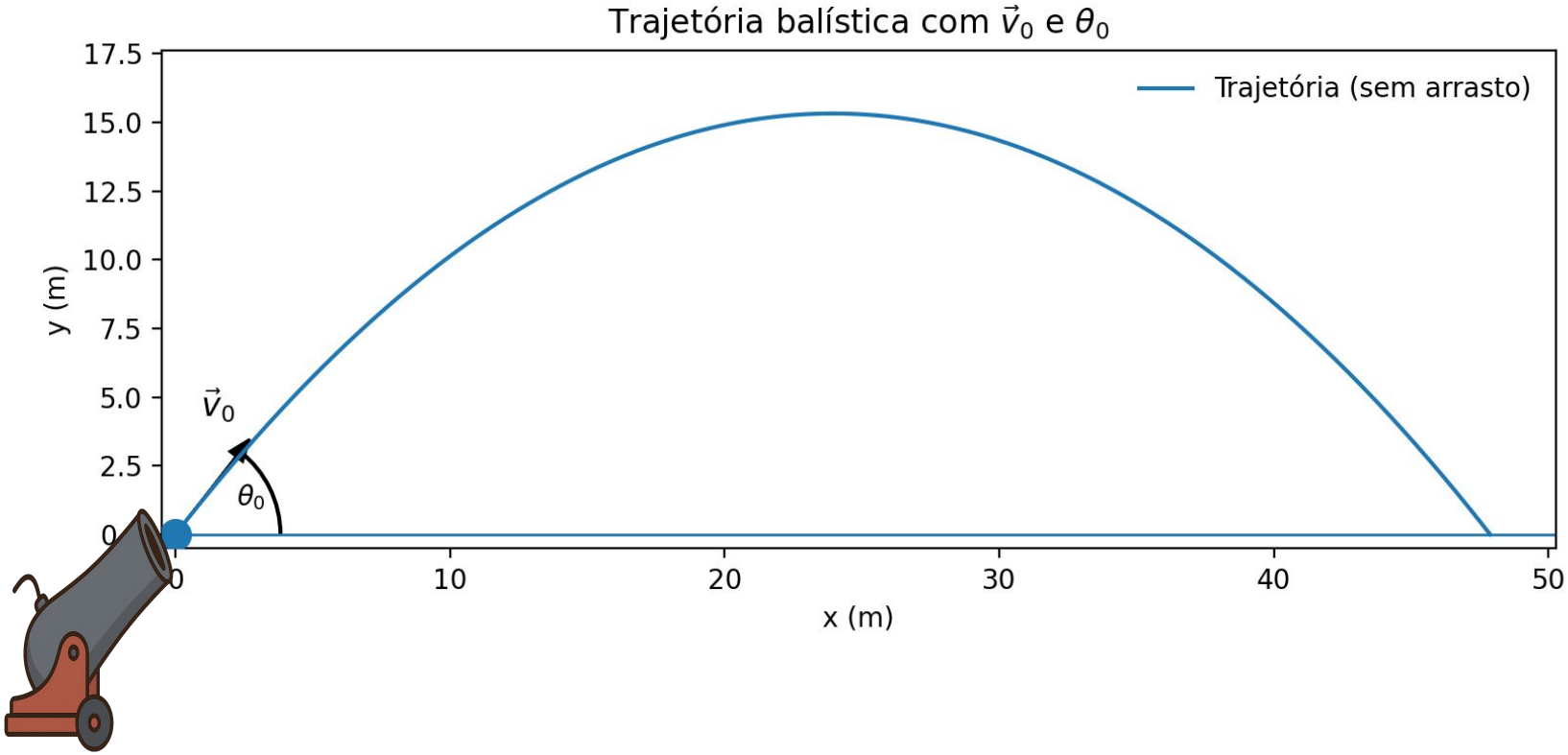

Disparando canhões...
...usando PINN


Trajetória balística com $\vec{v}_0$ e θ_0
Trajetória (sem arrasto)
$\vec{v}_0$
θ_0
y (m)
x (m)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
0
10
20
30
40
50

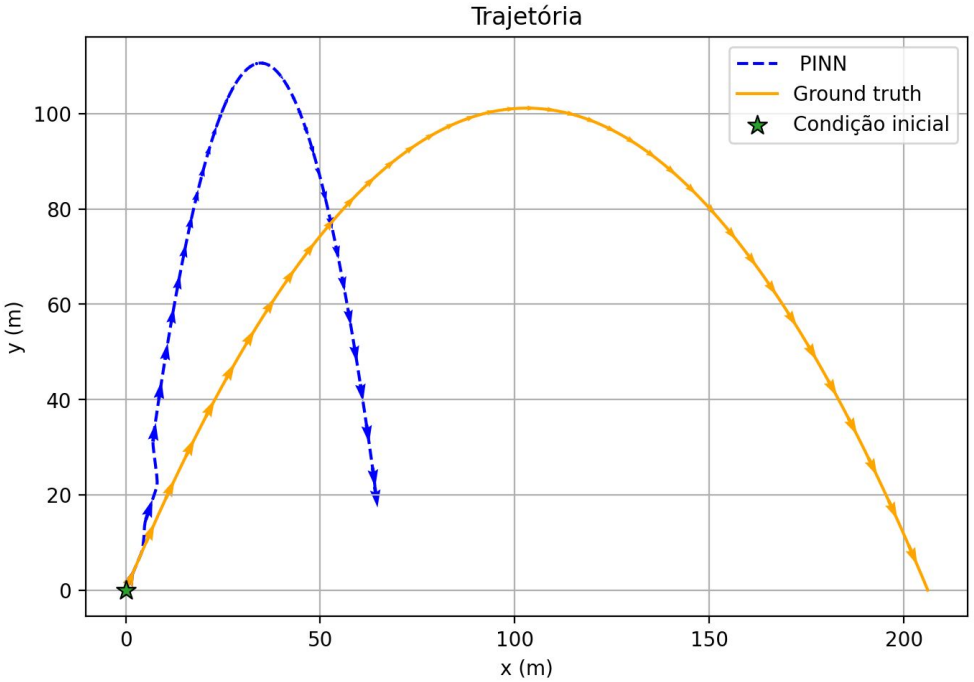
Física do disparo balístico... ...com arrasto! (mas sem vento)

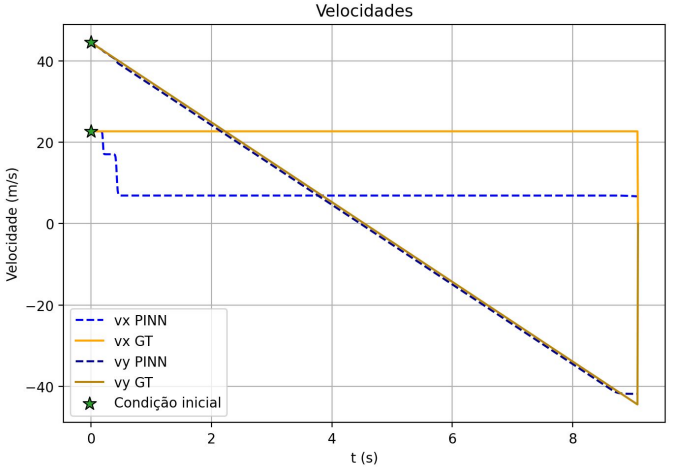
$$\dot{x} = v_x$$
$$\dot{y} = v_y$$
$$\dot{v}_x = -k\,|\vec{v}|\,v_x$$
$$\dot{v}_y = -g - k\,|\vec{v}|\,v_y$$

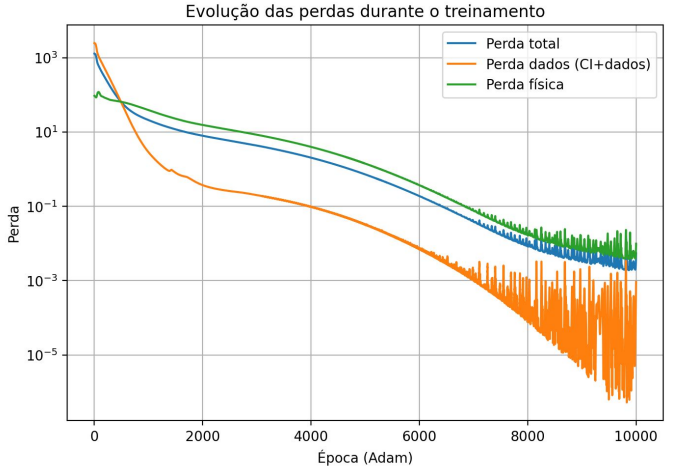
onde $k = \dfrac{\rho C_d A}{2m}$

e $|\vec{v}| = \sqrt{v_x^2 + v_y^2}$

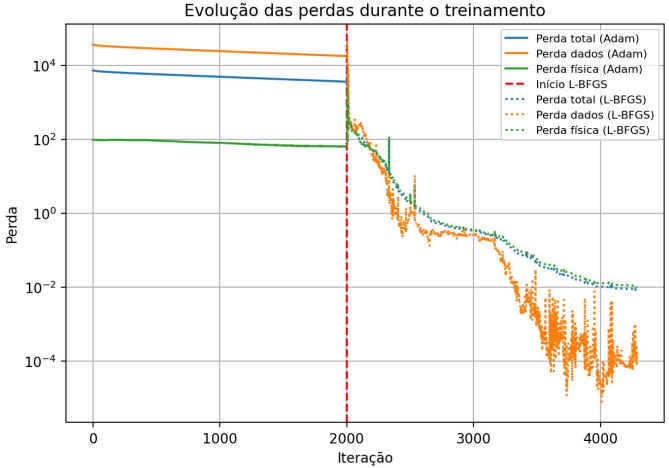
Rede[1-64-64-64-4], Adam_steps=**10000**, N_phys=500

