NUMERISCHE METHODEN IN DER PHYSIK
Zweite Übung WS 2012/2013 [C]

Schnelle diskrete Fourier-Transformation
(Fast Fourier Transform FFT)

Darstellung der Methode: Skriptum Kap. 3.3 und 3.4.

Die folgenden Programme werden von mir per Internet zur Verfügung gestellt:

`four1.c` `randn.c`

Alles, was Sie über diese Routinen wissen müssen, finden Sie im VL-Skriptum, Abschnitt 3.3.2 bzw. auf den nächsten Seiten dieser Übungsbeschreibung.

Liste der Aufgaben im Rahmen dieser Übung:

1. Austesten der FFT-Routinen an Hand der Testdaten im VL-Skriptum S. 84.
   Dokumentation der Wirksamkeit der sog. "Frequenz-Optimierung" (VL-Skriptum, Abschnitt 3.4.3).
2. Weitere Tests zu den Themen "Frequenzanalyse", "Frequenzauflösung" und "Aliasing-Effekt" (s. VL-Skriptum, Abschnitt 3.3.4).
3. Entfaltung von glatten und verrauschten Meßwerten; Dirichlet-Glättung (s. VL-Skriptum, Abschnitte 3.4.1 und 3.4.2).

Es folgen nun einige Hinweise zur Verwendung des Programms *four1.c* im Zusammenhang mit dem Testbeispiel im Skriptum, S. 84 (bzw. Aufgabe 1):

```
/*  Programm FTTEST1   Diskrete Fourier-Transformation
    Testbeispiel Skriptum Num. Methoden in der Physik, S. 84

    H. Sormann    27-10-2000
    Last Update   24-10-2007
*/

#include <stdio.h>
#include <math.h>

#include "nrutil.c"
```

```
#include "four1.c"      // Einbeziehung der FFT-Routine

int main()
{
    int nn,nn2,isign, .... ;
    double ..... ;
    double *daten;          // (eindim. Feld)
    .
    .
    nn=8;                   // nn muss eine Potenz von 2 sein!
    nn2=2*nn;

    daten=dvector(1,nn2);   // Def. DOUBLE vector mittels NRUTIL:
                            // "daten" = eindim. DOUBLE-Vektor,
                            // Indexbereich = 1 .... nn2


// Testdaten von Skriptum, S. 84 (Abspeicherung s. S. 77):

// Beachten Sie bitte, dass die Indizierung des Feldes "daten"
// bei Eins beginnt:
    daten[1]=0.7013;        // Realteil  der ersten Zahl  (j=0)
    daten[2]=0.0437;        // Imag.teil der ersten Zahl  (j=0)
    .
    .
    daten[15]=0.2660;       // Realteil  der achten Zahl  (j=7)
    daten[16]=-0.3813;      // Imag.teil der achten Zahl  (j=7)
    .
    .
    isign=1;
    four1(daten,nn,isign);  // Aufruf der Routine FOUR1
                            // (Beschreibung der Parameter s. Skriptum,
                            // Kap. 3.3.2)
                            // Achtung:  daten  muss ein eindim. DOUBLE-
                            //                  Feld sein;
                            // nn und isign   muessen vom Typ "int" sein.
    .
    .
    free_dvector(daten,1,nn2);   // Schliessen des Feldes "daten"
    return 0;
}
```

Programm four1.c: Listing und Parameter s. Skriptum, S. 77ff.

```
#include <math.h>
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

void four1(double data[], int nn, int isign)
```

```
{
          // Beachten Sie die Uebereinstimmung des Aufrufbefehls im
          // main Programm mit dieser Headline:
          //      daten <======>  double data[]     (eindim. Feld)
          //      nn     ======>  int    nn         (Skalar)
          //      isign  ======>  int    isign      (Skalar)
  .
  .
  .
}
```

1. Aufgabe

Austesten von four1.c an Hand der Testdaten Skriptum S. 84.

