

UTILIZAREA SISTEMELOR HPC ÎN REȚELE NEURONALE

Vladimir MURSA,

masterand, anul II, Program SMMSPA, Facultatea Matematică și Informatică,
Universitatea de Stat din Moldova

CZU: 004.032.26

E-mail: mursa.079443885@gmail.com

I propose to approach, from a rigorous mathematical perspective, the problem of electronic signal propagation in two-dimensional spaces. For this: All mathematical calculations and processing necessary for training the neural network will be performed on an HPC computing cluster. The goal is to demonstrate that distributing the tasks across multiple computing nodes not only shortens the training time, but also increases the accuracy of the results. Subsequently, the neural network should be trained to understand and predict the behavior of waves.

Ecuția undelor reprezintă o ecuație diferențială cu derivate parțiale de tip hiperbolic, formulată pentru a descrie oscilațiile transversale de amplitudine mică ale unei membrane subțiri sau ale unei corzi. Aceasta modelează, de asemenea, alte procese oscilatorii în medii continue, inclusiv fenomene acustice în regim predominant liniar și fenomene electromagnetice. Ecuția se extinde și la alte ramuri ale fizicii teoretice, cum ar fi studiul undelor gravitaționale, reprezentând astfel unul dintre pilonii ecuațiilor fundamentale ale fizicii matematice.

Forma generală a ecuației undelor într-un spațiu n -dimensional este dată de:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u, \quad (1.1)$$

unde $u(x, t)$ este funcția necunoscută, reprezentând deplasarea undei la poziția $x \in \mathbb{R}^n$ la intervalul t , c este viteza de propagare a undei, ∇ este operatorul Laplace.

În spațiu bidimensional, ecuația undelor se formulează astfel:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \quad (1.2)$$

unde $u = u(x, y, t)$ este funcția necunoscută ce descrie amplitudinea undei la un punct (x, y) în momentul de timp t , c este viteza de propagare a undei, o constantă ce depinde de proprietățile fizice ale mediului.

Pentru a rezolva ecuația numeric, împărțim spațiul și timpul în noduri de rețea cu pași Δx , Δy și Δt , folosind metoda explicită a diferențelor finite. entralizate:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2}, \quad \frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2}, \quad (1.3)$$

$$\frac{\partial^2 u}{\partial t^2} \approx \frac{u_{i,j}^{n+1} - 2u_{i,j}^n + u_{i,j}^{n-1}}{\Delta t^2}$$

Înlocuind (1.3) în ecuația inițială (1.2), obținem schema explicită:

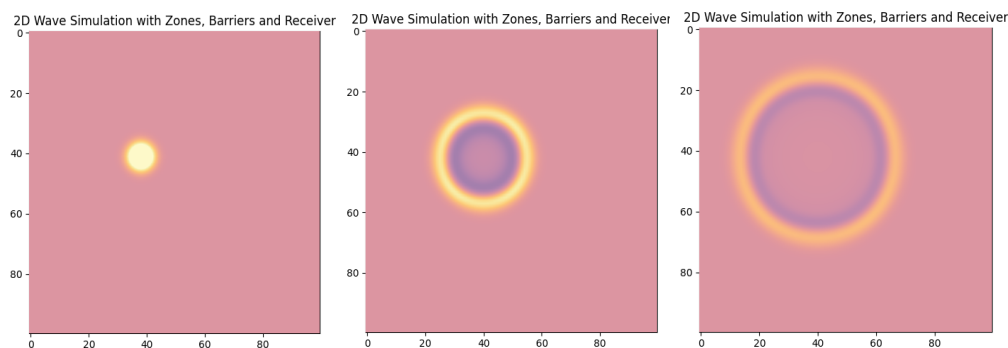
$$\frac{u_{i,j}^{n+1} - 2u_{i,j}^n + u_{i,j}^{n-1}}{\Delta t^2} = c^2 \left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right) \quad (1.4)$$

Rearanjând pentru $u_{i,j}^{n+1}$ obținem:

$$u_{i,j}^{n+1} = 2u_{i,j}^n - u_{i,j}^{n-1} \lambda_x (u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \lambda_y (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n), \quad (1.5)$$

unde $\lambda_x = \frac{c^2 \Delta t^2}{\Delta x^2}$, $\lambda_y = \frac{c^2 \Delta t^2}{\Delta y^2}$ condiționalui Courant.

