Update of spherical Bessel transform: FFTW and OpenMP

Peter Koval a, J.D. Talman b,*

a Centro de Física de Materialies, Paseo Manuel de Lardizábal 5, San Sebastián, Spain
b Department of Applied Mathematics, University of Western Ontario, London, Ontario, Canada N6A 5B7

ABSTRACT

We present an update of spherical Bessel transform subroutines. The update makes use of the widely available FFTW library for performing necessary fast Fourier transforms. Moreover, the update is prepared for shared memory OpenMP parallelization that is popular on multi-core processors. Furthermore, the updated version uses the module concept to achieve a better control over the memory allocation/deallocation, which also contributes to the performance of the updated package.

New version program summary

Program title: SBT, version number 3
Catalogue identifier: AANZ_v3_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AANZ_v3_0.html
Program obtainable from: CPC Program Library, Queen’s University, Belfast, N. Ireland
No. of lines in distributed program, including test data, etc.: 683
No. of bytes in distributed program, including test data, etc.: 6903
Distribution format: tar.gz
Programming language: Fortran 90
Computer: Any computer with a conforming Fortran 90 compiler
Operating system: Any system with a conforming Fortran 90 compiler
Has the code been vectorised or parallelized?: OpenMP is used in the subroutine
Classification: 4.6
External routines: FFTW3 (http://www.fftw.org/)
Does the new version supersede the previous version?: No
Catalogue identifier of previous version: AANZ_v2_0
Nature of problem: The Fourier transform of a spherically symmetric angular momentum eigenfunction leads to the spherical Bessel transform of the corresponding radial function. Radial functions are often given numerically, on a radial grid. Therefore, the direct computation of the spherical Bessel transform would require of order \(N\) arithmetical operations for each value of the transform variable, where \(N\) is the number of points on the radial grid. At large values of the transform variable, the rapid oscillation of the integrand makes usual methods of integration impractical. However, if the radial and transform variables are defined on logarithmic grids, this problem is overcome [1]. Moreover, the approach requires two applications of a fast Fourier transform so that the operational count can be lowered to \(N \log N\) arithmetical operations.
Solution method: The program applies a procedure that treats the problem as a convolution [1]. The calculation then requires two applications of the fast Fourier transform method.
Reasons for new version: The previous versions of the program use a built-in FFT routine, which can only treat array sizes of \(2^n\). Employing the widely available FFTW package [2] allows greater flexibility in the grid dimensions, which can be any multiple of small primes, e.g. 80, 81, 100, etc. Moreover, the FFTW package is much more efficient than the previous FFT routine and also can be used in shared-memory parallelized programs. A number of changes have also been made to make the subroutine thread-safe. We use the OpenMP standard to run the subroutine within OpenMP-parallelized loops.

* Corresponding author.
E-mail address: jtalman@uwo.ca (J.D. Talman).

© 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2010.08.024
Summary of revisions:

1. FFTW library is used; FFT transforms for real-valued arrays are used where ever appropriate.
2. The subroutine is initialized by a call to sbt_plan(nr, lmax, rr, kk) where rr and kk are the arrays giving the radial grid points in coordinate and momentum space, correspondingly. The integer nr specifies the number of points in radial grids rr and kk. The integer lmax is the maximum angular momentum l value.
3. The subroutine is executed by a call to sbt_execute(ff, gg, l, d) where ff is the input array, gg is the output array, the integer l is the order of the transform (angular momentum) and the integer d controls the direction of the transform. If d=+1, then the input array ff must be given in coordinate space (on the grid rr), the output array gg will be computed in the momentum space (on the grid kk). If d=-1, then the input array ff must be given in momentum space (on the grid kk), the output array gg will be computed in the coordinate space (on the grid rr).
4. Calculational subroutine is made thread-safe in OpenMP parallelized loops.
5. The example program is rewritten to demonstrate the usage of the package in OpenMP environments. The input data is now chosen randomly to generate a larger loop body. This enables the speed up, achieved by using OpenMP, to be more easily measured.
6. The package is written as a Fortran 90 module to allow a better control over the allocation/deallocation/thread-safety of internal variables.
7. Additional subroutines sbt_mesh(nr, rr, kk, rhomin, rhomax, kmax) and sbt_interp(nr, ff, r, f, rhomin, dr) are included to initialize the rr and kk grid points (see item 2 and 3 in this list) and to interpolate the output results at a point r on the radial mesh. The usage of the subroutines is demonstrated in the example program.

Running time: Run time is dominated by FFT transform (68%). The test calculation of overlap integrals runs 18 seconds with one thread, computing 360000 integrals on Intel Core2 Quad CPU Q9400 2.66 GHz using 256 radial grid points.

References:

© 2010 Elsevier B.V. All rights reserved.