THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 15: PINNs Applications

Sergei V. Kalinin

Physics-Informed Neural Networks

We have:

- a differential equation $g(x, y) = 0$,
- some data $\{x_j, y_j\}$ and
- a neural network $f(x \mid \theta)$ that approximates y.

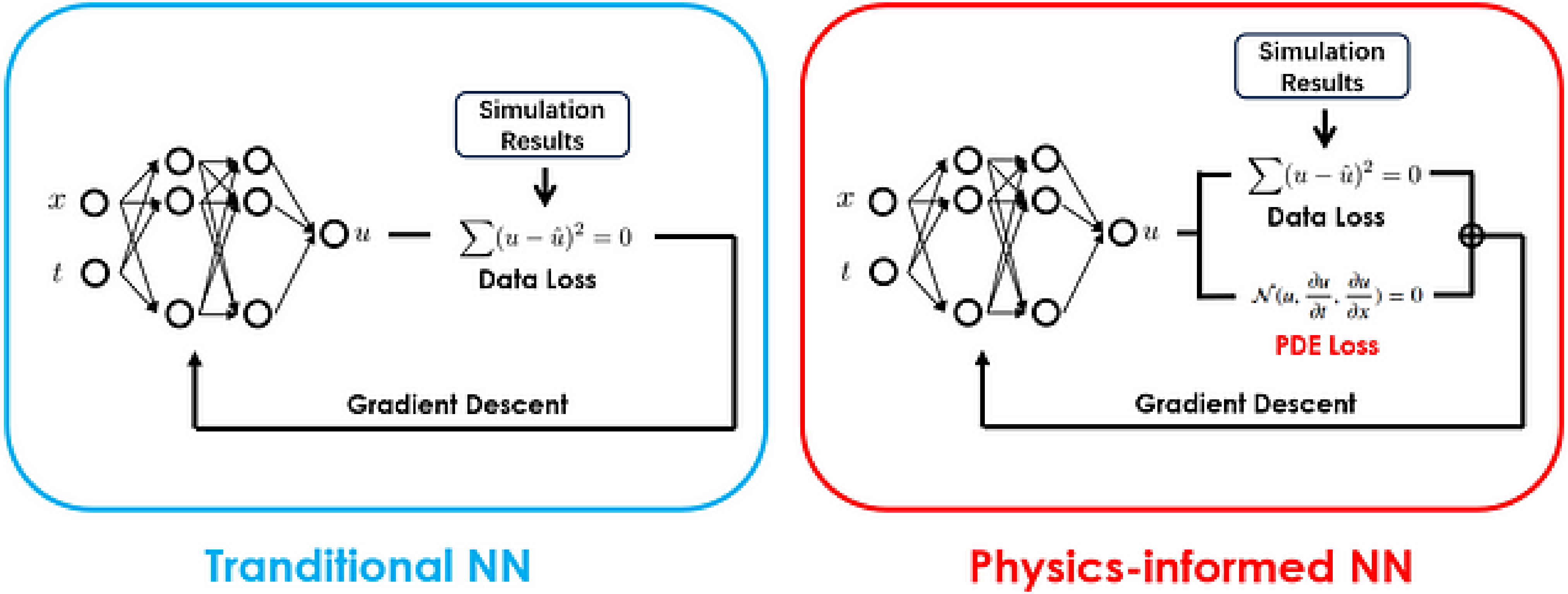
For a PINN, we would get a loss function that looks like the following,

$$Loss_{PINN} = \underbrace{\frac{1}{N}\sum_{j}^{N} ||f(x_j|\theta) - y_j||_2^2}_{\text{Data loss}} + \lambda \underbrace{\frac{1}{M}\sum_{i}^{M} ||g(x_i, f(x_i, |\theta))||_2^2}_{\text{Physics loss}}$$

- Here x_i are *collocation* points. These can be any value we want them to be, usually you would want them to be in the range of values we are interested in.
- The x_j and y_j are our data.
- We can also add a parameter controlling the relative strength of the data loss function and the physics loss function, here we use λ.
- And then just train as you would any other neural network.

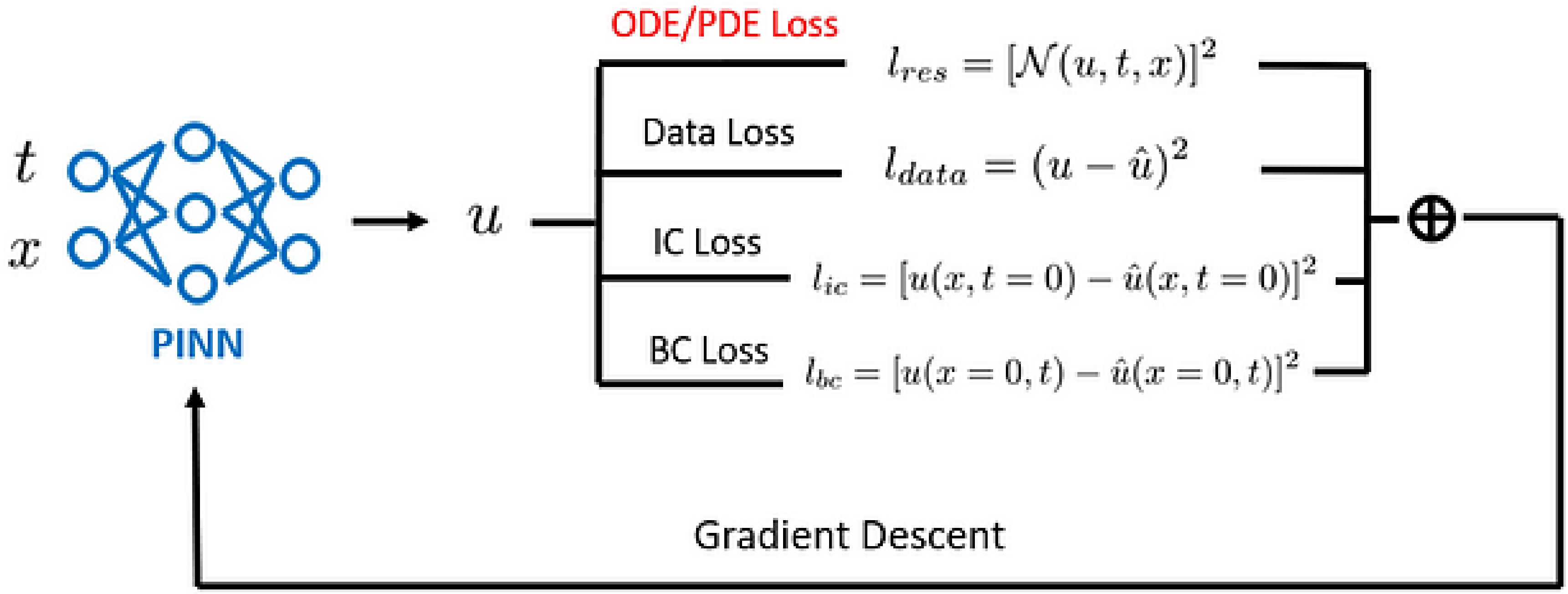
https://medium.com/@theo.wolf/physics-informed-neural-networks-a-simple-tutorial-with-pytorch-f28a890b874a

