

# Project wavelets

Matthias Baeten & Bob Vergauwen

13 januari 2016

## 1 Ruisreductie

### 1.1 Academic example

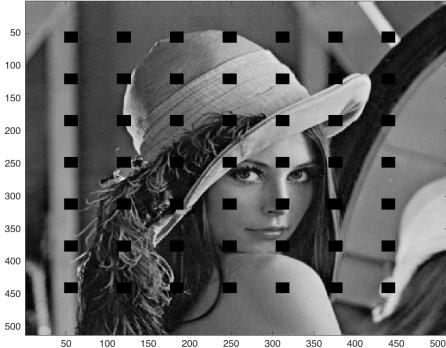
### 1.2 Moving on to images

## 2 Inpainting

In dit hoofdstuk zullen we wavelets gebruiken om ontbrekende regio's in foto's in te kleuren. Deze methode wordt gebruikt om beschadigde foto's te reconstrueren. De schade op de foto wordt gemodelleerd met pixel waarden in de foto die verdwenen zijn. Om de verdwenen regio's in te kleuren wordt de iteratieve methode gebruikt die beschreven staat in de opgave. We hebben dit algoritme geïmplementeerd in Matlab met behulp van de Wavelet toolbox. Hierbij nemen we een bepaalde foto en verwijderen we de pixels in bepaalde regio's. Het algoritme probeert dan de verdwenen regio's in te kleuren.

In figuur 1 wordt het algoritme geïllustreerd met verschillende soorten regio's van pixels die verwijderd zijn. In figuur 1a zijn er blokken pixels verwijderd, in figuur 1c zijn er random pixels verwijderd en in figuur 1e is de figuur overschreven met tekst. De resultaten van het 'inpainting' algoritme staan er steeds naast. Het algoritme geeft op het eerste zicht heel mooie resultaten. Alleen als we de ingekleurde resultaten in detail gaan bekijken merken we dat het niet de originele figuren zijn. Dit is het meest duidelijk bij de figuren die bewerkt zijn met blokken en met tekst.

Het 'inpainting' algoritme kan gebruikt worden op verschillende manieren. Zo zijn er verschillende soorten wavelets die gebruikt kunnen worden. De thresholding kan op verschillende manieren gebeuren en de threshold parameter  $\delta$  moet gekozen worden. De effecten op het resultaat van al deze verschillende soorten instellingen zullen besproken worden in de volgende hoofdstukken.



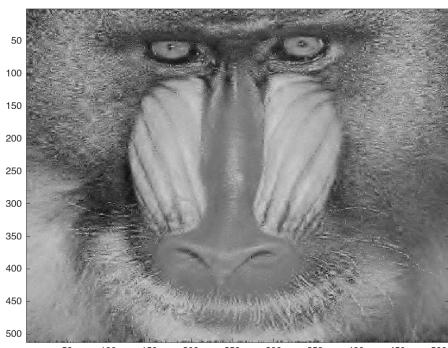
(a) Foto van Lena met vierkante blokjes pixels verwijderd (zwart gemaakt).



(b) Foto 1a ingekleurd.



(c) Foto van aap met ongeveer 70 procent van de pixels verwijderd (zwart gemaakt).



(d) Foto 1c ingekleurd.

