Predikce silových a momentových korekčních faktorů pro válcování kovů pomocí AI

Adam Peichl*[1], Cyril Oswald[1]

[1]*ČVUT v Praze, Fakulta strojní, Ústav přístrojové a řídicí techniky, Technická 4, 166 07 Praha 6, Česká republika*

Abstrakt

Tato práce se zabývá predikcí korekčních faktorů silových a momentových parametrů pro válcování kovů získaných matematickým modelem, což vede ke zvýšení přesnosti celého procesu. Cílem práce je návrh softwarového řešení, jeho implementace a následné otestování na reálných datech dodaných firmou PT SOLUTIONS WORLDWIDE spol. s r.o. Vzhledem ke komplexnosti matematicko-fyzikálního modelu je v této práci prezentována metoda složená z algoritmů předzpracování dat, neuronových sítí a strojového učení. Dodaný modul v jazyce C++ je momentálně součástí softwarových balíků firmy PTSW, které mají být nasazeny v systémech řízení válcovacích linek.

Klíčová slova: predikce; válcování kovů; umělá inteligence; HONU; strojové učení

1. Úvod

Pro návrh výrobního postupu pro válcování plechů je třeba určit přítlačné síly a momenty tak, aby nebyly překročeny jejich maximální hodnoty. V současnosti se tyto síly vypočítávají pomocí velice komplexního matematicko-fyzikálního modelu.

Model je nejen složitý, ale vstupuje do něho mnoho parametrů, které jsou specifické pro konkrétní válcovaný materiál a jsou spíše odhadovány než přímo známy. Drobné nepřesnosti na vstupech modelu v konečném důsledku vedou na významné chyby modelem vypočtených sil, které se nakonec liší od skutečných změřených během následného výrobního procesu. Z chyb mezi výpočtem a realitou vzniká tzv. *korekční faktor*, tedy hodnota pro úpravu vypočtené přítlačné síly a momentu.

Korekční faktory jsou vypočteny po každém průchodu plechu válcovací stolicí. Nejedná se o konstantu, ale proměnou, která značně mění svou hodnotu na základě parametrů konkrétního válcovacího procesu. Obtížné je i nalézt lineární závislost mezi korekčním faktorem a vstupními parametry válcování.

Cílem práce je navrhnout metodu využívající algoritmy strojového učení k predikci korekčních faktorů z prosté znalosti parametrů polotovaru, výstupu matematicko-fyzikálního modelu a historie předchozích průchodů válcovací stolicí. Predikované korekční faktory lze pak využít k úpravě výsledků matematicko-fyzikálního modelu a tím i k příslušné úpravě výrobního postupu ještě před samotným výrobním procesem.

V sekci 2. (Použité metody) jsou představeny veškeré matematické metody a algoritmy nutné k sestavení prediktivního modelu.

V sekci 3. (Navržený model) je stručně popsán sestavený model včetně zvolených komponent.

V sekci 4. (Implementace) je okomentovaná implementace a to i z pohledu vývoje.

V sekci 5. (Validace a simulace) je graficky zdokumentován proces učení v *offline* režimu, společně s výsledky implementovaného modelu pro konkrétní válcovací stolici RM2.

2. Použité metody

2.1. Normalizace

Normalizace je standartní součástí předzpracování dat. Jestliže jsou mezi naměřenými veličinami určenými pro učení navrhovaného modelu rozdíly velikosti řádů, vyplatí se tyto veličiny normalizovat.

Například *z-scoring* je obvykle používaný algoritmus normalizace, jehož předpis je:

$$x_z = \frac{x - \overline{x}}{\sigma_x} \tag{1}$$

Kde x je naměřená veličina, $\overline{x}$ je průměrná hodnota x a σ_x je směrodatná odchylka.

Tato transformace se často používá v offline předzpracování dat, protože zvyšuje výkonost a robustnost adaptivního algoritmu.

