



NEUROVERKKOJEN HYÖDYNTÄMINEN DIFFERENTIAALIYHTÄLÖIDEN RATKAISEMISESSA

Lappeenrannan–Lahden teknillinen yliopisto LUT

Laskennallisen tekniikan kandidaatintyö

2024

Mette Suvanto

Tarkastaja: Prof. Tapio Helin

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT
School of Engineering Science
Laskennallinen tekniikka

Mette Suvanto

Neuroverkkojen hyödyntäminen differentiaaliyhtälöiden ratkaisemisessa

Kandidaatintyö

2024

21 sivua, 9 kuvaa

Tarkastaja: Prof. Tapio Helin

Hakusanat: neuroverkko, differentiaaliyhtälö, syväoppiminen

Differentiaaliyhtälöitä voidaan hyödyntää monien oikean elämän ongelmien, kuten radioaktiivisen hajoamisen ja aaltoyhtälön, mallintamisessa. Täten niiden ratkaisemiseen on tärkeää löytää tehokkaita ratkaisutapoja. Neuroverkkoja hyödynnetään monien ongelmien ratkaisemisessa, kuten kääntäjässä, puheentunnistuksessa ja differentiaaliyhtälöissä. Eteenpäin kytketyt neuroverkot ovat erityisen käyttökelpoisia koneoppimisessa.

Työn tavoitteena on esitellä, kuinka neuroverkkoja voidaan hyödyntää differentiaaliyhtälöiden ratkaisemisessa. Menetelmää havainnollistetaan ratkaisemalla ensimmäisen kertaluvun differentiaaliyhtälö $y' = -2xy + x$ alkuarvolla $y(0) = 1$ Matlab-ympäristössä. Työssä käytettävä neuroverkko on MLP (multilayer perceptron) yhdellä piilokerroksella, jossa on kymmenen piilokerroksen neuronia. Differentiaaliyhtälön ratkaisemiseen käytettävä neuroverkko luotiin Matlabin Deep Learning -työkalun avulla. Neuroverkon antama tulos on luotettava, kun opetusdataa annetaan tarpeeksi monta kappaletta ja tarpeeksi pitkältä väliltä. Esimerkin tapauksessa neuroverkon antama tulos oli luotettava välillä $[0, 4]$, kun opetusdataa annettiin välillä $[0, 3]$ ja opetusdatapisteitä oli 1000 kappaletta. Neuroverkot on siis tehokas tapa ratkaista differentiaaliyhtälöitä hyvällä tarkkuudella.

SYMBOLI- JA LYHENNELUETTELO

σ	sigmoid-funktio
\mathbf{a}	piilokerroksen arvot
$\mathbf{b}^{[1]}$	vakiokertoimet syötekerroksen ja piilokerroksen neuroneiden välillä
$\mathbf{b}^{[2]}$	vakiokertoimet piilokerroksen ja tuloskerroksen neuroneiden välillä
h_{θ}	neuroverkon antama tulos
$W^{[1]}$	painokertoimet syötekerroksesta piilokerrokseen
$W^{[2]}$	painokertoimet piilokerroksesta tuloskerrokseen
\mathbf{x}	syötekerroksen arvot
\mathbf{z}	painotetut summat
BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm
GD	gradient descent
MLP	multilayer perceptron
SGD	stochastic gradient descent
SGDM	stochastic gradient descent with momentum
SLP	single layer perceptron

SISÄLLYSLUETTELO

1 JOHDANTO	5
1.1 Tausta	5
1.2 Tavoitteet, rajaus ja rakenne	6
2 NEUROVERKOT	7
2.1 Perseptroni	8
2.2 MLP yhdellä piilokerroksella	9
2.3 Neuroverkon opettaminen	10
3 DIFFERENTIAALIYHTÄLÖIDEN RATKAISEMINEN NEUROVERKON AVULLA	12
4 SIMULAATIOT	13
4.1 Ohjelmistot	13
4.2 Menetelmät	13
4.3 Tulokset	13
5 YHTEENVETO	20
LÄHTEET	21

1 JOHDANTO

1.1 Tausta

Neuroverkkoja voidaan hyödyntää monenlaisten ongelmien ratkaisemiseen. Tällaisia ongelmia ovat esimerkiksi kääntäjä, chatbot ja puheentunnistus [1]. Erityisesti eteenpäin kytetyt neuroverkot ovat tärkeä osa-alue koneoppimisessa [2]. Eteenpäinkytketyissä neuroverkoissa neuronien väliset yhteydet kulkevat ainoastaan eteenpäin. Esimerkiksi konvoluutioneuroverkkoja, jotka ovat erityistapaus eteenpäinkytketyistä neuroverkoista, käytetään objektintunnistamiseen kuvasta [2].

Differentiaaliyhtälöitä käytetään mallintamaan monia oikean elämän ongelmia, kuten erilaisia fysikaalisia ilmiöitä, joita ovat esimerkiksi aaltoyhtälö, Newtonin toinen laki ja radioaktiivinen hajoaminen [3]. Differentiaaliyhtälöiden ratkaisemiseen on olemassa useita numeerisia menetelmiä, kuten Eulerin menetelmä ja Runge-Kutta-menetelmät [3]. Oletetaan, että kappale on hetkellä x_0 kohdassa y_0 ja kulkee nopeudella y'_0 . Lyhyessä ajassa nopeus ei kerkeä muuttumaan merkittävästi alkuperäisestä nopeudesta y'_0 , jolloin Eulerin menetelmällä seuraava piste y_1 saadaan, kun kerrotaan nopeus y'_0 matkaan käytetyllä ajalla $h = x_1 - x_0$ ja lisätään tämä edelliseen kohtaan y_0 [4]. Eulerin menetelmä voidaan siis ilmaista kaavalla

$$y_{i+1} = y_i + hy'_i. \quad (1)$$

Tunnetuimmassa Runge-Kutta-menetelmässä seuraava piste saadaan laskettua kaavalla

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad (2)$$

missä

$$k_1 = hf(x_i, y_i), \quad (3)$$

$$k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right), \quad (4)$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right), \quad (5)$$

$$k_4 = hf(x_i + h, y_i + k_3), \quad (6)$$

ja missä funktio $f(x, y)$ on differentiaaliyhtälön

$$\frac{dy}{dx} = f(x, y). \quad (7)$$

oikea puoli [5].

Differentiaaliyhtälöitä voidaan ratkaista myös neuroverkkojen avulla. Tämän menetelmän hyötyjä ovat esimerkiksi se, että ratkaisu on derivoituva ja suljetussa analyttisessä muodossa [3]. Muilla numeerisilla tavoilla ratkaisu on usein diskreetissä muodossa tai sitä voidaan derivoida vain rajoitetusti [3]. Lisäksi laskennallinen vaativuus kasvaa hitaasti neuroverkkoja hyödyntämällä [3].

