Verwendung von Bibliotheksfunktionen

Wie in der Vorlesung besprochen, werden für die meisten numerisch anspruchsvollen Programmteile vorgefertigte Routinen verwendet, die in „Bibliotheken“ zusammengefasst sind (Sammlung oftmals verwendeter Routinen, z.B. alle Mathematikgrundfunktionen; optimierte Lösungen von Standardproblemen, z.B. Probleme der linearen Algebra, Anpassungsprobleme, ...). Sie bilden Archive, in denen die Programme in extrahierbarer Form abgespeichert sind (Inhalt, Zeitstempel, Zugriffsrechte, Eigentümer, ... sind gespeichert!) und vom Compiler mit dem Programm verknüpft werden. Dabei können sie „static“ (Binärcode wird beim Linken zum Hauptprogramm kopiert) oder „shared“ (Binärcode wird vor dem Ausführen zum Hauptprogramm kopiert) sein.

Anleitung zur statischen Verknüpfung (Beispiel für Fortan):

```
gfortran –Ldir program.F95 –lbibname
```

wobei die Option `–L` den Pfad zur Bibliothek angibt (nur notwendig, wenn die Bibliothek nicht an einem der Standard-Suchpfade gespeichert ist) und `–l` den Namen der Bibliothek angibt (lautet am Speicherort *libbibname.a*). Sollten Sie mehrere Bibliotheken verwenden kann es für eine korrekte Compilierung auf die Reihenfolge ankommen, in denen die Bibliotheken angeführt sind.

Beispiel Fouriertransformation mithilfe von FFTW

Die Bibliothek `fftw3` bietet numerisch optimierte Routinen zur Berechnung der Fouriertransformation in einer und mehr Dimensionen an. Im Unterschied zu vielen anderen derartigen Routinen, legt `fftw` einen „Plan“ an, wie die Transformation am effizientesten durchgeführt werden kann, danach folgt der eigentlich Aufruf der Routine.

In Fortran95 enthält ein Programm, das ein Array der Länge *N* transformieren soll, folgende Bestandteile (Anlegen des Plans, Durchführung der Transformation, evtl. Löschen des Plans):

```
include ‘fftw3.f’   ! enthält für die Ausführung notwendige Definitionen
double complex, dimension(N) :: in, out
integer*8 plan
…
call dfftw_plan_dft_1d(plan,N,in,out,FFTW_FORWARD,FFTW_ESTIMATE)
call dfftw_execute_dft(plan, in, out)
call dfftw_destroy_plan(plan)
…
```

In C sieht das Programm folgendermaßen aus:

```
#include <fftw.h>
fftw_complex in[N], out[N];
fftw_plan plan;
…
plan = fftw_plan_dft_1d(N,in,out,FFTW_FORWARD,FFTW_ESTIMATE);
fftw_execute(plan);
fftw_destroy_plan(plan);
…
```

Auch `fftw` bietet in-place Transformations-Routinen (Output überschreibt Input) an – die umfangreiche Dokumentation der Bibliothek finden Sie unter http://www.fftw.org/fftw3_doc/.

Warnung! Lesen Sie sich die Beschreibung von FFT-Bibiliotheksroutinen immer sehr genau durch. Insbesondere für real→complex-Transformationen kann die Speicherung des Ergebnisses zwar sehr effizient aber nicht leicht nachvollziehbar sein. Durch die Verwendung komplexer Arrays im Beispiel oben ist auch das Ergebnis `out` komplexwertig, aber wie sind die Elemente angeordnet?