Back to functions from data

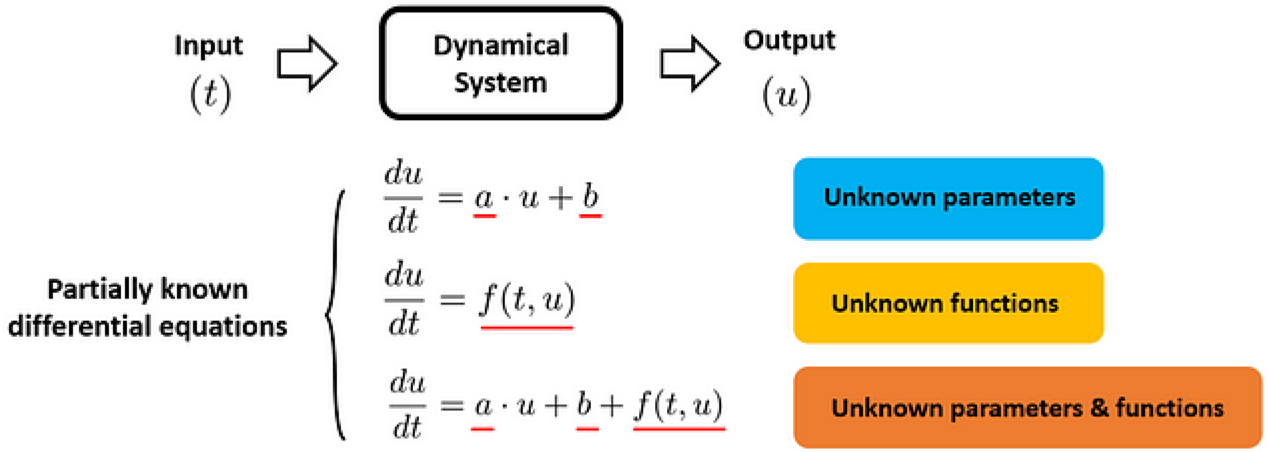


https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

PINN Problems



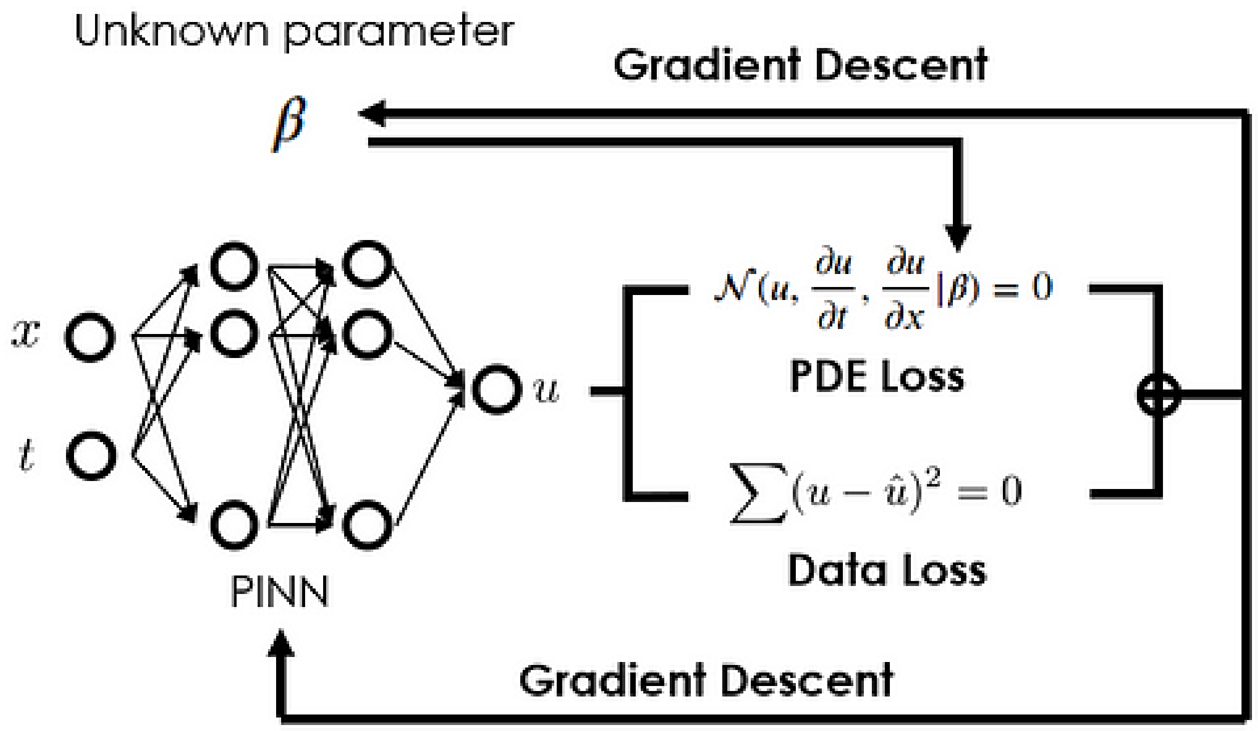
PINN Problems



• The **parameters** of the differential equation are unknown. For example, the governing equations of fluid dynamics are well-established, but the coefficients are highly uncertain.
• The **functional forms** of the differential equations are unknown. For instance, in chemical engineering, the exact functional form of the rate equations may not be fully understood due to the uncertainties in rate-determining steps and reaction pathways.
• Both **functional forms** and **parameters** are unknown. Example is battery state modeling, where the commonly used equivalent circuit model only partially captures the current-voltage relationship (the functional form of the missing physics is therefore unknown). The model itself contains unknown parameters (i.e., resistance and capacitance values).

