Vier voorbeelden van Fourier

Gjerrit Meinsma

Overzicht

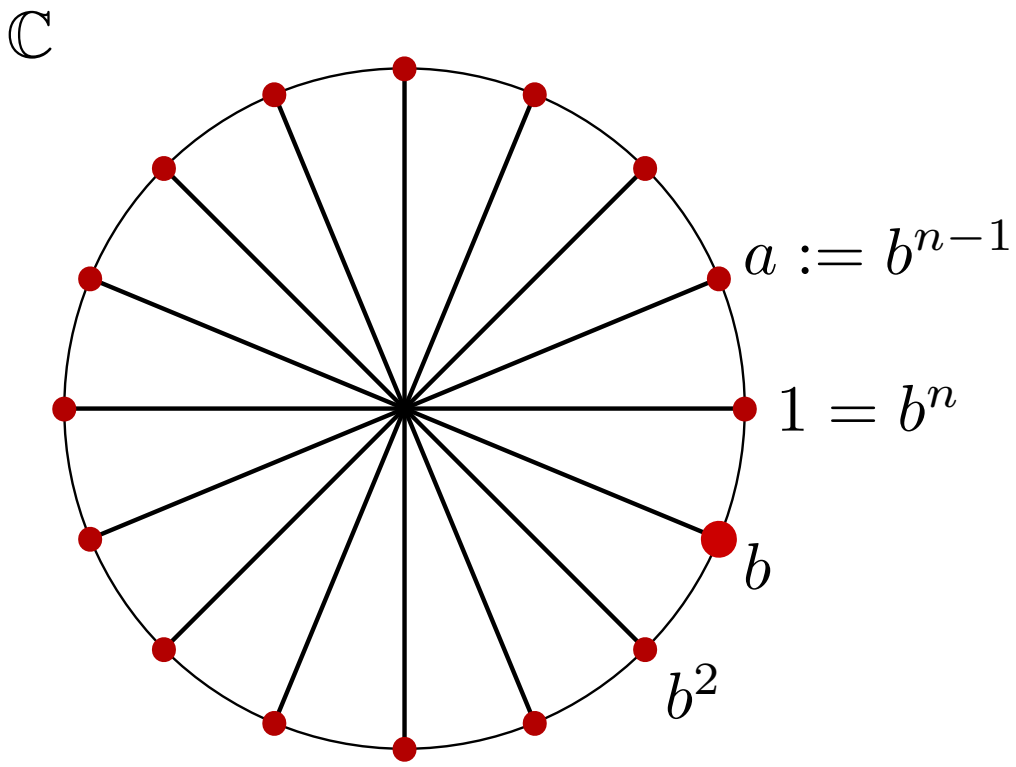
1. Wereldkampioen matrix-vectorproduct
2. Fast Fourier Transform (FFT)
3. Voorbeelden:
   - routers
   - jpeg
   - producten
   - antialiasing

Wereldkampioen matrix-vectorproduct

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-1} \end{bmatrix} := \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & b^1 & b^2 & b^3 & \cdots & b^{n-1} \\ 1 & b^2 & b^4 & b^6 & \cdots & b^{2n-2} \\ 1 & b^3 & b^6 & b^9 & \cdots & b^{3n-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & b^{n-1} & b^{2n-2} & b^{3n-3} & \cdots & \cdots \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

Het transformeert n datapunten $y_0, y_1, y_2, \ldots, y_{n-1}$
naar n Fouriercoëfficiënten $z_0, z_1, z_2, \ldots, z_{n-1}$

b is "het" complexe getal op de eenheidscirkel met $b^n = 1$:



