



BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM  
GÉPÉSZMÉRNÖKI KAR  
MŰSZAKI MECHANIKAI TANSZÉK



TDK dolgozat

SZÁSZ ZSOLT

Nemlineáris anyagi viselkedés  
adatalapú mechanikai modellezése  
neurális hálózatok segítségével

Konzulens:

*Dr. Kossa Attila*  
Egyetemi Docens

BUDAPEST, 2024

# Tartalomjegyzék

|                                                                            |           |
|----------------------------------------------------------------------------|-----------|
| <b>1. Bevezetés</b>                                                        | <b>1</b>  |
| 1.1. Adataalapú megközelítés előnyei . . . . .                             | 1         |
| 1.2. Irodalmi áttekintő . . . . .                                          | 2         |
| 1.3. Általános megközelítés . . . . .                                      | 4         |
| <b>2. A probléma felvetése</b>                                             | <b>5</b>  |
| 2.1. Anyagi nemlineáritás, kétdimenziós végeelem esetén . . . . .          | 5         |
| 2.1.1. Newton-Rhapon iteráció . . . . .                                    | 7         |
| 2.1.2. Növekményes terhelés módszere . . . . .                             | 8         |
| 2.2. Egydimenziós rugalmas-képlékeny anyagmodell . . . . .                 | 9         |
| 2.3. Egydimenziós rugalmas-képlékeny anyagmodell numerikus megoldása . . . | 11        |
| <b>3. A probléma általánosítása</b>                                        | <b>12</b> |
| <b>4. Lehetséges megoldások</b>                                            | <b>15</b> |
| 4.1. Példa választása . . . . .                                            | 16        |
| 4.2. Adat generálás . . . . .                                              | 17        |
| 4.3. Modell alkotás . . . . .                                              | 20        |
| 4.4. MLP hálózatok . . . . .                                               | 22        |
| 4.4.1. Direkt modellezés . . . . .                                         | 25        |
| 4.4.2. Inkrementális modellezés . . . . .                                  | 27        |
| 4.5. LSTM hálózatok . . . . .                                              | 30        |
| 4.5.1. Direkt modellezés . . . . .                                         | 32        |
| 4.5.2. Inkrementális modellezés . . . . .                                  | 35        |
| <b>5. Összegzés</b>                                                        | <b>38</b> |
| <b>Hivatkozások</b>                                                        | <b>41</b> |
| <b>Függelék</b>                                                            | <b>44</b> |

# 1. Bevezetés

A szimuláció mérnöki szempontból elengedhetetlen, mivel költséghatékony és időtakarékos megoldást nyújthat a tervezés optimalizálására és a bonyolult rendszerek viselkedésének elemzésére. Ugyanakkor lehetővé teszi a valós tesztek számának redukálását, miközben segíthet előre jelezni, hogyan is viselkednek a rendszerek különböző körülmények között.

A mérnöki területek nagy részén a szimulációk során szükséges figyelembe vennünk az anyagok mechanikai viselkedését. Ez a tény nagyon fontos feladattá teszi az anyagmodellezést és szükségessé teszi a minél pontosabb és átfogóbb konstitutív modellek fejlesztését, ugyanakkor egy kiemelkedően nehéz feladat, hiszen az alapanyag viselkedése rendkívül összetett és ebből kifolyólag a legtöbb anyag a modellezni kívánt tartományban erős nemlineáris viselkedést mutat, ami megnehezíti a tisztán analitikus megközelítést. Emellett az analitikus modellek alkalmazása esetén gyakran nemlineáris egyenletek numerikus megoldásához kell folyamodnunk, ami egy sok elemből és időlépésből álló végeselemes szimuláció esetén viszonylag magas számítás igényvel bírhat, ugyanis ezek megoldása szintén csak numerikusan, iteráció segítségével lehetséges.

## 1.1. Adataalapú megközelítés előnyei

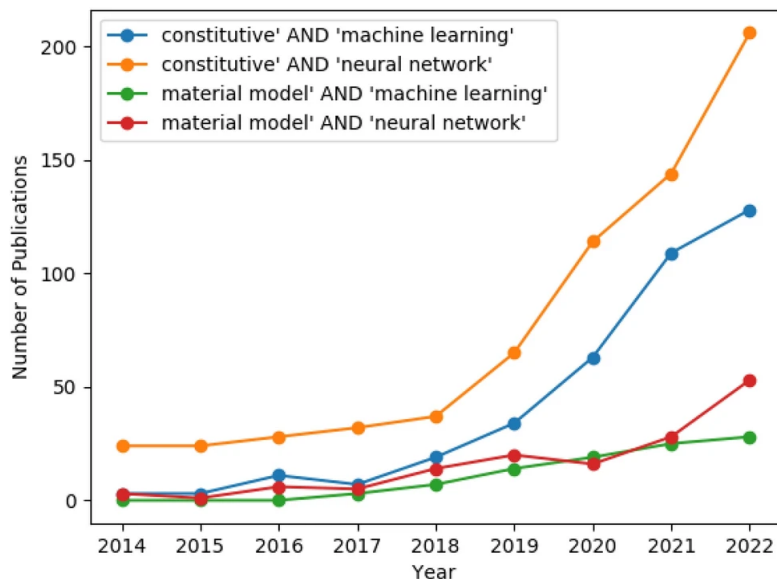
Ez a dolgozat olyan irányból közelíti ezt a témakört, ami megoldást nyújthat az anyagmodellezés fő nehézségeire. Ez az adataalapú megközelítés, amely alkalmas arra, hogy pusztán megfelelő mennyiségű és megfelelően változatos minta adat/adatsor alapján kellően pontos modellt eredményezzen. Erre a legelterjedtebb és legszélesebb körben alkalmazható eszközök a neurális hálózatok, melyek legelső előnye az, hogy megfelelően megválasztott adatok és optimalizációs eljárás alapján képesek akár ismeretlen, nemlineáris viselkedést mutató rendszerek viszonylag pontos modellezésére is. Ezek további előnye akkor mutatkozik meg, amikor a konstitutív modell egy végeselemes szimuláció során betöltött szerepét vizsgáljuk. Ennek megértéséhez, vegyük a későbbiekben a dolgozatban példaként szolgáló egydimenziós rugalmas képlékeny-modellt, mely esetén a folyáshatár változását egy nemlineáris  $Y(\lambda)$  függvény definiálja, ugyanis így jobb közelítése a terheléses vizsgálatok során tapasztalt keményedési karakterisztikának. Viszont ez a nemlinearitás a végeselemes szimuláció szempontjából az idő növekményen és a Newton-Raphson iteráción (a későbbiekben röviden a dolgozat értekezik ezekről) felül, még egy iterációs ciklust jelent, ugyanis amikor rugalmas-képlékeny alakváltozás megy végbe, akkor a felhalmozódott képlékeny alakváltozás inkrementjét egy nemlineáris egyenletből (1.1) tudjuk kiszámolni:

$$|\sigma_n + E\Delta\varepsilon| - E\Delta\lambda - Y(\lambda_n + \Delta\lambda) = 0. \quad (1.1)$$

Ennek megoldása szintén csak numerikusan, iteráció segítségével lehetséges. Ezt az iteratív ciklust megspórolhatjuk az adatalapú megközelítéssel, ugyanis ott a háló betanítását követően, a feszültség számolása minden lépés esetén egyenes vonalú. Emellett a neurális hálózatok hardware optimalizálása is rohamosan fejlődik, melyből kifolyólag a hálók predikációs sebessége is rohamosan növekszik. Így ez a megközelítés, megfelelő illesztéssel és hardware-el egy sok időlépést és sok végeselemet tartalmazó szimuláció során lehetőséget adhat a szimulációs idő csökkentésére.

## 1.2. Irodalmi áttekintő

Viszonylag korán felismerték a neurális hálók fent is röviden ismertetett előnyeit a konstitutív modellezésben. Ghaboussi 1991-es [1] értekezése számít az úttörő munkának ebben a témában. Ezt követően a mai napig számos próbálkozás születik, ahol különböző anyagi viselkedéseket különböző körülmények között próbálnak adatalapon modellezni. Ezen terület fejlődéséről olvashatunk Dornheim 2024-es [2] irodalmi áttekintőjében, mely kiemelkedő részletességgel járja körül a téma fejlődését és ennek főbb mérföldköveit, az 1991-es megjelenésétől napjainkig. Legyen szó bármilyen adatalapú alkalmazásról, a számítástechnika és a gépi tanulási metódusok fejlődéséből kifolyólag mind rohamosan növekvő tendenciát mutatnak, ez alól ez a terület sem kivétel, melyet a [2]-ben szereplő statisztika szemléltet a legjobban.



1.1. ábra. Az adatalapú konstitutív modellezés növekvő tendenciája [2].

Ugyancsak korán keletkeztek már olyan munkák, amelyek ezeket a konstitutív modellek végeselemes szimulációk részeként alkalmazták, mint például [3]. Szintén viszonylag korán, 1998-ban Furukawa [4] munkájában a konstitutív modellezést az irányítástechnikából ismert leírásmóddal látta el, annak érdekében, hogy a neurális

hálózatok esetén használatos ábrázolási módhoz és jelölésrendszerhez jobban illeszkedjen. Ezt a megközelítés több jövőbeli publikációban megfigyelhető. Ebben a cikkben a szerző egy diszkrét dinamikai rendszerként tekint a konstitutív modellre. Ezzel kapcsolatban fontos megjegyezni, hogy annak ellenére, hogy mechanikai problémáról beszélünk a "dinamika" kifejezés nem tévesztendő össze a mechanikában használatos dinamikával, esetünkben a dinamikai rendszer matematikai értelemben vett kifejezés, amelyet általánosan a (1.2) egyenlet definiál.

$$y_{n+1} = f(y_n, u_n), \quad n = \{1, 2, 3, \dots\}, \quad (1.2)$$

ahol az  $u_n$  sorozat minden  $n$ -re adottnak tekinthető. A későbbiekben a dolgozatban is részletesen be lesz mutatva a konstitutív modellek ezen formulációja, egészen az alapoktól kiindulva.

A korai munkák esetén mint [1], [4] vagy [3] sikerült arányosan változó terhelésre eredményes interpolatív modelleket alkotni, mely kifejezés itt arra vonatkozik, hogy a modellek a háló által a tanítás során látott idősorokra alkalmazható eredményesen. Természetesen a kései munkák esetén már jóval szélesebb körben használható modellek keletkeztek, melyek többnyire a pontosság érdekében korlátozott karakterisztikára, viszont extrapolatív módon is alkalmazhatóak. Vegyük például Gorji 2019-es munkáját [5], melyben egy rugalmas-képlékeny, síkfeszültség feladatot vesz példaként. Ezt követően oly módon generál terhelés idősorokat, hogy azok a fémek megmunkálása során jellemzően előforduló terheléstörténeteket tükrözzenek, ezáltal természetesen, ahogy a cikk is kiemeli nem-arányos terheléstörténetek születnek. Ezeket felhasználva az analitikus modell segítségével előállítja az ezeknek megfelelő terhelésválaszokat és ezekkel tanít be egy modellt. A módszer esetünkben is hasonló lesz, viszont ebben a dolgozatban nagyobb hangsúlyt kap a változatos terheléstörténetre való alkalmazhatóság, ugyanis így érhető el az, hogy a modell az ne illeszkedjen túl a bemenő adatokra és magát a belső működést modellezze. Viszont mivel a [5]-ben konkrét karakterisztikájú terheléstörténetre törekedtek, ezért az ott szereplő modellek nem láttak például leterhelést, illetve oszcillációt így várhatóan merőben más bemenetre nem adnának olyan pontos választ. Ebben az értekezésben a szerző szintén bemutatta, hogy hogyan lehet egy ilyen modellt végeselemes szimuláció során alkalmazni. Esetében egy kereskedelmi forgalomban lévő végeselemes szoftverbe is becsatornázták a modelljét és szimulációkat is összehasonlított. Egy másik megközelítés, amikor a [4]-től eltérően nem diszkrét rendszerként tekintik a konstitutív modelleket, hanem egész idősorokat használnak fel bemenetként. Az ily módon készült modellek esetén a legnagyobb probléma értelemszerűen a hálóméret, illetve ezáltal a szimuláció során való alkalmazhatóság kérdése. Az ilyen megközelítésekre egy megoldást nyújtott du Bos 2021-es [6] munkája, mely esetén a háló bemeneteként szolgáló pontokat

egy viszonylag sűrű idősorból mintavételezi, ezekre tanít modellt, majd pedig mindig a háló által predikált pontokra interpolálja az alakváltozás feszültség függvényét.

A korábban említett munkák mind MLP (Multi Layer Perceptron) hálózatokat használtak a modellezésre, ugyanakkor léteznek olyan hálózatok is, amelyeket idősorok modellezésére fejlesztettek ki. Név szerint ezek a RNN (Recurrent Neural Network) hálózatok, melyek már egyfajta memória jelleggel is rendelkeznek és ebből kifolyólag egy cella működését a  $\mathbf{h}_n = \mathbf{f}(\mathbf{h}_{n-1}, \mathbf{x}_n)$  egyenlet adja meg, ahol  $\mathbf{h}_n$  a cella kimenete és  $\mathbf{x}_n$  pedig a cella bemenete. Jól látható, hogy a cella valójában egy dinamikai rendszert reprezentál. Bár kisebb számban, de léteznek olyan publikációk is, ahol ilyen rekurens hálózatokkal próbálkoznak konstitutív modellek létrehozására. Egy korai példa erre [7], ahol felejtő reológiai anyagok feszültségállapotának meghatározására használnak eredményesen RNN hálózatot. Az LSTM (Long-Short Term Memory) hálózatok a legelterjedtebb rekurens hálózatok, talaj illetve más granuláris anyagok anyagi viselkedésének LSTM-el történő modellezésről olvashatunk Zhang 2020-es [8] és 2021-es [9] cikkében.

### 1.3. Általános megközelítés

Általánosítva a problémát lehetőségünk nyílik arra, hogy nem csak a neurális háló alapú konstitutív modellezés területén elért eredményekre támaszkodjunk, hanem figyelembe vesszük a nemlineáris diszkrét dinamikai rendszerek és idősorok neurális hálózatokkal történő modellezésében elért eredményeket is. Mivel ez a terület más ipari alkalmazás fejlesztését is elősegíti, ezért szintén rohamosan fejlődik. Az elmúlt évek egyik legérdekesebb és legeredményesebb előrelépést a témában a 2018-ban megjelent nODE (Neural Ordinary Differential Equation) [10] hálózatok jelentetik, mely a Runge-Kutta integrátor formulán és az újnak és népszerűnek számító Automatic differentiation [11] módszeren alapul és lehetőséget ad a dinamikai rendszerek belső dinamikájának modellezésére. Ebben a dolgozatban a már klasszikussá vált neurális hálózatokkal fogunk próbálkozni, alapul véve Gonalez 2018-as [12] munkáját, ahol felméri a klasszikus struktúrák modellezési lehetőségeit a diszkrét dinamikai rendszerekre vonatkozólag.

A dolgozat elsősorban egy részletes példaként szolgál arra, hogy miként is lehet egy ilyen fizikai interpretációval rendelkező rendszert adat-vezérelt módon modellezni. Más hasonló megközelítésekhez hasonlóan mi is egy ismert egydimenziós analitikus anyagmodellt veszünk alapul, pontosabban egy rugalmas képlékeny anyagmodellt, majd ezt megpróbáljuk adatain modellezni. Az olyan értekezések esetén, ahol ugyancsak rugalmas-képlékeny modellt járnak körül, egyaránt a régebbi [4], illetve újabb [5] publikációk esetén felhasználják a modell tanítás során a felhalmozódott képlékeny alakváltozást. Ebben a dolgozatban arra törekszünk, hogy egy általánosan használható modellezési módszert alkossunk, mely független az előre definiált belső változóktól, ezért

mi kizárólag a feszültség és alakváltozás értékeket használjuk fel a modellalkotás során. A korábbi eredményektől eltérően mi azt is kitűzzük célul, hogy a lehető legextrapolatívabb modellt alkossuk, tehát a lehető legáltalánosabb bemenetre is elfogadható pontosságú legyen a végső modellünk. Téve ezt úgy, hogy a szimuláció szempontjából lényeges tulajdonságokat ne veszítsük el. Szimuláció szempontjából a legfontosabb az, hogy a modellünk nem csak arra kell képes legyen, hogy a korábbi állapotok alapján pontosan predikálja a következő állapotot, hanem arra, hogy a kezdeti állapotból az egész állapottörténetet pontosan meg tudja jósolni. Szimuláció szempontjából a másik fontos követelmény a kevés paraméter.

## 2. A probléma felvetése

Első körben megnézzük, hogy egy csupán anyagi nemlinearitást feltételező, kétdimenziós végeselemes szimuláció esetén hol is jön képbe a konstitutív modell. Ezt követően megvizsgálunk egy ismert egydimenziós analitikus konstitutív modellt, hogy megértsük miként is történik a különböző anyagi veselkedések matematikai megfogalmazása, majd megnézzük annak a numerikus megoldását, ezáltal megértve, hogy miként is van megvalósítva egy végeselemes szimuláció során.

### 2.1. Anyagi nemlineáritás, kétdimenziós végeselem esetén

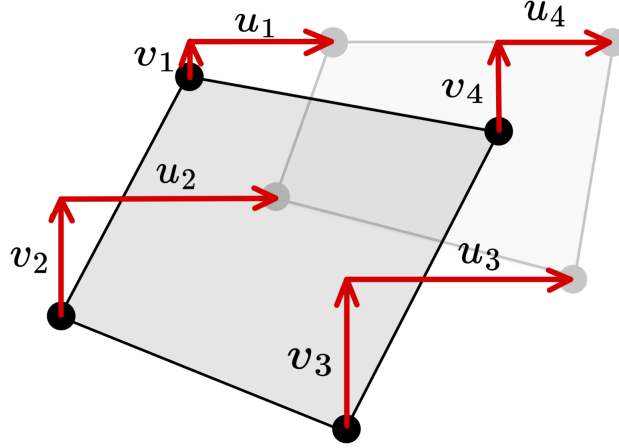
Egy kétdimenziós végeselemen megvizsgáljuk, hogy egy végeselemes szimuláció során mit is eredményez számunkra az anyagi nemlinearitás. Csak a főbb lépéseket említjük, a téma részletes kifejtéséről olvashatunk a [13] és [14] könyvekből. Ezekben azt is láthatjuk, hogy konkrét anyagmodellek végeselemes szimulációba történő integrálása miként is történik.