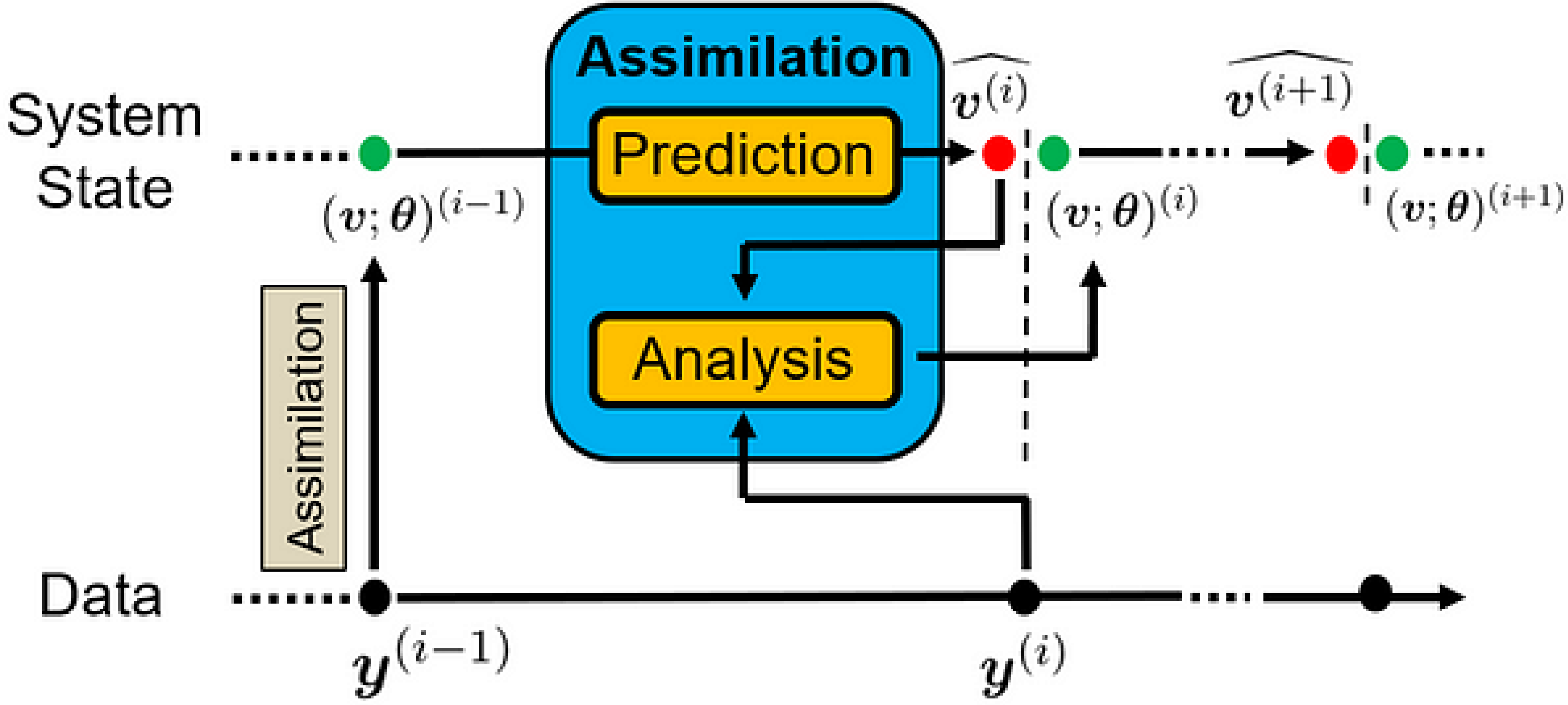
[Discovering Differential Equations with Physics-Informed Neural Networks and Symbolic Regression | by Shuai Guo | Towards Data Science]

Parameter Estimation



https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

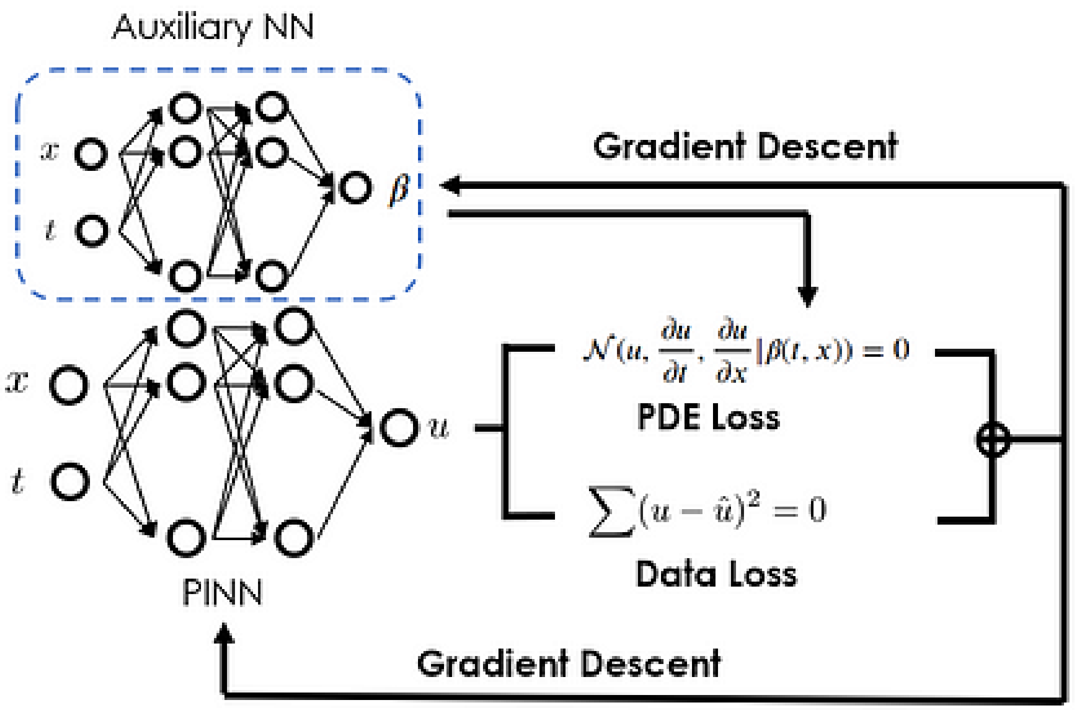
Data Assimilation



https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

System Identification

$$\frac{dp}{dt} = f(\cdot) - min(h \cdot p, H_{max})$$



Unlike traditional methods, PINNs are capable of working with partially known differential equations, thus not confined by a complete equation to run simulations.