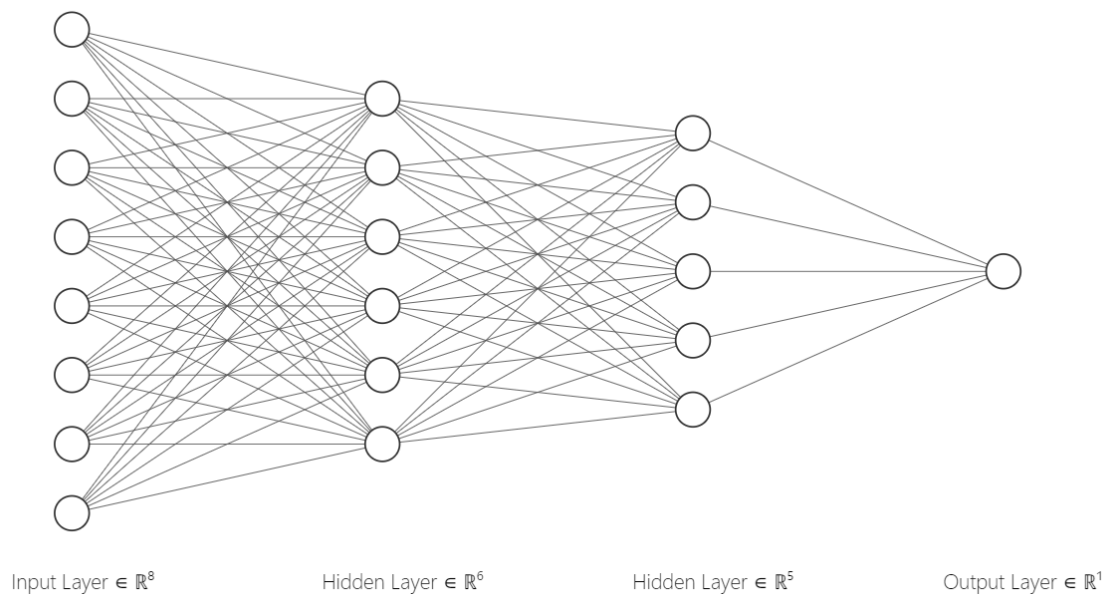
1.2 Tavoitteet, raja- ja rakenne

Tässä työssä esitellään neuroverkkojen toimintaa ja selvitetään, kuinka neuroverkkoja voidaan hyödyntää differentiaaliyhtälöiden ratkaisemisessa. Työn tavoitteena on esitellä tapa, jolla voidaan ratkaista ensimmäisen kertaluvun differentiaaliyhtälö eli ODE (ordinary differential equation) eteenpäin kytketyllä monikerroksisella perseptroniverkolla eli MLP:llä (multilayer perceptron), jossa on yksi piilokerros. Esimerkkinä ratkaistaan differentiaaliyhtälö $y' = -2xy + x$ alkuehdolla $y(0) = 1$ ja kyseisen ODE:n tapauksessa toteutetaan simulaatioita Matlab-ohjelmistolla hyödyntäen Deep Learning -työkalua. Deep Learning -työkalulla voidaan luoda halutunlainen neuroverkko, jolla differentiaaliyhtälö ratkaistaan.

Tässä työssä luvussa 2 esitellään neuroverkkoja yleisesti ja käsitellään matemaattisesti perseptronin ja MLP:n rakenne. Lisäksi esitellään, kuinka neuroverkkoja voidaan opettaa. Luvussa 3 tarkastellaan, kuinka neuroverkkoja voidaan hyödyntää differentiaaliyhtälöiden ratkaisemisessa. Simulaatioissa hyödynnetyt ohjelmistot ja menetelmät sekä simulaatioilla saadut tulokset esitellään luvussa 4. Luvussa 5 tehdään yhteenveto työn aiheesta ja saaduista tuloksista.

2 NEUROVERKOT

Neuroverkko on mukautuva matemaattinen malli, jonka rakenne perustuu biologisten aivojen rakenteeseen [6]. Neuroverkot ovat yksi syväoppimisen muoto [2]. Neuroverkko koostuu syötekerroksesta, piilokerroksista ja tuloskerroksesta [6]. Tätä neuroverkon perusrakennetta on havainnollistettu kuvassa 1. Neuroverkkoa opettamalla pyritään löytämään syöteen ja tulosteen välinen yhteys [6] eli funktio, joka kuvaa syöteen tulosteeksi. Oppiminen tapahtuu syöttämällä neuroverkolle opetusdataa, jonka avulla neuronien välisten yhteyksien painokertoimia optimoidaan siten, että neuroverkon antaman tuloksen virhe minimoituu [6].



Kuva 1. Esimerkki täysin kytketystä neuroverkosta. Kyseisessä neuroverkossa on syötekerros, kaksi piilokerrosta ja tuloskerros.

Tämän työn kannalta oleellinen neuroverkko differentiaaliyhtälöiden ratkaisemiseen on eteenpäin kytketty neuroverkko eli MLP yhdellä piilokerroksella, koska tämän tyyppistä neuroverkkoa voidaan hyödyntää minkä tahansa funktion approksimointiin mielivaltaisella tarkkuudella [7].

2.1 Perseptroni

Yksinkertainen neuroverkko on nimeltään perseptroni eli SLP (single layer perceptron). SLP rakentuu syötekerroksen neuroneista $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$, jotka ovat yhteydessä tuloskerroksen neuroneihin [6]. Näiden neuronien yhteyksiä kuvataan painokertoimilla $\mathbf{w} = [w_1, w_2, \dots, w_d]^T$. SLP toimii siten, että se vastaanottaa syötteen, joista lasketaan painotettu summa ja lisätään vakiotermi b

$$z = x_1w_1 + x_2w_2 + \dots + x_dw_d + b = \sum_{i=1}^d (x_iw_i) + b = \mathbf{w}^T \mathbf{x} + b. \quad (8)$$