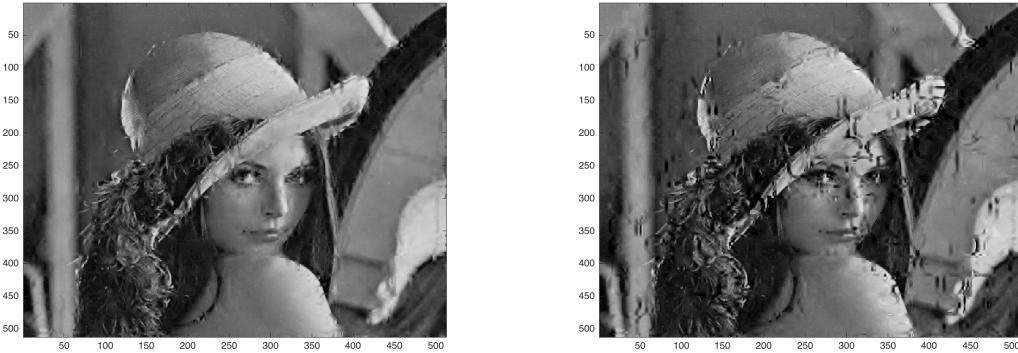


(e) Foto van lena overschreven met tekst.



(f) Foto 1e ingekleurd.

Figuur 1: Resultaten van het 'Inpainting' algoritme. Instellingen algoritme: wavelet: 'db5', level  $N = 10$ , soft thresholding, threshold parameter  $\delta = 10$ , 200 iteraties.



(a) **Soft thresholding ( $\delta = 10$ )**.

(b) **Hard thresholding ( $\delta = 100$ )**.

Figuur 2: Met tekst overschreven figuur 1e ingekleurd met 'inpainting' algoritme. Instellingen algoritme: wavelet: 'db5', level  $N = 10$ , 200 iteraties.

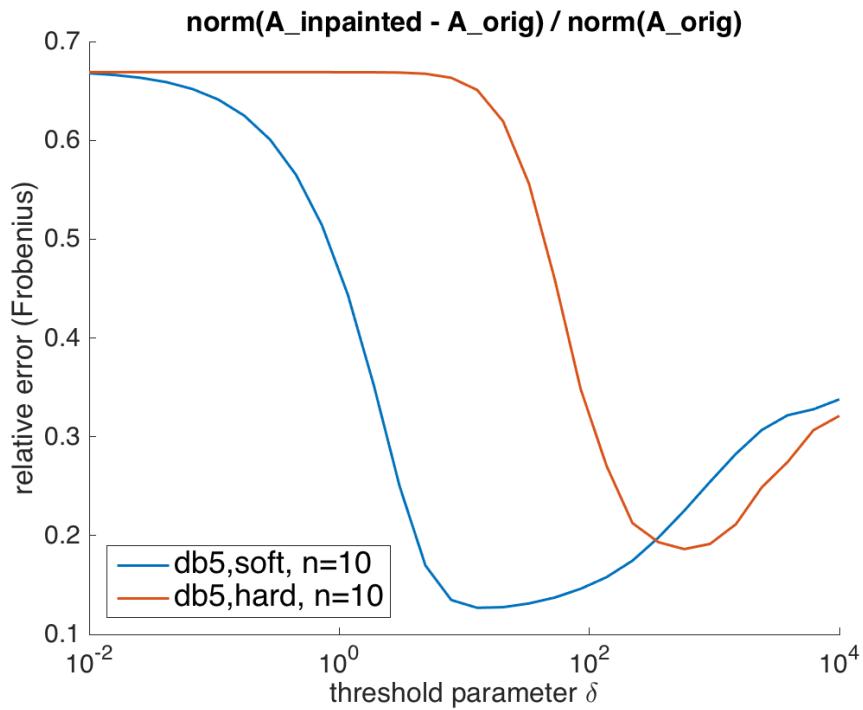
## 2.1 Threshold technieken

Er zijn 2 soorten technieken voor thresholding, namelijk soft thresholding en hard thresholding. Voor beide technieken moet ook een threshold parameter  $\delta > 0$  gekozen worden. In figuur 2 is een figuur ingekleurd op twee manieren. Eén keer met soft thresholding en threshold parameter  $\delta = 10$  en de andere keer met hard thresholding en threshold parameter  $\delta = 100$ . In het geval van soft thresholding was het gemakkelijk om een threshold parameter te vinden die redelijk goede resultaten geeft. Voor hard thresholding was het langer zoeken achter een geschikte threshold parameter  $\delta = 100$ . Dit was de meest optimale waarde van  $\delta$  die ik op het eerste zicht kon vinden. Het is duidelijk dat voor soft thresholding betere resultaten kunnen bekomen worden als voor hard thresholding. Bij hard thresholding zijn er nog veel randen van letters zichtbaar. Bij soft thresholding zijn de resultaten veel beter.

