Model!T

In opdracht van RIKZ

De bepaling van ontbrekende golfparameters op basis van Neurale Netwerken Beschrijving van de optimalisatiemethode

Modelit
Rotterdamse Rijweg 126
3042 AS Rotterdam
Telefoon +31 10 4623621

info@modelit.nl
www.modelit.nl

Model!T

In opdracht van RIKZ

De bepaling van ontbrekende golfparameters op basis van Neurale Netwerken Beschrijving van de optimalisatiemethode

Datum 2 Oktober 2004
Modelit
KvK Rotterdam 24290229

Model!T

Documentatiepagina

Opdrachtgever	RIKZ
Titel	De bepaling van ontbrekende golfparameters op basis van Neurale Netwerken
Datum	2 Oktober 2004
Projectteam opdrachtgever	T. Kremers E.R.A. Marsman B. Roskam
Projectteam Modelit	Nanne van der Zijpp Kees-Jan Hoogland
Projectomschrijving	
Trefwoorden	

Versies

25 Aug 2004	Eerste handgeschreven versie
22 Sept 2004	Uitgewerkte versie Formules uitgewerkt
2 Okt 2004	Versie 1 Titelblad, Hoofdstuk indeling, Inleiding en globale opzet toegevoegd

Inhoud

1 Notatie

Opmerking: de onderstaande tabel is nog niet compleet

P	Aantal periodes
R	Aantal reeksen
W	Vector die de waargenomen reeksen bevat *W*((*r*-1)*P*+*p*): waarneming voor reeks *r* in periode *p*.
Σ_w	Met W corresponderende varianties Σ_w((*r*-1)*P*+*p*): variantie voor reeks *r* in periode *p*.
V	Matrix met geschatte reeksen *V*((*r*-1)*P*+*p*): geschatte waarde voor reeks *r* in periode *p*.
Σ_v	Met V corresponderende varianties Σ_v((*r*-1)*P*+*p*): variantie voor reeks *r* in periode *p*.
X	Te variëren variabelen. *X* is gelijk aan *W* behalve voor de elementen van *W* die hiaat zijn. Het aantal vrijheidsgraden waarover *X* gevarieerd kan worden is dus gelijk aan het aantal hiaten in *W*.
A ()	De neurale netwerk operator die een PxR vector afbeeldt op een PxQ (1<=Q<=R)
∇A	Jacobiaan horende bij het neurale netwerk. Hoogte PxQ, breedte PxR.

2 Inleiding

2.1 Doel

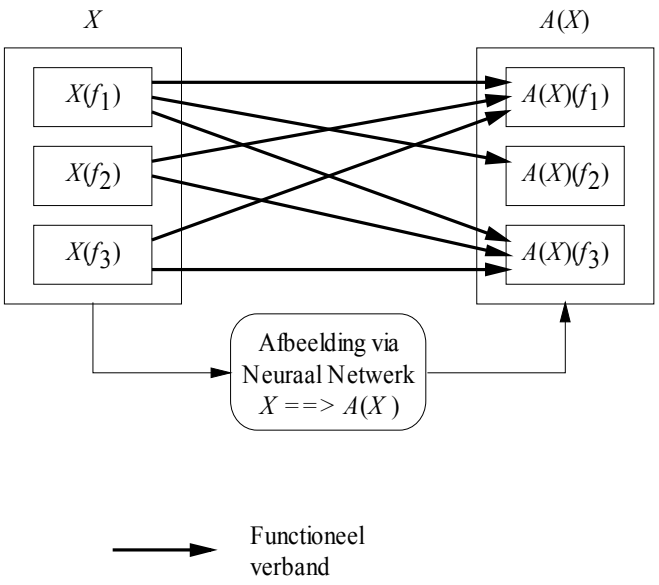
Het schatten van een aantal ontbrekende elementen uit de vector W

2.2 Gemeten waarden, enkelvoudige hiaten en wederzijdse hiaten

Spoor de hiaten in *W* op. Deel de indices als volgt in:

- f_1 indices van de gemeten waarden.
- f_2 indices van hiaten die bij toepassing van het neurale netwerk *niet* beïnvloed worden door andere hiaten. Dit noemen we *enkelvoudige* hiaten.
- f_3 indices van hiaten die bij toepassing van het neurale netwerk *wel* beïnvloed worden door andere hiaten. Dit noemen we *wederzijdse* hiaten.

Ten behoeve van de eenvoud van notatie gaan we ervan uit dat de vectoren *W en V* zijn gesorteerd als $W=[W(f_1); W(f_2); W(f_3)]$ en $V=[V(f_1); V(f_2); V(f_3)]$.



Figuur 1: *De wederzijdse hiaten (indices f_3) worden bij toepassing van het Neurale Netwerk beïnvloed door (de geschatte waarden van) ander hiaten*

2.3 Uitgangspunt voor de berekening

Er wordt een vector *X* met voorspelde waarden die corresponderen met *W* aangemaakt. Deze vector wordt als volgt gevuld:

- De elementen f_1 worden overgenomen uit W :
  $X(f_1)=W(f_1)$
- De elementen f_2 worden geschat op basis van W:
  $X(f_2)=A(W)(f_2)$
- De elementen f_3 worden zodanig gekozen dat een nog te definiëren Consistency Measure, $J(X(f_3))$, wordt geminimaliseerd. De Consistency Measure geeft aan in hoeverre de neurale netwerk voorspelling $A(x)$ overeenkomt met de beschikbare meetgegevens.

