# Oefenzitting NMB3: oplossingen

**Michiel Vandecappelle, Nico Vervliet**, KU Leuven 11 mei 2018

## 1 Opgave 1

Bepaal de meest in het oog vallende frequentie en de overeenkomstige periode in de zonnevlekkencyclus met behulp van de snelle Fourier-transformatie. Meetgegevens zijn terug te vinden in de Matlab-bestanden `dayssn.dat` en `yearssn.dat` (zie Toledo). Om de snelle Fourier-transformatie uit te rekenen kan je het Matlab-commando `fft` gebruiken.

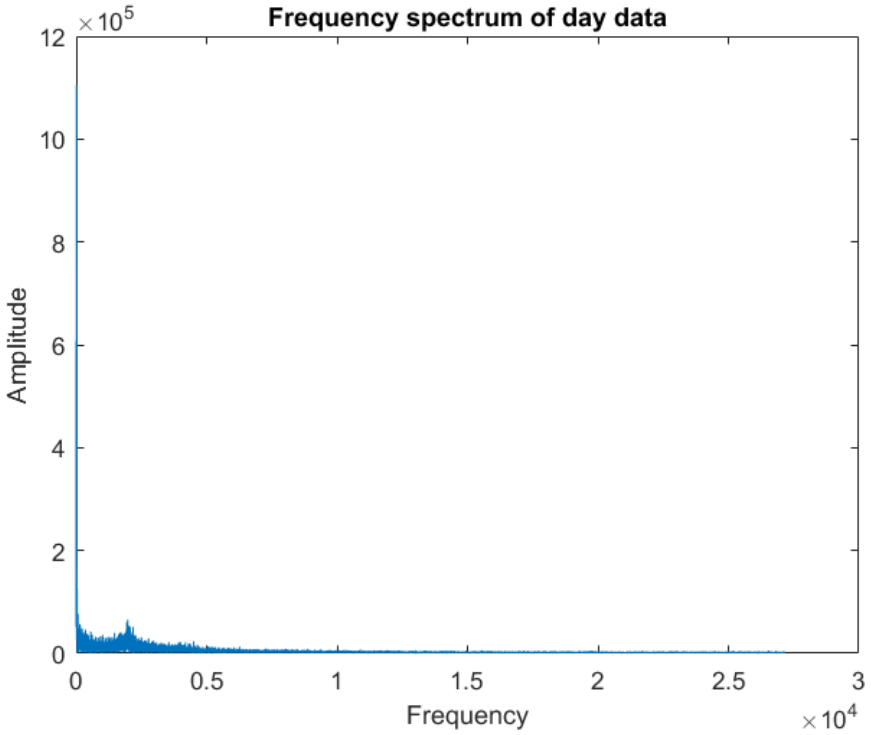
Strategie: De zonnecyclus kan gevonden worden als $1/f$ met $f$ de meest dominante frequentie (buiten DC). Voor de dagdata moeten we ook nog eens delen door het aantal dagen om de frequentie in jaren te vinden.

```
dayssn = load('dayssn.dat');
yearssn = load('yearssn.dat');

%% Day data
Xday = fft(dayssn(:,2));
Xday = abs(Xday); % Compute amplitude
nyqday = floor(length(Xday)/2); % Nyquist frequency

% Plot frequency spectrum

figure;
plot(Xday(2:nyqday)); % Only frequencies up to Nyquist frequency (no DC)
title('Frequency spectrum of day data');
xlabel('Frequency');
ylabel('Amplitude');

% Find period
[~, indexday] = max(Xday(2:nyqday)); % Find index of dominant frequency
tday = length(Xday) / (indexday*365); % Convert to period (in years)
```

```
fprintf('Period computed from day data is %d years.\n', tday);
```

```
Period computed from day data is 1.063366e+01 years.
```

Enkel waarden uitzetten die horen bij frequenties lager dan de Nyquist frequentie. Het spectrum van een reele rij is immers symmetrisch rond deze frequentie.