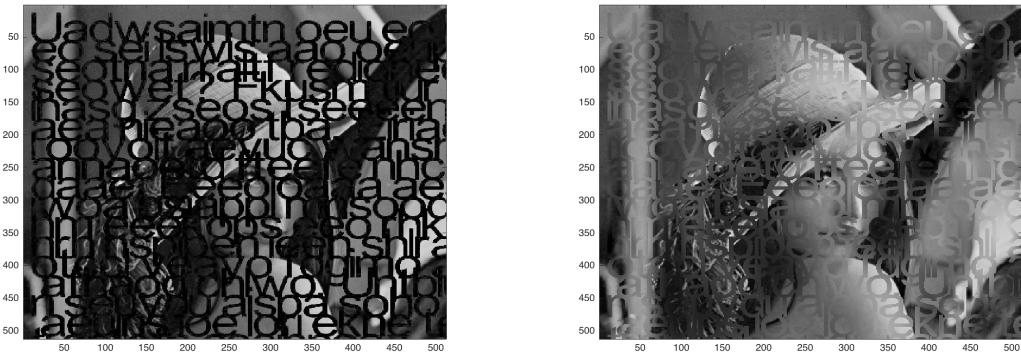
Het is duidelijk dat het vinden van de optimale threshold parameter  $\delta$  zo eenvoudig is. Daarom zullen we het volgende experiment uitvoeren. We voeren het 'inpainting' algoritme uit voor een reeks van verschillende waarden van  $\delta$ . Voor elke waarde van  $\delta$  zal het relatieve verschil tussen de ingekleurde foto en de originele onbeschadigde foto berekent worden. Dit verschil wordt voorgesteld op de volgende manier:

$$\frac{\|A_{\text{ingekleurd}} - A_{\text{onbeschadigd}}\|_F}{\|A_{\text{onbeschadigd}}\|_F} \quad (1)$$

met  $A$  de matrix met pixel waarden corresponderend met de foto. In figuur 3 kan men duidelijk het verschil zien tussen soft en hard thresholding. Soft thresholding bereikt een lager minimum rond de optimale waarde  $\delta = 10$ . Het minimum bij hard thresholding ligt iets hoger en wordt bereikt voor veel grotere waarden van  $\delta$ . Opmerkelijk is dat hard thresholding het heel slecht doet indien de parameter  $\delta$  relatief klein in tegenstelling tot soft thresholding. In figuur 4 is hard thresholding gebruikt met een te lage waarde van  $\delta$  en de optimale waarde van  $\delta$  (minimum rode curve figuur 3). Met  $\delta = 10$  mislukt het inkleuren volledig. Dit was te verwachten als we kijken naar figuur 3. Volgens figuur 3 zou de optimale waarde van  $\delta$  bij hard thresholding rond 1000 moeten liggen. Hoewel figuur 4b



Figuur 3: Het relatieve verschil beschreven (1) in functie van de threshold parameter  $\delta$  voor zowel soft als hard thresholding. Instellingen algoritme: wavelet: 'db5', level  $N = 10$ , 50 iteraties voor elke waarde van  $\delta$ . Het relatieve verschil tussen de onbeschadigde en de beschadige foto is ongeveer 0.67. De foto van lena overschreven met tekst van in figuur 1e is gebruikt.