3 Optionele bewerkingen

3.1 Inleiding

Voordat met iteratieve procedures wordt gestart kunnen een aantal betrekkelijk eenvoudige optimalisaties worden uitgevoerd. Deze staan beschreven in dit hoofdstuk.

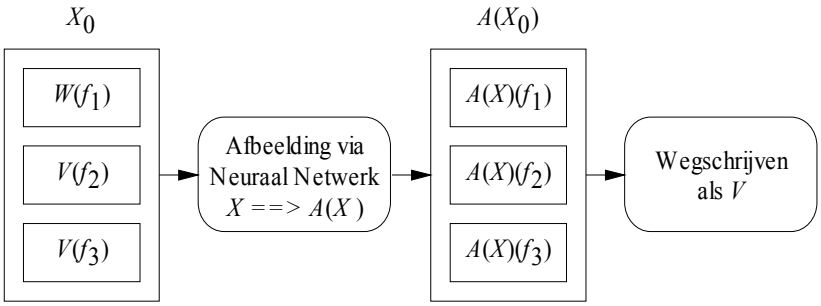
3.2 Verbeteren initiële schattingen

De schattingen die met het Neurale Netwerk worden gemaakt worden superieur beschouwd ten opzichte van de initiële schatting *V*. Vanwege deze reden wordt voorafgaande aan andere berekeningen de *V* waarden vervangen door schattingen die gemaakt zijn op basis van 1 iteratie van het neurale netwerk. Deze bewerking komt neer op:

$$V(f_2) := U(f_2)$$
$$V(f_3) := U(f_3)$$

met:

$$U = A\left(\begin{bmatrix} W(f_1) \\ V(f_2) \\ V(f_3) \end{bmatrix}\right) \tag{0}$$



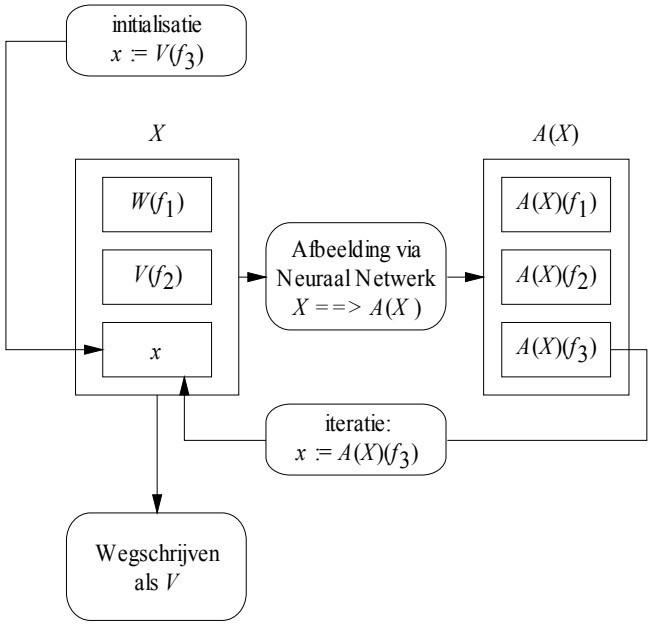
Figuur 2: *Door 1 slag van het neurale netwerk toe te passen worden de initiële schattingen verbeterd*

3.3 Fast repair

De procedure “verbeter initiële schattingen” kan desgewenst een aantal malen herhaald worden, met dien verstande dat na de eerste toepassing van de procedure alleen de elementen met indices f_3 hoeven te worden aangepast, omdat de elementen met indices f_2 volledig afhangen van variabelen $W(f_1)$ die niet meer wijzigen.

De methode die op deze manier ontstaat (zie Figuur 3) heeft als voordelen dat hij makkelijk te begrijpen is, eenvoudig te implementeren is en geen hoge eisen stelt aan rekenkracht.

Tegenover deze eenvoud staat het nadeel dat de methode bij het bepalen van een optimum geen rekening kan houden met informatie over de nauwkeurigheid van waarnemingen en voorspellingen.



Figuur 3: *De methode Fast Repair: De wederzijdse hiaten worden iteratief bijgeschat*