https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

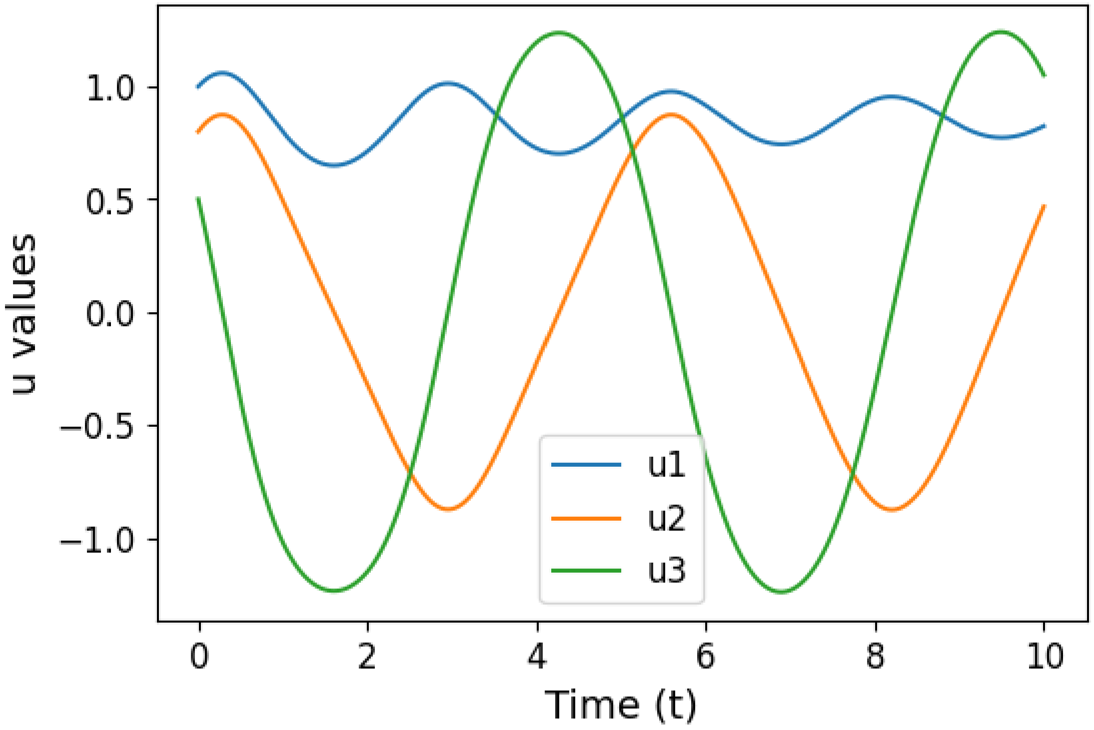
Colab

Craichnan-Orzag Equations

$$\frac{du_1}{dt} = e^{-t/10} u_2 u_3$$

$$\frac{du_2}{dt} = u_1 u_3$$

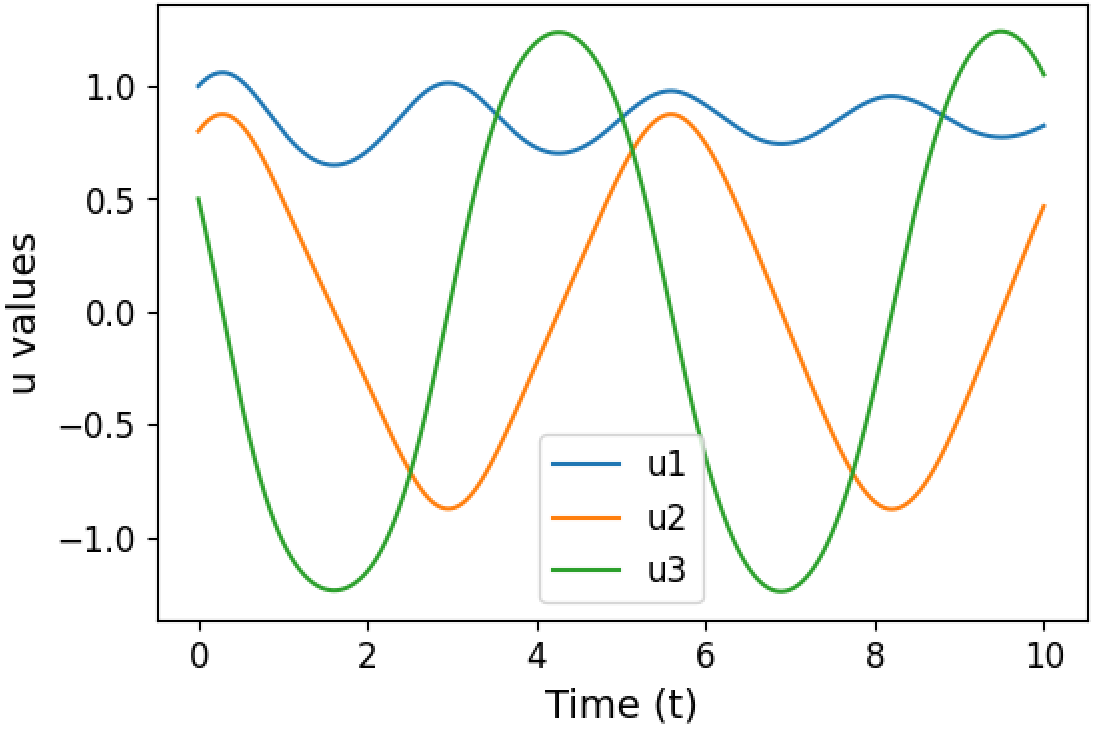
$$\frac{du_3}{dt} = -2 u_1 u_2$$



$u_1(0)=1$, $u_2(0)=0.8$, $u_3(0)=0.5$

https://medium.com/towards-data-science/discovering-differential-equations-with-physics-informed-neural-networks-and-symbolic-regression-c28d279c0b4d

