LAB 17 | المخبر 17

PINN Solution of the Swing Equation

حل معادلة التأرجح باستخدام شبكة عصبية موجهة بالفيزياء

MATLAB custom training loop with second-order automatic differentiation

حلقة تدريب MATLAB مخصصة مع اشتقاق تلقائي من الرتبة الثانية

Prepared by | إعداد: Dr.-Ing. Aouss Gabash | د. م. أوس غباش

🎯 Laboratory Objectives

- Build a PINN that approximates rotor angle δ(t).
- Compute first- and second-order derivatives using automatic differentiation.
- Minimize swing-equation and initial-condition residuals.
- Compare the PINN with a numerical ODE solver.
- Analyze the influence of loss weights and collocation points.

🎯 أهداف المخبر

- بناء PINN لتقريب زاوية الدوار.
- حساب المشتقات الأولى والثانية تلقائيًا.
- تقليل متبقي معادلة التأرجح والشروط الابتدائية.
- المقارنة مع محلل ODE عددي.
- تحليل أوزان الخسارة ونقاط الفحص.

1. Dynamic Model

$$M\delta'' + D\delta' = Pm - Pmaxsin\delta$$

$$\delta(0) = \delta_0, \delta'(0) = \omega_0$$

The PINN receives time t and outputs δ(t).

1. النموذج الديناميكي

تدخل قيمة الزمن إلى الشبكة وتنتج زاوية الدوار، ثم تُستخدم المشتقات في معادلة التأرجح.

2. Requirements

- MATLAB
- Deep Learning Toolbox

The code uses a `dlnetwork`, `dlgradient` with higher derivatives, and `adamupdate`.

2. المتطلبات

يلزم MATLAB مع Deep Learning Toolbox لدعم الشبكات الديناميكية والاشتقاق التلقائي والتدريب المخصص.

3. Complete MATLAB Script

```
%% Lab 17: PINN for the Swing Equation
clear; clc; close all;
rng(17);

%% 1) Physical parameters
M = 4.0;
D = 1.0;
Pm = 0.80;
Pmax = 1.20;

delta0 = 0.30;
omega0 = 0.00;

tMin = 0;
tMax = 10;

%% 2) Collocation points
numPoints = 500;
tData = linspace(tMin,tMax,numPoints);
dlT = dlarray(tData,"CB");

%% 3) Neural network
layers = [
    featureInputLayer(1,Normalization="none",Name="time")
    fullyConnectedLayer(32,Name="fc1")
    tanhLayer(Name="tanh1")
    fullyConnectedLayer(32,Name="fc2")
    tanhLayer(Name="tanh2")
    fullyConnectedLayer(32,Name="fc3")
    tanhLayer(Name="tanh3")
    fullyConnectedLayer(1,Name="delta")
];

net = dlnetwork(layers);

%% 4) Training settings
numIterations = 8000;
learnRate = 1e-3;
```

```
lambdaPhysics = 1.0;
lambdaDelta0 = 50.0;
lambdaOmega0 = 50.0;

averageGrad = [];
averageSqGrad = [];

lossHistory = zeros(numIterations,1);
physicsHistory = zeros(numIterations,1);
icHistory = zeros(numIterations,1);

%% 5) Custom training loop
for iteration = 1:numIterations

    [loss,gradients,lossPhysics,lossIC] = ...
        dlfeval(@modelLoss,net,dlT, ...
        M,D,Pm,Pmax,delta0,omega0, ...
        lambdaPhysics,lambdaDelta0,lambdaOmega0);

    [net,averageGrad,averageSqGrad] = adamupdate( ...
        net,gradients,averageGrad,averageSqGrad, ...
        iteration,learnRate);

    lossHistory(iteration) = ...
        double(gather(extractdata(loss)));
    physicsHistory(iteration) = ...
        double(gather(extractdata(lossPhysics)));
    icHistory(iteration) = ...
        double(gather(extractdata(lossIC)));

    if mod(iteration,500)==0
        fprintf(['Iteration %5d | Total %.3e | ' ...
                 'Physics %.3e | IC %.3e\n'], ...
            iteration,lossHistory(iteration), ...
            physicsHistory(iteration), ...
            icHistory(iteration));
    end
end

%% 6) PINN prediction
```

```
tTest = linspace(tMin,tMax,1000);
dlTTest = dlarray(tTest,"CB");
deltaPINN = extractdata(forward(net,dlTTest));

%% 7) Numerical reference solution
state0 = [delta0;omega0];

odefun = @(t,x) [
    x(2)
    (Pm-Pmax*sin(x(1))-D*x(2))/M
];

[tODE,xODE] = ode45(odefun,[tMin tMax],state0);

deltaODE = interp1(tODE,xODE(:,1),tTest,"pchip");

%% 8) Error
MAE = mean(abs(deltaPINN-deltaODE));
RMSE = sqrt(mean((deltaPINN-deltaODE).^2));

fprintf('\nPINN MAE  = %.4e rad\n',MAE);
fprintf('PINN RMSE = %.4e rad\n',RMSE);

%% 9) Plots
figure;
plot(tTest,deltaODE,"k",LineWidth=1.5); hold on;
plot(tTest,deltaPINN,"--",LineWidth=1.5);
grid on;
xlabel("Time (s)");
ylabel("Rotor Angle delta (rad)");
legend("ODE45 Reference","PINN");
title("Swing-Equation Solution");

figure;
semilogy(lossHistory,LineWidth=1.2); hold on;
semilogy(physicsHistory,LineWidth=1.2);
semilogy(icHistory,LineWidth=1.2);
grid on;
xlabel("Iteration");
ylabel("Loss");
```

```
legend("Total","Physics","Initial Conditions");
title("PINN Training History");

%% Local loss function
function [loss,gradients,lossPhysics,lossIC] = ...
    modelLoss(net,t,M,D,Pm,Pmax,delta0,omega0, ...
    lambdaPhysics,lambdaDelta0,lambdaOmega0)

    % Network prediction
    delta = forward(net,t);

    % First derivative
    delta_t = dlgradient(sum(delta,"all"),t, ...
        EnableHigherDerivatives=true);

    % Second derivative
    delta_tt = dlgradient(sum(delta_t,"all"),t, ...
        EnableHigherDerivatives=true);

    % Swing-equation residual
    residual = M*delta_tt + D*delta_t ...
        - Pm + Pmax*sin(delta);

    lossPhysics = mean(residual.^2,"all");

    % Initial conditions
    t0 = dlarray(0,"CB");
    deltaAt0 = forward(net,t0);

    delta0_t = dlgradient(sum(deltaAt0,"all"),t0, ...
        EnableHigherDerivatives=true);

    lossDelta0 = (deltaAt0-delta0).^2;
    lossOmega0 = (delta0_t-omega0).^2;
    lossIC = lossDelta0+lossOmega0;

    % Total loss
    loss = lambdaPhysics*lossPhysics ...
        + lambdaDelta0*lossDelta0 ...
        + lambdaOmega0*lossOmega0;
```

```
    % Gradients with respect to network parameters
    gradients = dlgradient(loss,net.Learnables);
end
```