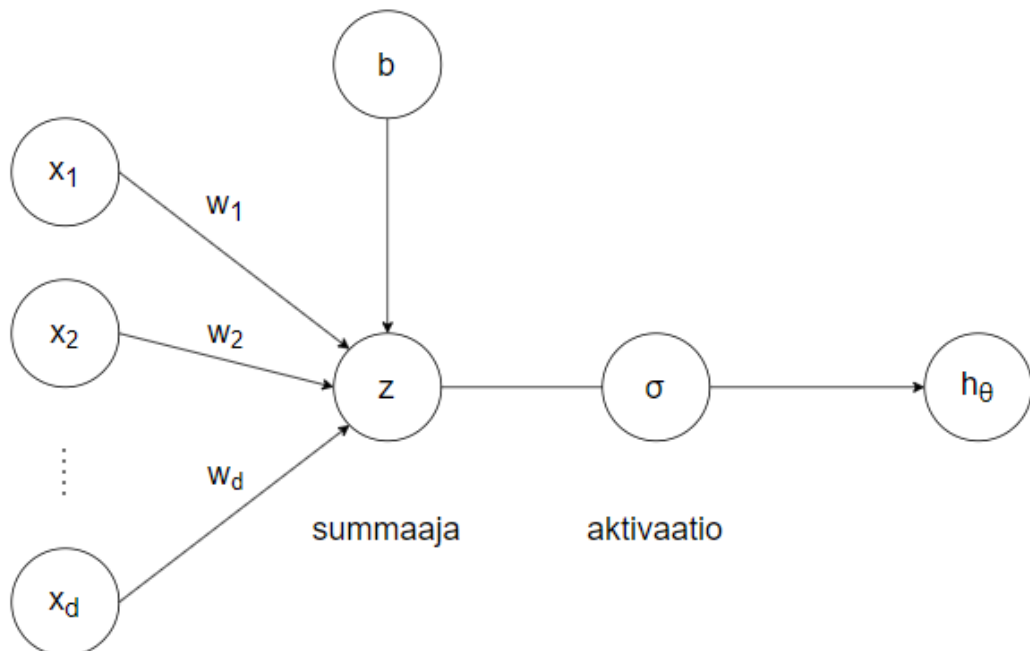
Tästä summasta otetaan epälineaarinen aktivaatio, jolloin saadaan tulos

$$h_{\theta}(\mathbf{x}) = \sigma(z) = \sigma(\mathbf{w}^T \mathbf{x} + b), \quad (9)$$

missä

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (10)$$

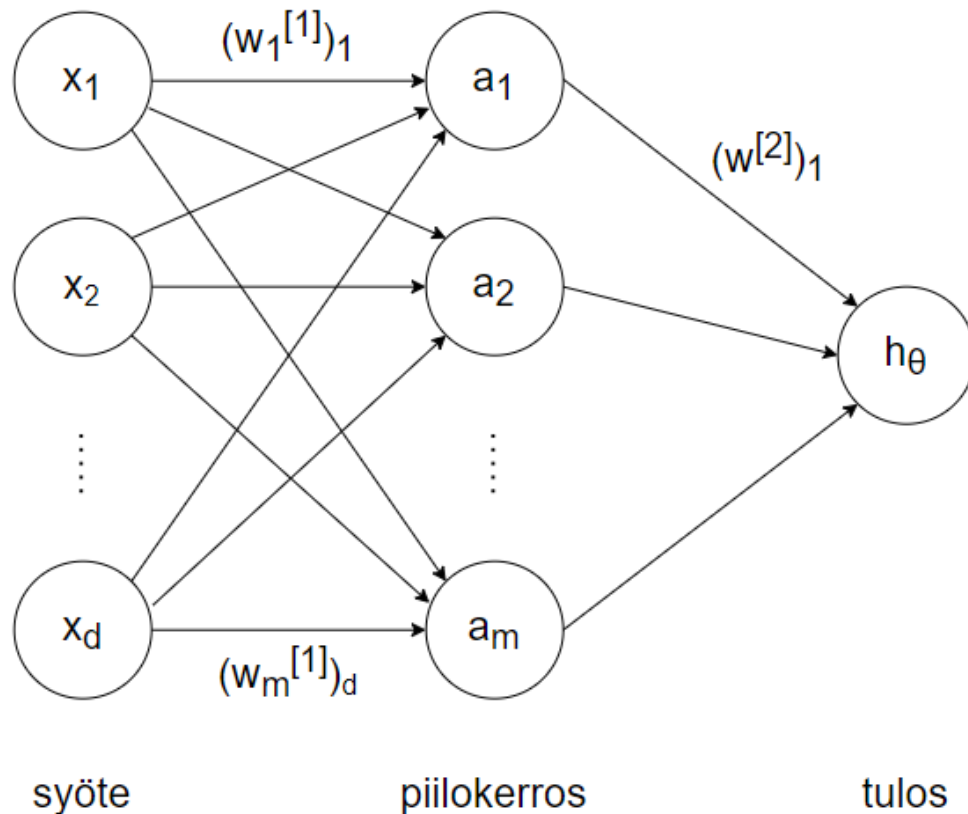
on aktivaatiofunktiona toimiva sigmoid-funktio. SLP:n rakennetta on havainnollistettu kuvassa 2.



Kuva 2. Perseptronin rakenne [6].

2.2 MLP yhdellä piilokerroksella

SLP voi oppia ainoastaan lineaarisia funktioita, joten differentiaaliyhtälöiden ratkaisemiseen tarvitaan monikerroksinen perseptroniverkko eli MLP. MLP koostuu syötekerroksesta, piilokerroksista ja tuloskerroksesta. On havaittu, että yhden piilokerroksen neuroverkolla pystytään ratkaisemaan differentiaaliyhtälöitä tehokkaasti ja hyvällä tarkkuudella [7], joten käsitellään tällaista yhdellä piilokerroksella varustettua MLP:tä. Tällaisen neuroverkon rakennetta on havainnollistettu kuvassa 3.



Kuva 3. MLP yhdellä piilokerroksella.

MLP verkkoa, joka on täysin kytketty ja jossa on yksi piilokerros, kuvaa yhtälö

$$h_\theta = \sigma(\mathbf{w}^{[2]T} \mathbf{a} + b^{[2]}), \quad (11)$$

missä $b^{[2]}$ on vakiotermi, $\mathbf{w}^{[2]} = [w_1^{[2]}, w_2^{[2]}, \dots, w_m^{[2]}]^T$ painokertoimet piilokerroksesta

tuloskerroksen neuroniiin ja

$$a_1 = \sigma(z_1) = \sigma(\mathbf{w}_1^{[1]T} \mathbf{x} + b_1^{[1]}) \quad (12)$$

$$a_2 = \sigma(z_2) = \sigma(\mathbf{w}_2^{[1]T} \mathbf{x} + b_2^{[1]}) \quad (13)$$

$$\vdots \quad (14)$$

$$a_m = \sigma(z_m) = \sigma(\mathbf{w}_m^{[1]T} \mathbf{x} + b_m^{[1]}) \quad (15)$$

$$(16)$$

ovat piilokerroksen neuronien saamat arvot, missä $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$ on syötekerroksen arvot ja $\mathbf{w}_1^{[1]} = [(w_1^{[1]})_1, (w_1^{[1]})_2, \dots, (w_1^{[1]})_d]^T$, $\mathbf{w}_2^{[1]} = [(w_2^{[1]})_1, (w_2^{[1]})_2, \dots, (w_2^{[1]})_d]^T$ jne. ovat painokertoimet syötekerroksesta piilokerroksen neuroneihin. Edellä esitellyt yhtälöt voidaan esittää myös matriisimuodossa. Olkoon

$$W^{[1]} = \begin{bmatrix} \mathbf{w}_1^{[1]T} \\ \mathbf{w}_2^{[1]T} \\ \vdots \\ \mathbf{w}_m^{[1]T} \end{bmatrix}, \mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_m^{[1]} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, W^{[2]} = [\mathbf{w}^{[2]T}], \quad (17)$$

jolloin piilokerroksen arvot ja tuloskerroksen arvo saadaan yhtälöiden

$$\mathbf{a} = \sigma(W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}) \quad (18)$$

$$h_\theta(\mathbf{x}) = W^{[2]}\mathbf{a} + \mathbf{b}^{[2]} \quad (19)$$

avulla.