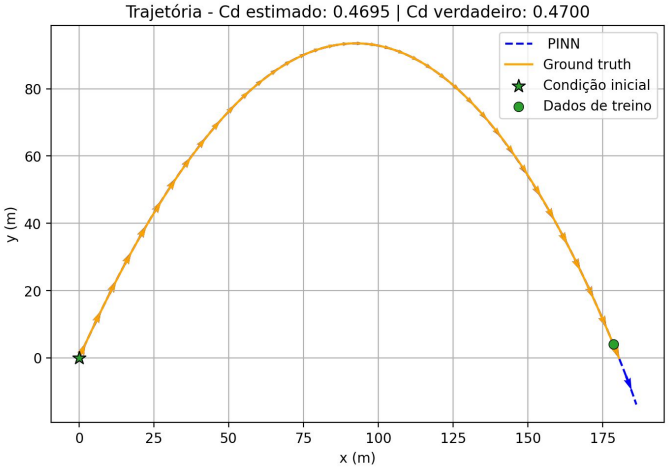


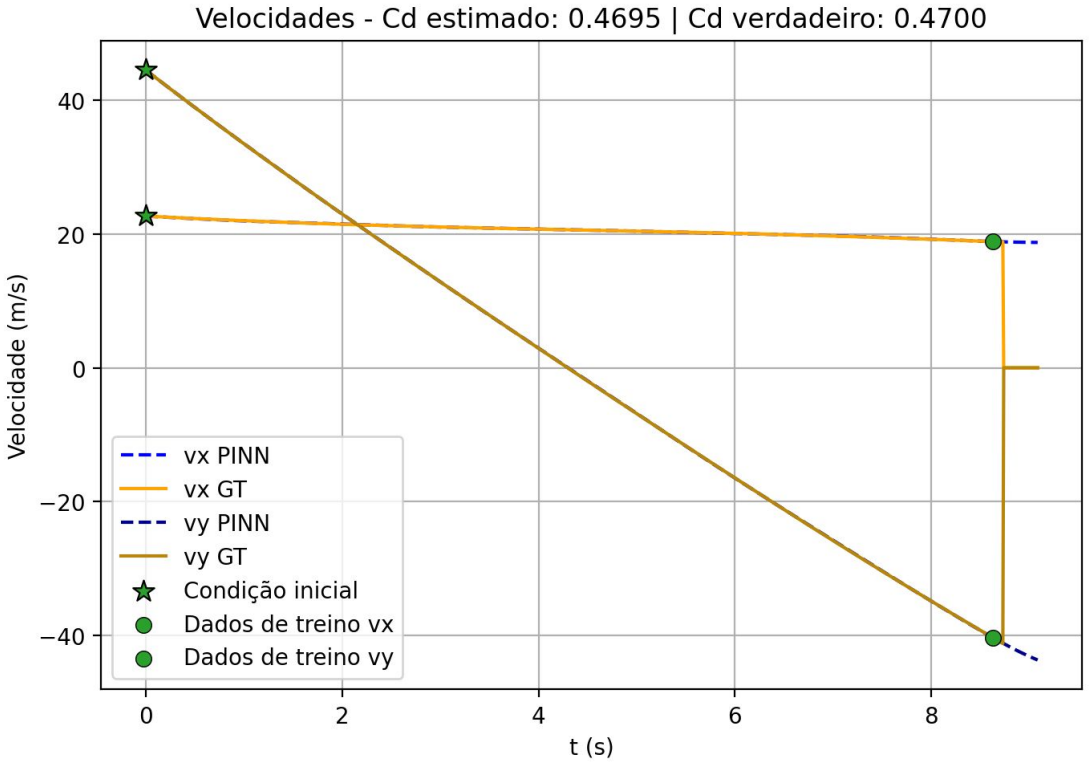




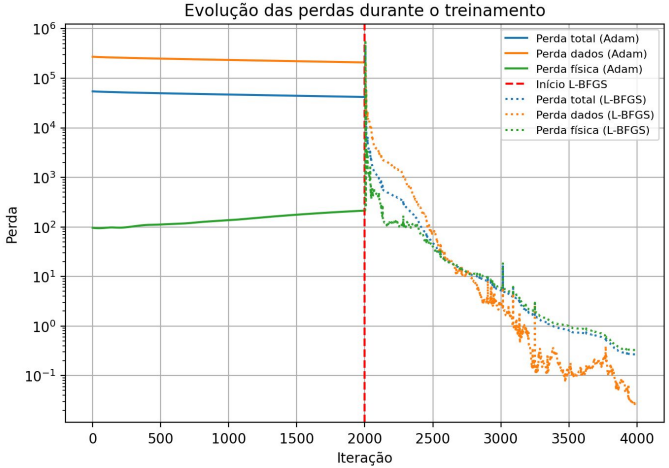
Adicionando apenas 1 dado extra

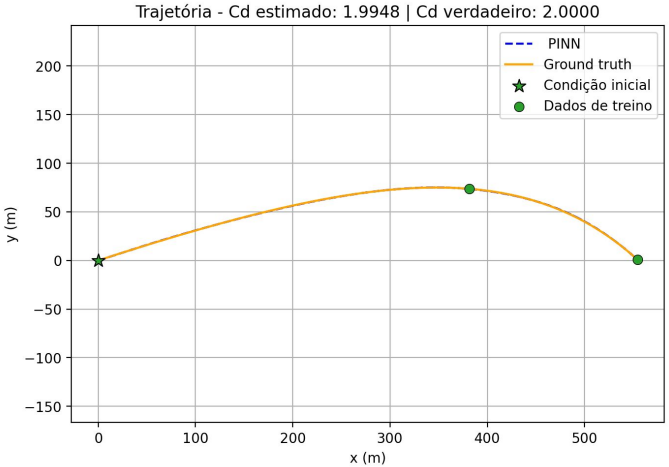


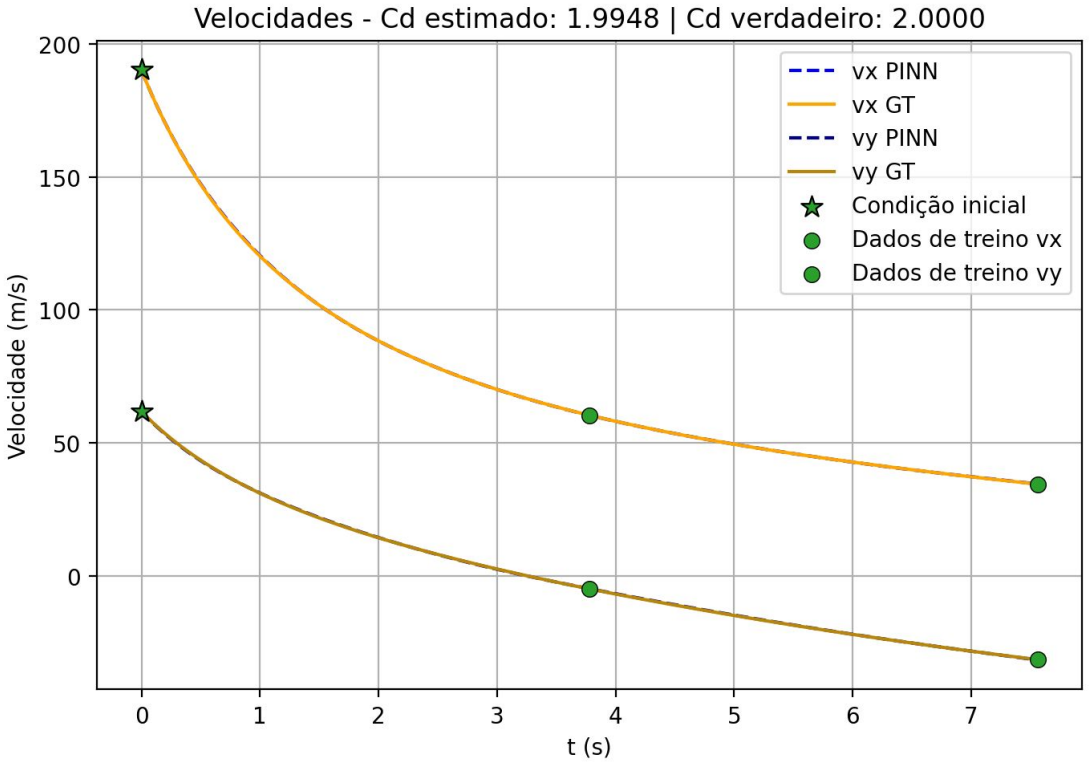




Mudando as condições iniciais - Tiro mais rasante, com alto arrasto





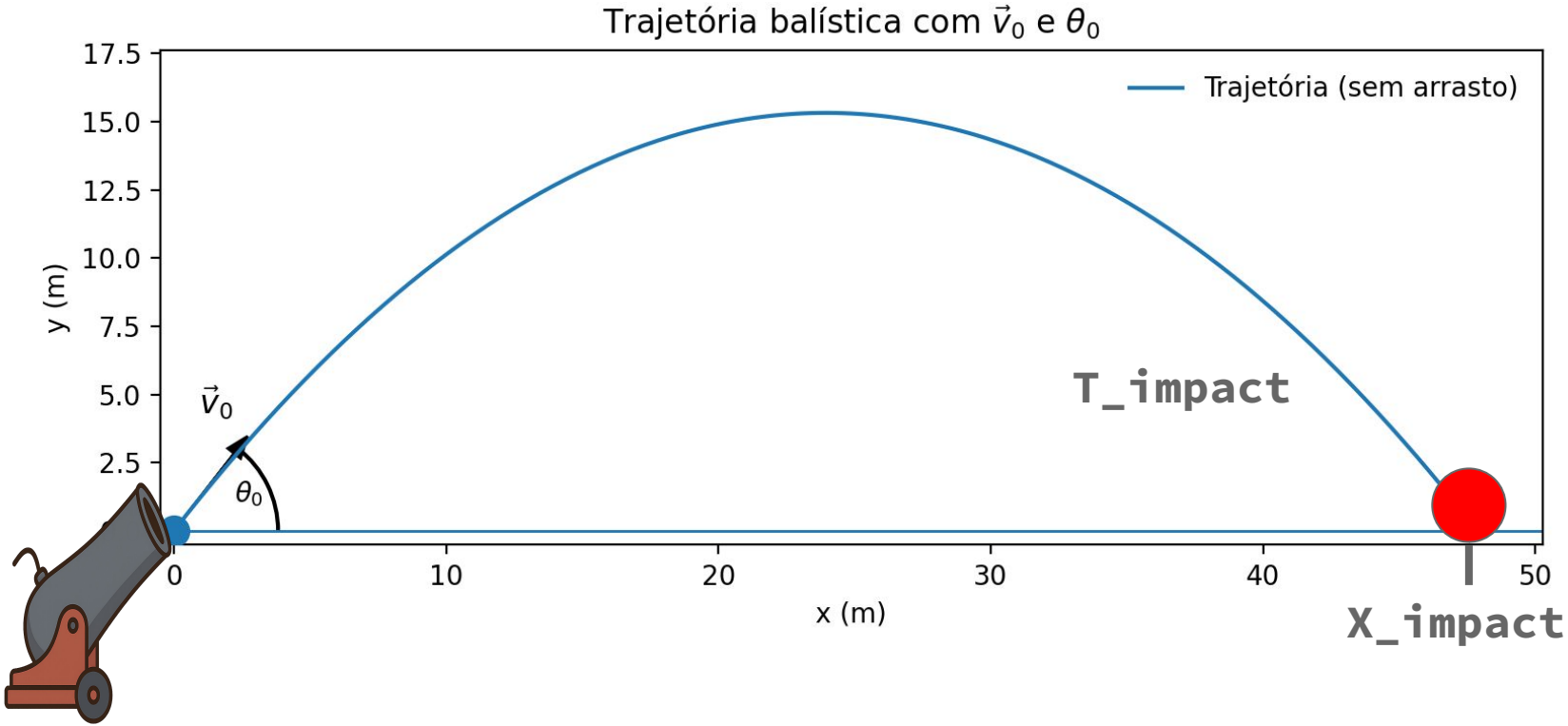


Próximos passos??