2.2. HONU

HONU neboli *neuronové jednotky vyšších řádů*[1] jsou speciálním případem S-PNN(*Neuronové sítě typu sigma-pi*[2]), které jsou schopné zachytit nejen lineární korelace mezi vstupními a výstupními veličinami, ale také korelace vyšších řádů [1].

Bylo prokázáno, že neuronové jednotky vyšších řádů mají dobré výpočetní a rozpoznávací vlastnosti, stejně tak jako vlastnost dobře se učit [1].

Schéma HONU je popsáno na obr. 1 a výstup je popsán rovnicí:

$$y = \phi(z) \tag{2}$$

Kde $\phi(z)$ je *výstupní funkce*[3].

$$z = col(\mathbf{x})\,\theta = \sum_{j=0}^{n_w-1} \theta_j x_j \tag{3}$$

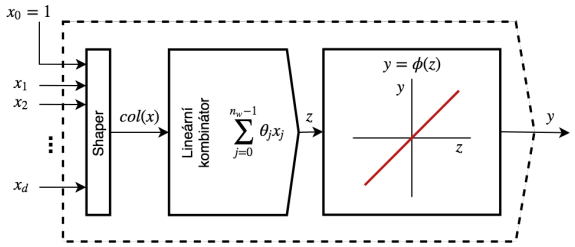
[1] *Higher-order neural units*

[2] *Sigma-pi neural networks*

[3] Něky také nazývaná *aktivační funkcí.*

*Kontakt na autora: Adam.Peichl@fs.cvut.cz

Kde θ je vektor vah[4], $col(\mathbf{x})$ je vektor kombinací vstupů odpovídající řádu HONU a n_w je počet vah HONU, který závisí na počtu vstupů a řádu HONU (tab. 1).



Obr. 1. *Schéma HONU*

2.3. Trénovací metody

Pro trénovaní HONU byly zvoleny a implementovány tyto dvě metody:

1. *Stochastický gradient descent* (SGD)
2. *Levenberg-Marquardtova metoda* (LMA)

Obě dvě metody mohou být užity pro offline *batchové* trénování. Metoda SGD může být použita i pro online ladění (dotrénovávání za chodu).

2.3.1. Stochastický gradient descent

Metoda *Stochastický Gradient Descent* (SGD) je numerický iterativní algoritmus pro hledání funkčního minima[2].

Nechť θ je vektor parametrů *HONU*, matice $\mathbf{X}$ je matice vstupů[5] a $\mathbf{y}$ je vektor výstupů. Dále předpokládejme, že funkce $h_\theta(\mathbf{x})$ je funkcí, jejíž parametry θ chceme optimalizovat:

$$h_\theta(\mathbf{x}) = \sum_{j=0}^{n-1} \theta_j x_j \tag{4}$$

Dále zavedeme funkci *cost*:

$$cost(\theta, (\mathbf{x}^{(i)}, y^{(i)})) = \frac{1}{2}\left[h_\theta(\mathbf{x}^{(i)}) - y^{(i)}\right]^2 \tag{5}$$

A naše chybové kritérium bude ve tvaru:

$$J_\theta = \frac{1}{M}\sum_{i=1}^{M} cost\big(\theta, (\mathbf{x}^{(i)}, y^{(i)})\big) \tag{6}$$

Potom aktualizační pravidlo:

$$\theta^{(l+1)} = \theta^{(l)} - \alpha\frac{\partial}{\partial\theta} cost\big(\theta, (\mathbf{x}^{(l)}, y^{(l)})\big) \tag{7}$$

$$\theta^{(l+1)} = \theta^{(l)} - \alpha\big(h_\theta(\mathbf{x}^{(l)}) - y^{(l)}\big)\mathbf{x}^{(l)} \tag{8}$$

kde l značí iteraci a parametr α ovlivňuje délku kroku. Nejdůležitější součástí této metody[6] je proces *náhodného výběru* před každou iterací. Lépe je tento postup znázorněn v pseudokódu algoritmu 1 (SGD).

