Numerieke Methoden
3de Bachelor Fysica

Practicum 3:
Interpolatie en splines

Contact: jeroen.mulkers@uantwerpen.be U.305

1. Beschouw de functie
$$f(x) = \frac{1}{1+x^2}.$$
Bepaal de polynoom van graad 10 die door de punten met x-waarden $-5, -4, ..., 4, 5$ gaat, je kan hiervoor het commando polyfit. Plot beide figuren. Bepaal en teken ook de spline interpolatie, gebruik hiervoor het commando spline. Let erop om het resultaat van beide interpolaties te evalueren op een fijner grid dan de gegeven x-waarden.

2. Maak een tekening van je hand. Start met

```
figure
[x,y]=ginput;
```

Plaats je hand op het computerscherm en klik met je muis enkele punten aan die de contouren vormen van je hand. Stop de input via ginput door op de `enter`-toets te drukken. Beschouw nu x en y als twee functies van een onafhankelijke variabele die gaat van 1 t.e.m. het aantal verzamelde punten. Beide functies kan je nu interpoleren op een fijnere grid. Maak een figuur van het resultaat. Doe dit zowel voor spline als voor polynomiale interpolatie.

3. Tot nu toe hebben met het spline commando steeds met drie argumenten gebruikt: de te fitten x- en y-waarden, en de x-waarden waarin we de spline interpolatie willen kennen. Echter kijk wat MATLAB geeft als je enkel twee argumenten meegeeft, nl de te fitten x- en y-waarden:

```
>> spdata = spline(x,y)
```

De matrix coefs bevat de coëfficiënten van de $n-1$ polynomen

$$S_j(x) = a_j(x-x_j)^3 + b_j(x-x_j)^2 + c_j(x-x_j) + d_j, \;\; x_j \leq x \leq x_{j+1}.$$

Om die te kunnen zien moet je de spline-informatie in spdata opsplitsen via het commando

≫ [breaks,spcoefs,pieces,order,dim] = unmkpp(spdata);

spcoefs bevat dan de spline coëfficiënten in de volgorde a_j, b_j, c_j, d_j. Toon deze coëfficiënten op het scherm. Waarom is $a_1 = a_2$ en $a_{n-2} = a_{n-1}$? Waarom is $d_j = y_j$? Kijk hiervoor naar wat er in de cursus over splines staat.

Met MATLAB kan je ook een stuksgewijze polynoom construeren met de commando's mkpp en ppval . Als voorbeeld gaan we de volgende stuksgewijze polynoom construeren:

$$y = 2(x-1) - 4 \text{ voor } 1 \leq x \leq 2;$$
$$y = 3(x-2) + 6 \text{ voor } 2 \leq x \leq 6.$$

(Merk op dat er telkens staat $(x - x_{\text{begin interval}})$.) Daarvoor moeten we een vector aanmaken die de intervallen weergeeft

≫ breaks = [1 2 6]

en de matrix met de coëfficiënten

≫ coef = [2 -4 ; 3 6]

De stuksgewijze polynoom kan je dan construeren en tekenen via

```
≫ lindata = mkpp(breaks,coef)
≫ xi = [1:0.1:6];
≫ yi = ppval(lindata, xi);
≫ plot(xi,yi)
```

Test dit.

Beschouw nu volgende dataset:

x	1.0	2.3	3.1	4.0	5.2	5.9	6.7
y	1.7	2.8	3.6	4.5	3.4	3.1	3.1

Gebruik nu bovenstaande procedure om de afgeleide van de spline interpolatie van deze dataset te construeren en te tekenen. Maak hiervoor gebruik van het commando polyder en polyval.

4. **Trigonometrische interpolatie**

Een functie schrijven als een lineaire combinatie van sinussen en cosinussen ontleedt de functie in zijn frequentiecomponenten, net zoals een prisma een lichtbundel opsplitst in zijn verschillende kleurcomponenten. De coëfficiënten van de trigonometrische basisfuncties zeggen ons dan welke frequenties aanwezig zijn in de functie en in welke mate. Bovendien is het in vele toepassingen handiger om te werken met de functierepresentatie in de frequentieruimte (i.p.v. met de voorstelling in de reële ruimte), bv. in signaalverwerking of bij het oplossen van differentiaalvergelijkingen. We zullen nu zien dat **trigonometrische interpolatie** van een gesamplede functie verrassend snel en efficiënt kan uitgevoerd worden met het algoritme van de 'fast fourier transform'.

