HP-SEE
FFTs Using FFTW and FFTE Libraries

www.hp-see.eu

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

Aleksandar Jović

Institute of Physics Belgrade, Serbia
Scientific Computing Laboratory
ajovic@ipb.ac.rs

The HP-SEE initiative is co-funded by the European Commission under the FP7 Research Infrastructures contract no. 261499

Overview

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Introduction
- FFT algorithms and FFT libraries
- FFTE library
  - Overview of mostly used subroutines
  - tests, compiling and running (serial, OPENMP, MPI, MPI+OPENMP)
- FFTW library
  - Overview of mostly used subroutines
  - tests, compiling and running (serial, OPENMP, MPI, MPI+OPENMP)

Introduction

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The **Discrete Fourier Transform** (**DFT**) plays an important role in many scientific and technical applications, including time series and waveform analysis, solutions to linear partial differential equations, convolution, digital signal processing, and image filtering.

DFT :

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad 0 \le k \le N-1$$

- Where

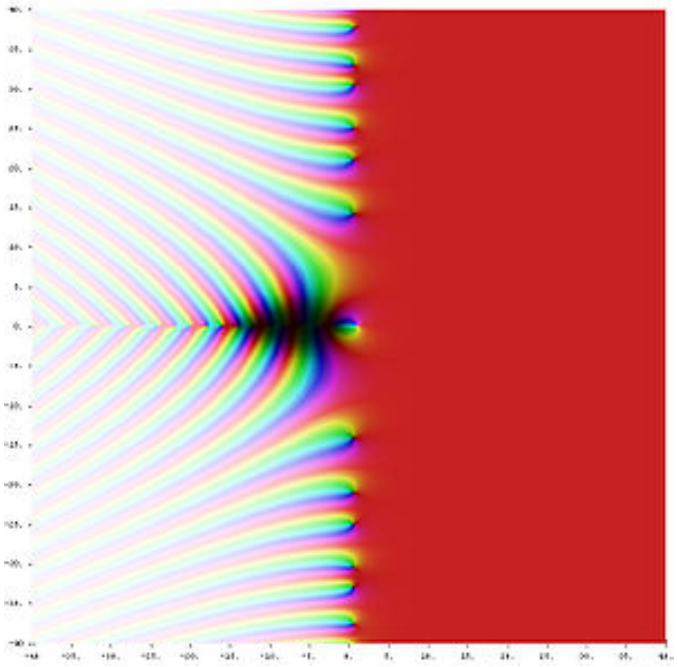
$$W_N = e^{-j2\pi/N} = \cos(2\pi/N) - j\sin(2\pi/N)$$

- IDFT:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk} \qquad \text{n} = 0, 1 \ldots\ldots \text{N-1}$$

- A fast Fourier transform (**FFT**) is an efficient algorithm to compute DFT and its inverse

FFT



- FFT algorithms
  - Cooley–Tukey algorithm
  - Prime-factor FFT algorithm (PFA)
  - Bruun's FFT algorithm
  - Rader's FFT algorithm
  - Bluestein's FFT algorithm
  - Odlyzko–Schönhage algorithm
- FFT libraries
  - **FFTW** http://www.fftw.org/
  - P3DFFT
  - FFTPACK
  - ACML
  - GSL
  - ESSL
  - .....
  - **FFTE** http://www.fft e.jp/

FFTE library

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- **FFTE** ("Fastest Fourier Transform in the East",) has been developed by Daisuke Takahashi of Tsukuba, Center for Computational Sciences, Graduate School of Systems and Information Engineering University of Tsukuba, Japan.
- The name FFTE, which is an acronym for "Fastest Fourier Transform in the East", is more of a tribute to FFTW than a signal of any serious attempt to offer a production-ready library to rival FFTW (even though FFTE has been observed to slightly outperform FFTW on very large FFTs).
- FFTE is a **FORTRAN** (77 and 90) subroutine library for computing the FFT in one or more dimensions.
- FFTE supports radix 2, 3, and 5 Discrete Fourier Transforms (DFTs), including optimized routines for radix-8, and has pure MPI and hybrid MPI+OpenMP.
- More about these algorithms can be found in the *FAST FOURIER TRANSFORM ALGORITHMS WITH APPLICATION*
- FFTE is open source, but FFTE **comes with little documentation**, and it is necessary to examine the source code in order to use it !!!!!!!

