Feb 14, 2006.

This is PGM's internal documention for Vladimir Rokhlin's 3D FMM code for Laplace potential evaluations. The version described is the one transferred on February 10, 2006.

1. General function

Given a set of particle locations $(x_i)_{i=1}^N$, the code computes the potentials $(\phi(x_i))_{i=1}^N$ generated by a number of charges placed at the particle locations. To be precise, it computes, for $i = 1, \dots, N$,

$$u_i = \phi(x_i) = \sum_{j \neq i} \left(q_j \frac{1}{|x_i - x_j|} + d_j \frac{v_j \cdot (x_i - x_j)}{|x_i - x_j|^3} \right),$$

where q_i are monopole charge strengths, d_j are dipole charge strengths, and v_j are dipole direction vectors. The code also generates the field $F_i = -\nabla\phi(x_i)$. To be precise, it computes, for $i = 1, \dots, N$,

$$F_i = \sum_{j \neq i} \left(-q_j \frac{x_i - x_j}{|x_i - x_j|^3} + d_j \frac{v_j}{|x_i - x_j|^3} - d_j \frac{3(v_j \cdot (x_i - x_j))\,(x_i - x_j)}{|x_i - x_j|^5} \right).$$

It is possible to direct the code to use another interaction mechanism for particles that are very close. As an example, in a particle simulation, it is possible to use the harmonic potentials for long-range interactions, but switch to, say, Lennard-Jones, once particles get very close to each other. The close-range interaction is computed via a user-supplied subroutine "`close1`".

2. Syntax

The calling sequence for the subroutine is:

```
 CALL fmmpart3d(sources, nsources, monocharge, dipstr,
1                dipvec, pot, potx, poty, potz, work, maxwrk, iprec,
2                iflagtype, iderivonoff, close1, epsclose, ier, inform)
```

The input parameters are:

`sources(3,N)`	The particles locations, x_i,
`nsources`	The number of sources, N,
`monocharge(N)`	The monocharges, q_i,
`dipstr(N)`	Dipole strengths, d_i,
`dipvec(3,N)`	Dipole direction vectors, v_i,
`work(maxwrk)`	Work array,
`maxwrk`	Length of work array,
`iprec`	Requested precision, `iprec` = 0/1/2/3 for 3/6/9/12 digits' accuracy.
`iflagtype`	What type of charges are present, see Remark 1, `iflagtype` = 0/1/2 for monopoles only / dipoles only / both,
`iderivonoff`	Whether the fields should be computed, see Remark 1, `iderivonoff` = 0/1 for no / yes.
`close1`	Function for close-range interactions, see Remark 3
`epsclose`	The distance at which to switch to close-range, see Remark 3

The output parameters are:

`pot(N)`	The potential, ϕ_i,
`potx(N)`	The x-component of the field, $-\partial_x \phi$,
`poty(N)`	The y-component of the field, $-\partial_y \phi$,
`potz(N)`	The z-component of the field, $-\partial_z \phi$,
`ier`	Flag that reports any errors encountered, see Remark 2,
`inform`	Information about some algorithmic parameters, see Remark 2.

Remark 1. Input or output arrays that are not used need not be allocated in full: If monopoles are not present, then `monocharge` may be allocated as an array of length 1. If dipoles are not present, then `dipcharge` may be allocated as an array of length 1, and `dipvec` may be allocated as an array of length 3. If fields are not requested, then `potx`, `poty`, and `potz` may all be allocated as fields of length 1.

Remark 2. The parameter `ier` reports any errors encountered:

```
ier = 0  -> normal execution
ier = 1  -> normal execution, but epsclose restricts
             the amount of division, performance
             may be degraded
ier = 2  -> one or several input flags not properly
            defined, terminal error
ier = 3  -> not enough memory provided,
            terminal error
ier = 4  -> size of box generated for these boxes
            is of size zero.  major problem, terminal
ier = 5  -> invalid choice of epsclose, solution
            not attempted
ier = 6  -> invalid nsources value, solution
```

```
              not attempted
ier = 7  -> problem with the license file,
              this must be incorporated at the
              level of the subroutine riv3dlic
```

The paremeter `inform(5)` reports information about how the code executed the FMM algorithm:

```
inform(1) = number of refinement levels
inform(2) = total number of boxes in the quadtree
inform(3) = order of mpole and local expansions
inform(4) = order of plane wave expansions
inform(5) = total amount of workspace used
```

Remark 3. It is possible to request that the code switches the way it computes interactions when particles get close. The distance at which the switch occurs is specified via the parameter `epsclose`, and the interaction mechanism that it switches to is specified in the function `close1`. This function must be declared as `external` in the calling program, and must have the following calling sequence given in Fig. 2.1.

3. Test files

Test 1: – `test1.f` – This file sets up a simple system containing only three particles. The potential and fields are calculated using `fmmpart3d` and via direct computation. The results are compared. Note that for such a small problem as this, the actual fast multipole algorithm is never enganged, `fmmpard3d` itself simply reverts to using direct calculations.

Test 2: – `test2.f` – This file scatters N particles at random in the box $[0, 1]^3$, and assigns each charge with random mono and dipole charges. It evaluates the fields using `fmmpart3d` and then compares with the result of a direct computation. The fast mutlipole algorithm engages at around 1500 particles. If there are more than 5000 particles present, the error checking is done for a single (randomly selected) particle only.

```
      subroutine close1(eps, xinfo1, xinfo2, iflag, xout)
      implicit real *8(a-h,o-z)
c
c     INPUT PARAMETERS:
c
c     eps (real *8) the distance that determines when this
c              subroutine needs to be called
c
c     xinfo1(8) (real *8) vector containing the information
c              for the first (incoming) point in the interaction
c           xinfo1(1) x position of incoming point
c           xinfo1(2) y position of incoming point
c           xinfo1(3) z position of incoming point
c           xinfo1(4) the monopole charge at the incoming point
c           xinfo1(5) the dipole charge at the incoming point
c           xinfo1(6) x part of dipole vector at incoming point
c           xinfo1(7) y part of dipole vector at incoming point
c           xinfo1(8) z part of dipole vector at incoming point
c
c     xinfo2(8) (real *8) vector containing the information
c              for the second (outgoing) point in the interaction
c           xinfo2(1) x position of outgoing point
c           xinfo2(2) y position of outgoing point
c           xinfo2(3) z position of outgoing point
c           xinfo2(4) the monopole charge at the outgoing point
c           xinfo2(5) the dipole charge at the outgoing point
c           xinfo2(6) x part of dipole vector at outgoing point
c           xinfo2(7) y part of dipole vector at outgoing point
c           xinfo2(8) z part of dipole vector at outgoing point
c
c     iflag(2) (integer *4) the array of all flags that must be
c           specified when calling the fmmpart subroutine
c            iflag(1) is iflagtype
c              (indicates whether monopoles, dipoles,
c              or both are present)
c            iflag(2) is iderivonoff
c              (indicates whether or not to compute the
c              derivatives.  1 yes, 0 no)
c
c     OUTPUT PARAMETERS:
c
c     xout(4) (real *8) array of output computed using the
c            techniques specified in the subroutine
c
c           xout(1) will be added to the potential term
c           xout(2) will be added to the x derivative
c           xout(3) will be added to the y derivative
c           xout(4) will be added to the z derivative
c
c     all of these output terms are added to the array positions
c     for particles corresponding to the data in xinfo1
c
```

FIGURE 2.1. Calling sequence for the “close1” subroutine.