Onderstel dat $f(t)$ een periodische functie is met periode T, en dat we de n datapunten $(t_0, f_0), \ldots, (t_{n-1}, f_{n-1})$ willen interpoleren, met $f_k = f(t_k)$ en

$$t_k = k\frac{T}{n}, \text{ met } k = 0 : n-1.$$

Omdat de data periodisch is heeft het meer zin om te interpoleren met periodische functies dan met een polynoom. Daarom gaan we een interpolant gebruiken die een lineaire combinatie is van sinussen en cosinussen i.p.v. een interpolant die een lineaire combinatie is van $1, x, x^2$, enz.

Als j geheel is dan hebben de functies $\cos(2\pi jt/T)$ en $\sin(2\pi jt/T)$ dezelfde periode T als f, en daarom schrijven we

$$F(t) = \frac{1}{n}\left(\sum_{j=0}^{m}\left(A_j \cos\left(\frac{2\pi j}{T}t\right) + B_j \sin\left(\frac{2\pi j}{T}t\right)\right)\right),$$

met $n = 2m$. Nu wensen we de coëfficiënten A_k en B_k te bepalen zodat $F(t_k) = f_k$ voor $k = 0 : n-1$. Dit levert dus n vergelijkingen op om de $2(m+1) = 2m+2$ onbekenden te bepalen. Maar merk op dat B_0 en B_m niet moeten bepaald worden want $\sin(2\pi jt_k/T) = 0$ als $j = 0$ en $j = m$. Dus we moeten enkel de coëfficiënten $A_0, \ldots, A_m$ en $B_1, \ldots, B_{m-1}$ bepalen zodat als

$$\begin{aligned} F(t) &= \frac{1}{n}\left(A_0 + \sum_{j=1}^{m-1}\left[A_j \cos\left(\frac{2\pi j}{T}t\right) + B_j \sin\left(\frac{2\pi j}{T}t\right)\right]\right. \\ &\quad \left. + A_m \cos\left(\frac{2\pi m}{T}t\right)\right), \end{aligned} \tag{1}$$

dan $F(t_k) = f_k$ voor $k = 0 : n-1$.

Met de substituties

$$Y_0 = A_0, Y_m = A_m, Y_j = (A_j - iB_j)/2, Y_{n-j} = (A_j + iB_j)/2,$$

met $j = 1, 2, \ldots, n-1$, kunnen we de interpolerende reeks herschrijven als

$$F(t) = \frac{1}{n} \sum_{j=0}^{n-1} Y_j e^{i2\pi jt/T}.$$

De coëfficiënten Y_k zijn nu complex. We vinden ze door volgend stelsel op te lossen:

$$F(t_k) = f_k = \frac{1}{n} \sum_{j=0}^{n-1} Y_j e^{i2\pi jk/n} = \frac{1}{n} \sum_{j=0}^{n-1} Y_j \omega_n^{-jk}, \text{ voor } k = 0 : n-1$$

met $\omega_n = e^{-2\pi i/n}$. In matrixvorm wordt dit

$$\mathbf{f} = \frac{1}{n} \Omega_n^{-1} \mathbf{Y}, \tag{2}$$

met $[\Omega_n^{-1}]_{jk} = \omega_n^{-jk}$. Zo ziet deze matrix eruit voor $n = 4$:

$$\Omega_4^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}.$$

Merk op dat Ω_n^{-1} een Vandermonde matrix is.

Belangrijk om op te merken is dat de inverse matrix eenvoudig is en expliciet kan opgeschreven worden:

$$\frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} =$$

$$\frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & -i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dit geeft $[\Omega_n^{-1}]^{-1} = \Omega_n/n$ met $[\Omega_n]_{jk} = \omega_n^{jk}$. We kunnen dus ook eenvoudig de coëfficënten Y_k berekenen:

$$Y_k = \sum_{j=0}^{n-1} f_j \omega_n^{jk}, \text{ voor } k = 0 : n-1 \text{ of } \mathbf{Y} = \Omega_n \mathbf{f}. \tag{3}$$

Dit noemt men de discrete Fourier transformatie (DFT). Dus i.p.v. het stelsel (2) op te lossen, wat $O(n^3)$ berekingen kost, is het beter om gewoon

bovenstaande som (3) uit te rekenen, wat niets anders is dan een matrix-vermenigvuldiging en slechts $O(n^2)$ berekeningen kost. Het doel van deze sectie is te laten zien dat er een algoritme bestaat, nl. de 'Fast Fourier Transform', dat de kost reduceert tot $O(n \log_2 n)$.