- Wenden Sie das Programm *four1.c* auf die folgenden 8 komplexen Werte an ($n = 8$, $\Delta = 1.$):

j	RT{y(j)}	IT{y(j)}
0	0.7013	0.0437
1	-0.0724	0.5133
2	0.0988	-0.2688
3	0.0715	-0.1162
4	0.4013	0.1188
5	-0.0901	-0.1408
6	-0.1263	-0.0688
7	0.2660	-0.3813

  Die gegebenen Stützstellen repräsentieren somit eine periodische Funktion mit der Periode $P = 8$, und das Programm FOUR1 liefert die folgenden Fourierkoeffienten:

k	RT{Y(k)}	IT{Y(k)}
0	1.2501	-0.3001
1	0.0001	0.3000
2	0.2601	0.0001
3	-0.7000	-0.7003
4	0.9001	-0.0501
5	0.9999	0.0000
6	2.0001	1.0001
7	0.9000	0.0999

- Wenden Sie anschließend die Routine *four1.c* (mit ISIGN=–1) auf die erhaltenen Fourierkoeffizienten an und demonstrieren Sie, daß sich wieder die ursprünglichen Daten ergeben.
  Achtung: die inverse FT mittels *four1.c* vernachlässigt den Normierungsfaktor $1/n$.

- Berechnen Sie unter Verwendung dieser Fourierkoeffizienten die Interpolationsfunktion $I(x)$ gemäß Skriptum Glg. (3.15) für $0 \leq x \leq 7$ mit $\Delta x = 0.05$.
  Erstellen Sie zwei Diagramme (getrennt für Real- und Imaginärteil), welche die gegebenen Datenwerte plus die berechnete Interpolationsfunktion zeigen.

- Wie im Abschnitt 3.4.3 des VL-Skriptums ("Frequenz-Analyse") erläutert wird, ist die oben berechnete Interpolationsfunktion nicht die günstigste (glatteste) Verbindung der gegebenen Punkte. Variieren Sie Ihr Programm unter Verwendung der Gleichungen Skriptum (3.26–3.28), und erstellen Sie **zwei weitere Diagramme**, welche die gegebenen Daten und die "glattest-mögliche" Interpolationskurve darstellen (bitte wieder getrennte Diagramme für die Real- und Imaginärteile).

2. Aufgabe

Einige Tests zum Thema "Frequenzanalyse" (s. Skriptum, Abschnitt 3.4.3).

- Schreiben Sie ein Programm, mit Hilfe dessen Sie eine Frequenz-Analyse von n reellen Daten durchführen können, wobei Sie die Testdaten aus der folgenden Funktion berechnen:

```
y(t) = 2*cos[2 pi f1 t] - 3*sin[2 pi f2 t] - 1*cos[2 pi f3 t] +
       2*sin[2 pi f4 t] + 2.5*cos[2 pi f5 t]

fuer  t = t_{j} = j*delta   mit  delta = P/n    (j=0,...,n-1).

n = Anzahl der Stuetzpunkte     P = Periode in Sekunden

Die f1 ... f5  sind die Frequenzen der Testfunktion in Hertz.
```

  Starten Sie die Testrechnungen mit den folgenden Angaben:

```
n = 64      P = 1 s
Frequenzen: f1 = 2.0  f2 = 4.0  f3 = 4.0  f4 = 7.0  f5 = 12.0 Hz
```

  Verifizieren Sie, daß die mit Ihrem Programm durchgeführte Frequenzanalyse genau die Frequenzen und Intensitäten der Testfunktion ermittelt. Geben Sie die Ergebnisse auf dem Bildschirm in Form einer Tabelle (wie im Skriptum, S. 93) aus.

  Anmerkung: In Ihrer Tabelle gibt es viele Frequenzen mit "Null-Intensitäten". Unterdrücken Sie durch eine geeignete Abfrage die Ausgabe solcher Zeilen.

- In diesem Test sollen Sie untersuchen, wie Ihr Programm reagiert, wenn Sie einen der oben angegebenen Input-Parameter ändern, nämlich:

```
f4 = 7.5 Hz
```

  - Diskutieren Sie die Ergebnisse.
  - Welchen weiteren Input-Parameter müssen Sie (im Sinne von Skriptum, Kap. 3.4.3) ändern, um korrekte Ergebnisse zu erhalten?