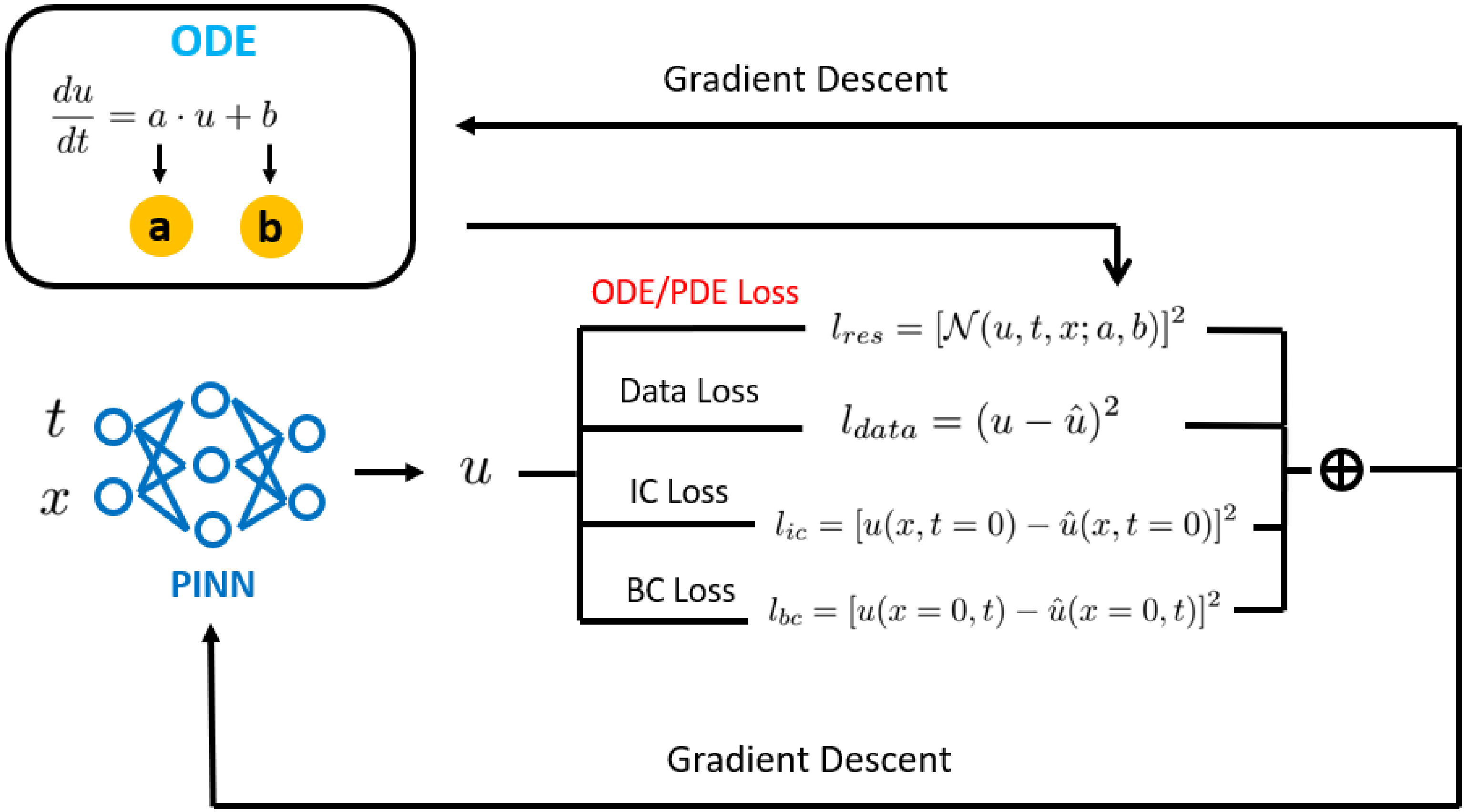
System Identification



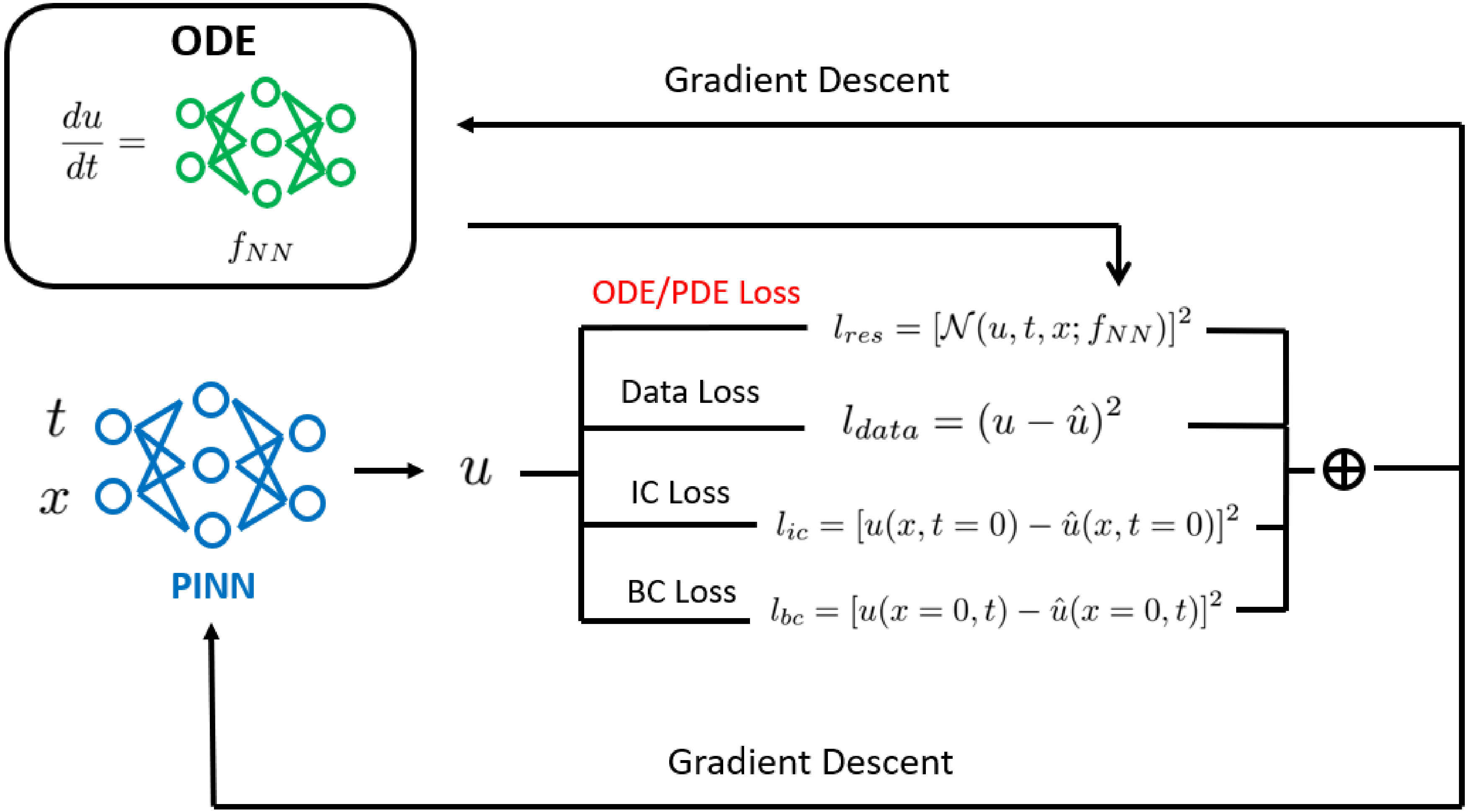
$$\frac{du_1}{dt} = f_1(t, u_1, u_2, u_3)$$
$$\frac{du_2}{dt} = f_2(t, u_1, u_2, u_3)$$
$$\frac{du_3}{dt} = au_1u_2 + b$$

Here, f_1 and f_2 denote the unknown functions, and a and b are the unknown parameters. **Our objective is to calibrate the values of a and b, as well as estimate the analytical functional form of f_1 and f_2.**

https://medium.com/towards-data-science/discovering-differential-equations-with-physics-informed-neural-networks-and-symbolic-regression-c28d279c0b4d