4 Het schatten van wederzijdse hiaten

4.1 Inleiding

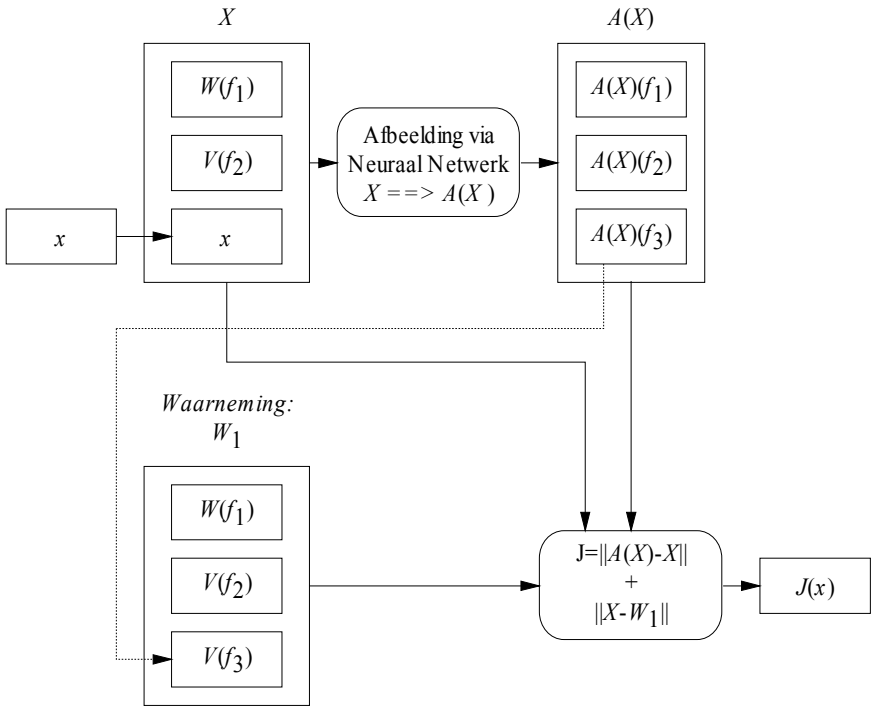
De wederzijdse hiaten (indices f_3) kunnen met de methodes die zijn beschreven in het vorige hoofdstuk niet optimaal worden geschat omdat in elke schatting de initiële waardes die zijn aangenomen voor de wederzijdse hiaten zullen. In het ideale geval dempt de invloed van de initiële waarde na een aantal iteraties uit, maar dit is zeker niet gegarandeerd.

4.2 Consistency Measure: Overzicht

Vanwege deze reden zoeken we naar een extra mogelijkheid om de wederzijdse hiaten te kunnen vullen. Een mogelijkheid hiervoor is het stellen van extra consistentie eisen. Iedere keuze die men maakt met betrekking tot de waarde van de wederzijdse hiaten heeft invloed op de voorspellingen die worden gemaakt met het neurale netwerk. We stellen ons nu op het standpunt dat we een vector X zoeken die zoveel mogelijk aan de volgende criteria voldoet:

- Er moet consistentie bestaan tussen de gebruikte invoer vector *X* en de voorspellingen die worden gebaseerd op deze invoer vector *A*(*X*);
- De afstand tussen *X* en de waargenomen waarden dient minimaal te zijn.

Er geldt dat alleen de elementen in *X* die overeenkomen met de wederzijdse hiaten mogen worden gevarieerd. De elementen die variëren duiden aan met de vector x. De mate waarin aan de bovenstaande criteria is voldaan kan worden uitgedrukt in een prestatie criterium *J*(x).



Figuur 4: *Berekening schema voor de consistentie maat*

4.3 Nadere beschouwing van de Consistency Measure

In de Figuur 1 wordt de consistentie uitgedrukt als de mate van overeenstemming tussen de ingevulde waarden *X* en de waargenomen waarden W_1 enerzijds en het verschil tussen *X* en *A*(*X*) anderzijds.

- Merk op dat de vector met waarnemingen voor een groot deel bestaat uit elementen die ook in de vector *X* vertegenwoordigd zijn.
- Merk tevens op dat $V(f_3)$ in veel gevallen het resultaat zal zijn van een schatting op basis van het neurale netwerk met een beginwaarde die slechter is dan *X*. Zie voor beschrijvingen hiervan het vorige hoofdstuk. In deze gevallen is $A(X)(f_3)$ waarschijnlijk een betere schatting.

Vanwege deze reden vervangen we in Figuur 4 $V(f_3)$ door de waarde van $A(X)(f_3)$ zoals met een stippellijn is aangegeven.

De totale afstandsmaat wordt nu:

$$J(x) = \left\|A(X) - X\right\|^A + \left\|A(X)(f_3) - x\right\|^B$$

met:

$$X = \begin{bmatrix} W(f_1) \\ V(f_2) \\ x \end{bmatrix} \qquad (0)$$

Merk op dat $\left[A(X)(f_3) - x\right]$ een deelvector is van $\left[A(X) - X\right]$. Vergelijking (0) kan daarom beschouwd worden als de som van twee afstandsmaten die beide op $\left[A(X) - X\right]$ werken, zij het dat afstandsmaat “B” alleen de elementen f_3 beschouwd. We kunnen daarom een nieuwe afstandsmaat definiëren die overeenkomt met de som van afstandsmaten “A” en “B” uit vergelijking (0):

$$J(x) = \left\|A(X) - X\right\|$$

met:

$$X = \begin{bmatrix} W(f_1) \\ V(f_2) \\ x \end{bmatrix} \qquad (0)$$

J(*X*) kan beschouwd worden als de euclidische afstand die rekening houdt met de varianties van de waarnemingen en de voorspelde waarden

4.4 Wiskundige formulering van de Consistency Measure

De consistency measure zoals gedefinieerd in (0) kan als volgt worden uitgeschreven:

$$J(x) = \left\|A(X) - X\right\| = \left(A(X) - X\right)'\Sigma^{-1}\left(A(X) - X\right) \qquad (0)$$

De variantie-covariantie matrix is hierin een diagonaal matrix (eventuele covarianties worden genegeerd). Op de diagonaal staan meetfouten die corresponderen met *X* en

de onnauwkeurigheid van de Neurale Netwerk afbeelding $A(X)$. In de onderstaande tabel staan deze meetfouten uitgeschreven. Hierin wordt de onnauwkeurigheid van de Neurale Netwerk afbeelding in het punt X genoteerd als $\Sigma_N(X)$.