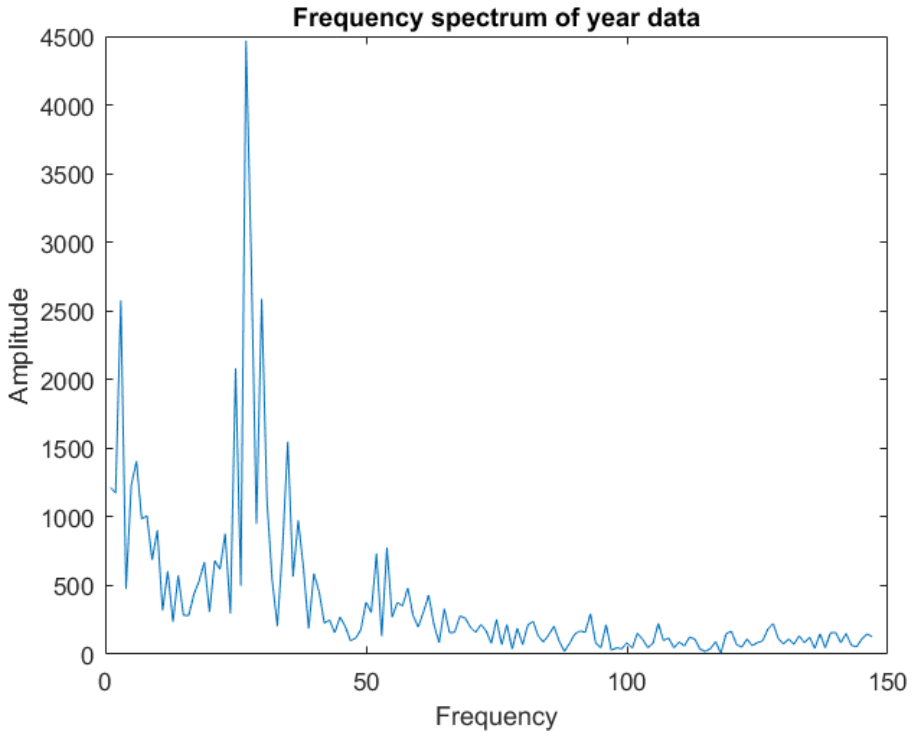



**Figuur 1:** FFT over de dagen

```
Xyear = fft(yearssn(:,2));
Xyear = abs(Xyear); % Compute amplitude
nyqyear = floor(length(Xyear)/2); % Nyquist frequency

% Plot frequency spectrum
figure;
plot(Xyear(2:nyqyear)); % Only frequencies up to Nyquist frequency (no DC)
title('Frequency spectrum of year data');
xlabel('Frequency');
ylabel('Amplitude');

% Find period
[~, indexyear] = max(Xyear(2:nyqyear)); % Find index of dominant frequency
```

```
tyear = length(Xyear)/indexyear; % Convert frequency to period

fprintf('Period computed from year data is %d years.\n', tyear);
```

```
Period computed from year data is 11 years.
```




**Figuur 2:** FFT over de jaren

## 2 Opgave 2

Comprimeer een foto. Volgende Matlab-commando's kunnen van pas komen: `imread`, `imshow`, `fft2` en `ifft2`.

De afbeelding kan gecomprimeerd worden door hoge frequenties te verwijderen, of m.a.w., door deze op nul te zetten. Net als bij de 1D FFT zitten de hoge frequenties in het midden, rond de Nyquist frequentie. In het geval van een 2D FFT van een $m \times n$ matrix hebben we dus twee banden met hoge frequenties: de rijen rond $m/2$ en de kolommen rond $n/2$.

```
f = imread('lena_gray_512.tif');
% f = imread('fourier.tif'); % Set k2 = 200 for this image
N1 = size(f,1);
```

```
N2 = size(f,2);
subplot(2,3,1)
imshow(f)

%% Compute 2D fft of image
F = fft2(f);

% Plot
subplot(1,2,2)
imshow(ifft2(F));
```

Enkele interessante manieren om te comprimeren:

- Verwijder DC: `F(1,1) = 0`.

```
%% Remove DC
Fdc = F;
Fdc(1,1) = 0;
% Plot
subplot(2,3,2)
imshow(ifft2(Fdc)); % Plot inverse fft
```

- Verwijder lage frequenties:

```
%% Remove low frequencies
k = 3; % Number of frequencies that are filtered out
Fl = F;
Fl([1:k+1, end-k+1:end],:) = 0;
Fl(:, [1:k+1, end-k+1:end]) = 0;

% Plot
subplot(2,3,3)
imshow(uint8(ifft2(Fl))); % Mainly sharp edges remain visible
```

- Verwijder de hoge frequenties:

```
k1 = 240; % Number of frequencies that are filtered out (vertical)
k2 = 200; % Number of frequencies that are filtered out (horizontal)
Fh = F;
Fh(floor(N1/2)+1+(-k1:k1),:) = 0;
Fh(:,floor(N2/2)+1+(-k2:k2)) = 0;

% Plot
```

```
subplot(2,3,4)
imshow(uint8(real(ifft2(Fh)))); % Sharp edges are smoothed
```