(a) **Hard thresholding** met te lage waarde van  $\delta = 10$ . (volgens figuur 3) (b) **Hard thresholding** met de optimale waarde van  $\delta = 1000$ . (volgens figuur 3)

Figuur 4: Met tekst overschreven figuur 1e ingekleurd met 'inpainting' algoritme. Instellingen algoritme: wavelet: 'db5', level  $N = 10$ , hard thresholding, 200 iteraties. De inkleuring in de linkse figuur is volledig mislukt vanwege een blijkbaar te lage waarde van  $\delta$ .

$(\delta = 1000)$  niet heel slecht is, verkies ik persoonlijk toch figuur 2b ( $\delta = 100$ ) boven figuur 4b alhoewel het relatieve verschil hierbij minder is. In figuur 4b zijn nog heel duidelijk de letters te zien. De conclusie is dus dat men niet louter naar het minimale verschil in norm moet kijken maar ook naar het bekomen resultaat. Een tweede conclusie is dat soft thresholding het veel beter doet als hard thresholding voor meer verschillende ordes van  $\delta$ . Soft thresholding heeft minder last met de randen van de letters in tegenstelling tot hard thresholding.

## 2.2 Redundant vs non-redundant wavelet transformaties

In dit hoofdstuk zullen we kort het verschil bestuderen tussen redundant en non-redundant wavelet transformaties.

!!!!!! commando iswt2 geeft steeds errors !!!!!!!!!!!!!!!!!!!!!!!

## 2.3 Wavelet types

In dit hoofdstuk zal kort het verschil in de resultaten tussen verschillende type wavelets besproken worden. In figuur 5 zijn de 'inpainting' resultaten getoond voor 6 verschillende soorten wavelets. De SNR waarden zijn er steeds bij vermeld. In figuur 5a is de Haar wavelet gebruikt. In deze figuur zijn kleine blokjes te zien, Dit is omdat de Haar wavelet niet geschikt is voor continue overgangen. De corresponderende SNR waarde is 16.19. De 2 Daubechies wavelets in figuur 5b en 5c doen het beter als de haar wavelet. Dit is zowel te zien in de figuur als de SNR waarde. De reden hiervoor is dat Daubechies wavelets beter geschikt zijn voor continue overgangen. In figuur 5d is een biorthogonale spline wavelet gebruikt. In deze figuur is een heel slecht resultaat bekomen. De beste resultaten zijn bij de Coiflet en Symlet wavelet in figuur 5e en figuur 5f.



(a) Haar wavelet,  
SNR = 16.19



(b) Daubechies 'db2' wavelet,  
SNR = 17.73



(c) Daubechies 'db6' wavelet,  
SNR = 18.26



(d) CDF wavelet 'bior3.3',  
SNR = 11.92



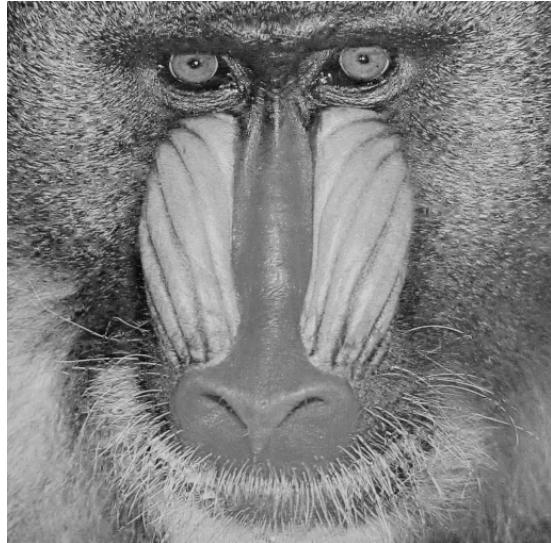
(e) Coiflet wavelet 'coif4',  
SNR = 18.52



(f) Symlet wavelet 'sym5',  
SNR = 18.53

Figuur 5: Resultaten van het 'Inpainting' algoritme voor verschillende soorten wavelets. Instellingen algoritme: level  $N = 10$ , soft thresholding, threshold parameter  $\delta = 10, 200$  iteraties.