Indices	Afwijking	Wegen met	Variantie/Covariantie Matrix
f_1	$A(X)(f_1) - W(f_1)$	$\Sigma_1 = \Sigma_N(X)(f_1) + \Sigma_W(f_1)$	$S_1 = diag(\Sigma_1)$
f_2	$A(X)(f_2) - V(f_2)$	$\Sigma_2 = \Sigma_N(X)(f_2) + \Sigma_V(f_2)$	$S_2 = diag(\Sigma_2)$
f_3	$A(X)(f_3) - x$	$\Sigma_3 = \Sigma_N(X)(f_3)$	$S_3 = diag(\Sigma_3)$

We kunnen de doelfunctie dus uitschrijven als:

$$\begin{aligned} J(x) = & \left(A(X)(f_1) - W(f_1)\right)'.S_1^{-1}\left(A(X)(f_1) - W(f_1)\right) \\ & + \left(A(X)(f_2) - V(f_2)\right)'.S_2^{-1}\left(A(X)(f_2) - V(f_2)\right) \\ & + \left(A(X)(f_3) - x\right)'.S_3^{-1}\left(A(X)(f_3) - x\right) \end{aligned} \qquad (0)$$

De tweede term in deze doelfunctie varieert overigens niet met x en kan dus genegeerd worden.

5 Numerieke oplosmethode

5.1 Inleiding

In dit hoofdstuk wordt de numerieke oplosmethode beschreven voor de minimalisatie van de Consistency Measure *J*(*x*). Deze functie is voor een groot deel een black box omdat de waarde ervan voor een afhangt van de toegepaste neurale netwerken waarvan we geen algebraïsche beschrijving beschikbaar hebben.

5.2 Opzet methode

De afbeelding $A(X(x))$ kan voor elke gewenste vector x geëvalueerd worden, en ook de afgeleide (gradiënt) van $A(X)$ kan daarbij bepaald worden. Het is dus mogelijk om de niet-lineaire afbeelding $A(X)$ te lineariseren, door nu deze gelineariseerde afbeelding in $\mathrm{J(x)}$ in te vullen ontstaat er een kwadratische benadering van de Consistency Measure $\mathrm{J(x)}$ waarvan op eenvoudige wijze het minimum kan worden gevonden.

Met behulp van dit minimum wordt het volgende iteratiepunt x^* bepaald zodanig dat $\mathrm{J(x^*)} \leq \mathrm{J(x)}$ .Omdat de afbeelding $A(X)$ niet lineair is, zal over het algemeen de afgeleide (gradiënt) in het nieuwe punt gewijzigd zijn, waardoor er een nieuwe iteratie nodig is waarin opnieuw een kwadratische benadering van $\mathrm{J(x)}$ bepaald moet worden.

Op deze wijze tekent het volgende algoritme zich af:

Algoritme	
stap 0:	Kies een startoplossing voor x
stap 1:	Lineariseer $A(X)$ rond het punt x
stap 2:	Bepaal de kwadratische benadering van $\mathrm{J(x)}$ deze is van de vorm $\mathrm{J(x)} = \mathrm{x^T Gx} + \mathrm{Hx} + \mathrm{C}$
stap 3:	Minimaliseer deze benadering van $\mathrm{J(x)}$ dit levert een kandidaat x^* voor de volgende iteratie op.
stap 4:	Gebruik dit kandidaat punt om een x^* te vinden waarvoor geldt dat $\mathrm{J(x^*)} \leq \mathrm{J(x)}$
stap 5:	Check of een volgende iteratie nodig is. Zet $x = x^*$ en ga naar stap 1.
stap 6:	Einde

Om dit algoritme te kunnen gebruiken moet het volgende worden aangeleverd:

- een geschikte startwaarde voor x
- een functie die (het variabele deel van) $\mathrm{J(x)}$ evalueert
- een functie die de gradiënt van $\mathrm{J(x)}$ evalueert

Het aanleveren van de gradiënt $\mathrm{J(x)}$ is niet strikt noodzakelijk maar levert een drastische versnelling van het algoritme op doordat op voorhand kan worden nagegaan wat een verandering in x voor invloed heeft op $\mathrm{J(x)}$. In de methode die in de “oude” wavix versie wordt toegepast worden alleen de functie waarden van $\mathrm{J(x)}$ geëvalueerd, en niet de gradiënt.

5.3 Bepaling van een geschikte startwaarde

De startwaarden voor x kunnen de gemeten waarden zijn aangevuld met de initiële schattingen en eventueel schattingen gemaakt d.m.v. neurale netwerken.

5.4 Berekening van het variabele deel van *J*(*x*)

Ten behoeve van de optimalisatie zijn we alleen geïnteresseerd in de elementen van *X* en *A*(*X*) die afhangen van de waarden van *x*. De indices f_2 hangen per definitie niet van *x* af, maar mogelijk is ook een deelverzameling van f_1 invariant voor x. De elementen f_3 hangen per definitie juist wel van x af (anders zou geen sprake zijn van wederzijdse hiaten).

Definieer f_4 als een verzameling indices die overlapt met de indices f_1 en f_3 en die aangeeft welke waarden van *A*(*x*) worden beïnvloed door de elementen f_1 van *X*.

Om rekentijd te besparen kunnen we de evaluatie van (0) beperken tot:

$$J(x) = \left\| A(X)(f_4) - X(f_4) \right\| \qquad (0)$$

Dit levert een drastische besparing op omdat f_4 typisch slechts 1% of minder van de elementen representeert.