Algoritmus 1 Stochastický gradient descent

```
function SGD(X, y, α, l_max)
    inicializace θ
    for l ← 1, ..., l_max do
        (x^(l), y^(l)) ← náhodný výběr z (X, y)
        e ← x^(l) θ − y^(l)
        θ ← θ − α e x^(l)
    return θ
```

2.3.2. Levenberg-Marquardtova metoda

Levenberg-Marquardtova metoda je numerický iterativní algoritmus pro hledání minima funkce, která kombinuje *Gauss-Newtonovu metodu* a *metodu nejmenšího spádu* tak, aby byly eliminovány možné problémy *Gauss-Newtonovy metody* daleko od extrému [3].

Tato metoda v sobě kombinuje klíčové vlastnosti jako jsou robustnost a efektivnost a dále také silné vlastnosti konvergence a numerické stability [4]. Aktualizační pravidlo při použití metody LMA je:

$$\theta^{(l+1)} = \theta^{(l)} + \left[\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right]^{-1}\mathbf{X}\mathbf{e} \tag{9}$$

Kde λ je *tlumící faktor*[7] a v podstatě určuje poměr metod, které LMA kombinuje, $\mathbf{X}$ je matice vstupů, $\mathbf{I}$ je jednotková matice a $\mathbf{e}$ je vektor chyb našeho modelu podle:

$$\mathbf{e} = \mathbf{X}\,\theta - \mathbf{y} \tag{10}$$

Kde $\mathbf{y}$ je vektor výstupů.

Algoritmus 2 Levenberg-Marquardt

```
function LMA(X, y, λ, l_max)
    inicializace θ
    for l ← 1, ..., l_max do
        e ← Xθ − y
        Δθ ← [X^T X + λI]^−1 X e
        θ ← θ + Δθ
    return θ
```

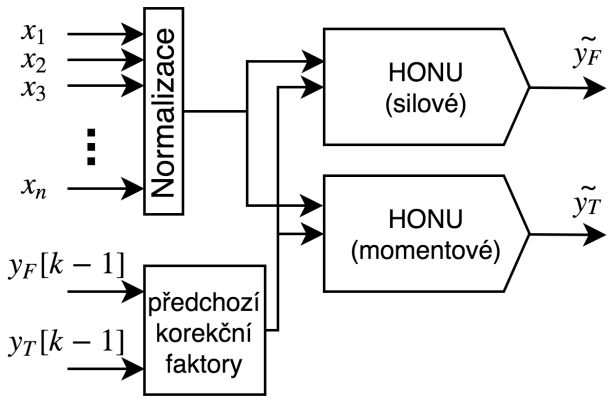
3. Navržený model

Pro každou válcovací stolici byl navržen jednoduchý model skládající se ze dvou HONU (silové a momentové), normalizace (z-scoring) a případně historie korekčních faktorů (obr. 2).

[4] Parametry systému, v našem případě HONU.
[5] Jeden řádek matice $\mathbf{X}$ odpovídá jedné realizaci vstupů. Tomu samozřejmě odpovídá řádek sloupcového vektoru $\mathbf{y}$.
[6] Proto se tato metoda jmenuje *Stochastic Gradient Descent*, ze slova stochastický nebo také náhodný.
[7] Dumping factor

Tabulka 1. *Řády HONU*

Řád	Název	Používaná zkratka	Počet vstupů	Počet vah
1.	Lineární neuronová jednotka	LNU	d	$d+1$
2.	Kvadratická neuronová jednotka	QNU	d	$\frac{(d+1)(d+2)}{2}$
3.	Kubická neuronová jednotka	CNU	d	$\sum_{j=0}^{3}\binom{(d+j)}{j}$
N-tá	Neuronová jednotka N-tého řádu	-	d	$\sum_{j=0}^{N}\binom{(d+j)}{j}$



Obr. 2. *Schéma navrženého modelu*

Jako HONU bylo zvoleno QNU (kvadratická neuronová jednotka) s lineární výstupní funkcí, což je kompromis mezi složitostí a výpočetní náročností.

