Check Partialtidenanalyse? - Aufgabe

Teste die Funktionen FFTPT, FFTPTScan, sFFTPTScan, sFFTPTScanCluster und FitPT.

In[1]:=
```
<< "\\\\Themis2\\system\\akprog\\Wolfram_Mathematica\\Zeitreihen\\MKToolsMM7_log.m"
? FFTPT
```

FFTPT[TS, SuchtidenListe, Optionen] –> {{Name, Frequenz[Grad/h], Amplitude, Phase[Grad], ErrorFrequenz[Grad/h], –, –, ErrorFrequenzTolerance}, ...}
Die Funktion bestimmt die Amplituden und Phasen der gesuchten Partialtiden über eine FFT.
SuchtidenListe –> {M_2, S_2, ...}

Optionen:
WindowFunction –> Hanning (Fensterfunktion; Auswahl siehe TScalcSpektrum)
ZeroPaddingFactor –> 10 (Die Anzahl der Nullen, die angehängt werden. Dieser
Faktor (N) bestimmt die N–fache Länge der Zeitreihe. Ein Faktor von 2 verdoppelt damit die Länge der Zeitreihe.)
ErrorFrequencyTolerance –> 0.0002 (Übersteigt der Wert von ErrorFrequenz den Wert von ErrorFrequencyTolerance, wird das Flag gesetzt und rot markiert. ErrorFrequenz
ist die Differenz zwischen der gesuchten PT–Frequenz und der der gesuchten Frequenz am nächsten kommenden Frequenz aus dem diskreten Spektrum.)
VerboseMode –> False (True –> mehr Infos)

Vorteile:
- eine einzige FFT
- schnell
- Frequenzgenauigkeit kann mit angegeben werden. Partialtiden, die außerhalb dieser Vorgabe liegen werden

markiert

Nachteile:
- Übereinstimmung der Peakspitzen mit dem Frequenzraster? teilweise...
- es gibt keine Garantie, dass im Zentrum des Peaks der Wert abgegriffen wird -> Amplituden- und Phaseninformation nur bedingt richtig (Energieverlust)
- Leakage und Verdeckung

In[3]:= **? FFTPTScan**

FFTPTScan[TS, SuchtidenListe, Optionen] –> {{Name, Frequenz[Grad/h], Amplitude, Phase[Grad],
Frequenzfehler[Grad/h], Wellenzahl (nur gültig bei no–zero–padding), eingefügte Nullen, Rückgabewert (komplex)}, ...}
Die Funktion bestimmt die Amplituden und Phasen der gesuchten Partialtiden über eine FFT. Zuvor wird die optimale einzufügende Anzahl von Nullen (zero padding) berechnet und zwar so, dass die optimale Abtastfrequenz (Partialtidenfrequenz) erreicht wird. In Frequenzfehler wird die Abweichung der Partialtidenfrequenz von der tatsächlich abgetasteten Frequenz zurückgegeben. Das Abbruchkriterium ist der Parameter MaxZeroes, der das Maximum an einzufügenden Nullen vorgibt.
SuchtidenListe –> {M_2, S_2, ...}

Optionen:
WindowFunction –> Hanning (Fensterfunktion [Rechteck, Fejer, Hanning, Hamming, FlatTop, Gauss, Kaiser, Blackman, Blackman2, Triplett])
MaxZeroes –> 100000 (Vorgabe der maximalen Anzahl einzufügender Nullen; Abbruchkriterium)
VerboseMode –> False (True –> mehr Infos)

Vorteile:
- für jede Partialtide wird die optimale Anzahl von Nullen angehängt. Optimal: so viele Nullen, dass die Fensterbreite möglichst gut der Partialtidenwellenlänge entspricht (und damit Abtastung sehr nahe des Zentrums)

Nachteile:
- eine FFT für jede Partialtide
- hoher Zeitbedarf
- Leakage und Verdeckung