Egy végeselemes szimuláció során az a célunk, hogy a csomóponti elmozdulások és erők között teremtsünk kapcsolatot, ahol az elmozdulás ismert, ott az erőt kell tudnunk meghatározni, ahol pedig az erő, ott pedig az elmozdulást. A 2.1 ábrán látható kétdimenziós végeselem esetén a csomóponti elmozdulások segítségével definiálhatjuk a csomóponti elmozdulások vektorát az (2.1) egyenlet szerint.

$$\mathbf{U} = \begin{bmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & u_4 & v_4 \end{bmatrix}^T. \quad (2.1)$$

Továbbá a csomóponti erők segítségével definiálhatjuk a csomóponti erők vektorát az (2.2) egyenlet szerint.

$$\mathbf{F} = \begin{bmatrix} F_{1x} & F_{1y} & F_{2x} & F_{2y} & F_{3x} & F_{3y} & F_{4x} & F_{4y} \end{bmatrix}^T. \quad (2.2)$$



**2.1. ábra.** Végelem csomóponti elmozdulásai

Annak érdekében, hogy kapcsolatot teremthessünk a csomóponti erők és elmozdulások között szükséges felírnunk a szerkezeti egyensúly egyenletét, viszont a belső feszültségekből eredő erők kifejezéséhez szükségünk van az elemre vonatkozó alakváltozási és feszültség értékekre, melyek kifejezéséhez először a csomóponti elmozdulások és a megfelelő interpolációs mátrix segítségével előállítjuk az elem elmozdulásmezőjét

$$\mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\mathbf{U}. \quad (2.3)$$

Mely segítségével a megfelelő alakváltozás definíció és az ennek megfelelően definiált elmozdulás-alakváltozás mátrix definiálásával előállítható az alakváltozás vektor alakja

$$\boldsymbol{\varepsilon}(\mathbf{x}) = \mathbf{B}(\mathbf{x})\mathbf{U}. \quad (2.4)$$

Az alakváltozási vektort felhasználva a megfelelő konstitutív modell segítségével meg tudjuk adni a feszültség vektort. Ezt a kapcsolatot komplexebb modellek esetén akár egy közönséges differenciálegyenlet is definiálhatja, mely esetekben diszkretizálással tudjuk megoldani az egyenletet. Mivel a módszer megértéséhez elegendő, vegyünk egy olyan esetet amikor a feszültség csak az alakváltozás egy nemlineáris függvénye

$$\boldsymbol{\sigma}(\mathbf{x}) = \mathbf{f}(\boldsymbol{\varepsilon}(\mathbf{x})). \quad (2.5)$$

Ahogy a [13]-ben is olvashatjuk a szerkezet akkor van egyensúlyban, amikor a virtuális alakváltozási energia egyenlő az elemre ható külső erők virtuális munkájával, azaz

$$\iint_{\Omega} \delta \boldsymbol{\varepsilon}^T \boldsymbol{\sigma} \, d\Omega = \int_{\partial\Omega} \delta \mathbf{u}^T \mathbf{t} \, d\Gamma + \iint_{\Omega} \delta \mathbf{u}^T \mathbf{b} \, d\Omega. \quad (2.6)$$

Tudva, hogy esetünkben a jobb oldali kifejezés pusztán a külső csomóponti erők virtuális munkája, továbbá felhasználva a (2.4) egyenletben szereplő interpolációt, az (2.6) egyenlet átírható mint

$$\delta \mathbf{U}^T \left( \iint_{\Omega} \mathbf{B}^T \boldsymbol{\sigma} \, d\Omega = \mathbf{F} \right). \quad (2.7)$$

Mivel az egyensúly egyenletnek tetszőleges virtuális elmozdulásra teljesülnie kell, ezért az

$$\iint_{\Omega} \mathbf{B}^T \boldsymbol{\sigma} \, d\Omega = \mathbf{F} \quad (2.8)$$

egyenletnek kell teljesülnie, ahol a  $\boldsymbol{\sigma}$  a fenti egyenletek alapján látható, hogy egy nemlineáris függvénye csomóponti elmozdulásoknak. Tehát közelítve az integrált, az egyenlet átírható egy

$$\mathbf{P}(\mathbf{U}) = \mathbf{F} \quad (2.9)$$

alakú nemlineáris egyenletrendszerre. Látható, hogy anyagi nemlinearitást feltételezve, mindenképpen nemlineáris egyenletek megoldáshoz kell majd folyamodnunk. A nemlineáris egyenletrendszerek megoldására használt numerikus módszerek közül négyzetes konvergenciájával a Newton-Raphson iteráció az egyik legelterjedtebb módszer. A Newton-Raphson iterációról és további nemlineáris végelelemes szimulációk során használt módszerekről részletesen olvashatunk a [13]-ben.

### 2.1.1. Newton-Rhapson iteráció

A Newton-Rhapson iteráció ahogy a neve is utal rá nemlineáris egyenletek/egyenletrendszerek megoldására szolgáló iteratív algoritmus. Az általunk vizsgált végelelemes feladat esetén az adott elem deformálatlan konfigurációját felhasználva teszünk egy kezdeti becslést a csomóponti elmozdulásokra  $\mathbf{U}^{(0)}$ . Ezt követően ezt a becslést minden lépésben módosítjuk a (2.9) egyenletben szereplő  $\mathbf{P}(\mathbf{U})$  függvény linearizált alakját felhasználva.

$$\mathbf{P}(\mathbf{U}^{(i+1)}) = \mathbf{P}(\mathbf{U}^{(i)} + \Delta \mathbf{U}^{(i)}) \approx \mathbf{P}(\mathbf{U}^{(i)}) + \frac{\partial \mathbf{P}}{\partial \mathbf{U}}(\mathbf{U}^{(i)}) \Delta \mathbf{U}^{(i)}. \quad (2.10)$$

Fontos itt belátnunk, hogy amennyiben ezt a módszert szeretnénk használni, úgy mivel  $\mathbf{P}(\mathbf{U})$  a feszültség segítségével lett előállítva, mely pedig a konstitutív modellen keresztül függ az  $\mathbf{U}$  csomóponti elmozdulásoktól, ezért a

$$\mathbf{K}_T^{(i)} = \frac{\partial \mathbf{P}}{\partial \mathbf{U}}(\mathbf{U}^{(i)}) \quad (2.11)$$

meghatározásához ismernünk kell a konstitutív modell deriváltját is. Ez a neurális hálózatok esetén nem jelent problémát, ugyanis ezek tanítása is azon az alapfelvetésen alapszik, hogy a kimeneti réteget a bemeneti rétegből deriválható függvények segítségével állítja elő.

Most már csak az a kérdés, hogy miként válasszuk meg a  $\Delta \mathbf{U}^{(i)}$  értékeket a linearizálás során annak érdekében, hogy a kezdeti becslés egyre közelebb legyen a megoldáshoz. Ehhez a (2.10) összefüggés bal oldalát egyenlővé tesszük az elérni kívánt  $\mathbf{F}$ -el:

$$\mathbf{P}(\mathbf{U}^{(i)}) + \frac{\partial \mathbf{P}}{\partial \mathbf{U}}(\mathbf{U}^{(i)}) \Delta \mathbf{U}^{(i)} = \mathbf{F}. \quad (2.12)$$

Felhasználva az (2.11)-ben definiált érintő merevségi mátrix jelölést, majd rendezve az egyenletet egy lineáris egyenletrendszeret kapunk

$$\mathbf{K}_T^{(i)} \Delta \mathbf{U}^{(i)} = \mathbf{F} - \mathbf{P}(\mathbf{U}^{(i)}) := \mathbf{R}^{(i)}, \quad (2.13)$$

amely segítségével már a keresett  $\Delta \mathbf{U}^{(i)}$  számolható. Ezt a folyamatot egészen addig folytatjuk amíg a  $k$ . iterációnál a definiált  $\mathbf{R}^{(k)}$  maradvány vektor már kellően kicsi, ami azt jelenti, hogy az adott  $\Delta \mathbf{U}^{(k)}$  elmozdulások már kellően kielégítik a strukturális egyensúlyt.

Az iteratív mivoltából kifolyólag a Newton-Raphson iteráció akkor használható eredményesen, ha a kezdeti becslés közel van a keresett megoldáshoz. Mivel a kezdeti becslés általában a deformálatlan konfigurációból, azaz a nulla elmozdulásból indul, a Newton-Raphson-módszer akkor konvergál gyorsan a megoldáshoz, ha az alkalmazott terhelés kis elmozduláshoz vezet. A konvergencia a nagy elmozduláshoz vezető terhelések esetén válik nehézkessé.

További iteratív megoldók is léteznek, mint például az úgynevezett arc-length módszer, melyet olyan esetekben szokás alkalmazni, amikor a klasszikussá vált Newton-Raphson módszer nem alkalmazható.

### 2.1.2. Növekményes terhelés módszere

Egy megoldás a konvergencia segítésére a növekményes terhelés módszere, melyről szintén a [13]-ben olvashatunk. Ennek a módszernek a lényege, hogy a terhelést lépésekben alkalmazzuk. Tehát a csomóponti elmozdulások és erők egyaránt időfüggővé válnak

$$\mathbf{F}(t_n) := \mathbf{F}_n \quad \text{és} \quad \mathbf{U}(t_n) := \mathbf{U}_n. \quad (2.14)$$

A növekményes terhelés módszere esetén az egyes terhelési lépéseken belül az eljárás megegyezik a szokásos Newton-Raphson-módszerrel. A következő terhelési lépcsőfokot azután alkalmazzuk, hogy az előző terhelési lépcsőfokhoz tartozó megoldás már szükségszerűen konvergált. Ez a módszer azért jelent előnyt a konvergencia szempontjából, mert a  $t_{n+1}$  időpillanat Newton-Raphson iterációját a  $t_n$  időpillanathoz tartozó bekonvergált értékek segítségével inicializálhatjuk

$$\mathbf{U}_{n+1}^{(0)} = \mathbf{U}_n. \quad (2.15)$$

Ezt követően a fent bemutatott Newton-Raphson iteráció segítségével meghatározhatjuk a  $t_{n+1}$  időpillanat értékeit is.

## 2.2. Egydimenziós rugalmas-képlékeny anyagmodell

Egy egydimenziós példán keresztül vizsgáljuk meg azt is, hogy analitikusan miként modellezünk egy adott anyagi viselkedést, miként is néz ki egy konstitutív modell. Vegyünk egy olyan egyszerű egydimenziós rugalmas-képlékeny, anyagmodellt, amely egy nemlineáris, izotrop keményedéssel viselkedést mutat.

A rugalmas-képlékeny viselkedés alapfelvetése, hogy az alakváltozási sebesség felbontható rugalmas (visszafordítható) és képlékeny (maradandó) részekre.

$$\dot{\varepsilon} = \dot{\varepsilon}^e + \dot{\varepsilon}^p. \quad (2.16)$$

A rugalmas viselkedés leírása a Hooke törvénnyel történik

$$\dot{\sigma} = E\dot{\varepsilon}^e \quad \Rightarrow \quad \dot{\sigma} = E(\dot{\varepsilon} - \dot{\varepsilon}^p). \quad (2.17)$$

Az egyenértékű feszültség esetünkben nem más, mint a feszültség abszolútértéke, az egyenértékű képlékeny alakváltozási sebesség pedig a képlékeny alakváltozási sebesség abszolútértéke

$$\bar{\sigma} = |\sigma| \quad \text{és} \quad \dot{\bar{\varepsilon}} = |\dot{\varepsilon}^p|, \quad (2.18)$$

melyből az utóbbit felhasználva a felhalmozott egyenértékű képlékeny alakváltozás definiálható mint

$$\bar{\varepsilon} = \int_0^t \dot{\varepsilon} \, dt, \quad (2.19)$$

továbbá az előbbi segítségével pedig definiálható a folyási függvény mint

$$F = |\sigma| - Y, \quad (2.20)$$

ahol  $Y$  az aktuális folyáshatár. Ezek alapján a folyási feltétel megadható mint

$$F \leq 0. \quad (2.21)$$

Az egyenlőség esetén beszélünk képlékeny állapotról

$$F = 0 \quad \Rightarrow \quad |\sigma| = Y. \quad (2.22)$$

Az asszociatív folyási törvény a képlékeny alakváltozás sebességét úgy definiálja mint

$$\dot{\varepsilon}^p = \dot{\lambda} \frac{\partial F}{\partial \sigma} = \dot{\lambda} \cdot \text{sgn}(\sigma), \quad (2.23)$$

ahol a  $\dot{\lambda}$  a képlékeny multiplikátor és egyenlő az egyenértékű képlékeny alakváltozás sebességével  $\dot{\varepsilon}$ .

Nemlineáris, izotrop keményedés esetén a folyáshatár az egyenértékű képlékeny alakváltozás függvénye:

$$Y = Y(\bar{\varepsilon}), \quad (2.24)$$

melyeknek a sebességét a láncszabály segítségével megadhatjuk mint

$$\dot{Y} = \frac{\partial Y}{\partial \bar{\varepsilon}} \dot{\varepsilon} = H(\bar{\varepsilon}) \dot{\varepsilon} = H(\bar{\varepsilon}) \dot{\lambda}. \quad (2.25)$$

A konzisztencia feltétel segítségével kifejezhetjük a képlékeny multiplikátort:

$$\dot{F} = 0 \quad \Rightarrow \quad \dot{\lambda} = \frac{\text{sgn}(\sigma) E \dot{\varepsilon}}{E + H(\bar{\varepsilon})}. \quad (2.26)$$

Ez alapján már megadható a feszültség sebessége mint

$$\dot{\sigma} = \frac{E \cdot H(\bar{\varepsilon})}{E + H(\bar{\varepsilon})} \dot{\varepsilon}. \quad (2.27)$$

Itt fontos megjegyezni, hogy bár a formuláció során idő szerinti deriváltakat használtunk, ezt pusztán kényelmi okokból tettük. Ezen anyagmodell esetén nem beszélhetünk klasszikus értelemben vett alakváltozássebesség függésről.

### 2.3. Egydimenziós rugalmas-képlékeny anyagmodell numerikus megoldása

Most pedig vizsgáljuk meg, hogy miként is tudjuk a korábban bevezetett analitikus modellt megoldani numerikusan, ezáltal alkalmassá téve egy végeselemes szimuláció során való használatra. Ez itt bevezetett modellt a szakirodalom *radial return mapping algorithm* néven ismert. Diszkretizálva az (2.17) egyenletet a legalapvetőbb derivált közelítési formulával, a következő diszkrét egyenlethez jutunk:

$$\frac{\sigma_{n+1} - \sigma_n}{\Delta t} = E \cdot \left( \frac{\varepsilon_{n+1} - \varepsilon_n}{\Delta t} - \frac{\varepsilon_{n+1}^p - \varepsilon_n^p}{\Delta t} \right). \quad (2.28)$$

Ezt beszorozva  $\Delta t$ -vel majd átrendezve a következő alakhoz jutunk:

$$\sigma_{n+1} = \sigma_n + E\Delta\varepsilon_n - E \cdot \Delta\varepsilon_n^p = \sigma_{trial} - E \cdot \Delta\varepsilon_n^p, \quad (2.29)$$

ahol a képlékeny alakváltozás inkrementet a (2.23) egyenlet alapján közelíthetjük mint

$$\Delta\varepsilon_n^p = \Delta\lambda_n \cdot \text{sgn}(\sigma_{n+1}). \quad (2.30)$$

Ezek alapján láthatjuk, hogy

$$\text{sgn}(\sigma_{n+1}) = \text{sgn}(\sigma_{trial}), \quad (2.31)$$

mely egyenlőséget felhasználva jutunk ahhoz, hogy

$$\sigma_{n+1} = \sigma_{trial} - E \cdot \Delta\lambda_n \cdot \text{sgn}(\sigma_{trial}). \quad (2.32)$$

A képlékeny multiplikátor inkrement minden lépés esetén úgy számolható, ha felírjuk a folyási feltételt  $t_{n+1}$  helyen:

$$F_{n+1} = |\sigma_{trial}| - E \cdot \Delta\lambda_n + Y(\bar{\varepsilon}_n + \Delta\lambda_n) = 0. \quad (2.33)$$

Tehát egy időlépés megtétele általánosan a következőképpen fejezhető ki:

$$\begin{cases} \sigma_{n+1} = \sigma_n + f_1(\sigma_n, \bar{\varepsilon}_n, \varepsilon_{n+1}, \varepsilon_n) \\ \bar{\varepsilon}_{n+1} = \bar{\varepsilon}_n + f_2(\sigma_n, \bar{\varepsilon}_n, \varepsilon_{n+1}, \varepsilon_n) \end{cases}. \quad (2.34)$$

Mely általános alakból látszik, hogy ez az anyagmodell matematikai szempontból valójában egy diszkrét dinamikai rendszerként viselkedik, amelyet a bevezetésben a (1.2) egyenlettel is előrevetítettünk. Ez az az általánosítás, amelyet a [4]-ben is bevezettek.