Usar PINN para estimar o alvo, não a trajetória

REDE NEURAL QUE:

Entrada	**(V0 ; θ0)**
Saída	**(X_impact ; T_impact)**


Trajetória balística com $\vec{v}_0$ e θ_0
Trajetória (sem arrasto)
y (m)
x (m)
$\vec{v}_0$
θ_0
T_impact
X_impact
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
10
20
30
40
50

Física - Equações Integrais - Vácuo

$$x(T) = \int_0^T v_x(t)\,dt = \int_0^T v_0 \cos\theta\,dt$$

$$y(T) = \int_0^T v_y(t)\,dt = \int_0^T \left(v_0 \sin\theta - gt\right) dt$$

Física - Contornos

v_y(T_{\text{impact}}) < 0 \quad\text{(evita solução trivial } T=0

$$y(T_{\text{impact}}) = 0$$

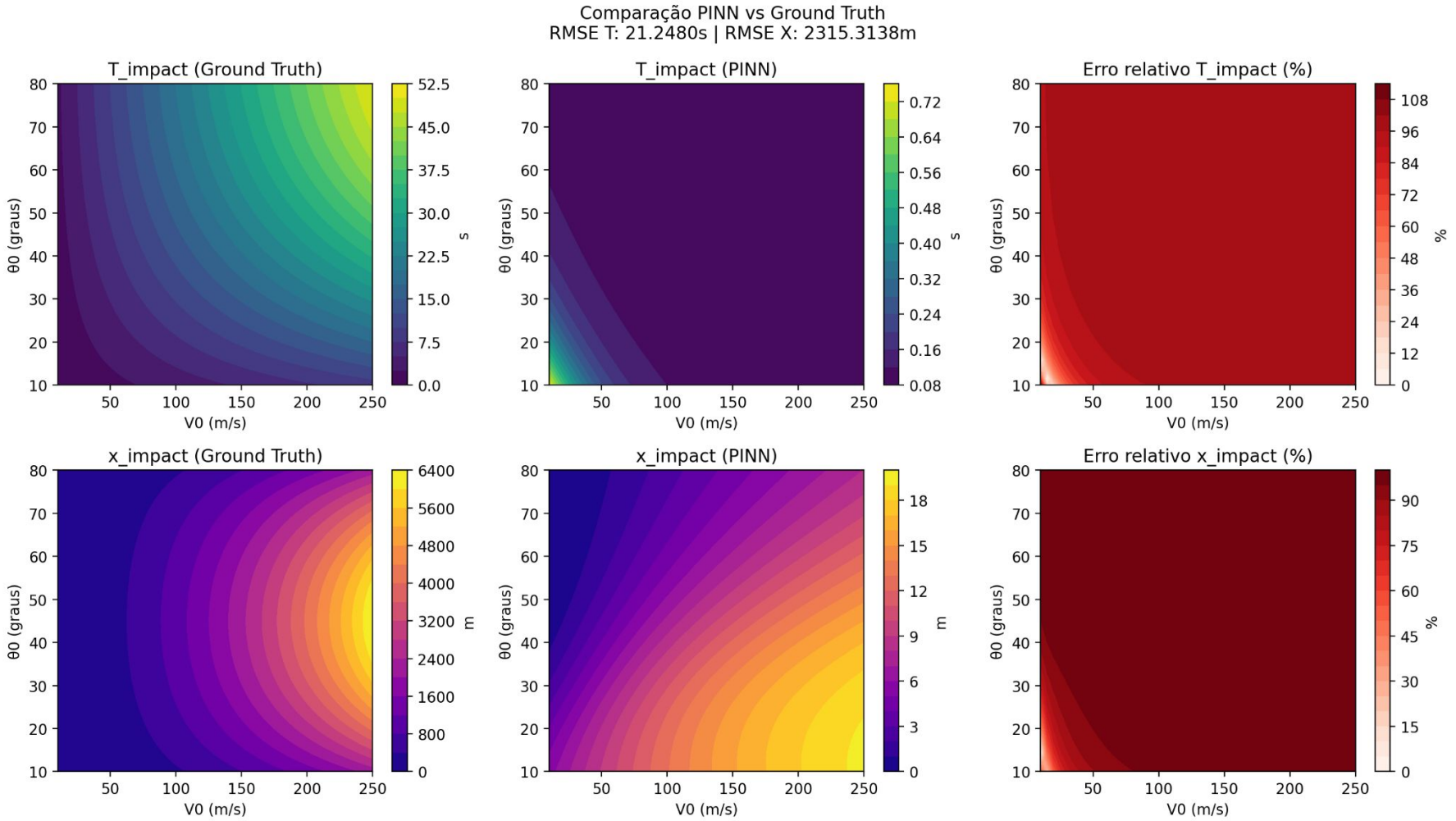
$$x(T_{\text{impact}}) = x_{\text{impact}}$$

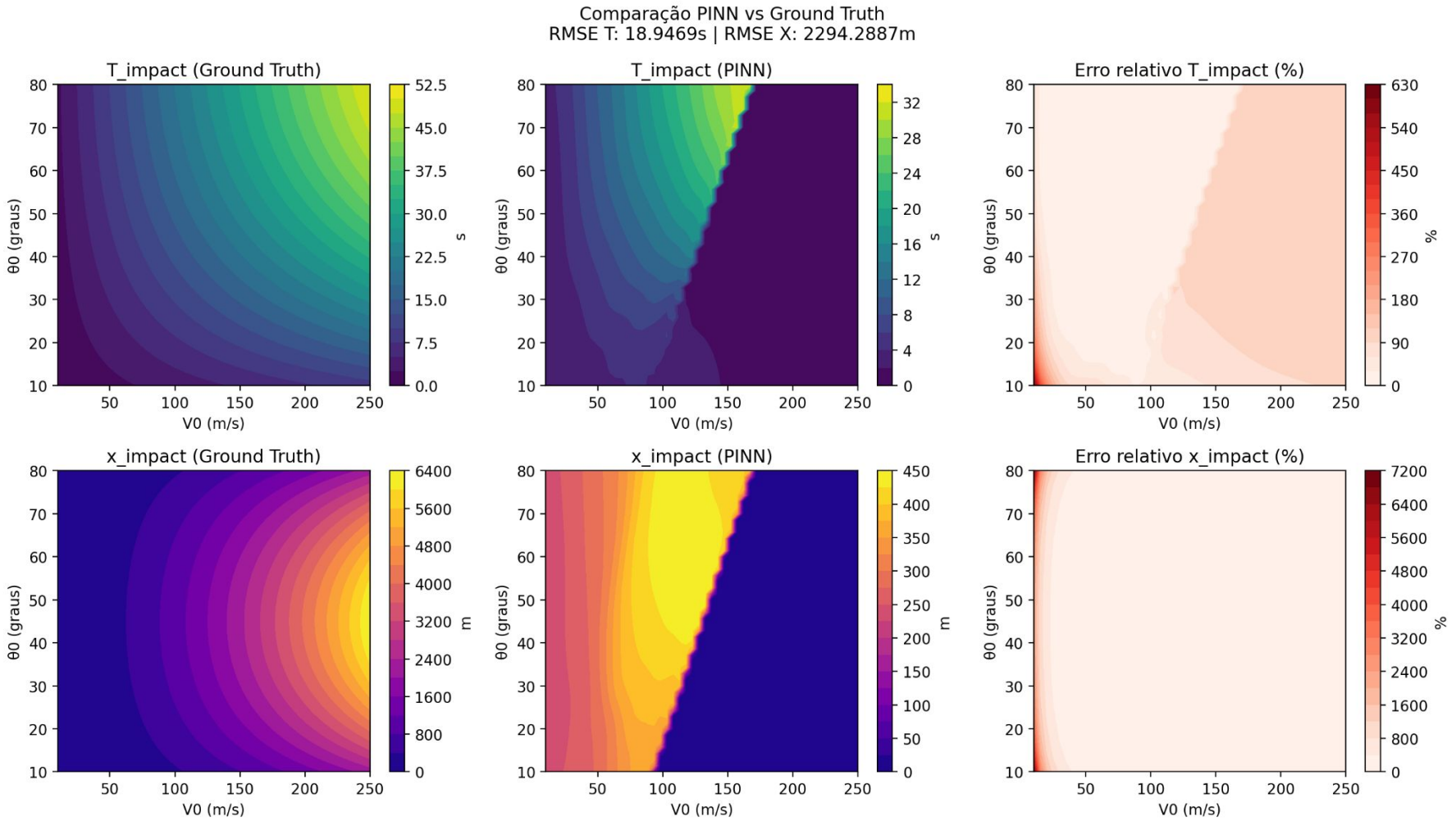
$$v_y(T_{\text{impact}}) < 0$$

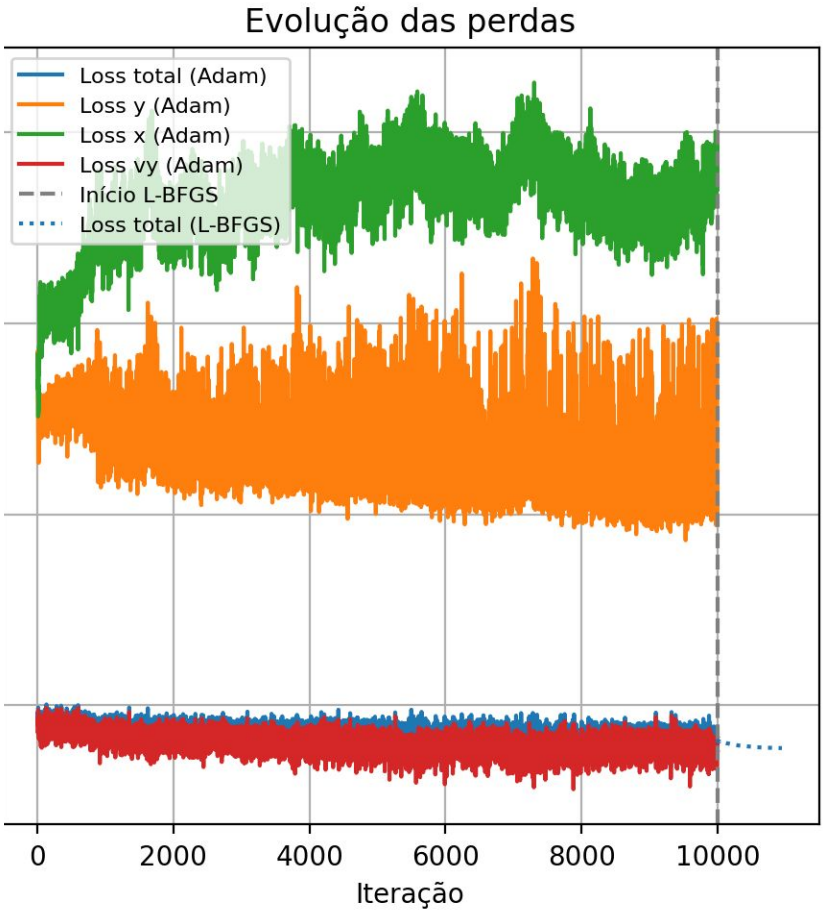
```python
raw = self.net(inputs)

# T_impact deve ser positivo (usamos softplus para garantir)
T_impact = torch.nn.functional.softplus(raw[:, 0:1]) + 0.1  # mínimo de 0.1s

# x_impact deve ser positivo
x_impact = torch.nn.functional.softplus(raw[:, 1:2])
```