2.3 Neuroverkon opettaminen

Neuroverkkoa tulee opettaa, jotta sitä voidaan käyttää halutun ongelman ratkaisemiseen. Opettamiseen käytetään joukkoa datapisteitä $\{(x_i, y_i)\}_{i=1}^n$, missä x_i on syöte ja y_i tuloste syötteen x_i arvolla. Neuroverkolle syötetään opetusdataa ja annetun datan perusteella pyritään optimoimaan neuroverkon parametreja siten, että neuroverkko approksimoi ratkaisua mahdollisimman hyvin. Pyritään siis minimoimaan neuroverkon antaman ratkaisun virhettä suhteessa analyttiseen ratkaisuun.

Neuroverkon antaman ratkaisun virhettä kuvataan jollakin hukkafunktiolla, jota pyritään minimoimaan. Hukkafunktion minimointiin on olemassa useita eri menetelmiä, esimerkiksi conjugate gradient method, quasi-Newton BFGS (Broyden–Fletcher–Goldfarb–Shanno algorithm) ja eri versiot GD-menetelmästä (gradient descent). Yleensä hukkafunktiona käytetään jonkinlaista pienimmän neliösumman funktiota. Hukkafunktion minimointiin käytetään monesti SGD-menetelmää (stochastic gradient descent) tai jotakin GD-menetelmän variaatiota, koska GD-menetelmät on helppo toteuttaa ja toimii yleensä tehokkaasti [8]. Tässä työssä simulaatioiden hukkafunktion minimoimiseen käytetään SGDM-menetelmää (stochastic gradient descent with momentum).

GD-menetelmät perustuvat siihen, että pyritään minimoimaan hukkafunktiota $f(\mathbf{x})$ gradientin avulla [2]. Minimikohdan löytämiseksi funktiolle $f(\mathbf{x})$ lasketaan gradientti pisteessä \mathbf{x} ja liikutaan pienen askeleen ϵ verran gradientin vastaiseen suuntaan, jolloin saadaan uusi piste $\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$ [2]. Tässä ϵ on opetusnopeus, positiivinen skalaari, mikä määrittää otetun askeleen suuruuden [2]. Askeleita otetaan niin kauan, kunnes $\nabla_{\mathbf{x}} f(\mathbf{x}) \approx 0$, jolloin ollaan löydetty minimikohta [2]. Menetelmällä löydetty minimikohta ei välttämättä ole globaali minimi, mutta menetelmällä pyritään löytämään minimikohdan, jossa funktion arvo on tarpeeksi lähellä globaalia minimiä [2].

SGD-menetelmä on stokastinen approksimaatio GD-menetelmästä [6]. Siinä sen sijaan, että gradientti laskettaisiin koko datasta, lasketaan gradientti satunnaisesti valitusta osajoukosta [6]. Työssä käytettävä SGDM-menetelmä perustuu SGD-menetelmään, mutta SGDM-menetelmällä löydetään minimikohta nopeammin, kuin SGD-menetelmällä. Gradientin laskemiseen näissä menetelmissä käytetään yleensä vastavirta-algoritmia.

3 DIFFERENTIAALIYHTÄLÖIDEN RATKAISEMINEN NEUROVERKON AVULLA

Differentiaaliyhtälön ratkaisuun käytettävä malli voidaan esittää kahden termin summana

$$y_\theta(x) = A(x) + F(x, h_\theta(x)), \quad (20)$$

missä ensimmäinen termi $A(x)$ sisältää differentiaaliyhtälön alkuehdon ja toinen termi $F(x, h_\theta(x))$ sisältää neuroverkon h_θ , jonka parametreja θ optimoidaan siten, että neuroverkon antama ratkaisu olisi mahdollisimman lähellä differentiaaliyhtälön analyyttistä ratkaisua [7]. Ensimmäisen kertaluvun ODE:n

$$\frac{dy}{dx} = f(x, y) \quad (21)$$

tapauksessa yhtälö (20) saa muodon

$$y_\theta(x) = A + xh_\theta(x), \quad (22)$$

missä A on alkuehto ja $h_\theta(x)$ opetettava neuroverkko [7].

Tässä työssä hukkafunktiona käytetty funktio on muotoa

$$L_\theta(x) = \|y'_\theta - f(x, y_\theta)\|^2 + k \|y_\theta(0) - y(0)\|^2, \quad (23)$$

missä θ on neuroverkon parametrit, k vakiokerroin, y_θ neuroverkon ennustama ratkaisu ja y'_θ ennustetun ratkaisun derivaatta. Termi $\|y'_\theta - f(x, y_\theta)\|^2$ kuvaa sitä, kuinka paljon ennustettu tulos eroaa alkuperäisestä differentiaaliyhtälöstä. Termi $\|y_\theta(0) - y(0)\|^2$ kuvaa sitä, kuinka paljon ennustetun ratkaisun alkuarvo eroaa alkuperäisestä alkuarvosta. Hukkafunktion minimointi toteutetaan SGDM-menetelmällä. SGDM-menetelmässä gradientin laskemiseen käytetään Matlabin valmista rutiinia.

Yllä esiteltyä menetelmää voidaan soveltaa myös korkeamman kertaluvun ODE:n, PDE:n tai ODE-ryhmien ratkaisemiseen [7]. Tällöin yhtälö (20) saa erilaisen muodon, mitä ensimmäisen kertaluvun ODE:n tapauksessa.

4 SIMULAATIOT

4.1 Ohjelmistot

Neuroverkon luominen ja opettaminen differentiaaliyhtälön ratkaisemiseen tehtiin Matlab-ohjelmistolla hyödyntäen Deep Learning -työkalua. Neuroverkon kerrokset luotiin käyttämällä featureInputLayer-funktiota luomaan syötekerros, fullyConnectedLayer-funktiota luomaan piilokerros ja sigmoidLayer-funktiota toteuttamaan kerrosten välinen aktivaatio. Tämän jälkeen neuroverkko luotiin aikaisemmista kerroksista dlnetwork-funktiolla.