4. Implementace

Pro rychlý vývoj, simulace a vizualizace byl použit programovací jazyk *Python*, speciálně následující knihovny:

1. *NumPy* - knihovna pro vědecké výpočty v jazyce Python, která kromě jiných obsahuje mimo jiné: algoritmy pro lineární algebru, sofistikované nástroje pro integraci C a C++ kódu a modul pro generování náhodných čísel [5].
2. *Pandas* - knihovna která umožňuje manipulaci s velkým objemem dat a jejich analýzu [6].
3. *Matplotlib* - knihovna pro vizualizaci dat [7].
4. *Seaborn* - knihovna pro statistické vizualizace [8].

Pro finální verzi modulu byl použit jazyk C++, který je kompilovaný a rychlejší než python, přičemž byla využita knihovna *Eigen*[9].

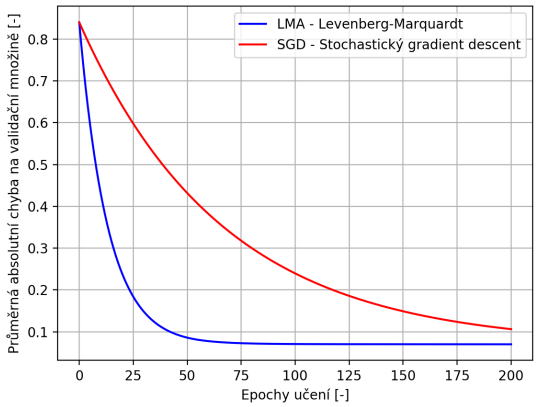
5. Validace a simulace

5.1. Učení

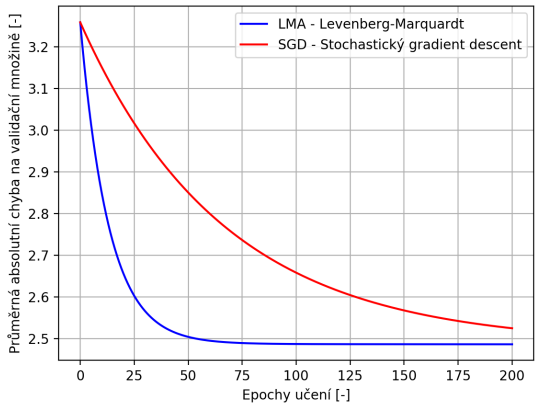
Bylo potřeba implementovat dva učící algoritmy (SGD a LMA), jejich srovnání je vidět na obrázcích 3 (pro silové QNU) a 4 (pro momentové QNU).

Na obrázcích 3 a 4 je vidět, že metoda LMA rychleji konverguje. Proto bude výhodnější pro použití v *offline* režimu (předtrénování). Tato metoda však není použitelná na *online* učení (učení „za běhu“).

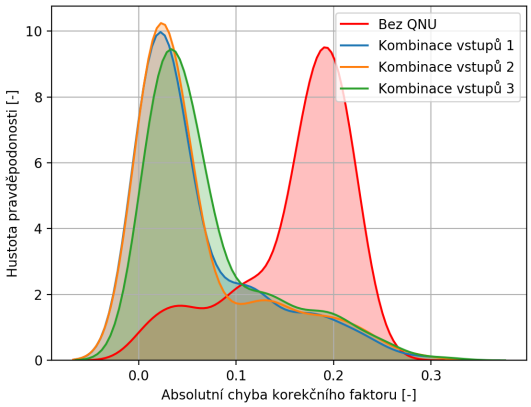
Na rozdíl od metody LMA je metoda SGD použitelná v obou případech a je výhodné ji využívat na jemné doladění modelu v online režimu.



