LIBCOUL90

Calculate real–valued Coulomb and Bessel functions for real–valued arguments. The original `COUL90()`, available at the Fresco website, has been repackaged here as an fpm package. See the original author's article [1] for more detail. This is largely not my work. No intentional changes to the algorithm have been made. Test suite not yet implemented, but the tester program from the Fresco website is available.

Building with fpm

In the package directory, just run

```
$ fpm build --profile release
```

The archive file `libcoul90.a` and `.mod` files will be placed in the generated `build` subdirectory. These files are necessary to compile another program that uses this library.

Using this version of `COUL90()`

To use this project within your fpm project, add the following to your `fpm.toml` file:

```
[dependencies]
libcoul90 = { git="https://github.com/banana-bred/libcoul90" }
```

or

```
[dependencies]
libvoul90 = {'namespace'='COULXX'}
```

The module `libcoul90` contains the following public procedures :

- `coulf(lambda,eta,x)` : return the regular Coulomb function $F_\lambda(\eta, x)$
- `coulg(lambda,eta,x)` : return the irregular Coulomb function $G_\lambda(\eta, x)$
- `sphbessj(lambda,x)` : return the spherical Bessel function of the first kind $j_\lambda(x)$
- `sphbessy(lambda,x)` : return the spherical Bessel function of the second kind $y_\lambda(x)$
- `cylbessj(lambda,x)` : return the (cylindrical) Bessel function of the first kind $J_\lambda(x)$
- `cylbessy(lambda,x)` : return the (cylindrical) Bessel function of the second kind $Y_\lambda(x)$
- `coul90_wrapper(xlmin, nl, eta, x, f, fp, g, gp, kfn)` : a wrapper used to call `COUL90()`, used by the above–mentioned procedures
- `COUL90(X,ETA_IN,XLMIN,LRANGE,FC,GC,FCP,GCP,KFN,IFAIL)` : the original code described in [1], with some minor modernizations.
- `ricbes(x, lmax, psi, chi, psid, chid, ifail)` : a subroutine that returns the Riccati–Bessel functions $zj_\lambda(z)$ and $zy_\lambda(z)$, and their derivatives in the arrays `psi`, `chi`, `psid`, and `chid` for orders $0 - \lambda_{\max}$
- `sbesjy(x, lmax, j, y, jp, yp, ifail)` : a subroutine that returns the spherical Bessel functions functions $j_\lambda(z)$ and $y_\lambda(z)$, and their derivatives in the arrays `psi`, `chi`, `psid`, and `chid` for orders $0 - \lambda_{\max}$

Above, `LMAX` is the same as $\lambda_{\max}$ and `nl` is the number of λ values. The variables `x` and `eta` are `real(real64)`, while the variable `lambda` can be an `integer` or `real(real64)`. The kind `real64` (64 bits / 8 bytes) is defined in the intrinsic module `iso_fortran_env`. All public procedures other than `COUL90()` are superfluous —

they're provided for convenience or as a simple example of calling `COUL90()`.

The following example program

```
program test

  use libcoul90, only: coulf
  use iso_fortran_env, only: rp => real64, stdout => output_unit

  integer :: l = 0
  real(rp) :: eta = -0.5_rp
  real(rp) :: x = 20.0_rp

  write(stdout, '("F_", I0, "(", F0.2, ",", F0.2, ") = ", e0.15)') l, eta, x, coulf(

end program test
```

should print the following:

```
F_0(-.50,20.00) = -0.102372301807428
```

Testing with fpm

The tester program can be run with the following command

```
$ fpm test
```

This will read the file `test/COULTEST.in` and produce a file `test/COULTEST.out`, whose output can be compared to the reference file `test/COULTEST.REF`.

Reference(s)

[1] A. R. Barnett *The calculation of spherical Bessel and Coulomb functions*, Computational Atomic Physics: Electron and Positron Collisions with Atoms and Ions. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. 181–202. URL: https://link.springer.com/chapter/10.1007/978-3-642-61010-3_9