5.5 Afbeelding: wederzijdse hiaten naar oplossingsvector

De vector is een lineaire transformatie van de te variëren parameters *x,* oftewel *X=Ax+b*. In deze sectie wordt deze transformatie uitgewerkt.

We kunnen *X* schrijven als:

$$X = E_1.W(f_1) + E_2.V(f_2) + E_3.x \qquad (0)$$

met:

E_1, E_2, E_3 De eenheidsmatrices waaruit de kolommen f_1, f_2 en f_3 zijn verwijderd

Het onderscheid tussen f_1 en f_2 is in het kader van deze afleiding niet meer zinnig en compliceert enkel de notatie. In feite kunnen we de indices f_2 zelfs volledig negeren omdat zowel $X(f_2)$ als $A(X)(f_2)$ tijdens het iteratieproces niet meer wijzigen. We onderscheiden daarom in het vervolg van deze afleiding alleen f_1 , f_3 en f_4.

Voorbeeld

Stel W is een vector met 5 elementen waarvan het derde en vierde element een hiaat zijn. Er geldt dan:

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad E_3 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Vergelijking (0) ziet er dan als volgt uit:

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Merk op dat de vector w desgewenst geschreven kan worden met:

$$w = E_1'.W \tag{0}$$

Merk tevens op dat E_1, E_2 en E_3 gerelateerd zijn via de uitdrukking:

$$E_1.E_1' + E_2.E_2' + E_3.E_3' = I \;,\; E_1'.E_1 = I \;,\; E_2'.E_2 = I \;,\; E_3'.E_3 = I \tag{0}$$

met:

I De eenheidsmatrix (van toepasselijke afmetingen)

Uitdrukkingen (0) en (0) komen mogelijk in een later stadium van pas om uitdrukkingen te vereenvoudigen.

5.6 Berekening van de gradient van $J(x)$

$$J(x) = \left\| A(X)(f_4) - X(f_4) \right\| \tag{0}$$

$$J(x) = \left(A(X) - X \right)'.E_4 S_4^{-1} E_4' \left(A(X) - X \right) \tag{0}$$

met :

$$S_4 = diag(E_4'\Sigma)$$

Benader:

$$A(X) = A(X_0) + \nabla A(X_0).(X - X_0) \quad (0)$$

En vul in:

$$X = E_1W + E_3x \quad (0)$$

Benader:

$$\begin{aligned} &A(X) - X \approx A(X_0) + \nabla A(X_0).(E_1W + E_3x - E_1W - E_3x_0) - E_1W - E_3x \\ &= A(X_0) + \nabla A(X_0).E_3(x - x_0) - E_1W - E_3x \\ &= A(X_0) - \nabla A(X_0).E_3x_0 - E_1W + (\nabla A(X_0)E_3 - E_3).x \end{aligned} \quad (0)$$

Merk op dat na-vermenigvuldigen met $\nabla A(X_0)$ neerkomt op het negeren van alle kolommen van $\nabla A(X_0)$ behalve de kolommen met indices f_3. We hoeven dus alleen de kolommen f_3 van $\nabla A(X_0)$ te berekenen.

Benader vervolgens:

$$\begin{aligned} &E_4'(A(X) - X) \approx \\ &= (E_4'A(X_0) - E_4'\nabla A(X_0).E_3x_0 - E_4'E_1W) + (E_4'\nabla A(X_0)E_3 - E_4'E_3).x \end{aligned} \quad (0)$$

We stellen nu vast dat de minimum voorwaarde voor het evalueren van vergelijking (0) en dus ook (0) is:

- De evaluatie van de rijen f_4 van A(X0). Met andere woorden: de elementen f_4 moeten berekend worden met het neurale netwerk
- De evaluatie van de rijen f_4 en de kolommen f_3 van de gradiënt $\nabla A(X_0)$.

We herschrijven (0) nog een maal:

$$\begin{aligned} J(x) &= (a + bx)'S_4^{-1}(a + bx) \\ &= a'S_4^{-1}a + 2a'.S_4^{-1}bx + x'b'S_4^{-1}bx \end{aligned} \quad (0)$$

met:

$$\begin{aligned} a &= (E_4'A(X_0) - E_4'\nabla A(X_0).E_3x_0 - E_4'E_1W) \\ b &= (E_4'\nabla A(X_0)E_3 - E_4'E_3) \end{aligned}$$

De gradiënt kan nu worden uitgedrukt als:

$$\nabla J(x) = 2b'S_4^{-1}a + 2b'S_4^{-1}bx \quad (0)$$

Wanneer de afbeelding $A(x)$ lineair zou zijn dan is de oplossing voor x waarvoor de gradiënt 0 is gegeven door de oplossing van het stelsel:

$$(b' S_4^{-1} b).x = - b' S_4^{-1} a \qquad (0)$$

Uitdrukking (0) kan verder vereenvoudigd worden door *a* en *b* te normaliseren met de elementen uit sqrt(S_4).

Definieer:

$$\begin{aligned} S_4^{-1/2} &= diag(1/\sqrt{E_4'\Sigma}) \\ \bar{a} &= a.S_4^{-1/2} \\ \bar{b} &= S_4^{-1/2} \end{aligned} \qquad (0)$$

Dit komt erop neer dat in *a* en *b* de rijen worden gedeeld door de wortel uit de corresponderende elementen uit S_4.

