FFTW - Fastest Fourier Transform in the West

FFTW is a comprehensive collection of fast C routines for computing the discrete Fourier transform (DFT) in one or more dimensions, of both real and complex data, and of arbitrary input size. FFTW also includes parallel transforms for distributed-memory systems. FFTW is usually faster (and sometimes much faster) than all other freely-available Fourier transform programs found on the Net.

Introductory Remarks

Authors of the Library

The FFTW package was developed at MIT by Matteo Frigo and Steven G. Johnson.

Using the FFTW installations at LRZ

FFTW is provided on all LRZ HPC platforms. To prepare for using the library, first load the appropriate environment module, for example

```bash
module load fftw
```

for the default version. An overview of available modules is provided in the following table.

<table>
<thead>
<tr>
<th>Module</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial or threaded processing</td>
<td>default version, provides all precisions and threading (Posix Threads as well as OpenMP)</td>
</tr>
<tr>
<td>fftw/serial/3.3</td>
<td>MKL based interface (other versions than 11.3 may be available). Whether this runs serially or OpenMP-parallel depends on the specific MKL module that was previously loaded. See the section on &quot;FFTW interface to MKL&quot; below</td>
</tr>
<tr>
<td>fftw/mpi/3.3</td>
<td>New FFTW MPI interface</td>
</tr>
<tr>
<td>fftw/mpi/v3_mkl113</td>
<td>MKL based parallel processing interface (other versions than 11.3 may be available)</td>
</tr>
</tbody>
</table>

After loading the appropriate environment module, you can use several environment variables to compile and link your application. For example, you may compile your code with

```bash
icc -c $FFTW_INC foo.c
```

and link it with

```bash
icc -o myprog.exe ... foo.o $FFTW_LIB
```

Here is the complete list of environment variables provided for linking against a version of FFTW. Further details are provided in the next section.

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FFTW_LIB</td>
<td>For static linking.</td>
</tr>
<tr>
<td>$FFTW_SHLIB</td>
<td>For dynamic linking.</td>
</tr>
</tbody>
</table>
Further Details and Special Cases

**Supported compiler**

The Intel compilers must be used to compile in the FFTW libraries.

**Varying Precisions**

Floating point data types of various size (4, 8 and 10 Bytes) are automatically supported in the 3.x releases.

*Note:* For the 2.x releases, separate modules were necessary for single and double precision and hence the two precisions could not be simultaneously used in the same program. Version 2.x is not supported any more on LRZ systems.

**Architecture**

Depending on the processor architecture automatically diagnosed by the module, the libraries compiled with appropriate vectorization settings will be used.

**Thread safeness**

Some FFTW calls, notably the plan generators, must be called outside any threaded regions.

**Shared libraries**

By using e.g. $FFTW_SHLIB in place of $FFTW_LIB you can link against the installed FFTW shared libraries. However, if you take your compiled binary off the system, you will need to provide the required shared library objects as well.

**MPI parallel transforms**

An MPI interface is offered for version 3.3 of FFTW. These interfaces are incompatible. The FFTW_LIB variable contains all libraries necessary to link a non-threaded MPI program:

```bash
mpicc -o my_mpi_prog.exe main.o a.o b.o c.o $FFTW_LIB
```

If you want to link a threaded library (version 3.3 and higher only), use the following link flags e.g., for generating a hybrid MPI + OpenMP executable:

```bash
mpicc -qopenmp -o my_mpi_prog.exe main.o a.o b.o c.o $FFTW_MPI_LIB $FFTW_OPENMP_LIB
```

**New Fortran interface**

Version 3.3 of FFTW supports a new Fortran interface that is based on the C interoperability features of Fortran 2003 and later. In order to use this interface, you can for example write a small auxiliary module

```fortran
module fftw
    use, intrinsic :: iso_c_binding
    include 'fftw3.f90'
end module
```

which allows you to access the explicit interface for double precision (8 byte real) processing. This allows the compiler to check usage against definition.

**FFTW interface to MKL**

Newer releases of the Intel MKL contain a partial implementation of the FFTW interface: Please load one of the environment modules from the above table with the last component $v3_mkl<version>$ to access the corresponding library using the same environment variables as described above. For these module, only the FFTW_LIB variable will be set, not the FFTW_OPENMP_LIB or FFTW_PTHREADS_LIB. Whether purely serial or shared memory mode is used depends on the previously loaded MKL module, and possibly the setting of OMP_NUM_THREADS.

**Example for compiling, linking and running a threaded code**

```bash
module load mkl/2017 fftw/serial/v3_mkl2017  # mkl/11.3 is threaded.

icc -c $FFTW_INC my_prog.c

icc -o my_prog.exe my_prog.c $FFTW_LIB
```
export OMP_NUM_THREADS=4
./my_prog.exe

Example for compiling, linking and running MPI code (with MPI-parallel FFTW calls)

module load mpi.intel mkl/2017_s fftw/mpi/v3_mkl2017
mpicc -c $FFTW_INC my_prog.c
mpicc -o my_prog.exe my_prog.c $FFTW_LIB
mpiexec -n 4 ./my_prog.exe

Note that it is a good idea to check for a returned NULL pointer after calling a fftw_plan setup routine, otherwise calls to unsupported parts of the FFTW interface will simply crash your code. In particular, using types other than 8 Byte floats (or 16 Byte complex) will not work. Please check the MKL Release Notes for details on which calls are unsupported.

The advantage of using the MKL is that you may obtain improved performance in some cases, even compared with standard FFTW.

Documentation

Please refer to the FFTW home page for documentation

- FFTW Home Page, especially the documentation section
- Frequently Asked Questions on the FFTW site