### 3. A probléma általánosítása

Most már tudjuk azt, hogy milyen szerepet tölt be számunkra az anyagmodell egy végeselemes szimulációban. Az anyagmodell egy matematikai értelemben vett diszkrét dinamikai rendszerként egy alakváltozás idősor ismeretében állít elő egy feszültség idősor választ. Most annak érdekében, hogy még általánosabb konklúzióra jussunk, vegyünk példának egy olyan anyagi viselkedést, ami szintén alakváltozás vezérelt, közönséges differenciálegyenlet alakban van megadva, valamint nem csak az alakváltozástól, hanem az alakváltozás sebességétől is függ:

$$\dot{\sigma} = \gamma(\sigma, \varepsilon, \dot{\varepsilon}). \quad (3.1)$$

Most az előző példától eltérően az idő szerinti deriváltakat közelítsük egy magasabb rendű integrátor formula segítségével:

$$\dot{x}_n = \frac{x_{n+1} - x_{n-1}}{2\Delta t}. \quad (3.2)$$

mely segítségével közelítve  $\dot{\varepsilon}$ -t és  $\dot{\sigma}$ -t az az (3.1) egyenletben, majd ezt rendezve a következőhöz jutunk:

$$\sigma_{n+1} = \sigma_{n-1} + 2\Delta t \cdot \gamma\left(\sigma_n, \varepsilon_n, \frac{\varepsilon_{n+1} - \varepsilon_{n-1}}{2\Delta t}\right). \quad (3.3)$$

Mivel nem feltételeztünk direkt időfüggést, ezért  $\Delta t$  paraméternek tekinthető, ezért a

$$f(\sigma_n, \varepsilon_{n+1}, \varepsilon_n, \varepsilon_{n-1}) := 2\Delta t \cdot \gamma\left(\sigma_n, \varepsilon_n, \frac{\varepsilon_{n+1} - \varepsilon_{n-1}}{2\Delta t}\right) \quad (3.4)$$

függvénydefinícióval általánosíthatjuk a diszkretizált alakot mint

$$\sigma_{n+1} = \sigma_{n-1} + f(\sigma_n, \varepsilon_{n+1}, \varepsilon_n, \varepsilon_{n-1}), \quad (3.5)$$

melyet még tovább általánosíthatunk, amennyiben definiáljuk a

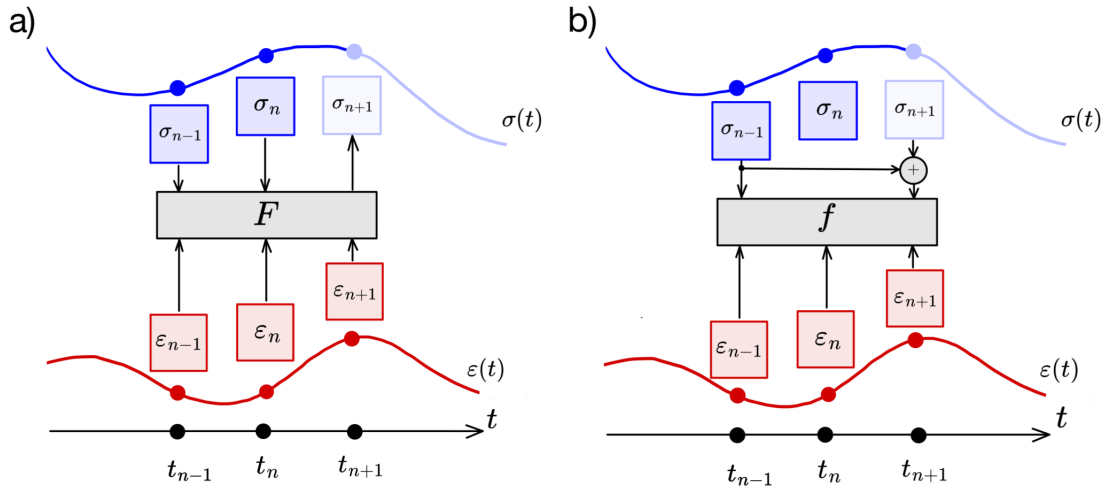
$$F(\sigma_n, \sigma_{n-1}, \varepsilon_{n+1}, \varepsilon_n, \varepsilon_{n-1}) = \sigma_{n-1} + f(\sigma_n, \varepsilon_{n+1}, \varepsilon_n, \varepsilon_{n-1}) \quad (3.6)$$

függvényt. Így a még általánosabb alak

$$\sigma_{n+1} = F(\sigma_n, \sigma_{n-1}, \varepsilon_{n+1}, \varepsilon_n, \varepsilon_{n-1}). \quad (3.7)$$

Ehhez hasonló elnevezéseket a későbbiekben is használni fogunk, ugyanis neurális hálózatok szempontjából számunkra az a fontos, hogy a változó mennyiségek miként függenek egymástól.

A 3.1 ábrán  $k = 1$ -re láthatjuk a két fentebb általánosított alakot.



**3.1. ábra.** Alakváltozás-sebességétől függő anyagmodell diszkrét másodrendű alakjai, ahol a) a (3.5) egyenlet megfelelője, míg b) a (3.7) egyenleté

Egy általános  $(k + 1)$  rendű integrátor formulát használva a következőképpen diszkrétizálható az egyenlet

$$\sigma_{n+1} = \sum_{i=0}^k a_i \cdot \sigma_{n-i} + (k+1)! \cdot \Delta t \cdot \gamma \left( \sigma_n, \varepsilon_n, \frac{\varepsilon_{n+1} - \sum_{i=0}^k a_i \cdot \varepsilon_{n-i}}{(k+1)! \cdot \Delta t} \right), \quad (3.8)$$

ahol az  $[a_i]_{i=0}^k$  egész együtthatókat a használt integrátor formula adja meg és minden esetben igaz rájuk, hogy

$$\sum_{i=0}^k a_i = 1. \quad (3.9)$$

Ez természetesen - mivel  $\Delta t$  paraméternek tekinthető - a (3.5) egyenlethez hasonlóan általánosítható mint

$$\sigma_{n+1} = \sum_{i=0}^k a_i \cdot \sigma_{n-i} + f(\sigma_n, \dots, \sigma_{n-k}, \varepsilon_{n+1}, \varepsilon_n, \dots, \varepsilon_{n-k}), \quad (3.10)$$

melyet a (3.7) egyenlethez hasonlóan általánosíthatunk mint

$$\sigma_{n+1} = F(\sigma_n, \dots, \sigma_{n-k}, \varepsilon_{n+1}, \varepsilon_n, \dots, \varepsilon_{n-k}). \quad (3.11)$$

Természetesen minél magasabb rendű integrátor formulát használunk, annál több kezdeti  $\sigma_i$  érték megadására lesz szükségünk ahhoz, hogy meg tudjuk adni a rendszer megoldását. Szám szerint mindig pontosan annyira, ahányad rendű az integrátor formula. Tehát egy fentihez hasonló  $(k+1)$ . rendű esetben szükségünk lesz  $(\sigma_0, \sigma_2, \dots, \sigma_k)$ -re. Ezek segítségével a  $(k+1)$ . feszültség kifejezhető ismert változók segítségével mint

$$\sigma_{k+1} = F(\sigma_k, \dots, \sigma_0, \varepsilon_{k+1}, \varepsilon_k, \dots, \varepsilon_0). \quad (3.12)$$

Egy lépést megtéve, azaz a (3.12) összefüggést behelyettesítve a  $(k+2)$  pillanat béli feszültséget kifejező egyenletbe a következőt írhatjuk, hogy

$$\begin{aligned} \sigma_{k+2} &= F(\sigma_{k+1}, \sigma_k, \dots, \sigma_1, \varepsilon_{k+2}, \varepsilon_{k+1}, \dots, \varepsilon_1) \\ \sigma_{k+2} &= F(F(\sigma_k, \dots, \sigma_0, \varepsilon_{k+1}, \varepsilon_k, \dots, \varepsilon_0), \sigma_k, \dots, \sigma_1, \varepsilon_{k+2}, \varepsilon_{k+1}, \dots, \varepsilon_1). \end{aligned} \quad (3.13)$$

Innen egy a korábbiakhoz hasonló elnevezéssel írhatjuk, hogy

$$\sigma_{k+2} = F_{k+1}(\sigma_k, \dots, \sigma_0, \varepsilon_{k+2}, \varepsilon_{k+1}, \dots, \varepsilon_0). \quad (3.14)$$

Induktívan beláthatjuk, hogy ily módon definiált

$$F_{n+1}(\sigma_k, \dots, \sigma_0, \varepsilon_{n+1}, \dots, \varepsilon_0) := F(F_n(\dots), \sigma_k, \dots, \sigma_0, \varepsilon_{n+1}, \dots, \varepsilon_0). \quad (3.15)$$

függvénysor segítségével az  $(n+1)$  feszültség érték kifejezhető a kezdeti feszültség értékek, valamint az alakváltozás értékek múltjának segítségével

$$\sigma_{n+1} = F_n(\sigma_k, \dots, \sigma_0, \varepsilon_{n+1}, \varepsilon_n, \dots, \varepsilon_0) \quad , \quad n > k+1. \quad (3.16)$$

A fentiek alapján célszerű lehet számunkra az alapfelvetésünket kiegészíteni azzal, hogy az idősorok közti kapcsolatot úgy modellezzük, hogy a befolyásolt jellemző  $(n+1)$  értékét

a vezérlő jellemző  $(n + 1)$  értékével, valamint egyaránt a befolyásolt és vezérlő jellemző korábbi értékeivel igyekszünk kifejezni.

$$\left. \begin{array}{c} \varepsilon_{n+1} \\ \varepsilon_n, \dots, \varepsilon_{n-k} \\ \sigma_n, \dots, \sigma_{n-k} \end{array} \right\} \xrightarrow{\text{model}} \sigma_{n+1}.$$

Illetve, ha olyan modellt választunk, amely alapvetően rendelkezik valamilyen belső memóriával, az idősorok közti kapcsolatot úgy is modellezhetjük, hogy a befolyásolt jellemző  $(n + 1)$ . értékét a vezérlő jellemző  $(n + 1)$ . időpillanat béli és múltbéli értékeivel igyekszünk kifejezni.

$$\left. \begin{array}{c} \varepsilon_{n+1} \\ \varepsilon_n, \dots, \varepsilon_{n-k} \end{array} \right\} \xrightarrow{\text{model}} \sigma_{n+1}.$$

## 4. Lehetséges megoldások

Láthattuk, hogy milyen szerepet játszik az anyagmodell egy VEM szimulációban, illetve azt is, hogy ezt a szerepet miként is tölti be. Láttuk, hogy az analitikus anyagmodellek valójában matematikai értelemben vett diszkrét dinamikai rendszerekként valósíthatóak meg egy szimuláció során.

A későbbiekben megpróbálunk adatalapon, neurális hálózatok segítségével olyan diszkrét dinamikai rendszert alkotni, ami modelljéül tud szolgálni egy "ismeretlen" anyagi viselkedésnek. Ez a gyakorlatban több szempontból is előnyös lehet, hiszen az anyagok összetettségéből kifolyólag az anyagok viselkedése erős nemlinearitást mutat, ezért többnyire az analitikus megközelítések is csak korlátozott körülmények között és kötött feltételek mentén alkalmazhatóak. Viszont az adatalapú megoldások legnagyobb előnye, hogy megfelelő mennyiségű és minőségű adat alapján képesek nemlineáris és komplex kapcsolatok modellezésére is.

Közelítendő anyagi viselkedésnek jelen esetben célszerű egy analitikus modell által leírt viselkedést venni, ugyanis ez is elegendő az adatalapú anyagmodellezés módszerének validálásához, viszont lehetőséget biztosít mérések nélkül, pusztán szimuláció által korlátlan mennyiségű feszültség-alakváltozás idősorpárt létrehozására, ezáltal a modell tanításához és annak a teszteléséhez szükséges adatmennyiség is adott.

## 4.1. Példa választása

A fenti megfontolások mentén a 2.3 fejezetben bemutatott nemlineáris keményedési modellt fogjuk megpróbálni adatalapon modellezni.

Mindenek előtt meg kell választanunk a  $Y(\bar{\varepsilon})$  függvényt, amely megadja folyáshatárt. Mi most ennek egy nemlineáris Swift modellt fogunk választani, ahol a folyási feszültség változását a

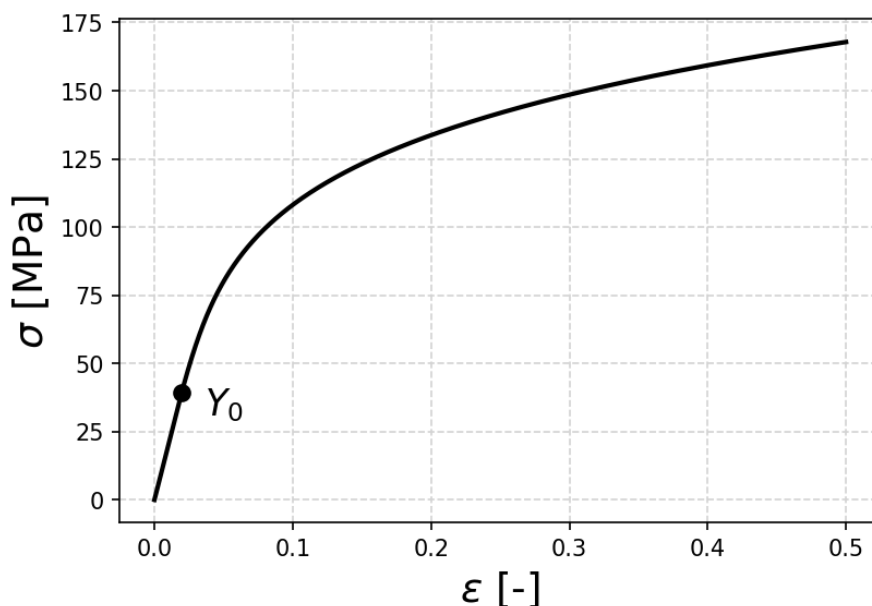
$$Y(\bar{\varepsilon}) = a \cdot (\varepsilon_0 + \bar{\varepsilon})^n \quad (4.1)$$

következő nemlineáris függvény adja meg. Emellett a választott anyagmodell paramétereit is rögzítenünk kell, ezzel konkretizálva az magát az anyagot. A Swift modell paramétereit mellett megválasztjuk a rugalmas-képlékeny anyagmodell rugalmassági modulusát is. Innentől a 4.1 táblázatban megadott paraméterekkel megadott anyaggal fogunk dolgozni.

**4.1. táblázat.** Anyag paramétereinek megválasztása.

| $E$ [Pa]        | $a$ [Pa]          | $\varepsilon_0$ [-] | $n$ [-] |
|-----------------|-------------------|---------------------|---------|
| $2 \times 10^9$ | $200 \times 10^6$ | 0.0003              | 0.2     |

Melynek nemlineáris karakterisztikáját megfigyelhetjük, ha ábrázoljuk azt a megszokott  $\varepsilon - \sigma$  koordinátarendszerben.



**4.1. ábra.** A választott rugalmas-képlékeny modell vizuális megjelenítése, ahol a kezdeti folyáshatárt a görbén az  $Y_0 = a\varepsilon_0^n$  jelöli.

## 4.2. Adat generálás

A korábban vázolt rendszert könnyedén meg tudjuk valósítani egy programnyelv segítségével, ezáltal lehetőségünk nyílik számtalan feszültség-alakváltozás idősorpár generálására. A rendszer egy Python nyelvű megvalósítása a következőképpen fest:

```
1 E = 2e9
2
3 def Y_F(x):
4     a, eps_0, n = 200e6, 0.0003, 0.2
5     return a*(eps_0+x)**n
6
7 def simulate(eps,E,Y_F):
8     sig = np.zeros_like(eps)
9     _lambda = np.zeros_like(eps)
10    Y = np.ones_like(eps)*Y_F(0)
11
12    for i in range(eps.shape[0]-1):
13
14        deps = (eps[i+1] - eps[i])
15
16        if abs(sig[i] + E*deps) - Y[i] <= 0:
17            # elastic
18            dlambd = 0
19        else:
20            # plastic
21            def eq(dlambd):
22                ret = abs(sig[i] + E*deps)
23                ret -= E*dlambd
24                ret -= Y_F(_lambda[i] + dlambd)
25                return ret
26            dlambd = root(eq,1e-8).x
27
28            # update
29            _lambda[i+1] = _lambda[i] + dlambd
30            Y[i+1] = Y_F(_lambda[i+1])
31            sig[i+1] = sig[i] + E*(deps - dlambd*sign(sig[i]+E*deps))
32
33    return sig
34
```

Az itt látható függvény megkapva az anyag tulajdonságait (Young modulus  $E$  és folyási függvény  $Y_F$ ) és egy tetszőleges alakváltozás idősort  $eps$ , majd ezek alapján megadja számunkra a kívánt feszültség idősort  $sig$ .

Egy adat alapú modell alkotásához viszonylag sok változatos és ugyanakkor életszerű

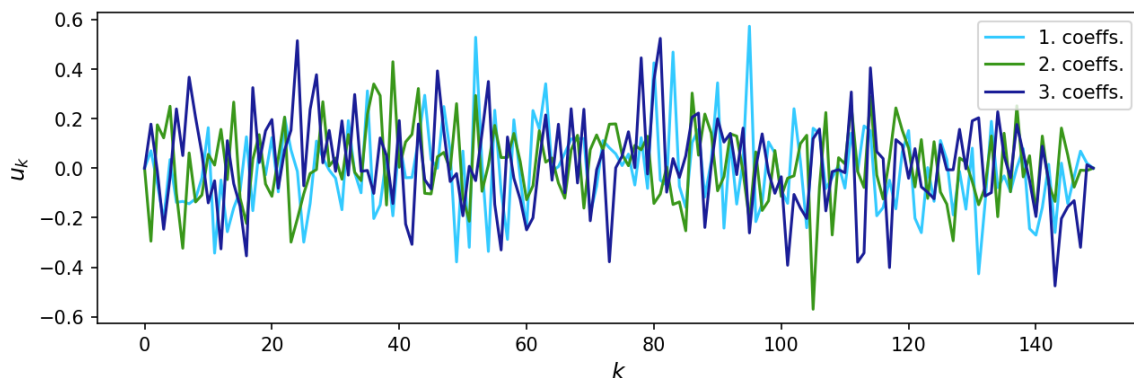
idősor párra lesz szükségünk, tehát olyan módon kell véletlenszerűen bemeneti alakváltozás idősorokat generálni, hogy azok valóságos, elképzelhető eseteket tükrözzenek.

Nulladik megfontolásunk, hogy a kezdeti alakváltozás mindegyik sor esetében 0 kell legyen. Ezért ezt kézenfekvő lehet különböző frekvenciájú szinusz függvények összegeként megadni, amelyek együtthatói véletlenszerűen választhatóak, ezáltal biztosítva a véletlenszerűséget.