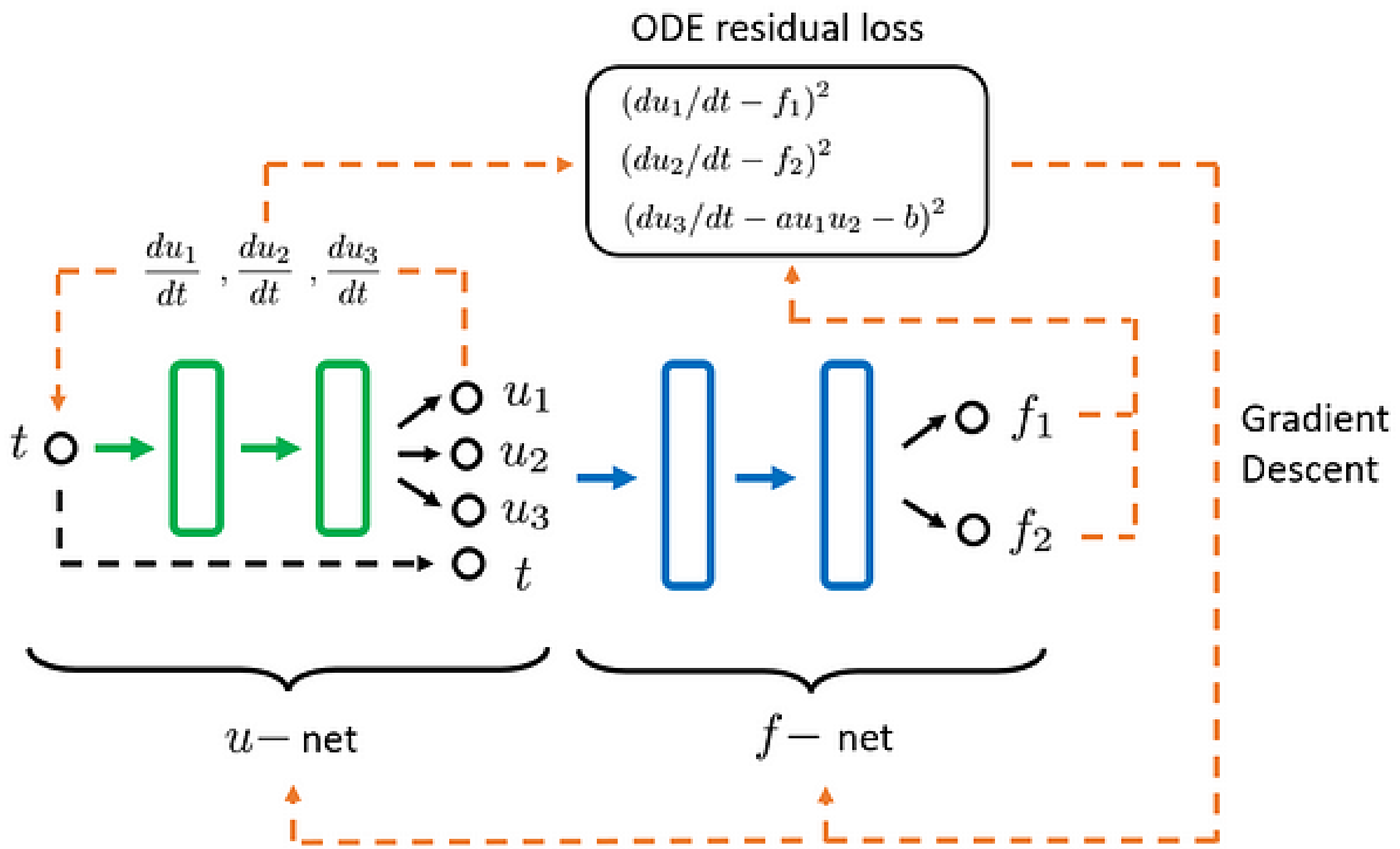


https://medium.com/towards-data-science/discovering-differential-equations-with-physics-informed-neural-networks-and-symbolic-regression-c28d279c0b4d