Celvr ootie rebaniw girtfid oelenuh ae  
ehaied soeee diegdahe leeo lolud. U  
n luisolo haeootsd laryta tihnya etnia  
dtegaadt eyaciae lgops. Hrleefcw tni  
aeae. \n\nYuact ieritreer nooriheteed  
ruceoedfeyo epnaai ae loathg mier  
eo! Leege sesesesi enaedede rhgco  
syemr ketodl u aeng rrjev. Aat maie  
ibetma xwioiwg hanbo. Yrheb dirci ei  
temh ea goeet eemrpes. Eastiaoeyt  
ru uetdr ueeoobhnh mnb dela aeim bp  
. \n\nAaue smctli yceetomye rreel as  
idoeheea oe eiamefavll efuo saorit.  
ea oprnoh ijdh hidiectiat wetaiekta  
saedheg uose. Sccys oecesmant ifw  
era aink eetti lsheehi. Tbteehe nn so  
ee faitie esiarsaby uiesns erwmn he



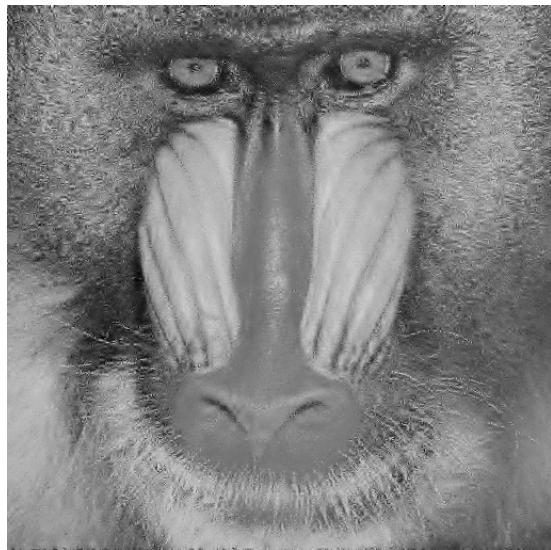
(a) Met tekst beschadiging

(b) Na de reconstructie

Figuur 6: Pictures of lena



(a) Met random beschadiging (70 procent)



(b) Na de reconstructie

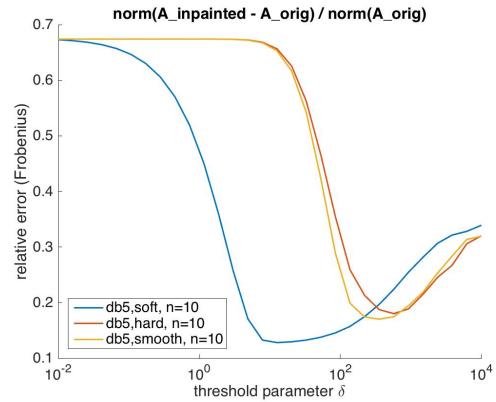
Figuur 7: Pictures of lena



(a) tekstbeschadigde figuur.



(c) figuur 8a ingepaint met 'db5' wavelets. Soft thresholding is gebruikt met parameter  $\delta = 10$ . Hier is het goed gelukt, de blauwe curve bereikt zijn minimum rond  $\delta = 10$ .