Vergelijking (0) wordt nu geschreven als:

$$(\bar{b}'\bar{b}).x = - \bar{b}'\bar{a} \qquad (0)$$

5.7 Optimalisatie methode

Wanneer de afbeelding $A(x)$ lineair zou zijn dan is de oplossing voor x direct te bepalen. De afbeelding $A(x)$ is echter niet lineair en lineaire benaderingen van deze afbeeldingen zijn alleen lokaal geldig.

Om tot een overall minimum van de doelfunctie te komen is daarom een serie van iteraties nodig waarbij tijdens iedere iteratie $A(x)$ opnieuw wordt gelinealiseerd rond het huidige iteratie punt.

Er is een optimalisatie methode gekozen die op basis van deze lineaire benadering (die resulteert in een kwadratische benadering van de doelfunctie) tijdens iedere iteratie de doelfunctie minimaliseert binnen een zogenaamde trust-region. De trust-region kan beschouwd worden als een bol waarbinnen gezocht wordt met een vrij te kiezen straal. Binnen de trust-region wordt de lineaire benadering van $A(x)$ beschouwd als acceptabel.

De straal van de trust region wordt door de optimalisatiemethode adaptief aangepast op basis van het verschil tussen de benaderde resultaten en de waarden gevonden op bases van functie evaluaties.

Het algoritme dat deze serie van bewerkingen uitvoert is ontleend aan de literatuur[1] en opnieuw gecodeerd ion Matlab.

Om deze functie te runnen is het volgende nodig:
- een startwaarde voor x
- een functie die de matrices $\bar{b}'\bar{b}$ en $\bar{b}'\bar{a}$ zoals genoemd in vergelijking (0) retourneert.

[1] -Section 5.2 in P.E. Frandsen, K. Jonasson, H.B. Nielsen, O. Tingleff: "Unconstrained Optimization", IMM, DTU. 1999.
- "damping parameter in marquardt's method" Hans Bruun Nielsen, IMM, DTU. 99.08.10 / 08.12

5.8 Rekentechnische aspecten

De uit te voeren berekeningen lijken rechttoe-rechtaan. Een mogelijke complicatie bij het uitvoeren van de berekeningen zou echter de afmetingen van de matrices kunnen zijn. De onderstaande tabel inventariseert de afmetingen van de gebruikte vectoren en matrices in vergelijking (0).

S_4	Diagonaal matrix, Hoogte = Breedte = lengte(f_4)
$E_4' A(X_0)$	Vector, Hoogte = lengte(f_4)
X_0	Vector, Hoogte = lengte (W)
$E_4' \nabla A(X_0).E_3$	Matrix, Hoogte = lengte(f_4), Breedte = lengte(f_3)
x_0	Vector, Hoogte = lengte(f_3)
E_1	Matrix, Hoogte = lengte (W) , Breedte = lengte(f_1)
E_3	Matrix, Hoogte = lengte (W) , Breedte = lengte(f_3)
E_4	Matrix, Hoogte = lengte (W) , Breedte = lengte(f_4)
$a = (E_4' A(X_0) - E_4' \nabla A(X_0).E_3 x_0 - E_4' E_1 W)$	Vector Hoogte = lengte(f_4)
$b = (E_4' \nabla A(X_0) E_3 - E_4' E_3)$	Matrix Hoogte = lengte(f_4) Breedte = lengte(f_3)

Matrices als S_4, E_1, E_3, en E_4 lijken groot, maar komen in de programmatuur niet terug, omdat vermenigvuldigen met deze matrices neerkomt op het uitvoeren van triviale bewerking zoals het “knippen” van een aantal rijen uit een matrix of het vermenigvuldigen van iedere kolom met een ander getal.

De matrix $E_4' \nabla A(X_0) E_3$

Wanneer matrices als S_4, E_1, E_3, en E_4 buiten beschouwing worden gelaten blijft de uitdrukking $E_4' \nabla A(X_0) E_3$ grootste matrix over. Het aantal elementen van deze matrix is gelijk aan: het aantal wederzijdse hiaten (=lengte(f_3)) maal het aantal datapunten dat hierdoor beïnvloed wordt (= lengte(f_4)).

Als het aantal hiaten groot is loopt het aantal elementen van deze matrix snel op: Wanneer bijvoorbeeld het gemiddelde aantal inputs per neuraal netwerk 10 bedraagt en het aantal hiaten is gelijk aan 1000, dan bedraagt het aantal cellen van de matrix

circa 10*1000^2 . Echter het aantal gevulde cellen bedraagt dan slechts circa 10*1000.

Om te voorkomen dat de methode beperkt wordt door het feit dat bij een groot aantal hiaten geheugenproblemen optreden dient gekozen te worden voor een zogenaamde sparse-matrix representation. Dit houdt in dat alleen de niet-nul cellen worden opgeslagen.

De matrix $\bar{b}'\bar{b}$

Een andere forse matrix is de matrix $\bar{b}'\bar{b}$. Deze heeft in het bovenstaande voorbeeld circa 1000^2 cellen. Ook hier geldt echter dat het aantal gevulde cellen erg klein is.

5.8.1 Mogelijkheid voor verdere optimalisatie

Een nog verdere efficiëntie verbetering wordt verkregen wanneer bedacht wordt dat de operator A feitelijk een samenstelling is van een aantal Neurale Netwerk operatoren. De operator A rekent een vector uit die bestaat uit de achter elkaar gezette kolommen van de matrix met geschatte waarden. Iedere kolom uit deze matrix wordt door een apart neuraal netwerk voortgebracht.

