Dokumentation der Programme `fftw4` und `fftwd2`

Zur Installation der FFTW-Bibliotheken

Die Bibliothek kann unter http://www.fftw.org/download.html runtergeladen werden. Ich habe Version 3.2.1 benutzt. Sie kann in etwa wie folgt installiert werden:

```
# ./configure --prefix=... --enable-sse2
# make
# make install
```

Bei prefix wird der Pfad angegeben, in den die Bibliothek installiert werden soll, z.B. /opt/fftw321/.

Man muss fftw nicht unbedingt selbstständig kompilieren; vielen Distributionen liegen entsprechende Pakete bei. In den Paketquellen von Ubuntu kann man mit Synaptic z.B. die folgenden Pakete installieren: `libfftw3-3`, `libfftw3-dev`, `libfftw3-doc`.

Die C-Programme werden mit `gcc` kompiliert und gelinkt

```
$ gcc -o test fftwd2.c -I/usr/local/include -L/usr/local/lib -lm
-lfftw3 -lfftw3_threads -lpthread
```

Dabei muss mit -I/... der Pfad zu den includes und mit -L/... der Pfad zu den libraries angegeben werden, was natürlich davon abhängt, wohin/auf welche Weise man die fftw installiert hat. Die oben angegebenen Pfade beziehen sich auf die Ubuntu-Installation der obigen Pakete. -lm linkt die math.h-Bibliothek, der Rest (-lfftw3, -lfftw3_threads und -lpthread) die nötigen fftw-Bibliotheken. Man muss also nur die Ordner anpassen.

Zum Programm `fftw4.c`

Das Programm ermöglicht es, diskretisierte Funktionen aus dem Originalbereich in den Fourier-Bereich zu transformieren. Weiter unten wird auch gezeigt, wie man das Ergebnis mit Gnuplot darstellen kann. Es sind mehrere Funktionen einprogrammiert, nämlich eine Gauss-Glocke (`gaussian`), ein Einfachspalt (`sslit`), Doppelspalt (`dslit`) und Giter (`grating`).

Programmaufruf

```
$ gcc -o test fftwd2.c -I/usr/local/include -L/usr/local/lib -lm
-lfftw3 -lfftw3_threads -lpthread
$ ./test
$ gnuplot
gnuplot> p "original.txt" w l, "int.txt" w l
```

Die letzten beiden Befehlszeilen ermöglichen das Plotten mit Gnuplot, welches natürlich installiert sein muss (unter Ubuntu: `$ sudo apt-get install gnuplot-nox`). Man kann natürlich auch andere Programme nehmen; für die Darstellung in Abb. 1 auf Seite 3 habe ich Matlab verwendet.

Standardeinstellungen und Modifikation

`grating` ist die Standardeinstellung; um eine der anderen Funktionen auszugeben (oder möglicherweise eine selbstgeschriebene) muss lediglich die Zeile

```
in[i][0]=grating(x);
```

angepasst werden.

Zum Programm `fftwd2.c`

Das Programm `fftwd2.c` vergleicht verschiedene numerische Methoden des Differnzierens: Die rechtsseitige Ableitung, die zentrierte Ableitung und die Ableitung mit Hilfe von FFT. Das Programm berechnet jeweils die größten vorkommenden Abweichungen gegenüber dem analytisch berechneten Wert der Ableitung. Abb. 2 auf der nächsten Seite wurde anhand der von `fftwd2.c` gelieferten Daten erstellt und zeigt einen Vergleich der verschiedenen numerischen Methoden des Differenzierens anhand des größten auftretenden Fehlers.

Programmaufruf

```
$ gcc -o test fftwd2.c -I/usr/local/include -L/usr/local/lib
-lm -lfftw3 -lfftw3_threads -lpthread
$ ./test
$ gnuplot
gnuplot> p "original.txt" w l, "analyt.txt" w l, "approx1.txt" w l,
"approx2.txt" w l, "back.txt" w l
```

Führt man die Gnuplot-Befehle wie angegeben aus, ergibt sich mit Standardeinstellungen des Programms genau der Graph aus Abb. 2 auf der nächsten Seite.

Standardeinstellungen und Modifikation

Als Standardeinstellung habe ich die folgenden Funktionen verwendet:

$$f(x) = \frac{1}{1+x^2} \quad \text{und} \quad \frac{df}{dx} = -\frac{2x}{(1+x^2)^2}$$

Für andere Funktionen müssen die folgenden Zeilen angepasst werden:

```
orig[i]=1/(1+x*x);
...
analyt[i]=-2.0*x/((1.0+x*x)*(1.0+x*x));
```

Um Randeffekte zu vermeiden, sollte die Funktion quadratintegrabel sein; eine gute Möglichkeit ist wieder die Gauß-Glocke, die auch als `gaussian` im Programm enthalten ist.