Vegyünk egy általános, tisztán szinuszos,  $T$  lépésből álló, diszkrét Fourier sort, mely  $N$  frekvenciából tevődik össze. Ennek együtthatói legyenek  $[a_k]_{k=1}^N$ , melyek véletlenszerűen megválaszthatóak, ekkor az idősor tagjait a következőképpen lehet előállítani:

$$\varepsilon_n = \sum_{k=1}^N a_k \cdot \sin\left(2\pi n \frac{k}{N}\right), \quad n \in \overline{1, T} \quad (4.2)$$

Válasszuk mondjuk az első  $N = 300$  tagot, illetve vegyünk mondjuk egy  $T = 150$  hosszú idősort és nézzünk meg, hogy többször véletlenszerűen megválasztva az  $[a_k]_{k=1}^N$  együtthatókat milyen jellegű idősorokat kapunk. Az így generálódott idősorokat a 4.2 ábrán láthatjuk. Látható, hogy kellően véletlenszerűek ugyanakkor annál kevésbé tükröznek egy valószerű terheléstörténetet, hiszen a magas frekvenciák jelentősen érvényesülnek. Ez könnyen orvosolható, ha módosítjuk a generálási módot mint



**4.2. ábra.** Fourier soros bemenőjel generálás.

$$\varepsilon_n = \sum_{k=1}^N \frac{a_k}{k^p} \cdot \sin\left(2\pi n \frac{k}{N}\right), \quad n \in \overline{1, T}. \quad (4.3)$$

Emellett az életszerűséghez még az is közrejátszik, ha előírjuk, hogy a nagy frekvenciájú tagok körülbelül első periódusa képezze csak részét az idősornak. A legnagyobb frekvenciájú tag az  $n = 1$  értékhez tartozik, melynek akkor lesz egy periódusa az idősorban, ha  $T = N$ . Ezért a generálási paraméterek megválasztásánál  $N \approx T$ -re fogunk törekedni.

Miután adott egy véletlenszerű diszkrét idősor, fontos az is, hogy meg tudjuk választani ennek a maximum értékét, ugyanis alakváltozásokra relatív kis értékek a jellemzők.

A fenti megfontolások alapján összeállított idősor generálási mód látható a következőkben:

```

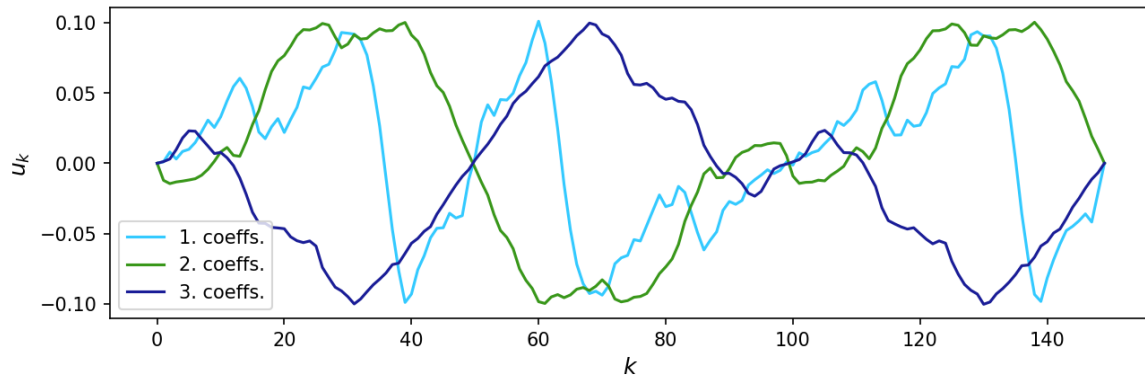
1 def generate_random_signal(T,N,p,eps_max):
2     n = np.linspace(0, T, T)
3     a = np.random.randn(N)
4     eps = np.zeros_like(n)
5     for k in range(1, N + 1):
6         eps += (a[k - 1] / k**p) * np.sin(2*np.pi*n*k/N)
7     # normalize
8     eps = eps/(np.max(eps) - np.min(eps))*2*eps_max
9     return eps
10

```

Most pedig válasszunk meg a paramétereket és nézzünk meg pár véletlen generálást, hogy milyen bemenőjeleket eredményeznek.

| $T$ | $N$ | $p$ | $\varepsilon_{max}$ |
|-----|-----|-----|---------------------|
| 150 | 100 | 1.5 | 0.2                 |

**4.2. táblázat.** Véletlenjel-generálás megválasztott paraméterei.

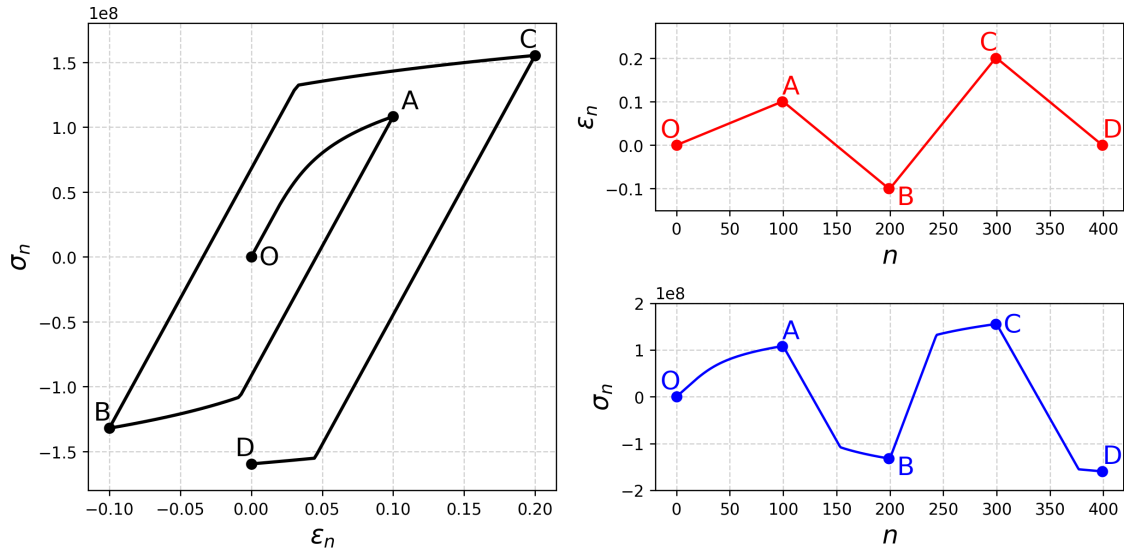


**4.3. ábra.** Korrigált Fourier soros bemenőjel generálás.

Az ábrán remekül látható, hogy a jelek kellően véletlenszerűek, illetve ugyanakkor egy életszerű terheléstörténet karakterisztikájával bírnak. Ezért ez a bemenőjel generálási mód alkalmas arra, hogy tanítóadatként szolgáljon majd a jövőben egy adatalapú modell számára. A tanítás során ezzel a módszerrel és a fent látható paraméterekkel lett készítve 200 bemenő alakváltozás idősor, illetve ebből a korábbi **simulate** függvénnyel meg lett határozva a rendszer által válaszul adott 200 feszültségválasz.

Ezt a generált 200 idősor-párt cél szerint felosztottam tanító (165), validációs (25) és teszt (10) halmazokra. A tanító és validációs adatok a modell létrehozását szolgálják, míg a teszt adatok pedig a háló (tanító adatoktól független) kiértékelését.

Ezen felül további kiértékelési céllal létre fogok hozni egy teljesen független, manuálisan generált benchmark idősor-párt is, ugyanis a tesz halmazban szereplő sorok bár nem szerepelnek a tanítóadatok között, azonos generálási móddal készültek és hosszban is megegyeznek. A benchmark sor esetén sem a hossz, sem a létrehozás nem fog semmilyen formában hasonlítani a tanító adatokra, így tökéletes metrika lesz a modellek végleges kiértékelésére.



4.4. ábra. Benchmark idősor-pár.

A modell alkotás során mindegyik modell típust és architektúrát ezekkel az adatokkal fogom tanítani, valamint kiértékelni, ezzel alapot adva egy konzisztens összehasonlításnak.

### 4.3. Modell alkotás

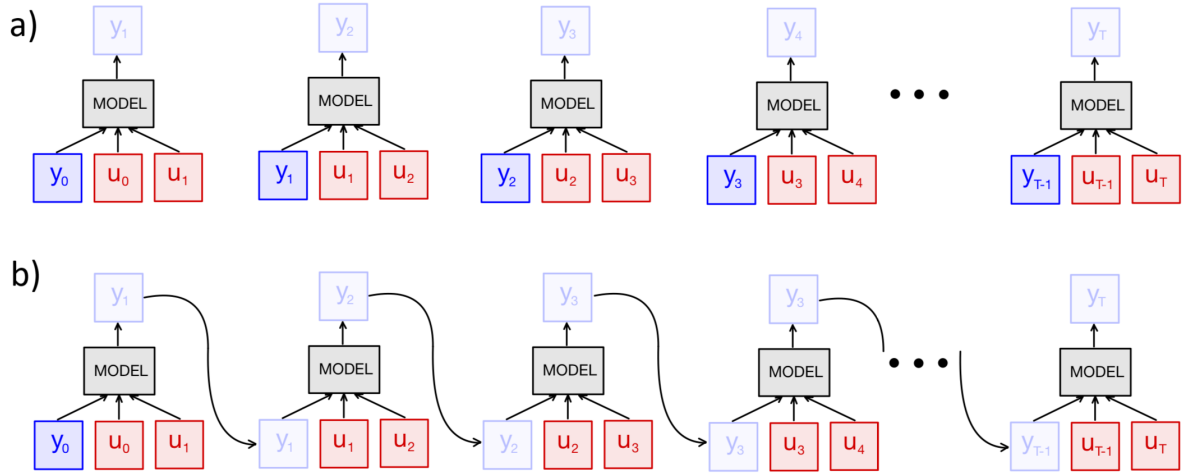
A továbbiakban az egyszerűség kedvéért ragaszkodjunk az irányításelméletből ismert jelölésekhez: bemenő jel  $u_n = \varepsilon_n$  és kimenő jel  $y_n = \sigma_n$ .

A 3. fejezetben is láthattuk, hogy több lehetőségünk is van az ilyen rendszerek modellezésére. Egyaránt modellezhetjük a feszültség inkrementet, vagy akár a feszültséget magát is, emellett a modell bemenete egyaránt lehet a pillanatnyi alakváltozás és az alakváltozás meghatározott hosszúságú múltja, vagy pedig lehet a pillanatnyi alakváltozás, valamint az alakváltozás és feszültség meghatározott hosszúságú múltja. Ezalapján ez a következő négy lehetőséget fogjuk vizsgálni:

$$y_{n+1} = \begin{cases} y_n + \begin{cases} f(y_n, \dots, y_{n-k}, u_{n+1}, u_n, \dots, u_{n-k}) \\ f_n(u_{n+1}, u_n, \dots, u_{n-k}) \end{cases} \\ F(y_n, \dots, y_{n-k}, u_{n+1}, u_n, \dots, u_{n-k}) \\ F_n(u_{n+1}, u_n, \dots, u_{n-k}) \end{cases} . \quad (4.4)$$

Fontos megjegyzés, hogy az inkrement modellezés ( $f$  és  $f_n$ ) megközelítése nagyon hasonlatos a [10] cikkben bevezetett nODE (Neural Ordinary Differential Equations) alapkoncepciójához, mely mentén elindulva, akár még több lehetőségünk nyílik dinamikai rendszerek modellezésére és ezáltal az anyagokéra is. Ezt az irányt a dolgozat részeként nem vizsgáljuk, viszont a későbbiekben mindenképp célszerű ilyen irányban is próbálkozni.

Fontos továbbá azt is megjegyezni, hogy mi nem predikációs, hanem szimulációs modellt szeretnénk tanítani, ugyanis számunkra a modell horizontjának megfelelő mennyiségű kezdeti feszültség értéket leszámítva ismeretlenek a feszültségállapotok, ezért a kezdeti értékeken túl mindig a modell által korábban számolt feszültség értékek fognak a modell bemeneteként szolgálni. Vegyünk mondjuk egy  $y_{n+1} = F(y_n, u_n, u_{n+1})$  esetet, ekkor a szimulációs és predikációs modell közti különbséget vizuálisan a 4.5 ábra szemlélteti.



**4.5. ábra.** a) Predikációs modell VS b) Szimulációs modell

Az ábrán látható, hogy a szimulációs modell esetén mindig a háló által korábbi időpillanatokban számolt kimenete szolgál a következő időpillanatban a háló bemeneteként, míg a predikációs modell esetén ezek mindig ismertnek tekinthetők. Innen az is látható, hogy a szimulációs modell esetén a hibák egy szimuláció során összeadódnak, ezért viszonylag pontos modellre lesz szükségünk. Ezért egy kézenfekvő megoldás lehet a neurális hálózatok használata, ugyanis a legalapvetőbb hálók is képesek erősen nemlineáris kapcsolatokat viszonylag gyorsan és pontosan megtanulni. Az irodalom alapján beláthatjuk, hogy erre a feladatra eredményesen használható a legalapvetőbb MLP (Multi Layer Perceptron) hálózatok is ezért első körben ilyen

irányban indulunk el. Láthattuk a feladat jellegéből, illetve az irodalomban szintén van erre példa, hogy a rekurens neurális hálózatok is jól működnek, ezért a másik irány amivel próbálkozunk azok az LSTM (Long-Short Term Memory) hálózatok, amelyeket időben változó adatok modellezésére fejlesztettek ki, ezáltal alapértelmezetten rendelkezik egyfajta memória jelleggel. Diszkrét dinamikai rendszerek LSTM és MLP hálózatokkal történő modellezésének lehetőségeiről és korlátairól olvashatunk az [12] cikkben.

#### 4.4. MLP hálózatok

A Multi-Layer Perceptron hálózatoknak nevezik a legalább három teljesen összekapcsolt neuron réteget számláló előre csatolt neurális hálózatokat, ahol egy adott réteg neuronjait mindig az előző réteg neuronjai segítségével állítjuk elő:

$$\mathbf{h}^{(i+1)} = \sigma \left( \mathbf{W}^{(i)} \mathbf{h}^{(i)} + \mathbf{b}^{(i)} \right), \quad (4.5)$$

ahol  $\mathbf{W}^{(i)}$  a réteghez tartozó súlymátrix,  $\mathbf{b}^{(i)}$  a réteghez tartozó bias vektor, valamint  $\sigma$  a nemlinearitást adó aktivációs függvény. A háló tanítása azt rejti magában, hogy ezeket a súlymátrixokat és a bias vektorokat egy backpropagation nevezetű algoritmus segítségével úgy hangoljuk, hogy a háló illeszkedjen az általunk definiált adatokra. Számunkra ezek jelentik a szabad paramétereket és ezek határozzák meg számunkra a háló méretét és ebből kifolyólag annak a gyorsaságát is. Fontos, hogy minél kevesebb paraméterrel rendelkező hálóra törekedjünk, ugyanis tudjuk, hogy mivel egyetlen időlépést modellezünk a bonyolultság nem lehet túl nagy, illetve mivel VEM szimuláció során szeretnénk alkalmazni a sebesség is fontos szempont.

Mivel ezek a hálózatok nem rendelkeznek memória jelleggel, ezért belátható, hogy a (4.4) modellezési módok közül azokat tudjuk eredményesen használni, ahol az elkövetkezendő kimenő jel nem csak a bemenőjelek múltjától, hanem a korábbi kimenőjelek múltjától is függ. Ezt a [12] eredményi is alátámasztják. Ebből kifolyólag MLP hálózatok esetén a következő két modellezési módszerrel fogunk próbálkozni:

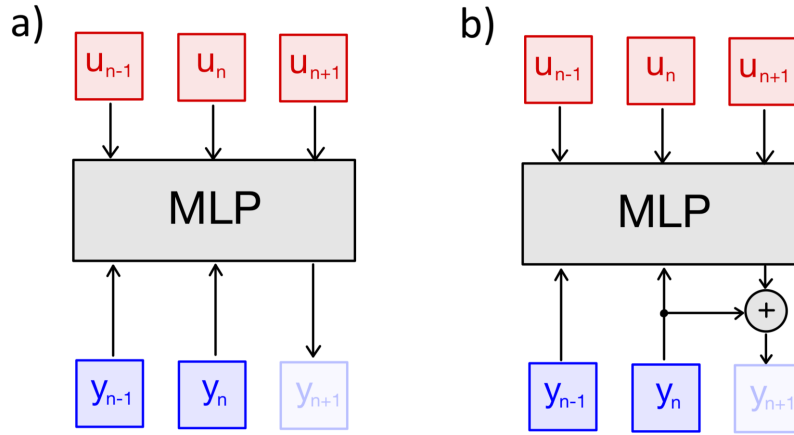
1. Direkt modellezés:

$$y_{n+1} = F(y_n, \dots, y_{n-k}, u_{n+1}, \dots, u_{n-k}). \quad (4.6)$$

2. Inkrementális modellezés:

$$y_{n+1} = y_n + f(y_n, \dots, y_{n-k}, u_{n+1}, \dots, u_{n-k}). \quad (4.7)$$

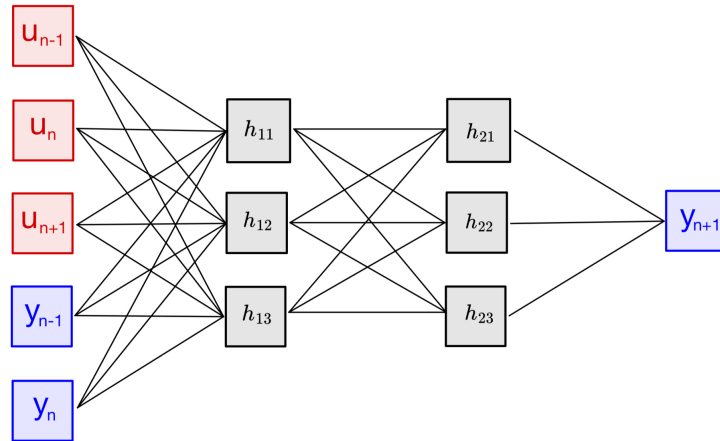
Mind a két modellezési lehetőséget a könnyebb értelmezhetőség kedvéért  $k = 1$  értékre az 4.6 ábrán ábrázoltuk.



**4.6. ábra.** MLP modellezési lehetőségek: a) Direkt modellezés, valamint b) Inkrementális modellezés.

Annak érdekében, hogy a lehető legjobb pontosságot érjük el, ezért mindkét esetben meg fogunk vizsgálni több háló struktúrát, mindegyikkel be fogunk tanítani egy modellt és megnézzük, melyik hogyan teljesít. Ezért paraméterezhetővé tesszük a rétegek számát  $q$ , illetve a rétegek neuronjainak a számát  $p$ .