De DFT (3) ontbindt dus een gegeven input $\mathbf{f}$ in zijn frequentiecomponenten die gegeven worden door $\mathbf{Y}$. Twee componenten zijn extra belangrijk. Ten eerste Y_0, die overeenkomt met frequentie nul (de DC component, een constante functie), en ten tweede $Y_{n/2}$, die overeenkomt met de *Nyquist frequentie*, de grootste representeerbare frequentie bij de gegeven sampling rate. Uit (1) volgt dat deze Nyquist fequentie $\nu_{max} = m/T = n/(2T)$ is. Het is dus belangrijk om een signaal te samplen met een voldoende snelheid zodat er tenminste 2 punten gesampled worden per cyclus van de hoogste frequentie component. Stel dat dit niet het geval is, bv. het signaal bevat een component met frequentie $\nu_1 > \nu_{max}$, dan zal dit een ongewenste bijdrage leveren tot het DFT spectrum bij een frequentie ν_2 waarvoor geldt dat $\nu_1 + \nu_2 = 2\nu_{max}l$ of $\nu_1 - \nu_2 = 2\nu_{max}l$, met l een geheel getal.

Het 'Fast Fourier Transform' algoritme

We illustreren het 'Fast Fourier Transform' algoritme (FFT) enkel met het voorbeeld voor $n = 4$. De eerste Fourier matrices Ω_n zijn:

$$\Omega_1 = 1, \qquad \Omega_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \Omega_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}.$$

Noem P_4 volgende permutatiematrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

en D_2 de diagonale matrix

$$D_2 = \mathrm{diag}(1, \omega_4) = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}.$$

Dan kunnen we schrijven

$$\begin{aligned}
\mathbf{Y} &= \Omega_4 P_4 P_4^{-1}\mathbf{f} \\
&= \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{array}\right] \left[\begin{array}{c} f_0 \\ f_2 \\ f_1 \\ f_3 \end{array}\right] \\
&= \left[\begin{array}{cc} \Omega_2 & D_2\Omega_2 \\ \Omega_2 & -D_2\Omega_2 \end{array}\right] \left[\begin{array}{c} \mathbf{f}(0:2:3) \\ \mathbf{f}(1:2:3) \end{array}\right] \\
&= \left[\begin{array}{cc} I & D_2 \\ I & -D_2 \end{array}\right] \left[\begin{array}{c} \Omega_2\mathbf{f}(0:2:3) \\ \Omega_2\mathbf{f}(1:2:3) \end{array}\right].
\end{aligned}$$

Ω_4 kan dus opgesplitst worden in 4 blokken die elk een geschaalde versie zijn van Ω_2. Op deze manier kan de 4-punts DFT $\mathbf{Y} = \Omega_4\mathbf{f}$ eenvoudig verkregen worden uit de twee 2-punts DFTs $Y_T = \Omega_2\mathbf{f}(0:2:3)$ en $Y_B = \Omega_2\mathbf{f}(1:2:3)$:

$$\begin{aligned}
\mathbf{Y}(0:1) &= Y_T + d_2 .* Y_B \\
\mathbf{Y}(2:3) &= Y_T - d_2 .* Y_B,
\end{aligned}$$

met d_2 de vector met gewichtsfactoren $[1 - i]^T$. Dus om de FFT te bekomen voor $n = 4$, moeten we enkel de FFT berekenen voor $n = 2$ en eventueel vermenigvuldigen met d_2. En zo kunnen we verder gaan tot $n = 1$ (de één-punts FFT van een getal is gewoon dat getal).