Obr. 3. *Porovnání metod učení v offline režimu pro silové HONU válcovací stolice RM2. Trénovací matice obsahuje 600 párů typu vstup-výstup, 200 učících epoch, $\alpha = 0.005$ - pro SGD, $\lambda = 0.005$ - pro metodu LMA, validační množina obsahuje 200 párů typu vstup-výstup.*



Obr. 4. *Porovnání metod učení v offline režimu pro momentové HONU válcovací stolice RM2. Trénovací matice obsahuje 600 párů typu vstup-výstup, 200 učících epoch, $\alpha = 0.005$ - pro SGD, $\lambda = 0.005$ - pro metodu LMA, validační množina obsahuje 200 párů typu vstup-výstup.*



Obr. 5. *Porovnání hustoty pravděpodobností absolutních chyb jednotlivých kombinací vstupů pro silové QNU, které jsou uvedeny v tab. 2 pro válcovací stolici RM2 vykreslené pomocí* jádrového odhadu hustoty pravděpodobnosti.

5.2. Porovnání modelů podle kombinací vstupů

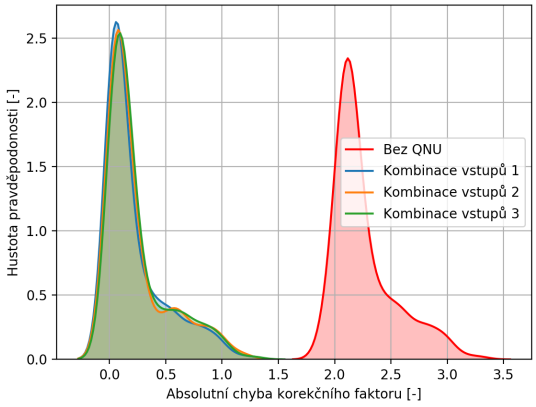
V následující subkapitole jsou porovnány modely pro tři různé kombinace vstupů (tab 2). Tyto vstupy byly doporučeny jako vstupy, které by měly mít s nejvyšší pravděpodobností vliv na výpočet korekčních faktorů.

Tabulka 2. *Použité kombinace vstupů*

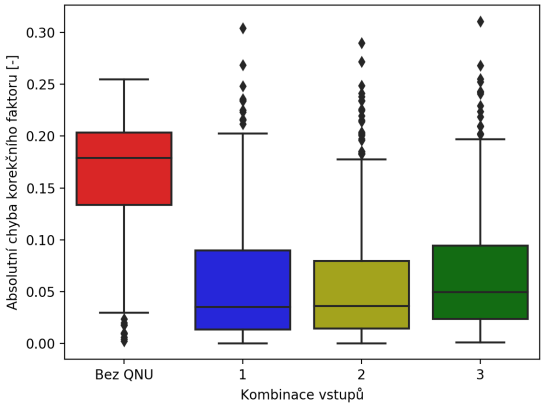
Název kombinace	Použité vstupy
Kombinace vstupů 1	Šířka polotovaru Teplota materiálu Namáhání Stupeň namáhání Poměr
Kombinace vstupů 2	Šířka polotovaru Teplota materiálu Namáhání Stupeň namáhání Poměr Předchozí korekční faktor
Kombinace vstupů 3	Šířka polotovaru Tloušťka při vstupu Tloušťka při opuštění Rychlost Teplota materiálu

V následujících grafech je vidět rozložení absolutní chyby pro jednotlivé kombinace vstupů pro válcovací stolici RM2 v porovnání s přístutupem bez našeho modelu.

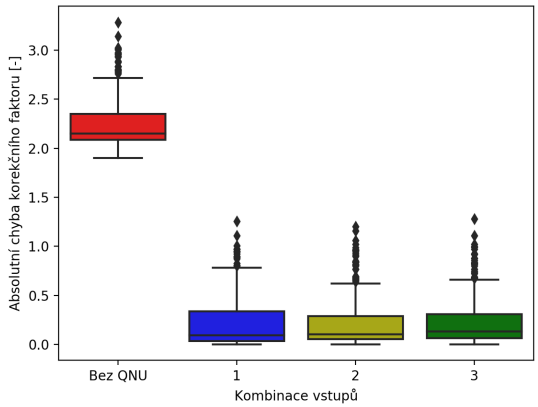
V grafech 5 a 6 je vidět pravdivostní rozložení vytvořené pomocí *jádrového odhadu hustoty pravděpodobnosti*. V grafech 7 a 8 je absolutní chyba vizualizována pomocí krabicového grafu.



