Oefenzitting NMB3: oplossingen

Nico Vervliet, *KU Leuven* *20 mei 2016*

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.

```
load dayssn; % Het .dat bestand kan met hetzelfde commando geladen worden.

x = abs(fft(dayssn(:,2)));
N = length(x);
plot(x(2:N/2));
[~, i] = max(x(2:N/2));
Deltat = 1/365;
tx = N / i * Deltat;
```

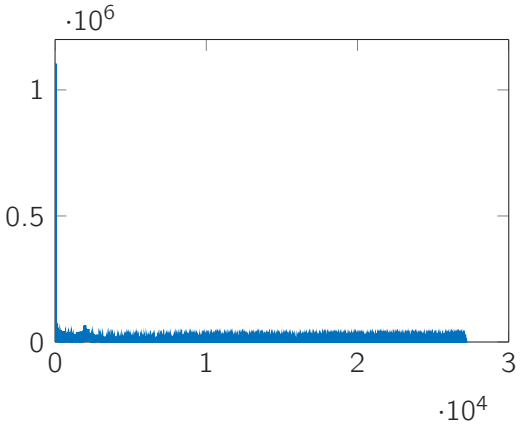
```
tx =
   10.6337
```

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

```
savetikz('img/dayfft');
```

```
load yearssn;

x = abs(fft(yearssn(:,2)));
N = length(x);
plot(x(2:floor(N/2)));
```

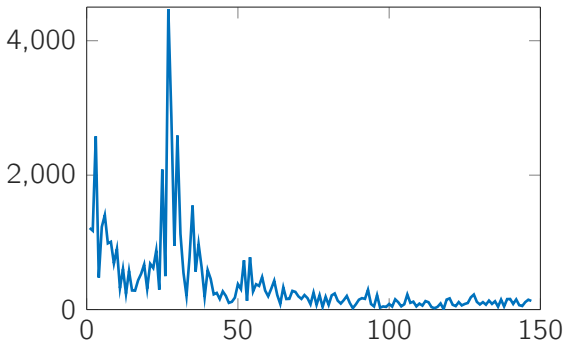


Figuur 1: FFT over de dagen

```
[~, i] = max(x(2:floor(N/2)));
tx = N / i
```

```
tx =
    11
```

```
savetikz('img/yearfft');
```



Figuur 2: FFT over de dagen

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 $n/2$.

Opmerking: In de oefenzitting is er enkel gesproken over het vierkant in het midden van de matrix. In dit vierkant heb je hoge frequenties voor x en y tegelijk. In het midden van een rand heb je echter hoge ook frequenties, maar dan slechts x of y. Een extra voorbeeld:

```
f = 1 + cos(2*pi*20*x) + cos(2*pi*50*y) + cos(2*pi*(10*x + 30*y));
surf(abs(fft2(f)));
```

```
f = imread('lena_gray_512.tif');
N = size(f,1);
subplot(1,2,1)
imshow(f)

% Verwijder DC
F = fft2(f);
Fdc = F(1,1);
F(1,1) = 0;

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

Enkele interessante manieren om te comprimeren:

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

  ```
  F([1:k+1 end-k+1:end],:) = 0;
  F(:, [1:k+1 end-k+1:end]) = 0;
  ```

- Verwijder de hoge frequenties:

  ```
  k = 240;
  F(N/2+1+(-k:k),:) = 0;
  F(:,N/2+1+(-k:k)) = 0;
  imshow(uint8(ifft2(F)))
  ```

- Verwijder hoge frequenties in een richting:

  ```
  k = 200;
  F(N/2+1+(-k:k),:) = 0;
  ```

```
%F(:,N/2+1+(-k:k)) = 0;
imshow(uint8(ifft2(F)))
```

Bekijk beide richtingen apart en tegelijk.

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)))`.

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 n = num2vec(n, l)
    if ~ischar(n)
        n = num2str(n);
    end
    n = str2num(n')';
    n = fliplr(n);
end
```

```
function n = vec2str(n)
    n = num2str(fliplr(n)')';
end
```

```
function cv = hpmult(av, bv)
    l = max(length(av), length(bv))*2;
    av = [av zeros(1, l - length(av))];
    bv = [bv zeros(1, l - length(bv))];

    C = fft(av).*fft(bv);
    cv = round(real(ifft(C)));

    for i=1:l-1;
        m = rem(cv(i),10);
        cv(i+1) = cv(i+1) + (cv(i)-m)/10;
        cv(i) = m;
    end
```

```
    cv = cv(1:find(cv ~= 0, 1, 'last'));
end
```

```
% 100!
n = 1;
for k = 2:100
    n = hpmult(n, num2vec(k));
end
n = vec2str(n)
```

```
933262154439441526816992388562667004907159682643816214685929
638952175999932299156089414639761565182862536979208272237582
511852109168640000000000000000000000000
```

De oplossing in gewone precisie is:

```
a = gamma(101)
b = str2num(n)
```

```
a =
  9.3326e+157
b =
  9.3326e+157
```