Considerăm acum modelarea propagării undelor folosind un cluster. În primul rând, vom pregăti mediul de dezvoltare și vom instala bibliotecile necesare, cum ar fi: *numpy matplotlib mxnet mpi4py pillow pyqt scipy zmq*. În cadrul acestei lucrări, s-a optat pentru o arhitectură de lucru hibridă, în care execuția calculelor intensive este realizată pe cluster, iar vizualizarea, interpretarea și analiza rezultatelor sunt efectuate local, pe un laptop. Această separare a sarcinilor permite utilizatorului să beneficieze atât de puterea de procesare a clusterului, cât și de confortul unei interfețe grafice locale pentru analiza și vizualizarea datelor. Un program Python va fi scris pentru fiecare dispozitiv (laptop și cluster), iar după ce îl lansăm făcând clic cu mouse-ul, putem crea o undă [1, p.2].



[1, p.2]

În cele din urmă, să creăm o rețea neuronală care va deprinde să clasifice comportamentul undelor. În primul rând, se inițializează în mod aleatoriu matricele de greutate $W^{[l]}$ și vectorii de bias $b^{[l]}$ pentru fiecare strat l al rețelei. Datele de intrare x sunt apoi preluate și preprocesate, iar fiecare exemplu este introdus în rețea prin atribuirea

valorii $a^{[0]} = x$. Procedură se repetă la fiecare strat l conform formulei unde g este funcția de activare:

$$a^{[l]} = g^{[l]}(W^{[l]} \cdot a^{[l-1]} + b^{[l]}) \quad (1.6)$$

Rezultatul final al rețelei, $\hat{y} = a^{[L]}$, reprezintă predicția modelului pentru exemplul curent.

Pentru a evalua performanța rețelei, se calculează funcția de cost $J(W, b)$ care măsoară diferența dintre predicțiile $\hat{y}$ și valorile reale y :

$$J(W, b) = \frac{1}{m} \cdot \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \quad (1.7)$$

Imediat după calculul erorii, se procedează la etapa de backpropagation. Această etapă constă în determinarea gradientelor funcției de cost față de fiecare parametru al rețelei, folosind regula lanțului. Pentru stratul final, eroarea este definită prin relația:

$$\delta^{[L]} = \frac{\partial J}{\partial z^{[L]}} \quad (1.8)$$

Această eroare este apoi retropropagată la straturile precedente folosind relația recursivă:

$$\delta^{[l]} = (W^{[l+1]T} \delta^{[l+1]}) \odot g'^{[l]}(z^{[l]}) \quad (1.9)$$

Aceste erori parțiale permit calculul derivatelor funcției de cost în raport cu greutatea și bias-urile fiecărui strat, conform relațiilor:

$$\frac{\partial J}{\partial W^{[l]}} = \frac{1}{m} \cdot \delta^{[l]} a^{[l-1]T} \text{ și } \frac{\partial J}{\partial b^{[l]}} = \frac{1}{m} \sum_{i=1}^m \delta^{[l]}(i) \quad (1.10)$$

Ulterior, acești parametri sunt actualizați folosind algoritmul de gradient descent. Actualizarea pentru fiecare parametru se face conform formulelor:

$$W^{[l]} := W^{[l]} - \eta \frac{\partial J}{\partial W^{[l]}} \text{ și } b^{[l]} := b^{[l]} - \eta \frac{\partial J}{\partial b^{[l]}} \quad (1.11)$$

unde η este rata de învățare care determină dimensiunea pasului efectuat în direcția gradientului negativ.

Acum putem antrena modelul în paralel pe mai multe procese utilizând un set de date creat din imaginile de simulare.

Modelul converge în mai puțin de 20 de epoci. Validarea pe date de așteptare generează o precizie de 96,5% și scoruri F1 de peste 0,95 pentru ambele clase, demonstrând o recunoaștere robustă a modelelor de difracție chiar și în condiții de zgomot sau configurații noi ale obstacolelor.

Referințe:• *Cărți și publicații monografice:*

1. EVANS, L. C. (2010). *Partial Differential Equations* (2nd ed.). American Mathematical Society.
2. COURANT, R., FRIEDRICHS, K., & LEWY, H. (1928). Über die partiellen *Differenzen-gleichungen der mathematischen Physik*. Mathematische Annalen.
3. BURDEN, R. L., & FAIRES, J. D. (2011). *Numerical Analysis* (9th ed.). Brooks Cole.
4. LEVEQUE, R. J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM.

*Recomandat****Boris HÎNCU***, dr., conf. univ.