Algemeen, als $n = 2m$, is $\mathbf{Y} = \Omega_n\mathbf{f}$ gegeven door

$$\begin{aligned}
\mathbf{Y}(0:m-1) &= \mathbf{Y}_T + d .* \mathbf{Y}_B \\
\mathbf{Y}(m:n-1) &= \mathbf{Y}_T - d .* \mathbf{Y}_B,
\end{aligned}$$

met $\mathbf{Y}_T = \Omega_m\mathbf{f}(0:2:n-1)$, $\mathbf{Y}_B = \Omega_m\mathbf{f}(1:2:n-1)$ en $d = [1\omega_n \cdots \omega_n^{m-1}]$. Voor $n = 2^t$ kan zo'n decompositie uitgevoerd worden totdat $n = 1$. Daarom neemt men vaak het aantal gesamplede datapunten een macht van twee. Deze recursieve aanpak is het FFT algoritme. Aangezien er $\log_2 n$ recursieniveaus zijn, en elk niveau $O(n)$ berekeningen kost, is de totale kost van de orde $O(n \log_2 n)$. Dit FFT algoritme kan natuurlijk ook gebruikt worden om de inverse DFT (2) efficiënt te berekenen.

MATLAB kent het commando fft dat de 'Fast Fourier transform' uitvoert. Het gebruik wordt geïllustreerd in het volgende voorbeeld.

Voorbeeld 1 We wensen de DFT te bepalen van 64 equidistante datapunten die gesampled worden over een periode van 3.2 s uit de functie $y = 0.5 + 2\sin(2\pi\nu_1 t) + \cos(2\pi\nu_2 t)$, met $\nu_1 = 3.125$ Hz en $\nu_2 = 6.25$ Hz. Script DemoFFT1.m doet dit. Bestudeer en run dit script. Zorg dat je elke stap begrijpt. Zoek de betekenis van het commando bar op via help. Beantwoord volgende vragen.

(a) Wat wordt er getoond in figuur 1, 2 en 3?

(b) Wat is de betekenis van real(Yss) en imag(Yss)?

(c) Waarom staan er de regels Yss(1)=Yss(1)/2; Yss(nt/2+1)=Yss(nt/2+1)/2; in het script?

(d) Waarom is de x-as van figuur 2 onderverdeeld in 64 k waarden, terwijl de x-as in figuur 3 maar in 33 frequentiewaarden?

Nu kunnen we spelen met de parameters (de frequenties, sampletijd en -stappen).

(a) Verander bv. ν_2 in 7 Hz. Wat gebeurt er met het frequentiespectrum? Wat denk je dat de verklaring hiervoor is? Kijk ook hoe het spectrum terug verandert als je ν_2 nog laat toenemen tot 7.5 Hz.

(b) We gaan nu het aantal samplestappen variëren (met $\nu_1 = 3.125$ Hz en $\nu_2 = 7.5$ Hz). Voor het FFT algoritme geïmplementeerd in MATLAB moet het aantal stappen geen tweede macht zijn, hoewel het dan wel het efficiëntst is. Kijk wat er gebeurt met de piek bij ν_2 als je het aantal stappen verandert in $50, 48, 46, \ldots$.

(c) Neem opnieuw 64 samplestappen, en kies $\nu_2 = 105$. Verklaar waarom je een grote component ziet in het spectrum bij $\nu = 5$.

(d) Om de piek bij $\nu = 105$ Hz zichtbaar te maken zou je de sampletijd kunnen verkleinen tot $T = 0.2$ s. Wat is het nadeel hiervan? Wat is beter om te doen?

Voorbeeld 2 We gaan nu het spectrum bepalen in een reeks van 512 punten gesampled in een tijdsinterval van 2 s uit de functie

$$y = 0.5 + 0.2\sin(2\pi\nu_1 t) + 0.35\cos(2\pi\nu_2 t) + \textbf{ruis},$$

met $\nu_1 = 20$ Hz en $\nu_2 = 50$ Hz. De ruis is Gaussisch met standaarddeviatie 0.5 en gemiddelde 0. Script DemoFFT2.m toont het spectrum zonder de ruis. Breidt het script uit door de ruis toe te voegen en bekijk het spectrum.

Voorbeeld 3 Bepaal de DFT van $y = 32\sin^5(2\pi\nu t)$ met $\nu = 30$ Hz door het script DemoFFT1.m aan te passen. Gebruik 512 punten gesampled over 1 s. Schat m.b.v. de DFT de coëfficiënten a_0, a_1 en a_2 in de uitdrukking

$$y = 32\sin^5(2\pi\nu t) = a_0\sin(2\pi\nu t) + a_1\sin(2\pi(3\nu)t) + a_2\sin(2\pi(5\nu)t).$$