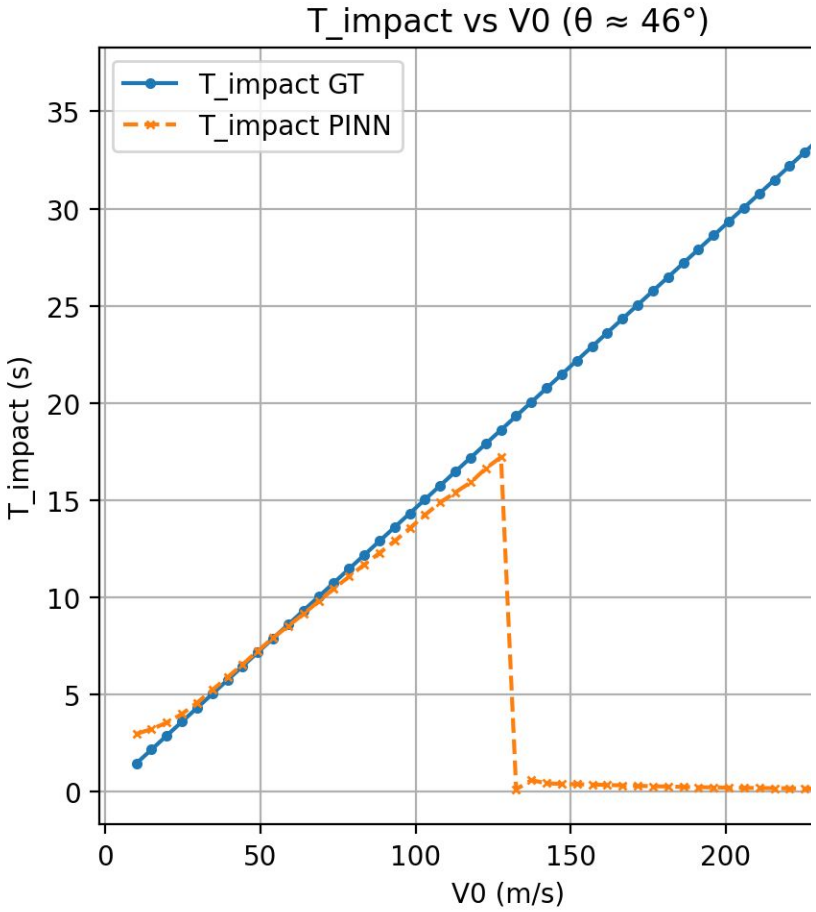
```python
vx = vx0.unsqueeze(1).expand(-1, self.n_integration)  # (batch, n_integration)
vy = vy0.unsqueeze(1) - self.g * t  # (batch, n_integration)

# Integração por trapézio
# x = ∫ vx dt
x_integrated = torch.trapezoid(vx, t, dim=1)  # (batch,)

# y = ∫ vy dt
y_integrated = torch.trapezoid(vy, t, dim=1)  # (batch,)

# vy no momento do impacto
vy_impact = vy0 - self.g * T_impact.squeeze(1)  # (batch,)
```

```python
# Loss 1: y(T_impact) = 0
loss_y = (y_integrated ** 2).mean()

# Loss 2: x integrado = x_impact predito
loss_x = ((x_integrated - x_impact.squeeze(1)) ** 2).mean()

# Loss 3: vy_impact < 0 (penaliza se vy >= 0)
# Usamos ReLU: max(0, vy_impact + epsilon_vy) para margem de segurança
loss_vy = torch.relu(vy_impact + self.epsilon_vy).mean()

# Perda total com pesos configuráveis
loss = self.lambda_y * loss_y + self.lambda_x * loss_x + self.lambda_vy * loss_vy
```


Comparação PINN vs Ground Truth
RMSE T: 21.2480s | RMSE X: 2315.3138m
T_impact (Ground Truth)
T_impact (PINN)
Erro relativo T_impact (%)
x_impact (Ground Truth)
x_impact (PINN)
Erro relativo x_impact (%)
θ0 (graus)
V0 (m/s)
s
m
%


Comparação PINN vs Ground Truth
RMSE T: 18.9469s | RMSE X: 2294.2887m
T_impact (Ground Truth)
T_impact (PINN)
Erro relativo T_impact (%)
x_impact (Ground Truth)
x_impact (PINN)
Erro relativo x_impact (%)
θ0 (graus)
V0 (m/s)
s
m
%

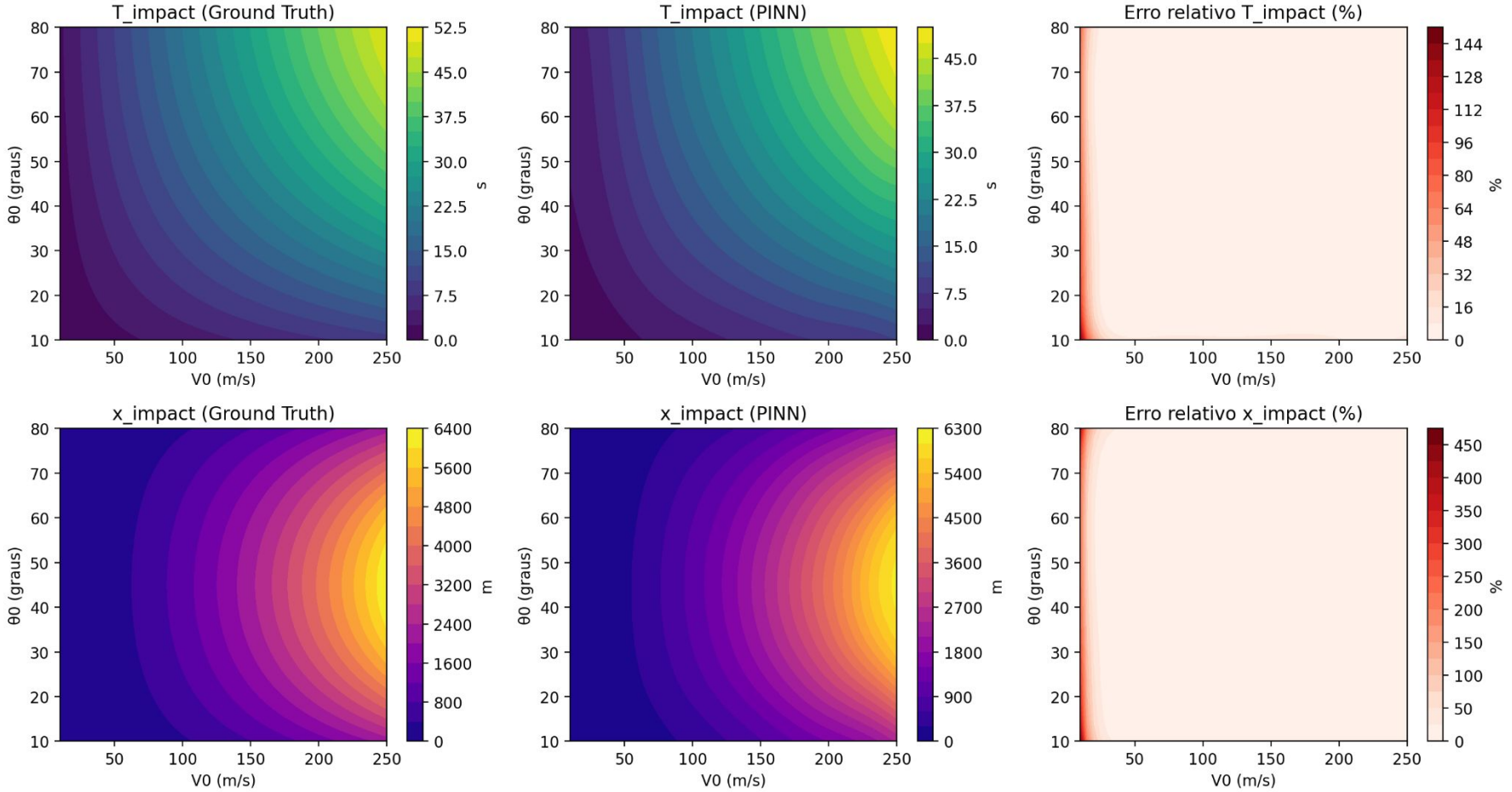

Evolução das perdas
Loss total (Adam)
Loss y (Adam)
Loss x (Adam)
Loss vy (Adam)
Início L-BFGS
Loss total (L-BFGS)
0
2000
4000
6000
8000
10000
Iteração

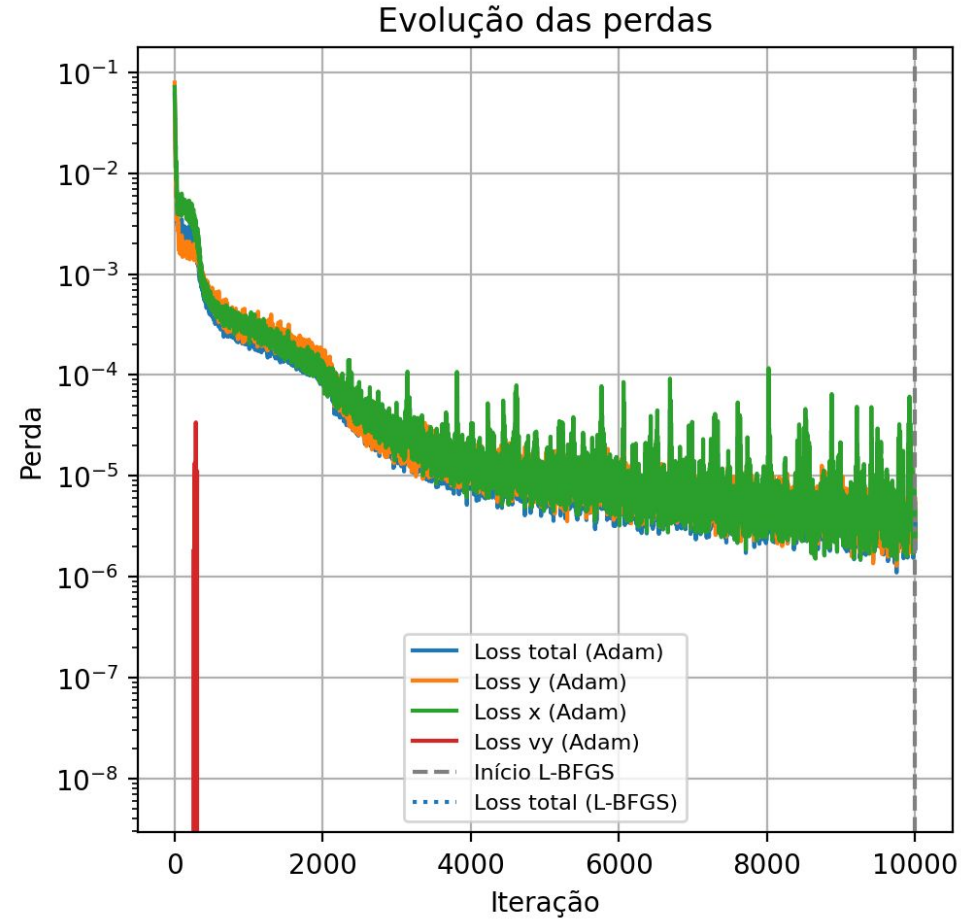

T_impact vs V0 (θ ≈ 46°)
T_impact GT
T_impact PINN
T_impact (s)
0
5
10
15
20
25
30
35
0
50
100
150
200
V0 (m/s)

