Übungen zu: "Echtzeitsysteme": Digitales Filter

Aufgabe 1 : **Modul digitale Filter**

Es ist ein **Programm-Modul "DigitalFilter" (Quellcode- und Header-Datei)** in C zu schreiben, das ein digitales Bandpassfilter realisiert.

Das Filter soll als C-Funktion mit `static`-Variablen als Speicher für alte Abtastwerte implementiert werden. Der Einfachheit halber sollen zunächst nur Mono-Signale verarbeitet werden können, für Stereo würde man zwei Funktionen brauchen: rechter und linker Kanal mit jeweils *eigenen* Speichern.

Ein digitales Filter wird durch eine Differenzengleichung beschrieben. Die Differenzengleichung eines digitalen Bandpassfilters (mono) ist gegeben durch:

$$y(n) = D_0[x(n-2) - x(n)] - C_0 y(n-2) - C_1 y(n-1) \tag{1}$$

Die Koeffizienten D_0, C_0 und C_1 sind reelle Zahlen, $x(n)$ ist der aktuelle Messwert des Signals, $y(n)$ der aktuell berechnete Ausgangswert, $y(n-1)$, $y(n-2)$ sind bereits berechnete Werte. Es gilt:

$$y(n) = x(n) = 0 \quad \text{für} \quad n < 0 \tag{2}$$

Filterkoeffizenten

Die Eigenschaften des Filters werden wesentlich durch die Filterkoeffizienten bestimmt. Ihre Berechnung ist nicht Gegenstand dieser Veranstaltung - sie werden hier vorgegeben.

Es wurde ein Filterentwurf für 8 Kanäle durchgeführt. Die Mittenfrequenzen sind beim Digitalfilter automatisch auf die Abtasfrequenz normiert, der Bandpass funktioniert also nicht nur bei einer, sondern bei jeder Abtastfrequenz. Der Entwurf wurde für eine Abtastfrequenz $f_a = 22$kHz vorgenommen. Bei dieser Abtastfrequenz ergeben sich Mittenfrequenzen f_0 von 10kHz, 5kHz, 2.5kHz, 1.25kHz, 625Hz, 313Hz, 156Hz, 78Hz.

f_0/f_a	D_0	C_0	C_1
0,4535	-0,092396	0,815208	1,738331
0,2268	-0,259144	0,481712	-0,215618
0,1134	-0,187715	0,624570	-1,229489
0,0567	-0,109755	0,780490	-1,668733
0,0283	-0,058942	0,882116	-1,852347
0,0142	-0,030530	0,938940	-1,931233
0,0071	-0,015468	0,969064	-1,967119
0,0035	-0,007796	0,984407	-1,983917

Die Filter reagieren empfindlich auf Rundung der Koeffizienten. Die Berechnung der Differenzengleichungen soll in Gleitkommaarithmetik erfolgen, um Rundungsprobleme zu vermeiden.

- Schreiben Sie eine Funktion `BP_Filter()`, die als Parameter den aktuellen Abtastwert $x(n)$ erhält und als Ergebnis den aktuellen Wert $y(n)$ berechnet.
- Schreiben Sie ein Hautpprogramm, in dem das Filter getestet wird:
  - Die "Impulsantwort" des Filters erhält man, wenn man die Zahlenfolge $x(n) = 1, 0, 0, 0, 0...$ als Eingangssignal wählt.

- Die "Sprungantwort" des Filters erhält man, wenn man die Zahlenfolge $x(n) = 1, 1, 1, 1, 1, 1, ...$ als Eingangssignal wählt.
- Interpretieren Sie die Ergebnisse.

Hinweise:

Folgende Funktion ist sehr nützlich, wenn man die Daten mit dem freien Programm "gnuplot" oder auch mit Excel / Openoffice anschauen will:

```
int SaveArray(double *a, int n, char *DateiName)
{   FILE *fp;
    int k;
    if(NULL==(fp=fopen(DateiName, "wt"))) return -1;
    for(k=0; k<n; k++)
    {  fprintf(fp,"%lf\n", a[k]);
    }
    fclose(fp);
    return 0;
}
```

Die Funktion speichert ein `double`-Feld in eine Textdatei, die von "gnuplot" gelesen und dargestellt werden kann.

Wer noch Zeit hat kann die Impuls- und Sprungantworten eines digitalen Tiefpasses analysieren:

$$y(n) = ((1-a)x(n) + ay(n-1) \quad \text{mit} \quad 0 \leq a < 1 \tag{3}$$

wobei gilt

$$y(n) = x(n) = 0 \quad \text{für} \quad n < 0 \tag{4}$$

Experimentieren Sie mit den Werten von a. Welche Wirkung hat der Parameter?