December 5, 2014
1:41

Contents

1 fmch

This code is for the oldest and most general and reliable of the methods of computing

$$F_\nu(x) = \int_0^1 t^{2\nu} \exp(-xt^2) dt$$

One of two possible series expansions is used depending on the value of x.

For $x \le 10$ (Small x Case) the (potentially) infinite series

$$F_\nu(x) = \frac{1}{2} \exp(-x) \sum_{i=0}^{\infty} \frac{\Gamma(\nu + \frac{1}{2})}{\Gamma(\nu + i + \frac{3}{2})} x^i$$

is used.

The series is truncated when the value of terms falls below 10^{-8}.

However, if the series seems to be becoming unreasonably long before this condition is reached (more than 50 terms), the evaluation is stopped and the function aborted with an error message on *ERROR_OUTPUT_UNIT*.

If $x > 10$ (Large x Case) a different series expansion is used:

$$F_\nu(x) = \frac{\Gamma(\nu + \frac{1}{2})}{2x^{\nu + \frac{1}{2})}} - \frac{1}{2} \exp(-x) \sum_{i=0}^{\infty} \frac{\Gamma(\nu + \frac{1}{2}}{\Gamma(\nu - i + \frac{3}{2})} x^{-i}$$

This series, in fact, diverges but it diverges so slowly that the error obtained in truncating it is always less than the last term in the truncated series. Thus, to obtain a value of the function to the same accuracy as the other series, the expansion is terminated when the last term is less than the same criterion (10^{-8}).

It can be shown that the minimum term is always for i close to $\nu + x$, thus if the terms for this value of i are not below the criterion, the series expansion is abandoned, a message output on *ERROR_OUTPUT_UNIT* and the function aborted.

The third argument, y, is $\exp(-x)$, since it is assumed that this function will only be used *once* to evaluate the function $F_\nu(x)$ for the maximum value of ν required and other values of $F_\nu(x)$ will be obtained by downward recursion:

$$F_{\nu-1}(x) = \frac{\exp(-x) + 2xF_\nu(x)}{2\nu - 1}$$

which also requires the value of $\exp(-x)$ to be available.

NAME

fmch

SYNOPSIS

```
double precision function fmch(nu,x,y);

implicit double precision (a-h,o-z);
double precision x, y;
integer nu;
```

DESCRIPTION

Computes

$$F_\nu(x) = \int_0^1 t^{2\nu} \exp(-xt^2) dt$$

given ν and x. It is used in the evaluation of GTF nuclear-attraction and electron-repulsion integrals.

ARGUMENTS:

nu Input: The value of ν in the explicit formula above (`integer`).

x Input: x in the formula (`double precision`).

y Input: $\exp(-x)$, assumed to be available.

DIAGNOSTICS

If the relevant series expansions used do not converge to a tolerance of 10^{-8}, an error message is printed on standard output and the computation aborted.

"fmch.f" 1 ≡

@m *ARB* 1
@m *YES* 0
@m *NO* 1
@m ERR −1

@m *BYTES_PER_INTEGER* 4
@m *LEAST_BYTE* 1

@m *END_OF_FILE* −1

@m *NO_OF_TYPES* 20
@m *INT_BLOCK_SIZE* 20

@m *LAST_BLOCK* 1
@m *NOT_LAST_BLOCK* 0

@m *ERROR_OUTPUT_UNIT* 6

@m *MAX_BASIS_FUNCTIONS* 255
@m *MAX_PRIMITIVES* 1000
@m *MAX_CENTRES* 50

@m *MAX_ITERATIONS* 60

@m *UHF_CALCULATION* 10
@m *CLOSED_SHELL_CALCULATION* 20

double precision function *fmch*(*nu*, *x*, *y*);

implicit double precision (*a* − *h*, *o* − *z*);
double precision *x*, *y*;
integer *nu*;

{

/∗ First, make the variable declarations; notice that comments, even in Fortran, are "C-style" ∗/

⟨ Internal Declarations 1.1 ⟩

/∗ Temporary integer and real values for $nu = \nu$ ∗/

m= *nu*;
a = ***dble***(***float***(*m*));

/∗ Now do the computation of $F_\nu(x)$ ∗/

if (*x* ≤ *ten*)

{

/∗ Use the expansion for the smaller value of *x* ∗/

⟨ Small x Case 1.2 ⟩

}

else

{

/∗ Use the other expansion for the larger value of *x* ∗/

⟨ Large x Case 1.3 ⟩

}

Here are the declarations and **data** statements which are entirely internal to *fmch*. Nomenclature is fairly obvious except possible *rootpi4* which is $\sqrt{(\pi/4)}$.