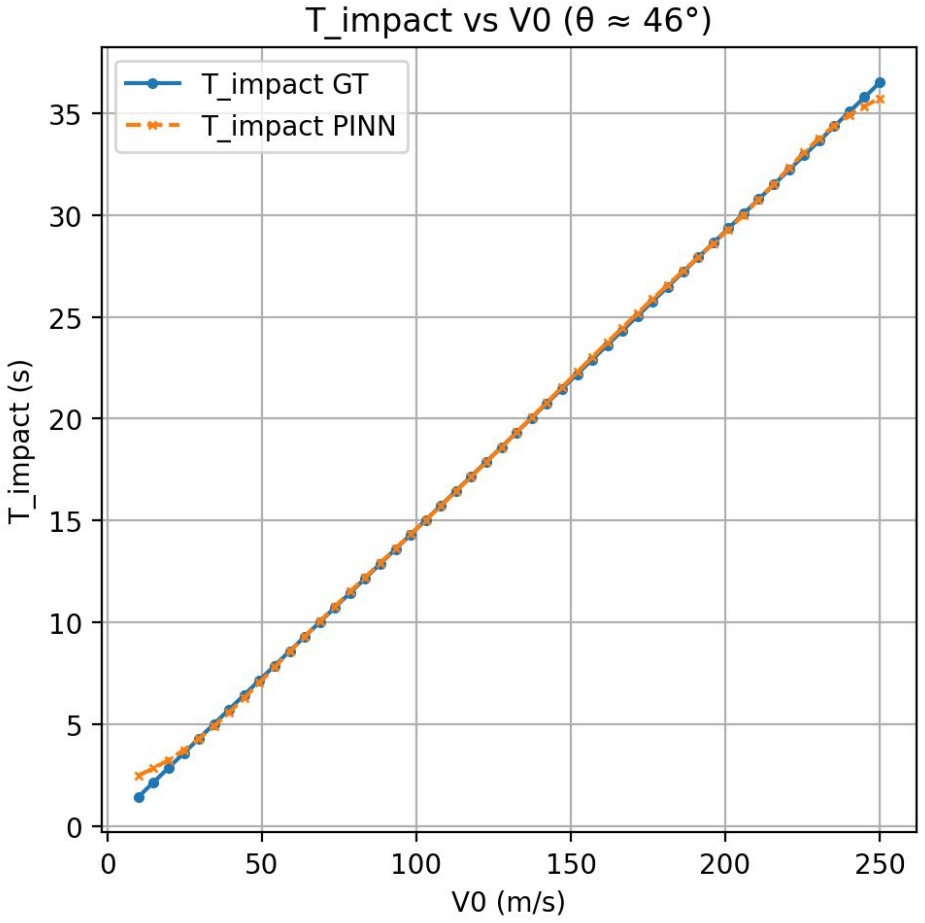
```python
# Escalas esperadas para outputs (baseadas na física)
# T_max ≈ 2 * V0_max * sin(theta0_max) / g
T_scale = 2 * V0_max * np.sin(np.radians(theta0_max)) / g  # ≈ 50s
# x_max ≈ V0_max² / g (para theta=45°)
X_scale = V0_max**2 / g  # ≈ 6370m
```

```python
# T_impact deve ser positivo (usamos softplus para garantir)
T_impact = torch.nn.functional.softplus(raw[:, 0:1]) + 0.1  # mínimo de 0.1s
# T_impact: sigmoid * T_scale (range: [0, T_scale])
# Adicionamos pequeno offset para evitar T=0
T_impact = torch.sigmoid(raw[:, 0:1]) * self.T_scale + 0.1
```

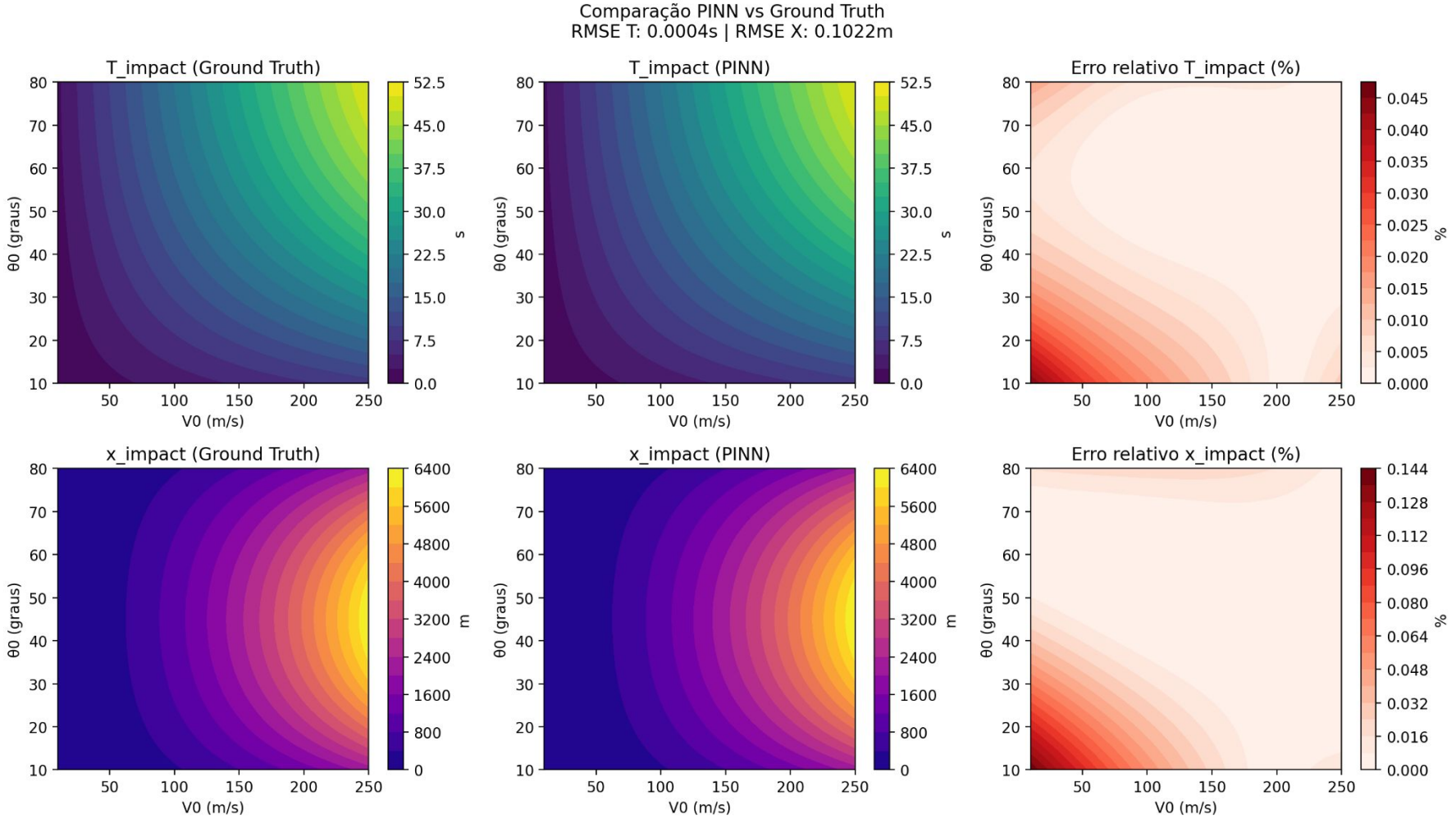
```python
# x_impact deve ser positivo
x_impact = torch.nn.functional.softplus(raw[:, 1:2])
# x_impact: sigmoid * X_scale (range: [0, X_scale])
x_impact = torch.sigmoid(raw[:, 1:2]) * self.X_scale
```

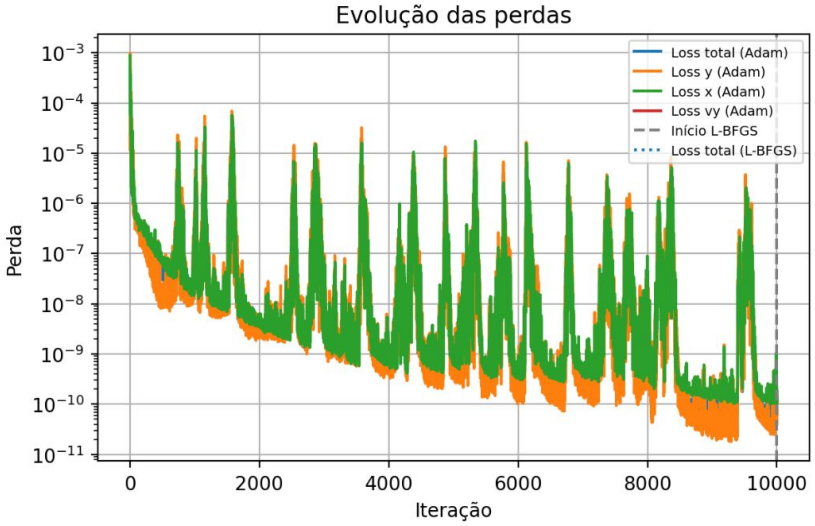

Comparação PINN vs Ground Truth
RMSE T: 0.2261s | RMSE X: 29.1307m
T_impact (Ground Truth)
T_impact (PINN)
Erro relativo T_impact (%)
x_impact (Ground Truth)
x_impact (PINN)
Erro relativo x_impact (%)
θ0 (graus)
V0 (m/s)
s
m
%

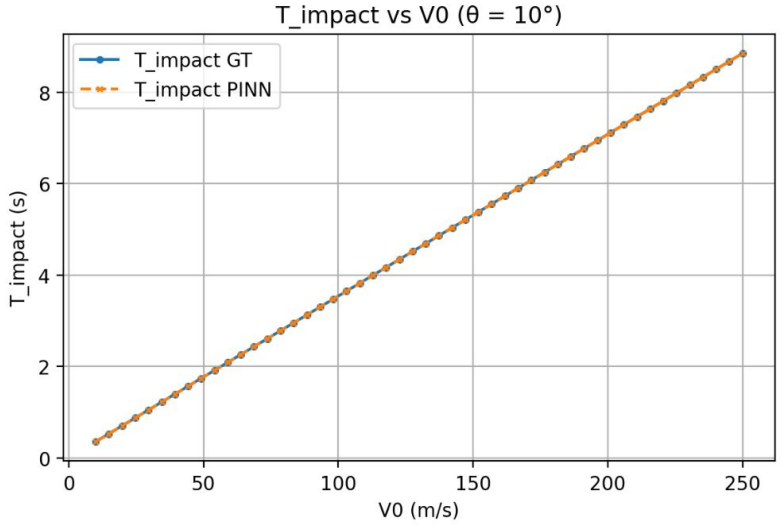

Evolução das perdas
Perda
Iteração
Loss total (Adam)
Loss y (Adam)
Loss x (Adam)
Loss vy (Adam)
Início L-BFGS
Loss total (L-BFGS)