4.2 Menetelmät

Differentiaaliyhtälön ratkaisemiseen käytettiin eteenpäin kytkettyä neuroverkkoa, joka on täysin kytketty ja jossa on yksi piilokerros. Täysin kytketyssä neuroverkossa kaikki edellisen kerroksen neuronit on yhdistetty kaikkiin seuraavan kerroksen neuroneihin. Piilokerroksessa käytettiin 10 neuronia. Syötekerroksen ja piilokerroksen välisenä aktivaatiofunktiona käytettiin sigmoid-funktiota, joka on määritelty yhtälössä (10). Hukkafunktiona käytettiin kaavassa (23) esitettyä funktiota. Hukkafunktion minimointiin käytettiin SGDM menetelmää, mikä toteutettiin Matlabin valmiilla rutiinilla.

Simulaatiot toteutettiin alkuarvotehtävän

$$y' = -2xy + x, \quad y(0) = 1 \quad (24)$$

tapuksessa. Neuroverkon avulla saatua ratkaisua verrattiin alkuarvotehtävän (24) analyttiseen ratkaisuun

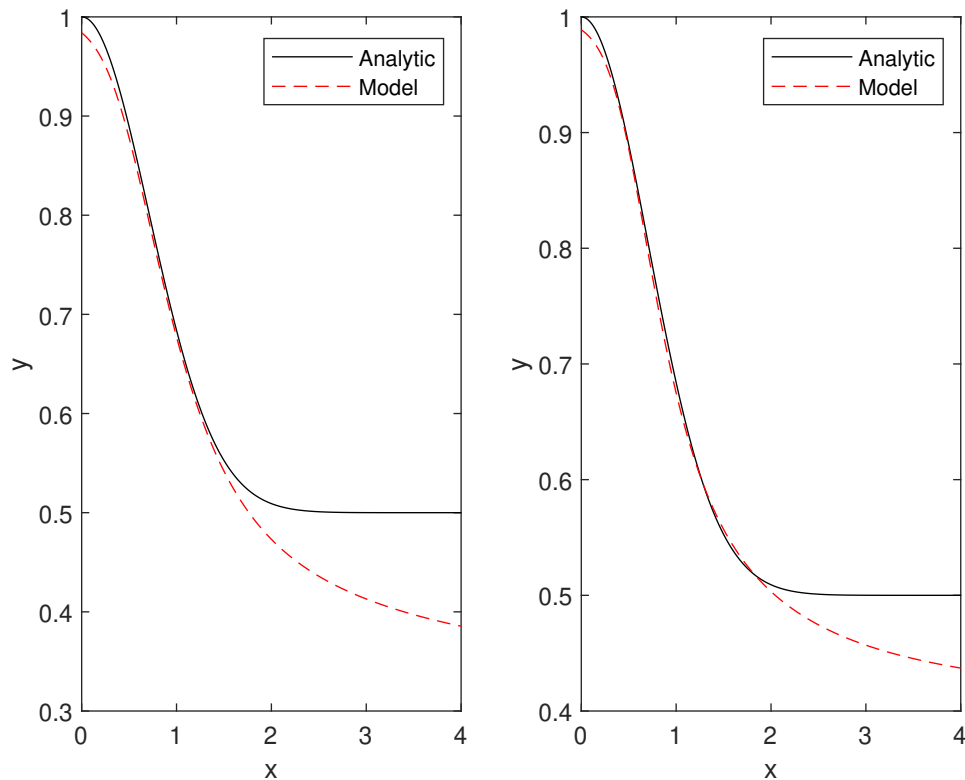
$$y = \frac{1}{2}e^{-x^2} + \frac{1}{2}.$$

4.3 Tulokset

Kuvassa 4 vasemmalla nähdään neuroverkon antama tulos verrattuna analyttiseen ratkaisuun, kun opetusdata on annettu välillä $[0, 1]$ ja datapisteitä on 1000 kappaletta. Neuroverkko ennustaa siis välin $[1, 4]$ arvot. Kuvasta huomataan, että neuroverkko mallintaa ODE:n ratkaisua huonosti, kun opetusdataa on annettu vain välillä $[0, 1]$. Neuroverkon ennustetta kokeiltiin parantaa syöttämällä datapisteitä tiheämmin välillä $[0, 1]$ ja lisäämällä

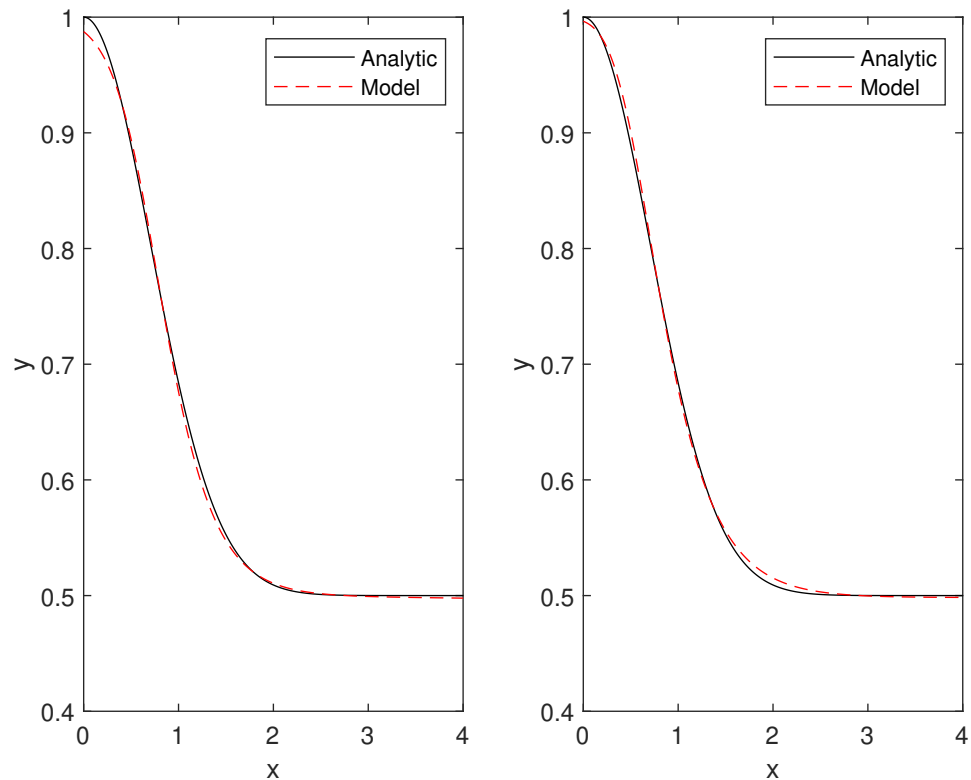
hukkafunktion minimoimiseen käytettävien kierrosten määrää, mutta näillä ei ollut ratkaisuun juurikaan vaikutusta.