FFTE library

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- http://www.fftw.jp/

- Extract file with `tar xvzf ffte-5.0.tgz`
- `[ajovic@int1 ~]$ cd ffte-5.0`
- Files :
  - `zfft1d.f, zfft2d.f, zfft3d.f, zdfft2d.f, zdfft3d.f, fft235.f , kernel.f , mfft235.f , vzfft1d.f , vzfft2d.f, vzfft3d.f, dzfft2d.f , dzfft3d.f , param.h, readme.txt`
  - `tests/` (Test directory, serial and OPENMP)
  - `mpi/` (MPI version directory)
    - `pzfft1d.f, pzfft2d.f, pzfft3d.f , pdzfft2d.f, pdzfft3d.f, pvzfft1d.f, pvzfft2d.f, pvzfft3d.f, pzdfft2d.f, pzdfft3d.f, pzfft3dv.f`
    - `tests/` (MPI, MPI+OPENMP version test directory)

ZFFT1D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `ZFFT1D(A,N,IOPT,B)` is 1D-COMPLEX FFT routine
  - FORTRAN 77 + OPENMP
  - A(N) is comlpex input/output vector (COMPLEX*16)
  - B(N*2) is work/coefficient vector (COMPLEX*16)
  - $N=2^{IP} *3^{*IQ} *5^{*IR}$ is the length of the transforms
  - IOPT= 0 , for initialization the coefficients
    -1 , for FORWARD transform
    1 , for INVERSE transform
  - IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `zfft1d.f`

ZFFT2D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `ZFFT2D(A,NX,NY,IOPT)` is 2D-COMPLEX FFT routine
  - FORTRAN 77 + OPENMP
  - A(NX*NY) is comlpex input/output vector (COMPLEX*16)
  - NX=2^{IP} *3^{*IQ} *5^{*IR} is the length of the transforms in the X-direction
  - NY=2^{JP} *3^{*JQ} *5^{*JR} is the length of the transforms in the Y-direction

  IOPT= 0 , for initialization the coefficients
  - IOPT= -1 , for FORWARD transform
    1 , for INVERSE transform

  - IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `zfft2d.f`

ZFFT3D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `ZFFT3D(A,NX,NY,NZ,IOPT)` is 3D-COMPLEX FFT routine
  - FORTRAN 77 + OPENMP
  - A(NX*NY*NZ) is comlpex input/output vector (COMPLEX*16)
  - NX=2^{IP} *3^{*IQ} *5^{*IR} is the length of the transforms in the X-direction
  - NY=2^{JP} *3^{*JQ} *5^{*JR} is the length of the transforms in the Y-direction
  - NZ=2^{KP} *3^{*KQ} *5^{*KR} is the length of the transforms in the Z-direction

  - IOPT= 0 , for initialization the coefficients
    -1 , for FORWARD transform
    1 , for INVERSE transform

  - IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `zfft3d.f`

DZFFT2D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `DZFFT2D(A,NX,NY,IOPT,B)`
  - 2D-REAL-TO-COMPLEX FFT routine
  - FORTRAN 77 + OPENMP
  - A(NX,NY) is real input vector (REAL*8)
  - A(NX/2+1,NY) is complex output vector (COMPLEX*16)
  - B(NX/2+1,NY) is work/coefficient vector (COMPLEX*16)
  - NX=2^{IP} *3^{*IQ} *5^{*IR} is the length of the transforms in the X-direction
  - NY=2^{JP} *3^{*JQ} *5^{*JR} is the length of the transforms in the Y-direction

    0 , for initialization the coefficients
  - IOPT= -1 , for FORWARD transform

  - IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `dzfft2d.f`

ZDFFT2D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `ZDFFT2D(A,NX,NY,IOPT,B)`
  - 2D-COMPLEX-TO-REAL FFT routine
  - FORTRAN 77 + OPENMP
  - A(NX/2+1,NY) is complex input vector (COMPLEX*16)
  - A(NX,NY) is real output vector (REAL*8)
  - B(NX/2+1,NY) is work/coefficient vector (COMPLEX*16)
  - NX=2^{IP} *3^{*IQ} *5^{*IR} is the length of the transforms in the X-direction
  - NY=2^{JP} *3^{*JQ} *5^{*JR} is the length of the transforms in the Y-direction