A 4.7 ábrán egy rögzített  $p = 3$  és  $q = 2$ , valamint  $y_{n+1} = F(y_n, y_{n-1}, u_{n+1}, u_n, u_{n-1})$  esetre láthatjuk, hogy miként is fest egy MLP struktúra.



**4.7. ábra.** Példa MLP struktúrára.

Most annak érdekében, hogy megértsük a háló összpáraméter számának számolásának módját, fejtsük ki, hogy ezen konkrét példa esetén, hogyan is fest a különböző rétegek számolásának módja:

$$\begin{bmatrix} h_1^{(1)} \\ h_2^{(1)} \\ h_3^{(1)} \end{bmatrix} = \sigma \left( \begin{bmatrix} W_{11}^{(0)} & W_{12}^{(0)} & W_{13}^{(0)} & W_{14}^{(0)} & W_{15}^{(0)} \\ W_{21}^{(0)} & W_{22}^{(0)} & W_{23}^{(0)} & W_{24}^{(0)} & W_{25}^{(0)} \\ W_{31}^{(0)} & W_{32}^{(0)} & W_{33}^{(0)} & W_{34}^{(0)} & W_{35}^{(0)} \end{bmatrix} \begin{bmatrix} u_{n+1} \\ u_n \\ u_{n-1} \\ y_n \\ y_{n-1} \end{bmatrix} + \begin{bmatrix} b_1^{(0)} \\ b_2^{(0)} \\ b_3^{(0)} \\ b_4^{(0)} \\ b_5^{(0)} \end{bmatrix} \right) \quad (4.8)$$

$$\begin{bmatrix} h_1^{(2)} \\ h_2^{(2)} \\ h_3^{(2)} \end{bmatrix} = \sigma \left( \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \end{bmatrix} \begin{bmatrix} h_1^{(1)} \\ h_2^{(1)} \\ h_3^{(1)} \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix} \right) \quad (4.9)$$

$$y_{n+1} = \sigma \left( \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} \end{bmatrix} \begin{bmatrix} h_1^{(2)} \\ h_2^{(2)} \\ h_3^{(2)} \end{bmatrix} + b_1^{(2)} \right) \quad (4.10)$$

A fenti példa esetén jól láthat a rétegekhez tartozó kifejtett súlymátrixok és bias vektorok alapján, hogy a háló összparaméter száma megadható mint

$$n_{params} = (5 \cdot 3 + 3) + (3 \cdot 3 + 3) + (3 \cdot 1 + 1) = 34. \quad (4.11)$$

A fentiek alapján látható, hogy a hálózat paramétereinek száma általános esetben a  $k$ ,  $p$  és  $q$  értékektől fog függeni:

$$n_{params} = (2k + 3) \cdot p + p + (q - 1) \cdot (p \cdot p + p) + (p \cdot 1 + 1). \quad (4.12)$$

Ahhoz, hogy meg tudjuk sejteni az ideális struktúrát meg fogjuk vizsgálni a  $(k + 1)$ ,  $p$  és  $q$  értékek egymástól független hatását a pontosságra. Ennek érdekében az  $A = \{2, 3, 5, 8\}$  halmaz értékei közül szerre választjuk mindhárom paramétert és minden esetre létrehozok egy modellt, tehát a  $(k + 1, p, q) \in A \times A \times A$  eseteket fogom megvizsgálni. Ami azt jelenti, hogy mindkét modellezési mód esetében  $|A \times A \times A| = 4^3 = 64$  modellt fogok betanítani és kiértékelni. Minden esetben  $n_{epochs} = 100$  alaklommal iteráltam végig a tanító adatokon. Ezután az így létrejött modelleket kiértékeltem a korábban említett 10, fejenként 150 időlépésből álló teszt halmazon, illetve a manuálisan generált 400 lépésből álló benchmark halmazon. A hibát  $MSE$  (Mean Squared Error) és  $MAE$  (Mean Absolute Error) mértem:

$$MSE = \sum_{n=1}^T \frac{(y_n - \hat{y}_n)^2}{T} \quad \text{és} \quad MAE = \sum_{n=1}^T \frac{|y_n - \hat{y}_n|}{T}, \quad (4.13)$$

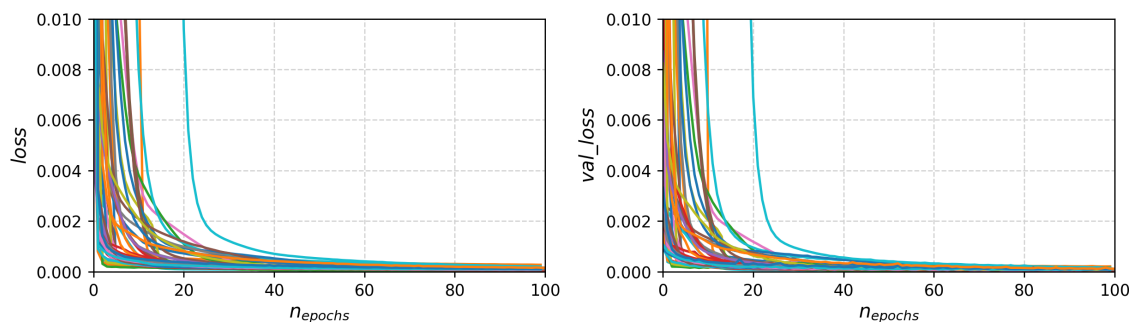
ahol  $T$  az időinkrementek száma,  $y_n$  a valós kimenet  $n$ . időpillan béli értéke és  $\hat{y}_n$  a háló

által adott kimenet  $n$ . időpillan béli értéke.

Itt nagyon fontos megjegyezni, amit korábban is említettünk, hogy szimulációs modellt szeretnénk létrehozni, ezért a későbbiekben használt hibamértékek közvetlen nem a háló pontosságára vonatkoznak, ugyanis a háló hibája az ismert múlt és a vezérlő jellemző függvényében adott predikciók és a valós értékek közti eltérés, viszont minket a kezdeti értékek és a vezérlő jellemző függvényében a 4.5 ábrán látható módon szimulált idősor és a valós idősor közti eltérés érdekel. Tehát ezek a hibaértékek már magukban hordozzák a háló hibájának a szimuláció során történő felhalmozódását is.

#### 4.4.1. Direkt modellezés

Mielőtt kiértékelnénk a 64 modellt, először is meg szeretnénk bizonyosodni arról, hogy a konvergencia megtörtént. Ehhez a tanítás során definiált veszteség értékeket fogom használni, melyet esetenben az  $MSE$  érték képvisel. Akkor tekintjük úgy, hogy elértük a szükséges konvergenciát, ha a tanító adatokon számolt veszteség `loss` és validációs adatokon számolt veszteség `val_loss` értékek további iterációs lépések után sem csökkennek tovább érdemben. Amennyiben a veszteség közelít egy minimumhoz, az jelzi számunkra, hogy a hálózat nem tanul már új információt a jelenlegi adatokból. A 4.8 ábrán láthatjuk az említett veszteség értékek evolúcióját az epoch szám függvényében, mely a neurális hálózatok esetén elterjedt kifejezés a tanítóadatokon való végig iterálások számára. Az 4.8 ábrán a 64 modell konvergenciája látható, mindegyik külön színnel ábrázolva.



4.8. ábra. Direkt MLP modellek konvergenciája.

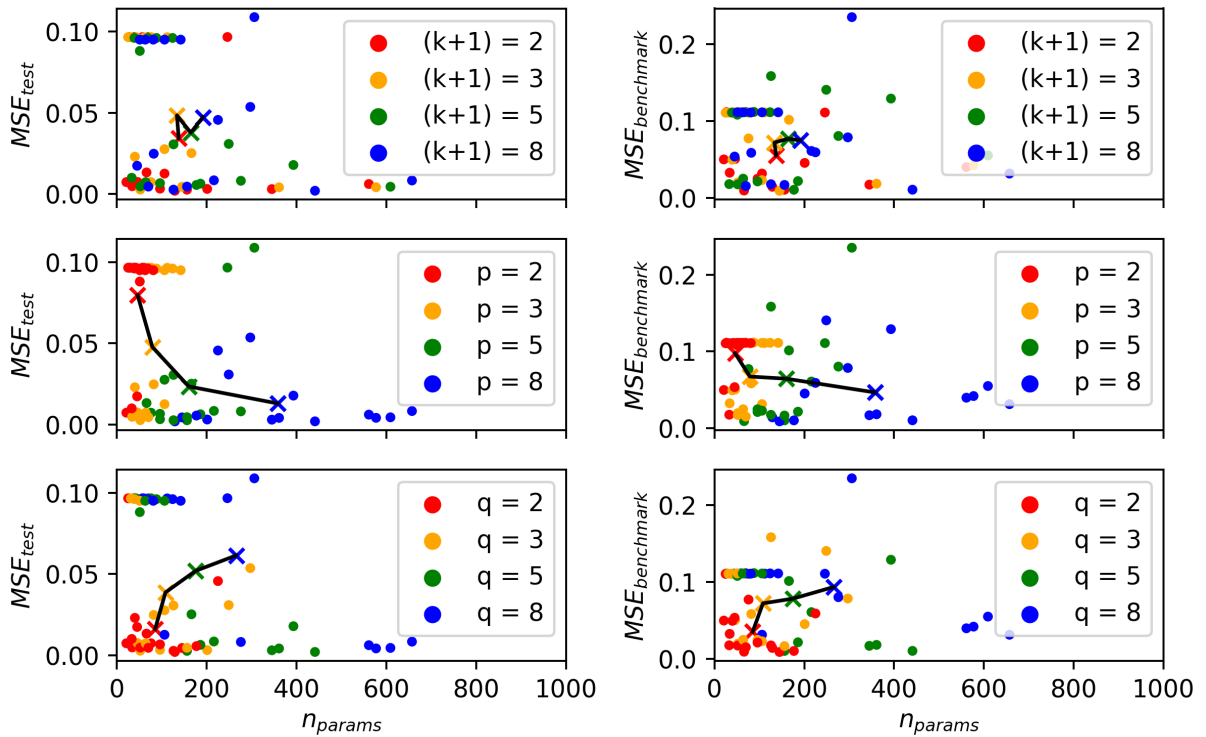
A 4.8 ábrán jól látható, hogy a egyaránt a tanító és validáló adatokon vizsgált veszteség értékek is kellően bekonvergáltak. 50 epochonként mentettem a modelleket, így ezeket az állapotukat is lehetőségünk van kiértékelni külön kiértékelni. Az összes mentett modell közül a legjobbakat és ezek mérőszámait a 4.3 foglalja magába.

A következőkben szeretnénk valamilyen kezdetleges következtetést levonni a  $k$ ,  $p$  és  $q$  értékek pontosságra vett hatását illetően, hogy ha szükséges tovább tudjuk finomítani a modell architektúránkat. Ehhez azt fogjuk tenni, hogy ábrázoljuk a 100 epoch utáni

**4.3. táblázat.** Direkt MLP modellezés legjobb kezdetleges modelljei.

| $(k, p, q)$ | $n_{epochs}$ | $MSE_{test}$           | $MSE_{benchmark}$       |
|-------------|--------------|------------------------|-------------------------|
| (4, 3, 2)   | 50           | $1.944 \times 10^{-3}$ | $10.955 \times 10^{-3}$ |
| (2, 8, 2)   | 50           | $2.576 \times 10^{-3}$ | $7.371 \times 10^{-3}$  |

modellek teszt és benchmark halmazokon elért  $MSE$  értékeit a háló összparaméter számának függvényében. Szeretnénk azt is látni, hogy melyik ponthoz milyen  $k$ ,  $p$  és  $q$  értékek társulnak. Ennek érdekében háromszor újra színezzük az említett két ábrát, úgy, hogy a színek az adott érték szerint vett opciókat jelöljék. Emellett ábrázoljuk a színenként vett pontthalmazok átlagát és összekötjük őket.



**4.9. ábra.**  $k, p$  és  $q$  hatása direkt MLP modellek esetén.

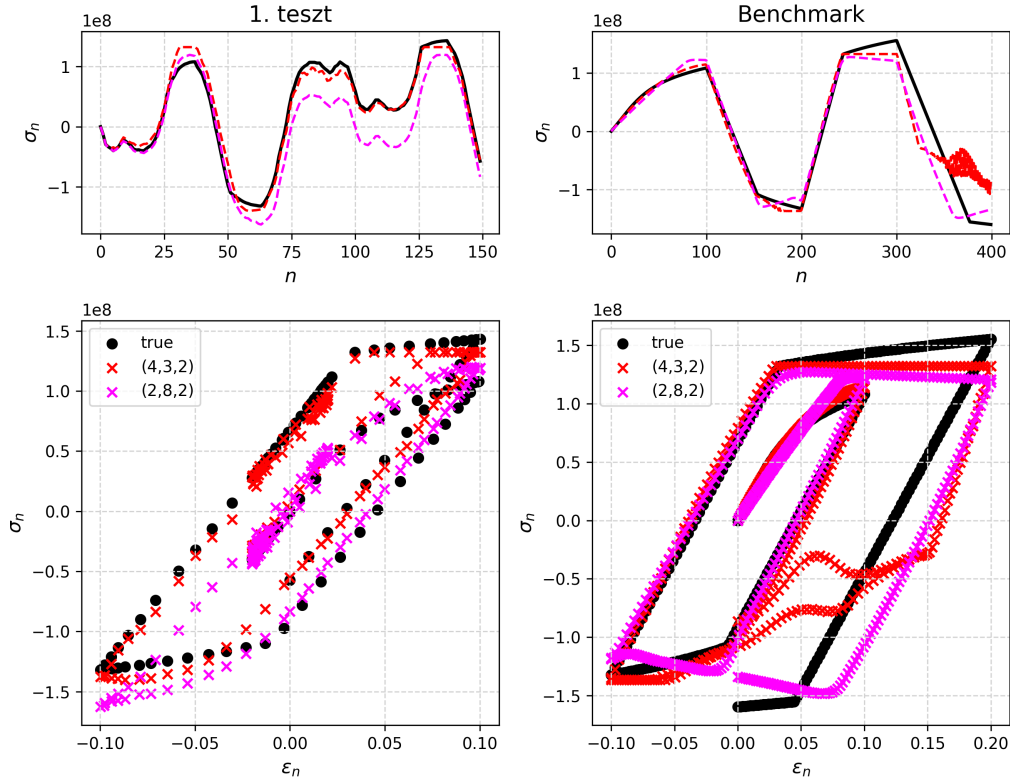
Az pontthalmazok átlagait összekötve az ábrán láthatóvá válik, hogy az adott érték ebben a tartományban milyen hatással van a háló méretére és annak pontosságára. Ezalapján azt mondhatjuk, hogy a  $q$  érték növelésével csökken az átlag hatékonyság, míg a  $p$  növelésével nő. Ebben az esetben ennyi pont alapján az optimális  $k$  értékről nem tudunk következtetést levonni. Ezért lerögzítjük a  $p = 8$  és  $q = 2$  értékeket, majd minden  $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$  értékre létrehozunk egy modellt, fejenként  $n_{epochs} = 200$  tanítóadatokon történő iterációval. Minden 50. iterációnál mentjük és kiértékeljük az új modelleket. Ezekkel kiegészülve az említett a test és benchmark halmazokon vett legjobb modelleket a 4.4 táblázat foglalja magába.

Ugyanakkor annak érdekében, hogy a számokhoz tudjunk értelmet társítani vizuálisan

#### 4.4. táblázat. Direkt MLP modellezés legjobb modelljei.

| $(k, p, q)$ | $n_{epochs}$ | $MSE_{test}$           | $MSE_{benchmark}$       |
|-------------|--------------|------------------------|-------------------------|
| (4, 3, 2)   | 50           | $1.944 \times 10^{-3}$ | $10.955 \times 10^{-3}$ |
| (2, 8, 2)   | 150          | $4.881 \times 10^{-3}$ | $7.312 \times 10^{-3}$  |

is megnézhetjük, hogy a teszt halmaz első idősorára és a benchmark idősorra miként is térnek el a valós rendszer és a modell által adott válaszok:



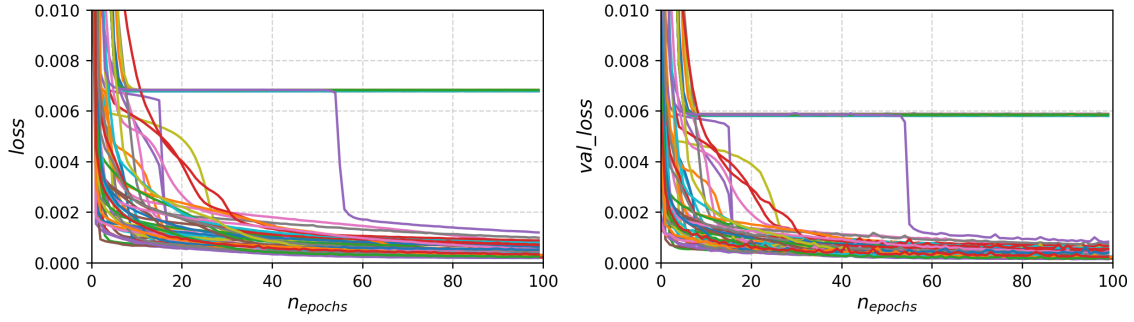
4.10. ábra. A legpontosabb direkt MLP modellek vizuális kiértékelése.

Itt a benchmark halmazon a (4,3,2) modell esetén megfigyelhető egy numerikus instabilitás, melyet feltehetően annak az eredménye, hogy a benchmark adathalmaz jelentősen hosszabb, mint a tanítás során látott idősorok.

#### 4.4.2. Inkrementális modellezés

Az **Direkt modellezés** fejezet lépéseit fogjuk megismételni, hogy az inkrementális esetre is tudjunk valamilyen következtetést levonni. Itt is megvizsgáljuk a konvergenciát. Itt a veszteség mértékének szintén az  $MSE$  értékeket definiáltam. Ekkor az előző fejezethez hasonlóan ábrázolhatjuk a tanító és validációs adatokon számolt veszteség értékeket az epoch szám függvényében.

Itt láthatóan vannak olyan modelljeink amelyek nagyságrendileg más értékhez konvergáltak be, melynek oka vélhetően a növekményes modellezésben keresendő. Ezen



**4.11. ábra.** Inkrementális MLP modellek konvergenciája.

64 modellek állapotát is 50 epochonként mentettem, az így keletkező összes állapot közül a legkisebb hiba mértéket a teszt és a benchmark adathalmazon is a  $(7, 8, 5)$  modell 50 epoch utáni állapota érte.