- Verwijder hoge frequenties in een richting:

```
%% Remove high frequencies in one direction
k1 = 240; % Number of frequencies that are filtered out (vertical)
Fx = F;
Fx(floor(N1/2)+1+(-k1:k1),:) = 0; % Compression in vertical dimension

% Plot
subplot(2,3,5)
imshow(uint8(real(ifft2(Fx))));

%% Remove high frequencies in other direction
k2 = 240; % Number of frequencies that are filtered out (horizontal)
Fy = F;
Fy(:,floor(N2/2)+1+(-k2:k2)) = 0; % Compression in horizontal dimension

% Plot
subplot(2,3,6)
imshow(uint8(real(ifft2(Fy))));
```

Hou bij het verwijderen rekening met de (geconjugeerde) symmetrie. Als het resultaat van `ifft2` een niet-nul imaginair deel heeft, dan zijn er verkeerde frequenties op nul gezet. Hint: bekijk ook eens `imagesc(log10(abs(F)))`.

```
%% Look at frequencies that are kept after compression
figure;
subplot(2,3,1)
imshow(f);
subplot(2,3,2)
imagesc(log10(abs(Fdc)))
subplot(2,3,3)
imagesc(log10(abs(Fl)))
subplot(2,3,4)
imagesc(log10(abs(Fh)))
subplot(2,3,5)
imagesc(log10(abs(Fx)))
subplot(2,3,6)
imagesc(log10(abs(Fy)))
```

# 3 Opgave 3

Implementeer een DFT-vermenigvuldigingsalgoritme. Gebruik daarvoor de volgende voorstelling van positieve gehele getallen. Zij $a$ een vector van lengte $n$ met $a(i) \in \{0, 1, 2, \ldots, 9\}, i = 1 \ldots n$ en $a(n) = 0$ als $n > 1$. De vector $a$ stelt het geheel getal $\sum_{i=1}^{n} a(i) \times 10^{i-1}$ voor. Bereken met behulp van je routine 100!.

```
function vecn = num2vec(n)
% NUM2VEC Convert a number or string representation of a number to a
% character array of digits, flipped left to right.
% INPUTS
% n     Number or string representation of a number to convert
% OUTPUTS
% vecn  Character array of digits, flipped left to right


% Convert number to string if necessary
if ~ischar(n)
n = num2str(n);
end

% Convert string to character array of digits
vecn = str2num(n')'; % Rotate string, split, and rotate back
vecn = fliplr(vecn); % Flip string left to right

end

function n = vec2str(vecn)
% VEC2STR Convert a character array of digits, flipped left to right, to a
% string representation of the number.
% INPUTS
% vecn     Character array of digits, flipped left to right
% OUTPUTS
% n        String representation of the number

% Flip and convert character array to string
n = num2str(fliplr(vecn)')';

end
```

```
function cv = hpmult(av, bv)
%HPMULT computes the product of two numbers in high precision by using the
%fft
% INPUTS
```

```
% av     Vector with the first number to be multiplied (flipped left-right)
% bv     Vector with the second number to be multiplied (flipped left-right)
% OUTPUTS
% cv     Vector with the product of the two numbers (flipped left-right)

% Pad both numbers (as digit vectors) with zeros
l = max(length(av), length(bv))*2;
av = [av zeros(1, l - length(av))];
bv = [bv zeros(1, l - length(bv))];

% Use fft to multiply vectors
C = fft(av).*fft(bv); % Compute element-wise product of fft of both vectors
cv = round(real(ifft(C))); % Compute inverse fft of product

% Carry over digits that exceed 10
for i = 1:l-1
m = mod(cv(i),10); % Compute remainder of ith digit after division by 10
cv(i+1) = cv(i+1) + floor(cv(i)/10); % Carry excess over to next digit
cv(i) = m; % Replace ith digit by its value mod 10
end

% Trim zeros from result
cv = cv(1:find(cv ~= 0, 1, 'last'));

end
```

```
%% Compute x!

n = 1;
fac = 100; % Faculty to compute

% Multiply iteratively up to requested number
for k = 2:fac
vecn = num2vec(k); % Convert number to character array
n = hpmult(n, vecn); % Multiply using fft
end

% Convert back to string and print
n = vec2str(n);
fprintf('%i! is equal to %s.\n', fac, n);
```

```
100! is equal to 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000.
```

```
400000000000000000000000.
```

De oplossing in gewone precisie is:

```
% Compare to normal precision: gamma(n) = (n-1)!
fprintf('%i! in normal precision is equal to %d.\n', fac, gamma(101));
```

```
100! in normal precision is equal to 9.332622e+157.
```