    0 , for initialization the coefficients
  - IOPT= 1 , for BACKWARD transform

  - IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `zdfft2d.f`

DZFFT3D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `DZFFT3D(A,NX,NY,NY,IOPT,B)`
  - 3D-REAL-TO-COMPLEX FFT routine
  - FORTRAN 77 + OPENMP
  - A(NX,NY,NZ) is real input vector (REAL*8)
  - A(NX/2+1,NY,NZ) is complex output vector (COMPLEX*16)
  - B(NX/2+1,NY,NZ) is work/coefficient vector (COMPLEX*16)
  - $NX=2^{IP} *3^{*IQ} *5^{*IR}$ is the length of the transforms in the X-direction
  - $NY=2^{JP} *3^{*JQ} *5^{*JR}$ is the length of the transforms in the Y-direction
  - $NY=2^{KP} *3^{*KQ} *5^{*KR}$ is the length of the transforms in the Z-direction

    0 , for initialization the coefficients
  - IOPT= -1 , for FORWARD transform
  - IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `dzfft3d.f`

ZDFFT3D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `ZDFFT3D(A,NX,NY,NZ,IOPT,B)`
  - 3D-COMPLEX-TO-REAL FFT routine
  - FORTRAN 77 + OPENMP
  - A(NX/2+1,NY,NZ) is complex input vector (COMPLEX*16)
  - A(NX,NY,NZ) is real output vector (REAL*8)
  - B(NX/2+1,NY,NZ) is work/coefficient vector (COMPLEX*16)
  - NX=2^{IP} *3^{*IQ} *5^{*IR} is the length of the transforms in the X-direction
  - NY=2^{JP} *3^{*JQ} *5^{*JR} is the length of the transforms in the Y-direction
  - NZ=2^{KP} *3^{*KQ} *5^{*KR} is the length of the transforms in the Z-direction

    0 , for initialization the coefficients
  - IOPT= 1 , for BACKWARD transform
  - IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `zdfft3d.f`

EXAMPLES

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- EXAMPLES:
  - `test1d.f` `zfft1d.f` (test program)
  - `test2d.f` `zfft2d.f` (test program)
  - `test3d.f` `zfft3d.f` (test program)
  - `speed3d.f` `zfft3d.f` (speed test program)
  - `rtest3d.f` test (`zdfft2d.f and dzfft2d.f`)program

- **CALL ZFFT1D(A,N,0,B) ! This call is needed for initialization FFTE**
- **Compiling and Running** :
  - `$ gfortran -O3 -fomit-frame-pointer -fopenmp test1d.f zfft1d.f kernel.f fft235.f`
  - `$ ifort -O3 -openmp test1d.f zfft1d.f kernel.f fft235.f`

Compiling and running FFTE

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- **Compiling and Running :**
  - Serial job :
    - `$ gfortran test1d.f zfft1d.f kernel.f fft235.f -o test1d`
    - `ifort , f95`
    - `$./test1d`
  - OPENMP job :
    - `$ gfortran -O3 -fomit-frame-pointer -fopenmp test1d.f zfft1d.f kernel.f fft235.f -o test1d_omp`
    - `$ ifort -O3 -xHost -openmp test1d.f zfft1d.f kernel.f fft235.f -o test1d_omp`
    - `OMP_NUM_THREADS=<NUMBER OF THREADS> ./test1d_omp`

mpi/

HP-SEE
High Performance Computing Infrastructure for South East Europe's Research Communities

- `PZFFT1D(A,B,W,N,ICOMM,ME,NPU,IOPT)` is 1D-COMPLEX FFT routine
  - FORTRAN 90 + MPI SOURCE PROGRAM
  - W(N/NPU) is coefficient vector (COMPLEX*16)
  - $N=2^{IP} *3^{*IQ} *5^{*IR}$ is the length of the transform
  - ICOMM is the communicator (INTEGER*4)
  - ME is the rank
  - NPU is the number of processors
  - A(N/NPU) is complex input vector (COMPLEX*16)
  - B(N/NPU) is complex output vector(COMPLEX*16)
  - IOPT=0, for initialization the coefficients
    =-1, for FORWARD transform
    = 1, for BACKWARD transform