Er geldt derhalve:

$$\nabla A(X_0) = \sum_{NN \in S} \nabla_{NN} A(X_0) \tag{0}$$

met

$$\left(\nabla_{NN1} A(X_0)\right)^t \left(\nabla_{NN2} A(X_0)\right) = 0 \text{ voor } \forall NN_1, NN_2 \left| NN_1 \neq NN_2 \right.$$

Net	in $f_3\rightarrow$ in $f_4\downarrow$	n	j	n	n	n	j	j	n	n	n	j	n	n	n	j	n
1	n	$\frac{\partial A_1}{\partial X_1}$	$\frac{\partial A_1}{\partial X_2}$														
	j	$\frac{\partial A_2}{\partial X_1}$	$\frac{\partial A_2}{\partial X_2}$														
	j																
	j																
	j																
	j																
2	n																
3	n																
	n																
	j																
	j																
4	n																
	n																
	j																
	j																

Legenda

Cellen uit de Jacobiaan die corresponderen met f3 (de wederzijdse hiaten)

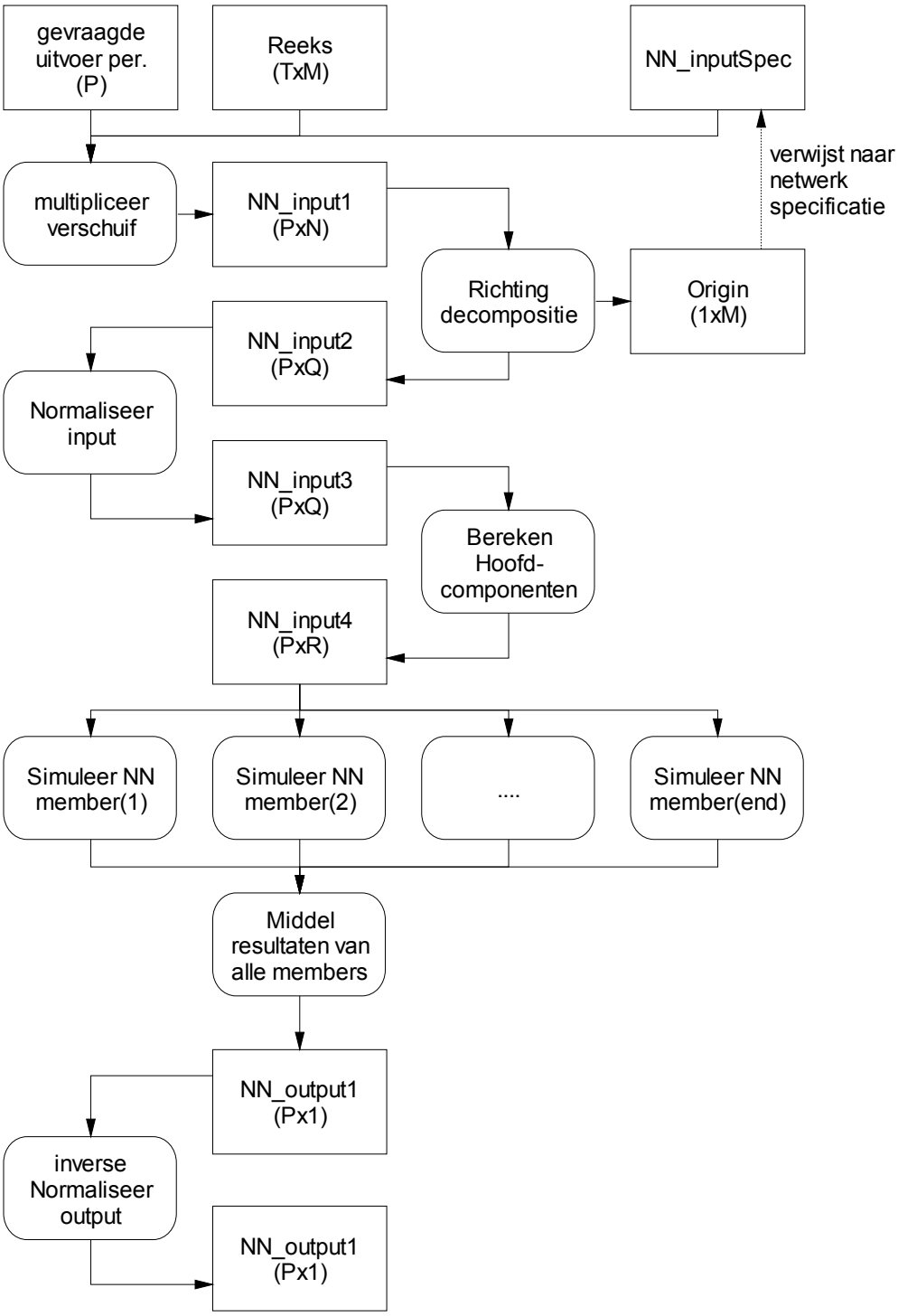
Cellen uit de Jacobiaan die corresponderen met f4 (de beinvloede waarden)

Cellen uit de Jacobiaan die corresponderen met f3 en f4: $E_4{}'\nabla A(X_0)E_3$

Figuur 5: *De hele tabel geeft bevat de Jacobiaan van de operator A. De groen gearceerde cellen representeren de te bepalen cellen* $E_4{}'\nabla A(X_0)E_3$**.**