- Für diesen Test arbeiten Sie mit den folgenden Parametern:

```
n = 64      P = 4 s
Frequenzen: f1 = 2.75  f2 = 4.0  f3 = 4.0  f4 = 7.0  f5 = 13.0 Hz
```

  - Liefert Ihre Frequenzanalyse die richtigen Ergebnisse?
    Wenn Nein, woran kann das liegen?
  - Was können Sie tun, um auch in diesem Fall ein korrektes Ergebnis zu erhalten?

- Alle bisherigen Testbeispiele dieser Aufgabe bestanden aus einer endlichen Zahl von Cosinus- bzw. Sinus-Termen. Eine Fourieranalyse ist in solchen Fällen kein Problem. Schwieriger wird die Sache, wenn die zu analysierende Testfunktion *Unstetigkeiten* im Funktionswert bzw. in einer Ableitung besitzt. Dieser Fall soll nun untersucht werden, und zwar an Hand der Testfunktion

```
P = 4 s        y(t) = 1                  fuer  0 <= t < 0.5   ,

               y(t) = 1 + 2(t-0.5)       fuer  0.5 <= t < 1.5  ,

               y(t) = 147/40 -3/10 t^2   fuer  1.5 <= t < 3.5  ,

               y(t) = 9 - 2 t            fuer  3.5 <= t < 4  .
```

  Führen Sie die Frequenzanalyse für die folgenden Stützpunktzahlen durch: $n = 256$, 512 und 1024. Dokumentieren Sie Ihre Ergebnisse in **einem** Diagramm der folgenden Art:
  Stellen Sie die *Absolutquadrate der Fourierkoeffizienten, dividiert durch die jeweiligen Werte von* n, in einem bzgl. der y-Achse halblogarithmischen Koordinatensystem dar. In Matlab steht Ihnen dafür die folgende Routine zur Verfügung:

```
semilogy(x-Vektor,y-Vektor,'...')
```

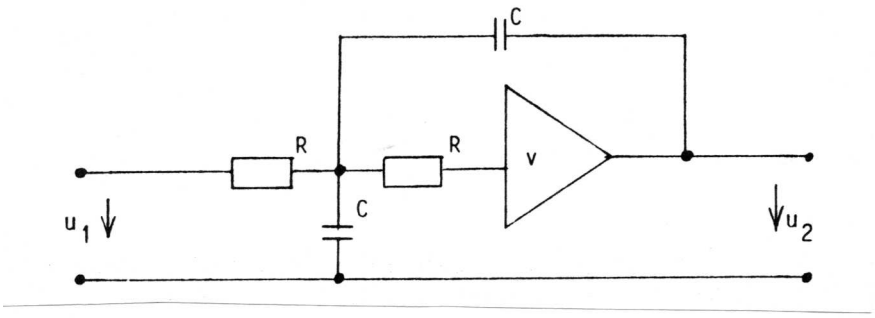
  Diskutieren Sie an Hand dieses Diagramms die Wirkung des *aliasing*-Effekts.

3. Aufgabe

Entfaltung von glatten und verrauschten Meßwerten. Dirichlet-Glättung

In dieser Aufgabe soll ein Problem untersucht werden, das bei der Auswertung experimenteller Daten eine Hauptrolle spielt. Es geht um die Filterung gemessener Werte von Einflüssen von zweierlei Art: (1) Einflüsse der Meßapparatur, (2) Einflüsse statistischer Meßfehler.

Beide Effekte werden im VL-Skriptum in Kürze beschrieben, und zwar in den Abschnitten 3.4.1 und 3.4.2.
Ausgangspunkt des Übungsbeispiels sind Messungen an einem sogenannten "aktiven Tiefpaß:



Quelle: P. Jacobs und S. Jancar, *BASIC: Interpolationen–Approximationen–Splines-FFT*, ed. H. Kohler, Vieweg-Verlag. $u_1 = s(t)$ $u_2 = m(t)$

Eine derartige Schaltung verändert das Eingangssignal $s(t)$ in das Ausgangssignal $m(t)$, wobei die Beziehung zwischen s und m durch das folgende Faltungsintegral gegeben ist:

$$m(t) = \int_{-\infty}^{+\infty} dt'\, r(t-t')\, s(t') \tag{1}$$

Die Funktion $r(t)$ stellt dabei die im Numerik-Skriptum als "Apparatefunktion" bezeichnete Größe dar. Im konkreten Fall zieht man für $r(t)$ die Bezeichnungen *Übertragungsfunktion* bzw. *Filterfunktion* vor.