IMPORTANT : Subroutines `fft235.f`, `kernel.f` , `zfft1d.f` and header file `param.h` are needed to use `pzfft1d.f`

PZFFT2D

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `PZFFT2D(A,B,NX,NY,ICOMM,NPU,IOPT)` is 2D-COMPLEX FFT routine
  - FORTRAN 77 + MPI SOURCE PROGRAM
  - NX=2^{IP} *3^{*IQ} *5^{*IR} is the length of the transforms in the X-direction
  - NY=2^{JP} *3^{*JQ} *5^{*JR} is the length of the transforms in the Y-direction
  - ICOMM is the communicator (INTEGER*4)
  - NPU is the number of processors
  - A(NX,NY/NPU) is complex input vector (COMPLEX*16)
  - B(NX,NY/NPU) is complex output vector(COMPLEX*16)

  - IOPT = 0 FOR INITIALIZING THE COEFFICIENTS (INTEGER*4)
    =-1 FOR FORWARD TRANSFORM
    =1 FOR BACKWARD TRANSFORM

IMPORTANT : Subroutines `fft235.f`, `kernel.f` and header file `param.h` are needed to use `pzfft2d.f`

PZFFT3D

HP-SEE
High Performance Computing Infrastructure for South East Europe's Research Communities

- `PZFFT3D(A,B,NX,NY,NZ,ICOMM,NPU,IOPT)` - 3D-COMPLEX
  - FORTRAN 77 + MPI SOURCE PROGRAM
  - $NX=2^{IP} *3^{*IQ} *5^{*IR}$ is the length of the transforms in the X-direction
  - $NY=2^{JP} *3^{*JQ} *5^{*JR}$ is the length of the transforms in the Y-direction
  - $NZ=2^{KP} *3^{*KQ} *5^{*KR}$ is the length of the transforms in the Y-direction
  - ICOMM is the communicator (INTEGER*4)
  - NPU is the number of processors
  - A(NX,NY,NZ/NPU) is complex input vector (COMPLEX*16)
  - B(NX,NY,NZ/NPU) is complex output vector(COMPLEX*16)
  - IOPT = 0 FOR INITIALIZING THE COEFFICIENTS (INTEGER*4)
    =-1 FOR FORWARD TRANSFORM
    =1 FOR BACKWARD TRANSFORM
- IMPORTANT : Subroutines `fft235.f, kernel.f` and header file `param.h` are needed to use `pzfft3d.f`

Examples, compiling and running

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- example_mpi_3D.f
  - MPI job :
    - $ mpif90 example_mpi_3D.f pzfft3d.f kernel.f fft235.f -o mpi3d
    - $ mpiexec -np 4 ./mpi3d
  - hybrid job :
    - $ mpif90 -openmp example_mpi_3D.f pzfft3d.f kernel.f fft235.f -o mpi3d
    - $ OMP_NUM_THREADS=2 mpiexec -np 2 ./mpi3d

FFTW library

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The Fastest Fourier Transform in the West (FFTW), is a software library for computing discrete Fourier transforms (DFTs), developed by Matteo Frigo and Steven G. Johnson at the Massachusetts Institute of Technology
- FFTW is written in the C language
- It can compute transforms of real and complex-valued arrays of arbitrary size and dimension in *O(n log n)* time.
-
- It works best on arrays of sizes with small prime factors, with powers of 2 being optimal and large primes being worst case
- for prime sizes it uses either *Rader's* or *Bluestein's* FFT algorithm
- In 1999, FFTW won the J. H. Wilkinson Prize for Numerical Software

Installation on Linux

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- **http://www.fftw.org/**
- Extract file with command `tar xvzf  fftw-3.3.2.tar.gz`
- `$ cd fftw-3.3.2`
- `$ ./configure --enable-openmp --enable-mpi --prefix=/home/ajovic/QE/ name_directory`
- `$ make`
- `$ make install`
- The configure script chooses the *gcc* compiler by default, if it is available; you can select some other compiler with: `./configure CC="<the name of your C compiler>"`
- `--enable-mpi`: Enables compilation and installation of the FFTW MPI library
- `--enable-openmp`: Enables compilation and installation of the FFTW OPENMP library