6 Het toepassen van Neurale Netwerken

6.1 Overzicht



Figuur 6: *Bewerkingsketen bij toepassing van een Neuraal Netwerk binnen WAVIX*

6.2 Het linealiseren van de Neurale Netwerk operator

6.2.1 Theorie

De neurale netwerk operator bestaat uit een groot aantal stappen die stuk voor stuk gelinealiseerd kunnen worden als:

$$L^{stap}(X^{stap-1}) = H^{stap}.X^{stap-1} + g^{stap} \tag{0}$$

Twee opeenvolgende stappen kunnen worden gelinealiseerd als:

$$\begin{aligned} L^{stap}(X^{stap-1}) &= H^{stap}.\left(H^{stap-1}X^{stap-2} + g^{stap-1}\right) + g^{stap} \\ &= \left(H^{stap}H^{stap-1}\right)X^{stap-2} + \left(H^{stap}g^{stap-1} + g^{stap}\right) \end{aligned} \tag{0}$$

Wanneer we een keten van operaties zoals in Figuur 6 linealiseren zijn we alleen geïnteresseerd in de multiplicatoren en niet in de constantes.

6.2.2 Discussie

Een praktisch probleem wordt gevormd door de afmetingen van de matrices en het feit dat de invoer vector

In principe kunnen de linealisatie uitvoeren door alle multiplicatoren te berekenen. Daarbij geldt dat de X steeds de vector is waarin alle elementen uit matrices als NN_input achter elkaar staan. Een typische afmeting van een transformatie is dan bijvoorbeeld:

aantal invoer elementen x aantal uitvoer elementen x (aantal tijdstippen)^2

In principe zouden we alle transformatie matrices in sparse matrices kunnen opslaan en doorvermenigvuldigen. Er is intern in de procedure echter met het oog op de rekensnelheid gekozen voor een alternatieve opslagmethode, waarbij transformatie matrices worden opgeslagen van kleinere afmetingen, die vervolgens op een specifieke wijze dienen te worden geinterpreteerd.

Voordat deze transformatie worden gedefinieerd constateren we het volgende:

- Er moet onderscheid gemaakt worden tussen tijd invariante en tijdsafhankelijke transformaties.
- De volgende transformaties zijn tijd-invariant
    - Multipliceren
    - Verschuiven in de tijd
    - Normaliseren
    - Hoofdcomponenten berekenen
    - inverse Normaliseren
    - Middelen via de “bagging” operator
- De volgende transformaties zijn tijd-afhankelijk:
    - De operator “richting decompositie”
    - De operator “neuraal netwerk toepassen”

- De operatoren Normaliseren en “inverse normaliseren” heffen elkaar op en kunnen daarom bij het linealiseren worden genegeerd

6.2.3 Uitwerking

Noteer de verzameling van invoerwaarden als Input = { Input (inputnr,periode } (NxP elementen). De tijdverschuiving en multiplicatie van reeksen is reeds in deze input matrix verwerk en wordt voor wat betreft de transformatie buiten beschouwing gelaten.

Een bijbehorende overall transformatie matrix noteren we als D (eveneens NxP elementen).

Deze transformatie berekent de output vector in Matlab notatie als:
Output = sum(Input.*D,1)

De transformatie is opgebouwd uit twee gedeeltes:

D1	Deze transformatie representeert • De operator “richting decompositie” Dit is een tijdsafhankelijke N naar Q transformatie (Q>=N). Om de benodigde informatie op te slaan is in theorie een matrix van Q.N x P matrix nodig. Omdat bij de decompositie van richtingen elk resulterend kanaal altijd van hooguit 1 invoerkanaal afhangt, kan volstaan worden met een matrix van QxP. In een losse matrix F dient te worden aangegeven welke uitvoer reeks van welke invoer reeks afhangt: F(input,output)=1: uitvoerkanaal output hangt af van invoerkanaal input
D2	Deze transformatie representeert: • Hoofdcomponenten berekenen • Middelen via de “bagging” operator • De volgende transformaties zijn tijdvariant: • De operator “neuraal netwerk toepassen” Dit is een tijdsafhankelijke Q naar 1 transformatie.

6.2.4 Kettingregel

De combinatie van de twee transformaties laat zich met het volgende Matlab commando berekenen:

$$D=F*D1.*D2 \qquad (0)$$

$$\begin{bmatrix} \frac{\partial z_1}{\partial x_1} \\ \vdots \\ \frac{\partial z_p}{\partial x_1} \\ \\ \frac{\partial z_1}{\partial x_m} \\ \vdots \\ \frac{\partial z_p}{\partial x_m} \end{bmatrix} = Diff\left(\begin{bmatrix} \frac{\partial y_1}{\partial x_1} \\ \vdots \\ \frac{\partial y_n}{\partial x_1} \\ \\ \frac{\partial y_1}{\partial x_m} \\ \vdots \\ \frac{\partial y_n}{\partial x_m} \end{bmatrix}, \begin{bmatrix} \frac{\partial z_1}{\partial y_1} \\ \vdots \\ \frac{\partial z_p}{\partial y_1} \\ \\ \frac{\partial z_1}{\partial y_m} \\ \vdots \\ \frac{\partial z_p}{\partial y_n} \end{bmatrix} \right) \tag{0}$$

$$\frac{\partial z_i}{\partial x_j} = \sum_{q=1}^{n} \frac{\partial z_i}{\partial y_q} \frac{\partial y_q}{\partial x_j} \tag{0}$$