Kuvassa 4 oikealla opetusdata on annettu välillä $[0, 2]$ ja datapisteitä on 1000 kappaletta. Kuvasta huomataan, että neuroverkko mallintaa ODE:n ratkaisua jo hyvin välillä $[0, 2]$, mutta neuroverkko ennustaa ratkaisua välillä $[2, 4]$ vielä melko huonosti. Myös tässä tapauksessa neuroverkon ennustetta koitettiin parantaa syöttämällä datapisteitä tiheämmin välillä $[0, 2]$ ja lisäämällä hukkafunktion minimoimiseen käytettävien kierrosten määrää, mutta tässäkin tapauksessa näillä ei ollut ratkaisuun juurikaan vaikutusta.



Kuva 4. Vasemmalla kuvassa on neuroverkon antama tulos katkoviivalla verrattuna analyttiseen ratkaisuun yhtenäisellä viivalla, kun opetusdataa on annettu 1000 pistettä välillä $[0, 1]$. Oikealla kuvassa on neuroverkon antama tulos katkoviivalla verrattuna analyttiseen ratkaisuun yhtenäisellä viivalla, kun opetusdataa on annettu 1000 pistettä välillä $[0, 2]$

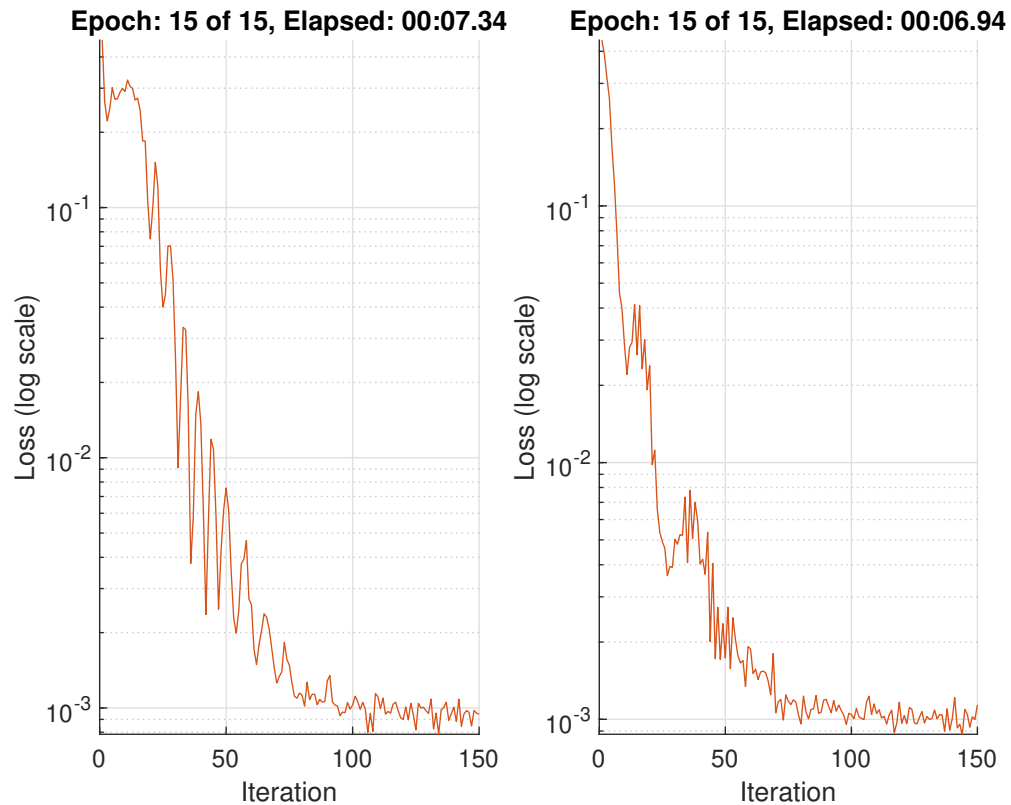
Kuvassa 5 vasemmalla opetusdata on annettu välillä $[0, 3]$ ja datapisteitä on 1000 kappaletta. Kuvassa 5 oikealla opetusdata on annettu koko piirtovälillä $[0, 4]$ ja datapisteitä on 1000 kappaletta. Kuvista huomataan, että neuroverkko mallintaa molemmissa tapauksissa analyttiistä ratkaisua hyvin koko piirtovälillä $[0, 4]$.



Kuva 5. Vasemmalla kuvassa on neuroverkon antama tulos katkoviivalla verrattuna analyttiseen ratkaisuun yhtenäisellä viivalla, kun opetusdataa on annettu 1000 pistettä välillä $[0, 3]$. Oikealla kuvassa on neuroverkon antama tulos katkoviivalla verrattuna analyttiseen ratkaisuun yhtenäisellä viivalla, kun opetusdataa on annettu 1000 pistettä välillä $[0, 4]$

Hukkafunktion minimoimiseen käytettiin oletuksena 15 kierrosta, joiden aikana toteutettiin 150 iteraatiota kussakin tapauksessa. Kuvissa 6 ja 7 nähdään hukkafunktion arvo iteraatioiden lukumäärän funktiona eli nähdään kuinka nopeasti ja tarkasti hukkafunktion minimikohta löydetään. Kuvista nähdään myös minimointiin kulunut aika. Kuvassa 6 vasemmalla nähdään hukkafunktion minimointi, kun opetusdata on annettu välillä $[0, 1]$ ja oikealla puolella, kun opetusdata on annettu välillä $[0, 2]$. Kuvassa 7 taas vasemmalla puolella nähdään hukkafunktion minimointi, kun opetusdata on annettu välillä $[0, 3]$ ja oikealla puolella, kun opetusdata on annettu välillä $[0, 4]$. Kuvista huomataan, että hukkafunktion minimikohta löydetään lähes yhtä nopeasti ja lähes samalla tarkkuudella. Myöskään minimointiin kuluneet ajat eivät eroa merkittävästi toisistaan.