COMPLEX 1D FFTW

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- `#include <fftw3.h> ...`
- `fftw_complex *in, *out; fftw_plan p; . . . . .`
- ```
  in = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
  out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
  p =fftw_plan_dft_1d(N, in, out, FFTW_FORWARD, FFTW_ESTIMATE);
  ```
- `... fftw_execute(p); /* repeat as needed */`
- `... fftw_destroy_plan(p);`
- `fftw_free(in); fftw_free(out); }`
- This function creates the plan:
  - `fftw_plan_dft_1d(int n, fftw_complex *in, fftw_complex *out, int sign, unsigned flags);`
  - n is the size of the transform
  - arguments in and out are pointers to the input and output arrays of the transform
  - `sign`, can be either `FFTW_FORWARD` (-1) or `FFTW_BACKWARD` (+1)
  ```

COMPLEX 1D FFTW

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The flags : `FFTW_MEASURE` or `FFTW_ESTIMATE`
  - `FFTW_MEASURE` instructs FFTW to run and measure the execution time of several FFTs in order to find the best way to compute the transform of size n. This process takes some time (usually a few seconds), depending on your machine and on the size of the transform
  - `FFTW_ESTIMATE`, on the contrary, does not run any computation and just builds a reasonable plan that is probably sub-optimal
- Computing the actual transforms via `fftw_execute(plan)` :
  - `void fftw_execute(const fftw_plan plan);`
- If you want to transform a *different* array of the same size, you can create a new plan with `fftw_plan_dft_1d` and FFTW automatically reuses the information from the previous plan, if possible.

COMPLEX 2D and 3D FFTW

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 2 and 3-dimensional transforms work much the same way as 1-dimensional transforms
- The 2d and 3d routines have the following signature:
  - `fftw_plan_dft_2d(int n1,int n2, fftw_complex *in, fftw_complex *out, int sign, unsigned flags);`
  - `fftw_plan_dft_3d(int n1,int n2, int n3, fftw_complex *in, fftw_complex *out, int sign, unsigned flags);`
- 1D FFTW of real to comlpex and complex to real data :
  - `fftw_plan_dft_r2c_1d(int n1,int n2, int n3, double *in, fftw_complex *out, int sign, unsigned flags);`
  - `fftw_plan_dft_c2r_1d(int n1,int n2, int n3, fftw_complex *in, double *out, int sign, unsigned flags);`

EXAMPLE 3D FFTW

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Example : `fftw-ser.c`
- You must link this code with the fftw3 library. On Unix systems, link with `-lfftw3 -lm`
- `$ icc fftw-ser.c -L/home/ajovic/QE/fftw-3.3.2/lib -I/home/ajovic/QE/fftw-3.3.2/include -lm -lfftw3 -o serijski`
- `./serijski`

Usage of OPENMP in FFTW

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- before calling any FFTW routines, you should call the function:
  - `int fftw_init_threads(void);`
- before creating a plan that you want to parallelize, you should call:
  - `void fftw_plan_with_nthreads(int nthreads);`
  - The `nthreads` argument indicates the number of threads you want FFTW to use
- If you want to get rid of all memory and other resources allocated internally by FFTW, you can call:
  - `void fftw_cleanup_threads(void);`
- Programs using the parallel complex transforms should be linked with `-lfftw3_omp -lfftw3 -lm` if you compiled with OpenMP

EXAMPLE with OPENMP

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- fftw-omp.c
- $ icc fftw-ser.c -L/home/ajovic/QE/fftw-3.3.2/lib -I/home/ajovic/QE/fftw-3.3.2/include -lm -openmp -lfftw3_omp -lfftw3 -o tredovan
- $ gcc fftw-ser.c -L/home/ajovic/QE/fftw-3.3.2/lib -I/home/ajovic/QE/fftw-3.3.2/include -lm -fopenmp -lfftw3_omp -lfftw3 -o tredovan

MPI FFTW

HP-SEE

- All programs using FFTW's MPI support should include its header file `#include <fftw3-mpi.h>`
- We must call `fftw_mpi_init()` after calling `MPI_Init()`
- when we create the plan with f`ftw_mpi_plan_dft_3d`, analogous to `fftw_plan_dft_3d`, we pass an additional argument: the *communicator*, indicating which processes will participate in the transform (hereMPI_COMM_WORLD, indicating all processes)

References

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- http://www.fftw.org/
- http://www.fft e.jp/