**4.5. táblázat.** Inkrementális MLP modellezés legjobb kezdetleges modelljei.

| $(k, p, q)$ | $n_{epochs}$ | $MSE_{test}$           | $MSE_{benchmark}$      |
|-------------|--------------|------------------------|------------------------|
| $(7, 8, 5)$ | 50           | $1.444 \times 10^{-3}$ | $7.052 \times 10^{-3}$ |

A következőkben az előző fejezethez hasonlóan szeretnénk valamilyen kezdetleges következtetést levonni a  $k$ ,  $p$  és  $q$  értékek pontosságra vett hatásait illetően. Ehhez megismételjük az ott látható műveletet és ábrázoljuk a 100 epoch utáni modellek teszt és benchmark halmazokon elért  $MSE$  értékeit a háló összparaméter számának függvényében. Azért, hogy lássuk mely ponthoz milyen  $k$ ,  $p$  és  $q$  értékek társulnak, háromszor újra színezzük az említett két ábrát, úgy, hogy a színek az adott érték szerint vett opciókat jelöljék. Emellett ismét ábrázoljuk a színenként vett ponthalmazok átlagát és összekötjük őket. Mindezt a 4.12 ábra foglalja magába, melyen megfigyelhető, hogy a  $q > 3$  esetekre csökken az átlag hatékonysága a modelleknek, míg a  $p$  és  $k$  növelésével nő.

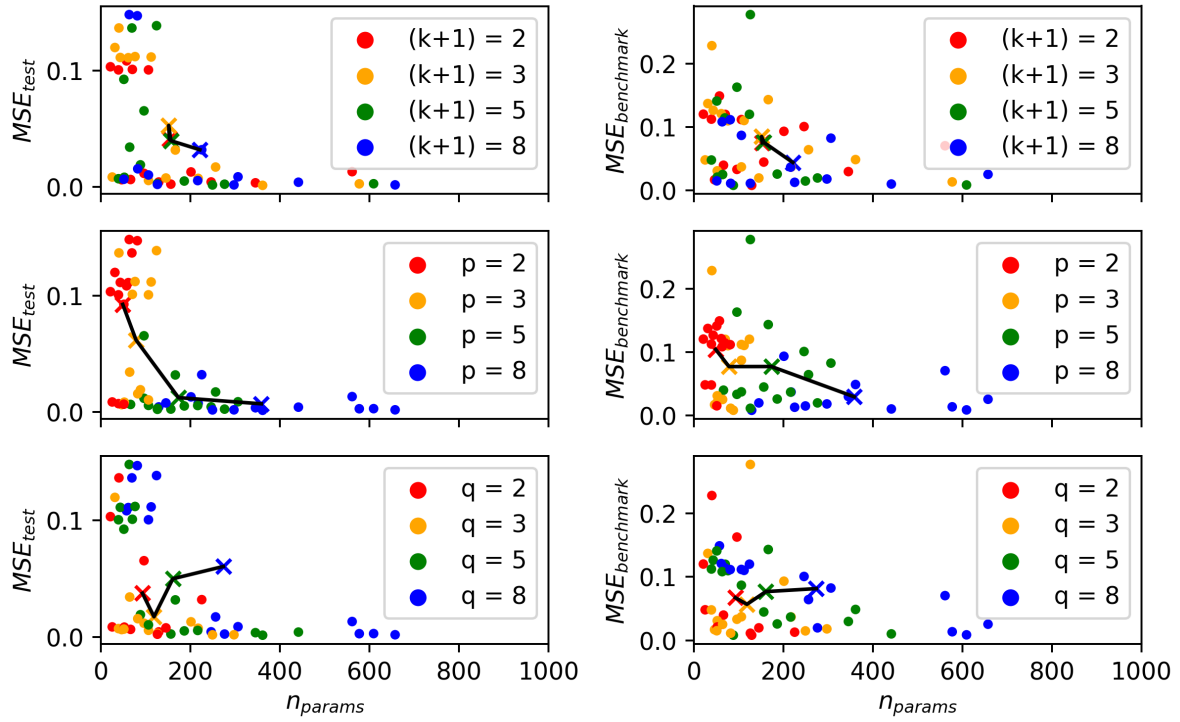
Ismételten lerögzítjük a  $p = 8$  és  $q = 3$  értékeket, majd minden  $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$  értékre létrehozunk egy modellt, fejenként  $n_{epochs} = 200$  tanítóadatokon történő iterációval. Minden 50. iterációnál mentjük és kiértékeljük a modelleket. Ezekkel kiegészülve az összes modell közül a legeredményesebbeket a 4.6 táblázat foglalja magába.

**4.6. táblázat.** Inkrementális MLP modellezés legjobb modelljei.

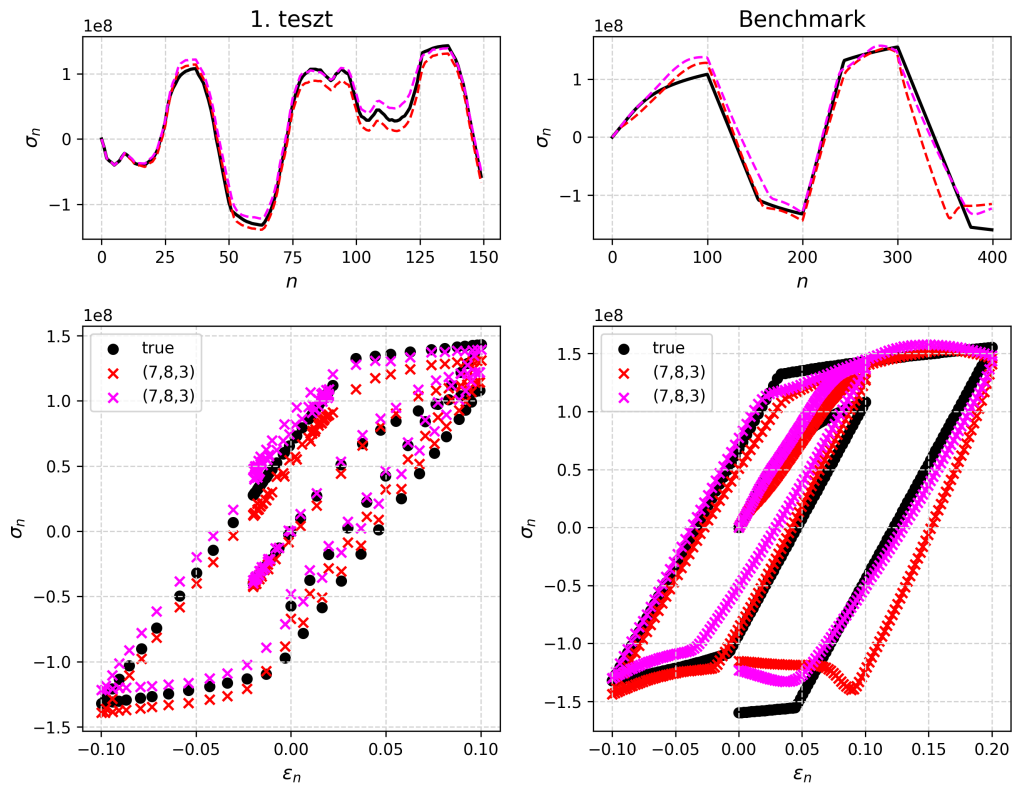
| $(k, p, q)$ | $n_{epochs}$ | $MSE_{test}$           | $MSE_{benchmark}$      |
|-------------|--------------|------------------------|------------------------|
| $(7, 8, 3)$ | 200          | $1.147 \times 10^{-3}$ | $8.369 \times 10^{-3}$ |
| $(7, 8, 3)$ | 150          | $2.452 \times 10^{-3}$ | $4.184 \times 10^{-3}$ |

Ismét nézzük meg vizuálisan is, hogy a két idősorra miként is térnek el a valós rendszer és a modell által adott válaszok. Ezt a 4.13 ábrán figyelhetjük meg. Mind a numerikus, mind a vizuális kiértékelésből látható, hogy ez a modellezési módszer sikeresebbnek bizonyult

az előzőnél.



4.12. ábra.  $k, p$  és  $q$  hatása inkrementális MLP modellek esetén



4.13. ábra. A legpontosabb inkrementális MLP modellek vizuális kiértékelése.

## 4.5. LSTM hálózatok

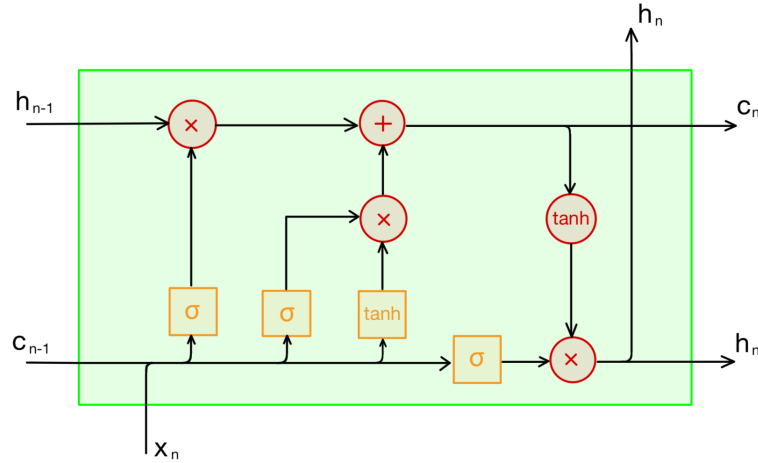
Az RNN (Recurrent Neural Network) hálók annyiban térnek el a korábban bemutatott MLP hálóktól, hogy egy adott  $(i + 1)$ . neuron réteg neuronjai nem csak az  $i$ . réteg neuronjainak értékétől függ, hanem az  $(i + 1)$ . réteg előző állapotától is. Tehát egy tetszőleges neuronra:

$$h_n^{(i+1)} = \sigma(W^{(i)}h_n^{(i)} + V^{(i)}h_{n-1}^{(i+1)} + b^{(i)}), \quad (4.14)$$

ahol  $W^{(i)}$  a rétegeket összekötő súlymátrixok,  $V^{(i)}$  a réteget a saját előző állapotával összekötő súlymátrixok, illetve  $b^{(i)}$  a réteghez tartozó súlymátrix.

A Long-Short Term Memory háló [15] az első olyan rekurens neurális háló, amely megoldást nyújtott a rekurens hálózatok fő problémáira a gradiens "eltűnésére" és "robbanására" (vanishing and exploding gradient problem). Ez a megközelítés a mai napig alapját képezi a legkorszerűbb időfüggő adatok modellezésére szolgáló adatalapú modelleknek.

Mivel több belső változót is definiálni fogunk és mindegyik réteg függő, ezért az átláthatóság kedvéért a bemutatás során ragaszkodjunk a következő jelöléshez:  $h_n^{(i+1)} = h_n$  és  $h_n^{(i)} = x_n$ . Ekkor egy LSTM cellát az 4.14 ábrán látható módon tudunk definiálni.



4.14. ábra. LSTM cella ábrázolása.

Ahogy az ábrán is látható, egy LSTM cella belsejében különféle kapuk (gates) szabályozzák, hogy az egyes időlépésekben milyen információk kerüljenek tárolásra. Ezeket a kapukat felírhatjuk egy konkrét neuron rétegre:

$$\begin{aligned}
f_n &= \sigma(V_f \cdot h_{n-1} + W_f \cdot x_n + b_f), \\
i_n &= \sigma(V_i \cdot h_{n-1} + W_i \cdot x_n + b_i), \\
o_n &= \sigma(V_o \cdot h_{n-1} + W_o \cdot x_n + b_o), \\
\tilde{c}_n &= \tanh(V_{\tilde{c}} \cdot h_{n-1} + W_{\tilde{c}} \cdot x_n + b_{\tilde{c}}).
\end{aligned} \tag{4.15}$$

Ezek a kapuk határozzák meg számunkra az LSTM cella belső állapotát, illetve a belső állapot adja a cella kimenetét:

$$\begin{aligned}
c_n &= c_{n-1} \cdot f_n + i_n \cdot \tilde{c}_n \\
h_n &= \tanh(c_n) \cdot o_n
\end{aligned} \tag{4.16}$$

Tehát egy általánosabb alakban a következőképpen írható le egy LSTM cella működése:

$$\begin{cases} h_n = f_h(h_{n-1}, c_{n-1}, x_n) \\ c_n = f_c(h_{n-1}, c_{n-1}, x_n) \end{cases} \tag{4.17}$$

Látható, hogy egy LSTM cella valójában egy diszkrét dinamikai rendszert reprezentál és ebből kifolyólag alapvetően rendelkezik egyfajta belső memóriával, ahogy természetesen az elnevezése is utal rá.

Mivel ezek a hálózatok rendelkeznek memória jelleggel, ezért belátható, hogy a (4.4) modellezési módok eredményesen tudjuk használni, azokat is ahol az elkövetkezendő kimenő jel csak az aktuális bemenőjeltől és annak meghatározott hosszúságú múltjától függ. Tehát LSTM hálózatok esetén a következő két modellezési módszerrel fogunk próbálkozni:

1. Direkt modellezés:

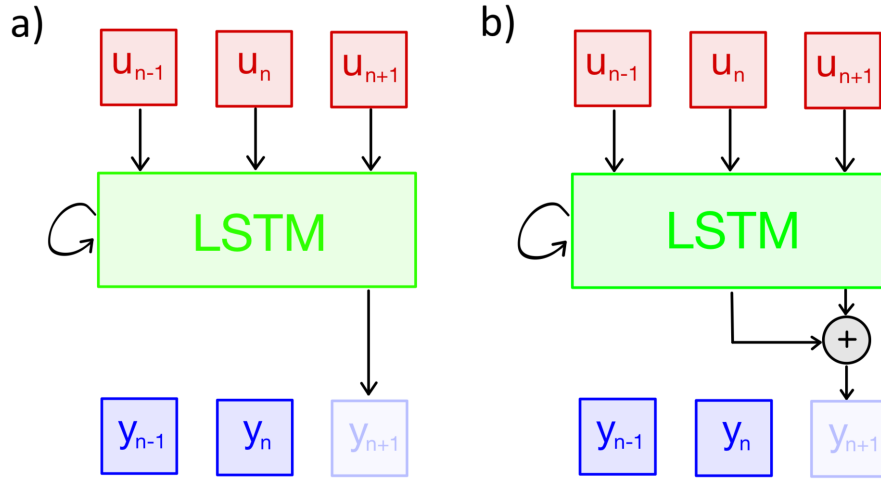
$$y_{n+1} = F_n(u_{n+1}, \dots, u_{n-k}) \tag{4.18}$$

2. Inkrementális modellezés:

$$y_{n+1} = y_n + f_n(u_{n+1}, \dots, u_{n-k}) \tag{4.19}$$

Mind a két modellezési lehetőséget a könnyebb értelmezhetőség kedvéért  $k = 1$  értékre az 4.15 ábrán ábrázoltuk.

Annak érdekében, hogy a lehető legjobb pontosságot érjük el, ezért mindkét esetben meg fogunk vizsgálni több háló struktúrát, mindegyikkel be fogunk tanítani egy modellt és megnézzük, melyik hogyan teljesít. Ezért paraméterezhetővé tesszük a rétegek számát  $q$ , illetve a rétegek neuronjainak a számát  $p$ . Ekkor a korábbi megfontolások alapján



**4.15. ábra.** LSTM modellezési lehetőségek: a) Direkt modellezés, valamint b) Inkrementális modellezés.

belátható, hogy egy ilyen LSTM struktúra paramétereinek száma a következőképpen adható meg:

$$n_{prams} = 4 \cdot (p \cdot (k + 2) + p^2 + p) + (q - 1) \cdot 4 \cdot (p \cdot p + p^2 + p) + (p + 1) \quad (4.20)$$

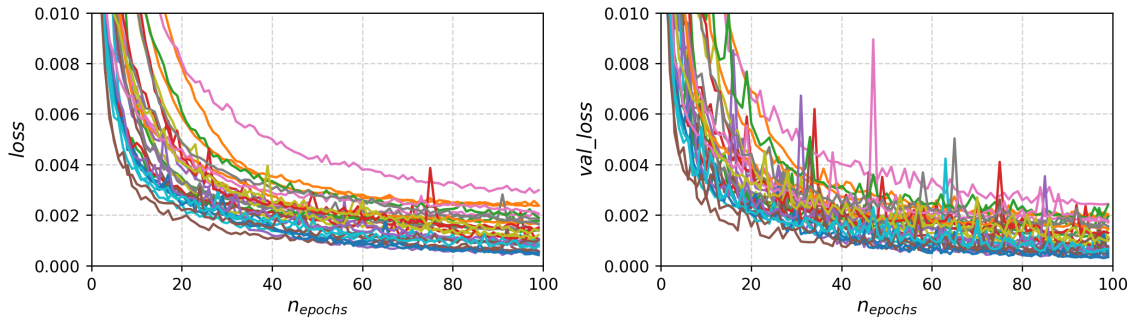
Mivel LSTM hálózatok esetén a tanítási procedúra még számításigényesebb, ezért az ideális struktúra megsejtéséhez a korábbi tapasztalataink alapján rögzítjük a  $q = 2$  értéket, majd ezt követően megvizsgáljuk a  $(k + 1)$ ,  $p$  értékek egymástól független hatását a pontosságra. Ennek érdekében a következő modelleket fogom megvizsgálni:

$$(k, p, 2) \quad \text{ahol} \quad (k + 1) \in \{0, 2, 3, 5, 8, 10\} \quad \text{és} \quad p \in \{2, 3, 5, 8, 10\} \quad (4.21)$$

Ami azt jelenti, hogy mindkét esetben  $6 \cdot 5 = 30$  modellt fogok betanítani és kiértékelni. Minden esetben  $n_{epochs} = 100$  alaklommal iteráltam végig a tanító adatokon. Ezután az így létrejött modelleket kiértékeltem a korábban említett 10, fejenként 150 időlépésből álló teszt adathalmazon, illetve a manuálisan generált 400 lépésből álló benchmark adathalmazon. A hibát  $MSE$  és  $MAE$  értékekkel mértem.