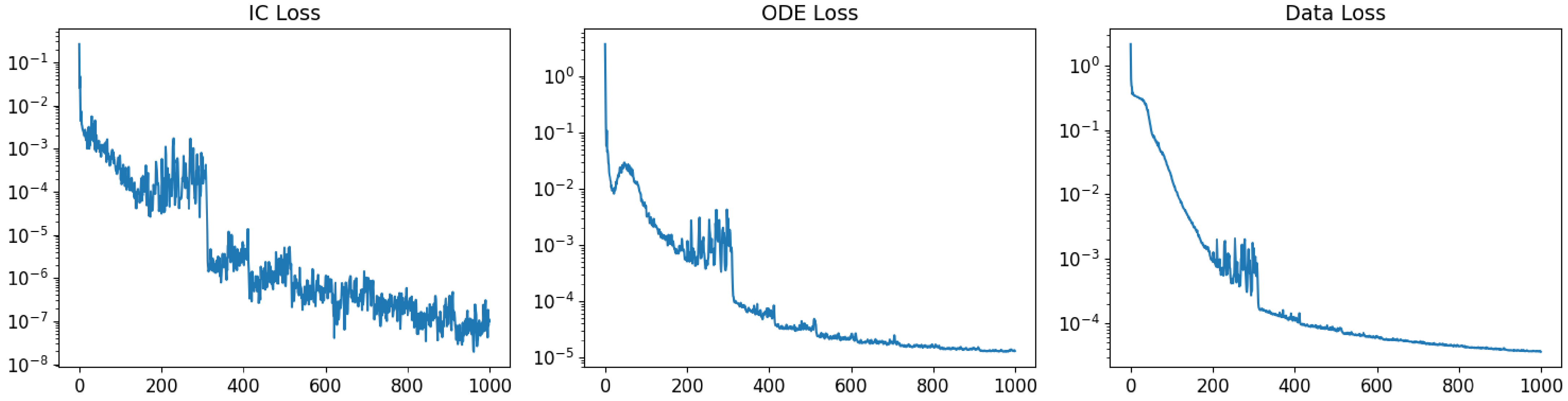


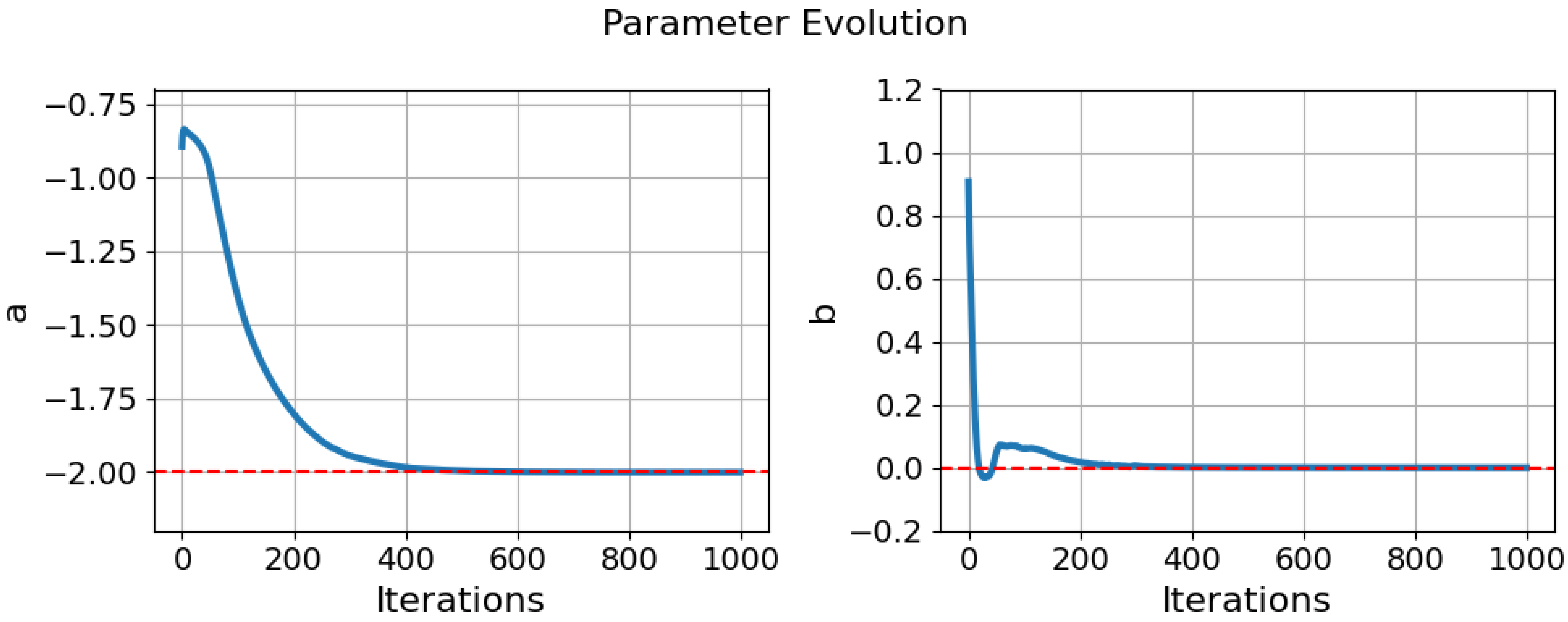
https://medium.com/towards-data-science/discovering-differential-equations-with-physics-informed-neural-networks-and-symbolic-regression-c28d279c0b4d

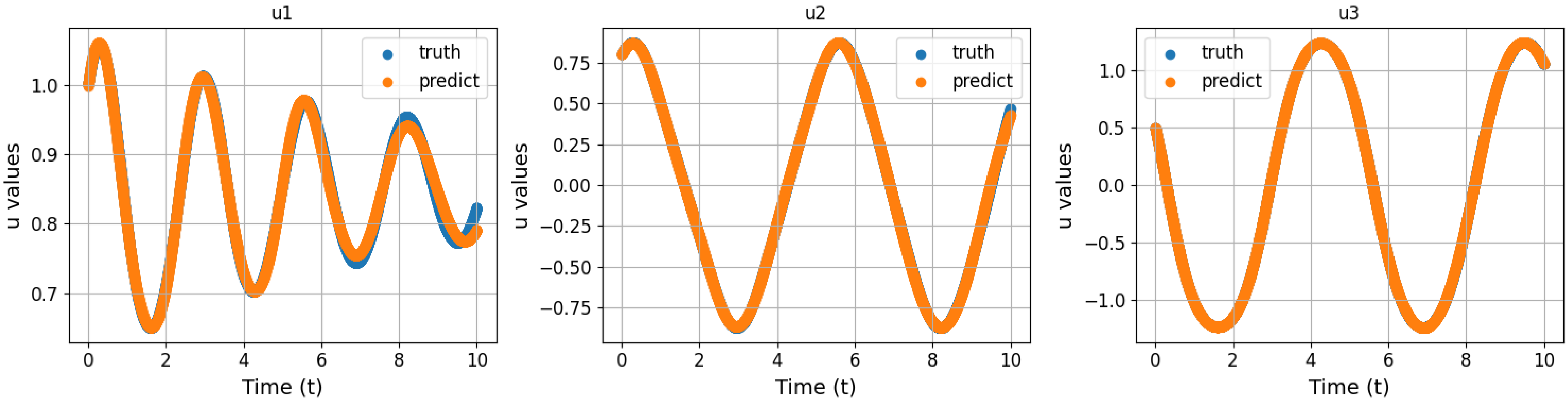
$$\frac{du_1}{dt} = f_1(t, u_1, u_2, u_3)$$
$$\frac{du_2}{dt} = f_2(t, u_1, u_2, u_3)$$
$$\frac{du_3}{dt} = au_1u_2 + b$$

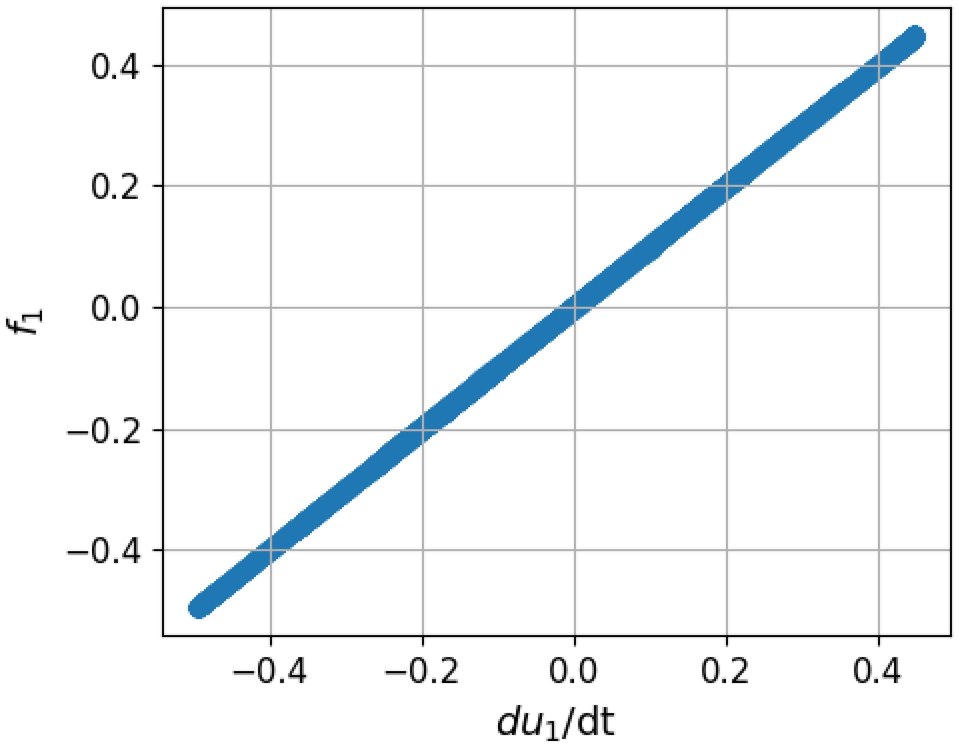


https://medium.com/towards-data-science/discovering-differential-equations-with-physics-informed-neural-networks-and-symbolic-regression-c28d279c0b4d