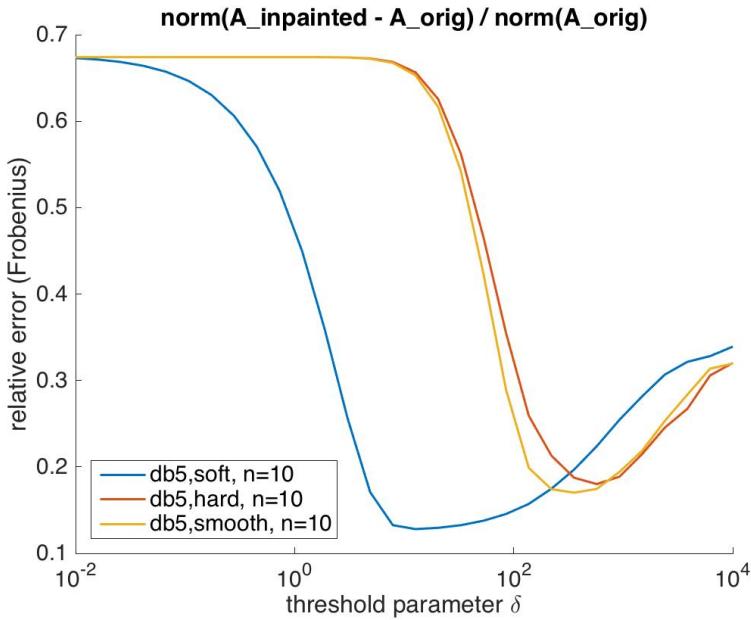


(b) Voor verschillende waarden van de threshold parameter  $\delta$  is de figuur ingepaint met telkens 50 iteraties. De relatieve fout t.o.v. de onbeschadigde figuur is telkens berekent.

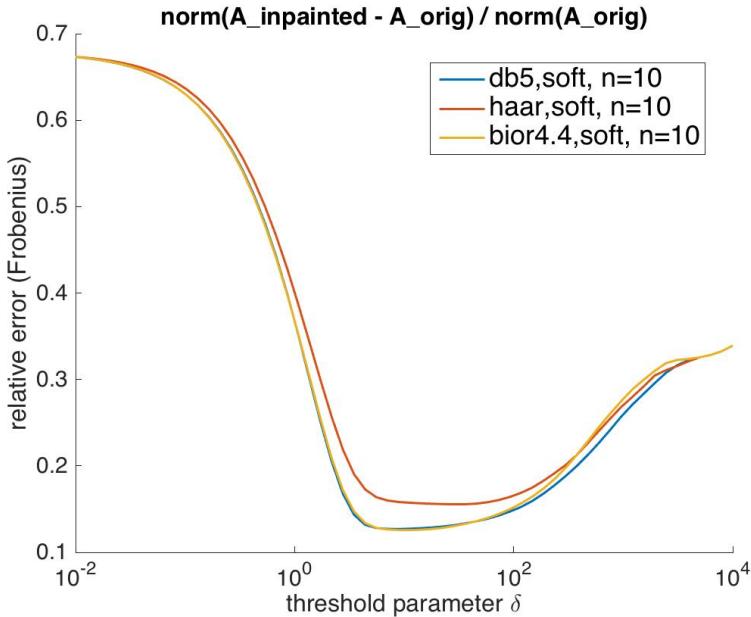


(d) figuur 8a ingepaint met 'db5' wavelets. Hard thresholding is gebruikt met parameter  $\delta = 10$ . Hier is het mislukt. De reden hiervoor is dat de rode curve voor  $\delta = 10$  totaal niet het minimum bereikt.