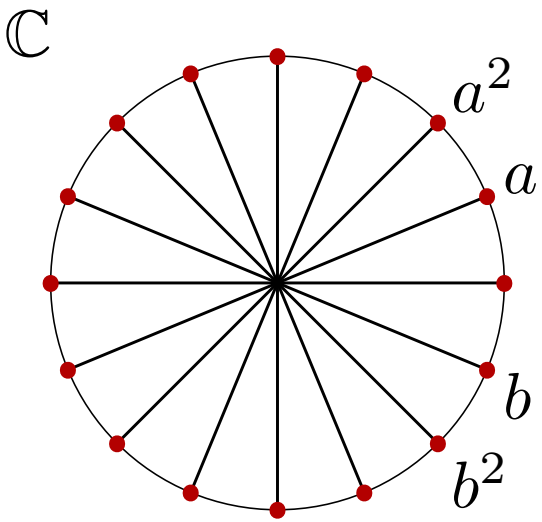
Voor later: $ab = 1$

Dit is op afstand het meest gebruikte matrix-vectorproduct ter wereld

Elke seconde worden er wereldwijd vele miljarden van uitgerekend.

Twee redenen:

- interpretatie
- snelheid



Curieus:

$$\begin{bmatrix} 1 & 1 & 1 & \cdots \\ 1 & a^1 & a^2 & \cdots \\ 1 & a^2 & a^4 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \cdots \\ 1 & b^1 & b^2 & \cdots \\ 1 & b^2 & b^4 & \cdots \\ \vdots & \vdots & \vdots & \cdots \end{bmatrix} = \begin{bmatrix} n & 0 & 0 & \cdots \\ 0 & n & 0 & \cdots \\ 0 & 0 & n & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

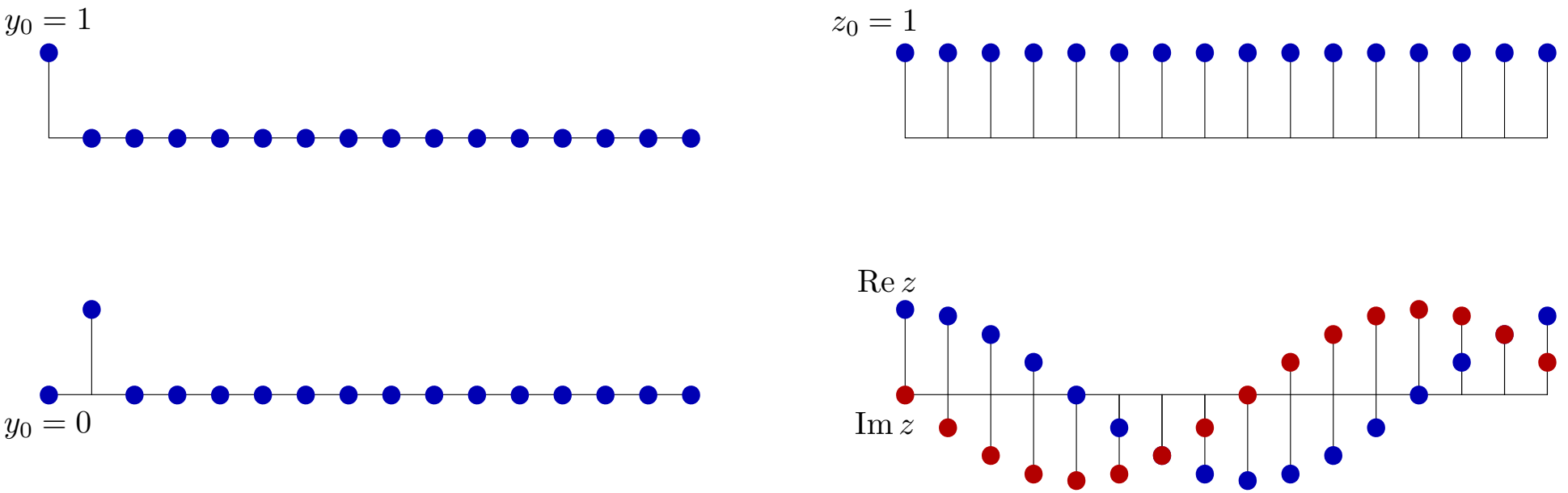
Conclusie: de Fouriertransformatie

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots \\ 1 & b^1 & b^2 & \cdots \\ 1 & b^2 & b^4 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix}$$

