Projekt 2.2 – Hail cannon simulation using PINN

Mateusz Cyganek, Adrian Chrobot

# Zmodyfikowany kod

Stan początkowy nie był zmieniany.

Podobnie jak w wersji cpp, od obliczanej wartości odejmowane są pochodne funkcji działka – bx() przemnożone przez dfdx(), oraz by() przemnożone przez dfdy().

1. def residual\_loss(self, pinn: PINN):

2. x, y, t = get\_interior\_points(

3. self.x\_domain, self.y\_domain, self.t\_domain,

4. self.n\_points, pinn.device())

5. loss = dfdt(pinn, x, y, t).to(device)\

6. - self.dTy(y, t) \* dfdy(pinn, x, y, t).to(device)\

7. - self.Kx \* dfdx(pinn, x, y, t,order=2).to(device)\

8. - self.Ky \* dfdy(pinn, x, y, t, order=2).to(device)\

9. - self.source(y,t).to(device)\

10. - bx(pinn, x, y, t).to(device) \* dfdx(pinn, x, y, t).to(device)\

11. - by(pinn, x, y, t).to(device) \* dfdy(pinn, x, y, t).to(device)

12.

13. return loss.pow(2).mean()

14.

Dodane linijki zaznaczono kolorem zielonym.

# Kod pochodnych działka

Funkcje bx() oraz by(), powstały analogicznie do funkcji f(), którą zastąpiono funkcją cannon().

Funkcje wywołując funkcje cannon(), a następnie obliczają pochodną względem y lub x.

Podobnie jak w wersji cpp, stałe mocy działka pozwalając zwiększać lub zmiejszać wartości zwracanego tensora.

1. def bx(pinn: PINN, x: torch.Tensor, y: torch.Tensor, t: torch.Tensor, order: int = 1):

2. f\_value = cannon(pinn, x, y, t)

3. return df(f\_value, y, order=order) \* cannon\_strength\_x

4.

5. def by(pinn: PINN, x: torch.Tensor, y: torch.Tensor, t: torch.Tensor, order: int = 1):

6. f\_value = cannon(pinn, x, y, t)

7. return df(f\_value, y, order=order) \* cannon\_strength\_y

8.

# Kod działka

* Kod funkcji działka zwraca tensor dla podanych tensorów x, y i t.
* Zmienna zero\_tensor, to tensor o wartościach zero,
* Zmienna valid\_i\_denom Tworzy tensor wartości logicznych, sprawdzając, czy i\_denom jest większe od 0.
* W linijce 9 ustawiana jest wartość time na 0.0 w miejscach, gdzie jest valid\_i\_denom fałszywe.
* W linijce 11 valid\_y = y <= time tworzy tensor wartości logicznych, sprawdzając, czy y jest mniejsze lub równe time.

Obliczamy w którym momencie wystrzału jesteśmy i możemy przez rozmiar siatki by traktować time jako współrzędną osi Y. Jeżeli y > time znaczy to tyle, że y jest nad falą wystrzału, zatem powinno zwrócić tam wartość 0.

0 zostanie zwrócone w przypadku gdy y jest nad y’, lub gdy kąt α rozchodzenia się fali jest więszky od zdefiniowanego maksymalnego kąta.

Jeżeli wszystkie te warunki są spełnione to ostatecznie zwracany tensor to różnica między y a y’, pomnożona przez cosinus α kąta.

1. def cannon(pinn: PINN, x: torch.Tensor, y: torch.Tensor, t: torch.Tensor) -> torch.Tensor:

2. iter = t

3. zero\_tensor = torch.zeros\_like(iter)

4.

5. i\_denom = torch.tensor((iterations / wave\_speed) - cannon\_shot\_time, device=iter.device)

6. valid\_i\_denom = i\_denom > 0

7.

8. time = (iter - cannon\_shot\_time) \* grid\_size / i\_denom

9. time = torch.where(valid\_i\_denom, time, zero\_tensor)

10.

11. valid\_y = y <= time

12.

13. x\_prim = torch.abs(cannon\_x\_loc - x)

14. alpha\_rad = torch.atan(x\_prim / time)

15. valid\_alpha = alpha\_rad < max\_alpha

16.

17. y\_prim = torch.sqrt(time \* time - x\_prim \* x\_prim)

18. valid\_y\_prim = y <= y\_prim

19.

20. cos\_term = torch.cos(alpha\_rad \* cone\_limiter \* 0.5)

21.

22. cannon\_value = (y\_prim - y) \* cos\_term

23. valid\_condition = valid\_i\_denom & valid\_y & valid\_alpha & valid\_y\_prim

24. cannon\_value = torch.where(valid\_condition, cannon\_value, zero\_tensor)

25.

26. return cannon\_value

27.

# Stałe działka

Stałę działka wykorzystywane przez funkcje działka.

1. grid\_size = 1.0

2. cannon\_x\_loc = grid\_size / 2

3. cannon\_shot\_time = 0.4

4. cannon\_strength\_x = 3.5

5. cannon\_strength\_y = 4.5

6. cone\_limiter = 6.0

7. max\_alpha = np.pi / cone\_limiter

8. wave\_speed = 2.0

9. iterations = 1.0

10.

* grid\_size – rozmiar siatki, w której przeprowadzana jest symulacja,
* cannon\_x\_loc – lokalizacja działa na osi X, w symulacji działo znajduje się na środku siatki,
* cannon\_shot\_time – klatka, w której działo strzela,
* cannon\_strength\_x – mnożnik siły działa w osi X,
* cannon\_strength\_y – mnożnik siły działa w osi Y,
* cone\_limiter – zmienna ograniczenia maksymalnego kąta α rozchodzenia się wystrzelonej fali, równego π / cone\_limiter,
* wave\_speed – szybkość rozchodzenia się fali.

# Uzyskany efekt

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Wykresy zbieżności procesu uczenia

A graph of a loss function

Description automatically generated

A graph of loss function

Description automatically generated

A graph of loss function

Description automatically generated