#### 4.5.1. Direkt modellezés

Mielőtt kiértékelnék a 30 modellt, először is meg szeretnék bizonyosodni arról, hogy mindegyik kellően bekonvergált a tanítás során. Ennek érdekében ábrázoljuk a tanítás során a tanító és validáló adatokon számolt veszteség értékeket, melyeknek szintén az  $MSE$  értéket választottam.



**4.16. ábra.** Direkt LSTM modellek konvergenciája.

Bár láthatóan kevésbé frekvenciált a konvergencia, de az elérték láthatóan elérték egy konvergencia tartományba. Ebben az esetben is minden 50 epoch után mentettem mind a 30 modellt, mely állapotok közül a teszt adathalmazon és a benchmark adathalmazon legjobban teljesítőket a 4.7 táblázat foglalja magába.

**4.7. táblázat.** Direkt LSTM modellezés legjobb kezdetleges modelljei.

| $(k, p, q)$  | $n_{epochs}$ | $MSE_{test}$           | $MSE_{benchmark}$       |
|--------------|--------------|------------------------|-------------------------|
| $(4, 10, 2)$ | 100          | $0.466 \times 10^{-3}$ | $59.276 \times 10^{-3}$ |
| $(7, 3, 2)$  | 50           | $2.987 \times 10^{-3}$ | $6.925 \times 10^{-3}$  |

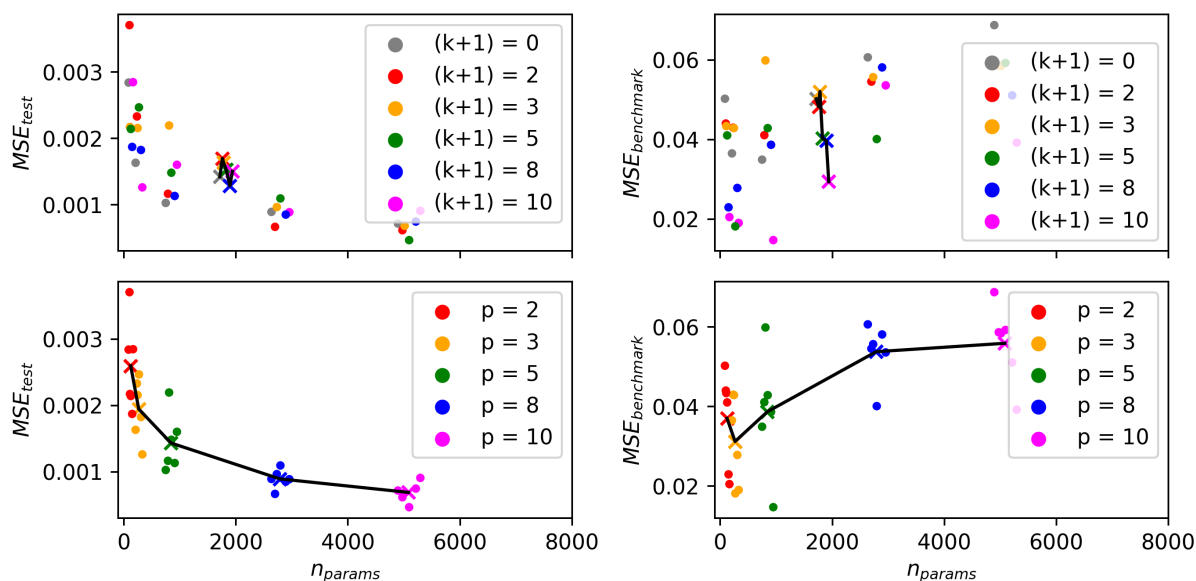
Most az előzőekhez hasonlóan vizsgáljuk meg a teszt és benchmark halmazokon számolt  $MSE$  értékeket a háló összparaméter számának függvényében. Szeretnénk azt is látni, hogy melyik ponthoz milyen  $p$  és  $k$  értékek társulnak. Ehhez kétszer újra színezzük az említett két ábrát, úgy, hogy a színek az adott érték szerint vett lehetőségeket jelöljék. A színenként vett ponthalmazok átlagát is ábrázoljuk.

Itt jól látható, hogy a  $p$  hatására folyamatosan nő a teszt halmazon az átlag pontosság, míg a benchmark halmazon folyamatosan csökken. Emellett az is látszik, hogy milyen jelentős hatása van a hálóméretre. Emellett az is látható, hogy a  $(k + 1)$  paraméter növelése pozitív hatással van a háló átlagpontosságára. Ezért rögzítem a  $p = 5$  értéket és  $(k + 1) \in \{8, 10, 12, 15, 18\}$  értékekkel tanítok egy-egy modellt mindegyiket 200 epochig, melyeket 50 epochonként mentek. Ezt követően a legeredményesebb modellek:

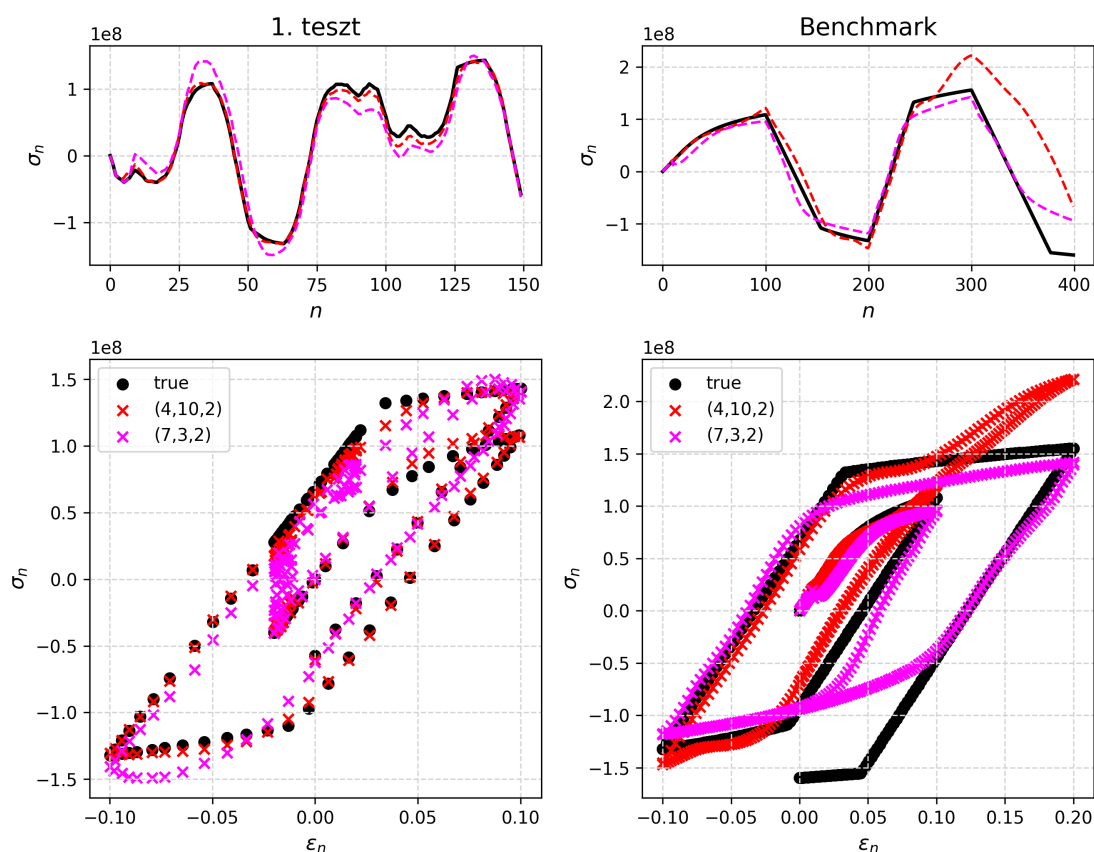
| $(k, p, q)$  | $n_{epochs}$ | $MSE_{test}$           | $MSE_{benchmark}$       |
|--------------|--------------|------------------------|-------------------------|
| $(4, 10, 2)$ | 100          | $0.466 \times 10^{-3}$ | $59.276 \times 10^{-3}$ |
| $(7, 3, 2)$  | 50           | $2.987 \times 10^{-3}$ | $6.925 \times 10^{-3}$  |

**4.8. táblázat.** Direkt LSTM modellezés legjobb modelljei

Ismét nézzük meg vizuálisan is, hogy a két idősorra miként is térnek el a valós rendszer és a modell által adott válaszok. Ezt a 4.18 ábrán figyelhetjük meg.



4.17. ábra.  $k, p$  és  $q$  hatása direkt LSTM modellek esetén.



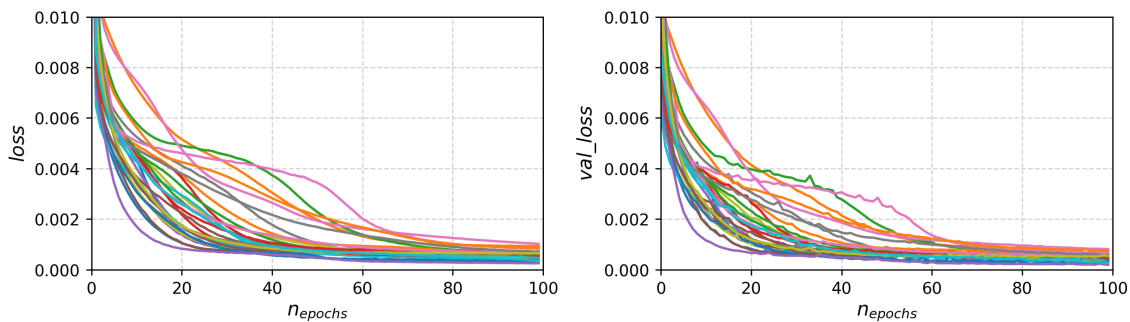
4.18. ábra. A legpontosabb direkt LSTM modellek vizuális kiértékelése

Itt ami a korábbi eredményekkel szemben figyelemre méltó, hogy a benchmark halmaz elején a modellek látszólag pontosabbak mint a korábbi módszerek esetén létrejött társaik, viszont ezt követően elromlik a pontosságuk. A rekurens hálók ezen jellemzőjét Bonatti

egy 2022-es [16] munkájában részletesen vizsgálta és adott rá egy alternatívát, melyet Linearized Minimal State Cell (LMSC)-nek nevezett.

#### 4.5.2. Inkrementális modellezés

Az **Direkt modellezés** fejezet lépéseit fogjuk megismételni, hogy az inkrementális esetre is tudjunk valamilyen következtetést levonni. Itt is megvizsgáljuk a konvergenciát. Itt a veszteség mértékének szintén az  $MSE$  értékeket választunk. Ekkor az előző fejezethez hasonlóan ábrázolhatjuk a tanító és validációs adatokon számolt veszteség értékeket az epoch szám függvényében.



4.19. ábra. Inkrementális LSTM modellek konvergenciája.

Itt a modellek konvergenciájában megfigyelhető egy érdekes inflexió szerű viselkedés. Valamint azt is láthatjuk, hogy az inkrementális LSTM-ek esetén jóval "simább" a konvergencia, mint a direkt LSTM-ek esetén.

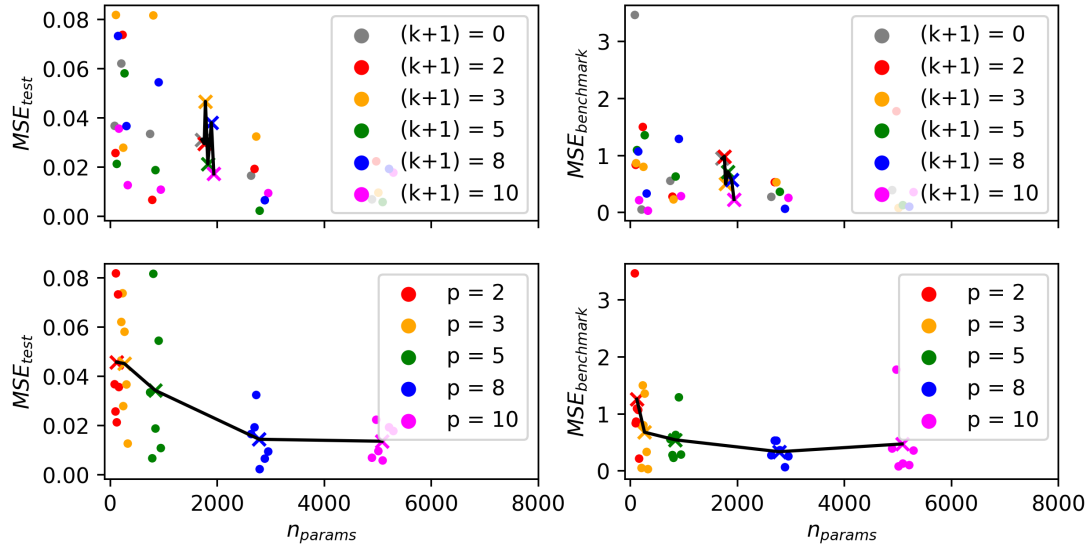
Ebben az esetben is minden 50 epoch után mentettem mind a 30 modellt, mely állapotok közül a teszt adathalmazon és a benchmark adathalmazon legjobban teljesítőket a 4.7 táblázat foglalja magába.

4.9. táblázat. Inkrementális LSTM modellezés legjobb kezdetleges modelljei.

| $(k, p, q)$ | $n_{epochs}$ | $MSE_{test}$            | $MSE_{benchmark}$        |
|-------------|--------------|-------------------------|--------------------------|
| (4, 8, 2)   | 100          | $2.224 \times 10^{-3}$  | $364.714 \times 10^{-3}$ |
| (9, 3, 2)   | 100          | $12.621 \times 10^{-3}$ | $32.555 \times 10^{-3}$  |

Most az előzőekhez hasonlóan vizsgáljuk meg a teszt és benchmark halmazokon számolt  $MSE$  értékeket a háló összparaméter számának függvényében. Szeretnénk azt is látni, hogy melyik ponthoz milyen  $p$  és  $k$  értékek társulnak. Ehhez kétszer újra színezzük az említett két ábrát, úgy, hogy a színek az adott érték szerint vett lehetőségeket jelöljék. A színenként vett ponthalmazok átlagát is ábrázoljuk.

Ebből is a **Direkt modellezés**hez hasonló következtetés vonható le, annyi különbséggel, hogy itt látszólag arányában kevesebbet ront a  $p$  érték növelése a benchmark pontosságra.



**4.20. ábra.**  $k, p$  és  $q$  hatása inkrementális LSTM modellek esetén.

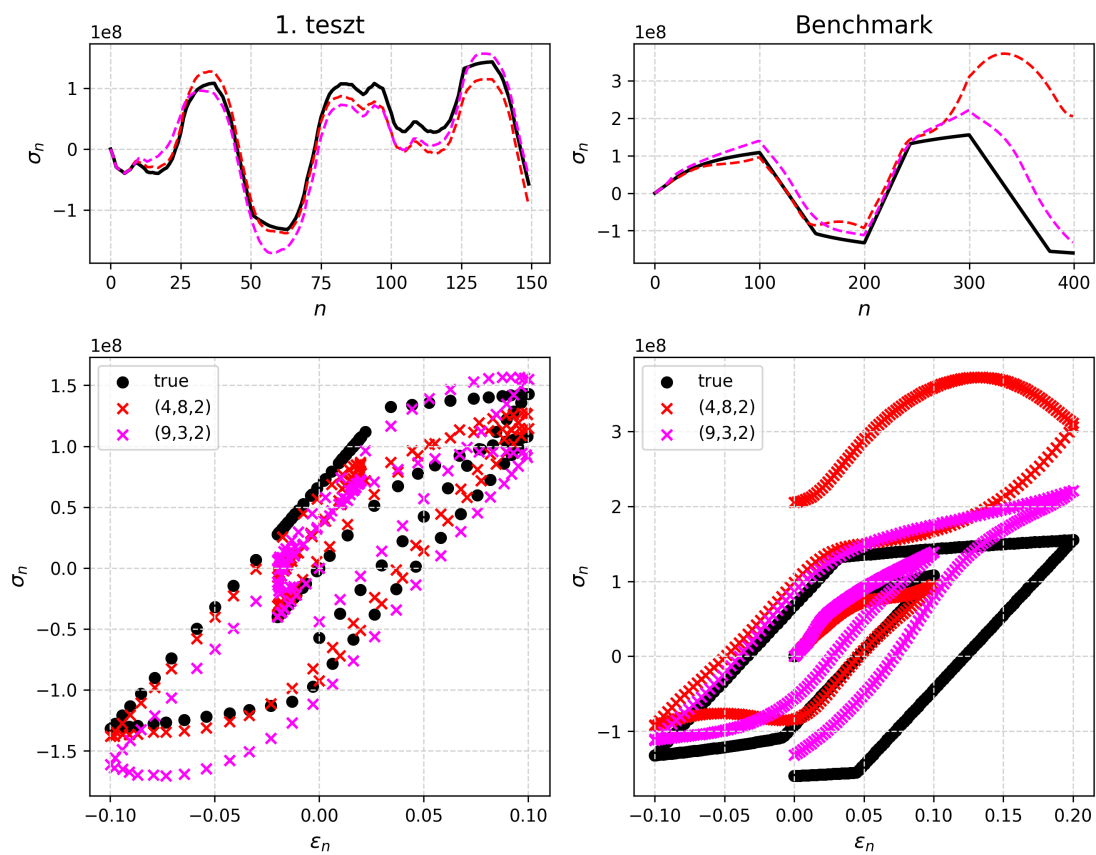
Itt is próbálkozni fogunk a  $p = 5$  rögzítéssel és a  $(k + 1) \in \{8, 10, 12, 15, 18\}$  irányba való tovább indulással. Ugyancsak 200 epochig tanítom a modelleket és 50 mentem minden 50 epoch után. Ezt követően a legeredményesebb modelleket

**4.10. táblázat.** Inkrementális LSTM modellezés legjobb modelljei.

| $(k, p, q)$ | $n_{epochs}$ | $MSE_{test}$            | $MSE_{benchmark}$        |
|-------------|--------------|-------------------------|--------------------------|
| $(4, 8, 2)$ | 100          | $2.224 \times 10^{-3}$  | $364.714 \times 10^{-3}$ |
| $(9, 3, 2)$ | 100          | $12.621 \times 10^{-3}$ | $32.555 \times 10^{-3}$  |

Ismét nézzük meg vizuálisan is, hogy a két idősorra miként is térnek el a valós rendszer és a modell által adott válaszok. Ezt a 4.18 ábrán figyelhetjük meg.