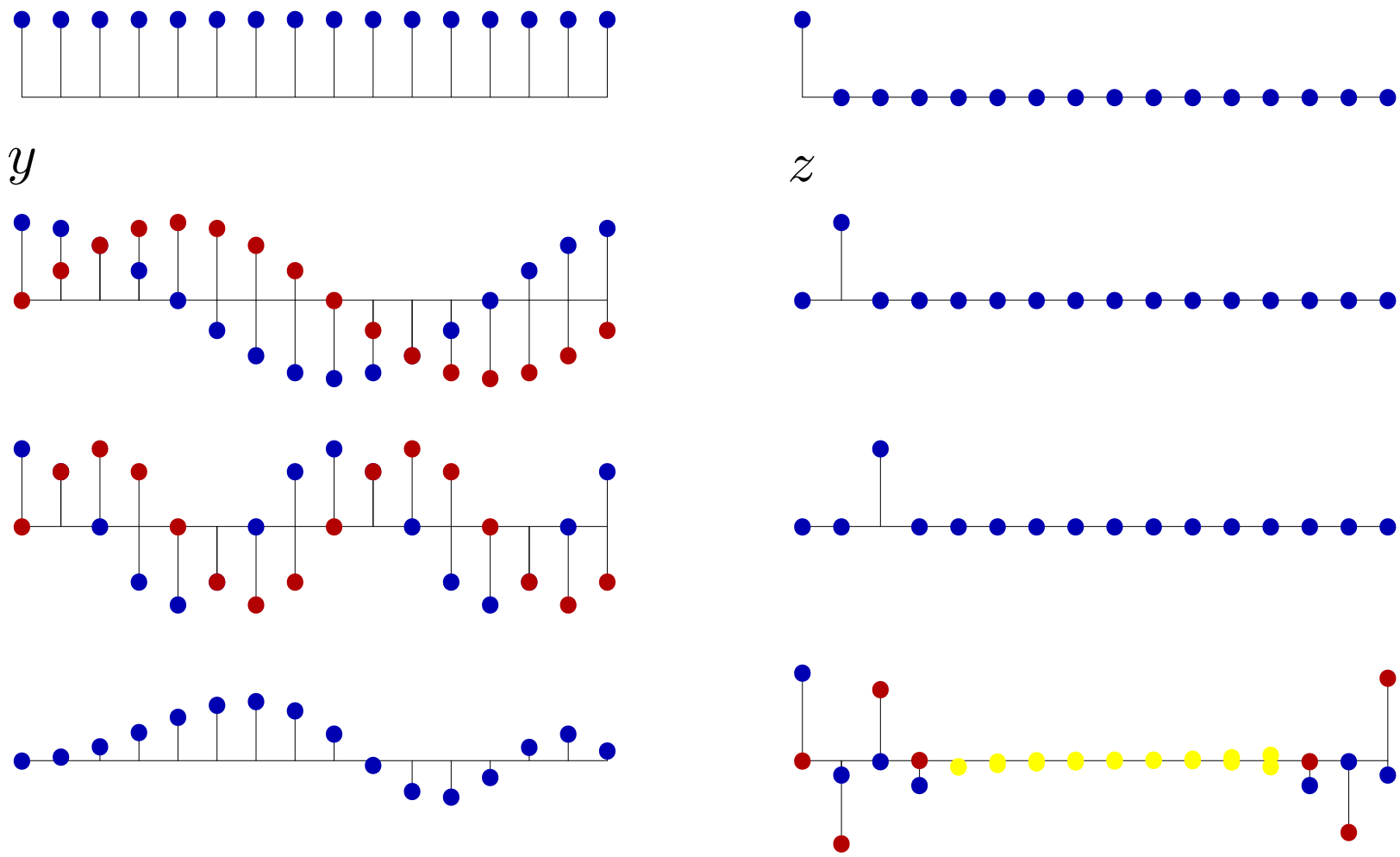
is inverteerbaar & de inverse heeft dezelfde structuur

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots \\ 1 & a^1 & a^2 & \cdots \\ 1 & a^2 & a^4 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \end{bmatrix}$$

Elke y_k wordt verdeeld over z:



Netzo wordt z_m verdeeld over y & geeft z_m trilling in y aan:



Het transformeert "gladde functie" y naar "kleine" z_m!