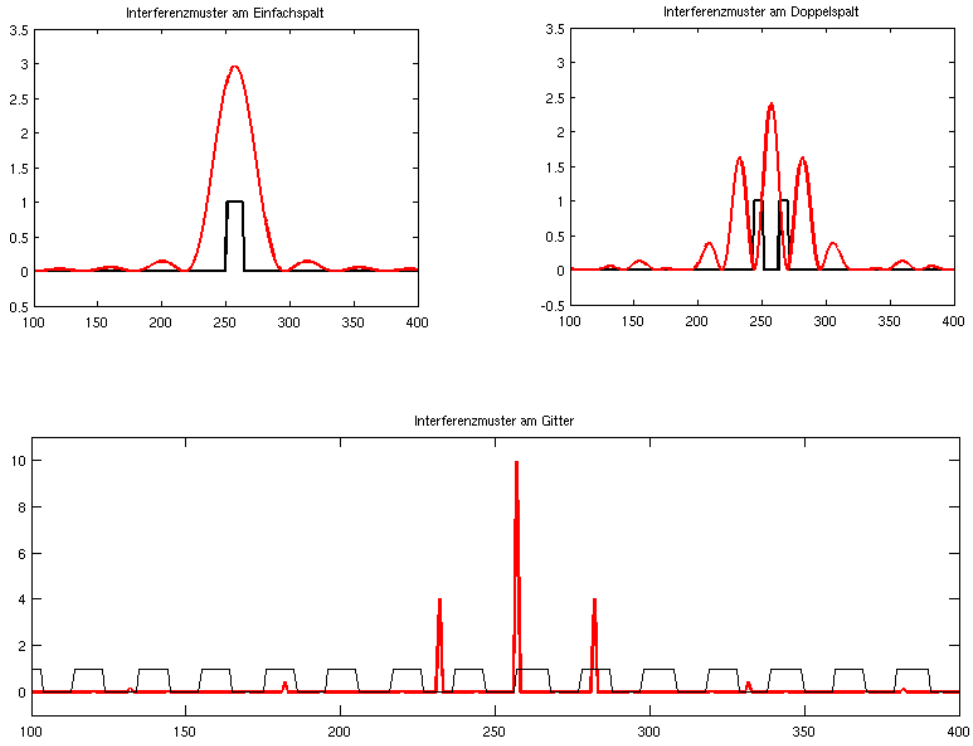

Abbildung 1: Intensitätsverteilungen, Daten erzeugt anhand von `fftw4.c`

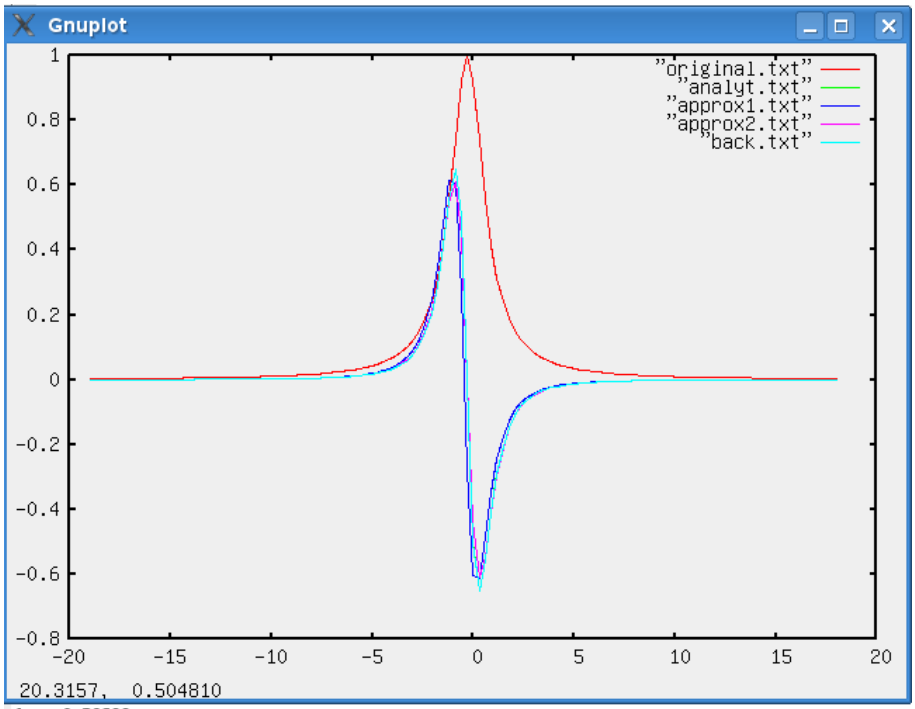

Abbildung 2: Vergleich verschiedener Methoden der numerischen Ableitung, Daten erzeugt anhand von `fftwd2.c`