Das Problem der folgenden Aufgabe besteht nun darin, aus der *gemessenen* Ausgangsfunktion $m(t)$ auf das unbekannte *Eingangssignal* $s(t)$ zurückzuschließen. Dies soll mit Hilfe der *diskreten Fast Fourier Transform* (DFFT) geschehen.

Dazu bedarf es natürlich der Kenntnis der Filterfunktion $r(t)$; eine Messung dieser Funktion kann z. B. so erfolgen, daß man dem System als Eingangssignal eine möglichst "delta-förmige" Funktion anbietet. Es folgt nämlich aus (1)

$$s(t) = \delta(t) \quad \rightarrow \quad m(t) = \int_{t'=-\infty}^{+\infty} dt'\, r(t-t')\, \delta(t') = r(t)\,.$$

Eine konkrete Filterfunktion $r(t)$ und ein konkretes Ausgangssignal ("Meßfunktion") $m(t)$ stehen in tabellarischer Form auf den Files

`filter_a.txt` und `messung6.txt`

zum Downladen bereit.

Wenn Sie die obige Filterfunktion Fourier-transformieren, sehen Sie sofort, daß dieser Filter vor allem die höheren Frequenzen unterdrückt; deshalb der in der Elektrotechnik bekannte Name "aktiver Tiefpaß".

Natürlich wird es auch hier das beste sein, das Faltungsintegral (1) aus dem Zeitraum in den Frequenzraum überzuführen. Tut man das, so ergibt sich die im Vorlesungsskriptum, Abschnitt 3.4.1, enthaltene Gleichung (3.22), deren Bedeutung darin liegt, daß man sie ohne Probleme nach der "ungestörten" Signalfunktion auflösen kann:

$$S_k = \frac{1}{\Delta} \frac{M_k}{R_k} . \tag{2}$$

Kennt man also die gemessene Funktion $m(t)$ und die Apparatefunktion $r(t)$ im realen Zeitraum, so kann man wie folgt vorgehen:

1. FT von $m(t)$ $\rightarrow$ M_k
2. FT von $r(t)$ $\rightarrow$ R_k
3. Berechnung von S_k aus Glg. (2)
4. Inverse FT von S_k $\rightarrow$ $s(t)$

Aufgabenstellung

- Es ist ein Programm zu entwickeln, mit Hilfe dessen man aus den tabellarisch gegebenen Meß- und Filterfunktionen *messung6.txt* und *filter_a.txt* die Eingangs- bzw. Signalfunktion $s(t)$ rekonstruieren kann. Es ist dafür die Methode der *Fast Fourier Transform* mit der C-Routine *four1.c* anzuwenden.

  Allgemeine Parameter: Anzahl n der Stützpunkte = 64
  zeitliche Periode $P = 0.00419$ s

- Stellen Sie die Meßfunktion und die rekonstruierte Signalfunktion im Zeit-Diagramm grafisch dar.

Bisher haben wir uns in der 3. Aufgabe dieser Übung ausschließlich mit einer Meßkurve $m(t)$ beschäftigt, welche keinerlei statistische Fehler enthielt. Solche Meßdaten ohne "Verrauschung" (*noise*) kommen natürlich in der experimentellen Praxis nicht vor. D. h., die Signal-Daten, mit denen man es i.a. zu tun hat, sind *in zweifacher Hinsicht* beeinträchtigt, nämlich

1. durch die Filterwirkung der Meßapparatur, und
2. durch die beim Experiment unvermeidlich auftretenden statistischen Fehler.

Im Folgenden geht es nun darum, den Einfluß der statistischen Fehler (des *noise*) auf die Rekonstruktion der Signalfunktion zu studieren und gegebenenfalls durch geeignete Methoden zu eliminieren.

Bei einem qualitätsvollen Experiment kann man davon ausgehen, daß die Beeinflussung der Meßwerte durch statistische Schwankungen relativ klein sein wird.