In[4]:= **? sFFTPTScan**

sFFTPTScan[TS, SuchtidenListe, Optionen] –> {{Name, Frequenz[Grad/h], Amplitude, Phase[Grad],
ErrorFrequenz[Grad/h], Wellenzahl (nur gültig bei no–zero–padding), eingefügte Nullen, Rückgabewert (komplex)}, ...}
Die Funktion bestimmt die Amplituden und Phasen der gesuchten Partialtiden sukzessiv über eine FFT (s. FFTPTScan). Nach jeder Analyse
wird die Partialtide mit der größten Amplitude bestimmt. Diese wird aus dem Signal entfernt und die Analyse geht weiter. Damit werden
kleine Partialtiden, die auf Schultern von sehr nahen und sehr hohen Partialtiden sitzen, realistischer detektiert als ohne die Elimination.
SuchtidenListe –> {M_2, S_2, ...}

Optionen:
WindowFunction –> Hanning (Fensterfunktion [Rechteck, Fejer, Hanning, Hamming, FlatTop, Gauss, Kaiser, Blackman, Blackman2, Triplett])
MaxZeroes –> 100000 (Vorgabe der maximalen Anzahl einzufügender Nullen; Abbruchkriterium)
VerboseMode –> False (True –> mehr Infos)

Vorteile:
- für jede Partialtide wird die optimale Anzahl von Nullen angehängt. Optimal: so viele Nullen, dass die Fensterbreite möglichst gut der Partialtidenwellenlänge entspricht (und damit Abtastung sehr nahe des Zentrums)
- durch das sukzessive Entfernen beginnend mit den Partialtiden mit der höchsten Energie (Amplitude) wird das Auslaufen (Leakage) verringert und eventuell verdeckte kleine Partialtiden sichtbar/detektierbar (ACHTUNG!: man kann damit nicht die Grenzen der Auflösbarkeit aufheben!)

Nachteile:
- eine FFT für jede Partialtide
- hoher Zeitbedarf (noch etwas höher als bei FFTPTScan durch die Entfernung aus dem Signal)
- wird der erste (höchste) Peak nicht richtig detektiert, wird in der Folge mehr/weniger Energie abgezogen als tatsächlich vorhanden war, die entstehenden Sidelobes dieses Restpeaks können umliegende Partialtiden beeinflussen

In[5]:= **? sFFTPTScanCluster**

sFFTPTScanCluster [TS, SuchtidenListe, Optionen] –> {{Name, Frequenz[Grad/h], Amplitude,
Phase[Grad], ErrorFrequenz[Grad/h], Wellenzahl (nur gültig bei no–zero–padding), eingefügte Nullen, Rückgabewert (komplex)}, ...}
Die Funktion bestimmt die Amplituden und Phasen der gesuchten Partialtiden sukzessiv über eine FFT (s. sFFTPTScan). Die Analyse folgt Cluster für Cluster.
SuchtidenListe –> {Cluster1, Cluster2, ..., Cluster N}; Cluster1..N = {PT1, PT2, ... PT M}

Optionen:
WindowFunction –> Hanning (Fensterfunktion [Rechteck, Fejer, Hanning, Hamming, FlatTop, Gauss, Kaiser, Blackman, Blackman2, Triplett])
MaxZeroes –> 100000 (Vorgabe der maximalen Anzahl einzufügender Nullen; Abbruchkriterium)
VerboseMode –> False (True –> mehr Infos)

Vorteile:
- für jede Partialtide wird die optimale Anzahl von Nullen angehängt. Optimal: so viele Nullen, dass die Fensterbreite möglichst gut der Partialtidenwellenlänge entspricht (und damit Abtastung sehr nahe des Zentrums)
- durch das sukzessive Entfernen beginnend mit den Partialtiden mit der höchsten Energie (Amplitude) wird das Auslaufen (Leakage) verringert und eventuell verdeckte kleine Partialtiden sichtbar/detektierbar (ACHTUNG!: man kann damit nicht die Grenzen der Auflösbarkeit aufheben!)
- performanter, da davon ausgegangen wird, dass sich nur Partialtiden innerhalb eines Cluster gegenseitig beeinflussen. Im Spektrum weit entfernte Partialtiden können dadurch gleichzeitig detektiert werden.