... toen gebeurde er een wonder (1965)

Het lijkt erop dat dit matrix-vectorproduct $\mathcal{O}(n^2)$ operaties vergt

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & b^1 & b^2 & b^3 \\ 1 & b^2 & b^4 & b^6 \\ 1 & b^3 & b^6 & b^9 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \text{of} \quad \begin{bmatrix} \tilde{z}_0 \\ \tilde{z}_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & b^1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

Echter de $n \times n$ matrix bevat slechts n verschillende elementen.
Die structuur kan worden uitgebuit!

Na wat herordenen wordt een 4-punts FT twee 2-punts FTs:

$$\begin{bmatrix} z_0 \\ z_2 \\ \hline z_1 \\ z_3 \end{bmatrix} = \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & b & 0 & 0 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & b \end{array}\right] \begin{bmatrix} y_0 + y_2 \\ y_1 + y_3 \\ \hline (y_0 - y_2)b^0 \\ (y_1 - y_3)b^1 \end{bmatrix}$$

Dus het aantal operaties $P(n)$ voldoet aan

$$P(n) = 2P(n/2) + \frac{3}{2}n, \qquad P(1) = 0.$$

Oplossing:

$$\boxed{P(n) = \tfrac{3}{2}\, n\, \log_2(n)} \quad \lll\lll\lll \quad \mathcal{O}(n^2) \qquad \boxed{n = 2^m}$$

Fast Fourier Transform (FFT)

Cooley & Tukey, 1965

- Bedacht door Tukey tijdens een President Kennedy's Science Advisory Committee meeting
- Aanleiding: detecteren van nucleaire tests (Sovjet) op grond van seismologische data
- Aanleiding: long-range acoustic detection of nuclear submarines
- Grootschalige snelle bemonstering (sampling) werd in dezelfde periode mogelijk

(Gauss 18?? & Yates 1932 & Runge 1903-1905)

"A paper by Cooley and Tukey described a recipe for computing Fourier coefficients of a time series that used many fewer machine operations than did the straightforward procedure...What lies over the horizon in digital signal processing is anyones guess, but I think it will surprise us all." [Bogert 1967]

"someday radio tuners will operate with digital processing units. I have heard this suggested with tongue in cheek, but one can speculate" [Cooley 1969]

Hewlett-Packard 5451A Fourier Analyzer, December 1972.

- IBM heeft FFT niet gepatenteerd

  (Tukey was niet IBM & software werd niet patenteerbaar geacht)
- In de film No Way Out zegt de DSP-guru:

  "Fourier transform the image!"

  ....om Kevin Costner te helpen details te zien in de foto
- Uit een boek:

  "If you speed up any nontrivial algorithm
  by a factor of a million or so
  the world will beat a path towards finding
  useful applications for it"
- Bewezen meest efficient

Vier toepassingen

- routers
- jpeg
- producten
- antialiasing

Voorbeeld 1: Routers

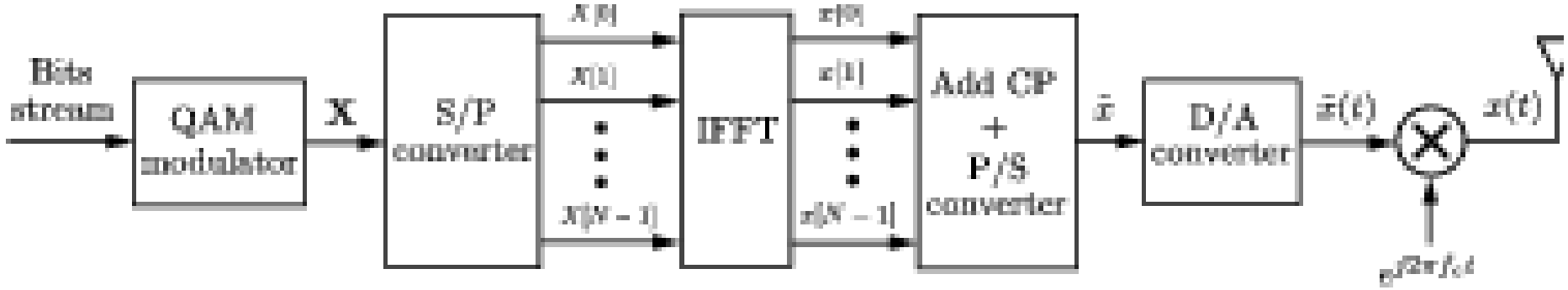


Figure 2.11: OFDM transmitter.