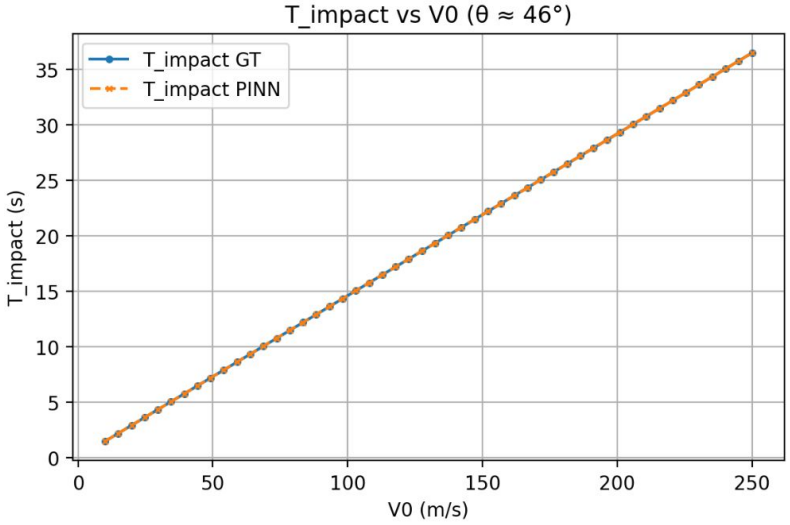

T_impact vs V0 (θ ≈ 46°)
T_impact (s)
V0 (m/s)
T_impact GT
T_impact PINN

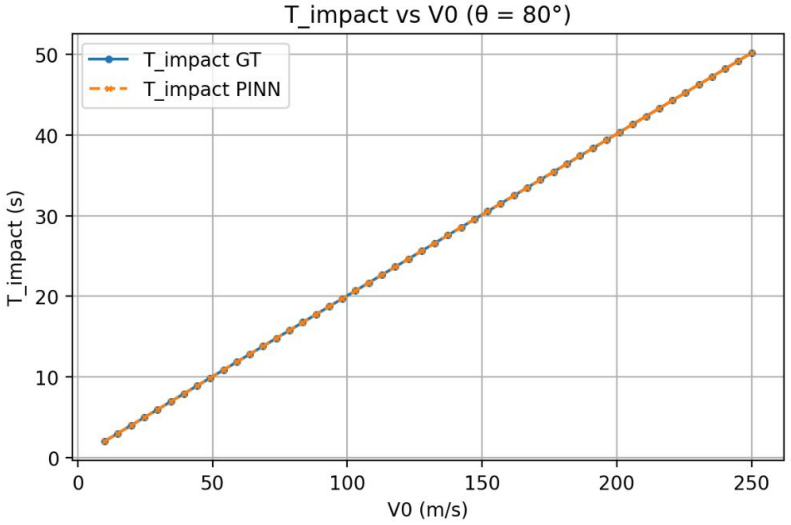
```python
# Escala esperada baseada na física (vácuo)
# T_expected = 2 * v0 * sin(theta0) / g
# X_expected = v0² * sin(2*theta0) / g
T_expected = 2 * v0 * torch.sin(theta0) / self.g
X_expected = v0**2 * torch.sin(2 * theta0) / self.g

# Rede prediz fator de ajuste: sigmoid dá [0, 1], multiplicamos por 2 para [0, 2x esperado]
# Isso garante que a sigmoid sempre opera numa faixa útil
T_impact = torch.sigmoid(raw[:, 0:1]) * 2 * T_expected
x_impact = torch.sigmoid(raw[:, 1:2]) * 2 * X_expected
```


Comparação PINN vs Ground Truth
RMSE T: 0.0004s | RMSE X: 0.1022m
T_impact (Ground Truth)
T_impact (PINN)
Erro relativo T_impact (%)
x_impact (Ground Truth)
x_impact (PINN)
Erro relativo x_impact (%)
θ0 (graus)
V0 (m/s)
s
m
%


Evolução das perdas
Loss total (Adam)
Loss y (Adam)
Loss x (Adam)
Loss vy (Adam)
Início L-BFGS
Loss total (L-BFGS)
Perda
Iteração


T_impact vs V0 (θ = 10°)
T_impact GT
T_impact PINN
T_impact (s)
V0 (m/s)


T_impact vs V0 (θ ≈ 46°)
T_impact GT
T_impact PINN
T_impact (s)
V0 (m/s)


T_impact vs V0 (θ = 80°)
T_impact GT
T_impact PINN
T_impact (s)
V0 (m/s)

```
Checkpoint salvo em D:\Code\cannon_shot_pinn\checkpoints\impact_vacuo\impact-2_64_64_64
Melhor loss: 5.734229e-11 no passo 9374

Treino finalizado.
```

✓ `# ====== Usar melhor modelo ======`

```
Usando melhor modelo (passo 9374) para avaliação
```

✓ `# ====== Avaliação ======`

```
RMSE T_impact: 4.3421e-04 s
RMSE x_impact: 1.0216e-01 m
Erro relativo médio T: 0.01%
Erro relativo médio X: 0.01%
```