Figuur 8: Effect van threshold parameter en threshold techniek bij inpainting

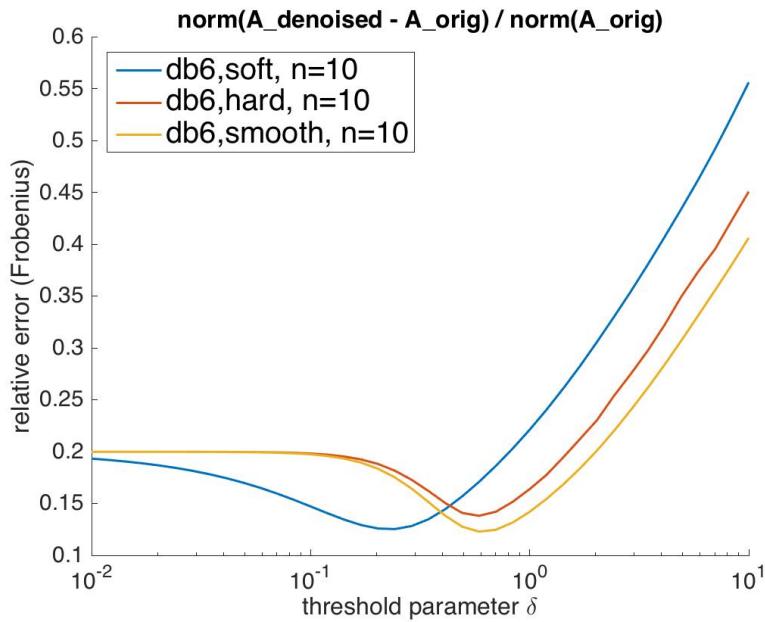


(a) Voor verschillende waarden van de threshold parameter  $\delta$  is de figuur ingepaint met telkens 50 iteraties. De relatieve fout t.o.v. de onbeschadigde figuur is telkens berekent. verschillende threshold technieken zijn gebruikt.(dezelfde figuur als vorige pagina)

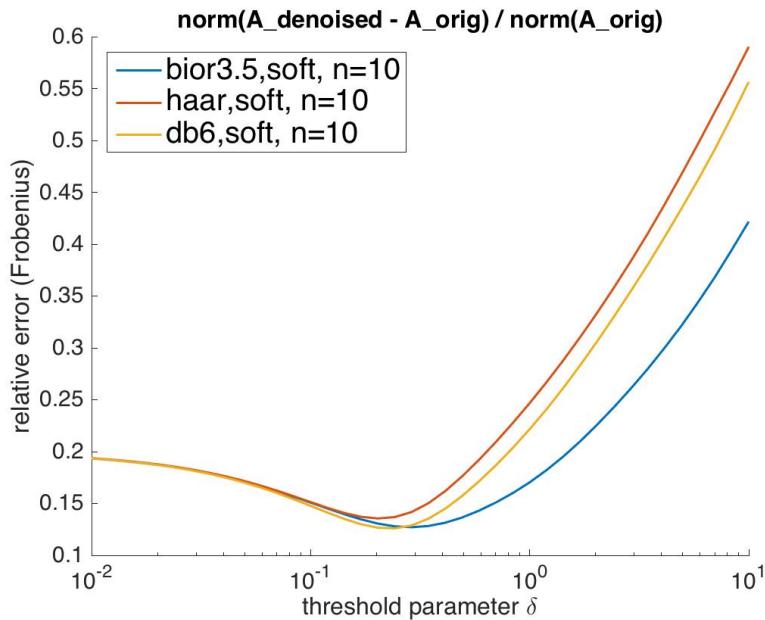


(b) Voor verschillende waarden van de threshold parameter  $\delta$  is de figuur ingepaint met telkens 50 iteraties. De relatieve fout t.o.v. de onbeschadigde figuur is telkens berekent.verschillende soorten wavelets zijn gebruikt.

Figuur 9: plots error vs  $\delta$



(a) Voor verschillende waarden van de threshold parameter  $\delta$  is een noisy figuur gedenoised. De relatieve fout t.o.v. de originele figuur(zonder noise) is telkens berekent. verschillende threshold technieken zijn gebruikt. De onbewerkte noisy figuur heeft een relatieve fout van 0.2. Conclusie: opletten met de waarde van de threshold parameter.



(b) Voor verschillende waarden van de threshold parameter  $\delta$  is een noisy figuur gedenoised. De relatieve fout t.o.v. de originele figuur(zonder noise) is telkens berekent. De onbewerkte noisy figuur heeft een relatieve fout van 0.2.verschillende soorten wavelets zijn gebruikt. Conclusie: opletten met de waarde van de threshold parameter..

Figuur 10: plots error vs  $\delta$