In solchen Fällen könnte man versucht sein, die (geringe) Verrauschung der Meßwerte bei der Rekonstruktion der Signalkurve einfach zu ignorieren.

Im Rahmen der folgenden Aufgabe sollen Sie untersuchen, ob eine solche Vorgangsweise vernünftig ist oder nicht!

Zu diesem Zweck soll die tabellarisch gegebene, glatte Meßfunktion $m(t)$ auf dem File *messung6.txt* künstlich verrauscht werden, und zwar durch Addition einer *noise*-Funktion $n(t)$:

$$\tilde{m}(t_j) = m(t_j) + n(t_j)\,, \tag{3}$$

wobei $n_j \equiv n(t_j)$ *normalverteilte Abweichungen mit einer vorgegebenen Standardabweichung* σ darstellen, d.h., die Werte n_j gehorchen der Verteilungsfunktion

$$p(n) = \frac{1}{\sigma\sqrt{2\pi}}\,\exp\left(-\frac{n^2}{2\sigma^2}\right)\,. \tag{4}$$

Wie kann nun eine normalverteilte Verrauschung programmiert werden?

Für C-UserInnen finden sich in verschiedenen Programmbibliotheken sicher Routinen, die ähnliches leisten wie das oben vorgestellte Matlab-Programm. Ich möchte hier aber die Gelegenheit nutzen, Ihnen einen einfachen, "selbstgestrickten" Algorithmus vorzustellen, mit dem man eine gegebene Punktmenge y_i mit einer simulierten Verrauschung n_i (Abweichungen normalverteilt mit Standardabweichung σ) versehen kann.

Ausgangspunkt ist die Gauss'sche Normalverteilung (4) als *differentielle Verteilungsfunktion*. Die daraus resultierende *integrale Verteilungsfunktion* lautet

$$P(n) \equiv \int_{-\infty}^{n} dx\, p(x) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{n}{\sigma\sqrt{2}}\right)\right]\,, \tag{5}$$

wobei die *error function* wie folgt definiert ist:

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z dt\, \mathrm{e}^{-t^2} \,.$$

Für die Berechnung der normalverteilten Abweichungen kann nun die sog. *Inversionsmethode* verwendet werden (s. jedes Statistik-Lehrbuch oder z.B. mein Skriptum *Computer-Simulation*), welche auf der Auswertung der Beziehung

$$P(n) = \kappa \tag{6}$$

beruht, wobei κ eine in $(0, 1)$ gleichverteilte Zufallszahl ist. Berechnet man die Lösung von Glg. (6) für die Abweichung n_i des i-ten y-Wertes mittels einer Newton-Iteration, so ergibt sich der folgende Algorithmus:

$$n_i^{(1)} = 0$$

$$n_i^{(j+1)} = n_i^{(j)} - \frac{a_j}{b_j} \qquad (j = 1, 2, \ldots)$$

mit

$$a_j = 1 + \mathrm{erf}\left(\frac{n_i^{(j)}}{\sigma\sqrt{2}}\right) - 2\kappa$$

und

$$b_j = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\frac{n_i^{(j)2}}{2\sigma^2}\right) \,.$$

Nach Ausführung dieser Iteration bis auf die gewünschte Genauigkeit ergibt sich der verrauschte y-Wert zu

$$\tilde{y}_i = y_i + n_i \,. \tag{7}$$

Ein C-Programm *randn.c*, welches auf diesen Formeln beruht, finden Sie auf der Website dieser Lehrveranstaltung.

Aufgabenstellung

Rekonstruieren Sie die Signalfunktion $s(t)$ unter Verwendung der als bekannt vorausgesetzten Filterfunktion $r(t)$ durch Entfaltung sowohl der *unverrauschten* Meßfunktion $m(t)$ als auch der *verrauschten* Meßfunktion $\tilde{m}(t)$.

