p(t,x,y)$
F
Relative L_2 Error
Spatial Data Points
Time Frames
c u v p


A
C
1.00
0.75
0.5
0.25
0.00
B Training Data
C
0.00 0.12 0.25 0.38 0.50
C
Navier-Stokes Informed Neural Networks
$e_1 = c_t + uc_x + vc_y + wc_z - Pe^{-1}(c_{xx} + c_{yy} + c_{zz})$
$e_2 = u_t + uu_x + vu_y + wu_z + p_x - Re^{-1}(u_{xx} + u_{yy} + u_{zz})$
$e_3 = v_t + uv_x + vv_y + wv_z + p_y - Re^{-1}(v_{xx} + v_{yy} + v_{zz})$
$e_4 = w_t + uw_x + vw_y + ww_z + p_z - Re^{-1}(w_{xx} + w_{yy} + w_{zz})$
$e_5 = u_x + v_y + w_z$
D
Reference $c(t,x,y,z)$
Regressed $c(t,x,y,z)$
Reference Velocity Magnitude
Regressed Velocity Magnitude
Reference $p(t,x,y,z)$
Regressed $p(t,x,y,z)$
E
Reference
Regressed
p-p$_{avg}$

Why not ReLU?

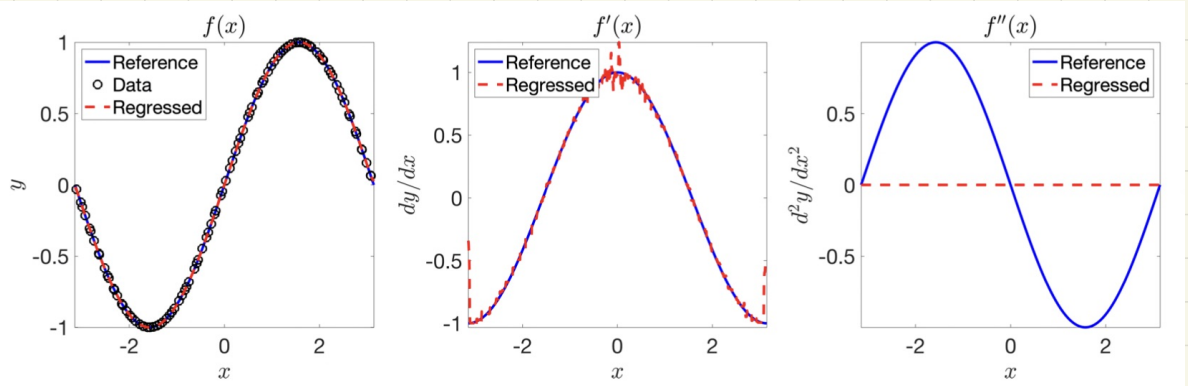
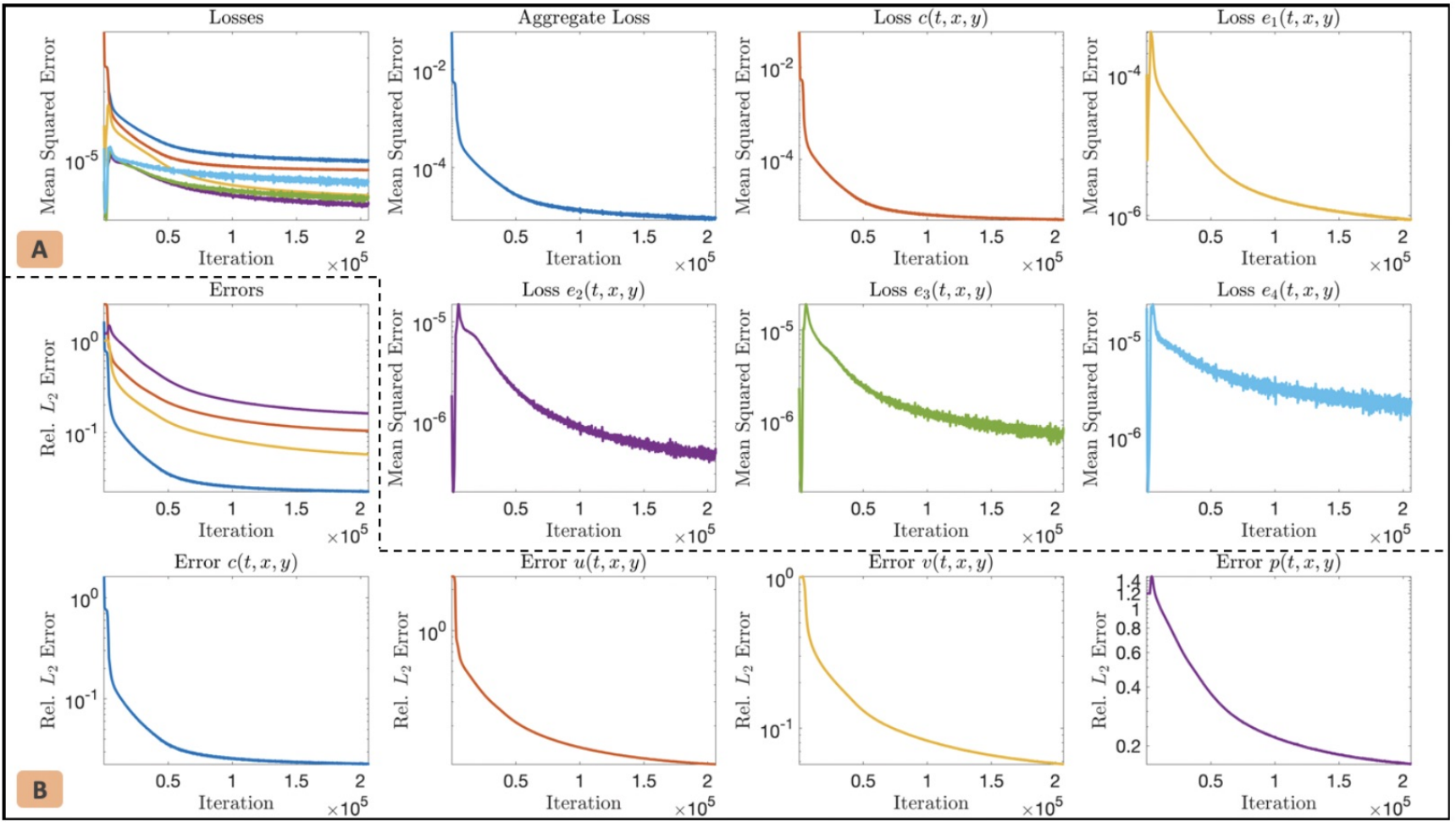


Fig. S2. Regressing data generated from $f(x) = \sin(x)$ using the ReLU activation function. A neural network with ReLU activation can accurately approximate $f(x)$ (left panel), while it leads to a non-smooth regression for the first derivative of f (middle panel) and a vanishing second derivative of f (right panel).


Losses
Aggregate Loss
Loss $c(t,x,y)$
Loss $e_1(t,x,y)$
Errors
Loss $e_2(t,x,y)$
Loss $e_3(t,x,y)$
Loss $e_4(t,x,y)$
Error $c(t,x,y)$
Error $u(t,x,y)$
Error $v(t,x,y)$
Error $p(t,x,y)$
Mean Squared Error
Rel. L_2 Error
Iteration
$\times 10^5$
A
B