⟨ Internal Declarations 1.1 ⟩ ≡

```
double precision ten, half, one, zero, rootpi4, xd, crit;
double precision term, partialsum;
integer m, i, numberofterms, maxone, maxtwo;
data zero, half, one, rootpi4, ten/0.0, 0.5, 1.0, 0.88622692, 10.0/;
  /* crit is the required accuracy of the series expansion */
data crit/1.0 · 10^-08/;
  /* maxone is the upper limit of the summation in the small-x summation; maxtwo is the
    corresponding limit in the large-x summation */
data maxone/50/, maxtwo/200/;
```

This code is used in section 1.

This is the case of x less than 10 where the series is:

$$F_\nu(x) = \frac{1}{2}\exp(-x)\sum_{i=0}^{\infty}\frac{\Gamma(\nu+\frac{1}{2})}{\Gamma(\nu+i+\frac{3}{2})}x^i$$

The series is summed by using a recursion relationship between succesive terms in the series. The sum is accumulated in *partialsum* and the terms are obtained from previous ones by the ratio x/a with a incremented by 1 for each new *term*.

```
⟨ Small x Case 1.2 ⟩ ≡
    a = a + half;
    term = one / a;
      /* Remember that a is ν */
    partialsum = term;
    for (i = 2; i ≤ maxone; i = i + 1)
      {
      a = a + one;
      term = term * x / a;
      partialsum = partialsum + term;     /* Here is the possibility of convergence */
      if (term / partialsum < crit)
        break;
      }
      /* If the upper summation limit is reached, it means that the series has failed to converge, so say so
        and exit. */
    if (i ≡ maxone)
      {
      write (ERROR_OUTPUT_UNIT, 200) ;
    200:
      format ('␣i␣>␣50␣in␣fmch') ;
      STOP;
      }
      /* Otherwise return silently */
    return (half * partialsum * y);
```

This code is used in section 1.

This is the other case for x greater than ten:

$$F_\nu(x) = \frac{\Gamma(\nu+\frac{1}{2})}{2x^{\nu+\frac{1}{2})}} - \frac{1}{2}\exp(-x)\sum_{i=0}^{\infty}\frac{\Gamma(\nu+\frac{1}{2}}{\Gamma(\nu-i+\frac{3}{2})}x^{-i}$$

again, the terms in the sum are obtained by recursion from the previous ones.

⟨ Large x Case 1.3 ⟩ ≡

```
b = a + half;
a = a - half;

  /* Once again, remember that a is ν */
xd = one / x;
approx = rootpi4 * dsqrt(xd) * xd^m;
if (m > 0)
  {
  for (i = 1; i ≤ m; i = i + 1)
    {
    b = b - one;
    approx = approx * b;
    }
  }
fimult = half * y * xd;
partialsum = zero;

  /* Take care of special case */

if (fimult ≡ zero)
  return (approx);

  /* Otherwise continue */

fiprop = fimult / approx;
term = one;
partialsum = term;
numberofterms = maxtwo;
for (i = 2; i ≤ numberofterms; i = i + 1)
  {
  term = term * a * xd;
  partialsum = partialsum + term;

    /* See if the criterion is satisfied, return silently if so */

  if (dabs(term * fiprop / partialsum) ≤ crit)
    return (approx - fimult * partialsum);
  a = a - one;     /* or carry on */
  }

  /* If i gets as far as numberofterms then the expansion has failed, print a message and quit. */

  write (ERROR_OUTPUT_UNIT, 201) ;

201:
  format ('␣␣numberofterms␣reached␣in␣fmch') ;
  STOP;     /* no convergence */
  }
```

This code is used in section 1.

2 INDEX

⟨ Internal Declarations 1.1 ⟩ Used in section 1.
⟨ Large x Case 1.3 ⟩ Used in section 1.
⟨ Small x Case 1.2 ⟩ Used in section 1.

COMMAND LINE: `"fweave fmch.web"`.
WEB FILE: `"fmch.web"`.
CHANGE FILE: `(none)`.
GLOBAL LANGUAGE: RATFOR.