Anwesenheitsübung 4 zur Vorlesung 'Numerische Methoden der Physik' SS 2014

Bastian Knippschild, Christian Jost und Mitarbeiter

Bearbeitung in den Übungen am 6. – 8. 05. 2014

Schnelle Fourier-Transformation

In dieser Übung wollen wir uns mit der Fourier-Transformation beschäftigen. Hierzu soll der Fast-Fourier-Transformation (FFT) Algorithmus implementiert werden und auf eine einfache Testfunktion angewendet werden.

In dieser Aufgabe werden komplexe Zahlen benutzt. C und C++ stellen komplexe Zahlen in einer Headerdatei zur Verfügung. Die Headerdatei heißt `complex.h` unter C und `complex` unter C++. Auf eCampus finden Sie Programme, die zeigen, wie die Headerdateien verwendet werden können.

Bei der schnellen Fourier-Transformation werden $n = 2^r$ Stützstellen w_i, $i = 1, \ldots, n$, und die Funktionswerte an den Stützstellen f_i, $i = 1, \ldots, n$ benötigt. In der Vorlesung wurde der folgende Pseudocode für die FFT angegeben:

```
m = n/2
K = 1
for i = 0, 1, ..., r-1; do
  for k = 0, 1, ..., K-1; do
    for j = 0, 1, ..., m-1; do
      a = 2 * k * m + j
      b = a + m
      f[a] = f[a] + f[b]
      f[b] = w[Kj] * ( f[a] - 2 * f[b])
    end for
  end for
  m = m/2
  K = 2*K
end for
```

Diese Funktion überschreibt die Funktionswerte an den Stützstellen mit den Fourier-Koeffizienten. Die Koeffizienten sind jedoch umsortiert. Überlegen Sie sich in diesem Zusammenhang warum in obigem Pseudocode die Wahl $a = 2{*}k{*}m{+}j$ notwendig ist. Um zu verstehen, wie die Umsortierung funktioniert, sollten Sie sich für z.B. $n = 4$ explizit überlegen, wie die durch den angegebenen Algorithmus verursachte Permutation aussieht. Hierbei ist es auch hilfreich die Permutation in einem Diagramm darzustellen.

Die Umsortierung kann durch folgenden Algorithmus rückgängig gemacht werden.

```
l=0
for i = 0, 1, ..., n-2; do
  k[i] = l
  m = n/2
  while m <= l; do
    l = l-m
    m = m/2
  end while
  l = l+m
end for
k[n=1] = n-1
```

In diesem Algorithmus werden nicht die Koeffizienten selbst sortiert, sondern eine Indexliste k_i erstellt. Dies bedeutet, dass g_{k_i} in der korrekten Reihenfolge ist. Es ist jedoch auch möglich, die Koeffizienten dirket zu sortieren. Für welche Möglichkeit Sie sich entscheiden, bleibt Ihnen überlassen.

Implementieren Sie die FFT. Dabei empfiehlt es sich die FFT in einer Funktion anzulegen, die als Argument ein Array mit den Funktionswerten übernimmt, so dass die Implementierung einfach für verschiedene Input-Funktionen bzw. Punktmengen wiederverwendet werden kann. Sie können dabei den Sortieralgorithmus entweder als eigene Funktion implementieren, oder direkt in die Funktion für die eigentliche FFT einbauen.

Testen Sie dann Ihren Algorithmus an der Funktion $f(x) = 8x + 1$. Mit $n = 8$ Stützstellen im Intervall $x \in [0, 1[$ sollten Sie die folgenden Werte erhalten:

Unsortiert:

$$\begin{aligned}
g[0] &= 12.727922 + i * 0.000000 \\
g[1] &= -1.414214 + i * 0.000000 \\
g[2] &= -1.414214 + i * 1.414214 \\
g[3] &= -1.414214 + i * -1.414214 \\
g[4] &= -1.414214 + i * 3.414214 \\
g[5] &= -1.414214 + i * -0.585786 \\
g[6] &= -1.414214 + i * 0.585786 \\
g[7] &= -1.414214 + i * -3.414214
\end{aligned}$$

Sortiert:

$$
\begin{aligned}
g[0 \to 0] &= 12.727922 + i * 0.000000 \\
g[1 \to 4] &= -1.414214 + i * 3.414214 \\
g[2 \to 2] &= -1.414214 + i * 1.414214 \\
g[3 \to 6] &= -1.414214 + i * 0.585786 \\
g[4 \to 1] &= -1.414214 + i * 0.000000 \\
g[5 \to 5] &= -1.414214 + i * -0.585786 \\
g[6 \to 3] &= -1.414214 + i * -1.414214 \\
g[7 \to 7] &= -1.414214 + i * -3.414214
\end{aligned}
$$

Anschließend können Sie ihren Algorithmus auf weitere Funktionen anwenden (z.B. $f(x) = \sin(ax)$ für verschiedene Werte von a und $f(x) = \exp(-0.5x^2)$). Überzeugen Sie sich, dass sich Ihr Resultat für hinreichend viele Stützstellen dem kontinuierlichen (analytisch bekannten) Ergebnis annähert.