Obr. 6. *Porovnání hustoty pravděpodobností absolutních chyb jednotlivých kombinací vstupů pro momentové QNU, které jsou uvedeny v tab. 2 pro válcovací stolici RM2 vykreslené pomocí* jádrového odhadu hustoty pravděpodobnosti



Obr. 7. *Porovnání absolutních chyb jednotlivých kombinací vstupů pro silové QNU, které jsou uvedeny v tab. 2 pro válcovací stolici RM2 vykreslené pomocí* krabicového grafu.



Obr. 8. *Porovnání absolutních chyb jednotlivých kombinací vstupů pro momentové QNU, které jsou uvedeny v tab. 2 pro válcovací stolici RM2 vykreslené pomocí* krabicového grafu.

6. Závěr

Vytvořený model složený z normalizace (z-scoring), dvou QNU (silové a momentové) a historie korekčních parametrů prokazatelně snižuje chybu matematicko-fyzikálního modelu pro válcování. (obr. 5, 6, 7 a 8)

Vzhledem k přísným požadavkům na nasazení do válcovacích linek, je náš model robustní a dokáže se adaptovat v online řežimu.

Celý výpočetní modul je napsaný jako hlavičková knihovna v C++ a momentálně se plánuje jeho nasazení.

Je zde také značný prostor pro budoucí vylepšení. Nyní je pro každou válcovací stolici a pro každý materiál nutné mít vlastní soustavu dvou QNU. Byly proto navrženy dvě budoucí cesty vylepšení:

1. **Clustery** - Bylo by možné materiály automaticky třídit do jednotlivých clusterů podle různých specifikací. Každý cluster bude mít soustavu dvou QNU, která predikují korekční parametry pro sílu a moment.
2. **SOM** - Neboli samoorganizační mapy - metoda na automatickou výstavbu clusterů.

Poděkování

Tato práce byla podpořena grantem Studentské grantové soutěže ČVUT č. *SGS18/177/OHK2/3T/12.*

Autoři děkují za spolupráci firmě PT SOLUTIONS WORLDWIDE spol. s r.o.

Všechny simulace byly provedeny v jazyce *Python*, výsledný modul je ve formě *hlavičkové knihovny C++*. Zdrojové kódy všech algoritmů simulací je možné dostat na požádání od autora.

Literatura

[1] Madan Gupta, Liang Jin a Noriyasu Homma. *Static and dynamic neural networks: from fundamentals to advanced theory*. John Wiley & Sons, 2004.

[2] Léon Bottou, Frank E Curtis a Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *SIAM Review* 60.2 (2018), s. 223–311.

[3] Jiří Limpouch. *Levenberg-Marquardtova - metoda*. 2000. Dostupné z: http://kfe.fjfi.cvut.cz/~limpouch/numet/extrem/node12.html (cit. 26. 03. 2019).

[4] Jorge J Moré. "The Levenberg-Marquardt algorithm: implementation and theory". In: *Numerical analysis*. Springer, 1978, s. 105–116.

[5] *Numpy - knihovna pro vědecké výpočty a lineární algebru*. Dostupné z: http://www.numpy.org/ (cit. 20. 03. 2019).

[6] *Pandas - Knihovna pro datovou analýzu*. Dostupné z: https://pandas.pydata.org/ (cit. 20. 03. 2019).

[7] *Matplotlib - knihovna pro vizualizace*. Dostupné z: https://matplotlib.org/ (cit. 20. 03. 2019).

[8] *Seaborn - knihovna pro statistické vizualizace*. Dostupné z: https://seaborn.pydata.org/ (cit. 20. 03. 2019).

[9] *Eigen dense - hlavičková knihovna C++ pro lineární algebru*. Dostupné z: http://eigen.tuxfamily.org/ (cit. 21. 03. 2019).