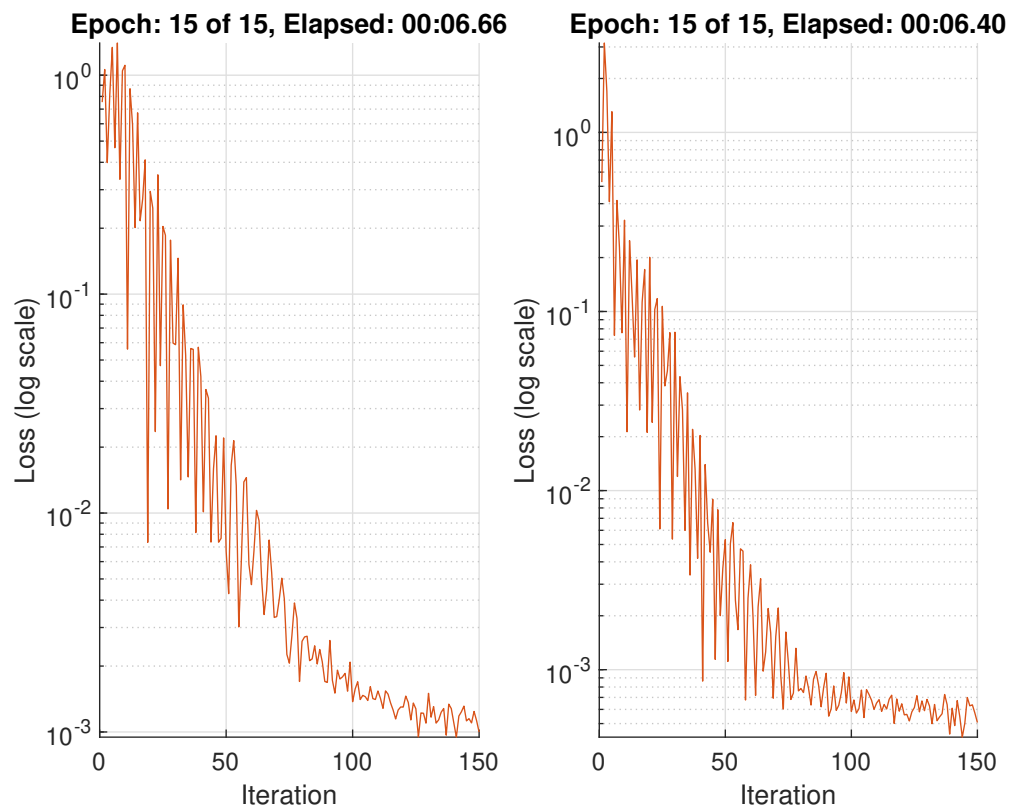
Kuvassa 8 vasemmalla nähdään ratkaisun virhe analyttiseen ratkaisuun verrattuna, kun opetusdataa on annettu välillä $[0, 1]$. Virhe on maksimissaan noin 0,12 ja se on suurimmillaan kohdassa $x = 4$. Kuvassa 8 oikealla nähdään ratkaisun virhe analyttiseen ratkaisuun verrattuna, kun opetusdataa on annettu välillä $[0, 2]$. Virhe on maksimissaan noin 0,06 ja



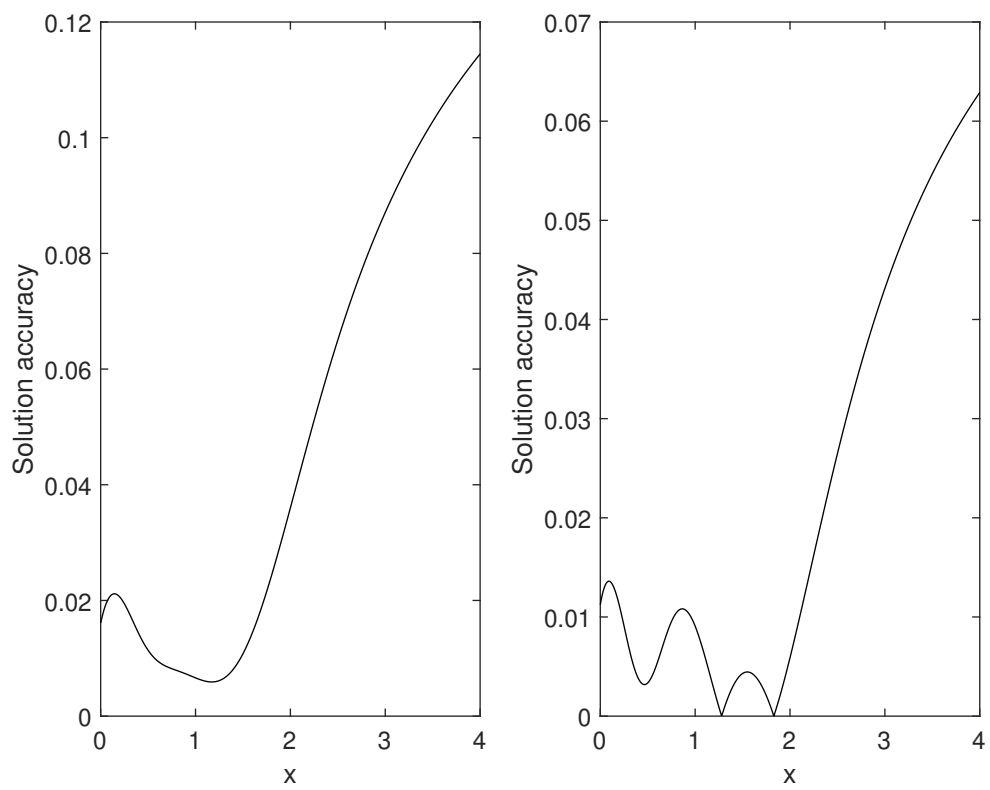
Kuva 6. Vasemmalla kuvassa nähdään hukkafunktion minimointi, kun opetusdataa on annettu välillä $[0, 1]$. Oikealla kuvassa nähdään hukkafunktion minimointi, kun opetusdataa on annettu välillä $[0, 2]$.

se on suurimmillaan kohdassa $x = 4$

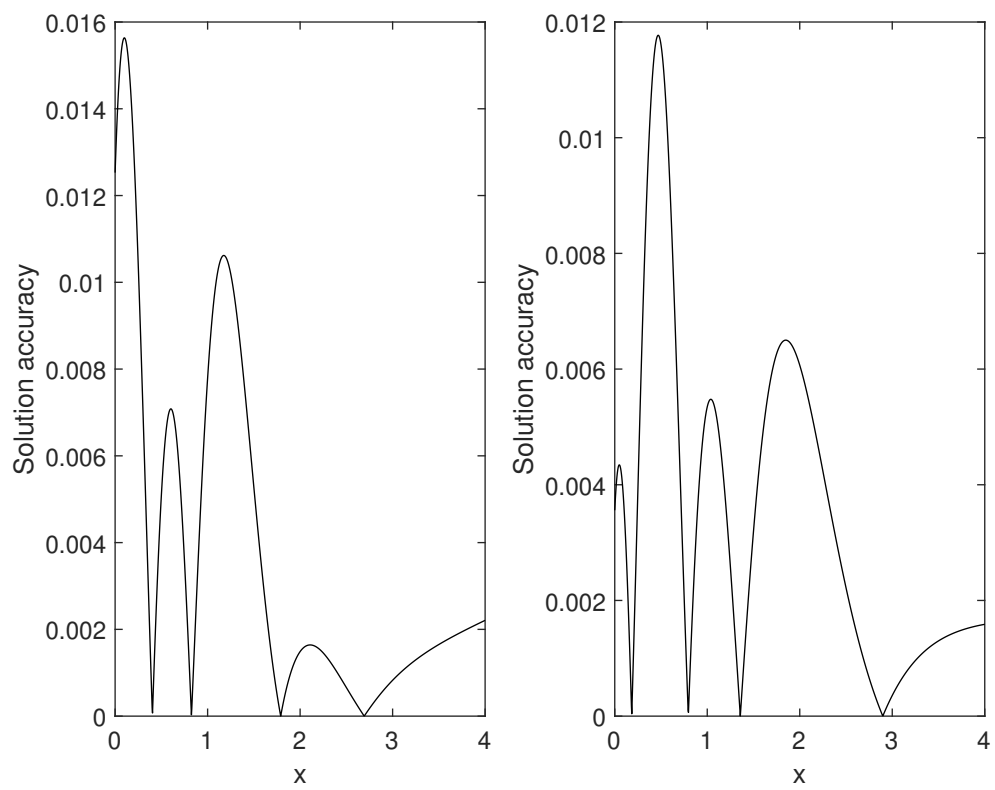
Kuvassa 9 vasemmalla nähdään ratkaisun virhe analyttiseen ratkaisuun verrattuna, kun opetusdataa on annettu välillä $[0, 3]$. Virhe on maksimissaan noin 0,016 ja se on suurimmillaan kohdassa noin $x = 0$. Kuvassa 9 oikealla nähdään ratkaisun virhe analyttiseen ratkaisuun verrattuna, kun opetusdataa on annettu välillä $[0, 4]$. Virhe on maksimissaan noin 0,012 ja se on suurimmillaan noin kohdassa $x = 0,5$.