Nachteile:
- hoher Zeitbedarf (jedoch geringer als bei sFFTPTScan, durch die Clusterung)
- wird der erste (höchste) Peak nicht richtig detektiert, wird in der Folge mehr/weniger Energie abgezogen als tatsächlich vorhanden war, die entstehenden Sidelobes dieses Restpeaks können umliegende Partialtiden beeinflussen

In[6]:=
```
Fit
```

Out[6]=
```
Fit
```

In[7]:= **? FitPT**

FitPT [TS, SuchtidenListe, Optionen] –> {{Name, Frequenz[Grad/h], Amplitude, Phase[2*Pi!!!]}, ...}
Die Funktion bestimmt die Amplituden und Phasen der gesuchten Partialtiden über ein Regressionsverfahren.
SuchtidenListe –> {M_2, S_2, ...}

Optionen:
MaxIterations –> 1000 (ohne Funktion)
VerboseMode –> False (True –> mehr Infos)

◄ | ►

Vorteile:
- kennt man die Zusammensetzung des Signals (nicht die Quantität, nur die Qualität) ist diese Methode exakt bei den Amplituden (Phasen ist noch ein anderes Problem, da transiente Funktionen)
- geht auch mit Datenlücken

Nachteile:
- man muss die Modellfunktion aufstellen, also vorher kennen!
- bei ungenügender Kenntnis der Modellfunktion, kann Energie von einer nicht berücksichtigten zu einer in der Modellfunktion berücksichigten Partialtide "wandern"