3. الكود الكامل

يبني الكود شبكة تستقبل الزمن وتنتج الزاوية، ثم يحسب المشتقتين ويصغر متبقي معادلة التأرجح والشروط الابتدائية.

4. Physics Loss

$$r(t) = M\delta\hat{}''+D\delta\hat{}'-Pm + Pmaxsin\delta\hat{}$$

$$Jphysics = mean(r^2)$$

The network is trained even without labeled trajectory data.

4. الخسارة الفيزيائية

تُدرب الشبكة باستخدام المعادلة نفسها، حتى دون توفير قيم صحيحة للزاوية داخل المجال.

5. Initial-Condition Loss

$$JIC = (\hat{\delta}(0) - \delta_0)^2 + (\hat{\delta}'(0) - \omega_0)^2$$

Large initial-condition weights help anchor the solution at t=0.

5. خسارة الشروط الابتدائية

تثبت قيمة الزاوية والسرعة في بداية المحاكاة، وتساعد الأوزان الكبيرة على منع انحراف الحل.

6. Why Higher Derivatives Are Enabled

The second derivative requires differentiating a derivative. Therefore the first call to automatic differentiation must preserve the derivative graph.

6. لماذا نفعّل الاشتقاقات العليا؟

لأننا نشتق المشتقة الأولى مرة ثانية، يجب الاحتفاظ بالرسم الحسابي أثناء الاشتقاق الأول.

7. Expected Results

- Total and physics losses decrease.
- The PINN follows the ODE45 reference trajectory.
- Largest errors may occur near rapid transients.
- Insufficient collocation points can produce oscillatory residuals.

7. النتائج المتوقعة

يجب أن ينخفض الخطأ ويتبع حل PINN مسار ODE45، وقد تظهر أكبر الأخطاء خلال التغيرات السريعة.

8. Student Experiments

1. Change inertia M from 4 to 2 and 8.
2. Change damping D.
3. Increase disturbance severity by changing Pm.
4. Compare 100, 500, and 2000 collocation points.
5. Test different loss weights.
6. Add 10 noisy measured angle samples to the loss.

8. تجارب الطالب

1. غيّر العطالة.
2. غيّر التخميد.
3. زد شدة الاضطراب.
4. غيّر عدد نقاط الفحص.
5. اختبر أوزان خسارة مختلفة.
6. أضف قياسات زاوية مشوبة بالضجيج.

9. Inverse PINN Extension

Make M or D learnable and estimate them from sparse trajectory measurements.

$$J = Jphysics + \lambda data \sum |\delta\hat{}(ti) - \delta meas, i|^2$$

9. توسع إلى مسألة عكسية

اجعل العطالة أو التخميد معاملًا قابلًا للتعلم، ثم قدره من عدد محدود من القياسات.

10. Mini Project

Multi-machine PINN: Extend the formulation to several coupled generators and compare a shared network with one network per machine.

10. مشروع مصغر

وسّع النموذج إلى عدة مولدات مترابطة وقارن بين شبكة مشتركة وشبكة مستقلة لكل مولد.

11. Laboratory Report

- Derive the swing residual.
- Explain automatic differentiation.
- Report loss weights and collocation points.
- Compare PINN and ODE45 using MAE and RMSE.
- Discuss failure cases.
- Present one inverse-problem extension.

11. تقرير المخبر

يتضمن اشتقاق المتبقي وشرح الاشتقاق التلقائي وأوزان الخسارة والمقارنة مع ODE45 وحالات الفشل وتوسعًا عكسيًا.