- Verwenden Sie für diese Tests wieder die Datenfiles

```
messung6.txt      % Faltung einer Saegezahnfunktion mit
                  % einem Chebyshev-Filter

filter_a.txt      % Chebyshev-Filter
```

```
Fuer beide Datenfiles gilt:  64 Stuetzpunkte, Periode P = 0.00419 s
```

Für die numerische Simulation der normalverteilten Abweichungen verwenden Sie nacheinander die folgenden Werte:

```
sigma =  0.00001 / 0.0001 / 0.0005 / 0.001 / 0.005
```

- **Produzieren Sie für jedes sigma die beiden folgenden Diagramme:**
  - Diagramm 1: Vergleichen Sie die Messfunktionen $m(t)$ und $\tilde{m}(t)$.
  - Diagramm 2: Vergleichen Sie die entsprechenden Signalfunktionen $s(t)$ und $\tilde{s}(t)$.

Dirichlet-Dämpfung:

Die letzten Rechnungen haben Ihnen deutlich gezeigt, daß ein simples Ignorieren von statistischen Fehlern in den Meßdaten völlig unmöglich ist, da bereits sehr kleine Verrauschungen, die im Zeitraum kaum ins Gewicht fallen, die Rekonstruktion der Signalfunktion aus dem Frequenzraum heraus komplett verderben.

Es muss daher bereits im Frequenzraum versucht werden, den in der Meßfunktion enthaltenen *noise* effizient zu unterdrücken, d.h. eine *Glättung* der Meßdaten vorzunehmen. Eine sehr einfache Methode dazu finden Sie im Vorlesungsskriptum, Abschnitt 3.4.2, unter dem Namen "Dirichlet-Dämpfung".

Die Dirichlet-Methode geht von der Idee aus, daß die statistischen Schwankungen in den Werten von $m(t)$ im Fourierraum, also in der Funktion $M(k)$, sich hauptsächlich im mittleren k-Bereich bemerkbar machen, also in den Fourierkoeffizienten mit den Indizes

```
istart bis n-istart,
```

wohingegen die $M(k)$ für

```
0 bis istart-1    sowie  n-istart+1 bis n-1
```

im wesentlichen die glatte Messfunktion beschreiben. Eine wirksame Art und Weise, den *noise* zu unterdrücken, besteht also darin,

```
M(j) = 0   fuer  j=istart, ..., n-istart
```

zu setzen.

Im folgenden Test sollen Sie sich nun davon überzeugen, ob und in welchem Ausmaß diese Dirichlet'sche Glättung funktioniert.

Aufgabenstellung

- Verwenden Sie dazu das Datenmaterial

```
messung6_1.txt      % Faltung einer Saegezahnfunktion mit
                    % einem Chebyshev-Filter

filter_a_1.txt      % Chebyshev-Filter
```

`Fuer beide Datenfiles gilt:  256 Stuetzpunkte, Periode P = 0.00419 s`

- Unterwerfen Sie die gegebenen Meßwerte einer künstlichen Verrauschung in Form von normalverteilten Abweichungen mit $\sigma = 0.0005$.

- Machen Sie eine diskrete FT der verrauschten Meßdaten und behandeln Sie die erhaltenen Fourierkoeffizienten nach der Methode von Dirichlet.

- Verwenden Sie diese *Dirichlet-bereinigten* Fourierkoeffizienten für eine Rekonstruktion der Signalfunktion $s(t)$[1].

- Stellen Sie die folgenden Ergebnisse grafisch dar:

  (1) Rekonstruktion der glatten, d.h. von statistischen Fehlern freien Signalfunktion,
  (2) Rekonstruktion der fehlerbehafteten Signalfunktion ohne Glättung,
  (3) Rekonstruktion der fehlerbehafteten Signalfunktion nach erfolgter Dirichlet-Glättung.

- Ein wichtiger Punkt bei der Anwendung der Dirichlet-Methode ist natürlich die richtige Wahl des Index *istart*; experimentieren Sie etwas mit diesem Parameter:
  Wie verändert sich die Qualität der Rekonstruktion, wenn Sie *istart zu groß oder zu klein wählen?*
  Dokumentieren Sie diese Effekte durch Diagramme.

[1] Achtung: wenn die Dirichlet-Dämpfung korrekt ist, müssen alle rekonstruierten Werte der Signalfunktion reell sein. Das Auftreten von Imaginärteilen weist auf einen Fehler in Ihrem Programm hin!