Física - Equações Integrais - ARRASTO

$$x(T) = \int_0^T v_x(t)\,dt$$

$$y(T) = \int_0^T v_y(t)\,dt$$

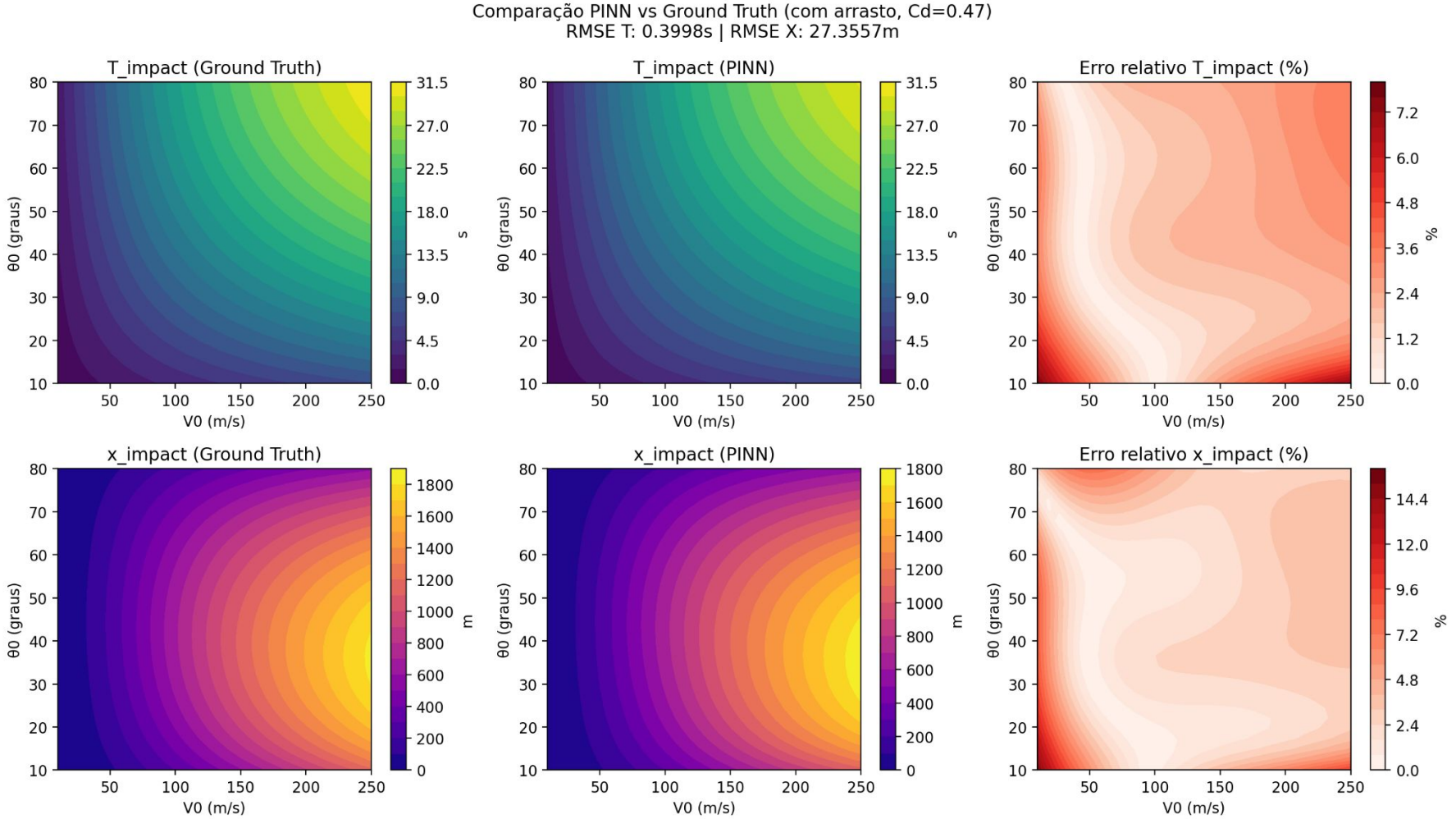
$$\dot{v}_x = -k\,|\vec{v}|\,v_x$$

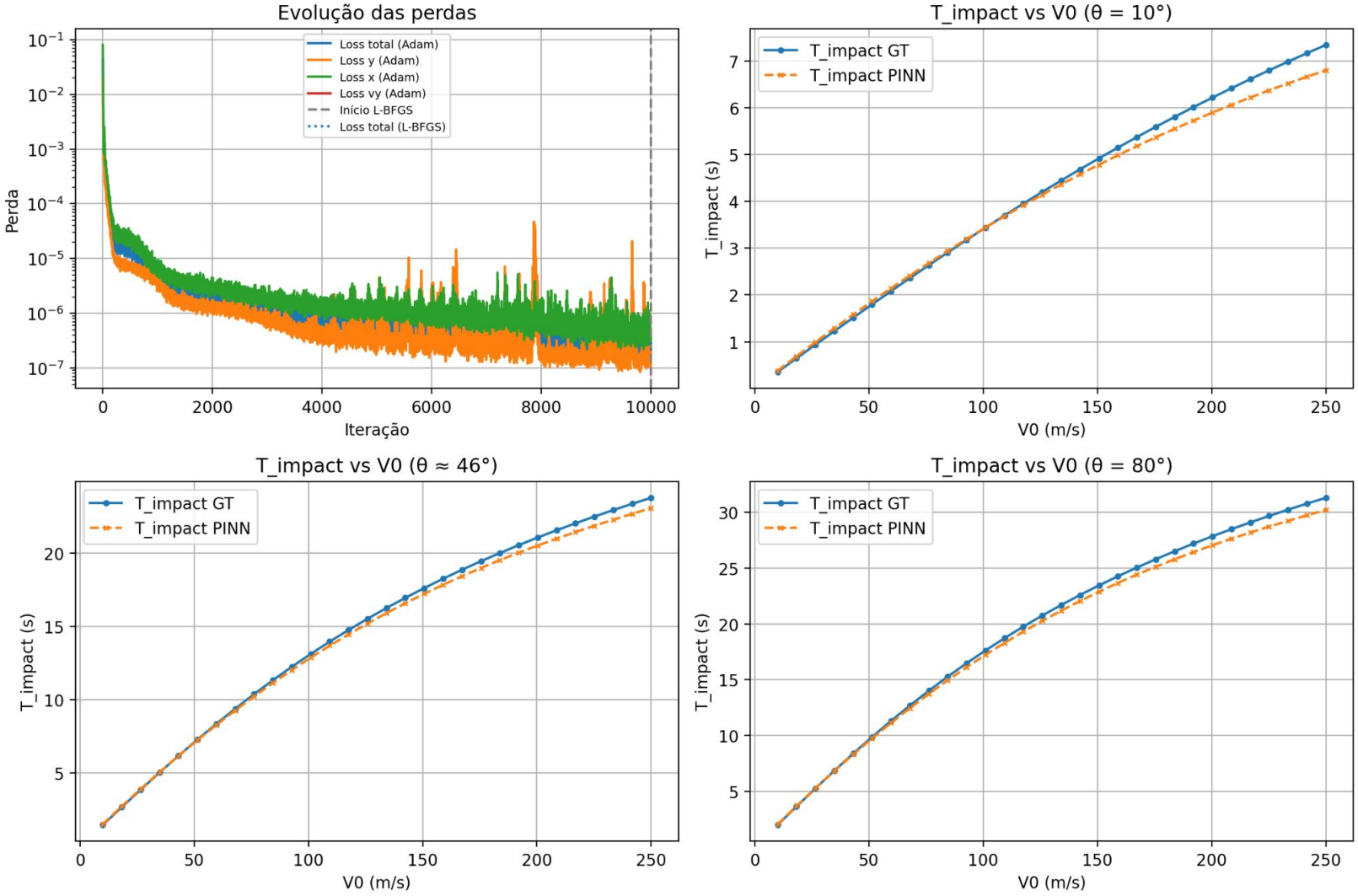
$$\dot{v}_y = -g - k\,|\vec{v}|\,v_y$$

$$\text{onde} \quad k = \frac{\rho\,C_d\,A}{2m}$$

$$\text{e} \quad |\vec{v}| = \sqrt{v_x^2 + v_y^2}$$

```python
def compute_physics(self, v0, theta0_rad, T_impact):
    """
    Calcula y_impact e x_integrated usando integração numérica (Euler) com arrasto.

    v0: tensor (batch,) - velocidade inicial
    theta0_rad: tensor (batch,) - ângulo em radianos
    T_impact: tensor (batch, 1) - tempo de impacto predito

    Retorna: y_final, x_final, vy_final
    """
    batch_size = v0.shape[0]
    device = v0.device

    # Condições iniciais
    vx = v0 * torch.cos(theta0_rad)  # (batch,)
    vy = v0 * torch.sin(theta0_rad)  # (batch,)
    x = torch.zeros(batch_size, device=device)
    y = torch.zeros(batch_size, device=device)

    # Passo de tempo
    dt = T_impact.squeeze(1) / self.n_integration  # (batch,)

    # Integração Euler
    for _ in range(self.n_integration):
        speed = torch.sqrt(vx**2 + vy**2 + 1e-8)  # evita divisão por zero
```


Comparação PINN vs Ground Truth (com arrasto, Cd=0.47)
RMSE T: 0.3998s | RMSE X: 27.3557m
T_impact (Ground Truth)
T_impact (PINN)
Erro relativo T_impact (%)
x_impact (Ground Truth)
x_impact (PINN)
Erro relativo x_impact (%)
θ0 (graus)
V0 (m/s)
s
m
%


Evolução das perdas
Loss total (Adam)
Loss y (Adam)
Loss x (Adam)
Loss vy (Adam)
Início L-BFGS
Loss total (L-BFGS)
Perda
Iteração
T_impact vs V0 (θ = 10°)
T_impact GT
T_impact PINN
T_impact (s)
V0 (m/s)
T_impact vs V0 (θ ≈ 46°)
T_impact GT
T_impact PINN
T_impact (s)
V0 (m/s)
T_impact vs V0 (θ = 80°)
T_impact GT
T_impact PINN
T_impact (s)
V0 (m/s)

```
================================================
Iniciando treinamento com Adam...
Parâmetros de arrasto: Cd=0.47, k=0.000690
================================================
[Adam] step=20000  loss=3.217063e-08  L_y=3.080e-08  L_x=4.069e-08  L_vy=0.000e+00
[Adam] step=20500  loss=9.668148e-08  L_y=1.380e-07  L_x=7.685e-08  L_vy=0.000e+00
[Adam] step=21000  loss=5.218972e-07  L_y=8.387e-07  L_x=3.211e-07  L_vy=0.000e+00
[Adam] step=21500  loss=1.051180e-07  L_y=1.722e-07  L_x=6.140e-08  L_vy=0.000e+00
[Adam] step=22000  loss=4.874240e-08  L_y=4.686e-08  L_x=6.146e-08  L_vy=0.000e+00
[Adam] step=22500  loss=4.403173e-08  L_y=2.351e-08  L_x=7.434e-08  L_vy=0.000e+00
[Adam] step=23000  loss=1.688441e-07  L_y=2.102e-07  L_x=1.650e-07  L_vy=0.000e+00
[Adam] step=23500  loss=3.462886e-08  L_y=2.861e-08  L_x=4.834e-08  L_vy=0.000e+00
[Adam] step=24000  loss=6.814546e-07  L_y=8.969e-07  L_x=6.174e-07  L_vy=0.000e+00
[Adam] step=24500  loss=2.501101e-07  L_y=5.227e-07  L_x=3.310e-08  L_vy=0.000e+00


✓ # ====== (Opcional) Refinamento com L-BFGS ====== ···

...

================================================
Iniciando refinamento com L-BFGS (max 3000 iterações)...
================================================
[L-BFGS] iter=001  loss=1.145920e-07  L_y=1.586e-07  L_x=9.605e-08  L_vy=0.000e+00
L-BFGS finalizado após 8 iterações
Loss final L-BFGS: 2.551184e-08
```

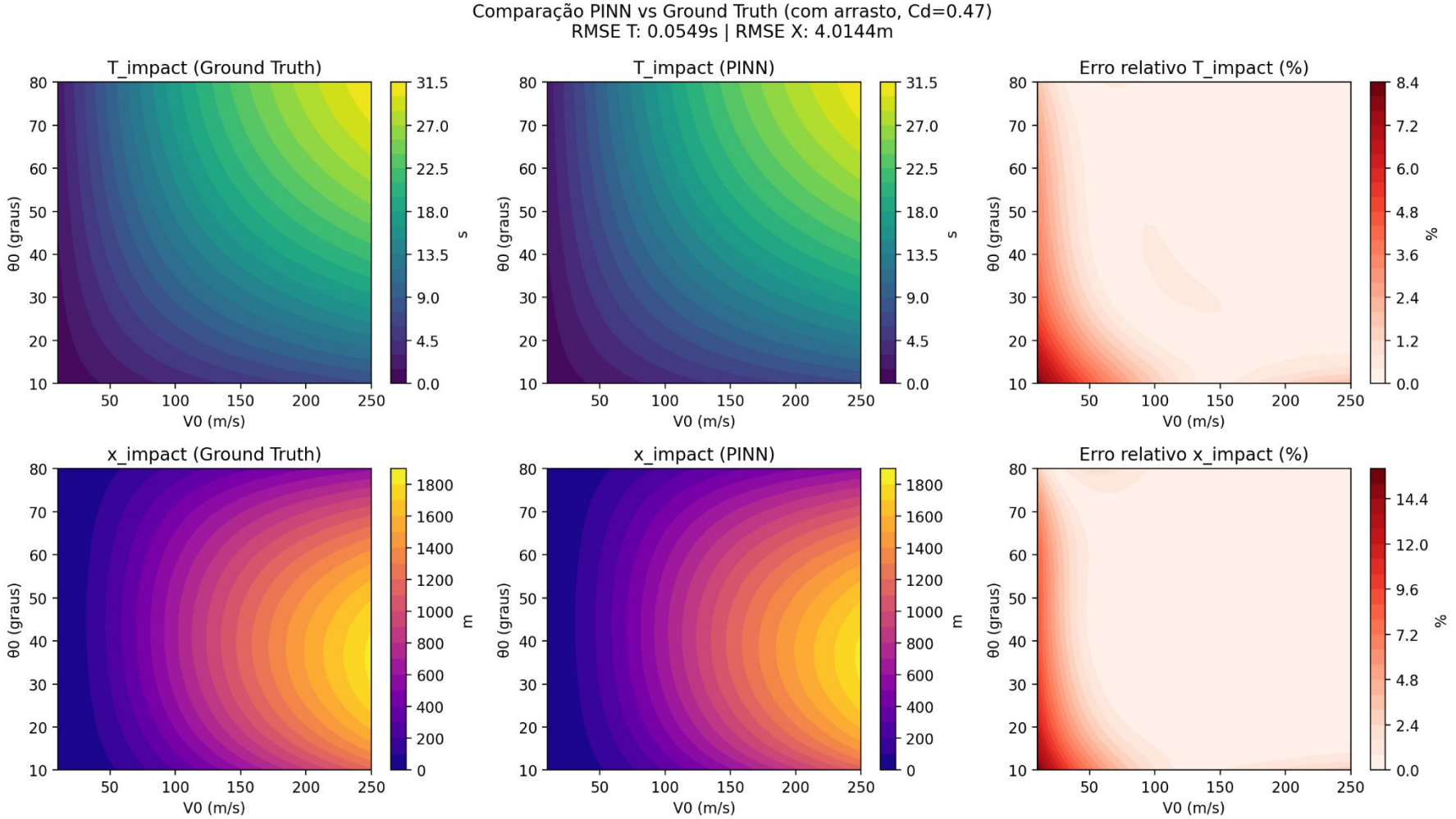
```
Checkpoint salvo em D:\Code\cannon_shot_pinn\checkpoints\impact_arrasto\impact_arrasto-2_64_
Melhor loss: 2.145501e-08 no passo 24822

Treino finalizado.
```

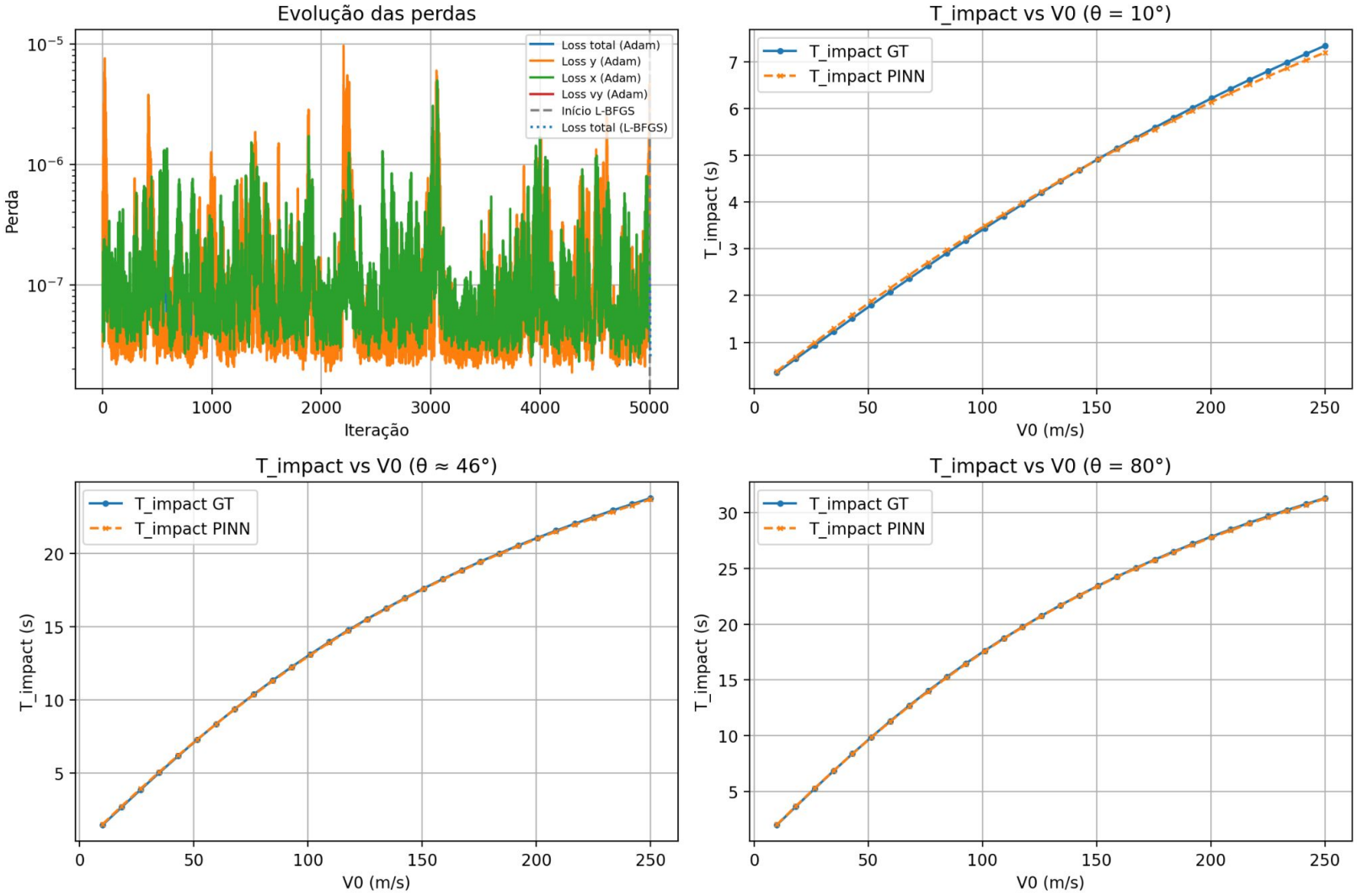
✓ `# ====== Usar melhor modelo ======`

```
Usando melhor modelo (passo 24822) para avaliação
```