Kuva 7. Vasemmalla kuvassa nähdään hukkafunktion minimointi, kun opetusdata on annettu välillä $[0, 3]$. Oikealla kuvassa nähdään hukkafunktion minimointi, kun opetusdata on annettu välillä $[0, 4]$.



Kuva 8. Vasemmalla kuvassa on ratkaisun virhe verrattuna analyyttiseen ratkaisuun, kun opetusdataa on annettu välillä $[0, 1]$. Oikealla kuvassa on ratkaisun virhe verrattuna analyyttiseen ratkaisuun, kun opetusdataa on annettu välillä $[0, 2]$



Kuva 9. Vasemmalla kuvassa on ratkaisun virhe verrattuna analyyttiseen ratkaisuun, kun opetusdataa on annettu välillä $[0, 3]$. Oikealla kuvassa on ratkaisun virhe verrattuna analyyttiseen ratkaisuun, kun opetusdataa on annettu välillä $[0, 4]$

5 YHTEENVETO

Työssä tutkittiin, miten neuroverkkoja voidaan hyödyntää differentiaaliyhtälöiden ratkaisemisessa. Esimerkkinä ratkaistiin ensimmäisen kertaluvun ODE Matlab-ohjelmistolla. Haivaittiin, että opetusdatapisteitä tulee antaa riittävän isolta väliltä ja datapisteitä tulee olla riittävästi, jotta neuroverkon antama tulos on luotettava. Neuroverkon antama tulos ei siis ole luotettava opetusdatan ulkopuolella ja tällä menetelmällä virhettä löytyy myös väliltä, jolla opetusdata on annettu. Neuroverkon antama ratkaisu toimii epäluotettavasti opetusdatan ulkopuolella, koska menetelmällä ekstrapoloidaan derivaattaa. Lisäksi neuroverkkoa opettaessa käytetään neuroverkon ennustamia arvoja y_θ sen sijaan, että käytettäisiin differentiaaliyhtälön oikeita arvoja y . Hukkafunktion (23) termissä $\|y'_\theta - f(x, y_\theta)\|^2$ käytetään neuroverkon ennustamia arvoja y_θ funktion f sisällä sen sijaan, että käytettäisiin differentiaaliyhtälön oikeita arvoja y , minkä takia neuroverkon antaman ratkaisun derivaattaa verrataan approksimaatioon funktion derivaatasta eikä oikeaan derivaataan.

Aikaisemmassa tutkimuksessa on saatu tuloksia tehokkaammin ja paremmalla tarkkuudella. Artikkelissa [7] on käytetty samanlaista neuroverkkoa eli MLP yhdellä piilokerroksella, jossa on kymmenen piilokerroksen neuronია. Kuitenkin artikkelissa opetusdatapisteitä on käytetty vain 10 kappaletta, kun taas tässä työssä opetusdatana käytettiin 1000 opetusdatapistettä. Artikkelissa saadut tulokset ensimmäisen kertaluvun ODE:n tapauksessa olivat kuitenkin samaa tarkkuusluokkaa. Korkeamman kertaluvun ODE:n tapauksessa tulokset olivat jopa tarkempia kuin tässä työssä saadut tulokset. Tämä voisi selittyä osittain sillä, että artikkelissa käytettiin hukkafunktion minimoimiseen tehokkaampaa quasi-Newton BFGS -menetelmää, kun taas tässä työssä hukkafunktio minimoitiin SGDM-menetelmällä. Lisäksi artikkelissa virhe oli nolla välillä, jolla opetusdatapisteet oli annettu, mutta tässä työssä virhe saattoi olla jopa maksimissaan välillä, jolla opetusdata oli annettu.

Työssä käytettyä menetelmää ensimmäisen kertaluvun ODE:n ratkaisemiseksi voidaan soveltaa myös korkeamman kertaluvun ODE:n, PDE:n ja ODE-ryhmien ratkaisemiseen. Jatkotutkimuksessa voitaisiinkin soveltaa menetelmää näiden ongelmien ratkaisemiseen. Lisäksi tehokkaampien ja tarkempien ratkaisujen saamiseksi differentiaaliyhtälöitä voitaisiin ratkaista neuroverkoille paremmin soveltuvassa ympäristössä.

LÄHTEET

- [1] Akshay Kulkarni, Adarsha Shivananda, Anoosh Kulkarni, and Dilip Gudivada. *Applied Generative AI for Beginners: Practical Knowledge on Diffusion Models, ChatGPT, and Other LLMs*. Apress, Berkeley, CA, 2023.
- [2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [3] Neha Yadav, Anupam Yadav, and Manoj Kumar. *An Introduction to Neural Network Methods for Differential Equations*. SpringerBriefs in Applied Sciences and Technology. Springer Netherlands, Dordrecht, 2015.
- [4] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Incorporated, Newark, UNITED KINGDOM, 2016.
- [5] G Shanker Rao. *Numerical Analysis*. New Age International Ltd, Daryaganj, INDIA, 2006.
- [6] Hervé Abdi, Dominique Valentin, and Betty Edelman. *Neural Networks*. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States of America, 1999.
- [7] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, September 1998. Conference Name: IEEE Transactions on Neural Networks.
- [8] Edwin K. P. Chong. *An Introduction to Optimization*. John Wiley & Sons, Incorporated, Newark, UNITED STATES, 2013.