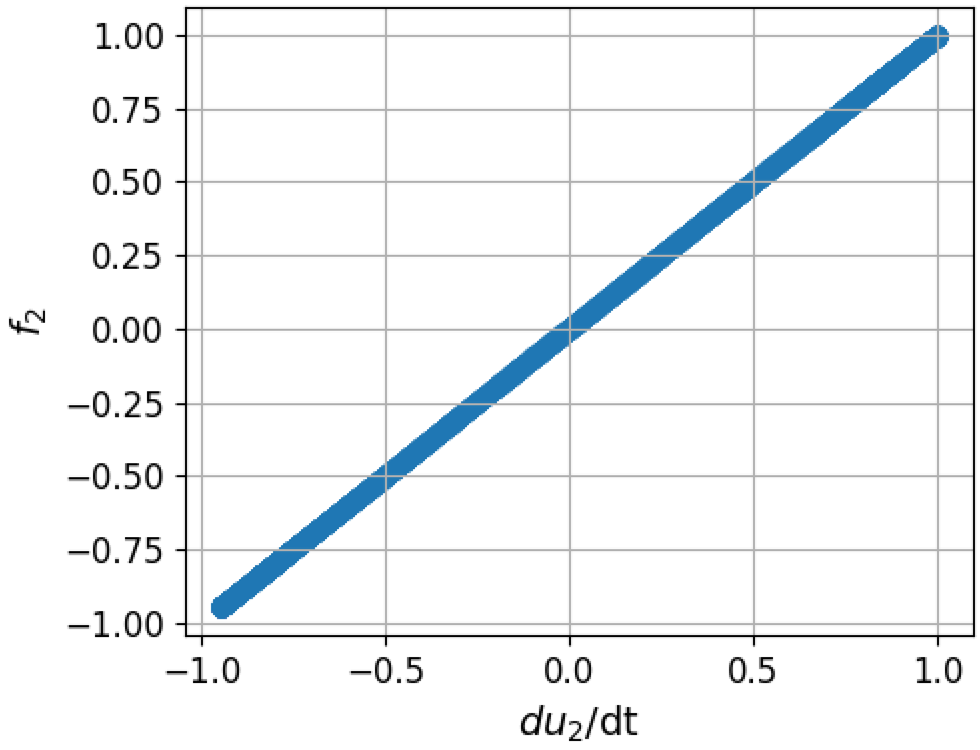

IC Loss
ODE Loss
Data Loss
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{0}
0
200
400
600
800
1000


Parameter Evolution
−0.75
−1.00
−1.25
−1.50
−1.75
−2.00
a
0
200
400
600
800
1000
Iterations
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
b
0
200
400
600
800
1000
Iterations


u1
u2
u3
truth
predict
u values
Time (t)
0 2 4 6 8 10
1.0
0.9
0.8
0.7
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
1.0
0.5
0.0
−0.5
−1.0

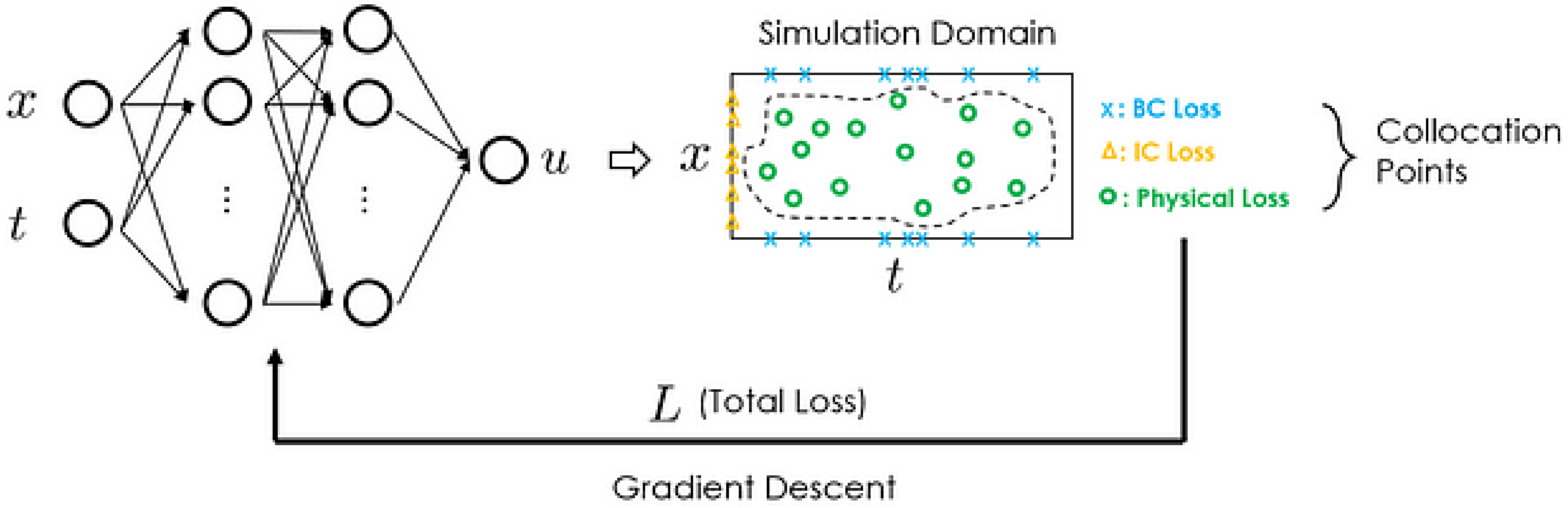


0.4
0.2
0.0
−0.2
−0.4
f_1
−0.4
−0.2
0.0
0.2
0.4
du_1/dt



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
f_2
−1.0
−0.5
0.0
0.5
1.0
du_2/dt

PINN Problems



https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

- Compared to the traditional numerical simulation approaches, PINNs are mesh-free. This enables PINNs to easily handle complex simulation domains, reduce manual effort in simulations, and potentially improve computational efficiency.
- Thanks to the universal approximation capability, PINNs are well-suited for modeling complex and nonlinear systems where traditional methods may struggle.
- Compared to the traditional paradigm of supervised learning, PINNs are in principle data-free, i.e., their training does not require collecting input-output pairs, i.e., (t, x, y) -(u, p, T), but entirely based on fulfilling the governing differential equations. However, if scarce data is available, PINNs can also effectively assimilate that.

https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

- Heat equation:

https://inductiva.ai/blog/article/heat-1-an-introduction
https://inductiva.ai/blog/article/heat-2-pinn
https://inductiva.ai/blog/article/heat-3-neurosolver

Helmholz equation

https://towardsdatascience.com/improving-pinns-through-adaptive-loss-balancing-55662759e701

https://towardsdatascience.com/10-useful-hints-and-tricks-for-improving-pinns-1a5dd7b86001

https://towardsdatascience.com/unraveling-the-design-pattern-of-physics-informed-neural-networks-series-01-8190df459527

https://towardsdatascience.com/building-an-expert-gpt-in-physics-informed-neural-networks-with-gpts-75ebf6925966