250000 (jawel: 250-duizend) keer per seconde wordt een FFT berekend van een pakketje van 64 datapunten.

Op die manier worden getallen y_k verdeeld over verschillende "muziekinstrumenten"

Voorbeeld 2: JPEG 1992

Er is ook een twee-dimensionale Fouriertransformatie.

Het transformeert matrices $y_{k,n}$ naar matrices $z_{k,n}$.

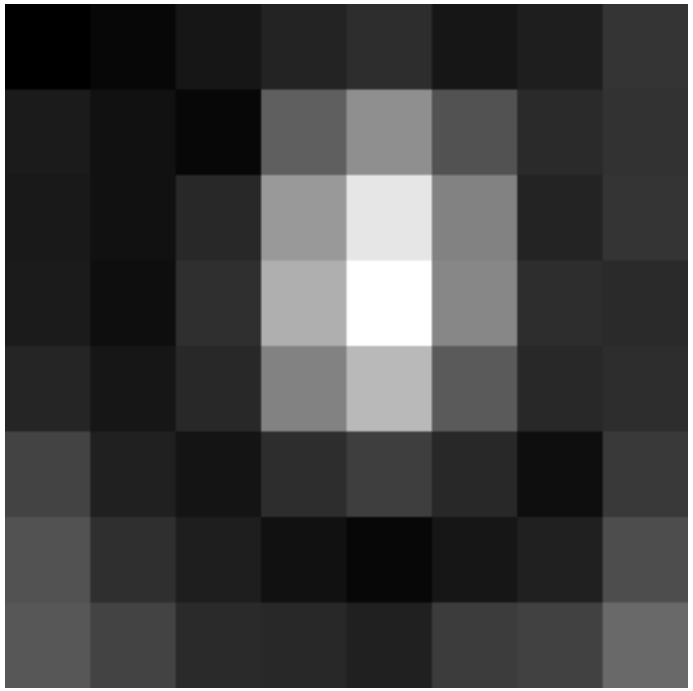
$$Y = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad Z = \begin{bmatrix} 45 & -1.5 + 0.87i & -1.5 - 0.87i \\ -13.5 + 7.79i & 3.46i & -3 + 1.73i \\ -13.5 - 7.79i & -3 - 1.73i & -3.46i \end{bmatrix}$$

Plaatjes zijn matrices van getallen (intensiteiten).
(0 =zwart en 255 =wit)

JPEG-Algoritme:

Stap 1: verdeel plaatje in een serie van 8×8 vakjes.

Stap 2: pas per 8×8 vakje (matrix) een 2D-FFT toe.



$$F = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

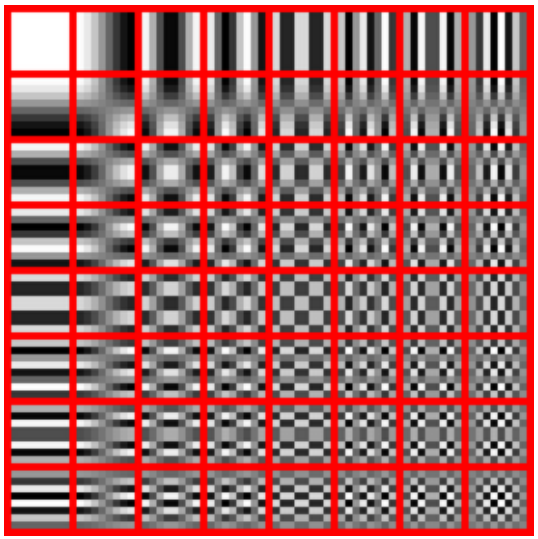
Stap 3: centreer (minus 128):