✓ `# ====== Avaliação ======`

```
Calculando ground truth (pode demorar)...
Ground truth calculado.

RMSE T_impact: 5.4946e-02 s
RMSE x_impact: 4.0144e+00 m
Erro relativo médio T: 0.76%
Erro relativo médio X: 1.23%
```

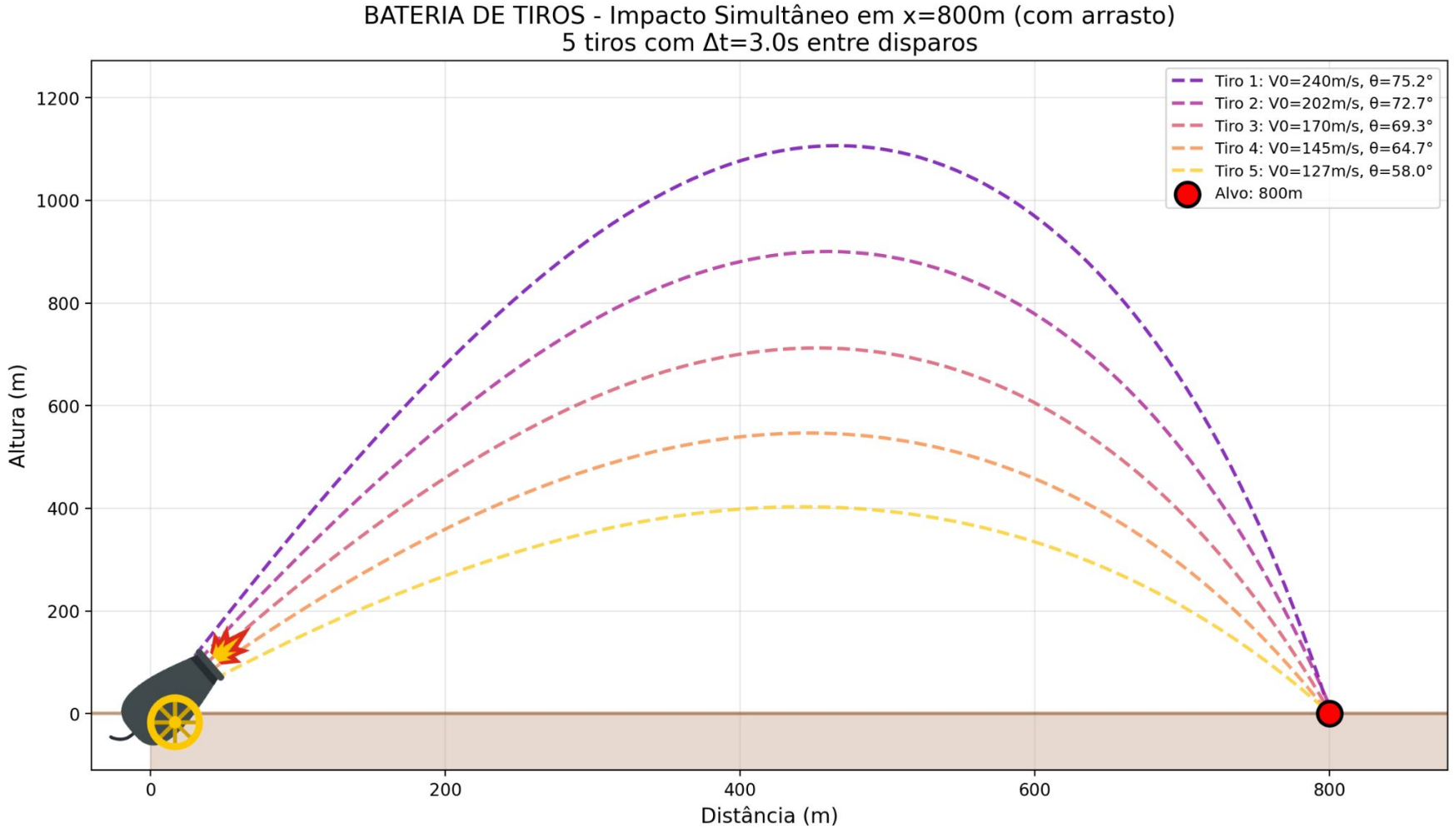

Comparação PINN vs Ground Truth (com arrasto, Cd=0.47)
RMSE T: 0.0549s | RMSE X: 4.0144m
T_impact (Ground Truth)
T_impact (PINN)
Erro relativo T_impact (%)
x_impact (Ground Truth)
x_impact (PINN)
Erro relativo x_impact (%)
θ0 (graus)
V0 (m/s)
s
m
%


Evolução das perdas
Perda
Iteração
Loss total (Adam)
Loss y (Adam)
Loss x (Adam)
Loss vy (Adam)
Início L-BFGS
Loss total (L-BFGS)
T_impact vs V0 (θ = 10°)
T_impact (s)
V0 (m/s)
T_impact GT
T_impact PINN
T_impact vs V0 (θ ≈ 46°)
T_impact (s)
V0 (m/s)
T_impact GT
T_impact PINN
T_impact vs V0 (θ = 80°)
T_impact (s)
V0 (m/s)
T_impact GT
T_impact PINN

```
================================================================================
PROBLEMA INVERSO: Dado x_target, encontrar (V0, θ0, T_impact)
================================================================================

 Objetivo: Minimizar V0 (com arrasto, θ ótimo < 45°)
--------------------------------------------------------------------------------
x_target |       V0       θ |   x_PINN   x_real | V0 Anal. θ Anal. |         Status
--------------------------------------------------------------------------------
     100 |    31.70   44.37° |    100.0     97.1 |    32.19    44.6° |     ✗ Real:2.9%
     500 |    81.26   42.73° |    500.0    503.2 |    80.92    42.8° |           ✓ OK
    1000 |   135.02   40.27° |   1000.0   1001.7 |   134.82    40.3° |           ✓ OK
    1400 |   184.84   38.13° |   1400.0   1402.6 |   184.49    38.1° |           ✓ OK
    1800 |   245.72   36.03° |   1800.0   1804.5 |   244.96    36.0° |           ✓ OK

 Objetivo: Ângulo fixo de 54.0°
--------------------------------------------------------------------------------
x_target |       V0       θ |   x_PINN   x_real | V0 Anal. θ Anal. |         Status
--------------------------------------------------------------------------------
     100 |    32.64   54.00° |    100.0     97.3 |    33.11    54.0° |     ✗ Real:2.7%
     500 |    84.93   54.00° |    500.0    502.0 |    84.69    54.0° |           ✓ OK
    1000 |   146.33   54.00° |   1000.0   1000.5 |   146.26    54.0° |           ✓ OK
    1400 |   210.40   54.00° |   1400.0   1405.4 |   209.42    54.0° |           ✓ OK
    1800 |     Não encontrado |      ---      --- |       Impossível |
```

Todas as soluções para x_target = 800m (PINN vs Ground Truth)

θ (°)	V0 PINN	T PINN	V0 GT	T GT	Erro V0 %
10.0	187.9	5.85	185.6	5.85	1.23
15.0	153.9	7.21	153.8	7.22	0.08
20.0	136.2	8.42	136.0	8.42	0.16
25.0	125.4	9.52	125.1	9.54	0.25
30.0	118.6	10.60	118.3	10.63	0.26
35.0	114.7	11.70	114.4	11.72	0.28
40.0	113.1	12.84	112.7	12.86	0.30
45.0	113.6	14.06	113.3	14.08	0.29
50.0	116.3	15.41	116.0	15.44	0.22
55.0	121.7	16.96	121.6	17.00	0.10
60.0	130.9	18.83	131.0	18.88	0.02
65.0	146.6	21.23	146.7	21.28	0.05
70.0	175.2	24.57	174.9	24.61	0.15
75.0	236.8	29.80	236.5	29.87	0.15


BATERIA DE TIROS - Impacto Simultâneo em x=800m (com arrasto)
5 tiros com Δt=3.0s entre disparos
Tiro 1: V0=240m/s, θ=75.2°
Tiro 2: V0=202m/s, θ=72.7°
Tiro 3: V0=170m/s, θ=69.3°
Tiro 4: V0=145m/s, θ=64.7°
Tiro 5: V0=127m/s, θ=58.0°
Alvo: 800m
Altura (m)
Distância (m)
0
200
400
600
800
1000
1200

```
================================================================================
RESULTADOS
================================================================================

Método                    | T_impact (s)   x_impact (m) |    Tempo (ms)        ± σ |    Speedup
--------------------------------------------------------------------------------
PINN (rede neural)        |      17.2115        1113.65 |       0.7142     0.4129 |         REF
solve_ivp RK45            |      17.2523        1115.00 |       1.6401     0.4085 |        2.3x
solve_ivp RK23            |      17.2548        1114.86 |       2.4854     1.1846 |        3.5x
solve_ivp DOP853          |      17.2500        1114.77 |       2.6300     0.7289 |        3.7x
Euler (dt=0.001)          |      17.2490        1114.67 |      57.5397    15.3974 |       80.6x
Euler (dt=0.01)           |      17.2400        1113.73 |       6.2523     2.5841 |        8.8x
RK4 manual (dt=0.01)      |      17.2600        1115.18 |      68.2141     7.7383 |       95.5x
RK4 manual (dt=0.001)     |      17.2510        1114.81 |     683.8556    56.2365 |      957.5x
--------------------------------------------------------------------------------
```

Próximos passos??

Fazer um operador neural

X_impact => [V0(θ0),T_Impact(θ0)]