Ebben az esetben is levonható az a következtetés, mint az előző LSTM próbálkozásnál, hogy a benchmark halmaz elején a modellek látszólag pontosak, viszont ezt követően elromlik a pontosságuk. Tehát itt is látható a [16]-ban vizsgált viselkedés.

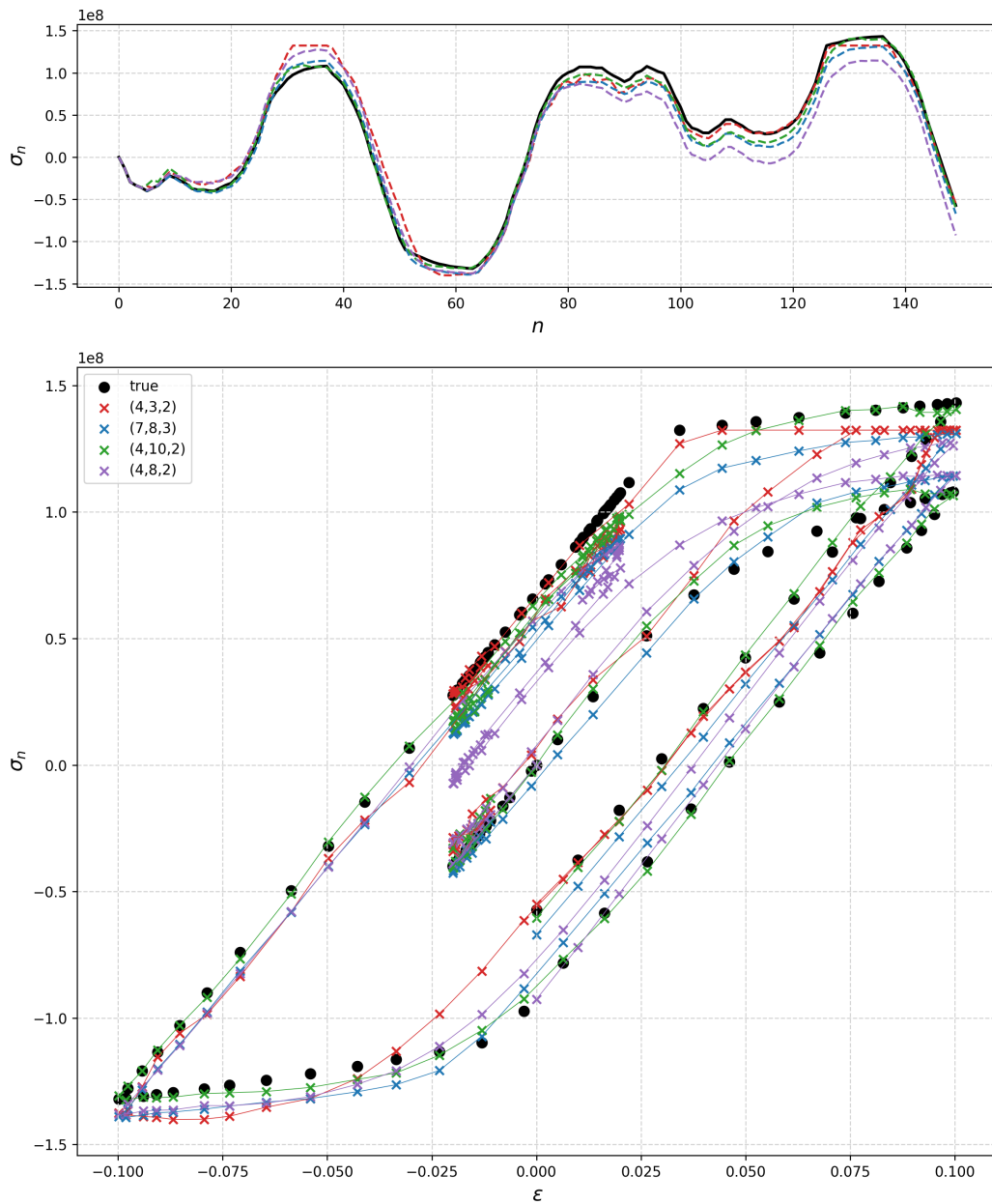


4.21. ábra. A legpontosabb inkrementális LSTM modellek vizuális kiértékelése.

## 5. Összegzés

5.1. táblázat. A legjobb modellek a teszt halmazon

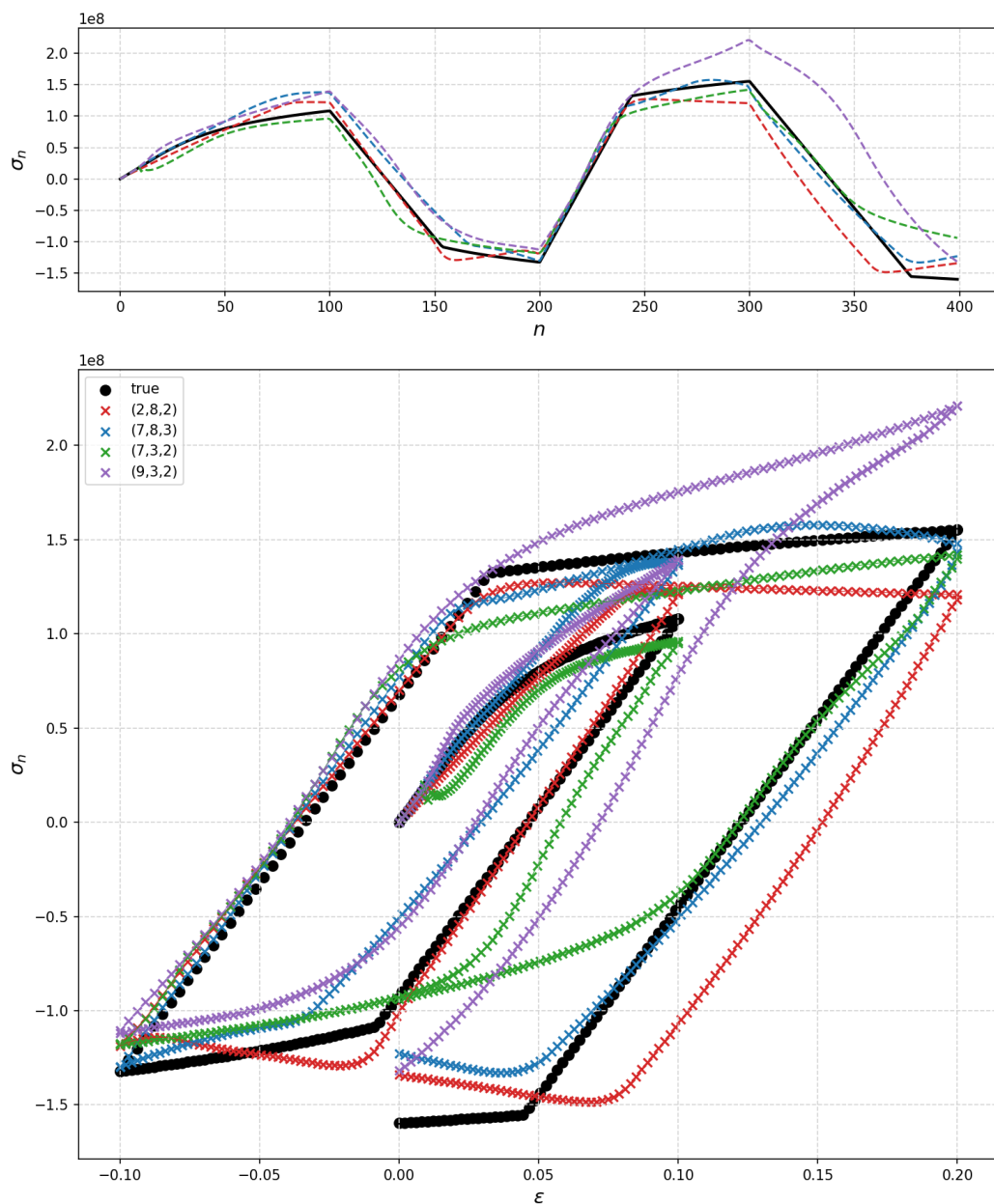
| Hálózat | Módszer       | Epoch | $(k, p, q)$  | $n_{params}$ | $MSE_{test}$           | $MSE_{benchmark}$        |
|---------|---------------|-------|--------------|--------------|------------------------|--------------------------|
| MLP     | Direkt        | 50    | $(4, 3, 2)$  | 52           | $1.944 \times 10^{-3}$ | $10.955 \times 10^{-3}$  |
| MLP     | Inkrementális | 200   | $(7, 8, 3)$  | 281          | $1.147 \times 10^{-3}$ | $8.369 \times 10^{-3}$   |
| LSTM    | Direkt        | 100   | $(4, 10, 2)$ | 1531         | $0.466 \times 10^{-3}$ | $59.276 \times 10^{-3}$  |
| LSTM    | Inkrementális | 100   | $(4, 8, 2)$  | 1033         | $2.224 \times 10^{-3}$ | $364.714 \times 10^{-3}$ |



5.1. ábra. A legjobb modellek a teszt halmaz első idősorán

5.2. táblázat. A legjobb modellek a benchmark halmazon

| Hálózat | Módszer       | Epoch | $(k, p, q)$ | $n_{params}$ | $MSE_{test}$            | $MSE_{benchmark}$       |
|---------|---------------|-------|-------------|--------------|-------------------------|-------------------------|
| MLP     | Direkt        | 150   | (2, 8, 2)   | 129          | $4.881 \times 10^{-3}$  | $7.312 \times 10^{-3}$  |
| MLP     | Inkrementális | 150   | (7, 8, 3)   | 281          | $2.452 \times 10^{-3}$  | $4.184 \times 10^{-3}$  |
| LSTM    | Direkt        | 50    | (7, 3, 2)   | 244          | $2.987 \times 10^{-3}$  | $6.925 \times 10^{-3}$  |
| LSTM    | Inkrementális | 100   | (9, 3, 2)   | 268          | $12.621 \times 10^{-3}$ | $32.555 \times 10^{-3}$ |



5.2. ábra. A legjobb modellek a benchmark halmazon

A teszt halmazon legjobban teljesítő modellek kiértékelésén jól látszik, hogy annak ellenére, hogy nem képezték részét a tanító és teszt adatoknak, a generálási mód mindhárom adathalmaz esetén ugyanaz volt, hiszen a hosszban és karakterisztikában teljesen független benchmark adathalmazon már nagyságrendileg rosszabbul teljesítenek ezek a modellek. Emellett viszont a benchmarkon jól teljesítő modellek esetén, a teszt halmazon vett átlag hibák is azonos nagyságrendűek. Ez alapján kijelenthető, hogy a benchmark szerint vett legjobb modellek általánosabb felhasználási lehetőséget rejtnek magukban, hiszen közelebb vannak a modellezni kívánt "ismeretlen" viselkedéshez.

Ezen modellek közül az inkrementális LSTM modellt leszámítva mindegyik átlag hibája 1% alatt van. Ez az inkrementális MLP esetén mind a benchmark, mind a teszt halmazon 0.5% alatt van. Szóval ez a megközelítésmód tekinthető a legeredményesebbnek.

A [12] cikk alapján várható eredményekhez képest egy nagyságrenddel kisebb pontosságot értünk el. Ennek az oka, hogy a [12]-ben szereplő modellek tanítására szolgáló adatok generálása esetén kevésbé volt szempont a véletlenszerűség, ebből kifolyólag bár a munka nem említi, a modellek várhatóan kevésbé lesznek alkalmasak általános bemenőjelre adott válaszok predikálására. Esetünkben szükségünk volt arra, hogy a modellek jól teljesítsenek minél általánosabb bemenőjelek esetén is, ezért a megközelítésmódot kiegészítettük egy tovább gondolt, véletlenszerű bemenőjel generálási móddal, mely segítségével sikerült pontos és ugyanakkor általános modelleket alkotni. Emellett számunkra a hálóméret minimalizálása is szempont volt, ezt figyelembe tartva sikerült elérni, hogy a legeredményesebb modellek mérete számottevően kisebb lett, mint amit a [12] eredményei alapján várhattunk.

Tehát sikerült egy a modell szempontjából ismeretlen anyagi viselkedést, pusztán szimulációból kinyert adatsorok alapján viszonylag pontosan közelíteni. Annak ellenére, hogy egy elméleti anyagi viselkedésre alkottunk modellt, a módszer átültethető bármilyen anyagi viselkedésre, amelyet képesek vagyunk megmérni, hiszen a probléma alaptermészete minden anyag esetén hasonló. Ez lehetőséget ad a nehezen modellezhető, erős nemlinearitást mutató anyagok eredményes modellezésére. Mind emellett mivel diszkrét rendszerként modelleztük az anyagot, így a modell egyenesen alkalmazható egy végeselemes szimuláció során.

## Hivatkozások

- [1] Jamshid Ghaboussi, JH Garrett Jr, and Xiping Wu. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics*, 117(1):132–153, 1991.
- [2] Johannes Dornheim, Lukas Morand, Hemanth Janarthanam Nallani, and Dirk Helm. Neural networks for constitutive modeling: From universal function approximators to advanced models and the integration of physics. *Archives of Computational Methods in Engineering*, 31(2):1097–1127, 2024.
- [3] M. Lefik and B.A. Schrefler. Artificial neural network as an incremental non-linear constitutive model for a finite element code. *Computer Methods in Applied Mechanics and Engineering*, 192(28):3265–3283, 2003. Multiscale Computational Mechanics for Materials and Structures.
- [4] Tomonari Furukawa and Genki Yagawa. Implicit constitutive modelling for viscoplasticity using neural networks. *International Journal for Numerical Methods in Engineering*, 43(2):195–219, 1998.
- [5] Maysam B Gorji and Dirk Mohr. Towards neural network models for describing the large deformation behavior of sheet metal. In *IOP Conference Series: Materials Science and Engineering*, volume 651, page 012102. IOP Publishing, 2019.
- [6] Maxime Lavech du Bos, Fadoua Balabdaoui, and Julian N. Heidenreich. Modeling stress-strain curves with neural networks: a scalable alternative to the return mapping algorithm. *Computational Materials Science*, 178:109629, 2020.
- [7] Markus Oeser and Steffen Freitag. Modeling of materials with fading memory using neural networks. *International journal for numerical methods in engineering*, 78(7):843–862, 2009.
- [8] Pin Zhang, Zhen-Yu Yin, Yin-Fu Jin, and Guan-Lin Ye. An ai-based model for describing cyclic characteristics of granular materials. *International Journal for Numerical and Analytical Methods in Geomechanics*, 44(9):1315–1335, 2020.
- [9] Ning Zhang, Shui-Long Shen, Annan Zhou, and Yin-Fu Jin. Application of lstm approach for modelling stress–strain behaviour of soil. *Applied Soft Computing*, 100:106959, 2021.
- [10] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

- [11] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [12] Jesús Gonzalez and Wen Yu. Non-linear system modeling using lstm neural networks. *IFAC-PapersOnLine*, 51(13):485–489, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.
- [13] Nam-Ho Kim. *Introduction to nonlinear finite element analysis*. Springer Science & Business Media, 2014.
- [14] René De Borst, Mike A Crisfield, Joris JC Remmers, and Clemens V Verhoosel. *Nonlinear finite element analysis of solids and structures*. John Wiley & Sons, 2012.
- [15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.
- [16] Colin Bonatti and Dirk Mohr. On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids. *Journal of the Mechanics and Physics of Solids*, 158:104697, 2022.

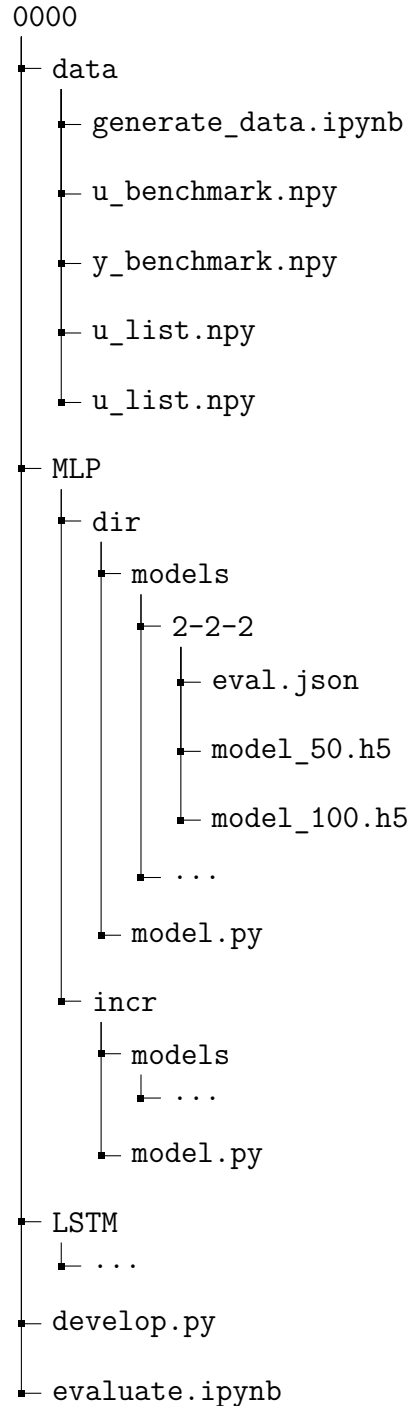
## Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani Dr. Kossa Attila egyetemi docensnek, témavezetőmnek, aki szakmai útmutatásaival és támogatásával hozzájárult a dolgozatom elkészítéséhez.

Jelen kutatás a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával valósult meg. Pályázati azonosító: FK 142457. Jelen TDK dolgozatban bemutatott tudományos eredmények az MTA Bolyai János Kutatási Ösztöndíj támogatásával készültek

# Függelék

A függelék tartalmazza a Python kódokat, amelyeket a példa megoldásához használtam. A kódok valamint az adatok és modellek tárolásra szolgáló fájlok rendszere:



Minden modell egy python osztályként példányosítható és a neki megfelelő mappában lévő `model.py` tartalmazza. A mappa alatt továbbá van egy `models` nevű mappa, ami a konkrét  $(k, p, q)$  értékekkel betanított modelleket és azok kiértékeléseit tartalmazza. A fő mappából elérhető minden modell és kiértékelés, az utófeldolgozáshoz.