$$Y := F - 128$$

$$= \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

Bepaal FFT (cosinus-transformatie) en rond af

$$Z = \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$



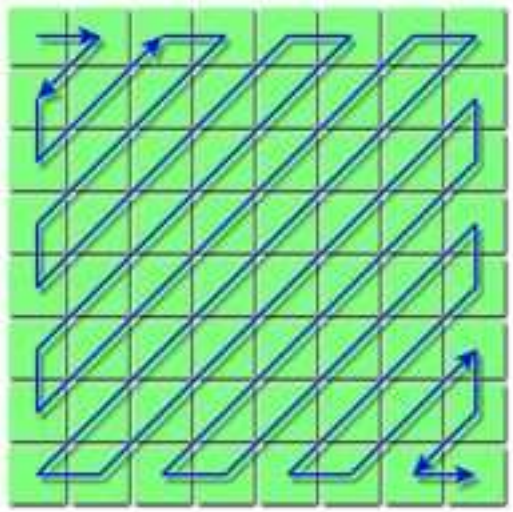
Raadpleeg de quantization matrix “drempelmatrix”

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Bepaal geschaalde (en afgeronde)

$$Z./Q = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Daar maken we een rijtje getallen van



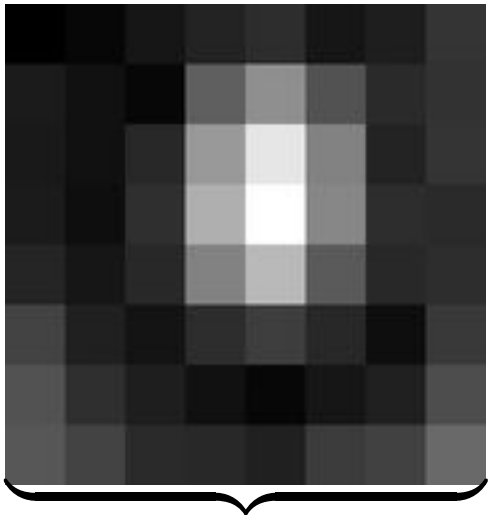
$$26, 3, 0, 3, 2, 6, 2, 4, 1, 4, 1, 1, 5, 1, 2, 1, 1, 1, 2, 0, 0, 0, 0, 0, 1, 1, 0, 0, \cdots$$

en dat zetten we gezipt (Huffman) in het jpeg-bestand. Klaar!

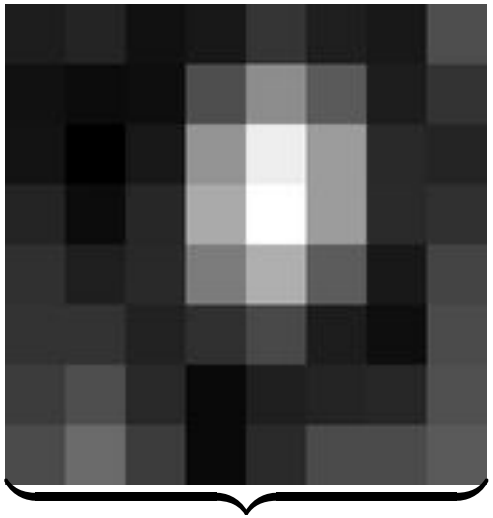
In jouw computer:

$$\text{JPEG} .* Q = \begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Haal door IFFT en tel er 128 bij op:



origineel



jpeg

Ziedaar het resultaat

$Q100$

$Q050$

$Q010$

met dank aan wikipedia

Producten van getallen

Er zit ook veel structuur (convolutie) in producten:

$$
\begin{array}{r}
123456789 \\
563150231 \\
\hline
123456789 \\
3703703670 \\
\vdots \\
6172839450000000 \\
\hline
695???????????????
\end{array}
$$

I.p.v. $\mathcal{O}(n^2)$ kan ook dit in $\mathcal{O}(n \log_2(n))$. Is ook Fouriertransformatie.

Antialiasing