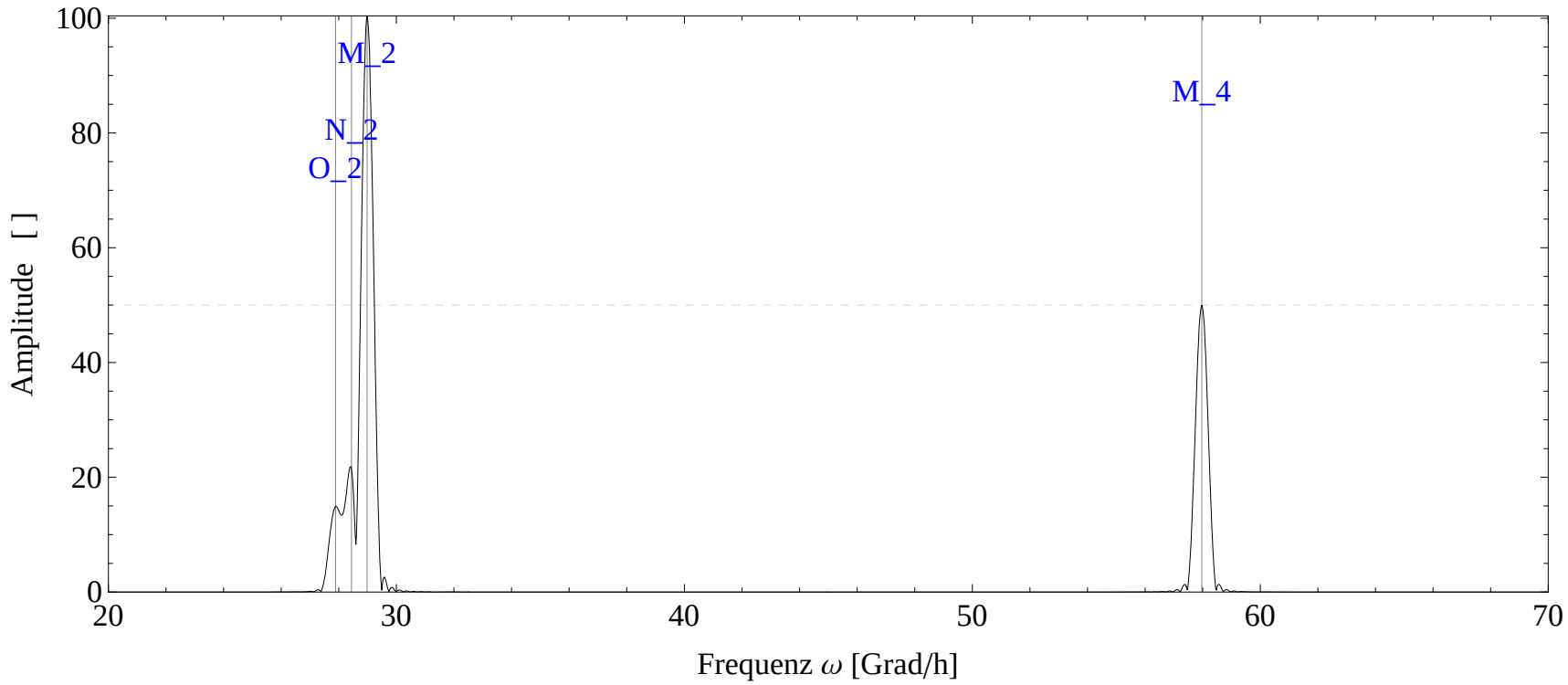
Check Partialtidenanalyse? - Lösung

Teste die Funktionen FFTPT, FFTPTScan, sFFTPTScan, sFFTPTScanCluster und FitPT.

In[8]:=
```
<< "\\\\Themis2\\system\\akprog\\Wolfram_Mathematica\\Zeitreihen\\MKToolsMM7_log.m"
TS = PTSynthese[{{"M_2", 100, 180}, {"N_2", 20, 0}, {"O_2", 15, 50}, {"M_4", 50, 0}}, AbsoluteTime[{2000, 1, 1, 0, 0, 0}],
   Anfangszeit -> AbsoluteTime[{2000, 1, 1, 0, 0, 0}], Endzeit -> AbsoluteTime[{2000, 3, 1, 0, 0, 0}]];
TSFrame[TS]
cspec = TScalcSpektrum[TS, ZeroPaddingFactor → 10];
ListPlotSpektrum[cspec, ISize → 800, showPTList → {"M_2", "M_4", "N_2", "O_2"}, XRange → {20, 70}, Threshold → 50]
```

Out[10]= 01.01.2000 00:00:00 - 01.03.2000 00:00:00 --> 2881 Datensätze. Länge: 0a60d0h0m0s

Out[12]=



In[13]:=
```
suchtidencluster = {{"M_2", "M_4", "N_2"}};
Flatten[%]
```

Out[14]=
```
{M_2, M_4, N_2}
```

In[15]:=
```
FFTPT[TS, Flatten[suchtidencluster]] // TableForm
```

Out[15]//TableForm=

M_2	28.9841	100.395	175.657	−0.00582981	0	0	1
M_4	57.9682	49.9299	351.605	−0.0116596	0	0	1
N_2	28.4397	21.4298	2.4433	−0.000395423	0	0	1

In[16]:=
```
FFTPTScan[TS, Flatten[suchtidencluster]] // TableForm
```

Out[16]//TableForm=

M_2	28.9841	100.404	179.863	-8.04561×10^{-7}	2563	60 787	0.239539 − 100.403 i
M_4	57.9682	50.	359.999	-1.60912×10^{-6}	2563	28 953	−0.00101106 + 50. i
N_2	28.4397	21.4409	2.74308	-9.41988×10^{-8}	1683	39 727	1.02611 + 21.4163 i

In[17]:=
```
sFFTPTScan[TS, Flatten[suchtidencluster]] // TableForm
```

Out[17]//TableForm=

M_2	28.9841	100.404	179.863	-8.04561×10^{-7}	2563	60 787	0.239539 − 100.403 i
M_4	57.9682	50.	359.999	-1.60912×10^{-6}	2563	28 953	−0.0010105 + 50. i
N_2	28.4397	19.6524	359.801	-9.41988×10^{-8}	1683	39 727	−0.0681738 + 19.6523 i

In[18]:=
```
sFFTPTScanCluster[TS, suchtidencluster] // TableForm
```

Out[18]//TableForm=

M_2	28.9841	100.404	179.863	-8.04561×10^{-7}	2563	60 787	0.239539 − 100.403 i
M_4	57.9682	50.	359.999	-1.60912×10^{-6}	2563	28 953	−0.0010105 + 50. i
N_2	28.4397	19.6524	359.801	-9.41988×10^{-8}	1683	39 727	−0.0681738 + 19.6523 i

In[19]:=
```
FitPT[TS, Flatten[suchtidencluster]] // TableForm
```

Out[19]//TableForm=

M_2	28.9841	99.119	−11.6933
M_4	57.9682	50.0422	33.1558
N_2	28.4397	21.2582	0.983739